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

Géraldine Olivier

THESE DE DOCTORAT DE L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité: MATHEMATIQUES APPLIQUEES

Ecole Doctorale de Mathématiques de Paris Centre - ED 386

Présentée par

Géraldine OLIVIER

Pour obtenir le grade de

Docteur de l'Université Pierre et Marie Curie

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

soutenue le Vendredi 22 AVRIL 2011

devant le jury composé de:

Rapporteurs :	Prof.	Thierry COUPEZ	-	Mines-ParisTech - CEMEF
	Prof.	Boniface NKONGA	-	Université de Nice - UMR CNRS 6621
Directeur de thèse :	M.	Frédéric ALAUZET	-	INRIA Rocquencourt - GAMMA3
Examinateurs :	Prof.	Olivier PIRONNEAU	-	UPMC Paris VI - LJLL
	M.	Paul-Louis GEORGE	-	INRIA Rocquencourt - GAMMA3
	Prof.	Oubay HASSAN	-	Swansea University
	Prof.	Alain DERVIEUX	-	INRIA Sophia-Antipolis - TROPICS
	M.	Olivier ALLAIN	-	Société LEMMA

Contents

I Anisotropic metric-based mesh adaptation for unsteady simulations 11

1 Multi-scale metric-based mesh adaptation 15
- 1.1 State of the art of metric-based mesh adaptation . . . 16
  - 1.1.1 A short history . . . 16
  - 1.1.2 Current impact in scientific computing. . . . 17
- 1.2 Basics of metric-based mesh adaptation . . . 17
  - 1.2.1 Riemannian geometry and unit mesh . . . 18
  - 1.2.2 Metric-based mesh adaptation . . . 25
  - 1.2.3 Interpolation issues . . . 26
  - 1.2.4 Gradient and Hessian recovery techniques . . . 31
  - 1.2.5 Operations on metrics . . . 34
  - 1.2.6 Metrics intersection . . . 34
  - 1.2.7 Metric interpolation . . . 35
- 1.3 The continuous mesh theory . . . 37
  - 1.3.1 The continuous element model . . . 37
  - 1.3.2 The continuous mesh model . . . 39
  - 1.3.3 The continuous linear interpolate model . . . 40
- 1.4 Multi-scale mesh adaptation . . . 44
  - 1.4.1 Global optimization problem . . . 44
  - 1.4.2 Optimal metric for a numerical approximation u_h . . . 46
  - 1.4.3 Adaptive strategy using the optimal $\mathbf{L}^p$ metric . . . 46
  - 1.4.4 Handling degenerated cases . . . 49

2 Unsteady mesh adaptation: theory and practice 59
- 2.1 The challenges of unsteady simulations . . . 60
  - 2.1.1 Motivations . . . 60
  - 2.1.2 Problematics linked to unsteadiness. . . . 60
  - 2.1.3 Problematics linked to unsteady anisotropic mesh adaptation. . . . 64
  - 2.1.4 State of the art . . . 65
  - 2.1.5 Our approach . . . 73
- 2.2 Space-time error analysis and CFL condition . . . 74
  - 2.2.1 Truncation error analysis in 1D . . . 74
  - 2.2.2 Illustration of the influence of the CFL number on the space-time approximation error . . . 78
  - 2.2.3 Multi-dimensional theoretical analysis of the interpolation error . . . 80
- 2.3 Extension of multi-scale mesh adaptation to unsteady simulations. . . . 86
  - 2.3.1 Controlling the error during a whole adaptation sub-interval with a single adapted mesh in a multi-scale framework. . . . 87
  - 2.3.2 Local space-time mesh complexity and global fixed-point strategy. . . . 89

2.3.3 Practical use of the optimal space-time metric inside the fixed-point algorithm. . . . 93
2.4 Numerical illustrations and comparison with the former algorithm . . . 99
2.4.1 Three-dimensional Double-Mach reflection simulation . . . 100
2.4.2 Three-dimensional blast in a city . . . 102

3 Metric-based adaptation for moving mesh simulations 107
3.1 The ALE metric . . . 108
3.2 Analytical examples . . . 111
3.2.1 Tests cases description . . . 111
3.2.2 Results analysis. . . . 115
3.3 Extension of the fixed-point algorithm to moving mesh simulations . . . 117

II Extension of anisotropic metric-based mesh adaptation to Arbitrary-Lagrangian-Eulerian simulations 139

4 Moving mesh strategies 143
4.1 Inner vertices movement prescription . . . 145
4.1.1 Internal movement equation . . . 145
4.1.2 Elasticity-dedicated mesh strategy . . . 146
4.1.3 Local material properties . . . 147
4.2 Mesh optimization . . . 148
4.2.1 Laplacian smoothing . . . 149
4.2.2 Edge/face swapping . . . 149
4.3 Moving mesh time steps . . . 151
4.3.1 Elasticity time step . . . 151
4.3.2 Geometric time step . . . 152
4.4 Numerical illustrations . . . 152
4.4.1 Two dimensional tests cases . . . 152
4.4.2 Three dimensional tests cases . . . 155

5 The Arbitrary-Lagrangian-Eulerian solver 167
5.1 Modelization of the physical problem . . . 168
5.1.1 The Arbitrary-Lagrangian-Eulerian framework . . . 168
5.1.2 The Euler equations in the Arbitrary-Lagrangian-Eulerian formulation . . . 169
5.2 Spatial discretization of the ALE Euler equations . . . 170
5.2.1 Finite-Volume edge-based formulation . . . 171
5.2.2 High-order schemes . . . 174
5.2.3 Mirror state boundary conditions . . . 176
5.3 ALE specific issues regarding time discretization . . . 178
5.3.1 The GCL law . . . 178
5.3.2 Accuracy preserving and DGCL temporal schemes . . . 178
5.3.3 Runge-Kutta Strong-Stability-Preserving schemes . . . 181
5.3.4 DGCL RKSSP schemes for moving mesh simulations. . . . 182

5.3.5 Practical computation of the areas swept . . . 187
5.4 A new changing-topology ALE formulation . . . 189
5.4.1 Problematics . . . 189
5.4.2 Our approach . . . 190
5.4.3 Swapping evanescent cell and volume redistribution . . . 190
5.4.4 Notations . . . 192
5.4.5 Swept areas computations . . . 194
5.4.6 The changing-topology ALE schemes . . . 200
5.4.7 Generalizations . . . 204
5.4.8 Scheme analysis . . . 208
5.5 Fluid Structure Interactions issues . . . 212
5.5.1 Description of the moving objects . . . 212
5.5.2 Movement of the geometries . . . 214
5.5.3 Resolution of the rigid bodies equations and loose coupling . . . 217
5.6 Numerical results . . . 221
5.6.1 Turbo-machinery . . . 221
5.6.2 Pitching NACA0012 airfoil . . . 223
5.6.3 Blast test case . . . 224

A Temporal Hessian computation **233**
A.1 Scheme to compute temporal Hessians . . . 233
A.2 Numerical illustration . . . 234

B The elasticity system resolution **237**
B.1 Problem formulation . . . 237
B.2 Finite-Element discretization . . . 238

C Strong-Stability-Preserving Runge-Kutta schemes **247**
C.1 Runge-Kutta schemes . . . 247
C.1.1 General formulation . . . 247
C.1.2 Explicit and implicit schemes . . . 248
C.1.3 Shu-Osher representation . . . 248
C.2 Order of accuracy . . . 249
C.3 Linear stability results . . . 249
C.4 Non-linear stability results . . . 250
C.5 SSP optimal schemes (i.e maximal CFL schemes) . . . 250
C.5.1 Linear case with $n_s = p$ [Gottlieb 2001] . . . 250
C.5.2 Non-linear case with $n_s = p$ . . . 251
C.5.3 Non-linear case with $n_s \geq p$ . . . 252
C.6 Implicit Backward Difference Formulae (BDF) methods . . . 256
C.6.1 First-order accurate implicit Backward Difference Formula . . . 256
C.6.2 Second-order accurate implicit Backward Difference Formula . . . 257
C.6.3 Higher-order accurate implicit Backward Difference Formula . . . 257

D ALE mirror state slipping boundary conditions **259**
D.1 Preliminary: Jacobian matrix in three dimensions 259
D.2 ALE mirror state description 261
D.3 Roe ALE mirror state slipping boundary condition 263
D.3.1 Upwinding term computation 264
D.4 HLLC ALE mirror state boundary condition 265
D.5 ALE mirror slipping boundary conditions coupled with DGCL Runge-Kutta schemes 268

Bibliography **271**

List of notations **283**

Acronyms **287**

Résumé

Cette thèse s'intéresse aux simulations dépendantes du temps impliquant des géometries fixes ou mobiles. Ce type de simulations est l'objet d'attentes grandissantes de la part des industriels, qui souhaiteraient voir réaliser ce type de calculs de façon systématique au sein de leurs centres de recherche, ce qui n'est clairement pas le cas à l'heure actuelle. Ce travail tente de satisfaire en partie cette demande et vise notamment à améliorer la précision ainsi que l'efficacité en termes de temps de calcul des algorithmes actuellement utilisés dans ce contexte.
Les méthodes d'adaptation de maillage anisotrope par prescription d'un champ de métriques, qui ont aujourd'hui atteint une certaine maturité, notamment dans leur application aux simulations stationnaires, constituent une piste très prometteuse pour l'amélioration des calculs évoluant en temps, mais leur extension dans ce contexte est loin d'être triviale. Quant à leur utilisation sur les simulations en géométries mobiles, seules quelques tentatives peuvent être répertoriées, et très peu portent sur des problèmes réalistes en trois dimensions.
Cette étude présente plusieurs nouveautés sur ces questions, notamment l'extension de l'adaptation de maillage multi-échelles par champ de métriques aux problèmes instationnaires en géométries fixes et mobiles. Par ailleurs, essentiellement dans une optique de réduction des temps de calculs, une stratégie originale à été adoptée pour réaliser des calculs impliquant des maillages mobiles. Notamment, il est démontré par la pratique dans cette thèse qu'il possible de déplacer des objets en trois dimensions sur de grandes distances en maintenant le nombre de sommets du maillage constant, c'est-à-dire en limitant les types d'opérations de modification de maillage autorisés. Il en résulte un gain conséquent en terme de temps de calcul aussi bien au niveau du déplacement de maillage qu'au niveau de la résolution numérique. Par ailleurs, un nouveau schéma est proposé qui permet de gérer les changements de connectivité du maillage de manière cohérente avec la description Arbitrary-Lagrangian-Eulerian des équations physiques.
La plupart de ces nouvelles méthodes ont été appliquées à la simulation d'écoulements fluides compressibles autour de géometries complexes en deux et trois dimensions d'espace.

Conventions

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, respectively.
Elements of vectorial space $\mathbb{R}^n$ are noted in bold font. The coordinates 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 triangle in two dimensions, no flat tetrahedron in three dimensions),
- *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, and edge or a face (in three dimensions) or is empty.

The *conformity* hypothesis, which is not always required (notably for Discontinuous Galerkin methods), is nevertheless adopted here for two main reasons. On the one hand, the meshes will be used in relation with a Finite Volume solver which needs the enforcement of this constraint. On the other hand, this effort on the meshing side facilitates the handling of data structures on the solver side and also enables to save a consequent amount of CPU time, notably when computing the neighbors of a given element, the ball of elements around a vertex or the shell of an edge.
Besides, the use of structured meshes for boundary layers which is often recommended for the accuracy of turbulent flow computations, is not required here as only non-viscous compressible Computational Fluid Dynamics simulations are considered.
Finally, a mesh is said to be *uniform* if all its elements are almost regular (equilateral) and have the same typical size h.

Notations and orientation conventions

The following notations will be used in this thesis: K is a mesh element, 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 long 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 $\boldsymbol{\eta}$. Inward normals will noted $\overline{\mathbf{n}}$ if normalized

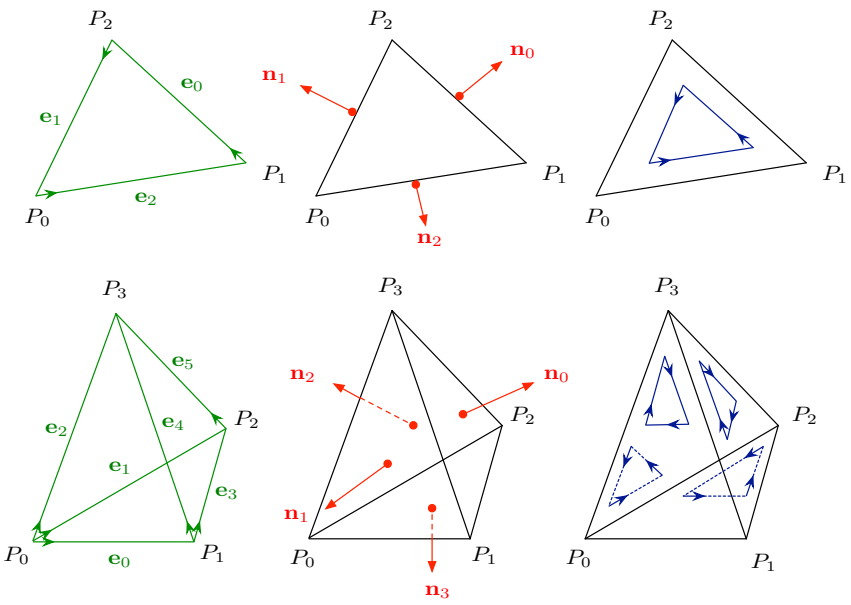


Figure 1: Conventions in a simplicial element K in two (top) and three (bottom) dimensions. Conventions for vertex numbering, edge numbering and orientation (left), unit normal numbering and orientation (middle) and face(s) orientation (right).

and $\overline{\boldsymbol{\eta}}$ if not.

Topological structures

Neighbors of an element. The set of the neighbors of element K, noted $\mathrm{Neigh}(K)$, is defined as the set of elements having an edge in two dimensions or a face in three dimensions in common with element K. An element always has 3 neighbors in two dimensions and 4 neighbors in three dimensions, see Figure 2.

Ball of a vertex. The ball of vertex P_i, noted $\mathrm{Ball}(P_i)$, is defined as the set of all the elements having P_i as a vertex. The number of elements in the ball of a vertex can be arbitrary large and generally varies for each vertex. An example is given in Figure 3. Besides, the connectivity of vertex P_i is defined as the number of vertices connected to P_i by an edge. In two dimensions, the average connectivity is 6 whereas it is 17 in three dimensions.

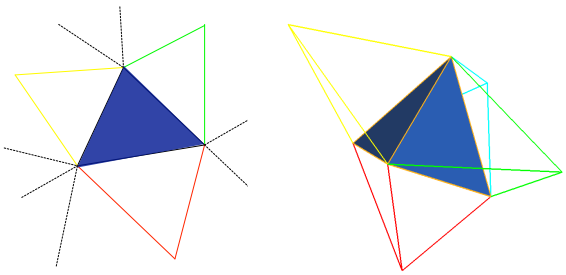


Figure 2: Neighbors of an element in two (left) and three (right) dimensions.

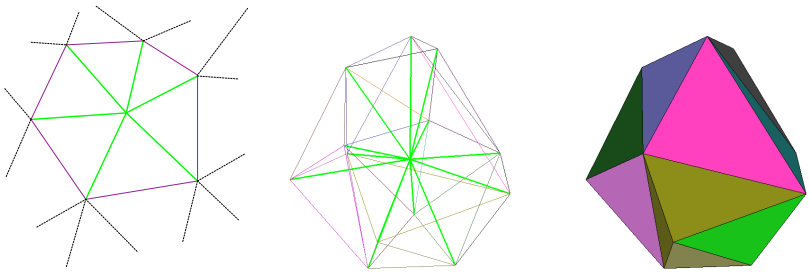


Figure 3: Ball of a vertex in two (left) and three dimensions (middle and right).

Shell of an edge. The shell of edge **e**, noted Shell(**e**), is the set of all elements having **e** as edge. In two dimensions, the shell of an edge always contains 2 triangles, except for boundary edges, whereas in three dimensions, this number varies for each edge and equals on average 5.7.

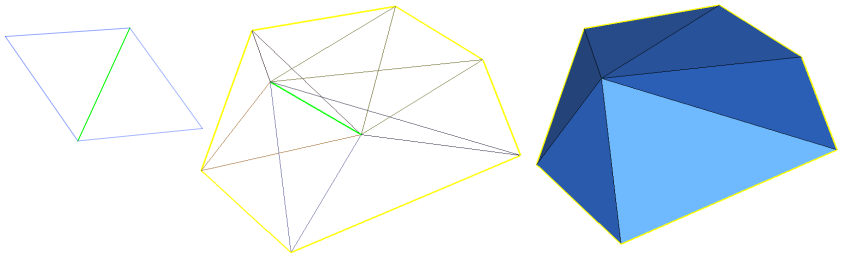


Figure 4: Shell of an edge in two (left) and three (right) dimensions.

INTRODUCTION

This thesis deals with Computational Fluid Dynamics (CFD) computations and more specifically with the challenging issue of *unsteady* simulations involving either fixed or time-evolving computational domains.
This type of simulations gets an ever-increasing interest from the industrial world, mainly because, for most of the problems addressed by industrials, unsteady features cannot be neglected as they have a strong influence on the quantities of interest (impulse, pressure, lift, drag...). Turbo-machinery, blast prediction, aeronautics, biomechanics are some of these fields which are very much in demand for new effective and trustworthy tools to handle unsteady numerical simulations.

Problematics

Unsteady simulations raises several problematics. On the one end, the certification of numerical computations in terms of solution **accuracy** is harder than for pseudo-simulations. This is a serious problem as trustworthy computations constitute an absolute necessity if industrial integration is expected. The **efficiency** of the computations in terms of CPU time also represents a great challenge, because unsteady simulations are generally much slower than pseudo-steady one. The main reason to this is that the stability and accuracy of the numerical schemes requires the enforcement of additional constraints regarding the time step size. Finally, when time-evolving domain computations are considered, new **moving mesh issues** appear. These issues are especially visible in the case of three-dimensional simulations involving complex geometries, *i.e.* geometries with complex and sharp boundaries.

State of the art

In the past twenty years, mesh adaptation has demonstrated its ability to improve the **accuracy** of numerical simulations while mastering the computational effort (CPU time). The idea of mesh adaptation has been applied under various form (refinement/coarsening, isotropic or anisotropic adaptation...) on many different problems ranging from aerodynamics to biomechanics and magnetodynamics. Among all these methods, metric-based mesh adaptation, thanks to its flexibility and automaticity, has enabled considerable gain in both accuracy and CPU time for various **pseudo-steady** problems [Hecht 1997, Bottasso 2004, Gruau 2005, Loseille 2010c, Guégan 2010].
A promising attempt to extend these methods to unsteady, fixed-mesh problems has been proposed in [Alauzet 2003a]. However, this strategy considers only the case where the global interpolation error is controlled in $\mathbf{L}^\infty$-norm. Moreover, it relies on several hypothesis regarding the local space-time error. Eventually, and to our knowledge, the problem of metric-based adaptation in the context of moving geometries has never been addressed in the details.
Regarding the success encountered by metric-based mesh adaptation methods for pseudo-steady

problems, it seems sensible to pursue the efforts towards the extension of these methods to unsteady problems, potentially involving moving computational domains.

Objective

This thesis therefore aims at **coupling anisotropic *metric-based* mesh adaptation and unsteady, possibly moving mesh simulations**. The objective is twofolds.
First, the unsteady strategy introduced in [Alauzet 2003a], which has already given promising results, is taken as a starting point. The goal is to enhance this method, notably to refine the error analysis and then to extend this method on the one hand to the multi-scale framework and on the other hand to moving mesh simulations.
Second, we intend to develop an efficient and accuracy-preserving solver to handle time-evolving domain computations. This means that an efficient strategy has to be designed to move the mesh, which is also compliant with highly anisotropic metric-based mesh adaptation. Efficient and accurate numerical schemes to solve these equations in this new moving mesh framework are also expected.

Context and numerical choices

In this work, only **simplicial unstructured meshes** are considered, *i.e.* triangular or tetrahedral meshes. The choice is justified by several facts. First, the existence of several fully-automatic unstructured three-dimensional meshing softwares, see Introduction I, is clearly advantageous. Next, it is much easier to mesh complex geometries using simplicial elements than with any other types of elements. Additionally, the use of simplicial elements brings the flexibility required for the introduction of the anisotropic mesh adaptation. And last, the existence of a mesh suited to a given boundary discretization can be proved when unstructured meshes are considered, whereas this existence is not guaranteed with structured meshes.

Only **metric-based** mesh adaptation techniques have been studied. It presents several advantages over other adaptation techniques, such as local refinement/coarsening approaches. First, metric-based mesh adaptation is appropriate to handle **anisotropic** meshes. This is a good asset as anisotropic mesh adaptation enables to optimize the mesh not only by improving node repartition, but also by optimizing the orientation of the elements according to the physical phenomena of the flow. Second, the underlying theory is now mature, in particular due to the recent development of the "continuous mesh" framework [Loseille 2010a, Loseille 2010b]. This framework enables to manipulate quantities related to meshes (vertices, number of vertices, orientation of elements, interpolation error) using a well-defined object (a Riemannian metric space). Meshes can then be treated like any other analytical quantities. Optimality problems involving meshes then become tractable. One of the main theoretical breakthrough regards. Third, the ability of metric-based mesh adaptation to handle complex three-dimensional pseudo-steady simulations has already been demonstrated by several research teams. These arguments justify the choice of anisotropic metric-based adaptation in this thesis.
Time-evolving computational domains are treated with a **body-fitted approach**, which means that the computational mesh follows time-evolving geometries in their movement. This ap-

proach has been preferred to Immersed Boundaries and Chimera methods. Immersed/Embedded Boundaries techniques consist in "embedding" the moving geometries inside a fixed-mesh using, for instance, level-set methods. They generally result in a loss of accuracy and have therefore not be retained for this work. The Chimera method associates to each moving geometry a dedicated sub-grid which follows its associated moving body in a fully rigid manner. These sub-grids overlap and specific projection methods are used to recover the solution at sub-grids interfaces. However, this projection phase spoils the computation accuracy, especially if sub-grids are of different precision, which is actually the typical situation that will be encountered if anisotropic mesh adaptation is envisaged. Therefore, the body-fitted approach, which clearly privileges accuracy, has been chosen. On the solver side, moving mesh computations are handled using an **ALE** formulation of the **Euler** equations.
Most application examples have been chosen in the field of Computational Fluid Dynamics, and only inviscid compressible mono-fluid problems described by the Euler equations are considered.

Outline

This work is made of two parts.

Part I focuses on metric-based multi-scale mesh adaptation issues. This metric-based adaptation approach enables the control of the *global* interpolation error in different $\mathbf{L}^p$-norms, and hence gives the user the opportunity to choose the features of the solution he wants to capture. After recalling the basics of metric-based mesh adaptation, the extension of mesh adaptation algorithms in the context of unsteady simulations is addressed and new unsteady error estimators driving the adaptation process are introduced. Part I ends with the generalization of metric-based mesh adaptation to moving mesh Arbitrary-Lagrangian-Eulerian (ALE) simulations.

Part II concentrates on moving mesh computations issues, but still with the final aim of coupling them with mesh adaptation. It first addresses dynamic mesh issues and explains in the details the techniques employed to move the mesh. An original framework has been retained in the perspective of mesh adaptation. By limiting the type of authorized meshing operations, the strategy is at the balance between local and global re-meshing methods. This study ends with the description of the Arbitrary-Lagrangian-Eulerian resolution of the equations which enables to take the movement of the mesh into account in the fluid dynamics equations. In particular, a new changing-topology ALE scheme is presented, which allows to relax the fixed-topology constraint imposed in classical ALE formulations while remaining fully ALE in the philosophy. Thanks to these efforts both on the meshing and on the solver side, simulations coupling metric-based multi-scale mesh adaptation have been successfully computed and are presented at the end of this work.

Part I

Anisotropic metric-based mesh adaptation for unsteady simulations

Introduction

The prevalence of unsteady problems in the industrial world, but also in nature in general, is an obvious fact. On the contrary, stationary problems must be seen as scarce exceptions, especially in CFD as the stationary feature of a flow most of the time results from several simplifications.

However, despite numerous studies on the subject, notably regarding turbulence and blast issues, unsteady flow problems still represent a great challenge. A basic but fundamental calculus accounts for this difficulty to address unsteady CFD problems, especially when convergence studies are considered. In three dimensions, dividing the spatial scale h by 2 multiplies the number of tetrahedra by 8. Moreover, due to the CFL condition, the time step is also approximately divided by 2, which means that the number of iterations is multiplied by 2. Therefore, a rough calculus shows that the CPU time is multiplied by $8 \times 2 = 16$, assuming the resolution algorithm speed is linear in the number of tetrahedra and in the number of iterations. In the same manner, if h is divided by 4, the number of tetrahedra is multiplied by 64 and the number of iterations by 4, meaning that the CPU time is multiplied by 256. For example, if the simulations of accuracies h and $h/4$ respectively are launched simultaneously on January 1^{st} and if the simulation of accuracy h lasts one day, then the simulation with accuracy $h/4$ is not likely to end before September 13^{th}!

This trivial observation shows that, without the help of new efficient tools, industrials willing to simulate unsteady phenomena will be forced to choose between accuracy and efficiency, which is certainly not a satisfactory alternative. Besides, if uniformly refined meshes are used, convergence studies, which represent a real problematic as they constitute a first step toward the certification of numerical simulations, will remain beyond reach for many decades.

Since the pioneering work of [Vallet 1992, Castro-Díaz 1997, Leservoisier 2001] and of [Frey 2005] for three-dimensional problems, metric-based mesh adaptation has proved its ability to improve at the same time the accuracy and the efficiency of steady numerical simulations. More recently, the continuous mesh theory and its application to multi-scale mesh adaptation [Loseille 2010c, Loseille 2010a] has enabled new breakthroughs. Indeed, by clarifying the theoretical framework which underlies multi-scale metric-based mesh adaptation, it has allowed the prescription of bigger element altitudes in shocks by controlling the error in $\mathbf{L^p}$ norm, thus preventing the generation of too flat elements. This well-defined framework has also been used to design new metrics which simultaneously take into account phenomena of different orders of magnitude, hence the name "multi-scale" mesh adaptation.

Thanks to these technical and theoretical advances, simulations involving decades of millions of tetrahedra are now attainable at a relatively low cost and in a reasonable amount of time. As way of an example, at INRIA-GAMMA, steady adaptive CFD simulations around supersonic aircrafts on a mesh having about 60 millions of tetrahedra are common practice [Alauzet 2010b]. This kind of simulations currently takes 3-4 days on a 8-Core Apple Xserve with two Intel Xeon quad cores at 2,26 GHz supplied with 24 Go of RAM memory. Considering these great successes, the extension of multi-scale metric-based mesh adaptation strategies to unsteady simulations appears of the utmost importance to cope with the dramatic increase in CPU time required for unsteady problems.

Several attempts to perform unsteady mesh adaptations have been made so far, with various success and theoretical backgrounds. This part of the thesis aims at improving the theoretical understanding of unsteady mesh adaptation and at giving numerical proofs of its efficiency, notably thanks to convergence studies on complex geometries. The main objective here is to demonstrate that multi-scale metric-based mesh adaptation enables accurate **and** efficient unsteady simulations, along with full convergence studies, which, according to the above sketchy estimation, seems utterly unreachable if uniform meshes are considered.

This part is made of three chapters. The first one sums up the main concepts and results related to steady metric-based mesh adaptation, including the basics of Riemannian geometry and metric-based mesh adaptation, the continuous mesh and multi-scale theories and some details about the algorithmics.
Chapter 2 deals with unsteady metric-based mesh adaptation. The original fixed-point approach initiated in [Alauzet 2003a] is used as a starting point to introduce a new theory and a new algorithm which generalize the continuous mesh theory and multi-scale mesh adaptation to space-time problems. Some perspectives and way of improvements are also described.
Eventually, the case of metric-based mesh adaptation for moving geometry problems is addressed in Chapter 3. Two- and three-dimensional analytical examples illustrate the impact of the movement of the mesh on the metric and the modifications that must be done in the adaptation algorithm to take the mesh movement into account.

Chapter 1

Multi-scale metric-based mesh adaptation for steady problems: theory and practice

Contents

1.1 State of the art of metric-based mesh adaptation 16

1.1.1 A short history 16

1.1.2 Current impact in scientific computing. 17

1.2 Basics of metric-based mesh adaptation 17

1.2.1 Riemannian geometry and unit mesh 18

1.2.2 Metric-based mesh adaptation 25

1.2.3 Interpolation issues 26

1.2.4 Gradient and Hessian recovery techniques 31

1.2.5 Operations on metrics 34

1.2.6 Metrics intersection 34

1.2.7 Metric interpolation 35

1.3 The continuous mesh theory 37

1.3.1 The continuous element model 37

1.3.2 The continuous mesh model 39

1.3.3 The continuous linear interpolate model 40

1.4 Multi-scale mesh adaptation 44

1.4.1 Global optimization problem 44

1.4.2 Optimal metric for a numerical approximation u_h 46

1.4.3 Adaptive strategy using the optimal $\mathbf{L}^p$ metric 46

1.4.4 Handling degenerated cases 49

1.1 State of the art of metric-based mesh adaptation

1.1.1 A short history

The idea of adapting the mesh associated with a numerical solution dates back to the early development of numerical simulation. Since the 1960's, a rather large number of papers have been published on the subject. For instance, the query "Mesh Adaptation" on Google Scholar exhibits 395 000 results! In most of these works, the adaptation is isotropic and done by successive refinements of the elements according to predefined patterns: a square is split into four squares, a triangle is split into four triangles...

The powerful idea of anisotropic mesh adaptation has emerged later at the end of the 80's due to error estimate and mesh generation concerns. In 1987, Peraire et al. [Peraire 1987] proposed a first attempt in two dimensions by providing error measures involving directions. They pointed out the directional properties of the interpolation error and initiated the idea of generating elements with aspect ratios. They considered a local mapping procedure to generate elongated elements. They coupled this with an advancing front technique to generate slightly anisotropic meshes, *i.e.* elements having a 1:5 ratio. Similar approaches have been considered in [Löhner 1989] and [Selmin 1992]. The first attempts in three dimensions were proposed in the early 1990's in [Löhner 1990] and [Peraire 1992], but numerical results were almost isotropic and the mesh anisotropy was not clearly visible. In 1994, Zienkiewicz [Zienkiewicz 1994] gave a qualified status on the subject. Despite some great successes with this new approach, they emphasized that: *"Unfortunately the amount of elongation which can be used in a typical mesh generation by such mapping is small..."* . Almost at the same time on the meshing side, Mavriplis [Mavriplis 1990] suggested to generate stretched elements using a Delaunay approach in two dimensions in order to obtain high-aspect ratio triangles in boundary layers and wake regions required by aeronautic numerical simulations. According to him, the Delaunay triangulation had to be performed in a locally stretched space: the idea of metric was almost there. The year after, George, Hecht and Vallet [George 1991] introduced the use of metrics in a Delaunay mesh generator. They noticed that the absolute value of the Hessian of a given scalar solution could be viewed as a metric and proposed a Delaunay-based mesh generator where edge lengths would be computed in the Riemannian metric space associated with this metric. This work enabled to encompass all the previous attempts.

The fruitful idea of metric was widely exploited for two-dimensional anisotropic mesh adaptation in the 90's and even more today (see, among many others, [Fortin 1996, Castro-Díaz 1997, Hecht 1997, Dompierre 1997, Buscaglia 1997]). In 1997, Baker [Baker 1997] gave a state-of-art and wrote: *"Mesh generation in three dimensions is a difficult enough task in the absence of mesh adaptation and it is only recently that satisfactory three-dimensional mesh generators have become available [...]. Mesh alteration in three dimensions is therefore a rather perilous procedure that should be undertaken with care"*. Indeed, three-dimensional meshing is much more complicated as new pathologies occur. The existence of a three-dimensional mesh enforcing a given discretized surface is not even guaranteed. Doing three-dimensional *anisotropic* mesh adaptation is even more complicated. These bottlenecks have been partly solved by the development

of local re-meshing techniques, which try to adapt the mesh by performing local modifications (insertion/deletion of vertices, vertices displacements, connectivity changes). One great asset of these techniques is to intrinsically get rid of the previous existence problem. At the beginning of the 2000's, the first three-dimensional results showing "truly" anisotropic adapted meshes were published [Tam 2000, Pain 2001, Bottasso 2004, Belhamadia 2004, Frey 2005, Gruau 2005, Li 2005]. In the meantime, new more accurate anisotropic error estimates have been proposed: a posteriori estimates [Picasso 2003, Formaggia 2004a], a priori estimates [Formaggia 2001, Alauzet 2006, Huang 2005] and goal-oriented estimates for functional outputs [Venditti 2003, Jones 2006, Loseille 2010c].

1.1.2 Current impact in scientific computing.

Thanks to its **generality** and **modularity**, metric-based mesh adaptation has been used various research fields: as way of an example, it has been applied successfully in three-dimensions to the sonic boom simulation [Alauzet 2010b], multi-fluid flows [Compère 2007, Guégan 2010], blast problems [Alauzet 2007], Stefan problems [Belhamadia 2004] and metal forming processes [Bruchon 2009]. It has also been used with various numerical methods, among which the Finite Volume [Alauzet 2010b], Finite Element [Allain 2009], Stabilized Finite Element [Bruchon 2009] and Discontinuous Galerkin Finite Element [Remacle 2005] methods. In all these cases, large improvements in terms of accuracy and CPU performances have been established. There are currently many adaptive softwares using Riemannian metrics. Let us cite `BAMG` [Hecht 1998] and `BL2D` [Laug 2003] in two dimensions, `YAMS` [Frey 2001] for discrete surface mesh adaptation and `Feflo.a` [Loseille 2009], `Forge3d` [Coupez 2000], `FUN3D` [Jones 2006], `GAMMANIC3D` [George 2003], `MadLib` [Compere 2010], `MeshAdap` [Li 2005], `MMG3D` [Dobrzynski 2008], `MOM3D` [Tam 2000], `TANGO` [Bottasso 2004] and `LibAdaptivity` [Pain 2001] in three-dimensions. It is worth mentioning that all these softwares have arisen from different mesh generation methods. The method used in [George 2003, Hecht 1998] is based on a global constrained Delaunay kernel. In [Laug 2003], the Delaunay method and the frontal approaches are coupled. [Frey 2001, Loseille 2009, Compere 2010, Dobrzynski 2008] are based on local mesh modifications. [Coupez 2000] is based on the minimal volume principle. Nowadays, metric-based mesh adaptation has become a mature field of research which has now proved its relevance for steady industrial problems. For instance in [Alauzet 2010b], the authors report a mean **anisotropic ratio of 1:400** and a **mean anisotropic quotient of 50 000** for adapted meshes containing **more than 50 millions tetrahedra**. Computations involving several millions of tetrahedra can now be considered on a daily basis, with moderate investments.

1.2 Basics of metric-based mesh adaptation

This section recalls the basics of metric-based mesh adaptation for *steady* simulations, as they are now commonly used by many research teams. First, some essential notions of Riemannian geometry are recalled, along with the central concept of *unit mesh*. Then, the main problematics linked to the error estimate are detailed. This being done, it is possible to give a global description of the adaptation process in the case of *steady* simulations. The three remaining subsections are devoted to more technical aspects.

1.2.1 Riemannian geometry and unit mesh

Riemannian geometry, by explaining how to *locally* modify geometrical measures, is the keystone of metric-based mesh adaptation. The notion of metric space is introduced in a first time, and is then generalized thanks to the concept of Riemannian space.

1.2.1.1 Euclidian affine space

Definition. Let Ω be the physical domain of the simulation. In affine space $(\Omega, \mathbb{R}^n)$, lengths of vectors and angles between vectors are usually measured by means of a scalar product, *i.e* a Symmetric Positive Definite (SPD) form. Generally, the canonical Euclidian dot product is preferred[1]:

$$\begin{array}{rccccc} (\cdot,\cdot)_{\mathcal{I}_n}: & \mathbb{R}^n \times \mathbb{R}^n & \longrightarrow & \mathbb{R}^+ & & \\ & (\mathbf{u},\mathbf{v}) & \longmapsto & (\mathbf{u},\mathbf{v})_{\mathcal{I}_n} & = \mathbf{u}^T \mathcal{I}_n \mathbf{v} & = \displaystyle\sum_{i=1}^{n} u_i v_i \,, \end{array}$$

where $\mathcal{I}_n$ is the identity matrix of $\mathbb{R}^{n\times n}$. However, it is possible to use another scalar product which will then be represented by a SPD matrix $\mathcal{M} = (m_{ij})_{1\leq i,j\leq n}$, called *metric*:

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

In any case, affine space $(\Omega, \mathbb{R}^n)$, when provided with a SPD real matrix $\mathcal{M}$, is named *Euclidian affine space*. $\mathcal{M}$ induces on $\mathbb{R}^n$:

- a metric space structure, *i.e.* a way of computing distances between points; the distance application is defined by:

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

- a normed space structure, the norm application being defined by:

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

All the geometrical quantities classically needed for mesh generation can then be computed using this new structure:

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

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

[1] superscript $.^T$ is used for the transposition operation on a vector, and $^{-T}$ holds for $\left((\cdot)^{-1}\right)^T$

- *the angle between two vectors* $\mathbf{v_1}$ and $\mathbf{v_2}$ is the unique real number $\theta \in [0, \pi]$ such that:

$$\cos\theta = \frac{(\mathbf{v_1}, \mathbf{v_2})_{\mathcal{M}}}{||\mathbf{v_1}||_{\mathcal{M}}\, ||\mathbf{v_2}||_{\mathcal{M}}}$$

- *the volume of element* K is:

$$|K|_{\mathcal{M}} = \sqrt{\det \mathcal{M}}\, |K|_{\mathcal{I}_n}\,.$$

Spectral decomposition. According to the *Spectral Theorem*, as $\mathcal{M}$ is a symmetric positive definite matrix (SPD), it is diagonalizable in an orthonormal basis:

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

$$\text{where} \begin{cases} \Lambda & = \text{diag}\,(\lambda_1, \ldots, \lambda_n) \text{ is the diagonal matrix made of the eigenvalues of } \mathcal{M}\,, \\ \mathcal{R} & = (\,\mathbf{r_1}\,|\,\mathbf{r_2}\,|\,\ldots\,|\mathbf{r_n}\,)^T \text{ is the unitary matrix (} i.e.\ \mathcal{R}^T\mathcal{R} = \mathcal{I}_n) \\ & \quad \text{made of the eigenvectors of } \mathcal{M}\,. \end{cases} \tag{1.1}$$

Unit ball. $\mathcal{M}$ can be represented by its unit ball, noted $\mathcal{E}_{\mathcal{M}}$, which is the set of all vectors the length of which is lower or equal to 1 with respect to $\mathcal{M}$:

$$\mathcal{E}_{\mathcal{M}} = \left\{\mathbf{x} \in \mathbb{R}^n \text{ such that } \mathbf{x}^T\,\mathcal{M}\,\mathbf{x} \leq 1\right\}\,.$$

Consider an ellipsoid of main axis $\mathbf{r_i}$ and of associated sizes $h_i = \lambda_i^{-\frac{1}{2}}$. The unit balls associated with $\mathcal{M}$ is the set of all points contained in an ellipsoid. The unit ball of $\mathcal{M}$ in two and three dimensions are given in Figure 1.1.

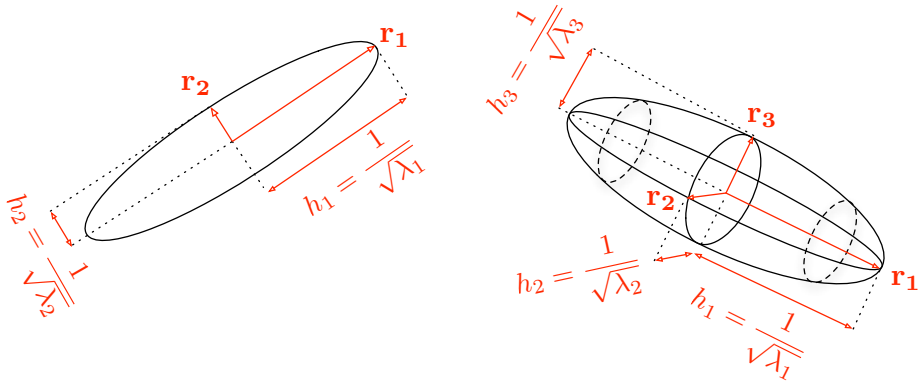


Figure 1.1: Unit balls associated with metric $\mathcal{M} = \mathcal{R}\,\Lambda\,\mathcal{R}^T$ in two and three dimensions.

Mapping. Another relevant information can be extracted from metric tensor $\mathcal{M}$: the expression of the linear transformation that maps the unit ball associated with $\mathcal{I}_n$ onto the one associated with $\mathcal{M}$. The matrix of this transformation in canonical basis $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e_n})$ is:

$$\mathcal{M}^{-\frac{1}{2}} = \mathcal{R}\,\Lambda^{-\frac{1}{2}}\,\mathcal{R}^T, \quad \text{where } \Lambda^{-\frac{1}{2}} = \text{diag}\left(h_1 = \lambda_1^{-\frac{1}{2}}, \ldots, h_n = \lambda_n^{-\frac{1}{2}}\right)\,.$$

This transformation is depicted in Figure 1.2.

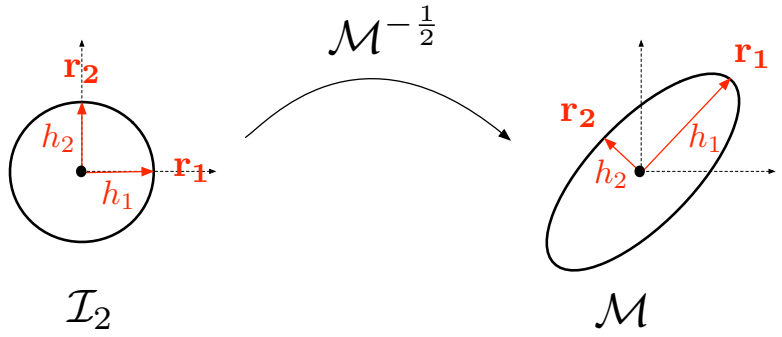


Figure 1.2: Natural mapping $\mathcal{M}^{-\frac{1}{2}}$ associated with metric $\mathcal{M}$ in two dimensions. It sends the unit ball of $\mathcal{I}_2$ onto the unit ball of $\mathcal{M}$.

1.2.1.2 Riemannian metric space

Definition. In the case of an Euclidian affine space $(\Omega, \mathcal{M})$ of , $\mathbb{R}^n$, the dot product induced by $\mathcal{M}$ is the same whatever the considered point $P \in \Omega$, *i.e.* $\mathcal{M}$ is independent of P. However, it is possible (and salutary) to change the definition of the dot product depending on the current localization in Ω. A collection of SPD tensors $\mathbf{M} = (\mathcal{M}(P))_{P\in\Omega}$ - also named tensor field - is then needed and must be defined on the whole domain. Locally at point $P \in \Omega$, $\mathcal{M}(P)$ induces a scalar product defined on $\mathbb{R}^n \times \mathbb{R}^n$. Space Ω, provided with this new structure, is named *Riemannian metric space* and is noted $(\Omega, \mathbb{R}^n, \mathbf{M})$. 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 be used only if this distinction needs to be made for pedagogical purposes.

Computation of geometrical quantities. In the case of a Riemannian metric space, the spatial variations of the metric must be taken into account while computing geometric quantities.

- *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:

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

- *the angle between two vectors* $\mathbf{v_1} = \overrightarrow{PQ}_1$ and $\mathbf{v_2} = \overrightarrow{PQ}_2$ is the unique real $\theta \in [0, \pi]$ such that:

$$\cos\theta = \frac{(\mathbf{v_1}, \mathbf{v}_2)_{\mathcal{M}(P)}}{||\mathbf{v}_1||_{\mathcal{M}(P)}\,||\mathbf{v}_2||_{\mathcal{M}(P)}}\,,$$

- *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:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}(\mathbf{x})}\, \mathrm{d}\mathbf{x}\,. \tag{1.2}$$

However, this volume can be approximated at first order:

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

Interpretation in terms of space curvature. Even if the above formal definitions are enough in practice, it is sometimes useful to interpret the introduction of a Riemannian structure as a way to *curve space*. To this aim, we address here the very specific case of graph surfaces in two dimensions.

Definition 1 (Graph surface)**.** *Let $\Omega \subset \mathbb{R}^2$. A surface S of $\mathbb{R}^3$ is called a* graph surface *if there exists a $\mathcal{C}^1$ function $f : \Omega \longrightarrow \mathbb{R}$ such that:*

$$S = \{\ (x,\, y,\, f(x,y)) \mid (x,\, y) \in \Omega\ \}\ .$$

Thanks to the smoothness hypothesis on f, there exists at each point $P \in \Omega$ a tangent plane to S.

Definition 2 (Tangent plane)**.** *The tangent plane $\Gamma_P S$ to surface S at point P is the vectorial sub-space of $\mathbb{R}^3$ of dimension 2 which contains all the vectors tangent to surface S at P.*

The normal to tangent plane $\Gamma_P S$ reads:

$$\mathbf{n}(P) = -\frac{\partial f}{\partial x}(P)\, \mathbf{e_x} - \frac{\partial f}{\partial y}(P)\, \mathbf{e_y} + \mathbf{e_z}\,.$$

We also define two vectors tangent to S at P:

$$\mathbf{t_1}(P) = \mathbf{e_x} + \frac{\partial f}{\partial x}(P)\, \mathbf{e_z}, \quad \mathbf{t_2}(P) = \mathbf{e_y} + \frac{\partial f}{\partial y}(P)\, \mathbf{e_z}\,.$$

As $\mathbf{t_1}(P) \cdot \mathbf{n}(P) = 0$ and $\mathbf{t_2}(P) \cdot \mathbf{n}(P) = 0$, both $\mathbf{t_1}(P)$ and $\mathbf{t_2}(P)$ belong to $\Gamma_P S$. Note that *a priori*, there is no reason for $\mathbf{t_1}(P)$ and $\mathbf{t_2}(P)$ to be orthogonal. The triple $\mathcal{B}_P = (\mathbf{t_1}(P),\, \mathbf{t_2}(P),\, \mathbf{n}(P))$ forms a basis of $\mathbb{R}^3$. We can then define the linear application which sends $\mathcal{B}_P$ onto $\mathcal{B} = (\mathbf{e}_x,\, \mathbf{e}_y,\, \mathbf{e}_z)$, the canonical basis of $\mathbb{R}^3$:

$$\begin{array}{rlcl} \phi^{-1}(P): & \mathbb{R}^3 & \longrightarrow & \mathbb{R}^3 \\ & \mathbf{r}(P) = \alpha\, \mathbf{t_1}(P) + \beta\, \mathbf{t_2}(P) + \gamma\, \mathbf{n}(P) & \longmapsto & \alpha\, \mathbf{e_x} + \beta\, \mathbf{e_y} + \gamma\, \mathbf{e_z}\,. \end{array}$$

By construction, ϕ_P^{-1} is an isomorphism as it is an endomorphism of $\mathbb{R}^3$ that sends a basis onto

another basis. Its inverse, ϕ_P, has a matrix in canonical basis $(\mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z})$ which writes:

$$\mathcal{F}_P = \begin{pmatrix} 1 & 0 & -\dfrac{\partial f}{\partial x}(P) \\ 0 & 1 & -\dfrac{\partial f}{\partial y}(P) \\ \dfrac{\partial f}{\partial x}(P) & \dfrac{\partial f}{\partial y}(P) & 1 \\ \mathbf{t_1}(P) & \mathbf{t_2}(P) & \mathbf{n}(P) \end{pmatrix} \begin{matrix} \mathbf{e_x} \\ \mathbf{e_y} \\ \mathbf{e_z} \end{matrix}$$

The matrix of ϕ_P^{-1} in the canonical basis can be calculated:

$$\mathcal{F}_P^{-1} = \frac{1}{1+\left[\dfrac{\partial f}{\partial x}(P)\right]^2+\left[\dfrac{\partial f}{\partial y}(P)\right]^2} \begin{pmatrix} 1+\left[\dfrac{\partial f}{\partial y}(P)\right]^2 & -\dfrac{\partial f}{\partial x}(P)\dfrac{\partial f}{\partial y}(P) & \dfrac{\partial f}{\partial x}(P) \\ -\dfrac{\partial f}{\partial x}(P)\dfrac{\partial f}{\partial y}(P) & 1+\left[\dfrac{\partial f}{\partial x}(P)\right]^2 & \dfrac{\partial f}{\partial y}(P) \\ -\dfrac{\partial f}{\partial x}(P) & -\dfrac{\partial f}{\partial y}(P) & 1 \end{pmatrix}.$$

It can be checked that $\mathcal{F}_P^{-1}\,\mathbf{t_1}(P) = \mathbf{e}_x$ and $\mathcal{F}_P^{-1}\,\mathbf{t_2}(P) = \mathbf{e}_y$, which confirms that the image of subspace $\Gamma_P S$ by ϕ_P^{-1} is exactly $\mathbb{R}^2$. $\Gamma_P S$ and $\mathbb{R}^2$ are diffeomorphic.
Now, let γ be a straight path drawn in Ω:

$$\begin{array}{rccl} \gamma: & [0,\,1] & \longrightarrow & \Omega \\ & t & \longmapsto & \gamma(t) = (\gamma_x(t),\,\gamma_y(t)) = P + t\,\overrightarrow{PQ}\,. \end{array}$$

It can be associated with another path c drawn on graph surface S and generally curved, see Figure 1.3:

$$\begin{array}{rccl} c: & [0,\,1] & \longrightarrow & S \\ & t & \longmapsto & c(t) = (\,\gamma_x(t),\,\gamma_y(t),\,f\,(\gamma(t))\,)\,. \end{array}$$

By definition, the length of curved path c is given by:

$$\begin{aligned} \ell(c) &= \int_0^1 ||\mathbf{c}'(t)||\,\mathrm{d}t = \int_0^1 ||\phi_{\gamma(t)}[\boldsymbol{\gamma}'(t)]||\,\mathrm{d}t = \int_0^1 \sqrt{\mathcal{F}_{\gamma(t)}\,[\boldsymbol{\gamma}'(t)]^T\;\mathcal{F}_{\gamma(t)}\,[\boldsymbol{\gamma}'(t)]}\,\mathrm{d}t \\ &= \int_0^1 \sqrt{\overrightarrow{PQ}^T\,\mathcal{F}^T_{P+t\,\overrightarrow{PQ}}\cdot\,\mathcal{F}_{P+t\,\overrightarrow{PQ}}\overrightarrow{PQ}}\,\mathrm{d}t \\ &= \int_0^1 \sqrt{\overrightarrow{PQ}^T\,\mathcal{S}(P+t\,\overrightarrow{PQ})\,\overrightarrow{PQ}}\,\mathrm{d}t = \int_0^1 \sqrt{\overrightarrow{PQ}^T\,\mathcal{M}(P+t\,\overrightarrow{PQ})\,\overrightarrow{PQ}}\,\mathrm{d}t\,. \end{aligned}$$

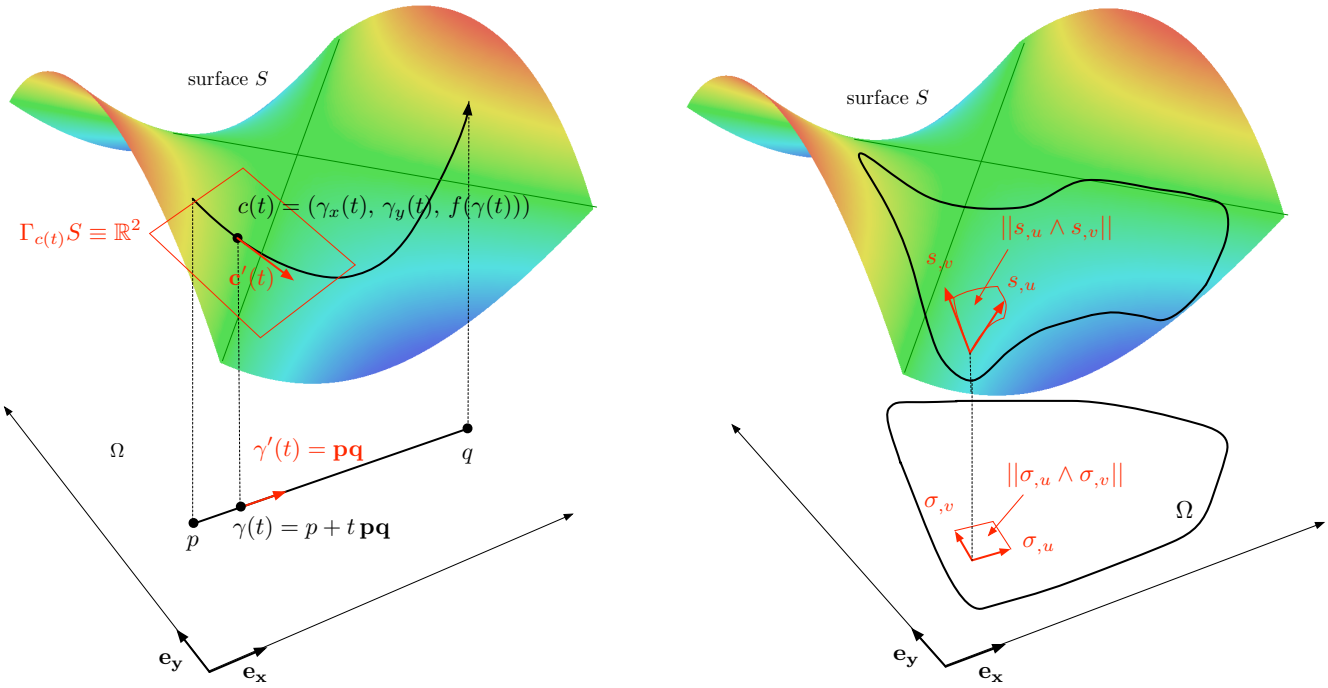


Figure 1.3: Left: A straight path drawn in Ω and its curved image path on S. Right: a patch of Ω and its image on cartesian surface S.

We have used the fact that $\overrightarrow{PQ}\,\mathbf{e_z} = 0$ and we have noted:

$$\mathcal{S} = \mathcal{F}^T\mathcal{F} = \left(\begin{array}{c|c} \mathcal{M} & 0 \\ \hline 0 & 1 + \left(\dfrac{\partial f}{\partial x}\right)^2 + \left(\dfrac{\partial f}{\partial y}\right)^2 \end{array}\right),$$

and $\mathcal{M}$ is the first fundamental form of S, which is of course a SPD tensor field:

$$\mathcal{M} = \begin{pmatrix} 1 + \left(\dfrac{\partial f}{\partial x}\right)^2 & \dfrac{\partial f}{\partial x}\dfrac{\partial f}{\partial y} \\ \dfrac{\partial f}{\partial x}\dfrac{\partial f}{\partial y} & 1 + \left(\dfrac{\partial f}{\partial y}\right)^2 \end{pmatrix}. \tag{1.3}$$

Therefore, computing the length of edge $\overrightarrow{PQ}$ in a Riemannian space comes to calculate the length of image path c associated with edge $\overrightarrow{PQ}$ on curved surface S. In other words, the introduction of a Riemannian structure can be seen as a curvature of space. The volume evaluation given by Formula 1.2 can also be interpreted in this framework. Let σ be a flat patch drawn in Ω:

$$\begin{array}{rccl} \sigma: & [0,1]\times[0,1] & \longrightarrow & \mathbb{R}^2 \\ & (u,v) & \longmapsto & \sigma(u,v) = (\sigma_x(u,v), \sigma_y(u,v)) \,. \end{array}$$

Its image on S is a curved patch s, see Figure 1.3:

$$\begin{array}{rccl} s: & [0,\,1]\times[0,\,1] & \longrightarrow & S \\ & (u,\,v) & \longmapsto & s(u,\,v)=(\sigma_x(u,\,v),\,\sigma_y(u,\,v),f(\sigma(u,\,v)))\,. \end{array}$$

By definition, the area of curved patch s is:

$$\mathcal{A}(s)=\int_0^1\int_0^1 ||\frac{\partial s}{\partial u}(u,\,v)\,\wedge\,\frac{\partial s}{\partial v}(u,\,v)||\,\mathrm{d}u\,\mathrm{d}v$$

Besides, the derivatives involved in the above formula can be detailed:

$$\frac{\partial s}{\partial u}=s_{,u}=\left(\sigma_{,u},\,\mathrm{d}f_{\sigma(u,v)}\left[\sigma_{,u}\right]\right)^T,\quad \frac{\partial s}{\partial v}=s_{,v}=\left(\sigma_{,v},\,\mathrm{d}f_{\sigma(u,v)}\left[\sigma_{,v}\right]\right)^T$$

Their vectorial product gives:

$$\frac{\partial s}{\partial u}\,\wedge\,\frac{\partial s}{\partial v}=\begin{pmatrix}\sigma_{x,u}\\ \sigma_{y,u}\\ \mathrm{d}f\left[\sigma_{,u}\right]\end{pmatrix}\wedge\begin{pmatrix}\sigma_{x,v}\\ \sigma_{y,v}\\ \mathrm{d}f\left[\sigma_{,v}\right]\end{pmatrix}=\begin{pmatrix}\sigma_{y,u}\mathrm{d}f\left[\sigma_{,v}\right]-\sigma_{y,v}\mathrm{d}f\left[\sigma_{,u}\right]\\ \sigma_{x,v}\mathrm{d}f\left[\sigma_{,u}\right]-\sigma_{x,u}\mathrm{d}f\left[\sigma_{,v}\right]\\ \|\sigma_{,u}\,\wedge\,\sigma_{,v}\|\end{pmatrix}$$

$$=\begin{pmatrix}\mathrm{d}f\begin{bmatrix}\sigma_{y,u}\sigma_{x,v}-\sigma_{y,v}\sigma_{x,u}\\ \sigma_{y,u}\sigma_{y,v}-\sigma_{y,v}\sigma_{y,u}\end{bmatrix}\\ \mathrm{d}f\begin{bmatrix}\sigma_{x,v}\sigma_{x,u}-\sigma_{x,u}\sigma_{x,v}\\ \sigma_{x,v}\sigma_{y,u}-\sigma_{x,u}\sigma_{y,v}\end{bmatrix}\\ \|\sigma_{,u}\,\wedge\,\sigma_{,v}\|\end{pmatrix}=\begin{pmatrix}\mathrm{d}f\begin{bmatrix}\|\sigma_{,u}\,\wedge\,\sigma_{,v}\|\\ 0\end{bmatrix}\\ \mathrm{d}f\begin{bmatrix}0\\ \|\sigma_{,u}\,\wedge\,\sigma_{,v}\|\end{bmatrix}\\ \|\sigma_{,u}\,\wedge\,\sigma_{,v}\|\end{pmatrix}=\|\sigma_{,u}\,\wedge\,\sigma_{,v}\|\begin{pmatrix}\frac{\partial f}{\partial x}\\ \\ \frac{\partial f}{\partial y}\\ \\ 1\end{pmatrix}$$

And finally, we get:

$$\|\frac{\partial s}{\partial u}(u,v)\,\wedge\,\frac{\partial s}{\partial v}(u,v)\|^2=\|\sigma_{,u}(u,v)\,\wedge\,\sigma_{,v}(u,v)\|^2\left(1+\left[\frac{\partial f}{\partial x}(\sigma(u,v))\right]^2+\left[\frac{\partial f}{\partial y}(\sigma(u,v))\right]^2\right)$$

$$=\|\sigma_{,u}\,\wedge\,\sigma_{,v}\|^2\left(\left[1+\left(\frac{\partial f}{\partial x}\right)^2\right]\left[1+\left(\frac{\partial f}{\partial y}\right)^2\right]-\left(\frac{\partial f}{\partial x}\right)^2\left(\frac{\partial f}{\partial x}\right)^2\right)$$

$$=\|\sigma_{,u}(u,v)\,\wedge\,\sigma_{,v}(u,v)\|^2\,\det\mathcal{M}(\sigma(u,v))\,.$$

Therefore, the total area of patch s is:

$$\mathcal{A}\,(s)=\int_0^1\int_0^1\sqrt{\det\mathcal{M}(\sigma(u,v))}\,\|\sigma_{,u}(u,v)\,\wedge\,\sigma_{,v}(u,v)\|\,\mathrm{d}v\,\mathrm{d}u=\int_\sigma\sqrt{\det\mathcal{M}(\sigma)}\,\,\mathrm{d}\sigma\,.$$

Remark 1. *Expression (1.3) justifies why Euclidian metric spaces are often described as "flat" spaces. If graph surface S is a plane of $\mathbb{R}^3$, i.e. $\exists\,(a,\,b,\,c)\,\in\,\mathbb{R}^3$ such that $f(x,y)=ax+by+c\,,\ \forall\,(x,\,y)\in\mathbb{R}^2$, then its metric is constant on Ω and, according to Expression (1.3), is given by:*

$$\mathcal{M}(P)=\mathcal{M}=\begin{pmatrix}1+a^2 & ab\\ ab & 1+b^2\end{pmatrix},\quad\forall\,P\in\Omega\,.$$

The natural Riemmanian space structure associated with S is therefore an Euclidian space structure.

As a conclusion of this small study, the particular case of graph surfaces shows how the computation of geometric quantities in a Riemannian metric space is tightly linked to differential geometry and to the notion of parametrized curves and surfaces.
However, this simple study does not enable to conclude that a graph surface can be associated with any metric field $\mathbf{M}$, and there is absolutely no reason for it to be true. The only conclusion that can be deduced from this analysis is that, if there exist a graph surface S having $\mathcal{M}$ as first fundamental form, it is not unique. Indeed, if S is associated with function f, the graph surface associated with function $-f$ also has the same metric.

1.2.2 Metric-based mesh adaptation

The notion of metric field has been introduced in the previous section mainly through its ability of changing locally the way of computing geometrical quantities such as length, distances, angles and volumes. Now, we illustrate how metric fields can be used in the context of mesh adaptation and we give an overview of the steady metric-based mesh adaptation process.

The main idea of metric-based mesh adaptation, initially introduced in [George 1991], is to generate *a unit mesh* in the prescribed Riemannian metric space, e.g. a mesh $\mathcal{H}$ of $\Omega_h \subset \mathbb{R}^n$ such that each edge has a unit length and each element is regular with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$:

$$\forall \mathbf{e},\ \ \ell_{\mathcal{M}}(\mathbf{e}) = 1 \quad \text{and} \quad \forall K \in \mathcal{H},\ \ |K|_{\mathcal{M}} \ = \ \begin{cases} \dfrac{\sqrt{3}}{4} \text{ in 2D}\,, \\ \\ \dfrac{\sqrt{2}}{12} \text{ in 3D}\,. \end{cases}$$

The resulting mesh in the canonical Euclidean space will be anisotropic and adapted. The mesh generator actually computes all the geometric quantities using the Riemannian structure induced by $\mathbf{M}$ instead of the canonical Euclidian one. In other words, it works mostly as it used to, except that all the geometric quantities are now computed with respect to metric field $\mathbf{M}$.

In practice, as it is not possible to tesselate $\mathbb{R}^3$ with the regular tetrahedron and due to boundary discretization enforcement, we look for a mesh such that all its edges have a length close to unity in Riemannian space (Ω, $\mathbf{M}$) and such that all the elements are almost regular in the considered Riemannian metric space. Such a mesh is said to be *quasi-unit* with respect to $\mathbf{M}$.
In practice, two quantities are controlled by the meshing software:

Edges length : the following bounds are enforced:

$$\forall\, i \in [\![0,\, n-1]\!],\ \ell_{\mathcal{M}}(\mathbf{e_i}) \in [\frac{1}{\sqrt{2}},\, \sqrt{2}]\,.$$

Elements volume : the volume of the elements is controlled *via* the notion of quality with respect to $\mathbf{M}$. The local quality of element K with respect to $\mathcal{M}$ is given by:

$$Q_{\mathcal{M}}(K) = \frac{36}{3^{\frac{1}{3}}} \frac{\sum_{i=0}^{5} \ell^2_{\mathcal{M}}(\mathbf{e_i})}{|K|^{\frac{2}{3}}_{\mathcal{M}}} \in [1, +\infty[\text{ in 3D}, \quad Q_{\mathcal{M}}(K) = \frac{12}{3^{\frac{1}{2}}} \frac{\sum_{i=0}^{2} \ell^2_{\mathcal{M}}(\mathbf{e_i})}{|K|_{\mathcal{M}}} \in [1, +\infty[\text{ in 2D}. \tag{1.4}$$

$Q_{\mathcal{M}}(K) = 1$ corresponds to an element almost regular with respect to $\mathcal{M}$ while $Q_{\mathcal{M}}(K) = +\infty$ indicates a degenerated element. The mesh adaptation software hence tends to create elements with a quality near from 1.

The continuous adapted mesh $\mathbf{M}$ is built from the estimation of the numerical errors made on sensor u. It therefore depends on whether the error is controlled locally or globally, on the norm used to control the error, whether the error estimation is directional or not...The estimates used in this Thesis are detailed in Section 1.3 for steady mesh adaptation and in Section 2.2 for unsteady mesh adaptation.

1.2.2.1 The non-linear steady adaptation loop.

The last thing concerns the intrinsic *non-linearity* of any adaptive process. Indeed, if the mesh is adapted on an initial solution and if a new solution is computed on the adapted mesh, this new solution will normally be more accurate than the initial one. More detailed information about the physical features of the solution are therefore available after this second computation, which again can enrich the information contained in the metric field and lead to a more optimal mesh. Consequently, we see that the goal is to algorithmically converge towards the stationary solution of the problem and in the same time to converge towards an adapted mesh thanks to an *iterative algorithm* on a sequence of consecutively adapted meshes. This iterative scheme is illustrated in Figure 1.4. The algorithm starts with a given initial, generally uniform, mesh $\mathcal{H}_0$ and initial solution W_0^0 on this mesh. A first solution W_1 is computed, from a which a metric field $\mathcal{M}_1$ is deduced, see Formula (1.22). $\mathcal{M}_1$ is used as input by the mesh generator which will build a new mesh $\mathcal{H}_1$ respecting the sizes and orientations prescribed by the metric field. Then, solution W_1 is interpolated on new mesh $\mathcal{H}_1$, see Section 1.2.3 and is used as initial guess for the computation of a new solution on $\mathcal{H}_1$. This process is repeated until the couple mesh/solution has converged. For most problems, convergence is reached in less than 10 iterations.

1.2.3 Interpolation issues

After each mesh adaptation, the solution needs to be transferred from the previous mesh to the new one in order to pursue the computation. This is the solution interpolation stage. As we will see in the next chapters, this stage becomes crucial in the context of unsteady problems and even more if a large number of interpolations are performed, as the error introduced by the interpolation can spoil the solution accuracy. This subsection gives some technical details on the interpolation algorithm. The following notations are used: $\mathcal{H}^{back}$ and W^{back} are the background mesh and solution, respectively, while $\mathcal{H}^{new}$ and W^{new} represent the new mesh and the interpolated solution to be computed, respectively.

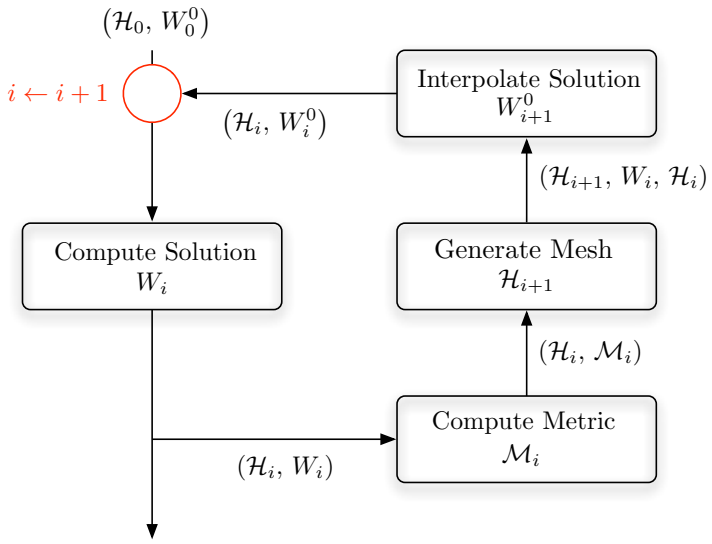


Figure 1.4: The non-linear steady adaptation loop. i is the adaptive loop iteration index, $\mathcal{H}^i$, W^i, W_0^i and $\mathbf{M}_i$ denote the mesh, the solution, the initial solution and the metric field at iteration i, respectively.

1.2.3.1 Localization

First, new vertices are located in the background mesh by identifying the elements containing them. Let P be a point of Ω. The barycentric coordinates of a point P with respect to a given element $K = (P_0, P_1, \ldots, P_n)$, noted $(\beta_j(P))_{j \in [\![1, n+1]\!]}$, are given by:

$$\beta_j(P) = \frac{|K_j(P)|}{|K|},$$

where $K_j(P)$ is the element formed by vertices $(P_0, \ldots, P_{j-1}, P, P_{j+1}, \ldots, P_n)$, see Figure 1.5, left. In two dimensions, the sign of the three barycentric coordinates (or barycentrics) defines explicitly seven regions of the plane where point P can be located with respect to element K. The possible combinations are given in Figure 1.5, right. The localization phase is based on the study of the sign of the barycentrics of each vertex P^{new} of $\mathcal{H}^{new}$ relative to all the elements of $\mathcal{H}^{back}$. The knowledge of these signs enables to determine to which element P^{new} belong. Of course, this simple explanation hides many tricky aspects used in the localization process. Notably, the handling of degenerated cases, for example when P^{new} belongs to an edge, a face or a vertex of $\mathcal{H}^{back}$, and non-identical domains requires a lot of rigor. Exhaustive computations of the barycentrics relative to *all* the background element must also be banished, otherwise the algorithm may become very slow (actually quadratic). More clever strategy consist in traversing the background mesh using its topology, *i.e.* the neighboring elements of each element, thanks to a barycentric coordinates-based algorithm [Alauzet 2010c, Frey 2008, Löhner 2001].

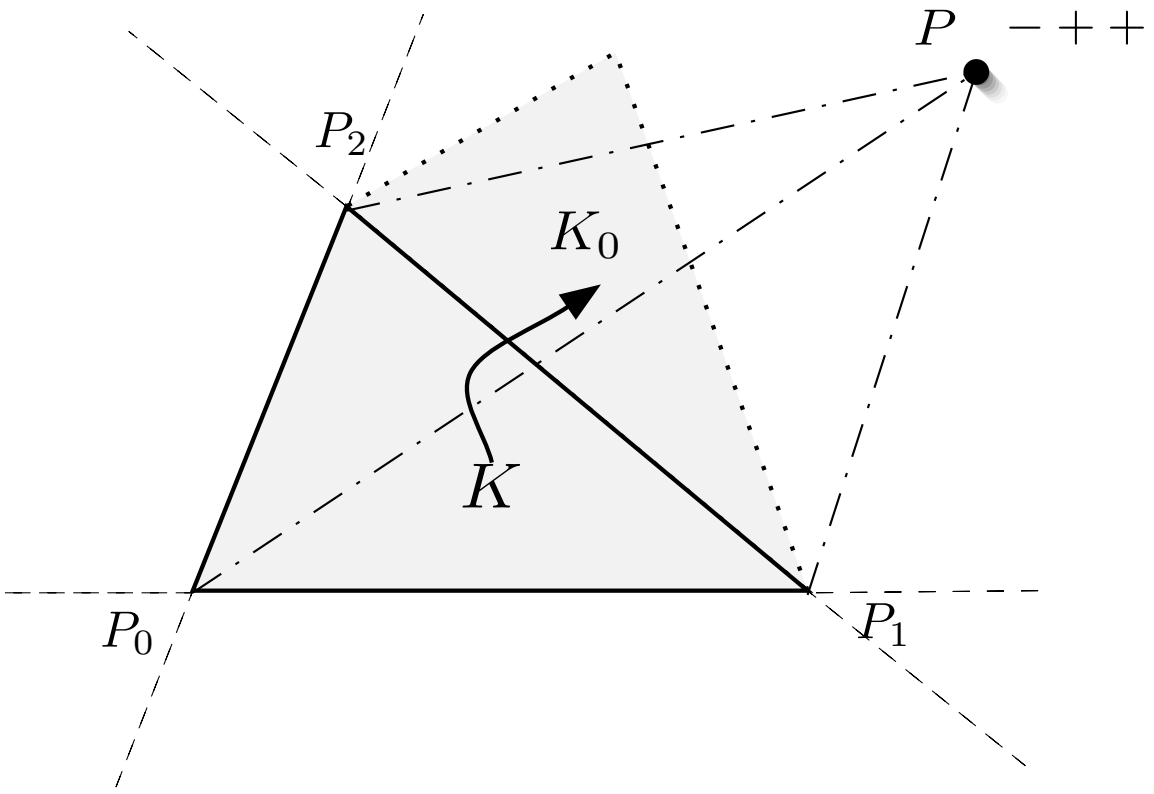


Figure 1.5: Left, definition of volume $V_j(P)$. Right, sign of the three barycentric coordinates depending on the position of P relative to triangle K.

1.2.3.2 Desirable properties for the interpolation stage

Once the localization problem has been solved, an interpolation scheme is used to extract the information from the solution field. This thesis is interested in the resolution of systems of Partial Differential Equation resulting from the writing of physical conservation laws, notably the compressible Euler system. If the numerical scheme reaches a second-order accuracy, it is mandatory for the interpolation operator Π to satisfy the following properties in order to obtain a mesh adaptation scheme which remains consistent with the solver:

- *Mass conservation.* At the continuous level, the conservativity property writes, for a conservative state W:

$$\int_{\Omega} W(\mathbf{x})\,\mathrm{d}\mathbf{x} \;=\; Constant\,.$$

  Now, let W_h^{new} be the interpolated conservative solution state obtained on $\mathcal{H}^{new}$ and W_h^{back} the background solution state on $\mathcal{H}^{back}$. Assuming $\mathcal{H}^{new}$ and $\mathcal{H}^{back}$ cover exactly the same domain, the interpolation is said to be *conservative* if:

$$\int_{\mathcal{H}^{new}} W_h^{new}(\mathbf{x})\,\mathrm{d}\mathbf{x} \;=\; \int_{\mathcal{H}^{back}} W_h^{back}(\mathbf{x})\,\mathrm{d}\mathbf{x}\,.$$

If W_h^{back} and W_h^{new} are $\mathbf{P}^1$ by element, this writes:

$$\sum_{K^{new} \in \mathcal{H}^{new}} |K^{new}|\, W_h^{new}\,{}_{|K^{new}} \;=\; \sum_{K^{back} \in \mathcal{H}^{back}} |K^{back}|\, W_h^{back}\,{}_{|K^{back}}\,.$$

- $\mathbf{P}^1$-*exactness.* An interpolation operator Π is said to be $\mathbf{P}^1$-exact if:

$$\Pi q \,=\, q, \quad \forall\, q \,\in\, \mathbf{P}^1\,.$$

- *Maximum principle enforcement.* The maximum principle, which is linked to the physics of the equation, states that no new local or global extremum can be created during the interpolation stage.

1.2.3.3 Interpolation schemes

Lagrangian interpolation. The most classical way to interpolate a solution on a new mesh is to perform a Lagrangian interpolation:

Linear: the solution is assumed $\mathbf{P}^1$ by element, the value of solution u at a given point P of Ω is:

$$u(P) = \sum_{j=0}^{n} u(P_j)\,\varphi_j(P)\,,$$

and $(\varphi_j)_{j\,\in[\![1,\,n]\!]}$ are the $\mathbf{P}^1$ shape functions, which are exactly the barycentric coordinates (β_j). The linear Lagrangian interpolation is $\mathbf{P}^1$-exact and respects the maximum principle, but it is not conservative.

Quadratic: $\mathbf{P}^2$ Lagrangian shape functions in triangle K are used to build a quadratic representation of the solution on K.

The $\mathbf{P}^2$ Lagrangian interpolation Π_h^2 requires the solution nodal values at triangle vertices P_0, P_1 and P_2, and the solution at the triangle mid-edges. We denote by P_3, P_4 and P_5 the middle of edges $\mathbf{e}_2$, $\mathbf{e}_0$ and $\mathbf{e}_1$, respectively. Thus, P_{j+3} is the middle of edge $\overrightarrow{P_jP_{j+1}}$, for $j \,\in\, [\![0,\,2]\!]$. The quadratic scheme is given by:

$$\Pi_h^2 u(P) = \sum_{j=0}^{n_{nodes}-1} u(P_j)\,\psi_j(P)\,, \quad \text{with } n_{nodes} = 6 \text{ in 2D and } n_{nodes} = 10 \text{ in 3D}\,.$$

$$\text{with } \begin{cases} \psi_j(P) & = & \beta_j(P)\,(2\,\beta_j(P) - 1) & \text{for } j \,\in [\![0,\,n]\!]\,, \\ \psi_{j+n}(P) & = 4 & \beta_j(P)\,\beta_{j+1}(P) & \text{for } j \,\in [\![n+1,\,n_{nodes}-1]\!]\,, \,. \end{cases}$$

If the mid-edge values are known, then this interpolation is of order 3. However, in our case the solution representation is continuous and piecewise linear by element. Therefore, we have to specify how the mid-edge solution values are obtained. From the solution nodal values, gradients at vertices can be reconstructed, see Section 1.2.4.1. Let $\nabla u(P_i)$ be the recovered gradient at vertex P_i. In this case, we have an over-determined system for the quadratic reconstruction. Consequently, we choose to perform a cubic reconstruction on each edge to get the mid-edge

values using the solution and the gradient at each extremity. After some algebra, mid-edge values are given by:

$$u(P_{j+3}) = \frac{u(P_j) + u(P_{j+1})}{2} + \frac{\nabla u(P_j) - \nabla u(P_{j+1})}{8} \cdot \overrightarrow{P_j P_{j+1}}, \text{ for } j \in [\![0, 2]\!].$$

The quadratic interpolation has an accuracy which depends on the quality of the gradient reconstruction. For the proposed approach, an order comprised between 2 and 3 is obtained. The reconstruction is $\mathcal{C}^1$ except in the direction normal to the edges of the elements. However, the quadratic interpolation is not conservative and does not verify the maximum principle. This latter insufficiency can be repaired by limiting the reconstructed values and thus avoiding the creation of new extrema.

Conservative interpolation. To remedy the disadvantages of the Lagrangian interpolation operators, conservative interpolation methods have been proposed, see for instance [Farrell 2009, Alauzet 2011].
In this method, 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 for conservative interpolation seems natural for unconnected meshes. The *locality* is primordial for efficiency and robustness. The idea is, for each element of the new mesh, to compute its geometric intersection with all the elements of the previous mesh that it overlaps and to mesh with simplexes this geometric intersection. We are then able to use Gauss quadrature formula to exactly compute the transferred mass.
The high-order accuracy is obtained by a solution gradient reconstruction from the discrete data and the use of Taylor formulae. This conservative interpolation can however generate a loss of monotonicity. If such situation occurs, the maximum principle is then enforced by correcting conservatively the interpolated solution. Finally, vertex valued solutions are reconstructed from this piecewise linear by element, discontinuous representation.
The algorithm can be summarized as follows:

1. A piecewise linear (continuous or discontinuous) representation of the solution on background mesh $\mathcal{H}^{back}$ is known,
2. $\forall K^{back} \in \mathcal{H}^{back}$, compute the solution mass $m_{K^{back}}$ and the solution (constant) gradient $\nabla_{K^{back}} u$,
3. $\forall K^{new} \in \mathcal{H}^{new}$, recover the solution mass $m_{K^{new}}$ and the solution gradient $\nabla_{K^{new}} u$:
   (a) Compute the intersection of K^{new} with all the elements $K_i^{back} \in \mathcal{H}^{back}$ its overlaps,
   (b) Mesh the intersection polygon of each couple (K^{new}, K_i^{back}) of elements,
   (c) Compute $m_{K^{new}}$ and $\nabla_{K^{new}} u$ using Gauss quadrature formula.

   $\Longrightarrow$ a piecewise linear discontinuous representation of the mass on $\mathcal{H}^{new}$ is obtained,
4. Correction to verify the maximum principle (application of a conservative slope limiter),
5. Displace the solution values at vertices by averaging.

It can be demonstrated that this interpolation process exhibits all the desirable properties described above.

1.2.4 Gradient and Hessian recovery techniques

The numerical approximation u_h computed on $\mathcal{H}$ is generally not smooth. However, $\mathbf{P}^2$ or conservative interpolation require the computation of the gradient of the solution. In Section 1.4, we will also see that a reconstruction of the Hessian of the solution is also mandatory to build the optimal metric field. Therefore, a smooth reconstruction of the solution, at least $\mathcal{C}^2$ must be achieved to be able to define gradients and Hessians. This subsection presents the most classical recovery techniques.

1.2.4.1 Continuous element-wise linear gradient recovery (nodal gradients)

Let K be an element and $P_0, P_1, \dots, P_n$ its vertices. The $\mathbf{P}^1$ approximation of u is:

$$u_h(\mathbf{x}) = \sum_{j=0}^{n} u_h(P_j)\, \varphi_j(\mathbf{x})\,,$$

where $(\varphi_j)_j$ are the $\mathbf{P}^1$ shape function given in two dimensions by:

$$\nabla_{\boldsymbol{x}}\varphi_0(\mathbf{x}) = \frac{1}{2|K|}\overline{\boldsymbol{\eta_0}}\,, \quad \nabla_{\boldsymbol{x}}\varphi_1(\mathbf{x}) = \frac{1}{2|K|}\overline{\boldsymbol{\eta_1}}\,, \quad \nabla_{\boldsymbol{x}}\varphi_2(\mathbf{x}) = \frac{1}{2|K|}\overline{\boldsymbol{\eta_2}}\,,$$

and in three dimensions by:

$$\nabla_{\boldsymbol{x}}\varphi_0(\mathbf{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_0}}, \quad \nabla_{\boldsymbol{x}}\varphi_1(\mathbf{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_1}}, \quad \nabla_{\boldsymbol{x}}\varphi_2(\mathbf{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_2}}, \quad \nabla_{\boldsymbol{x}}\varphi_3(\mathbf{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_3}}\,.$$

By derivation, a piecewise constant gradient is obtained:

$$\nabla u_h|_K = \sum_{j=0}^{n} u_h(P_j)\, \nabla_{\mathbf{x}}\varphi_j(\mathbf{x})\,. \tag{1.5}$$

The

As ∇u_h is not defined at mesh vertices, the nodal gradients are recovered from the piecewise constant representation of ∇u_h, using a $\mathbf{L}^2$-projection method. The local $\mathbf{L}^2$-projection operator is based on the Clément interpolation operator [Clément 1975].

Let P_i be a vertex of mesh $\mathcal{H}$. The stencil of shape function φ_i is the topological ball of P_i, $\mathrm{Ball}(P_i)$. The following approximation spaces are introduced:

$$\begin{aligned} V_h^0 &= \{v \in \mathbf{L}^2(\Omega) \;\big|\; v_{|K} \in \mathbf{P}^0 \quad \forall K \in \mathcal{H}\} \\ V_h^1 &= \{v \in \mathcal{C}^0(\Omega) \;\big|\; v_{|K} \in \mathbf{P}^1 \quad \forall K \in \mathcal{H}\}\,. \end{aligned}$$

where $\mathbf{P}^0$ and $\mathbf{P}^1$ are the set of constant and linear polynomials. The aim is to find, in a $\mathbf{L}^2$-norm sense, the best constant gradient $\nabla_R u_h$ approximating piecewise constant field ∇u_h on $\mathrm{Ball}(P_i)$. More precisely, for $v \in \mathbf{L}^2(\Omega)$, we define $\Pi_{\mathbf{L}^2} v \in V_h^0$ by:

$$\forall\, \mathrm{Ball}(P_i) \subset \mathcal{H}\,, \quad \begin{cases} (\Pi_{\mathbf{L}^2} v)_{|\mathrm{Ball}(P_i)} \in \mathbf{P}^0 \\ \displaystyle\int_{\mathrm{Ball}(P_i)} (\Pi_{\mathbf{L}^2} v - v)\; w = 0\,, \;\; \forall w \in \mathbf{P}^0\,. \end{cases}$$

We then define the Clément interpolation operator $\Pi_c : V_h^0 \longrightarrow V_h^1$:

$$\Pi_c v := \sum_{i=0}^{n} \Pi_{\mathbf{L}^2} v(P_i)\, \varphi_i \,.$$

Using Clément interpolation operator, we now describe how we recover nodal gradients from $\nabla u_h \in \mathbf{P}^0$. For each $\mathrm{Ball}(P_i) \subset \mathcal{H}$ we have for the particular choice $v = 1 \in \mathbf{P}^0$ for the test function:

$$\begin{aligned}
\int_{\mathrm{Ball}(P_i)} (\Pi_{\mathbf{L}^2}(\nabla u_h) - \nabla u_h) = 0 \quad &\Longleftrightarrow \quad \int_{\mathrm{Ball}(P_i)} \Pi_{\mathbf{L}^2}(\nabla u_h) = \int_{\mathrm{Ball}(P_i)} \nabla u_h \\
&\Longleftrightarrow \quad |\mathrm{Ball}(P_i)|\, \Pi_{\mathbf{L}^2}(\nabla u_h)_{|\mathrm{Ball}(P_i)} = \sum_{K_j \in \mathrm{Ball}(P_i)} \int_{K_j} \nabla u_h \\
&\Longleftrightarrow \quad \Pi_{\mathbf{L}^2}(\nabla u_h)_{|\mathrm{Ball}(P_i)} = \frac{\sum\limits_{K_j \in \mathrm{Ball}(P_i)} |K_j|\, \nabla u_{h|K_j}}{\sum\limits_{K_j \in \mathrm{Ball}(P_i)} |K_j|}
\end{aligned}$$

where $|K|$ and $|\mathrm{Ball}(P_i)|$ denote the volume of element K and of topological ball $\mathrm{Ball}(P_i)$, respectively. For each vertex P_i, we thus have the following gradient reconstruction:

$$\nabla_R\, u_h(P_i) = \frac{\sum\limits_{K_j \in \mathrm{Ball}(P_i)} |K_j|\, \nabla u_{h|K_j}}{\sum\limits_{K_j \in \mathrm{Ball}(P_i)} |K_j|} \tag{1.6}$$

In fact, this procedure is equivalent to a reconstruction by means of a volume-weighted averaging. The recovery procedure provides us with gradient nodal values and thus we get a piecewise linear representation of the gradient thanks to the Clément interpolation operator.

1.2.4.2 Hessian recovery based on a double $\mathbf{L}^2$-projection method

We can use the $\mathbf{L}^2$-projection method described in Section 1.2.4.1 to recover the Hessian of a smooth representation $R_h u_h$ of u_h, see Section 1.4.2. To this end, the recovery described above is simply applied to each component of recovered gradient $\nabla_R u_h$.

1.2.4.3 Hessian recovery based on the Green formula

A continuous representation of the Hessian of the solution can be recovered using a weak formulation based on the Green formula. We consider the same notations as previously. In three dimensions, for each vertex P_k of $\mathcal{H}$, we have for $1 \leq i, j \leq 3$:

$$\begin{aligned}
\int_{\mathcal{H}} \frac{\partial^2 u_h}{\partial x_i \partial x_j}\, \varphi_k \;&=\; \int_{\mathrm{Ball}(P_k)} \frac{\partial^2 u_h}{\partial x_i \partial x_j}\, \varphi_k = -\int_{\mathrm{Ball}(P_k)} \frac{\partial u_h}{\partial x_j}\, \frac{\partial \varphi_k}{\partial x_i} + \int_{\partial \mathrm{Ball}(P_k)} \nabla_{\mathbf{x}} \partial u_h|_{\partial \mathrm{Ball}(P_k)} \cdot \mathbf{n}\, \varphi_k \, d\sigma \\
&=\; -\sum_{K \in \mathrm{Ball}(P_k)} \int_K \frac{\partial u_h}{\partial x_j}\, \frac{\partial \varphi_k}{\partial x_i} \,,
\end{aligned}$$

as the shape function is zero on the boundary of Ball(P_k). A specific treatment is done close to the boundary. Each component of the Hessian is then recovered with relation:

$$\frac{\partial^2 R_h u_h}{\partial x_i \partial x_j}(P_k) := \frac{-\int\limits_{\text{Ball}(P_k)} \frac{\partial u_h}{\partial x_j}\,\frac{\partial \varphi_k}{\partial x_i}}{\int\limits_{\text{Ball}(P_k)} \varphi_k} = -\frac{\sum\limits_{K\,\in\,\text{Ball}(P_k)} (\frac{\partial u_h}{\partial x_j})|_K \int_K \frac{\partial \varphi_k}{\partial x_i}}{\frac{|\text{Ball}(P_k)|}{4}}\,.$$

which is equivalent to lumping the mass matrix of the left-hand side of the above relation.

1.2.4.4 A least-square approach

In the sequel, R_h denotes a recovery operator, at least two-times differentiable. R_h enables to build, from the piecewise linear numerical approximation u_h, a two-times differentiable representation of the solution, noted $R_h u_h$. R_h is defined so that $R_h u_h(P_i) = u_h(P_i)$, for any vertex P_i of $\mathcal{H}$. Moreover, its gradients is $\nabla_{\mathbf{x}} R_h u_h = \nabla_R u_h$, where $\nabla_R u_h$ is the piecewise linear gradient defined by Formula 1.6.

The objective is to find the value of $H_{R_h u_h}$ at each vertex. Let P be a vertex and P_i a vertex of its ball, *i.e.* $P_i \in \text{Ball}(P)$. A Taylor expansion of $R_h u_h$ between vertices P and $P_i$leads to:

$$R_h u_h(P_i) = R_h u_h(P) + \overrightarrow{PP_i}\,.\,\nabla_{\mathbf{x}} R_h u_h(P) + \frac{1}{2}\,\langle\, \overrightarrow{PP_i}^T,\, H_{R_h u_h}(P)\,\overrightarrow{PP_i}\rangle + \mathcal{O}(||\overrightarrow{PP_i}||^3)\,.$$

By truncating this development at second order and using the properties of operator R_h, we get:

$$\begin{aligned} & u_h(P_i) = u_h(P) + \overrightarrow{PP_i}\,.\,\nabla_R u_h(P) + \frac{1}{2}\,\langle\, \overrightarrow{PP_i}^T,\, H_{R_h u_h}(P)\,\overrightarrow{PP_i}\rangle \\ \Leftrightarrow\quad & \frac{1}{2}\,\langle\, \overrightarrow{PP_i}^T,\, H_{R_h u_h}(P)\,\overrightarrow{PP_i}\rangle = u_h(P_i) - u_h(P) - \overrightarrow{PP_i}\,.\,\nabla_R u_h(P)\,. \end{aligned}$$

This relation can be developed using the notations:

$$\overrightarrow{PP_i} \;=\; (x_i\ y_i\ z_i)^T\,,\quad \nabla_R u_h(P) \;=\; (\alpha\ \beta\ \gamma)^T\,,\quad H_{R_h u_h}(P) = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}\,,$$

which leads to:

$$\frac{1}{2}(a x_i^2 + 2b x_i y_i + 2c x_i z_i + d y_i^2 + 2e y_i z_i + f z_i^2) = u_h(P_i) - u_h(P) - (\alpha x_i + \beta y_i + \gamma z_i)\,. \quad (1.7)$$

This resulting system is usually over-determined[2] of the form:

$$AX = B\,,\qquad \text{with}\qquad X^T = \begin{pmatrix} a & b & c & d & e & f \end{pmatrix}$$

where A is a $n_{ball} \times 6$ matrix, $n_{ball} = Card(\text{Ball}(P)))$ function of $(x_i\, y_i,\, z_i)$ and B is a vector of dimension n_{ball} given by the right-hand side of Relation (1.7), and function of

[2]The system is overdetermined as 6 coefficients must be computed and the vertex P is usually connected to more than 6 vertices P_i in three dimensions.

$(\alpha, \beta, \gamma, x_i, y_i, z_i, u, u_i)$. This system is solved using a least-square approximation, *i.e.,* it consists in minimizing the distance between the vectors AX and B of $\mathbb{R}^6$ by minimizing the square of the Euclidean norm of their difference. The problem is then to:

$$\text{Find } X \in \mathbb{R}^6 \text{ such that } \|AX - B\|^2 = \inf_{Y \in \mathbb{R}^6} \left(\|AY - B\|^2 \right).$$

It can be shown that the solution of this problem is the solution of the linear 6×6 system of normal equations:

$$A^T AX = A^T B\,.$$

which is solved using a standard Gauss elimination method.

Remark 2. *In the case where the system is under-determined,* i.e. $n_{ball} < 6$*, additional vertices connected to the vertices of Ball*(P) *can be taken into account.*

1.2.5 Operations on metrics

The main advantage when working with metric spaces is the well-posedness of operations on metric tensors, among which the metric intersection and the metric interpolation. These operations have a straightforward geometric interpretation when considering the ellipsoid associated with a metric.

1.2.6 Metrics intersection

When several metrics are specified at a point of the domain, all these metric tensors must be reduced to a single one due to mesh generation concerns. The **metric intersection** consists in keeping the most restrictive size constraint in all directions imposed by this set of metrics.

Formally speaking, let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two metric tensors given at a point. The metric tensor $\mathcal{M}_{1\cap 2}$ corresponding to the intersection of $\mathcal{M}_1$ and $\mathcal{M}_2$ is the one prescribing the largest possible size under the constraint that the size in each direction is always smaller than the sizes prescribed by $\mathcal{M}_1$ and $\mathcal{M}_2$. Let us give a geometric interpretation of this operator. Metric tensors are geometrically represented by an ellipse in 2D and an ellipsoid in 3D. But the intersection between two metrics is not directly the intersection between two ellipsoids as their geometric intersection is not an ellipsoid. Therefore, we seek for the largest ellipsoid representing $\mathcal{M}_{1\cap 2}$ included in the geometric intersection of the ellipsoids associated with $\mathcal{M}_1$ and $\mathcal{M}_2$, cf. Figure 1.6, left. The ellipsoid (metric) verifying this property is obtained by using the simultaneous reduction of two metrics.

Simultaneous reduction. The simultaneous reduction enables to find a common basis $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ such that $\mathcal{M}_1$ and $\mathcal{M}_2$ are congruent to a diagonal matrix in this basis, and then to deduce the intersected metric. To do so, the matrix $\mathcal{N} = \mathcal{M}_1^{-1}\mathcal{M}_2$ is introduced. $\mathcal{N}$ is diagonalizable with real-eigenvalues. The normalized eigenvectors of $\mathcal{N}$ denoted by $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ constitute a common diagonalization basis for $\mathcal{M}_1$ and $\mathcal{M}_2$. The entries of the diagonal matrices, that are associated with the metrics $\mathcal{M}_1$ and $\mathcal{M}_2$ in this basis, are obtained with the Rayleigh formula[3]:

$$\lambda_i = \mathbf{e}_i^T \mathcal{M}_1 \mathbf{e}_i \quad \text{and} \quad \mu_i = \mathbf{e}_i^T \mathcal{M}_2 \mathbf{e}_i\,, \quad \text{for} \quad i = 1 \ldots 3\,.$$

[3] λ_i and μ_i are not the eigenvalues of $\mathcal{M}_1$ and $\mathcal{M}_2$. They are spectral values associated with basis $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$.

Let $\mathcal{P} = (\mathbf{e}_1\, \mathbf{e}_2\, \mathbf{e}_3)$ be the matrix the columns of which are the eigenvectors $\{\mathbf{e}_i\}_{i=1,\dots,3}$ of $\mathcal{N}$. $\mathcal{P}$ is invertible as $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ is a basis of $\mathbb{R}^3$. We have:

$$\mathcal{M}_1 = \mathcal{P}^{-T} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \mathcal{P}^{-1} \quad \text{and} \quad \mathcal{M}_2 = \mathcal{P}^{-T} \begin{pmatrix} \mu_1 & 0 & 0 \\ 0 & \mu_2 & 0 \\ 0 & 0 & \mu_3 \end{pmatrix} \mathcal{P}^{-1}.$$

Computing the metric intersection. The resulting intersected metric $\mathcal{M}_{1\cap 2}$ is then analytically given by:

$$\mathcal{M}_{1\cap 2} = \mathcal{M}_1 \cap \mathcal{M}_2 = \mathcal{P}^{-T} \begin{pmatrix} \max(\lambda_1, \mu_1) & 0 & 0 \\ 0 & \max(\lambda_2, \mu_2) & 0 \\ 0 & 0 & \max(\lambda_3, \mu_3) \end{pmatrix} \mathcal{P}^{-1}. \tag{1.8}$$

The ellipsoid associated with $\mathcal{M}_{1\cap 2}$ is the largest ellipsoid included in the geometric intersection region of the ellipsoids associated with $\mathcal{M}_1$ and $\mathcal{M}_2$, the proof is given in [Alauzet 2003b].

Numerically, to compute $\mathcal{M}_{1\cap 2}$, the real-eigenvalues of $\mathcal{N}$ are first evaluated with a Newton algorithm. Then, the eigenvectors of $\mathcal{N}$, which define $\mathcal{P}$, are computed using the algebra notions of image and kernel spaces.

Remark 3. *The intersection operation is not commutative. Consequently, when more than two metrics are intersected, the result depends on the order of intersection. In this case, the resulting intersected metric is not anymore optimal. If, we seek for the largest ellipsoid included in the geometric intersection region of several (> 2) metrics, the John ellipsoid has to be found thanks to an optimization problem [Loseille 2008].*

1.2.7 Metric interpolation

In practice, the metric field is only known discretely at mesh vertices. The definition of an interpolation procedure on metrics is therefore mandatory to be able to compute the metric at any point of the domain. For instance, the computation of the volume of an element using

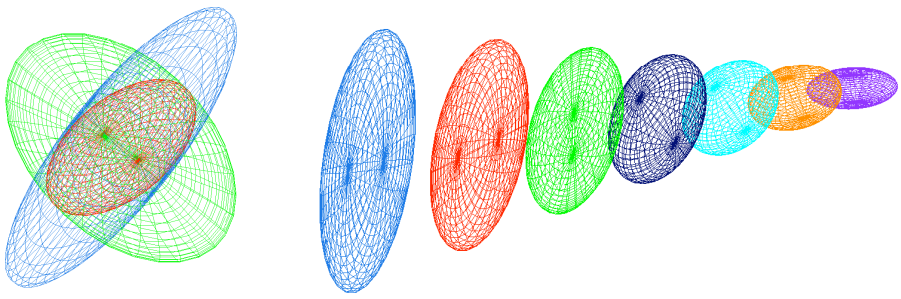


Figure 1.6: *Left, view illustrating the metric intersection procedure with the simultaneous reduction in three dimensions. In red, the resulting metric of the intersection of the blue and green metrics. Right, metric interpolation along a segment where the endpoints metrics are the blue and violet ones.*

quadrature formula with Relation (1.2) requires the computation of some interpolated metrics inside the considered element.

Several interpolation schemes have been proposed in [Alauzet 2003b] which are based on the simultaneous reduction. The main drawback of these approaches is that the interpolation operation is not commutative. Hence, the result depends on the order in which the operations are performed when more than two metrics are involved. Moreover, such interpolation schemes do not satisfy useful properties such as the maximum principle. Consequently, to design an interpolation scheme on these objects, one needs a consistent operational framework. We suggest to consider the log-Euclidean framework introduced in [Arsigny 2006].

Log-Euclidean framework. We first define the notion of metric logarithm and matrix exponential. The **metric logarithm** is defined on the set of metric tensors. For metric tensor $\mathcal{M} = \mathcal{R}\Lambda\mathcal{R}^T$, it is given by:

$$\ln(\mathcal{M}) := \mathcal{R}\ln(\Lambda)\mathcal{R}^T\,,$$

where $\ln(\Lambda) = \mathrm{diag}(\ln(\lambda_i))$. The **matrix exponential** is defined on the set of symmetric matrices. For any symmetric matrix $\mathcal{S} = \mathcal{Q}\,\Xi^T\,\mathcal{Q}$, it is given by:

$$\exp(\mathcal{S}) := \mathcal{Q}\,\exp(\Xi)^T\,\mathcal{Q}\,,$$

where $\exp(\Xi) = \mathrm{diag}(\exp(\xi_i))$. We can now define the **logarithmic addition** $\oplus$ and the **logarithmic scalar multiplication** $\odot$:

$$\begin{aligned} \mathcal{M}_1 \oplus \mathcal{M}_2 &:= \exp\left(\ln(\mathcal{M}_1) + \ln(\mathcal{M}_2)\right) \\ \alpha \odot \mathcal{M} &:= \exp\left(\alpha.\ln(\mathcal{M})\right) = \mathcal{M}^\alpha\,. \end{aligned}$$

The logarithmic addition is commutative and coincides with matrix multiplication whenever the two tensors $\mathcal{M}_1$ and $\mathcal{M}_2$ commute in the matrix sense. The space of metric tensors, supplied with the logarithmic addition $\oplus$ and the logarithmic scalar multiplication $\odot$ is a vector space.

Remark 4. *This framework allows more general computations to be carried out on metric tensors, such as statistical studying or the resolution of PDE's on metric tensors.*

Metric interpolation in the log-Euclidean framework. We propose to use the linear interpolation operator derived from the log-Euclidean framework. Let $(\mathbf{x}_i)_{i=1\ldots k}$ be a set of vertices and $(\mathcal{M}(\mathbf{x}_i))_{i=1\ldots k}$ their associated metrics. Then, for a point $\mathbf{x}$ of the domain such that:

$$\mathbf{x} = \sum_{i=1}^{k} \alpha_i \mathbf{x}_i \quad \text{with} \quad \sum_{i=1}^{k} \alpha_i = 1\,,$$

the interpolated metric is defined by:

$$\mathcal{M}(\mathbf{x}) = \bigoplus_{i=1}^{k} \alpha_i \odot \mathcal{M}(\mathbf{x}_i) = \exp\left(\sum_{i=1}^{k} \alpha_i\, \ln(\mathcal{M}(\mathbf{x}_i))\right)\,. \tag{1.9}$$

This interpolation is commutative, but its bottleneck is to perform k diagonalizations and to request the use of the logarithm and the exponential functions which are CPU consuming.

However, this procedure is essential to define continuously the metric map on the entire domain. Moreover, it has been demonstrated in [Arsigny 2006] that this interpolation preserves the maximum principle, *i.e.,* for an edge $\overrightarrow{PQ}$ with endpoints metrics $\mathcal{M}(P)$ and $\mathcal{M}(Q)$ such that $\det(\mathcal{M}(P)) < \det(\mathcal{M}(Q))$ then we have $\det(\mathcal{M}(P)) < \det(\mathcal{M}(P + t\overrightarrow{PQ})) < \det(\mathcal{M}(Q))$ for all $t \in [0, 1]$.

Accordingly, this metric interpolation enables a continuous metric field to be defined throughout the entire discretized domain. When a metric is required at a point, we determine to which element the point belongs. Then, we apply Relation (1.9), where α_i are the barycentric coordinates of the point with respect to the element. Figure 1.6 illustrates metric interpolation along a segment, for which the initial data are the endpoints metrics.

Remark 5. *The interpolation formulation (1.9) reduces to*

$$\mathcal{M}(\mathbf{x}) = \prod_{i=1}^{k} \mathcal{M}(\mathbf{x}_i)^{\alpha_i},$$

if all the metrics commute. Therefore, an arithmetic mean in the log-Euclidean framework could be interpreted as a geometric mean in the space of metric tensors.

1.3 The continuous mesh theory

Over the past few years, a brand new framework has been developed which enables to handle discrete object like meshes, elements or interpolation operators as if they were continuous entities. The development of this new theory has allowed the resolution of problems involving new type of unknowns such as meshes, problems which were previously intractable due to their complexity. This section is devoted to the description of this new *continuous mesh* theory. It aims at explaining how most of the entities usually manipulated by numericians, which most of the time can be considered as discrete, can nevertheless be fully described from a continuous point of view.

1.3.1 The continuous element model

1.3.1.1 Equivalence class of elements

The first step to build this new framework consists in finding a continuous object able to summarize the discrete information contained in a single element of a given mesh. The notion of *unit element* with respect to a metric field plays a central role in this perspective. Indeed, the following result, proved in [Loseille 2010a], shows that the set of all the elements which are unit with respect to a given metric tensor forms an equivalence class of elements.

Indeed, let $\mathcal{M}$ be a metric tensor. $\mathcal{M}$ is first assumed equal to identity tensor $\mathcal{I}_n$. Then, K_0, the regular element of $\mathbb{R}^n$ is a unit element with respect to $\mathcal{M}$. The set of all the other unit elements, denoted $\mathcal{K}$, is obtained by rotating K_0.
Now, if $\mathcal{M} \neq \mathcal{I}_n$, a unit element with respect to $\mathcal{M}$ is the image of K_0 by application $\mathcal{M}^{-\frac{1}{2}}$, see Section 1.2.1.1. Similarly, all the other elements which are unit with respect to $\mathcal{M}$ are obtained

by taking the images of the elements of $\mathcal{K}$ through $\mathcal{M}^{-\frac{1}{2}}$.
Conversely, any arbitrary element K can be put in an equivalence class and can therefore be associated with a unique metric. The sought metric $\mathcal{M}$ is the one for which K is unit and can be explicitly determined by solving a linear system having as unknowns the coefficient of tensor $\mathcal{M}$.

As a consequence, any discrete element can be viewed as a discrete representative of some equivalence class formed by all the unit elements of some metric $\mathcal{M}$. Using a minor abuse of notation, this equivalence class will be itself noted $\mathcal{M}$. Figure 1.7 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".**

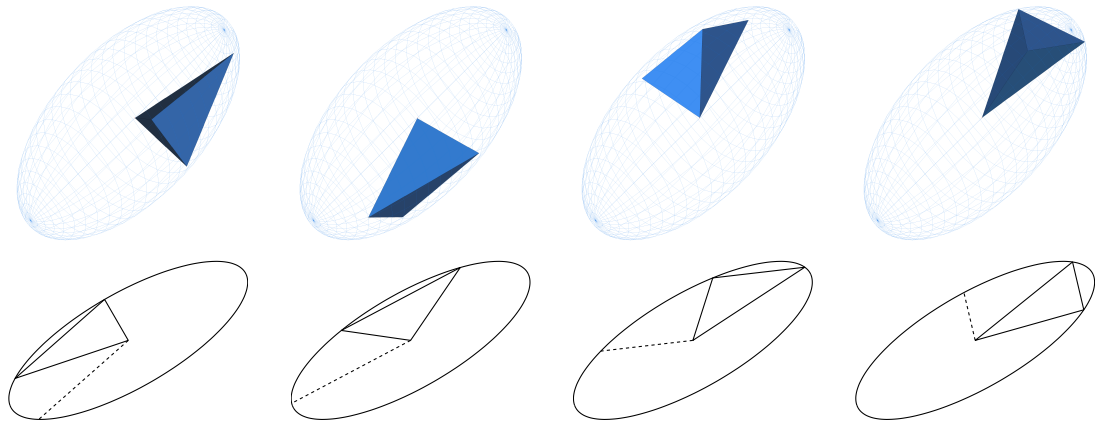


Figure 1.7: Several unit elements with respect to a continuous element in 2D and 3D.

1.3.1.2 Geometric invariants

All the discrete representatives of a given equivalence class $\mathcal{M}$ share some common properties, which can be described using only metric tensor $\mathcal{M}$. These properties actually connect the geometric properties of the elements which are unit with respect to $\mathcal{M}$ to the linear algebra properties of $\mathcal{M}$.

Proposition 1 (Geometric invariants)**.** *Let $\mathcal{M}$ be a metric tensor and K be a unit element with respect to $\mathcal{M}$. We denote by $(\mathbf{e}_i)_{1\leq i\leq n}$ its edges list and $|K|$ its Euclidean volume. Then, the following invariants hold:*

- *standard invariants:*

$$\forall\,(\mathbf{e}_i,\mathbf{e}_j),\qquad \begin{cases} \mathbf{e}_i^T\,\mathcal{M}\,\mathbf{e}_i = 1, \\ 2\,\mathbf{e}_i^T\,\mathcal{M}\,\mathbf{e}_j + 1 = 0 \;\; \textit{if } i \neq j. \end{cases} \tag{1.10}$$

- *invariant related to the Euclidean volume $|K|$:*

$$|K| = \frac{\sqrt{3}}{4}\,\det(\mathcal{M}^{-\frac{1}{2}}) \;\; \textit{in } 2D \quad \textit{and} \quad |K| = \frac{\sqrt{2}}{12}\,\det(\mathcal{M}^{-\frac{1}{2}}) \;\; \textit{in } 3D. \tag{1.11}$$

- *invariant related to the square length of the edges for any symmetric matrix H:*

$$\begin{aligned}\sum_{i=1}^{3} \mathbf{e}_i^T H \mathbf{e}_i &= \frac{3}{2}\text{trace}(\mathcal{M}^{-\frac{1}{2}} H \mathcal{M}^{-\frac{1}{2}}) \text{ in } 2D,\\ \sum_{i=1}^{6} \mathbf{e}_i^T H \mathbf{e}_i &= 2\,\text{trace}(\mathcal{M}^{-\frac{1}{2}} H \mathcal{M}^{-\frac{1}{2}}) \text{ in } 3D.\end{aligned} \tag{1.12}$$

1.3.2 The continuous mesh model

Definition. In this subsection, $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ represents a given metric tensor field. As for the local duality, we would like to define equivalence classes of meshes, each class being represented by a single continuous object. To this aim, the notion of unit mesh with respect to metric field $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is used. **$\mathbf{M}$ denotes the class of equivalence of all the meshes which are quasi-unit with respect to $\mathbf{M}$ and is called "continuous mesh".**

Another decomposition of $\mathcal{M}(\mathbf{x})$. We have already seen that at each point $P \in \Omega$ of coordinates vector $\mathbf{x}$, $\mathcal{M}(\mathbf{x})$ admits a spectral decomposition given by Relation (1.1). In practice, another decomposition can be used that points out the local characteristics of $\mathbf{M}$. This decomposition is given by the following proposition.

Proposition 2. *$\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ locally writes:*

$$\mathcal{M}(\mathbf{x}) = d^{\frac{2}{n}}(\mathbf{x})\,\mathcal{R}(\mathbf{x})\,diag\left(r_1^{-\frac{2}{n}}(\mathbf{x}), \ldots, r_n^{-\frac{2}{n}}(\mathbf{x})\right)\mathcal{R}(\mathbf{x})^T, \tag{1.13}$$

where[4]

- *the continuous mesh local density d is equal to: $d = (h_1 \ldots h_n)^{-1} = (\lambda_1, \ldots, \lambda_n)^{\frac{1}{2}}$,*
- *the continuous mesh local anisotropic quotients κ_i are equal to: $\kappa_i = \frac{h_i^n}{\prod_{i=1}^{n} h_i}$.*

Complexity. The complexity $\mathcal{C}$ of a continuous mesh $\mathbf{M}$ is the continuous equivalent of the number of vertices N_v of a discrete mesh and is defined by:

$$\mathcal{C}(\mathbf{M}) = \int_\Omega d(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_\Omega \sqrt{\det(\mathcal{M}(\mathbf{x}))}\,\mathrm{d}\mathbf{x}. \tag{1.14}$$

This real-value parameter is useful to quantify the global level of accuracy of $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\mathbf{\Omega}}$.

Embedded continuous meshes. Two continuous meshes, saying $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\mathbf{\Omega}}$ and $(\mathcal{N}(\mathbf{x}))_{\mathbf{x}\in\mathbf{\Omega}}$, are embedded if a constant c exists such that:

$$\forall \mathbf{x} \in \Omega, \quad \mathcal{N}(\mathbf{x}) = c\,\mathcal{M}(\mathbf{x})\,. \tag{1.15}$$

Conversely, from $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\mathbf{\Omega}}$, we can deduce $\mathbf{N} = (\mathcal{N}(\mathbf{x}))_{\mathbf{x}\in\mathbf{\Omega}}$ of complexity N having the same anisotropic properties (anisotropic orientations and ratios) by considering:

$$\mathcal{N}(\mathbf{x}) = \left(\frac{N}{\mathcal{C}(\mathbf{M})}\right)^{\frac{2}{n}} \mathcal{M}(\mathbf{x}).$$

[4] Note that in two dimensions, we find back the usual definition $\kappa_1 = \frac{h_1}{h_2} = \frac{1}{\kappa_2}$

In the context of error estimation, this notion enables to perform *convergence order studies* with respect to an increasing complexity N.

Uniform continuous refinement. In the continuous framework, uniform refinement consists in dividing by a constant factor c the length of each edge of a uniform continuous mesh $\mathcal{M}(\mathbf{x})$. This writes:

$$\mathcal{M}_i = 4^i \, \mathrm{diag}(\frac{1}{h_1^2}, \ldots, \frac{1}{h_n^2})$$

where i is the level of refinement. $(\mathcal{M}_i)_{i=1\ldots k}$ defines a sequence of embedded continuous meshes (or embedded Riemannian spaces).

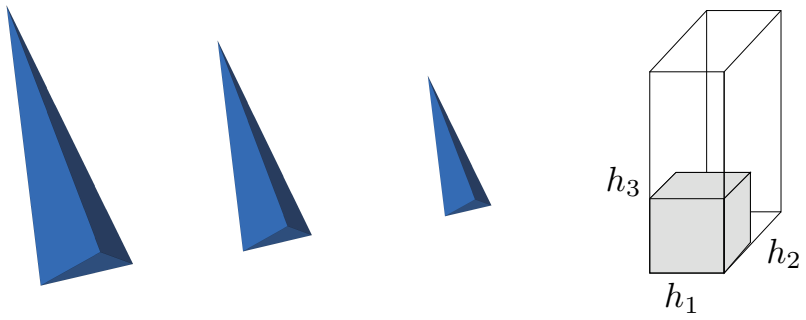


Figure 1.8: Left, from left to right: different unit elements with increasing density. Right, the geometric interpretation of anisotropic quotients as quotients of parallelepipeds volumes.

1.3.3 The continuous linear interpolate model

In the previous section, a continuous framework has been introduced to model elements and meshes. Now, we aim at applying this framework in the context of error estimation. However, as our intent is to propose a fully discrete-continuous duality, it is not enough to derive only the optimal mesh arising from an interpolation error bound as in classical studies on interpolation error [Castro-Díaz 1997, Frey 2005, Huang 2005]. Instead, we want to evaluate the interpolation error for any functions on any continuous meshes without imposing some optimality conditions as alignment, equi-distribution, ...

Let $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ be a continuous mesh of a domain Ω and let u be a non-linear scalar or vectorial function which is assumed to be only twice continuously differentiable. We seek a well-posed definition of the continuous linear interpolation error $\|u - \pi_{\mathcal{M}} u\|_{\mathbf{L}^1(\Omega)}$ related to a continuous mesh $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ which implies a well-posed definition of a linear continuous interpolate $\pi_{\mathcal{M}} u$. More precisely, we would like the continuous linear interpolation error to be a reliable mathematical model of $\|u - \Pi_h u\|_{\mathbf{L}^1(\Omega_h)}$ where Π_h is defined by a mesh $\mathcal{H}$ of a discretized domain Ω_h which is a unit mesh with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$. *In fine*, this means that considering $\|u - \pi_{\mathcal{M}} u\|_{\mathbf{L}^1(\Omega)}$ is equivalent to considering $\|u - \Pi_h u\|_{\mathbf{L}^1(\Omega_h)}$.

1.3.3.1 Linear interpolation on a continuous element

Let $\mathcal{M}$ be a continuous element and u be a quadratic positive function. We study the interpolation error for the class of all unit discrete elements with respect to $\mathcal{M}$. The main result is:

Theorem 1. *For all unit elements K with respect to $\mathcal{M}$, the interpolation error of u in $\mathbf{L}^1$ norm does not depend on the element shape and is only a function of the Hessian H_u of u and of continuous element $\mathcal{M}$.*

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

$$\|u - \Pi_h u\|_{\mathbf{L}^1(K)} = \frac{\sqrt{2}}{240} \det(\mathcal{M}^{-\frac{1}{2}}) \operatorname{trace}(\mathcal{M}^{-\frac{1}{2}} H_u \mathcal{M}^{-\frac{1}{2}}). \tag{1.16}$$

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

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

We note the strong analogy with classical interpolation error estimate for Lagrange interpolation [Ciarlet 1978]:

- The term $\det \mathcal{M}^{-\frac{1}{2}}$ stands for the Jacobian of the affine transformation from the reference element $\hat{K}$ onto the current element K. In our continuous framework, it is the Jacobian of the affine mapping between the reference continuous element unit ball $\mathcal{B}_{I_3}$ onto the current continuous element unit ball $\mathcal{B}_{\mathcal{M}}$.
- The term $\operatorname{trace}\left(\mathcal{M}^{-\frac{1}{2}} H_u \mathcal{M}^{-\frac{1}{2}}\right)$ stands for the semi-norm involved in classical error estimates. Generally, this semi-norm contains the anisotropic behavior of the estimate. In the continuous framework, the trace-term gives the alignment correlation between the principal directions of Hessian H_u and the principal directions of metric $\mathcal{M}$.

Relation (1.16) shows that the infinite set of discrete elements that are unit for a given continuous element $\mathcal{M}$ achieves the same interpolation error, and moreover, shows that this interpolation error is only expressed with continuous quantities: the continuous element $\mathcal{M}$ and the Hessian of function u. Consequently, Theorem 1 points out that the metric alone contains enough information to describe completely the linear interpolation error in $\mathbf{L}^1$ norm. In other words, this theorem confirms that the use of metric-based mesh adaptation is particularly well suited to control anisotropically the interpolation error.

1.3.3.2 Continuous linear interpolate $\pi_{\mathcal{M}}$

The main difficulty in defining the continuous linear interpolate is to connect a discrete error computed on an element to a local continuous error that is defined point-wise. Indeed, the discrete interpolation error in norm $\mathbf{L}^1$ is integrated on the element K. On the contrary, a continuous mesh is a function $\mathbf{x} \mapsto \mathcal{M}(\mathbf{x})$ defined at each point $\mathbf{x}$ of Ω. In [Loseille 2010a], such a continuous linear interpolate is defined exhibiting very interesting properties.

Theorem 2 (Discrete-continuous equivalence)**.** *Let u be a twice continuously differentiable function of domain Ω and $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ be a continuous mesh of Ω. We denote by $u_Q(\mathbf{a};\cdot)$ the quadratic approximation of u at point $\mathbf{a}$ which is defined in the vicinity of $\mathbf{a}$ as the truncated second order Taylor expansion of u:*

$$\forall \mathbf{x} \in \mathcal{V}(\mathbf{a}), \quad u_Q(\mathbf{a};\mathbf{x}) = u(\mathbf{a}) + \nabla u(\mathbf{a})(\mathbf{x}-\mathbf{a}) + \frac{1}{2}\langle(\mathbf{x}-\mathbf{a}),\, H(\mathbf{a})(\mathbf{x}-\mathbf{a})\rangle.$$

When no confusion is possible, notation $u_Q(\mathbf{a};\mathbf{x})$ is replaced by $u_Q(\mathbf{x})$. Then, there exists a unique continuous linear interpolate function $\pi_{\mathcal{M}}$ such that:

$$\forall \mathbf{a} \in \Omega, \quad |u - \pi_{\mathcal{M}} u|(\mathbf{a}) = 2\,\frac{\|u_Q - \Pi_h u_Q\|_{\mathbf{L}^1(K)}}{|K|},$$

for every K unit element with respect to $\mathcal{M}(\mathbf{a})$.
Moreover, the following piecewise continuous linear interpolation estimate holds in 3D:

$$\begin{aligned} \forall \mathbf{a} \in \Omega, \quad |u - \pi_{\mathcal{M}} u|(\mathbf{a}) &= \frac{1}{10}\,\mathrm{trace}\big(\mathcal{M}(\mathbf{a})^{-\frac{1}{2}}\,|H_u(\mathbf{a})|\,\mathcal{M}(\mathbf{a})^{-\frac{1}{2}}\big) \\ &= \frac{1}{10}\,\Big(\, d(\mathbf{a})^{-\frac{2}{3}} \sum_{i=1}^{3} r_i(\mathbf{a})^{\frac{2}{3}}\ {\mathbf{v}_i}^T(\mathbf{a})\,|H_u(\mathbf{a})|\,\mathbf{v}_i(\mathbf{a})\,\Big). \end{aligned} \tag{1.17}$$

In 2D, the estimate is:

$$\begin{aligned} \forall \mathbf{a} \in \Omega, \quad |u - \pi_{\mathcal{M}} u|(\mathbf{a}) &= \frac{1}{8}\,\mathrm{trace}\big(\mathcal{M}(\mathbf{a})^{-\frac{1}{2}}\,|H_u(\mathbf{a})|\,\mathcal{M}(\mathbf{a})^{-\frac{1}{2}}\big) \\ &= \frac{1}{8}\,\Big(\, d(\mathbf{a})^{-1} \sum_{i=1}^{2} r_i(\mathbf{a})\ {\mathbf{v}_i}^T(\mathbf{a})\,|H_u(\mathbf{a})|\,\mathbf{v}_i(\mathbf{a})\,\Big). \end{aligned} \tag{1.18}$$

This result shows that the continuous point-wise linear interpolation can be decomposed into the product of two terms:

- a first term that control the accuracy, this density term is directly connected to the size of the continuous element,
- a second term that measures alignment deviation between the continuous element orientation and the anisotropy features of the function u.

1.3.3.3 Global interpolation error in $\mathbf{L}^1$-norm

The above results demonstrate that both the *local* interpolation error $||u - \Pi_h u||_{\mathbf{L}^1(K)}$ and the linear interpolate Π_h have continuous counterparts. It remains to define the *global* interpolation error in $\mathbf{L}^1$ norm. From a practical point of view, the following analogy is used:

Definition 3. *Given a 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:*

$$\|u - \Pi_h u\|_{\mathbf{L}^1(\Omega_h)} = \sum_{K\in\mathcal{H}} \|u - \Pi_h u\|_{\mathbf{L}^1(K)}. \tag{1.19}$$

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

$$\|u - \pi_{\mathcal{M}} u\|_{\mathbf{L}^1(\Omega)} = \int_{\Omega} |u - \pi_{\mathcal{M}} u|(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{1.20}$$

Note that there is no global guarantee on the continuous interpolation error reliability given by Relation (1.20). For instance, there is no *a priori* relationship between (1.19) and (1.20). The only guarantee is the local equivalence given by Theorem 2. However, the local guarantee becomes global when the mesh is unit with respect to a constant metric tensor (this does not necessary implied that the mesh is uniform) and when the function is quadratic. In this specific case, by neglecting error due to the boundary discretization, we have the equality:

$$2\,\|u - \Pi_h u\|_{\mathbf{L}^1(\Omega_h)} = \|u - \pi_{\mathcal{M}} u\|_{\mathbf{L}^1(\Omega)},$$

for all unit meshes $\mathcal{H}$ with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$. Several analytical and numerical examples in two and three dimensions are given in [Loseille 2010b] which shows that:

- this model is accurate and the equivalence (1.19)≈(1.20) is observed even for non quadratic functions and non-constant continuous meshes,

- 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 1.2.2 ensures reliable numerical results.

These examples reveal that the interpolation error can be computed continuously without any discrete support. Discrepancies between continuous and discrete interpolation errors depends on the mesh generator used and the difficulty to generate the desired unit mesh.

1.3.3.4 Synthetic array

To conclude this section, a synthetic array showing the equivalence between discrete entities and their continuous representation is given:

DISCRETE	CONTINUOUS
Element K	Metric tensor $\mathcal{M}$
Element volume $\lvert K\rvert$	$d^{-1} = \sqrt{\det(\mathcal{M}^{-1})}$
Element perimeter $\lvert K\rvert$	$\text{trace}\left(\mathcal{M}(\mathbf{x})^{-1}\right)$
Mesh $\mathcal{H}$ of Ω_h	Riemannian metric space $\mathbf{M}(\mathbf{x}) = \left(\mathcal{M}(\mathbf{x})\right)_{\mathbf{x}\in\Omega}$
Number of vertices N_v	Complexity $\mathcal{C}(\mathcal{M}) = \int_\Omega d(\mathbf{x})\,\mathrm{d}\mathbf{x}$
$\mathbf{P}^1$ interpolate Π_h	$\mathbf{P}^1$-continuous interpolate $\pi_\mathcal{M}$
Local element-wise interpolation error $\lvert\lvert u - \Pi_h u\rvert\rvert_K$	Local point-wise interpolation error $e(\mathbf{x}) = \left(u - \pi_\mathcal{M} u\right)(\mathbf{x})$
Global interpolation error $\sum_{K\in\mathcal{H}} \lvert\lvert u - \Pi_h u\rvert\rvert_K$	Global interpolation error $\int_{\mathbf{x}\in\Omega} e(\mathbf{x})\,\mathrm{d}\mathbf{x}$

1.4 Multi-scale mesh adaptation

The main advance enabled by the continuous mesh formalism is to provide us with new *continuous representations* of discrete objects such as meshes, elements and interpolation operators. A large variety of mathematical tools then become available for these continuous representation while they were not when directly dealing with discrete entities. Notably, it enables to **work with metric fields instead of working with meshes**, which is far more convenient insofar as new operations like integration, derivation, calculus of variations are available for metric fields and are ill-defined when dealing directly with discrete meshes.
This section illustrates the powerfulness of this new framework when attempting to solve the complex problem of finding the optimal mesh minimizing the interpolation error in $\mathbf{L}^p$-norm.

1.4.1 Global optimization problem

The problem of finding the optimal mesh minimizing the interpolation error of a known function u in $\mathbf{L}^p$-norm *a priori* writes:

$$\text{Find } \mathcal{H}_{opt} \text{ having } N_v \text{ vertices such that } \mathcal{H}_{opt} = \underset{\mathcal{H}}{\text{argmin}}\, ||u - \Pi_h u||_{\mathcal{H},\mathbf{L}^p(\Omega_h)}$$

Stated like this, this problem is purely and simply intractable. Indeed, as the unknown of the problem is a mesh, *i.e.* a set of vertices associated with a topology. Unknown $\mathcal{H}$ hides a very high number of degrees of freedom, which makes the resolution impossible. Moreover, the problem is ill-posed as several optimal meshes can be found for a single function u
On the contrary, if one accepts to reformulate the problem using the new continuous mesh theory, it is then possible to set the well-posed global optimization problem of finding the optimal continuous mesh minimizing the continuous interpolation error in $\mathbf{L}^p$ norm:

$$\text{Find } \mathbf{M}_{\mathbf{L}^p} = \underset{\mathbf{M}}{\operatorname{argmin}}\, E_p(\mathbf{M}) \;=\; \left(\int_\Omega (u - \pi_{\mathcal{M}} u)^p\right)^{\frac{1}{p}}, \tag{1.21}$$

under constraint

$$\mathcal{C}(\mathbf{M}) = \int_\Omega d = N.$$

The constraint on the complexity is added to avoid the trivial solution where all $(h_i)_{i=1,3}$ are zero which provides a null error.

This minimization problem is solved *analytically* using standard calculus of variations techniques which are now available as we have chosen to work with continuous mesh $\mathbf{M}$. Formulation (1.13) of metric tensor $\mathcal{M}$ is also of great help for the resolution of this problem.

Theorem 3. *Let u be a twice continuously differentiable function defined on $\Omega \subset \mathbb{R}^n$, H_u its Hessian, the optimal continuous mesh $\mathbf{M}_{\mathbf{L}^p}(u)$ minimizing Problem (1.21) reads locally:*

$$\mathcal{M}_{\mathbf{L}^p} = D_{\mathbf{L}^p}\, \det(|H_u|)^{\frac{-1}{2p+n}}\, |H_u|\,, \qquad \textit{with} \qquad D_{\mathbf{L}^p} = N^{\frac{2}{n}} \left(\int_\Omega \det(|H_u|)^{\frac{p}{2p+n}}\right)^{-\frac{2}{n}}. \tag{1.22}$$

It verifies the following properties:

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

$$\|u - \pi_{\mathcal{M}_{\mathbf{L}^p}} u\|_{\mathbf{L}^p(\Omega)} = n\, N^{-\frac{2}{n}} \left(\int_\Omega \det\left(|H_u|\right)^{\frac{p}{2p+n}}\right)^{\frac{2p+n}{np}}. \tag{1.23}$$

The outstanding character of this theorem as compared to existing results lies on the one hand in the *unicity* of the optimal continuous mesh and on the other hand on the *global* nature of this optimum.

Remark 6 ($\mathbf{L}^\infty$-norm)**.** *Passing to the limit for $p \to \infty$ leads to the classical metric that controls the interpolation error in $\mathbf{L}^\infty$ as used in [Castro-Díaz 1997, Frey 2005].*

1.4.2 Optimal metric for a numerical approximation u_h

According to Theorem 3, optimal Metric (1.22) is valid only for $u \in \mathcal{C}^2$. In this section, we describe how the interpolation theory is applied when only u_h, represented by a piecewise linear function, is known. In this particular case, the interpolation error estimate is not applied directly to u nor u_h, but to a smooth representation of u_h obtained thanks to operator R_h, see Section 1.2.4.

Let $\bar{V}_h^k$ be the space of piecewise polynomials of degree k and V_h^k be the space of continuous piecewise polynomials of degree k associated with a given mesh $\mathcal{H}$ of domain Ω_h. R_h is a reconstruction operator applied to numerical approximation u_h. This reconstruction operator can be either a recovery process, a hierarchical basis, or an operator connected to an a posteriori estimate. 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 triangular inequality, we deduce:

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

If the reconstruction operator R_h satisifies:

$$\Pi_h R_h \phi_h = \phi_h \,, \quad \forall \phi_h \in V_h^1 \,,$$

the approximation error of the solution can be bounded by the interpolation error of the reconstructed function $R_h u_h$:

$$\|u - u_h\| \leq \frac{1}{1-\alpha} \|R_h u_h - \Pi_h R_h u_h\| \,. \tag{1.24}$$

From Theorem 3, we can exhibit the following upper bound of the approximation error:

$$\|u - u_h\| \leq \frac{6\, N^{-\frac{2}{3}}}{1-\alpha} \left(\int_\Omega \det \left(|H_{R_h u_h}| \right)^{\frac{p}{2p+3}} \right)^{\frac{2p+3}{3p}} \,,$$

and $H_{R_h u_h}$ is computed with one of the three techniques described in Section 1.2.4 ($\mathbf{L}^2$-projection, Green formula or least square).

Remark 7. *It is important to note that $\mathcal{M}_{\mathbf{L}^p}$ defined by Relation (1.22) applied to $R_h u_h$ does not allow to generate an optimal adapted mesh to control the approximation error $\|u - u_h\|$. If all the above assumptions are verified, this analysis states that such generated adapted meshes controls the approximation error. But only an upper bound is obtained and no lower bound. As this analysis is likely to be applied to the compressible Euler or Navier-Stokes systems, for which no anisotropic error estimate is available so far, this is results can however be considered as satisfactory.*

1.4.3 Adaptive strategy using the optimal $\mathbf{L}^p$ metric

Let us choose $p \in [1, +\infty[$. Theorem 3 provides an analytical expression of the optimal continuous mesh controlling the linear interpolation error in $\mathbf{L}^p$-norm for a fixed complexity. In this subsection, the advantages induced by using the optimal continuous mesh $\mathbf{M}_{\mathbf{L}^p}$ in the adaptation strategy, described in Figure 1.4, are enumerated.

1.4.3.1 The "h_{min}" parameter

One obvious asset of this new adaptive strategy is that it automatically handles the "h_{min}" parameter setting. Indeed, for adaptation strategies that only enable a control of the local interpolation error, the resulting metric can prescribe the construction of edges of very small length, notably in shocks. This phenomenon can lead to the generation of meshes containing an excessive number of vertices. Some elements can also have a very small altitude[5], which can drastically diminish the efficiency of the numerical solver, especially when an explicit temporal scheme is used.
To prevent this kind of pathological behavior, a usual technique consists in artificially enforcing a minimal edge size criterion. Practically, the metric eigenvalues which are too high are replaced *a posteriori* by $\lambda_{max} = 1/h_{min}^2$.
The problem is that the choice of h_{min} is left to the user, which is in conflict with our automaticity objective. Moreover, this *a posteriori* truncation does not enable to guarantee the control of the interpolation error anymore.
Thanks to the resolution of a *global* optimization problem, this h_{min}-parameter is implicitly prescribed through the prescription of the desired complexity N or desired error E. The optimal metric guarantees that h_{min} in never zero. However, in practice, a value for h_{min} can be prescribed to avoid too small elements (especially for unsteady adaptation).

1.4.3.2 Recovering second-order accuracy

It is shown in [Loseille 2010b] that the optimal $\mathbf{L}^p$ metric enables to recover the expected second order accuracy when performing a convergence study on smooth analytical functions. Indeed,

Theorem 4 (Asymptotic convergence)**.** *Let $(\mathbf{M}^N_{\mathbf{L}^p}(u))_{N=1...\infty}$ be a family of optimal continuous meshes parametrized by increasing complexity N and having the same orientation and anisotropic quotients (i.e. embedded continuous meshes in the sense of Definition 1.15). Let u be a twice continuously differentiable function. Then, the asymptotic order of convergence of the interpolation error on u in $\mathbf{L}^p$-norm satisfies:*

$$\|u - \pi_{\mathcal{M}^N_{\mathbf{L}^p}} u\|_{\mathbf{L}^p(\Omega)} \leq \frac{Cst}{N^{2/n}}. \tag{1.25}$$

This theorem states that, in the case of an adaptation to *a smooth analytical function*, a second-order of convergence is obtained for the error, which is the least we could demand. Indeed, a second-order convergence is also reached when simply using a uniform refinement strategy[6].

Lower error. The convergence curve obtained when plotting $\log \sum_{K\in\mathcal{H}} ||u - \Pi_h u||_{\mathbf{L}^p(\Omega_h)}$ v.s N_v is expected to be a straight line of slope 2, at least asymptotically. This is also the case when the sequence of embedded meshes is built by successive uniform refinements. However, one big difference is that in our case, as the successive adapted meshes obtained for increased values of the complexity N are all optimal. Consequently, the convergence curve obtained with

[5] The altitude of an element is the smallest of its $n+1$ altitudes.
[6] A uniform refinement strategy consists in recursively splitting the edges in two new edges of equal lengths

this approach will be lower than other second-order methods. In other words, constant Cst is optimal, which is not the case for example with a uniform refinement strategy.

Better behavior on numerical solutions involving singularities. As explained in the introduction, our main objective is to perform CFD simulations involving flow singularities in the most accurate possible manner. To reach this goal, specific methods such as MUSCL type methods are used to locally increase the accuracy of the numerical solution. However, when shocks are present, these techniques generate oscillations of the numerical solution which are incompatible with the expected physical features. In these regions, the order of accuracy is thus intentionally decreased back to one thanks to slope limiters, see Section 5.2. This numerical barrier leads to a global order of convergence of the interpolation error which is strictly lower than two when a uniform refinement strategy is applied.
However, although an accuracy limit has been reached by the resolution scheme, it is still possible to act at the mesh level to recover the desired order of accuracy. Indeed, in [Coudière 2002], it has been pointed out that the convergence rate of the approximation of an *analytical shock*, *i.e.* a discontinuous analytical function, can be increased by applying an *adaptive* refinement strategy, to be opposed with a uniform refinement strategy. However, this study has also emphasized that if only isotropic strategies are considered, the attainable order of convergence for the global interpolation error in $\mathbf{L}^p$ norm is intrinsically limited:

Lemma 1 (Isotropic adaptation barrier)**.** *Let u a scalar function involving discontinuities, which is smooth outside the discontinuity area. If a $\mathbf{P}^1$ interpolation method is applied to u, then the convergence order α of the interpolation error in $\mathbf{L}^p$-norm when the mesh is refined with an adaptive isotropic strategy*[7] *cannot exceed:*

$$\alpha \leq \frac{1}{p}\frac{n}{n-1}\,.$$

For instance, in three dimensions and for $p = 1$, the maximal order that can be legitimately expected using a purely isotropic strategy equals $1.5 < 2$. Consequently, *anisotropy appears as a necessary condition* to build a mesh adaptation strategy which is consistent with an expected order of accuracy of 2.
However, anisotropy is not a sufficient condition to recover the second-order convergence, which means that some anisotropic adaptive methods are more efficient than others as regards their ability to take advantage from the anisotropic features of the solution and to collect the information from the adaptation variable when dealing with numerical solutions. Incidentally and to our knowledge, very few convergence studies have been done in a mesh adaptation framework, which makes the comparison of metric-based multi-scale anisotropic mesh adaptation with other methodologies very difficult. However, second-order accuracy has been *numerically* observed on analytical examples as well as on complex CFD computations. Moreover, the method presented here has already exhibited other interesting behaviors on two- and three-dimensional computations, see [Loseille 2007, Loseille 2010b, Alauzet 2010b]:

- *Early convergence.* The asymptotic order of convergence is reached very soon.

[7] an adaptive isotropic strategy can generate elements of very different sizes, which is not the case when using a uniform refinement strategy

- *Super-convergence.* Some anisotropic adaptive strategies are more efficient than others to extract the relevant information from the numerical solution [Coudière 2002]. As a result, the convergence plots obtained with such methods will be able to reach in certain cases a convergence rate higher than 2. For instance, it can be the case when the numerical solution is almost constant in one direction. This phenomenon is called *super-convergence*.

1.4.3.3 Catching different scales of the solution

Another interest of the optimal metric in $\mathbf{L}^p$-norm lies in its ability to catch physical phenomena of the solution which are of different scales. To illustrate this, we copy here the analytical example given in [Loseille 2010b]. We consider a function f_1 which is a smooth function involving variations of small and large amplitudes. The function is defined as follows:

$$f_1(x,y) = \begin{cases} 0.01\sin(50xy) & \text{if} \quad xy \leq \dfrac{\pi}{50}, \\ \sin(50xy) & \text{else if} \quad xy \leq 2\dfrac{\pi}{50}, \\ 0.01\sin(50xy) & \text{elsewhere.} \end{cases}$$

This function is composed of variations having a unit amplitude along with small variations having an amplitude of 0.01. This feature is illustrated in Figure 1.9 (top left) where a cut through the line $y = 0$ is depicted.

The mesh adaptation process based on the control of the interpolation error is analyzed for the $\mathbf{L}^1$, $\mathbf{L}^2$ and $\mathbf{L}^4$ norms. Figure 1.9 shows adapted meshes composed of almost 7 000 vertices for each norm. We observe that the small amplitude waves regions are better captured when using a $\mathbf{L}^p$ norm with a lower p ($\mathbf{L}^1$ is the best) whereas the $\mathbf{L}^4$ norm ignores small amplitudes regions and clearly more refined large amplitudes areas. This behavior is due to the term $\det |H_u|^{\frac{-1}{2p+n}}$ in Relation (1.22) which gives more sensitivity to lower p norm. It illustrates that controlling the interpolation error in $\mathbf{L}^p$ norm with a small value of p enable to capture all the scales of the solution.

1.4.4 Handling degenerated cases

In Section 1.4.3.1, we have seen that degenerated case ($\lambda \to +\infty \Longleftrightarrow h \to 0$) was automatically handled thanks to a global optimization strategy enforcing a prescribed complexity constraint $\mathcal{C} = N$. Unfortunately, this equality constraint does not enable to get rid of the other degenerated case ($\lambda \to 0 \Longleftrightarrow h \to +\infty$). An *a posteriori* correction is therefore performed on the metric to eliminate this pathological feature.

Notations. In this section, eigenvalues notations are temporarily changed to better distinguish between the different sets of eigenvalues at stake:

- $(\gamma_1, \ldots, \gamma_n)$ are the eigenvalues of the *pure Hessian* metric $|H_u|$, *i.e.*

$$\gamma_l = \mathbf{r}_l^T \, |H_u| \, \mathbf{r}_l \, ,$$

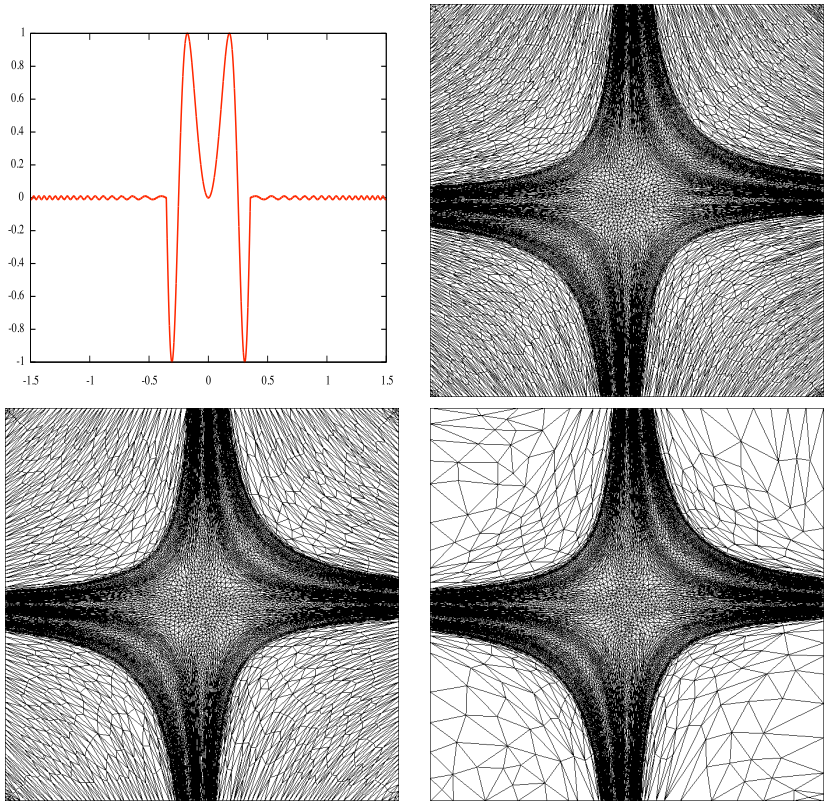


Figure 1.9: Top left, representation of function f_1 along the cut line $y = 0$. Optimal adapted meshes for norms $\mathbf{L}^1$ (top right), $\mathbf{L}^2$ (bottom left) and $\mathbf{L}^4$ (bottom right). Each mesh is composed of about 7 000 vertices.

- $(\lambda_1, \ldots, \lambda_n)$ are the eigenvalues of the *locally normalized Hessian* metric :

$$\tilde{\mathcal{M}} = \det\left(|H_u|\right)^{-\frac{1}{2p+n}} |H_u|\,,$$

- $(m_1, \ldots, m_n)$ are the eigenvalues of the *complete metric* $\mathcal{M}$, *i.e.* the locally normalized Hessian plus the global normalization term:

$$\mathcal{M} = \left(\frac{N}{\displaystyle\int_\Omega \sqrt{\det \tilde{\mathcal{M}}}\,\mathrm{d}\mathbf{x}}\right)^{\frac{2}{n}} \tilde{\mathcal{M}}\,,$$

- $h_l = \dfrac{1}{\sqrt{m_l}}$ is the mesh size prescribed by complete metric $\mathcal{M}$ in direction $\mathbf{r}_l$
- h_{max} is the maximal size for the edges of the mesh prescribed by the user,
- $m_{thres} = \dfrac{1}{h_{max}^2}$ is the threshold for complete metric eigenvalues.

1.4.4.1 Rough truncation

The first solution consists in replacing all the eigenvalues falling below a prescribed minimal threshold $m_{thres} = 1/h_{max}^2$ by the value of this threshold. Nevertheless, such a brutal cut yields some damages on the metric and on the control of the error. These outcomes are detailed below.

Violation of the complexity constraint. Suppose that before truncation, the global normalization constant has been adjusted so that prescribed complexity N is reached. If the truncation algorithm settles for bringing every size exceeding h_{max} back to h_{max}, it is quiet clear according to Formula (1.14) that this will increase the complexity of the metric. Thus, the complexity constraint will not be satisfied anymore after the truncation algorithm.

Loss of local and global optimality. Imagine that at one vertex, one of the Hessian eigenvalues γ_i, with $i \in [\![1, n]\!]$, is very small, close to 0. Actually, this situation is frequent, it happens as soon as the function is almost linear in one direction. Because of the local normalization term $\det(|H_u|)^{\frac{-1}{2p+n}}$ appearing in Formula (1.22), the eigenvalues of the optimal metric $(m_j)_{j \in [\![1,n]\!], j \neq i}$ associated with the other directions become really big as compared to the one associated with direction i. Indeed, if γ_i is the only eigenvalue close to 0:

$$m_i = \left(\prod_{l=1}^{n} \gamma_l\right)^{-\frac{1}{2p+n}} \gamma_i = \left(\prod_{l=1, l\neq i}^{n} \gamma_l\right)^{-\frac{1}{2p+n}} \gamma_i^{\frac{2p+n-1}{2p+n}} \approx 0\,,$$

$$m_j = \left(\prod_{l=1, l\neq j, l\neq i}^{n} \gamma_l\right)^{-\frac{1}{2p+n}} \gamma_i^{-\frac{1}{2p+n}} \gamma_j^{\frac{2p+n-1}{2p+n}} \approx +\infty\,.$$

Therefore, prescribed sizes h_j, with $j \neq i$, will be very small as compared to prescribed size in direction i, h_i. In this case, this is precisely the very small Hessian eigenvalue for direction i that has imposed the choice of very small sizes in the other directions for $\mathbf{M}_{\mathbf{L}^p}$. However, if we finally increase γ_i so that the size prescribed in direction $\mathbf{r}_i$ is reduced to h_{max}, there is no reason anymore for the sizes in the other directions to be so small. Actually, what we would like to do is to relax the constraint on the other directions while increasing the constraint in direction i. In other words, this comes to squeeze the unit ball in direction i while swelling it in the other directions, see Figure 1.10. Eventually, this approach is typically a *local* one which is not consistent with the choice previously made of a *global* optimization strategy. The objective of this section is to present a slightly more sophisticated truncation algorithm enabling to get rid of the above mentioned drawbacks.

Some essential remarks.

Truncation must be performed on the eigenvalues of the Hessian. As illustrated in Figure 1.10, we would like the enforcement of the maximal size criterium in one direction to be associated with a relaxation of small size constraints imposed by the metric in the other directions. To better understand the coupling between the sizes prescribed in the different directions, it is important to understand here that the two directions of the Hessian are not correlated (a change of one of its eigenvalue $\gamma_{k,i}$ does not affect the other eigenvalues) whereas it is not the case for the

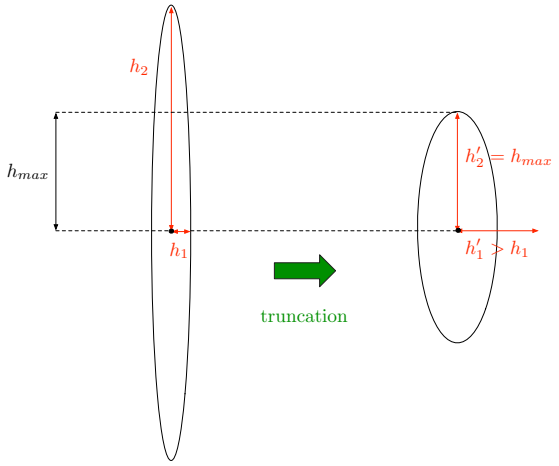


Figure 1.10: Squeeze and swell mechanism in two dimensions: the unit ball is squeezed to reach h_{max} in direction $\mathbf{r}_1$ and swollen in the other directions.

complete metric. Indeed, due to the local normalization term $\prod_{k=1}^{n} {\gamma_{k,i}}^{-\frac{1}{2p+n}}$, a change of one of the eigenvalues $\lambda_{k,i}$ results in a change in all the others eigenvalues. For example, let $x \in \Omega$ and let us imagine that $\gamma_1(\mathbf{x})$ increases while $\gamma_2(\mathbf{x})$ and $\gamma_3(\mathbf{x})$ keep the same value. As $\gamma_1(\mathbf{x})$ is associated with a positive power, $h_1(\mathbf{x})$ increases :

$$h_1(\mathbf{x}) = D_{\mathbf{L}^p} \left[\gamma_2(\mathbf{x})\gamma_3(\mathbf{x})\right]^{-\frac{1}{2p+n}} \gamma_1(\mathbf{x})^{\frac{2p+n-1}{2p+n}} .$$

On the contrary, the sizes in the other directions diminish because $\gamma_1(\mathbf{x})$ is associated with a negative power:

$$h_2 = D_{\mathbf{L}^p} \gamma_1(\mathbf{x})^{-\frac{1}{2p+n}} \gamma_3(\mathbf{x})^{-\frac{1}{2p+n}} \gamma_2(\mathbf{x})^{\frac{2p+n-1}{2p+n}} ,$$

$$h_3 = D_{\mathbf{L}^p} \gamma_1(\mathbf{x})^{-\frac{1}{2p+n}} \gamma_2(\mathbf{x})^{-\frac{1}{2p+n}} \gamma_3(\mathbf{x})^{\frac{2p+n-1}{2p+n}} .$$

This simple example emphasizes the crucial role played by the local normalization term $\det\left(|H_u(\mathbf{x})|\right)^{-\frac{1}{2p+n}}$ regarding the coupling between the sizes prescribed in the different eigen-directions. Notably, this shows that to observe the desired "squeeze and swell" effect, the truncation has to be performed on the Hessian eigenvalues rather than on the locally normalized or complete metric eigenvalues.

The non-linear coupling requires a global iterative strategy. As explained in the previous paragraph, enforcing a maximal size criterium by truncating the Hessian eigenvalues results in an increase in the complexity of the metric. On the other hand, modifying the complexity changes the expression of the maximal size criterium. This means that the sizes prescribed by the metric and the complexity of the metric are tightly related to each other. Moreover, the coupling involving $D_{\mathbf{L}^p}$ and the eigenvalues of the Hessian γ_i is highly *non-linear*:

$$D_{\mathbf{L}^p} = \frac{N}{\displaystyle\int_\Omega \left[\gamma_1(\mathbf{x})\gamma_2(\mathbf{x})\gamma_3(\mathbf{x})\right]^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x}}$$

For example, if γ_1 is modified, the normalization constant $D_{\mathbf{L}^p}$ changes. Moreover, the three maximal size criteria $D_{\mathbf{L}^p}\left[\gamma_1\gamma_2\gamma_3\right]^{-\frac{1}{2p+n}}\gamma_i \geq m_{thres}\,, \quad \forall\, i \in [\![1,3]\!]$ are also different due to changes in γ_1 and $D_{\mathbf{L}^p}$, which may lead us to correct one or two of the other eigenvalues, depending on whether the associated sizes violate threshold m_{thres}. If corrections are performed, the complexity will again be changed...etc. All this shows that a *global* iterative strategy is needed if one wants to respect the maximal size criterium while enforcing the complexity constraint.

1.4.4.2 Our approach

Initial correction of the eigenvalues of the Hessian. The very first problem to be solved is due to the possible presence of null Hessian eigenvalues, notably due to the local normalization term $\det(|H_u|)^{\frac{-1}{2p+n}}$ appearing in Formula (1.22). Indeed, if one of the γ_i's is zero at one vertex of the computational domain, this term is not defined. Here again, one solution would consist in setting any γ_i to a very small arbitrary value as soon as it falls below a pre-defined threshold ε.
However, the definition of this threshold is problematic, all the more as no clear interpretation can be found for values $\gamma_i(\mathbf{x})$. Indeed, the Hessians are not directly linked to any kind of "maximal prescribed size". They are actually *dimensioned* objects, *i.e.* their eigenvalues have a dimension: $[U]\cdot[L]^{-2}$, $[U]$ being the dimension of the field on which the adaptation is performed.

The solution adopted here to sidestep this difficult question of dimensioned ε consists in prescribing a maximal anisotropic ratio[8] instead of enforcing a minimal value for the eigenvalues of the Hessian. From our point of view, it has much more sense than directly prescribing a γ_{min} threshold and the physical meaning is much more clear for the user. Indeed, the anisotropic ratio of $|H_u(\mathbf{x})|$ is the same as the one of complete metric $\mathcal{M}(\mathbf{x})$, which means that by providing a maximal anisotropic ratio, we solve the null Hessian eigenvalues problem and at the same time guarantee that the final metric will also enforce this maximal anisotropic ratio criterium.

Let us now explain how the prescription of a maximal anisotropic ratio r_{max} implicitly defines a minimum eigenvalue γ_{min}. Let $r_{max} > 1$ be the maximal anisotropic ratio given by the user. Let $\mathbf{x} \in \Omega$ and let us assume that the sizes prescribed by the optimal metric at $\mathbf{x}$ are classified by increasing order:

$$h_1(\mathbf{x}) \leq h_2(\mathbf{x}) \leq h_3(\mathbf{x}) \quad \text{for } n=3 \Longleftrightarrow \frac{1}{\sqrt{m_1(\mathbf{x})}} \leq \frac{1}{\sqrt{m_2(\mathbf{x})}} \leq \frac{1}{\sqrt{m_3(\mathbf{x})}}$$

$$\Longleftrightarrow m_1(\mathbf{x}) \geq m_2(\mathbf{x}) \geq m_3(\mathbf{x}) \Longleftrightarrow \gamma_1(\mathbf{x}) \geq \gamma_2(\mathbf{x}) \geq \gamma_3(\mathbf{x})\,.$$

If $r(\mathbf{x})$ designates the anisotropic ratio of the continuous element given by the optimal metric

[8] the anisotropic ratio of an element is defined as the ratio between the length of its biggest edge and the length of its smallest edge.

at $\mathbf{x}$, we would like to enforce:

$$r(\mathbf{x}) < r_{max} \iff \frac{h_3(\mathbf{x})}{h_1(\mathbf{x})} < r_{max} \iff \frac{\gamma_3(\mathbf{x})}{\gamma_1(\mathbf{x})} > R_{min}\sqrt{\frac{m_1(\mathbf{x})}{m_3(\mathbf{x})}} < r_{max} \iff \frac{m_3(\mathbf{x})}{m_1(\mathbf{x})} > \frac{1}{r_{max}^2}$$

$$\iff \quad \iff \gamma_3(\mathbf{x}) > \gamma_1(\mathbf{x})\, R_{min}\,, \qquad \text{and} \qquad R_{min} = \frac{1}{r_{max}^2} < 1\,.$$

To ensure $r(\mathbf{x}) < r_{max}$, we must therefore correct the Hessian eigenvalues in the following way:

$$\begin{aligned} \gamma_2(\mathbf{x}) &\leftarrow \max \gamma_2(\mathbf{x}), \gamma_1(\mathbf{x}) R_{min} \\ \gamma_3(\mathbf{x}) &\leftarrow \max \gamma_3(\mathbf{x}), \gamma_1(\mathbf{x}) R_{min} \end{aligned} \tag{1.26}$$

If eigenvalue $\gamma_1(\mathbf{x})$ of $|H_u(\mathbf{x})|$ is non-zero, we obtain after Correction (1.26):

$$\gamma_j(\mathbf{x}) \geq \gamma_1(\mathbf{x})\, R_{min} = \gamma_{min}, \quad \forall\, j > 1\,.$$

In the case where $\forall\, i,\ \gamma_i(\mathbf{x}) = 0$, the only solution is to directly prescribe an arbitrary minimum value γ_{min}, for instance $\gamma_{min} = 10^{-40} \cdot U_{ref} \cdot L_{ref}^{-2}$, where U_{ref} and L_{ref} are some reference quantities associated with the adaptation variable and the length units, respectively. They can be provided by the user or may be inferred from known quantities (domain box size, value of the solution at one point...).

Coupling management. As explained in the preliminary remarks, an iterative strategy is mandatory due to the non-linear coupling. Each iteration is made of two stages:

1. a local cutting phase of the metric, during which all the sizes exceeding h_{max} are truncated,
2. a global growing phase of then metric, during which the metric unit balls are uniformly enlarged to try to make up for the complexity induced by Stage 1.

One can therefore legitimately speak about *explicit* coupling regarding our approach insofar as the global normalization term and the Hessian eigenvalues are not simultaneously corrected but rather one after the other. Figure 1.12 shows the evolution of a given metric field during the iterative non-linear loop. The progressive enlargement of the metric is easily observed. Note that for visualization issues, the metric has been voluntarily multiplied by 40 to avoid the intersection and superposition of many metrics on the images. The global truncation algorithm is detailed in Figure 1.11.

Convergence issues. To demonstrate the termination and the exactness of the algorithm, we will proceed in two steps:

1. We first demonstrate that sequence $\left(D^j_{\mathbf{L}^p}\right)_j$ is decreasing while the number of iterations j increases, hence the denomination "growing phase",
2. Then, the convergence of $\left(D^j_{\mathbf{L}^p}\right)_j$ is shown,
3. Finally, it is proved that the metric obtained when passing to the limit respects the complexity constraint and the maximal size criterium.

Proof of 1. First, according to Relation (1.27), the numerical complexity $\mathcal{C}^{j+1}$ of the metric computed at iteration j of the truncation algorithm satisfies:

$$\mathcal{C}^{j+1} \;=\; \mathcal{C}\left(\mathcal{M}^{j+1}\right) \;=\; \sum_{i=1}^{N_v}\left[\prod_{l=1}^{n}\lambda_l^{j+1}\left(\mathbf{x}_i\right)\right]^{\frac{1}{2}} \times |C_i| \;\geq\; |\Omega|\left(\frac{m_{thres}}{D^j_{\mathbf{L^p}}}\right)^{\frac{n}{2}},$$

where we have noted $|C_i|$ the volume of the dual-cell associated with vertex $P_i = (\mathbf{x}_i)$ (see Section 5.2) and $D^j_{\mathbf{L^p}}$ the global normalization constant computed at iteration j of the truncation algorithm. Therefore, the global normalization constant computed at the next iteration verifies:

$$D^{j+1}_{\mathbf{L^p}} = N^{\frac{2}{n}}\left(\mathcal{C}^{j+1}\right)^{-\frac{2}{n}} \leq \left(\frac{N}{\mathcal{C}_{thres}}\right)^{\frac{2}{n}} D^j_{\mathbf{L}^p},$$

and we have noted:

$$\mathcal{C}_{thres} = \int_\Omega \sqrt{\det \mathcal{M}_{thres}}\,\mathrm{d}\mathbf{x} = \int_\Omega m_{thres}^{\frac{n}{2}}\,\mathrm{d}\mathbf{x} = |\Omega|\, m_{thres}^{\frac{n}{2}}.$$

$\mathbf{M}_{thres}$ is the isotropic continuous mesh prescribing size h_{max} everywhere in all directions. Suppose that parameters N and h_{max} are prescribed in such a way that the following relation is satisfied:

$$\frac{N}{\mathcal{C}_{thres}} < 1. \tag{1.28}$$

In this case, sequence $(D^j_{\mathbf{L}^p})_j$ is decreasing, which justify the name "growing phase" for the metric. Note that Condition (1.28) is totally justified and can be interpreted as a consistency criterium to be satisfied by the parameters of the truncation algorithm. Indeed, if a maximal size criterium is enforced, the prescribed complexity cannot be arbitrary low: there must be enough vertices just to mesh the domain without exceeding h_{max}. The coarsest continuous mesh that respects both criteria is defined by metric $\mathcal{M}_{thres}$. It is therefore inconsistent to ask for a complexity lower than C_{thres}, C_{thres} being the lowest possible complexity compatible with prescribed parameter h_{max}.

Proof of 2. $(D^j_{\mathbf{L}^p})_j$ is decreasing, but it is also a positive sequence by definition. Therefore, this sequence converges towards a limit $D_{\mathbf{L}^p}$.

Proof of 3. Finally, we must show that the metric obtained at the end of the algorithm indeed satisfies both criteria. Corrected Hessian eigenvalues $(\gamma_l^{j,cor})_{l\,\in\,[\![1,n]\!]}$ are indeed calculated so that the resulting corrected metric sizes enforce the maximal size constraint. As they are used at the end to compute the final metric, this metric also enforces the maximal size prescription. ■

As a conclusion, this new algorithm solves most of the problematics encountered with the rough truncation strategy. **By relaxing the very small size constraints induced by very big sizes prescription along shocks, it enables in many cases to enlarge the smallest altitudes of the mesh while preserving the desired mesh accuracy. The mesh obtained after truncation is closer to the true optimal mesh associated with the complexity constraint.**
Due to the CFL condition, which links the smallest altitude of the mesh to the maximal time

step of the solver, this algorithm impacts favorably on the CPU time of CFD simulations for a given accuracy, and notably those involving strong shock waves. **A gain of a factor 2 in terms of CPU time has been observed in most of our simulations**.

Initialization.

- For each vertex of coordinates $\mathbf{x}$:
  - Perform the initial correction on the eigenvalues $(\gamma_l)_l$ of the pure Hessian, see Formula (1.26). These corrected eigenvalues, named $(\gamma^0)_l$, are stored. They will always be used as a starting point for the non-linear loop.
  - Add the contribution of this vertex to the new global complexity $\mathcal{C}^0$.
- Compute global normalization term $D^0_{\mathbf{L}^p} = \left(\frac{N}{\mathcal{C}^0}\right)^{\frac{2}{n}}$, $\quad j \longleftarrow 0$

Non-linear loop. Do:

- For each vertex P_i of coordinates vector $\mathbf{x}_i$:
  - Recalculate the eigenvalues of the complete metric $\left(m^j_l(\mathbf{x}_i)\right)_{l\in[\![1,n]\!]}$ using the initial corrected Hessian eigenvalues $\left(\gamma^0_l(\mathbf{x}_i)\right)_{l\in[\![1,n]\!]}$,
  - Cutting stage: If the metric field locally prescribes a size $h_l(\mathbf{x}_i) > h_{max}$ in direction $\mathbf{r}_l$, compute corrected Hessian eigenvalues $\gamma^{cor,j+1}_l(\mathbf{x}_i)$ as a function of $D^j_{\mathbf{L}^p}$, h_{max} and $\gamma^0_l(\mathbf{x}_i)$ and such that the new corresponding size $h_l(\mathbf{x}_i)$ is truncated to h_{max}.
    This comes to ensure that:
    $$\lambda^{j+1}_l(\mathbf{x}_i) \geq \frac{m_{thres}}{D^j_{\mathbf{L}^p}}, \quad \forall\, l \in [\![1,n]\!]\,. \tag{1.27}$$
  - Add the contribution of this vertex to the new global complexity:
    $$\mathcal{C}^{j+1} \longleftarrow \mathcal{C}^{j+1} + \left(\prod_{l=1}^{n} \lambda^{j+1}_l(\mathbf{x}_i)\right)^{\frac{1}{2}} |C_i|\,,$$
    and $|C_i|$ is the control volume associated with vertex P_i.
- If the new complexity is close enough to the desired one, break.
- Growing stage: Otherwise, update the global normalization term
  $$D^{j+1}_{\mathbf{L}^p} \longleftarrow \left(\frac{N}{\mathcal{C}^{j+1}}\right)^{\frac{2}{n}}.$$
  This comes to make the unit balls of the metric field become larger.
- $j \longleftarrow j+1$

Continue until desired complexity N is not reached, *i.e.* $\dfrac{|D^{j-1}_{\mathbf{L}^p}\mathcal{C}^j - N|}{N} \geq \varepsilon$.

Ending. Compute the final metric at each vertex using $\left(\gamma^{j,cor}_l(\mathbf{x}_i)\right)_l$ and $D^j_{\mathbf{L}^p}$:

$$\mathcal{M}(\mathbf{x}_i) \longleftarrow D^j_{\mathbf{L}^p}\left[\det\left(|H^{j,cor}_u|(\mathbf{x}_i)\right)\right]^{-\frac{1}{2p+n}} |H^{j,cor}_u|(\mathbf{x}_i)\,,$$

where $H^{j,cor}_u$ is the Hessian matrix obtained with the same directions as the initial Hessian but with the last corrected eigenvalues $\left(\gamma^{j,cor}_l\right)_l$.

Figure 1.11: The truncation algorithm for steady adaptation.

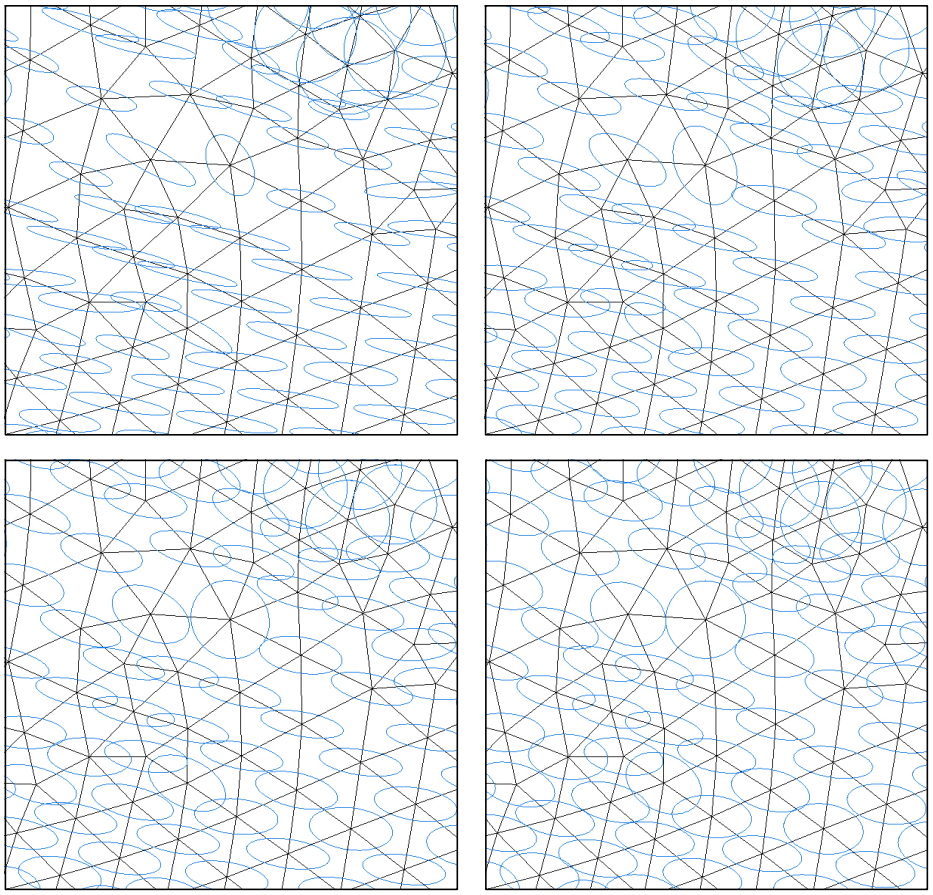


Figure 1.12: Metric field obtained at iterations 1, 3, 5 and 7 of the truncation algorithm process on a given functional in two dimensions. The global growth of the metrics is clearly visible.

Chapter 2

Unsteady metric-based mesh adaptation: theory and practice

Contents

2.1 The challenges of unsteady simulations 60
2.1.1 Motivations 60
2.1.2 Problematics linked to unsteadiness. 60
2.1.3 Problematics linked to unsteady anisotropic mesh adaptation. 64
2.1.4 State of the art 65
2.1.5 Our approach 73
2.2 Space-time error analysis and CFL condition 74
2.2.1 Truncation error analysis in 1D 74
2.2.2 Illustration of the influence of the CFL number on the space-time approximation error 78
2.2.3 Multi-dimensional theoretical analysis of the interpolation error 80
2.3 Extension of multi-scale mesh adaptation to unsteady simulations. . . 86
2.3.1 Controlling the error during a whole adaptation sub-interval with a single adapted mesh in a multi-scale framework. 87
2.3.2 Local space-time mesh complexity and global fixed-point strategy. 89
2.3.3 Practical use of the optimal space-time metric inside the fixed-point algorithm. 93
2.4 Numerical illustrations and comparison with the former algorithm . . 99
2.4.1 Three-dimensional Double-Mach reflection simulation 100
2.4.2 Three-dimensional blast in a city 102

In the previous chapter, the usefulness of metric-based mesh adaptation in the context of pseudo-steady solutions of unsteady numerical simulations has been investigated. Notably, substantial gains both in terms of CPU time and automatization have been emphasized.
This chapter deals with metric-based mesh adaptation applied to *unsteady* simulations. Section 2.1 recalls the main problematics linked to unsteadiness in the context of mesh adaptation and gives an overview of existing methodologies. Section 2.2 is interested in the study of space-time errors. Notably, the continuous interpolation error analysis is extended to unsteady problems and clarifies the role played by the CFL condition in such simulations. Section 2.3 is devoted to the practical implementation of the fixed-point algorithm extended to multi-scale simulations. Finally, Section 2.4 presents several three-dimensional unsteady adaptive numerical results.

2.1 The challenges of unsteady simulations

2.1.1 Motivations

2.1.1.1 Motivation: real-life *is* unsteady

To begin with, it is important to insist on the fact that most of the realistic problems emanating from the industrial world are intrinsically unsteady. The accuracy control of unsteady simulations, notably in three dimensions, therefore appears as a major stake for the years to come. This can sound as a triteness but, seeing the number of papers dealing with this subject, it seems that the problematic has still not grabbed the attention it would really deserve.
Here are some unsteady simulation examples in fluid mechanics, among many others:

- *Blast simulations.* For safety purpose, the simulation of shock waves induced by explosions and their impact on complex-shaped buildings get increasing interest. Examples of such simulations can be found in [Baum 1993b, Baum 1995a, Baum 2003, Alauzet 2007, Stück 2009].
- *Turbulence and instabilities.* Every turbulent flow is intrinsically unsteady even if it can sometimes be considered as statistically stationary. In aeronautics, simulating the turbulent wake behind aircrafts or the turbulent boundary layer is of major interest. The accurate capture of instabilities is also crucial, see [Woodward 1984, Langseth 2000].
- *Multi-fluid flows.* Multi-fluid flows computations, which require an accurate capture of time-dependent complex interfaces between different fluids [Guégan 2010, Allain 2009, Compère 2007, Lesage 2007, Mesri 2008].
- *Simulations involving moving geometries.* Aeroelasticity [Farhat 2003], seat ejection [Baum 1997a], biomedical applications [Astorino 2009], turbo-machinery [Shyam 2010].

2.1.2 Problematics linked to unsteadiness.

2.1.2.1 The Courant-Friedrichs-Lewy condition

As already said, this work is interested in the simulation of flows governed by hyperbolic PDE. Due to the presence of convective terms (linear or not), these computations are submitted to a Courant-Friedrichs-Lewy (CFL) condition. This condition and some of its possible interpretations are recalled below.

The one-dimensional linear advection equation. The following problem is considered:

$$(\mathcal{P}) \quad \begin{cases} \dfrac{\partial u}{\partial t}(x,\,t) + a\,\dfrac{\partial u}{\partial x}(x,\,t) = 0\,, \quad \forall\,(x,\,t)\,\in\,\mathbb{R}\times\mathbb{R}^{+} \qquad a>0\,, \\ \\ u(x,\,t=0) = u_0(x), \quad \forall\,x\,\in\,\mathbb{R}\,. \end{cases}$$

This Cauchy system has an analytical solution given by: $u(x,t) = u_0(x - a\,t)$.[1] On a $(x,\,t)$-diagram, the characteristics[2] of $(\mathcal{P})$ form a family of parallel straight lines of slope $1/a$, see

[1] This expression shows that a can be seen as a *phase speed*, or celerity.
[2] The characteristics are the lines of the (x,t)-diagram along which u remains constant.

Figure 2.1.
Let x_j for $j = 1, \ldots, N_v$ be a uniform spatial discretization of the one-dimensional domain and let t^k for $k = 0, ..., N_{ite}$ be a uniform time discretization. N_{ite} is the number of solver time steps. We note $h = x_j - x_{j-1}$ and $\tau = t^k - t^{k-1}$, which are assumed to be constant. An upwind scheme is chosen to discretize $(\mathcal{P})$:

$$u_j^{k+1} = u_j^k + a\,\frac{\tau}{h}\left(u_j^k - u_{j-1}^k\right), \quad j \in [\![1,\, N_v]\!],\ k \in [\![0, N_{ite}]\!]\,. \tag{2.1}$$

For this scheme, the $\mathbf{L}^2$ and $\mathbf{L}^\infty$ Von Neumann stability analysis both reveal a necessary condition of stability, named Courant-Friedrichs-Lewy condition, involving the ratio h/τ, and that reads:

$$a\frac{\tau}{h} \leq 1 \Longleftrightarrow \tau \leq \frac{h}{a}, \quad \forall\, j \in [\![1,\, N_v]\!].$$

This inequality states that time step τ must be lower than the time taken by the wave to cross one element of the spatial mesh. In other words, the propagation speed of the numerical information must be lower than the one of the physical information.

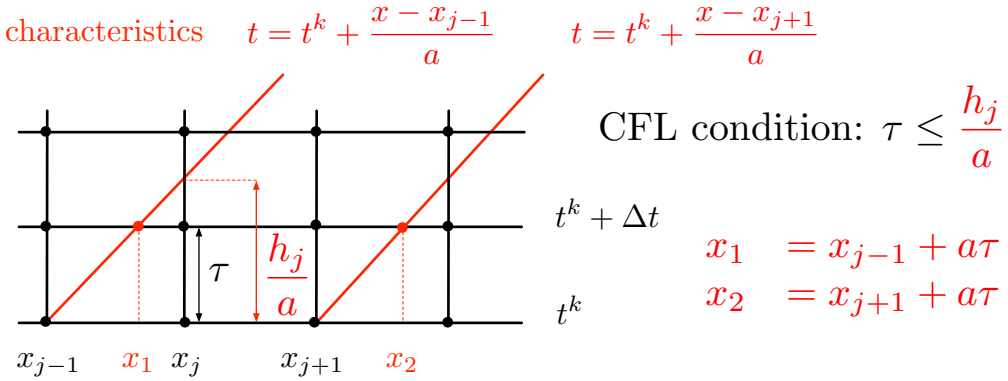


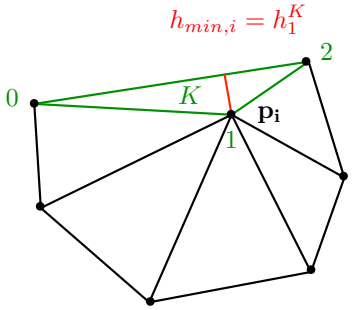
Figure 2.1: The CFL condition in one spatial dimension. The parallel characteristics passing through $(x_{j-1},\, t^k)$ and $(x_{j+1},\, t^k)$ are drawn in red.

The CFL condition for the Euler equations. As opposed to the previous linear advection problem, the Euler equations form a system of multi-dimensional non-linear coupled equations. They are generally solved on unstructured meshes to handle complex geometries. Therefore, the Von Neumann method cannot be used for the stability analysis of the numerical schemes used to discretize these equations. However, the stability of the linearized system around a given solution is a necessary condition for stability. The idea is therefore to perform a local analysis of the linearized Euler equations.
Now, let us consider the Euler system of equations and an arbitrary direction of propagation $\mathbf{r}$, with $||\mathbf{r}|| = 1$. Locally, the Euler system of equations can be linearized into:

$$\frac{\partial W'}{\partial t} + \mathcal{A}(W)\frac{\partial W'}{\partial \mathbf{x}} \;=\; 0\,,$$

where W is supposed to be a fixed solution state, W' is close to W and $\mathcal{A}$ is the Jacobian tensor of the Euler fluxes. A simple Fourier analysis shows that this system has wave-like solutions of

the form $W(\mathbf{x}, t) = W_0 \exp(at - \mathbf{r}\cdot\mathbf{x})$, a being the phase speed, *i.e* a is an eigenvalue of matrix $\mathcal{A}(W)\,\mathbf{r} = r_x\mathcal{A}_x + r_y\mathcal{A}_y + r_z\mathcal{A}_z$. The study of the eigenvalues of $\mathcal{A}(W)\,\mathbf{r}$ reveals three waves of phase speed $\mathbf{u}\cdot\mathbf{r}$, $\mathbf{u}\cdot\mathbf{r}+c$ and $\mathbf{u}\cdot\mathbf{r}-c$ respectively, where c is the local sound speed of the flow and $\mathbf{u}$ the Eulerian velocity of the flow. The biggest phase velocity is obtained for a direction $\mathbf{r}$ collinear to $\mathbf{u}$ and equals $||\mathbf{u}|| + c$. Besides, we are interested in the resolution of the Euler system on an unstructured mesh by a Finite Volume (FV) method. Therefore, the spatial size parameter h which represented the distance between two adjacent cells is not constant anymore. It has to be define locally. In our case, the size parameter is defined numerically at each vertex as:



$$h_{min,i} = \min_{K\,\in\,\mathrm{Ball}(P_i)} \min_{q\,\in\,[\![0,2]\!]} h_q^K\,,$$

and $\begin{cases} \mathrm{Ball}(P_i) & \text{the ball of vertex } P_i\,, \\ h_q^K & \text{the altitude of } K \text{ w.r.t local vertex } q\,. \end{cases}$ [3]

According to the previous paragraph, a necessary condition to enforce the CFL condition *locally* around P_i in all directions of propagation and for all waves is:

$$\tau \leq \frac{h_{min,i}}{||\mathbf{u}_i|| + c_i} = \tau_{min,i}\,.$$

As this has to be satisfied for each vertex, we get:

$$\tau \leq \min_{i\,\in\,[\![1,N_v]\!]} \tau_{min,i} = \tau_{min}\,.$$

In general, for robustness issues, τ is scarcely taken equal to τ_{min} but rather to:

$$\tau = CFL \times \tau_{min}\,, \quad \text{where } 0 < CFL \leq 1\,.$$

and CFL is the *Courant number* (or *CFL number*). Note that the use of an unconditionally stable implicit scheme for the time integration would theoretically enable to get rid of the constraint imposed by the CFL condition, but at the price of an increased numerical dissipation, see Section 2.2.2. Therefore, if high accuracy is desired, the time step must remain of the order of magnitude of the one prescribed by the CFL condition.

2.1.2.2 Consequences of the CFL condition

CPU time and convergence studies. As a corollary of the CFL condition, when the minimal altitude h_{min} is reduced, the upper limit for the time step also decreases and the number of solver time-advancing steps increases.
Convergence studies for unsteady simulations are therefore very time-consuming. For instance, let us a consider a simulation on a uniform mesh of typical size h. Then, in three dimensions,

[3] w.r.t is used for "with respect to"

the number of vertices N_v varies like h^{-3}. If the steady solver has a linear complexity in N_v and the unsteady solver a linear complexity in $N_v \times N_{ite}$, N_{ite} being the number of iterations, Table (2.1) is obtained.

	N_v	N_{ite}	CPU Unsteady
$h/2$	$\times 8$	$\times 2$	$\times 16$
$h/4$	$\times 64$	$\times 4$	$\boxed{\times 256}$!

Table 2.1: CPU time increase induced by successive divisions of mesh size h for unsteady simulations enforcing a CFL condition.

More generally, imagine a uniform complexity refinement of factor c is used to perform a convergence study, *i.e.* the simulation is performed for $N_v(0) = N_0$, $N_v(1) = cN_0$, ..., $N_v(k) = c^k N_0$. For unsteady simulations enforcing a CFL constraint, the following equivalence occurs:

$$\begin{aligned} \text{CPU time(k)} &= \mathcal{O}(N_{ite}(k) \times N_v(k)) = \mathcal{O}(h_{min}^{-1}(k) \times N_v(k)) = \mathcal{O}(N_v^{\frac{1}{3}}(k) \times N_v(k)) \\ &= \mathcal{O}(N_v^{\frac{4}{3}}(k)) = \mathcal{O}(c^{\frac{4k}{3}}) \,. \end{aligned}$$

For instance, if $c = 2$, the 5^{th} computation will take $2^{\frac{4\times 5}{3}} \approx 102$ times more CPU time than the first simulation. If the first simulation has taken 20 minutes, the 5^{th} one will take 34 hours.

This qualitative approach hence shows that **CPU time can be considered as a quality criterium to distinguish between different unsteady adaptive methods**, especially when convergence studies are considered. Moreover, it is clear that convergence studies cannot be considered unless mesh adaptive strategies are adopted.

The h_{min} bottleneck: a meshing challenge. Due to the CFL condition, a single small-altitude element in the whole mesh is sufficient to considerably reduce the time step and hence increase the CPU time of the simulation. This phenomenon is not specific to unsteady simulations as the CFL condition must also be enforced in the case of pseudo-steady solutions of unsteady simulations for stability purposes. However, for stationary problems, this issue can be sidestepped thanks to a local time stepping technique. For unsteady problems, this trick cannot be used anymore and a global time-stepping strategy is unfortunately mandatory. This is a serious problem, especially in the context of highly anisotropic mesh adaptation, which involves highly stretched elements and hence very small values of h_{min}.

One method to remedy this problem is to use several different time steps for different regions of the flow, as suggested in [Esnault 2011]. The idea is to partition the domain into several sub-domains. Flow equations are simultaneously advanced on each sub-mesh associated with each subdomain, each sub-mesh having its own local time step adapted to satisfy the *local* CFL condition. This strategy seems quiet promising but is not addressed in this Thesis.

Due to the impossibility to use local time stepping schemes, the only remedy is to reduce this constraint as far as possible by generating anisotropic meshes controlling the error with the highest h_{min} value. This implies a substantial effort on the anisotropic mesh generator, notably in three dimensions, as the quality of the mesh must be perfect. As way of an example, if the

mesh generator fails to generate the minimal altitude element for which a size of h_{target} had been prescribed, and instead build an element of altitude $h_{min} = 0.01 \times h_{target}$, the number of iterations of the solver is multiplied by 100! Consequently, not a single meshing mistake is allowed on millions of generated tetrahedra!

2.1.3 Problematics linked to unsteady anisotropic mesh adaptation.

Four-dimensional space-time meshes. The general problem of "true" space-time adaptation, which would consist in finding the best space-time unstructured mesh controlling the space-time error, currently remains intractable. Indeed,

- its application to three-dimensional CFD computations would certainly be tricky as it implies the generation of four-dimensional anisotropic unstructured meshes of complex space-time domains,

- A problem appears when Fluid-Structure Interaction is considered. Indeed, in this case the position in time of moving objects, *i.e.* the definition of the fourth-dimensional space-time computational domain, is part of the unknowns of the equations because they are computed depending on the forces applied on the moving bodies by the surrounding fluid. This means that an adapted mesh for a space-time domain must be built while this space-time domain is not even known in advance,

- it would require the implementation of new numerical schemes, such as space-time Discontinuous Galerkin methods. Indeed, for a general space-time discretization, the method of lines, which consists in computing the solution time after time, would not be valid anymore and most of the schemes currently implemented could not be used.

For the moment, we are therefore forced to restrict ourselves to "time-advancing" schemes: the space-time computations are carried out for one space-time "slab" at a time, where the "slab" is the slice of the space-time domain between time t^n and t^{n+1}. This spares a three-dimensional problem from becoming a four-dimensional problem including time dimension. In other words, a strong constraint is imposed on the allowed space-time meshes: space time elements must be right prisms the basis of which is a triangle in two dimensions and a tetrahedra in three dimensions. In the sequel, such kind of space-time meshes will be referred to as "time-advancing" meshes.

Adaptation lateness. The adaptation loop presented in Chapter 1 is intrinsically not suited to unsteady simulations, because in such simulations, the temporal evolution of the physical phenomena inside the physical domain is totally unpredictable. Indeed, for pseudo-steady solutions of unsteady simulations, the adaptation algorithm consists in finding a fixed-point for the couple mesh/solution. If this algorithm is applied as it to unsteady simulations, the adapted mesh will always be late, behind the solution. In other words, the mesh is adapted for the solution at time t^{n-1} while we are calculating the solution at t^n. Notably, physical phenomena can evolve outside the adapted area between t^{n-1} and t^n, which can spoil the solution accuracy.

2.1.4 State of the art

If *steady* metric-based mesh adaptation has been the subject of many papers for the last 20 years, one cannot tell as much about *unsteady* metric-based mesh adaptation. Three main types of methods can be listed: adaptation by mesh movement, re-meshing methods and the fixed-point method.

2.1.4.1 Moving mesh methods

These methods are sometimes called "r-methods" or "dynamical adaptive methods". In these techniques, the mesh nodes are typically moved or reallocated continuously, via a mesh equation, to adapt to the evolutionary features of the solution. The mesh keeps the same topology and number of vertices, but vertices are allowed to move toward the regions of main interest. However, in this case, the movement of the inner vertices has to be taken into account in the equations while they follow the unsteady features of the solution. Generally, moving mesh methods differ by:

- the considered adaptation criterium,
- the way the adaptation criterium governs the mesh movement,
- the way the influence of the vertices movement on the governing equations is taken into account (quasi-Lagrangian approaches v.s. conservative rezoning/interpolation approaches).

The attractiveness of these methods lies in their ability to get rid of spoiling interpolation stages by formulating the problem in a global, fully consistent framework. Indeed, the equation governing the movement of the mesh and the one governing the fluid dynamics are solved simultaneously, the fluid equation being rewritten in a moving mesh framework. One usually distinguish two main families of methods: velocity-based and location-based.

Velocity-based methods. For *velocity-based* methods, the movement of each vertex is found through its velocity, which is then integrated in time to find the new location of the considered vertex. These methods are classified in three categories:

- Lagrangian methods,
- Moving Finite Elements,
- and GCL methods.

Lagrangian methods: In this case, the velocity of the vertices is exactly the Lagrangian velocity of the fluid. Actually, mesh vertices are assimilated to fluid particles and follow the dynamics of the fluid:

$$\boldsymbol{w} = \mathbf{u}_{fluid}\,.$$

$\boldsymbol{w}$ is the instantaneous velocity of the mesh, see Section 5.1 for more details. The mesh therefore tends to adapt to the streamlines of the flow. Of course, for this method, the equations governing the fluid dynamics are solved in their Lagrangian formulation [Loubère 2010].

Moving Finite Elements: The Moving Finite Element introduced by [Miller 1981] computes the solution and the mesh velocity at the same time by minimizing the residual of the considered PDE rewritten in a moving mesh framework. More precisely, suppose that the PDE reads:

$$\frac{\partial u}{\partial t} = \mathcal{L}u\,,$$

with u the unknown solution and $\mathcal{L}$ a spatial differential operator. Then, the following minimization problem is solved:

$$\min_{\boldsymbol{w}(\cdot,t),\,\frac{\mathrm{D}u}{\mathrm{D}t}(\cdot,t)} \int_\Omega \left(\frac{\mathrm{D}u}{\mathrm{D}t}(\mathbf{x},\,t) - \boldsymbol{w}(\mathbf{x},\,t)\cdot\nabla_{\mathbf{x}}u(\mathbf{x},\,t) - \mathcal{L}u(\mathbf{x},\,t)\right)\phi(\mathbf{x},\,t)\,\mathrm{d}\mathbf{x}\,,$$

where ϕ is a weight function to be determined and $\frac{\mathrm{D}}{\mathrm{D}t}$ holds for the Lagrangian derivative. Penalty terms are generally added to avoid the apparition of singularities in the mesh movement equations [Carlson 1998a, Carlson 1998b, Baines 1995].

GCL method: This method, based on the Geometric Conservation Law, has been introduced in [Cao 2002]. According to the Eulerian framework, for any transformation $\mathbf{x} = \mathbf{x}(\xi,t)$ of a continuous medium, with $\mathbf{x}(\xi,t=0)=\xi$, the instantaneous relative variation of an elementary volume V is:

$$-\frac{1}{\mathrm{d}V}\frac{\mathrm{D}V}{\mathrm{D}t}(\xi,t) = \mathrm{div}_\xi \boldsymbol{w}\,.$$

By analogy, the prescribed density of vertices $d = \sqrt{\det\mathcal{M}}$ is assumed to satisfy:

$$-\frac{1}{d}\frac{\mathrm{D}d}{\mathrm{D}t} = \mathrm{div}_\xi \boldsymbol{w} \Longrightarrow \frac{\partial d}{\partial t} + \mathrm{div}_\xi(d\,\boldsymbol{w}) = 0\,.$$

However, this equation alone is not sufficient. Indeed, according to the Helmholtz Decomposition theorem, which states that a vector field is uniquely determined by its divergence and its curl, we should also get information about the curl of $\boldsymbol{w}$ to be able to reconstruct $\boldsymbol{w}$ and integrate it to get the new vertices locations. Different approaches can be chosen, depending on the adaptation criterium. Cao and his coworkers [Cao 2002] think that the adaptive mesh should follow the flow of some given vector field $\mathbf{a}(x,\,t)$ and suggest the use of the following equation:

$$\mathrm{rot}_\xi(\phi(\mathbf{a}-\boldsymbol{w})) = 0\,,$$

and ϕ is a scalar weight function to be chosen. Practically, it is often taken equal to 1 or to fluid pressure p. Then, combining the equation on the divergence and on the curl of $\boldsymbol{w}$, and reformulating it into a minimization problem, he gets the following problem:

$$\min_{\boldsymbol{w}} \int_{\Omega_\xi} |\frac{\partial d}{\partial t} + \mathrm{div}_\xi(d\,\boldsymbol{w})|^2 + \left(\frac{d}{\phi}\right)|\,\mathrm{rot}_\xi\,(\phi(\mathbf{a}-\boldsymbol{w}))\,|^2\,\mathrm{d}\xi\,.$$

Note that the so-called deformation map method of Moser [Moser 1965] actually comes to take $\mathbf{a}=0$ and $\phi=\rho$. Both methods provide the same divergence for $\boldsymbol{w}$, which means that the local density of the mesh will be identical. The difference lies in the local orientation of the resulting

mesh. GCL methods provide locally non-singular mappings but generate skewed meshes. Moreover, the link between the curl of $\boldsymbol{w}$ and the local anisotropy and orientation of the mesh is still not clear.

Generally, the main problem with these methods lies in their tendency to quickly generate tangling meshes.

Location-based methods. The other class of moving mesh adaptive methods is referred to as *location-based* methods as they directly provide the new location of the vertices without computing the mesh velocity. These techniques typically employ a *variational approach*. Adaptive meshes are seen as images of a computational, regular mesh by a coordinate transformation $\mathbf{x} = \mathbf{x}(\xi)$. This transformation is obtained by minimizing an adaptation functional expected to measure the difficulty in the numerical approximation of the physical solution. This functional generally involves a metric (also called monitor function), which is obtained either heuristically or by an estimation of some numerical error. Most of these methods find their origin in the Laplace-equation-based mesh generator introduced in [Winslow 1963]. The following methods are briefly explained here:

- Poisson type methods,
- Harmonic maps,
- Moving Mesh Partial Differential Equation (MMPDE).

Poisson type methods: Godunov and Prokopov [Godunov 1967] were the first to suggest the use of a Poisson-type equation instead of a Laplace equation. The left hand side, as for Winslow's method, is a Laplacian written in curvilinear coordinates while the right-hand side plays the role of a variable gravity term used to control locally the attraction/repulsion between vertices:

$$\Delta_{\mathbf{x}} \xi^i = P^i \,,$$

in which the "control functions" P^i can be fashioned to control the spacing and orientation of constant coordinate lines[4]. Thompson and Warsi [Thompson 1983] then enhanced this method by further explaining how to enrich source terms P^i with the information contained in a target metric. The equi-distribution approach of Anderson [Anderson 1982] aims at relocating mesh vertices in such a way that some mesh property Q is equi-distributed over the domain. This is generally done by minimizing a weighted integral measure of this property written under the form:

$$\int_{\Omega} Q(\mathbf{x}) \sqrt{\det \mathcal{M}(\mathbf{x})} \, \mathrm{d}\mathbf{x} \,.$$

The Euler-Lagrange equation corresponding to this minimization problem then constitutes the mesh adaptation PDE. This method was originally used to generate meshes enforcing some desired properties such as smoothness or orthogonality near the boundary [Brackbill 1982]. However, it turned out that this variational framework was much more suited to mesh adaptation

[4] Coordinates lines are the parametrized curves/surfaces defined by $\{\mathbf{x}(\xi_1 = 0, \ldots, \xi_{i-1} = 0, \xi_i, \xi_{i+1} = 0, \ldots, \xi_n = 0), \; \forall \xi_i \in \mathbb{R}\}$

problematics than to direct mesh generation, which led to the development of the theory of harmonic mappings.

Harmonic maps: The theory of harmonic mappings has been formalized by [Dvinsky 1991] and can be considered as an extension of equi-distribution methods that enables to take into account the alignment of the mesh through the prescription of a metric tensor field. This theory relies on the Hamilton-Schoen-Yau theorem, which gives a method to get a mapping, called harmonic mapping, between two Riemannian spaces through a variational problem. Harmonic maps are defined as critical points of an energy functional which is defined in terms intrinsic to the geometry of the domain, the target manifold and the map between them. In a certain sense, it represents the deformation energy to transform a flat space into a target manifold. This energy is defined as:

$$E(\xi(\mathbf{x})) = \int_\Omega \frac{1}{2} \left(\sum_{i=1}^{n} \nabla_{\mathbf{x}} \xi_{\mathbf{i}}(\mathbf{x})^T \cdot \mathcal{M}^{-1}(\mathbf{x}) \cdot \nabla_{\mathbf{x}} \xi_{\mathbf{i}}(\mathbf{x}) \right) \sqrt{\det \mathcal{M}(\mathbf{x})}\, \mathrm{d}\mathbf{x}\,.$$

The associated Euler-Lagrange equations form a system of size n:

$$\mathrm{div}\left(\sqrt{\det \mathcal{M}(\mathbf{x})} \mathcal{M}^{-1}(\mathbf{x}) \cdot \nabla_{\mathbf{x}} \xi_{\mathbf{i}}(\mathbf{x}) \right) = 0, \quad i \in [\![1, n]\!]\,.$$

This work has been applied to different kind of metrics: [Brackbill 1993, Ceniceros 2001, Chacón 2006, Di 2007]. Huang [Huang 2001] proposed the following variant for the adaptation functional, which is deduced from a rigorous analysis of the interpolation error:

$$E(\xi(\mathbf{x})) = \theta \int_\Omega \left(\sum_{i=1}^{n} \nabla_{\mathbf{x}} \xi_{\mathbf{i}}(\mathbf{x})^T \cdot \mathcal{M}^{-1}(\mathbf{x}) \cdot \nabla_{\mathbf{x}} \xi_{\mathbf{i}}(\mathbf{x}) \right) \sqrt{\det \mathcal{M}(\mathbf{x})}\, \mathrm{d}\mathbf{x}$$

$$+ (1 - 2\theta)\, n^{\frac{n\gamma}{2}} \int_\Omega \frac{\sqrt{\det \mathcal{M}(\mathbf{x})}}{\left(\sqrt{\det \mathcal{M}(\mathbf{x})} \det \nabla_\xi \mathbf{x} \right)^\gamma} \mathrm{d}\mathbf{x}\,.$$

The first term corresponds to an alignment requirement while the second term represents and equi-distribution requirement, the combination between these two being controlled by a weighting parameter θ. $\gamma > 1$ is a real parameter. Another kind of functional based on manifold first and last invariants has been proposed in [Azarenok 2008].

MMPDE: In [Huang 1999], the time parameter is introduced in the moving mesh equation by rewriting the variational formulation under the form of a gradient flow equation of the adaptation functional:

$$\frac{\partial \xi}{\partial t} = -\frac{1}{t_{ref} \alpha(\mathbf{x}, t)} \frac{\delta E}{\delta \xi}\,,$$

where α is a balancing factor and t_{ref} is the parameter specified to adjust the time scale of the movement. Once again, a variable change is performed to get an equation in function $\mathbf{x} = \mathbf{x}(\xi)$ instead of $\xi = \xi(\mathbf{x})$.

2.1.4.2 Frequent remeshing methods

To be able to write a global consistent system of equations governing at the same time the fluid dynamics and the movement of the mesh, moving mesh methods assume that the topology of the mesh as well as the number of vertices remain constant during the adaptation process. This assumption turns out to be very constraining when attempting to generate highly anisotropic meshes. Actually, it is not even clear that these methods are able to adapt the mesh with respect to any arbitrary smooth metric $\mathbf{M}$. For this reason, more pragmatic methods have been designed using existing steady adaptation techniques.

Frequent re-meshings. [Löhner 1992] and [Rausch 1992] suggested to perform adaptive re-meshing very frequently, but only involving refinement/coarsening and no vertex displacement to avoid interpolation errors. They also mentioned the usefulness of spreading the adaptation around critical regions. Thus, a safety zone is built that prevents the solution from going out of the adapted region between two adaptations. Such approaches have many drawbacks. First, they say nothing about the control of the temporal error. Moreover, refinement/coarsening methods do not enable the generation of optimal meshes in terms of vertices number and distribution, and forbid the generation of truly anisotropic meshes. Eventually, the "safety area" method intrinsically leads to the generation of non-optimal meshes.

Indicator based re-meshing. Unsteady adaptation algorithms attempting to generate optimal meshes in terms of vertices number and location have been first proposed in [de Sampaio 1993, Wu 1990]. The error is estimated every n_1 iterations with an indicator $\eta = \left(\sum_K \eta_K^2\right)^{\frac{1}{2}}$ and a re-meshing procedure is performed to readapt the mesh as soon as the indicator exceeds a prescribed threshold. In any case, the mesh is re-adapted every n_2 time steps.
A similar strategy is used in [Picasso 2003, Lozinski 2009, Micheletti 2008, Picasso 2009]. For instance, in [Picasso 2003], a global error indicator η controls mesh adaptation: if $\eta < 0.75\varepsilon$ or if $\eta > 1.25\varepsilon$, the domain is re-meshed, where ε is a prescribed threshold. For each re-meshing, the metric used to adapt the mesh is found using local directional error indicators. If one of the directional indicator associated with an element falls below $\alpha\varepsilon'$, with $\alpha \in [0, 1]$ a prescribed parameter and ε' another threshold, the eigenvalues of the metric in the considered direction are reduced; if it exceeds $(1-\alpha)\varepsilon'$, then the eigenvalues of the metric are increased of a prescribed scaling factor. By dichotomy, a nearly optimal metric is found. This method enables to increase or decrease the complexity of the mesh depending on the complexity of the solution and gives an answer to the problem of controlling space-time error in unsteady simulations. It also relies on a strong theoretical background. However, in practice, the mesh is re-adapted only when the error exceeds 1.25ε, which means that a degradation of mesh accuracy is allowed between two re-meshing stages. The influence of this degradation is not known and may affect the accuracy of the computation.

In practice, these methods still require frequent re-meshing stages, which leads to:

- *Increased CPU time:* First, as already mentioned, the re-meshing itself is not CPU-time free. Second, if static data structures are used in the solver, each re-meshing imposes to quit the solver, to restart it, to rebuild and initialize all the data structures, notably topological ones,

before the simulation is resumed. Obviously, when re-meshings are frequent, these restarting operations can be prohibitive in terms of CPU time. As a consequence, a solver based on dynamical data structures is generally preferred as in [Compere 2010]. But dynamical data structures do not allow fast search operations and tends to generate cache-misses, which may also negatively impact computational time.

- *Interpolation spoiling:* These frequent re-meshings go with very frequent interpolations of the solution from the old mesh to the new one. This may spoil the solution accuracy and convergence and may compromise the gain of accuracy brought by mesh adaptation. This effect is scarcely taken into account in the error analysis.

In our opinion, controlling the number of re-meshings seems to be a credible alternative.

2.1.4.3 Adaptive time-stepping for implicit simulations

Implicit computations, by relaxing the strong time step size constraint required for the stability of explicit time schemes, naturally raises the question of the *optimal* size for the solver time steps. The idea of adapting the time step for implicit schemes is not new and has been addressed for example by Gear [Gear 1971] in the early 70's. The idea is to choose the solver time step according to an estimation of the local error, notably the time error. For explicit schemes, this problematic does not appear as the largest time step satisfying the stability condition is generally chosen for efficiency purpose. The error estimates used in these methods are mostly heuristic. The truncation error is often used as a starting point. For instance, in [Kavetski 2002, Korhonen 2008], as the time integration scheme is of order one, the following error estimates based on a Taylor development is used :

$$e^t(P_i,\, t^k) \;=\; \frac{1}{2}\tau \frac{||\nabla W(P_i,\, t^k+\tau^k) - \nabla W(\mathbf{x},\, t^k)||}{W(P_i,\, t^k+\tau^k)}\,.$$

The local time step is reduced if this value is locally large and increased otherwise. In [Mani 2009, Mani 2010], the following local time error estimate is considered:

$$e^t(P_i,\, t^k) \;=\; |\left(\frac{\mathrm{d}\,(|C_i|W_i)}{\mathrm{d}t}\right)_{|BDF3} \;-\; \left(\frac{\mathrm{d}\,(|C_i|W_i)}{\mathrm{d}t}\right)_{|BDF2}|\,,$$

where $\left(\frac{\mathrm{d}(|C_i|W_i)}{\mathrm{d}t}\right)_{|BDF2}$ (resp. $\left(\frac{\mathrm{d}(|C_i|W_i)}{\mathrm{d}t}\right)_{|BDF3}$) is the numerical approximation of $|C_i|W_i$ obtained at current time step t^k using a BDF2 (resp. BDF3) implicit scheme.
The link with our approach is explained at the end of Section 2.2.3.

2.1.4.4 The $L^\infty - L^\infty$ fixed-point (or transient fixed-point) algorithm

To control the number of adaptations, an innovative strategy based on a fixed-point algorithm has been initiated in [Alauzet 2003a] and fully developed in [Alauzet 2007]. It has been successfully applied to a three-dimensional dam break problem [Guégan 2010] and to a blast simulation in a three-dimensional city [Alauzet 2003a, Alauzet 2007] with *isotropic* adaptation. This new strategy starts with the observation that direct extension of steady adaptation algorithms to

unsteady problems is not appropriate: specific algorithms must be developed, which truly take into account the transient nature of the solution. The fixed-point algorithm aims at avoiding the generation of a new mesh at each time-advancing step which would imply that a re-mesher is coded inside the solver. It is also an answer to the lag problem occurring when computing the solution at t^n and accordingly adapt the mesh at each time step. Indeed, by doing this, the mesh is always late as compared to the solution as it is not adapted for the displacement of the solution between t^n and t^{n+1}.

The basic idea consists in splitting the simulation time frame $[0,\, T]$ into n_{adap} adaptation sub-intervals:

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

and to keep the same adapted mesh for each whole sub-interval. Thus, a mesh must be built which controls the interpolation error for several solver time steps in a given adaptation sub-interval $[t_i,\, t_{i+1}]$. To this aim, a $\mathbf{L}^\infty(t_i, t_{i+1}; \mathbf{L}^\infty(\Omega))$ space-time adaptation strategy has been retained in [Alauzet 2007], which consists in requiring that the adapted mesh fulfills a specified ε-error criterion for the whole sub-interval $[t_i,\, t_{i+1}]$. The denomination $\mathbf{L}^\infty - \mathbf{L}^\infty$ must be understood as follows:

1. the first $\mathbf{L}^\infty$ means that the spatial interpolation error is controlled in $\mathbf{L}^\infty$ norm at each time of the adaptation sub-interval, *i.e.* the maximal value of the local spatial interpolation error $e^s(\mathbf{x},\, t)$ on Ω is mastered:

$$\forall\, t \;\in\; [t_i,\, t_{i+1}]\;, \quad \max_{\mathbf{x}\,\in\,\Omega} e^s(\mathbf{x},\, t) = \max_{\mathbf{x}\,\in\,\Omega} |u(\mathbf{x},\, t) - \Pi_h u(\mathbf{x},\, t)| \;\leq\; \varepsilon\,,$$

2. the second $\mathbf{L}^\infty$ refers to the way a single adapted mesh will control the spatial interpolation error during a whole sub-interval $[t_i,\, t_{i+1}]$. If a so-called "$\mathbf{L}^\infty$" strategy is chosen, this means that we choose to control the worst $e^s_{max}(t),\; t \in [t_i,\, t_{i+1}]$:

$$\max_{t\,\in\,[t_i,\,t_{i+1}]} e^s(t) = \max_{t\,\in\,[t_i,\,t_{i+1}]} e^s(t) \;\leq\; \varepsilon\,.$$

Practice. The solution is regularly sampled between t_i and t_{i+1}. The collected samples are noted $\{\mathcal{S}_{i,k}\}_{k=1,n_k}$ with $\mathcal{S}_{i,1} = u(\cdot, t_i)$ and $\mathcal{S}_{i,n_k} = u(\cdot, t_{i+1})$. The optimal metric controlling the global spatial interpolation error $E^s_{\mathbf{L}^\infty}(t)$ in $\mathbf{L}^\infty$ norm is exactly the Hessian matrix of the solution:

$$\mathcal{M}_{i,k} = |H_u(\cdot,\, t^k)| = |H_{i,k}|\,, \quad k \;\in\; [\![1, n_k]\!]\,.$$

Therefore, the Hessian of the solution is computed from each sample $\mathcal{S}_{i,k}$. Thus, each metric $|H_{i,k}|$ enables to control the local spatial interpol error $e^s_{\mathbf{L}^\infty}(t^k)$, as explained in 1.
To guarantee the $\mathbf{L}^\infty$ control of the spatial interpolation error during the whole sub-interval $[t_i,\, t_{i+1}]$ with a single mesh in the sense of Point 2 above, these metrics/Hessians are intersected as described in Section 1.2.6 and the resulting metric/Hessian is noted:

$$|H_{i,\max}| = \bigcap_{k=1}^{n_k} |H_{i,k}| = \bigcap_{k=1}^{n_k} |\mathcal{M}_{i,k}|\,.$$

For the sake of clarity, the number of samples n_k is assumed to be identical for each sub-interval. The time interval between two samples is also considered as constant equal to δt. Then, the time at which the k^{th} sampling of the solution between t_i and t_{i+1} is performed is:

$$t_{i,k} = t_i + (k-1)\delta t\,.$$

Eventually, a global normalization procedure is applied to enforce the spatial complexity constraint $\mathcal{C}(\mathcal{M}_i) = N^{ptfx}$ for each of the n_{adap} fixed point meshes. The $\mathbf{L}^\infty - \mathbf{L}^\infty$ continuous mesh for the whole sub-interval $[t_i, t_{i+1}]$ then reads:

$$\mathcal{M}_{i,\mathbf{L}^\infty-\mathbf{L}^\infty} = D_{\mathbf{L}^\infty\mathbf{L}^\infty}\,|H_{i,\max}| \tag{2.2}$$

$$\text{with} \quad D_{\mathbf{L}^\infty\mathbf{L}^\infty} = \left(N^{ptfx}\right)^{\frac{2}{n}} \left(\int_\Omega \sqrt{\det|H_{i,\max}(\mathbf{x})|}\mathrm{d}\mathbf{x}\right)^{-\frac{2}{n}} \tag{2.3}$$

$$\text{and} \quad |H_{i,\max}(\mathbf{x})| = \max_{t\in[t_i,\,t_{i+1}]} |H_u(\mathbf{x},\,t)|\ .$$

Fixed-point loop: converge jointly the mesh and the solution.

`For i=1,...,` n_{adap}

`For j=1,...,` n_{ptfx}

- $\mathcal{S}^j_{i,1}$ = `InterpolateSolution`$(\mathcal{H}^j_{i-1},\,\mathcal{S}^{j-1}_{i-1,n_k},\,\mathcal{H}^j_i)$
- $\{\mathcal{S}^j_{i,k}\}_{k=1,n_k}$ = `SolveState`$(\mathcal{H}^j,\,\mathcal{S}^j_{i,1})$
- $\mathcal{M}^j_i$ = `ComputeUnsteady`$\mathbf{L}^\infty - \mathbf{L}^\infty$`Metric`$(\{\mathcal{S}^j_{i,k}\}_{k=1,n_k})$
- $\mathcal{H}^{j+1}_i$ = `GenerateAdaptedMesh`$(\mathcal{H}^j_i,\,\mathcal{M}^j_i)$

`End for`
`End for`

Figure 2.2: The $\mathbf{L}^\infty - \mathbf{L}^\infty$ fixed-point algorithm.

The fixed-point algorithm is represented in Figures 2.2 and 2.3. The solution is computed on the first adaptation sub-interval $[t_0, t_1]$ and the variable of interest is sampled at regular time intervals of length δt. We have therefore gathered n_k samples of the solution between t_0 and t_1. The Hessian associated with each of these samples is computed and we now have n_k Hessian fields. These Hessian fields $\{|H_{0,k}|\}_{k=1,n_k}$ are intersected to get the maximal Hessian $|\,H_{0,\max}\,|$ for interval $[t_0\,, t_1]$ and a single resulting metric is computed according to Formula (2.2). This metric will prescribe to each vertex the maximal acceptable size guaranteeing the control of the spatial error in $\mathbf{L}^\infty$ norm during the first time sub-interval. A new mesh adapted to this sub-interval is then generated as described in Section 1.2.2 and we start again the sampling procedure on the same period. We loop until convergence of the mesh-solution couple for this

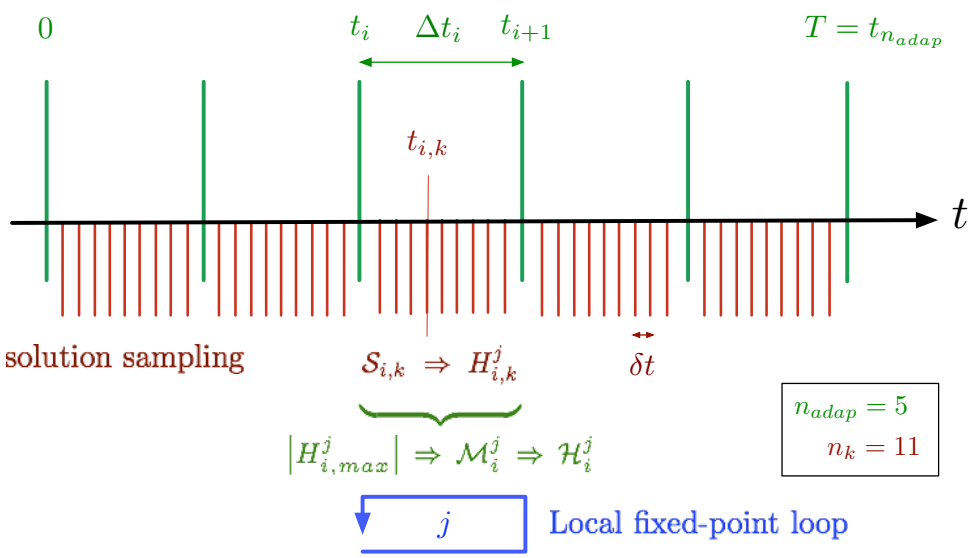


Figure 2.3: Sampling of the solution during each adaptation sub-interval and fixed-point loop.

period is reached, hence the name "fixed-point" algorithm. Generally, the system is assumed to have converged after n_{ptfx} number of fixed-point iterations. Once the solution has converged on $[t_0, t_1]$, we switch to the next sub-interval $[t_1, t_2]$ and repeat the process. Each of the n_{adap} sub-intervals is treated the same way. Note that in this algorithm, all the adaptation sub-intervals can be treated in a completely decoupled manner.

2.1.5 Our approach

In this thesis, the fixed-point strategy described in Section 2.1.4.4 has been chosen as starting point because of its:

- *Reduced CPU time consumption*, as compared to frequent re-meshing methods,
- *Theoretical foundations*, even incomplete, notably its attempt to guarantee a control of both spatial and temporal interpolation errors,
- *Generic nature*, *i.e.* its principle is general and the algorithm can be applied with any interpolation or meshing softwares and any type of solvers.

However, two main improvements have been brought to this algorithm:

- *Multi-scale*: The algorithm has been extended to perform *multi-scale* [Loseille 2010b] unsteady mesh adaptation,
- *Three-dimensional anisotropy*: Substantial efforts have been made regarding meshing softwares to handle anisotropy, especially in three space dimensions. Indeed, from now on, only *isotropic* adaptation had been shown in 3D for unsteady simulations. This thesis presents **three-dimensional anisotropic unsteady adaptation** results, performed with the re-meshing tool [Loseille 2010d].

This work has raised many new questions:

1. How to guarantee the control of *space-time* interpolation errors?
2. How can the continuous mesh framework be extended to space-time meshes? Notably, how to take into account the specific role played by the time dimension in numerical schemes?
3. How is this control performed in practice?
4. How to preserve mesh consistency, *i.e.* how to guarantee that the same phenomenon, for example a stationary shock, will be meshed in the same manner during the whole simulation?

Questions 1 and 2 have been answered thanks to a detailed analysis of the local and global space-time errors, carried out in Section 2.2. Questions 3 and 4 have led to a new fixed point algorithm, described in Section 2.3, in which the n_{adap} adaptation sub-intervals are not treated separately anymore but rather handled in a global, coupled way.

2.2 Space-time error analysis and CFL condition

In the case of pseudo-steady solutions, a meticulous analysis of the interpolation error and of the global optimization problem have been carried out in [Loseille 2010b, Loseille 2010c] to find the optimal metric controlling the interpolation error in $\mathbf{L}^p$ norm. This result has been obtained in a purely theoretical manner, which has been made possible thanks to the powerful continuous mesh formalism, see Chapter I. On the contrary, the efficiency of the fixed-point algorithm described in [Alauzet 2007] in terms of control of the space-time interpolation error relies on a simple truncation analysis performed on a one-dimensional, linear transport problem, see Section 2.2.1. This section attempts to remedy this lack of theoretical background and notably extends the notion of continuous interpolation error to time-advancing, unsteady simulations.

2.2.1 Truncation error analysis in 1D

We first recall here the space-time truncation error analysis detailed in [Alauzet 2007]. The linear scalar transport equation in one dimension is considered:

$$\frac{\partial u}{\partial t} + a\,\frac{\partial u}{\partial x} = 0, \quad \text{with } a > 0\,. \tag{2.4}$$

Let x_j for $j = 1, \ldots, N_v$ be a uniform spatial discretization of the one-dimensional domain and let t^k for $k = 0, \ldots, N_{ite}$ be a uniform time discretization. We note $h = x_j - x_{j-1}$ and $\tau = t^k - t^{k-1}$. The numerical solution obtained at vertex x_j and time t^k is noted u_j^k and is expected to be a good approximation of $u(x_j, t^k)$.

Euler-explicit upwind scheme. A time-explicit first-order upwind Finite-Difference scheme is chosen to discretize Equation (2.4):

$$\frac{u_j^{k+1} - u_j^k}{\tau} \;+\; a\,\frac{u_j^k - u_{j-1}^k}{h} = 0\,.$$

Using a Taylor expansion, the local truncation error reads:

$$\varepsilon_{h,\tau}(x_j,\, t^k) = a\frac{h}{2}\frac{\partial^2 u}{\partial x^2}(x_j,\, t^k) - \frac{\tau}{2}\frac{\partial^2 u}{\partial t^2}(x_j,\, t^k) + \mathcal{O}(h^2,\, \tau^2)\,.$$

The truncation error can then be split in a spatial and temporal truncation error:

$$\varepsilon_{h,\tau}(x_j,\, t^k) = \varepsilon_h^s(x_j,\, t^k) \;+\; \varepsilon_\tau^t(x_j,\, t^k)\,,$$

$$\text{with} \quad \varepsilon_h^s(x_j,\, t^k) \;=\; a\,\tfrac{h}{2}\,\tfrac{\partial^2 u}{\partial x^2}(x_j,\, t^k) \quad \text{and} \quad \varepsilon_\tau^t(x_j,\, t^k) \;=\; -\,\tfrac{\tau}{2}\,\tfrac{\partial^2 u}{\partial t^2}(x_j,\, t^k)\,.$$

Besides, deriving Equation (2.4) in time, using Schwarz theorem to exchange space and time derivatives of the second term and finally replacing $\frac{\partial u}{\partial t}$ by $-a\,\frac{\partial u}{\partial x}$ in the second term of the resulting expression, we get:

$$\frac{\partial^2 u}{\partial t^2} \;-\; a^2\,\frac{\partial^2 u}{\partial x^2} \;=\; 0, \quad \forall\,(x,\, t)\,.$$

The temporal truncation error can then be rewritten as:

$$\varepsilon_\tau^t(x_j,\, t^k) \;=\; -\,\frac{\tau}{2}\frac{\partial^2 u}{\partial t^2}(x_j,\, t^k) \;=\; -\,\frac{a\tau}{h}\,\frac{a\,h}{2}\,\frac{\partial^2 u}{\partial x^2}(x_j,\, t^k) \;=\; -\,\frac{a\tau}{h}\,\varepsilon^s(h) \;=\; -\,CFL\,\varepsilon_h^s(x_j,\, t^k)\,.$$

Therefore, the time error is bounded by the spatial error if a CFL condition is enforced.

RKSSP(3,3) and one-dimensional upwind Finite Volume scheme. For the Finite Volume approach in 1D, a cell C_j is defined as the interval $[x_{j-\frac{1}{2}},\, x_{j+\frac{1}{2}}]$, where $x_{j-\frac{1}{2}} = \frac{x_j + x_{j-1}}{2}$ and $x_{j+\frac{1}{2}} = \frac{x_j + x_{j+1}}{2}$. The numerical fluxes on each side of cell C_j are given by:

$$\Phi_{j-\frac{1}{2}} = \Phi(u^-_{j-\frac{1}{2}},\, u^+_{j-\frac{1}{2}})\,, \quad \Phi_{j+\frac{1}{2}} = \Phi(u^-_{j+\frac{1}{2}},\, u^+_{j+\frac{1}{2}})\,,$$

where $u^-_{j\pm\frac{1}{2}}$ denotes the extrapolated value of the numerical solution on the left side of interface $j \pm \frac{1}{2}$, and $u^+_{j\pm\frac{1}{2}}$ its extrapolated value on the right side.
Moreover, numerical flux Φ is upwind in the sense of [Harten 1983] if it writes:

$$\Phi(u,\, v) = \frac{1}{2}\Big[f(u) + f(v) - d(u,\, v)\Big]\,, \quad \text{with } d(u,\, v) = |f'(\frac{u+v}{2})|(v-u) + o(|u-v|)\,.$$

To control this numerical dissipation, a parameter δ is introduced:

$$\Phi^\delta(u,\, v) = \frac{1}{2}\Big[f(u) + f(v) - \delta\, d(u,\, v)\Big]\,.$$

In our specific case, the flux writes $f(u) = au$ with $a > 0$ and Φ^δ reduces to:

$$\Phi^\delta(u,\, v) = \frac{a}{2}\Big[(1+\delta)u + (1-\delta)v\Big]\,.$$

Then, the spatial numerical operator writes:

$$\frac{\Phi_{j+\frac{1}{2}} - \Phi_{j-\frac{1}{2}}}{h} = \frac{a}{2h}\Big[(1+\delta)u^-_{j+\frac{1}{2}} + (1-\delta)u^+_{j+\frac{1}{2}} - (1+\delta)u^-_{j-\frac{1}{2}} - (1-\delta)u^+_{j-\frac{1}{2}}\Big]\,. \tag{2.5}$$

The values of the solution at cells interfaces are extrapolated as follows:

$$\begin{aligned}
u^-_{j+\frac{1}{2}} &= u_j + \frac{1}{2}\Big[(1-\beta)(u_{j+1}-u_j) + \beta(u_j - u_{j-1})\Big]\,,\\
u^+_{j+\frac{1}{2}} &= u_{j+1} - \frac{1}{2}\Big[(1-\beta)(u_{j+1}-u_j) + \beta(u_{j+2} - u_{j+1})\Big]\,,\\
u^-_{j-\frac{1}{2}} &= u_{j-1} + \frac{1}{2}\Big[(1-\beta)(u_j-u_{j-1}) + \beta(u_{j-1} - u_{j-2})\Big]\,,\\
u^+_{j-\frac{1}{2}} &= u_j - \frac{1}{2}\Big[(1-\beta)(u_j-u_{j-1}) + \beta(u_{j+1} - u_j)\Big]\,,
\end{aligned}$$

and β is an upwinding parameter used to lower dissipation, see [Debiez 2000]. Substituting $u^{\pm}_{j\pm\frac{1}{2}}$ by their values in Relation (2.5), we get:

$$\frac{\Phi_{j+\frac{1}{2}} - \Phi_{j-\frac{1}{2}}}{h} = \frac{a}{4h}\Big[(1+\delta)\beta u_{j-2} - 2(1+\beta+2\delta\beta)u_{j-1} + 6\delta\beta u_j + 2(1+\beta-2\delta\beta)u_{j+1} - (1-\delta)\beta u_{j+2}\Big]\,.$$

A Taylor expansion around point $(x_j,\, t^n)$ leads to:

$$\frac{\Phi_{j+\frac{1}{2}} - \Phi_{j-\frac{1}{2}}}{h} = a\left[\frac{\partial u}{\partial x}(x_j,\, t^k) + (1-3\beta)\frac{h^2}{6}\frac{\partial^3 u}{\partial x^3}(x_j,\, t^k) + \delta\beta\frac{h^3}{4}\frac{\partial^4 u}{\partial x^4}(x_j,\, t^k)\right] + \mathcal{O}(h^4)\,.$$

We note this spatial differential operator:

$$\mathcal{L}(t)(u) = -a\Big[\frac{\partial u}{\partial x}(\cdot,\, t) + (1-3\beta)\frac{h^2}{6}\frac{\partial^3 u}{\partial x^3}(\cdot,\, t^n) + \delta\beta\frac{h^3}{4}\frac{\partial^4 u}{\partial x^4}(\cdot,\, t^n)\Big] + \mathcal{O}(h^4)\,. \tag{2.6}$$

If $(\lambda^l(t))_l$ are the eigenvalues of $\mathcal{L}(t)$ and $(v^l(\cdot,t))_l$ their associated eigenvectors, u can be decomposed on this eigen-basis as:

$$u(x,\, t) \;=\; \sum_l u^l(t)\, v^l(x,\, t)\,.$$

Reinserting this decomposition into the initial equation, we get:

$$\sum_l \left(\frac{\mathrm{d}u^l}{\mathrm{d}t} + \lambda^l\, u^l\right) v^l = 0\,.$$

As $(v^l)_l$ forms an independent family of functions, this implies that:

$$\frac{\mathrm{d}u^l}{\mathrm{d}t} \;+\; \lambda^l(t)\, u^l = 0, \quad \forall\, l\,.$$

The exact solution of this equation is:

$$u^l(t) \;=\; u^l(t^k)\exp\Big[\lambda^l(t)(t-t^k)\Big]\,.$$

As RKSSP(3,3) is a third-order scheme, this solution is approximated at order 3 then, according to the series development of the exponential, the exact solution u^l of the above linear ODE writes:

$$u^l(\cdot,\, t^{k+1}) = u^l(\cdot,\, t^k)\left(1 + \lambda^l(t)\tau + \frac{\left(\lambda^l(t)\tau\right)^2}{2} + \frac{\left(\lambda^l(t)\tau\right)^3}{6}\right) + \mathcal{O}(\tau^4)\,.$$

$$\begin{aligned}\frac{u^l(\cdot,\, t^{k+1}) - u^l(\cdot,\, t^k)}{\tau} &= \lambda^l(t) + \frac{\tau}{2}\left(\lambda^l(t)\right)^2 + \frac{\tau^2}{6}\left(\lambda^l(t)\right)^3 \\ &= \frac{\partial u^l}{\partial t}(\cdot,\, t^k) + \frac{\tau}{2}\frac{\partial^2 u^l}{\partial t^2}(\cdot,\, t^k) + \frac{\tau^2}{6}\frac{\partial^3 u^l}{\partial t^3}(\cdot,\, t^k) + \frac{\tau^3}{24}\frac{\partial^4 u^l}{\partial t^4}(\cdot,\, t^k) + \mathcal{O}(\tau^4)\,.\end{aligned}$$

Besides, the $(\lambda^l)^2$ and $(\lambda^l)^3$ are exactly the eigenvalues of operators $\mathcal{L}^2(t)$ and $\mathcal{L}^3(t)$, respectively, thus, reconstructing the solution, we get:

$$\mathcal{L}(t^k) + \frac{\tau}{2}\mathcal{L}^2(t^k) + \frac{\tau^2}{6}\mathcal{L}^3(t^k) = \frac{\partial u}{\partial t}(\cdot,\, t^k) + \frac{\tau}{2}\frac{\partial^2 u}{\partial t^2}(\cdot,\, t^k) + \frac{\tau^2}{6}\frac{\partial^3 u}{\partial t^3}(\cdot,\, t^k) + \frac{\tau^3}{24}\frac{\partial^4 u}{\partial t^4}(\cdot,\, t^k) + \mathcal{O}(\tau^4)\,. \tag{2.7}$$

Operators $\mathcal{L}^2(t)$ and $\mathcal{L}^3(t)$ can be computed using Expression (2.6):

$$\mathcal{L}^2(t^k) = a^2\left[\frac{\partial^2}{\partial x^2} + (1-3\beta)\frac{h^2}{3}\frac{\partial^4}{\partial x^4} + c^2\delta\beta\frac{h^3}{2}\frac{\partial^5}{\partial x^5}\right] + \mathcal{O}(h^4)\,,$$

$$\mathcal{L}^3(t^k) = -a^3\left[\frac{\partial^3}{\partial x^3} + (1-3\beta)\frac{h^2}{2}\frac{\partial^5}{\partial x^5} + \frac{\delta\beta h^3}{4}\frac{\partial^6}{\partial x^6}\right] + \mathcal{O}(h^4)\,.$$

Substituting $\mathcal{L}^2(t)$ and $\mathcal{L}^3(t)$ in (2.7) and removing terms of order strictly higher than 3, we get:

$$-a\left[\frac{\partial u}{\partial x}(\cdot,\, t^k) + (1-3\beta)\frac{h^2}{6}\frac{\partial^3 u}{\partial x^3}(\cdot,\, t^k) + \delta\beta\frac{h^3}{4}\frac{\partial^4 u}{\partial x^4}(\cdot,\, t^k)\right] + a^2\frac{\tau}{2}\left[\frac{\partial^2 u}{\partial x^2}(\cdot,\, t^k) + (1-3\beta)\frac{h^2}{3}\frac{\partial^4 u}{\partial x^4}(\cdot,\, t^k)\right]$$

$$-a^3\frac{\tau^2}{6}\frac{\partial^3 u}{\partial x^3} + \mathcal{O}(h^4) = \frac{\partial u}{\partial t}(\cdot,\, t^k) + \frac{\tau}{2}\frac{\partial^2 u}{\partial t^2}(\cdot,\, t^k) + \frac{\tau^2}{6}\frac{\partial^3 u}{\partial t^3}(\cdot,\, t^k) + \frac{\tau^3}{24}\frac{\partial^4 u}{\partial t^4}(\cdot,\, t^k) + \mathcal{O}(\tau^4)\,.$$

If a low dissipation scheme is used, β is taken equal to $1/3$ and:

$$\left(\frac{\partial u}{\partial t}(\cdot,\, t^k) + a\frac{\partial u}{\partial x}(\cdot,\, t^k)\right) + \frac{\tau}{2}\left(\frac{\partial^2 u}{\partial t^2}(\cdot,\, t^k) - a^2\frac{\partial^2 u}{\partial x^2}(\cdot,\, t^k)\right) + \frac{\tau^2}{6}\left(\frac{\partial^3 u}{\partial t^3}(\cdot,\, t^k) + a^3\frac{\partial^3 u}{\partial x^3}(\cdot,\, t^k)\right)$$

$$+ a\delta\frac{h^3}{12}\frac{\partial^4 u}{\partial x^4}(\cdot,\, t^k) + \frac{\tau^3}{24}\frac{\partial^4 u}{\partial t^4}(\cdot,\, t^k) + \mathcal{O}(\tau^4) + \mathcal{O}(h^4) = 0$$

From Equation (2.4), we deduce:

$$\frac{\partial^2 u}{\partial t^2}(\cdot,\, t^k) = a^2\frac{\partial^2 u}{\partial x^2}(\cdot,\, t^k)\,, \quad \frac{\partial^3 u}{\partial t^3}(\cdot,\, t^k) = -a^3\frac{\partial^3 u}{\partial x^3}(\cdot,\, t^k)\,,$$

which leads to:

$$\varepsilon_{h,\tau}(\cdot,\, t^k) = a\,\delta\frac{h^3}{12}\frac{\partial^4 u}{\partial x^4}(\cdot,\, t^k) + \frac{\tau^3}{24}\frac{\partial^4 u}{\partial t^4}(\cdot,\, t^k) + \mathcal{O}(\tau^4) + \mathcal{O}(h^4)\,.$$

As we also have:

$$\frac{\partial^4 u}{\partial t^4}(\cdot, t^k) \;=\; a^4 \frac{\partial^4 u}{\partial x^4}(\cdot, t^k)\,,$$

the following truncation errors are obtained:

$$\varepsilon_\tau^t(\cdot, t^k) \;=\; \tfrac{CFL^3}{2\delta}\, \varepsilon_h^s(\cdot, t^k)$$

$$\varepsilon_{h,\tau}(\cdot, t^k) \;=\; \left(1 + \tfrac{1}{2\delta}\left(\tfrac{a\tau}{h}\right)^3\right) \varepsilon_h^s(\cdot, t^k) \;+\; \mathcal{O}(\tau^4, h^4)\,.$$

Finally, we deduce that again, the temporal truncation error is bounded by the spatial truncation error under CFL condition $a\,\frac{\tau}{h} \leq 1$.

Actually, this result can be shown for Problem (2.4) with any spatial and temporal discretizations forming a stable scheme under CFL condition using the *Equivalent Differential Equation* method (see [Hirsch 1988a] for further details on this method). Incidentally, according to the *Equivalence Theorem of Lax*, a proof of which can be found in [Richtmyer 1967], for a linear, consistent and stable numerical scheme, controlling the truncation error comes to controlling the approximation error. Thus, the control of the *spatial* truncation error enables to control the approximation error, at least for this simple problem.

2.2.2 Illustration of the influence of the CFL number on the space-time approximation error

For problems more complex that the one-dimensional transport equation, the influence of the CFL condition on the approximation error can be shown numerically. A three-dimensional Sod shock tube problem, modeled by the Euler equations, is considered.
The computational domain is $\Omega = [0,\, 1] \times [-0.025,\, 0.025] \times [-0.025,\, 0.025]$. At initial time, the solution is discontinuous and the equation of the surface of discontinuity is $x = 0.5$. The following initial conditions are considered on both sides of the interface, with heat capacity ratio $\gamma = 1.4$:

$$W_{left} = (\rho = 1,\, \mathbf{u} = 0,\, p = 1)\;, \quad W_{right} = (\rho = 0.125,\, \mathbf{u} = 0,\, p = 0.1)\;,$$

where p is the pressure of the fluid and ρ is the volume mass or density of the fluid. A Harten-Lax-van Leer Contact wave (HLLC) approximate Riemann solver has been chosen, coupled with the V6 low-dissipation MUSCL reconstruction introduced in [Debiez 2000] and the Dervieux limiter described in [Cournède 2006b], see Chapter 5. A first order Backward Differentiation Formula implicit scheme is used for both $CFL > 1$ and $CFL < 1$, see Appendix C. Thanks to these choices, the global error is assumed to give a better understanding of the influence of the CFL number on space-time errors.
The numerical solution is analyzed at dimensionless time $t = 1$ for different values of the CFL number and compared to the analytical solution, which is computed using the *Method of Characteristics*, see [Hirsch 1988b] for more details. These results are presented in Figure 2.4. The solution is computed using a three-dimensional solver. However, using the axis symmetry around axis $\mathbf{e_x}$, it is sufficient to the solution as a function of x only. On the one hand, in the case

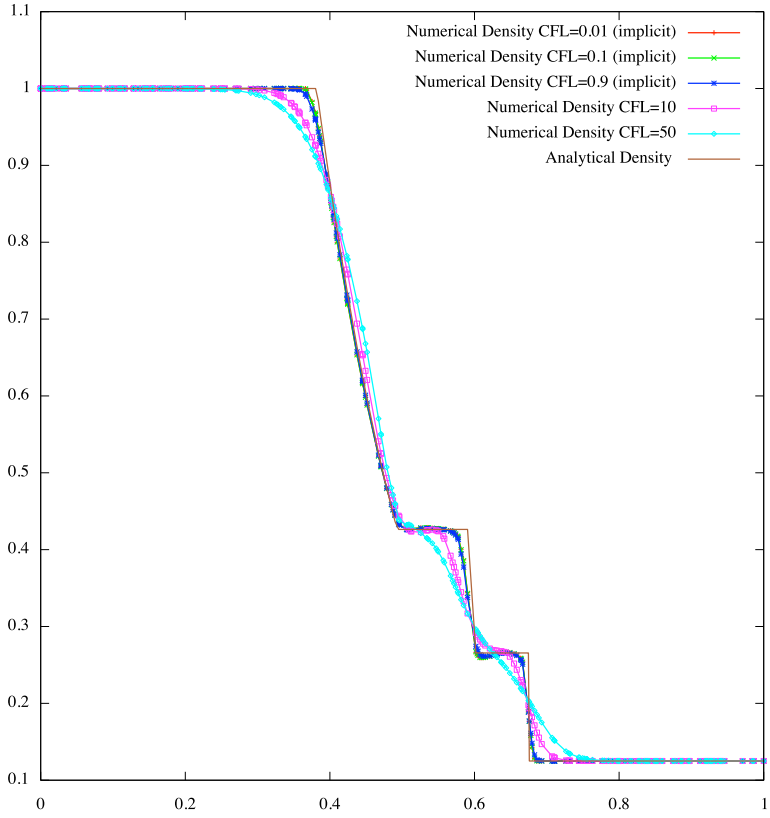


Figure 2.4: Solution of a Sod Shock tube problem at time $t = 1$ for different CFL numbers. The value of the solution density is represented on vertical axis as a function of spatial coordinate x. For $CFL \leq 1$, curves are almost identical while temporal dissipation is clearly visible for $CFL > 1$.

$CFL > 1$, the higher the CFL number, the bigger the approximation error. For $CFL = 50$ (blue curve), accuracy has been lost especially in shock waves areas. Increasing the CFL number actually comes to increase the artificial *temporal* dissipation of the numerical scheme. Therefore, even if implicit schemes exhibit good stability properties, which theoretically allows the use of arbitrary large CFL numbers, increasing the resolution time step is always done at the price of accuracy.

On the other hand, in the case $CFL \leq 1$, decreasing the CFL number has almost no influence on the accuracy of the solution: curves for $CFL = 0.01$ to $CFL = 0.9$ are almost superposed and very near from the exact solution. It can hence be inferred that the result of the previous truncation error analysis is still true for the approximation error obtained on this multi-dimensional non-linear problem: for $CFL \leq 1$, the temporal approximation error is dominated by the spatial one. The approximation error due to spatial discretization remaining the same for all the simulations, the time error is controlled by the same quantity in all these simulations, leading to almost superposed curves. Incidentally, Figure 2.5 shows that the influence of the scheme, implicit or explicit, is almost negligible when $CFL < 1$.

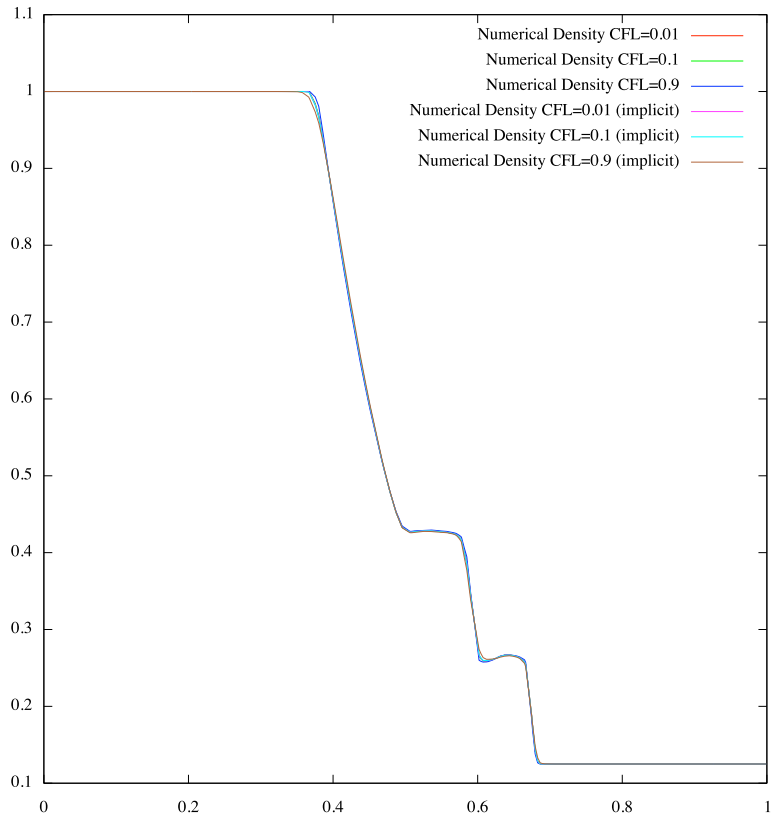


Figure 2.5: Comparison between the numerical solution obtained using an explicit or an implicit scheme using same CFL numbers 0.01, 0.1 and 0.9. A slight numerical dissipation introduced due to the implicit scheme is visible for the expansion fan, whereas the contact discontinuity and the shock wave are captured with the same accuracy in both cases.

To conclude, this numerical analysis tends to show that the optimal CFL number, *i.e.* the one realizing the best trade-off between accuracy and the time discretization complexity (number of time steps), is reached at $CFL = 1$.

2.2.3 Multi-dimensional theoretical analysis of the interpolation error

As explained in Section 2.1.3, only time-advancing schemes are considered, which restricts the set of authorized space-time meshes to "time-advancing" meshes. This strong constraint has to be enforced in the adaptation strategy, which precludes a direct extension of our steady adaptation algorithm to space-time computations. Figure 2.6 shows the difference between a "true" space-time adaptation and the restricted adaptation process we are forced to consider.

We now explain how this constraint is taken into account in the error estimate. If a true space-time adaptation is considered, the extension of the continuous mesh theory described in Chapter I is straightforward. Indeed, time is just an additional dimension which works exactly like any other spatial dimension. Therefore, all the formalism, which has been set for any arbitrary

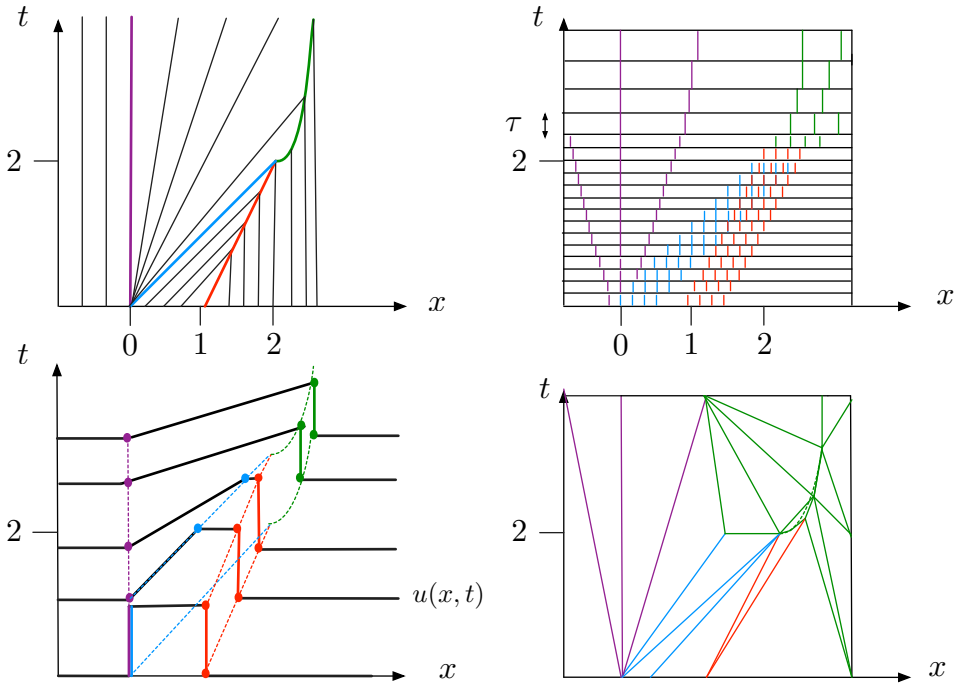


Figure 2.6: General space-time adaptation v.s "time-advancing" adaptation. Bottom left, a one dimensional solution $u = u(x,\, t)$ of a non-linear hyperbolic problem. Top left, its characteristic diagram in (x, t). Bottom right, representation of a "true" space-time anisotropic adaptation for this solution. Top right, representation of a time-advancing adaptation strategy for which the mesh is adapted at each time step.

dimension, is still valid. Notably, the space-time continuous interpolation error for any given two-times-differentiable function u and any given space-time continuous mesh $\mathbf{M} = (\mathcal{M}(\mathbf{x}, t))_{\Omega\times[0,T]}$ is:

$$\begin{aligned} e^{st}_{\mathcal{M}}(\mathbf{x}, t) \;=\; (u - \pi_{\mathcal{M}} u)(\mathbf{x}, t) \;&=\; c_{n+1}\,\mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},\, t)\,|H_u(\mathbf{x},\, t)|\,\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},\, t)\right) \\ &=\; c_{n+1}\sum_{i=1}^{n+1} \lambda_i^{-1}(\mathbf{x},\, t)^t \mathbf{v}_i(\mathbf{x},\, t)\,|H_u(\mathbf{x},\, t)|\,\mathbf{v}_i(\mathbf{x},\, t) \end{aligned}$$

where H_u is the space-time Hessian of function u and $(\lambda_i)_{i=1,n+1}$ are the eigenvalues of $\mathcal{M}(\mathbf{x}, t)$. However, as only time-advancing meshes must be considered, the continuous mesh must be such that $\mathbf{e}_{n+1} = \mathbf{e_t}$, *i.e.* the direction corresponding to time is an eigen-direction of $\mathcal{M}(\mathbf{x},\, t)$, for all $(\mathbf{x},\, t)$. This means that the space-time metric writes:

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

$$\text{where}\quad \Lambda = \mathrm{diag}\,(\lambda_1,\,\ldots,\,\lambda_n,\,\lambda_t) \quad\text{and}\quad \mathcal{R} = (\,\mathbf{r_1}\,|\,\mathbf{r_2}\,|\,\ldots\,|\,\mathbf{r_n}\,|\,\mathbf{e_t}\,)^T\,,$$

and we have noted λ_t the eigenvalue associated with $\mathbf{e_t}$. $h_i(\mathbf{x},\, t) = \lambda_i(\mathbf{x},\, t)^{-\frac{1}{2}}$, $i \in [\![1,\, n]\!]$ and $\tau(\mathbf{x},\, t) = \lambda_t(\mathbf{x},\, t)^{-\frac{1}{2}}$ are the spatial and the time sizes associated with $\mathcal{M}(\mathbf{x},\, t)$, respectively. As

$\mathbf{e_t} = (0, \dots, 0, 1) \in \mathbb{R}^{n+1}$, the space-time metric associated with a continuous time-advancing mesh writes:

$$\mathcal{M}^{st}(\mathbf{x}, t) = \left[\begin{array}{c|c} \mathcal{M}^s(\mathbf{x}, t) & 0 \\ \hline 0 & \tau^{-2}(\mathbf{x}, t) \end{array}\right],$$

and we have noted $\mathcal{M}^s$ the spatial part of the space-time metric:

$$\mathcal{M}^s(\mathbf{x}, t) = \mathcal{R}^s(\mathbf{x}, t)\, \Lambda^s(\mathbf{x}, t)\, (\mathcal{R}^s(\mathbf{x}, t))^T,$$

$$\text{where} \quad \Lambda^s = \text{diag}\,(\lambda_1, \dots, \lambda_n) \quad \text{and} \quad \mathcal{R}^s = (\,\mathbf{r_1}\,|\,\mathbf{r_2}\,|\,\dots\,|\,\mathbf{r_n}\,)^T.$$

The continuous space-time interpolation error model then reads:

$$e^{st}_{\mathcal{M}}(\mathbf{x}, t) \;=\; c_{n+1}\,\text{trace}\left(\left[\begin{array}{c|c} (\mathcal{M}^s)^{-\frac{1}{2}} & 0 \\ \hline 0 & \tau \end{array}\right]\left[\begin{array}{c|c} |H^s_u| & V \\ \hline V^T & u_{tt} \end{array}\right]\left[\begin{array}{c|c} (\mathcal{M}^s)^{-\frac{1}{2}} & 0 \\ \hline 0 & \tau \end{array}\right]\right),$$

where

$$H^s_u \;=\; H^s_u(\mathbf{x}, t) = \begin{pmatrix} \frac{\partial^2 u}{\partial x_1^2}(\mathbf{x}, t) & \dots & \frac{\partial^2 u}{\partial x_1 \partial x_n}(\mathbf{x}, t) \\ \vdots & \vdots & \vdots \\ \frac{\partial^2 u}{\partial x_n \partial x_1}(\mathbf{x}, t) & \dots & \frac{\partial^2 u}{\partial x_n^2}(\mathbf{x}, t) \end{pmatrix}$$

$$\text{and} \quad V = V(\mathbf{x}, t) = \left(\frac{\partial^2 u}{\partial x_1 \partial t}(\mathbf{x}, t), \dots, \frac{\partial^2 u}{\partial x_n \partial t}(\mathbf{x}, t)\right)^T.$$

$$\begin{aligned} e^{st}_{\mathcal{M}}(\mathbf{x}, t) \;&=\; c_{n+1}\,\text{trace}\left(\left[\begin{array}{c|c} (\mathcal{M}^s)^{-\frac{1}{2}}\,|H^s_u|\,(\mathcal{M}^s)^{-\frac{1}{2}} & \tau\,(\mathcal{M}^s)^{-\frac{1}{2}}\,V \\ \hline \tau V^T\,(\mathcal{M}^s)^{-\frac{1}{2}} & \tau^2 u_{tt} \end{array}\right]\right) \\ &= \text{trace}\left((\mathcal{M}^s)^{-\frac{1}{2}}(\mathbf{x}, t)\,|H^s_u|(\mathbf{x}, t)\,(\mathcal{M}^s)^{-\frac{1}{2}}(\mathbf{x}, t)\right) + \tau^2(\mathbf{x}, t)\,u_{tt}(\mathbf{x}, t) \\ &= \sum_{i=1}^{n} h_i^2(\mathbf{x}, t)\,\mathbf{r_i}^T(\mathbf{x}, t)\,|H^s_u|(\mathbf{x}, t)\,\mathbf{r_i}(\mathbf{x}, t) + \tau^2(\mathbf{x}, t)\,u_{tt}(\mathbf{x}, t). \end{aligned}$$

From now on, we consider $n = 3$ for clarity purpose. Still following the continuous mesh analysis [Loseille 2010b], the continuous mesh density $d = (h_1 h_2 h_3 \tau)^{-1}$ and the first three anisotropic quotients $\kappa_i = \frac{h_i^4}{h_1 h_2 h_3 \tau}$, $i \in [\![1, 3]\!]$ are considered instead of (h_1, h_2, h_3, τ). The one-to-one mapping is then given by:

$$h_i = d^{-\frac{1}{4}} \kappa_i^{\frac{1}{4}},\ \forall\, i \in [\![1, 3]\!] \quad \text{and} \quad \tau = (\kappa_1\,\kappa_2\,\kappa_3\,d)^{-\frac{1}{4}}.$$

Finally, the local continuous space-time interpolation error reads:

$$e^{st}_{\mathcal{M}} \;=\; d^{-\frac{1}{2}}\left(\kappa_1^{\frac{1}{2}}\gamma_1 + \kappa_2^{\frac{1}{2}}\gamma_2 + \kappa_3^{\frac{1}{2}}\gamma_3 + (\kappa_1\,\kappa_2\,\kappa_3)^{-\frac{1}{2}}\,u_{tt}\right),$$

and $(\gamma_l)_{l\in[\![1,3]\!]}$ are the eigenvalues of $|H_u^s|$.

The mesh adaptation problem consists in finding the optimal continuous mesh minimizing the previous space-time error model in $\mathbf{L}^p$ norm. Therefore, the following global optimization problem must be solved:

$$\text{Find } \mathbf{M}^{st}_{\mathbf{L}^p} \text{ such that } \mathbf{M}^{st}_{\mathbf{L}^p} = \underset{\mathbf{M}^{st}}{\operatorname{argmin}}\, E^{st}_{\mathbf{L}^p}(\mathbf{M}^{st}) \quad \text{with} \quad E^{st}_{\mathbf{L}^p}(\mathbf{M}^{st}) = \left(\int_0^T \int_\Omega \left(e^{st}_{\mathcal{M}^{st}}(\mathbf{x},\, t) \right)^p \, \mathrm{d}\mathbf{x}\, \mathrm{d}t \right)^{\frac{1}{p}}, \tag{2.8}$$

under the following constraint on the space-time complexity:

$$\mathcal{C}(\mathbf{M}^{st}) = \int_0^T \int_\Omega d(\mathbf{x},t)\, \mathrm{d}\mathbf{x}\, \mathrm{d}t = \int_0^T \int_\Omega \left(\prod_{i=1}^3 h_i^{-1}(\mathbf{x},t) \right) \tau^{-1}(\mathbf{x},t)\, \mathrm{d}\mathbf{x}\, \mathrm{d}t = N^{st}. \tag{2.9}$$

The variational analysis detailed in [Loseille 2010a] remains entirely valid in our case and the resolution of Problem (2.8-2.9) leads to the following optimal solution:

$$\mathcal{M}^{st}_{\mathbf{L}^p} = \begin{bmatrix} \mathcal{M}^s_{\mathbf{L}^p} & 0 \\ 0 & \lambda_{t,\mathbf{L}^p} \end{bmatrix},$$

$$\text{with } \mathcal{M}^s_{\mathbf{L^p}} = D_{\mathbf{L^p}} \left(\det |H_u^s|\, |u_{tt}| \right)^{-\frac{1}{2p+4}} |H_u^s|, \quad \lambda_{t,\mathbf{L^p}} = D_{\mathbf{L^p}} \left(\det |H_u^s|\, |u_{tt}| \right)^{-\frac{1}{2p+4}} |u_{tt}|$$

$$\text{and} \quad D_{\mathbf{L^p}} = \left(N^{st}\right)^{\frac{1}{2}} \left(\int_0^T \int_\Omega \left(\det |H_u^s|\, |u_{tt}| \right)^{\frac{p}{2p+4}} \mathrm{d}\mathbf{x}\, \mathrm{d}t \right)^{-\frac{1}{2}}. \tag{2.10}$$

The optimal sizes in space and time for a three-dimensional spatial domain are then given by:

$$\boxed{\begin{aligned} h_i^{opt} &= D_{\mathbf{L^p}}^{-\frac{1}{2}} \left[|u_{tt}| \prod_{l=1}^3 \gamma_l \right]^{\frac{1}{2(2p+4)}} \gamma_i^{-\frac{1}{2}} = D_{\mathbf{L^p}}^{-\frac{1}{2}} \left[|u_{tt}| \det ||H_u^s| \right]^{\frac{1}{2(2p+4)}} \gamma_i^{-\frac{1}{2}}, \\ \tau^{opt} &= D_{\mathbf{L^p}}^{-\frac{1}{2}} \left[|u_{tt}| \prod_{l=1}^3 \gamma_l \right]^{\frac{1}{2(2p+4)}} |u_{tt}|^{-\frac{1}{2}} = D_{\mathbf{L^p}}^{-\frac{1}{2}} \left[|u_{tt}| \det ||H_u^s| \right]^{\frac{1}{2(2p+4)}} |u_{tt}|^{-\frac{1}{2}}. \end{aligned}} \tag{2.11}$$

and γ_l still denotes the eigenvalue of the "pure" spatial Hessian associated with eigen-direction $\mathbf{r_l}$.

It can finally be demonstrated that the global $\mathbf{L}^p$ interpolation errors in space and in time on continuous space-time mesh $\mathcal{M}^{st}_{\mathbf{L^p}}$ are given by:

$$E^s_{\mathbf{L}^p}(\mathcal{M}^{st}_{\mathbf{L^p}}) = \left(\int_0^T \int_\Omega \left[e^s(\mathbf{x},\, t) \right]^p \mathrm{d}\mathbf{x}\, \mathrm{d}t \right)^{\frac{1}{p}} = 3(N^{st})^{-\frac{1}{2}} \left(\int_0^T \int_\Omega \left(|u_{tt}| \det |H_u^s| \right)^{\frac{p}{2p+4}} \mathrm{d}\mathbf{x}\, \mathrm{d}t \right)^{\frac{p+2}{2p}}, \tag{2.12}$$

$$E^t_{\mathbf{L}^p}(\mathcal{M}^{st}_{\mathbf{L^P}}) \;=\; \left(\int_0^T\int_\Omega \left[e^t(\mathbf{x},\,t)\right]^p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t\right)^{\frac{1}{p}} \;=\; (N^{st})^{-\frac{1}{2}} \left(\int_0^T\int_\Omega (|u_{tt}|\,\det|H^s_u|)^{\frac{p}{2p+4}}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\right)^{\frac{p+2}{2p}}, \tag{2.13}$$

$$\begin{aligned} E^{st}_{\mathbf{L}^p}(\mathcal{M}^{st}_{\mathbf{L^P}}) \;&=\; \left(\int_0^T\int_\Omega \left[e^{st}(\mathbf{x},\,t)\right]^p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t\right)^{\frac{1}{p}} \;=\; 4\,(N^{st})^{-\frac{1}{2}} \left(\int_0^T\int_\Omega (\det|H^s_u|\,|u_{tt}|)^{\frac{p}{2p+4}}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\right)^{\frac{p+2}{2p}} \\ &=\; E^s_{\mathbf{L}^p}(\mathcal{M}^{st}_{\mathbf{L^P}}) + E^t_{\mathbf{L}^p}(\mathcal{M}^{st,\,opt}_{\mathbf{L^P}})\,. \end{aligned} \tag{2.14}$$

This shows that, once again, the adaptation process tends to equi-distribute and to decouple the error in the different directions.

Link with the CFL condition. Let us consider the following n-dimensional linear advection problem, with $\mathbf{a}$ independent of t and $\mathbf{x}$:

$$\frac{\partial u}{\partial t} \;+\; \mathbf{a}\cdot\nabla_{\mathbf{x}} u \;=\; 0\,, \quad \text{with } \mathbf{a} = (a_1,\,\ldots,\,a_n)\,.$$

Then, the following equation holds: $\dfrac{\partial^2 u}{\partial t^2} \;-\; \mathbf{a}^T\,H^s_u\,\mathbf{a} = 0$. Indeed,

$$\begin{aligned} &\frac{\partial u}{\partial t} \;+\; \sum_{i=1}^n a_i\,\frac{\partial u}{\partial x_i} \;=\; 0 \underbrace{\Longrightarrow}_{a_1\frac{\partial}{\partial x_1}} a_1\frac{\partial^2 u}{\partial x_1\partial t} \;+\; \sum_{i=1}^n a_1 a_i\,\frac{\partial^2 u}{\partial x_1\partial x_i} \;=\; 0 \\ \Longrightarrow\;& \frac{\partial}{\partial t}\left(-\frac{\partial u}{\partial t} \;-\; \sum_{i=2}^n a_i\,\frac{\partial u}{\partial x_i}\right) \;+\; \sum_{i=1}^n a_1 a_i\,\frac{\partial^2 u}{\partial x_1\partial x_i} \;=\; 0 \\ \Longrightarrow\;& -\frac{\partial^2 u}{\partial t^2} \;+\; a_1^2\frac{\partial^2 u}{\partial x_1^2} \;+\; \sum_{i=2}^n a_i\,\frac{\partial}{\partial x_i}\left(-\frac{\partial u}{\partial t} \;+\; a_1\frac{\partial u}{\partial x_1}\right) \;=\; 0 \\ \Longrightarrow\;& -\frac{\partial^2 u}{\partial t^2} \;+\; a_1^2\frac{\partial^2 u}{\partial x_1^2} \;+\; \sum_{i=2}^n a_i\,\frac{\partial}{\partial x_i}\left(\sum_{j=1}^n a_j\,\frac{\partial u}{\partial x_j} \;+\; a_1\frac{\partial u}{\partial x_1}\right) \;=\; 0 \\ \Longrightarrow\;& -\frac{\partial^2 u}{\partial t^2} \;+\; a_1^2\frac{\partial^2 u}{\partial x_1^2} \;+\; \sum_{i,j=2}^n a_i a_j\,\frac{\partial^2 u}{\partial x_i\partial x_j} \;+\; \sum_{i=2}^n a_1 a_i\,\frac{\partial^2 u}{\partial x_1\partial x_i} \;=\; 0 \\ \Longrightarrow\;& -\frac{\partial^2 u}{\partial t^2} \;+\; \sum_{i,j=1}^n a_i a_j\,\frac{\partial^2 u}{\partial x_i\partial x_j} \;=\; 0 \;\Longrightarrow\; \frac{\partial^2 u}{\partial t^2} \;-\; \mathbf{a}^T\,H^s_u\,\mathbf{a} = 0\,. \end{aligned}$$

$\mathbf{a}$ can be decomposed on orthonormal basis $(\mathbf{r_1},\,\ldots,\,\mathbf{r}_n)$: $\mathbf{a} = \displaystyle\sum_{i=1}^n (\mathbf{a}\cdot\mathbf{r_i})\;\mathbf{r_i}$. Thus,

$$u_{tt} \;-\; \sum_{i=1}^n (\mathbf{a}\cdot\mathbf{r_i})^2\;\gamma_i = 0\,.$$

Besides, from Relation (2.11), we have:

$$\frac{h_i^{opt}}{\tau^{opt}} = \left(\frac{\gamma_i}{u_{tt}}\right)^{-\frac{1}{2}}\,. \tag{2.15}$$

Therefore:

$$\sum_{i=1}^{n} (\mathbf{a}\cdot\mathbf{r_i})^2 \, \frac{\gamma_i}{u_{tt}} = 1 \iff \sum_{i=1}^{n} \left[(\mathbf{a}\cdot\mathbf{r_i}) \, \frac{\tau^{opt}}{h_i^{opt}} \right]^2 = 1 \,,$$

which can be considered as an extension of the CFL condition on unstructured multi-dimensional continuous meshes. Indeed,

$$\sum_{i=1}^{n} \left[(\mathbf{a}\cdot\mathbf{r_i}) \, \frac{\tau^{opt}}{h_i^{opt}} \right]^2 = 1 \iff \left(||\mathbf{a}||\, \tau^{opt} \right)^2 \, || \, (\mathcal{M}^s)^{\frac{1}{2}} \, \frac{\mathbf{a}}{||\mathbf{a}||} ||^2 = 1 \,. \tag{2.16}$$

To interpret the term $||\mathcal{M}^{\frac{1}{2}} \frac{\mathbf{a}}{||\mathbf{a}||}||$, let us compute the spatial size $h(\mathbf{a})$ prescribed by the metric

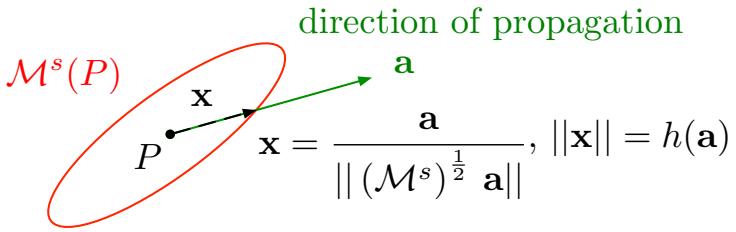


Figure 2.7: Geometrical interpretation of $h(\mathbf{a})$ as the size prescribed by metric $\mathcal{M}^s$ in the advection direction $\mathbf{a}$.

in direction $\mathbf{a}$. We must find $\mathbf{x}$ collinear to $\mathbf{a}$ and which is unit with respect to $\mathcal{M}^s$:

$$\text{Find } \mathbf{x} = \alpha \, \mathbf{a} \text{ such that } \mathbf{x}^T \, \mathcal{M} \, \mathbf{x} = 1$$

$$\implies \alpha^2 \mathbf{a}^T \, \mathcal{M}^s \, \mathbf{a} = 1 \implies \mathbf{x} = \frac{\mathbf{a}}{|| \, (\mathcal{M}^s)^{\frac{1}{2}} \, \mathbf{a}||} \,.$$

and consequently,

$$h(\mathbf{a}) = ||\mathbf{x}|| = \frac{||\mathbf{a}||}{|| \, (\mathcal{M}^s)^{\frac{1}{2}} \, \mathbf{a}||} \,.$$

A geometrical description of $h(\mathbf{a})$ is given in Figure 2.7. Therefore, Relation (2.16) leads to:

$$\frac{||\mathbf{a}||\, \tau^{opt}}{h(\mathbf{a})} = 1 \,,$$

which means that the optimal local continuous time step and spatial sizes are such that $CFL = 1$ in the propagation direction. This is in accordance with the analysis developed in Sections 2.2.1 and 2.2.2.

Remark 8. *As for the truncation error and the numerical approximation error, we see that for a linear advection problem under CFL condition, if the continuous representation of the mesh is the optimal space-time metric, the local temporal interpolation error is dominated by the local spatial interpolation error. Indeed, Relation (2.15) implies that:*

$$\frac{\tau^2}{h_i^2} = \frac{\gamma_i}{u_{tt}} \iff e^t(\mathbf{x},\, t) = \tau^2(\mathbf{x},\, t) u_{tt}(\mathbf{x},\, t) = h_i^2(\mathbf{x},\, t) \gamma_i(\mathbf{x},\, t) \,, \quad \forall \, i \in [\![1,\, n]\!] \; \forall \mathbf{x},\, t \,.$$

The temporal part of the local space-time interpolation error can therefore be rewritten as:

$$e^t(\mathbf{x},\,t) = \frac{1}{n}\sum_{i=1}^{n} \tau^2(\mathbf{x},\,t)\,u_{tt}(\mathbf{x},\,t) = \frac{1}{n}\sum_{i=1}^{n} h_i^2(\mathbf{x},\,t)\gamma_i(\mathbf{x},\,t) = \frac{1}{n}\,e^s(\mathbf{x},\,t)\,.$$

The local interpolation error then reads:

$$e_{\mathcal{M}}(\mathbf{x},\,t) = e^s(\mathbf{x},\,t) + e^t(\mathbf{x},\,t) = \frac{n+1}{n}\,e^s(\mathbf{x},\,t)\,.$$

Actually, this remark is even stronger: the local space-time interpolation error is not just dominated, it is proportional *to the spatial interpolation error. This results validates the fixed-point strategy, which consists in controlling the spatial interpolation error for* $[t_i,\,t_{i+1}]$ *to guarantee the control of the whole space-time interpolation error during this time interval.*

Remark 9. *As explained in Section 2.1.4.3, adapting the time step in the case of implicit time-advancing schemes appears as a good strategy to reduce computational cost while preserving time accuracy. The above result, i.e. the fact that the optimal local continuous time step and spatial sizes are such that* $CFL = 1$ *in the propagation direction, suggests that even for implicit computations, the local time step provided by the* CFL *condition is the optimal one. If this may be true for linear problems, this result remains to be proved for non-linear problems.*
Moreover, in the case of multi-physics simulation, the time step provided by the CFL condition is obviously not always the good one. For instance, let us consider the case of a combustion problem for which the Flame speed is about $30\,m.s^{-1}$ *and the fluid acoustic velocity is about* $300\,m.s^{-1}$*. The "good" time step the one associated with the flame speed and not the one dictated by the acoustics. In this case, the* CFL *condition associated with the acoustics limits the time step size although the capture of the acoustical details is absolutely unnecessary. A larger time step could have been taken while preserving the accuracy of quantities linked to deflagration. This simple example shows that the time step proposed by the* CFL *condition for unsteady simulations is not always the good one.*

2.3 Extension of multi-scale mesh adaptation to unsteady simulations.

This section is dedicated to the extension of multi-scale strategies, described in Section 1.4 for pseudo-steady simulations, to unsteady simulations.
In the previous section, a detailed analysis of the *local* error for unsteady problems, notably the local time errors, has been performed. It remains to explain how to locally equidistribute this error in all directions and how to control the integrated (in space and time) global error under a given complexity constraint.
In the transient fixed-point algorithm (see Section 2.1.4.4), the global spatial interpolation error is control in $\mathbf{L}^\infty$ norm both in space and time, *i.e* the following error is considered:

$$E^s = \max_{t\,\in\,[0,\,T]}\ \max_{\mathbf{x}\,\in\,\Omega_h} e^s(\mathbf{x},\,\mathbf{t})\,.$$

This section first proposes an extension of the transient fixed-point algorithm which enables to introduce other $\mathbf{L}^p$ norms, with $p \in [1, +\infty[$, at least for the spatial integration. In other words, the approach developed in this Thesis allows to control global interpolation errors as:

$$E^s = \max_{t \in [0,T]} ||e^s(\cdot, t)||_{\mathbf{L}^p} ..$$

Next, we attempt to use the results demonstrated in Section 2.2.3 to locally equidistribute not only the local *spatial* error $e^s(\mathbf{x}, t)$ but the complete local spatiotemporal error $e^{st}(\mathbf{x}, t)$.

2.3.1 Controlling the error during a whole adaptation sub-interval with a single adapted mesh in a multi-scale framework.

In the $\mathbf{L}^\infty - \mathbf{L}^\infty$ fixed-point algorithm, continuous mesh $\mathcal{M}_{i,\,\mathbf{L}^\infty-\mathbf{L}^\infty}$ associated with sub-interval $[t_i,\, t_{i+1}]$ ensures the control of the spatial interpolation error during this whole sub-interval thanks to the intersection procedure. This technique enables to always minimize the mesh size prescription according to the information gathered in the solution samples.

Of course, the same algorithm can be used for multi-scale unsteady adaptation, but a new problematic appears. The spatial Hessian associated with sample $\mathcal{S}_{i,k}$ is noted $|H^s_{i,k}|$. We assume that metric associated with sample $\mathcal{S}_{i,k}$ in the absence of any intersection procedure can be written under the form:

$$\mathcal{M}^*_{i,k} = D_{\mathbf{L}^p}\, \det |H^*_{i,k}|^{-\frac{1}{2p+n}}\, |H^*_{i,k}|\,, \tag{2.17}$$

where $H^*_{i,k}$ is a matrix built with $|u_{tt}(\mathbf{x}, t_{i,k})|$ and $|H^s_{i,k}| = |H^s(\mathbf{x}, t_{i,k})|$. Then, three possibilities exist for the intersection stage:

1. intersect pure Hessians $|H^*_{i,max}| = \bigcap_{k=1}^{n_k} |H^*_{i,k}|$ and use the eigenvalues of the resulting Hessian $\left(\gamma^*_{i,max}\right)_{i \in [\![1,n]\!]}$ inside optimal metric Formula (2.10) to get $\mathcal{M}_{i,max}$.

2. intersect locally normalized Hessians:

$$|H|^{\tilde{}}_{i,max} = \bigcap_{k=1}^{n_k} |\tilde{H}|^*_{i,k} = \bigcap_{k=1}^{n_k} \left[\det |H^*_{i,k}|\right]^{-\frac{1}{2p+n}}\, |H^*_{i,k}|\,,$$

   and normalize to enforce the complexity constraint and to obtain $\mathcal{M}_{i,max}$.

3. directly intersect complete metrics:

$$\mathcal{M}^*_{i,max} = \bigcap_{k=1}^{n_k} \mathcal{M}^*_{i,k} = \bigcap_{k=1}^{n_k} D_{\mathbf{L}^p} \det |H^*_{i,k}|^{-\frac{1}{2p+n}} |H^*_{i,k}|\,,$$

   and renormalize to enforce the complexity constraint.

To answer this issue, let us first introduce two properties satisfied by the intersection procedure (1.8):

$$\forall\ (\mathcal{A}_k)_{k\in[\![1,\,n_k]\!]}\ \text{ with } \mathcal{A}_k \text{ Symmetric Positive Definite :}$$

$$(i)\quad \det\mathcal{A}_i\ \leq\ \det\left(\bigcap_{k=1}^{n_k}\mathcal{A}_k\right),\quad \forall\, i\ \in[\![1,\,n_k]\!]\,,$$

$$(ii)\quad \bigcap_{k=1}^{n_k}\alpha\,\mathcal{A}_k=\alpha\bigcap_{k=1}^{n_k}\mathcal{A}_k\,,\quad \forall\,\alpha\ \in\mathbb{R}\,.$$

We immediately see that Strategies (2) and (3) are totally equivalent. Indeed, according to Property (ii), $|\tilde{H}|_{i,max}$ and $\mathcal{M}_{i,max}$ are identical *modulo* a scaling factor $D_{\mathbf{L^p}}$:

$$\mathcal{M}_{i,max}=\bigcap_{k=1}^{n_k}D_{\mathbf{L}^p}\det|H^*_{i,k}|^{-\frac{1}{2p+n}}|H^*_{i,k}|\underbrace{=}_{(ii)}D_{\mathbf{L}^p}\bigcap_{k=1}^{n_k}\det|H^*_{i,k}|^{-\frac{1}{2p+n}}|H^*_{i,k}|\,.$$

But the normalization procedure will remove this scaling factor: the resulting metrics are equal.

Note that in the case of the $\mathbf{L}^\infty-\mathbf{L}^\infty$ transient fixed-point, Alternative (1)-(2) did not even exist as pure maximal Hessian and the locally normalized maximal Hessian were identical - the local normalization term tends to 1 when $p\longrightarrow\infty$.

If Strategy (1) is used, according to the demonstration of Theorem 3 given in [Loseille 2010b], the local spatial interpolation error is equi-distributed in all directions and we obtain:

$$\begin{aligned}e^s_{\mathcal{M}_{i,max}}(\mathbf{x},\,t_{i,k})\ &=c_n\sum_{l=1}^{n}\left(\lambda^*_{l,i,max}\right)^{-1}(\mathbf{x},\,t)\gamma_l(\mathbf{x},\,t)=n\,c_n\left(\prod_{l=1}^{n}\lambda^*_{l,i,max}\right)^{-\frac{1}{n}}\left(\prod_{l=1}^{n}\gamma_{l,i,k}\right)^{\frac{1}{n}}\\&=n\,c_n\left(\frac{\det|H^s_{i,k}|}{\det\mathcal{M}^*_{i,max}}\right)^{\frac{1}{n}}\\&=\frac{n\,c_n}{D_{\mathbf{L}^p\mathbf{L}^\infty}}\left[\det|H^*_{i,max}|\right]^{-\frac{2p}{n(2p+n)}}\left[\det|H^s_{i,k}|\right]^{\frac{1}{n}},\end{aligned}\tag{2.18}$$

where $\left(\lambda^*_{l,i,max}\right)_{l\,\in\,[1,\,n]}$ are the eigenvalues of:

$$\mathcal{M}^*_{i,max}\ =\ D_{\mathbf{L}^p\mathbf{L}^\infty}\left(\det|H^*_{i,max}|\right)^{-\frac{1}{2p+n}}|H^*_{i,max}|\,,\quad\text{with}\qquad H^*_{i,max}\ =\ \bigcap_{k=1}^{n_k}|H^*_{i,k}|\,,$$

and $(\gamma_{l,i,k}(\mathbf{x}))_{l,i,k}$ are the eigenvalues of $H^s_{i,k}=|H^*(\mathbf{x},\,t_{i,k})|$.
Due to Property (i) of the intersection operator, $\det|H^*_{i,max}|\ \geq\ \det|H^*_{i,k}|$ and the following inequality holds:

$$e^s_{\mathcal{M}_{i,max}}(\mathbf{x},\,t_{i,k})\ \leq\ \frac{n\,c_n}{D_{\mathbf{L}^p\mathbf{L}^\infty}}\left[\det|H^*_{i,k}|\right]^{-\frac{2p}{n(2p+n)}}\left[\det|H^s_{i,k}|\right]^{\frac{1}{n}}.$$

The previous inequality can be rewritten as:

$$e^s_{\mathcal{M}_{i,max}}(\mathbf{x},\, t_{i,k}) \;\leq\; n\, c_n\, \frac{D_{\mathbf{L}^p}}{D_{\mathbf{L}^p\mathbf{L}^\infty}}\, \Big[\det \mathcal{M}^*_{i,k}\Big]^{-\frac{1}{n}}\, \Big[\det |H^s_{i,k}|\Big]^{\frac{1}{n}} \;=\; \frac{D_{\mathbf{L}^p}}{D_{\mathbf{L}^p\mathbf{L}^\infty}}\, e^s_{\mathcal{M}^*_{i,k}}(\mathbf{x},\, t_{i,k})\,, \quad \forall\, k \,\in\, [\![1,\, n_k]\!]\,,$$

with

$$\mathcal{M}^*_{i,k} \;=\; D_{\mathbf{L}^p}\, \left(\det |H^*_{i,k}|\right)^{-\frac{1}{2p+n}}\, |H^*_{i,k}|\,.$$

If there are enough solution samples, the above inequality can be assumed to be valid $\forall\, t \,\in\, [t_i,\, t_{i+1}]$, and not only for $t \,\in\, \{t_{i,0},\, \ldots,\, t_{i,n_k-1},\, t_{i+1}\}$. Finally, raising the previous inequality to power p and integrating in space and time, the following inequality on the global spatial interpolation error holds:

$$E^s_{\mathbf{L^p}(\Omega)}(\mathcal{M}^{s,*}_{i,max}) \leq \frac{D_{\mathbf{L}^p}}{D_{\mathbf{L}^p\mathbf{L}^\infty}}\, E^s_{\mathbf{L^p}(\Omega)}(\mathcal{M}^{s,*}_{i,k})\,, \quad \forall\, k \,\in\, [\![1,\, n_k]\!] \quad \text{and} \quad \forall\, p \,\in\, [1,\, +\infty]\,.$$

This means that the global spatial interpolation error obtained when continuous "intersected" mesh $\mathcal{M}^*_{i,max}$ is used is always smaller than the one observed with any of the continuous optimal meshes $\mathcal{M}^*_{i,k}$ corresponding to sampling times $(t_{i,k})_{k\,\in [\![0,\, n_k]\!]}$. Therefore, Strategy (1) guarantees that the spatial local and global interpolation errors are controlled *for all samples*. No such interpretation can be made for Strategy (2), which justifies the choice of Strategy (1) in our new multi-scale fixed-point algorithm.

2.3.2 Local space-time mesh complexity and global fixed-point strategy.

Problematics. The $\mathbf{L}^\infty - \mathbf{L}^\infty$ fixed-point strategy presented in [Alauzet 2007], which decouples completely the treatment of each adaptation sub-interval, cannot be transposed *as it* for multi-scale unsteady adaptation using $\mathbf{L}^p$ norms. Indeed, direct application of the transient fixed-point formula to the multi-scale unsteady adaptation case would consist in computing the non-dimensioned continuous mesh complexity as:

$$\mathcal{C}(\mathcal{M}_{\mathbf{L}^p\mathbf{L}^\infty}) \;=\; \int_\Omega \Big[\, \det |H_{i,\max}(\mathbf{x})| \,\Big]^{\frac{p}{2p+n}}\, \mathrm{d}\mathbf{x}\,. \tag{2.19}$$

As this quantities varies between each time adaptation sub-interval $[t_i,\, t_{i+1}]$, *i.e.*, the global normalization constant,

$$D_{\mathbf{L}^p\mathbf{L}^\infty} \;=\; N^{\frac{2}{n}} \left(\int_\Omega \Big[\, \det |H_{i,\max}(\mathbf{x})| \,\Big]^{\frac{1}{2}}\, \mathrm{d}\mathbf{x} \right)^{-\frac{2}{n}} \;=\; D_{i,\,\mathbf{L}^\infty - \mathbf{L}^\infty}\,,$$

is also different from one adaptation sub-interval to the other. Consequently, if new physical phenomena develop in the flow during the simulation, this integral grows and hence the global normalization constant diminishes. This means that a physical phenomenon which does not vary in time, such as a stationary shock defined by a step function, will be more or less refined depending on the physics of the whole flow. These meshing inconsistencies are illustrated in Figure 2.8 on a simple bump test case. This shows that the global normalization procedure has to be thought over to fit the multi-scale framework.

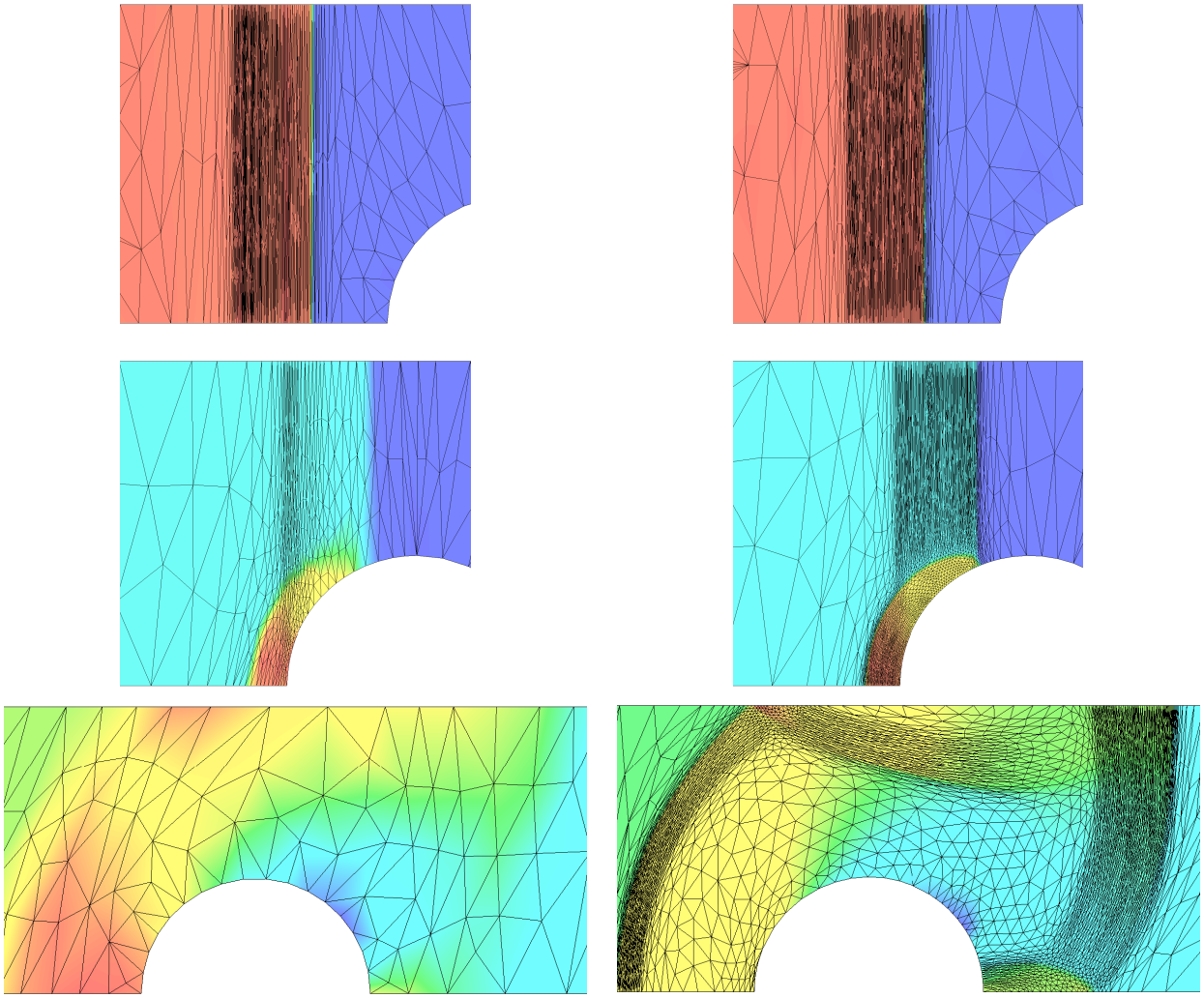


Figure 2.8: Illustration of meshing inconsistencies on a simple two-dimensional bump test case. This phenomenon appears when the local fixed-point algorithm 2.2 is used *as it is* for multi-scale unsteady adaptation, the evaluation of the complexity being local and given by Expression 2.19. Left, with local normalization. Right, with global optimization. Top, before shock wave interaction with the bump. Bottom, final solution.

Moreover, one of the main reproaches made to "frequent-re-meshings" methods in Section 2.1.4.2 was that the number of re-meshings performed during the adaptation process was totally unpredictable. This means that the computational effort for the whole simulation, *i.e.* the sum of the spatial complexities of all generated meshes, was left totally unmastered. Even if the fixed-point algorithm responds partially to this issue with the prescription of a target spatial complexity N^{ptfx} common to each mesh $\mathcal{H}_i$, the *total* computational effort, *i.e.* both in space and time, is not taken into account to get the optimal metric.

These weaknesses have motivated the extension of the notion of complexity to space-time

meshes and have led to the development of a global fixed-point strategy in which adaptation sub-intervals are treated in a coupled manner.

Space-time complexity. Meshing inconsistencies highlighted in Figure 2.8 suggest to change the complexity evaluation in the fixed-point algorithm. By analogy with spatial complexity Definition (1.14), the space-time complexity associated with a continuous space-time mesh $\mathbf{M}^{st}_{\mathbf{L}^p\mathbf{L}^\infty}$ is defined as:

$$\mathcal{C}(\mathbf{M}^{st}_{\mathbf{L}^p\mathbf{L}^\infty}) \;=\; \int_0^T \int_\Omega \sqrt{\det \mathcal{M}^{st}_{\mathbf{L}^p\mathbf{L}^\infty}(\mathbf{x},t)}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t \;=\; \sum_{i=1}^{n_{adap}} \int_{t_i}^{t_{i+1}} \int_\Omega \left[\prod_{l=1}^{n} h_{l,i,max}(\mathbf{x})\right]^{-1} \tau^{-1}(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\,. \tag{2.20}$$

This space-time complexity takes into account all the phenomena developing in the fluid during the whole simulation. Thus, the normalization constant is global and common to the n_{adap} intersected metrics, which enables to avoid meshing inconsistencies. This new global evaluation of the complexity forces us to restructure the fixed-point algorithm. The global normalization constant becomes:

$$D_{\mathbf{L}^\infty\,\mathbf{L}^p} \;=\; N^{\frac{2}{n+1}} \left[\sum_{i=1}^{n_{adap}} \int_{t_i}^{t_{i+1}} \int_\Omega \left[\prod_{l=1}^{n} h_{l,i,max}(\mathbf{x})\right]^{-1} \tau^{-1}(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\right]^{-\frac{2}{n+1}}. \tag{2.21}$$

Global fixed-point strategy. The price to pay is that a global fixed-point strategy is now mandatory. Indeed, the decoupling of the adaptation sub-intervals described in [Alauzet 2007] cannot be applied anymore as the evaluation of normalization Constant (2.21) requires the completion of the whole simulation (from 0 to T). Consequently, a new *global* fixed-point strategy, described in Figures 2.9 and 2.10, has been adopted.

Remark 10. *Actually, this* global *fixed-point strategy also naturally appears when adjoint-based unsteady mesh adaptation is considered, see [Belme 2010]. This tends to confirm that such strategy is intrinsically the good one when performing metric-based mesh adaptation for unsteady simulations.*

Algorithmic issues. First, the new fixed-point algorithm requires a lot more storage. Indeed, with the former methodology, once the intersected spatial Hessian $|H_{i,max}|$ was computed for one sub-interval, the complete metric $\mathcal{M}_{i,\,\mathbf{L}^\infty\mathbf{L}^\infty}$ for the current sub-interval was computed on the fly. This was possible because only the spatial complexity of this non-dimensioned metric was needed to calculate the global normalization constant. In the new algorithm, the complete metric cannot be computed on the fly as the whole space-time complexity is needed to compute the global normalization constant. It is therefore mandatory to store the $\{\,|H_{i,max}|\,\}_{i\,\in\,[\![1,\,n_{\mathrm{adap}}]\!]}$ and wait for the simulation completion to compute the list of complete optimal metrics $\{\mathcal{M}_{i,\,\mathbf{L}^\mathbf{p}\mathbf{L}^\infty}\}_{i\,\in\,[\![1,\,n_{\mathrm{adap}}]\!]}$.

Secondly, an important asset of this method lies in the possibility to parallelize the generation of the n_{adap} meshes in a straightforward manner, see Figure 2.10. The whole adaptation loop is actually run in parallel:

For $j = 1, \dots, n_{ptfx}$ (SEQUENTENTIAL)

  For $i = 1, \dots, n_{adap}$ (SEQUENTENTIAL)

    $\mathcal{S}^j_{i,0} = \textbf{InterpolateSolution}(\mathcal{H}^j_{i-1}, \mathcal{S}^j_{i-1}, \mathcal{H}^j_i)$

    $\{\mathcal{S}^j_{i,k}\}_{k\in[\![0,n_k]\!]} = \textbf{SolveState}(\mathcal{S}^j_{i,1}, \mathcal{H}^j_i)$

    For $k = 1, \dots, n_k$ (DISTRIBUTED PARALLELIZATION)

      $|H^j_{i,k}| = \texttt{ComputeSamplesHessians}(\mathcal{H}^j_i, \mathcal{S}^j_{i,k})$

    End for

    $|H^j_{i,max}| = \texttt{ComputeMaxHessianMetric}(\{|H^j_{i,k}|\}_{k\in[\![1,n_k]\!]})$

  End for

  $\mathcal{C}^j = \texttt{ComputeSpaceTimeComplexity}(\{|H^j_{i,\max}|\}_{i\in[\![1,n_{adap}]\!]})$

  For $i = 1, \dots, n_{adap}$ (DISTRIBUTED PARALLELIZATION)

    $\mathcal{M}^j_i = \texttt{ComputeUnsteadyLpMetrics}(\mathcal{C}^j, |H^j_{i,\max}|)$

    $\mathcal{H}^{j+1}_i = \texttt{GenerateAdaptedMeshes}(\mathcal{H}^j_i, \mathcal{M}^j_i)$

  End for

End for

Figure 2.9: The global space-time mesh adaptation fixed-point algorithm. Quantities in red are associated with one solution sample, quantities in green to one adaptation sub-interval and those in blue to one fixed point loop. Two loops are split using distributed parallelization and inside the loop which cannot be parallelized, procedures in bold font are multi-threaded (shared-memory parallelization).

- The solution computation and the interpolations are performed in parallel via multi-threading using shared memory between cores,
- A distributed parallelization strategy has been retained for the computation of the samples Hessians and of multi-scale metrics computations as well as for meshes generation. Notably, if n_{adap} meshes must be generated and if $n_{proc} \leq n_{adap}$ processors are available, the work distribution is done in such a way that a processor is never inactive. As soon as one of them has achieved its task, it is entrusted with a new workload. This strategy requires that all the maximal Hessians $\{|H^j_{i,max}|\}_{i\in[\![1,n_{adap}]\!]}$ and all the continuous meshes $\{\mathcal{M}^j_i\}_{i\in[\![1,n_{adap}]\!]}$

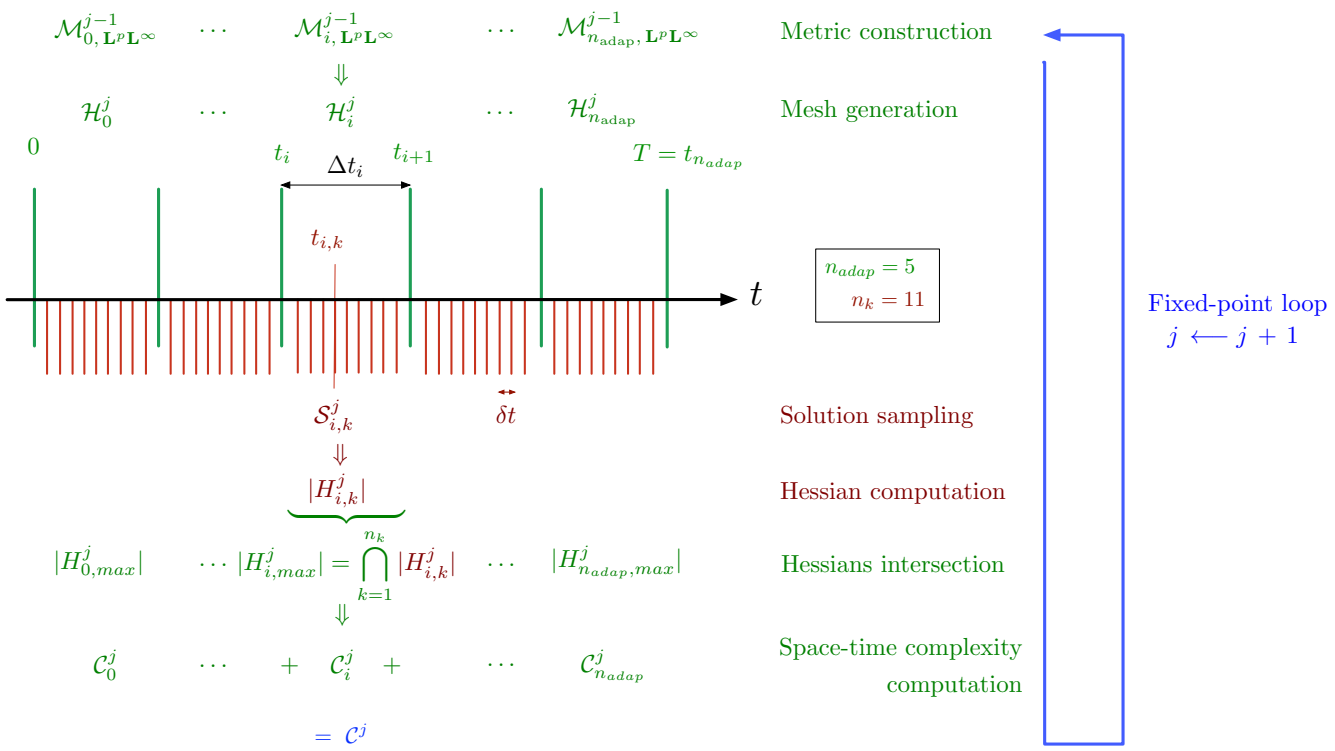


Figure 2.10: New global fixed-point algorithm: computation of the total space-time complexity $\mathcal{C}^j$ at each loop and global normalization. Greens stages can be performed in parallel.

are stored simultaneously.

In the next section, the practical use of the optimal space-time Metric (2.11) inside this new global algorithm is discussed, and several difficulties are also highlighted.

2.3.3 Practical use of the optimal space-time metric inside the fixed-point algorithm.

In Section 2.2.3, the temporal dimension has been treated exactly like spatial ones, except for the constraint imposed on the temporal eigen-direction. Of course, in practice, time is scarcely treated as a mere additional dimension. Indeed, as explained in Section 2.1.3, most Euler solvers use the method of lines to handle time dependency in order to avoid the generation of four dimensional meshes. The consequence is that the time discretization, *i.e.* time steps, are imposed by the solver and cannot be prescribed externally by a "time" metric.
Therefore, additional treatments must be performed on space-time Metric (2.10) to get a metric only prescribing spatial sizes but which controls at best the global space-time interpolation error.

2.3.3.1 Strategy without temporal Hessians

In a first time, we have voluntarily ignored the u_{tt} in Expression (2.10) of $\mathcal{M}^{st}$ to focus on the evaluation of the space-time complexity and the set up of the new global fixed-point strategy.

As this strategy does not control the whole space-time error but only the instantaneous spatial error, it will be referred to as "partially space-time" unsteady mesh adaptation.

The following metric is used to adapt the mesh between t_i and t_{i+1}:

$$\mathcal{M}^{part.\,st}_{\mathbf{L}^p\mathbf{L}^\infty,i}(\mathbf{x}) = D^{part,st}_{\mathbf{L}^p\mathbf{L}^\infty} \left[\det |H^s_{i,max}(\mathbf{x})|\right]^{-\frac{1}{2p+n}} |H^s_{i,max}(\mathbf{x})|\,,$$

$$\text{with } H^s_{i,max}(\mathbf{x}) \;=\; \bigcap_{k=1}^{n_k} H^s_{i,k} \;=\; \bigcap_{k=1}^{n_k} H^s_u(\mathbf{x},\, t_{i,k})\,.$$

The non-dimensioned complexity associated with continuous space-time mesh $\mathbf{M}^{part.\,st}_{\mathbf{L}^p\mathbf{L}^\infty}$ then reads:

$$\mathcal{C}(\mathbf{M}^{part.\,st}_{\mathbf{L}^p\mathbf{L}^\infty}) \;=\; \sum_{i=1}^{n_{\text{adap}}} \int_{t_i}^{t_{i+1}} \int_\Omega \sqrt{\det \mathcal{M}^{part.\,st}_{\mathbf{L}^p\mathbf{L}^\infty,i}(\mathbf{x})}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t \;=\; \sum_{i=1}^{n_{\text{adap}}} \Delta t_i \int_\Omega \prod_{l=1}^{n} h^{-1}_{l,i,max}(\mathbf{x})\,\mathrm{d}\mathbf{x}\,,$$

and the global normalization is given by:

$$D^{part,st}_{\mathbf{L}^p\mathbf{L}^\infty} = N^{\frac{2}{n}} \left(\sum_{i=1}^{n_{\text{adap}}} \Delta t_i \int_\Omega \left[\det |H^s_{i,max}(\mathbf{x})|\right]^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x} \right)^{-\frac{2}{n}}.$$

Actually, this strategy considers that there is no time error during the simulation, and that consequently, the space-time mesh size in the time direction can be constant and arbitrarily large. In other words, there is no mesh adaptation in the temporal direction. The unsteadiness is taken into account only through the integration of the spatial error in the normalization constant $D^{part,st}_{\mathbf{L}^p\mathbf{L}^\infty}$.
This strategy is the one used for all the three-dimensional examples presented in Section 2.4.

2.3.3.2 Strategies with temporal Hessians

A time-advancing space-time metric. Our first idea was to transform $\mathcal{M}^{st}$ into a "time advancing" metric $\overline{\mathcal{M}^{st}}$. Indeed, once Problem (2.8-2.9) has been solved, each vertex has been attributed an optimal size both in space and time. If we were able to generate and to handle four-dimensional space-time meshes, this metric could be used directly under this form and, according to Section 2.2.3, the resulting space-time mesh would be such that each vertex has a local CFL number equal to 1.
However, only time-advancing schemes can be considered so far and local time stepping procedures cannot be considered for unsteady simulations. Therefore, we attempted to perform some additional *a posteriori* treatment on space-time optimal metric $\mathcal{M}^{st}_{\mathbf{L^p}}$ so that each vertex gets the same temporal size prescription.

A first idea is to take:

$$\tau(t) = \tau_{min}(t) = \min_{\mathbf{x}\in\Omega} \tau\,(\mathbf{x}, t)\;,$$

without modifying spatial sizes. However, this *a posteriori* rough truncation does not at all ensure that the resulting mesh is optimal. For instance, the prescription of a big temporal size

at one vertex may be responsible for the prescription of small associated spatial sizes, due to the local normalization term. It is possible that if we had known in advance that the final temporal size of the considered vertex would finally be reduced to τ_{min}, a greater spatial size could have been prescribed while still enforcing the complexity constraint.

One way to remedy this problem is to perform an iterative truncation/correction procedure just like in the case of a locally degenerated Hessian for steady adaptation, see Section 1.4.4.
In the algorithm described in Section 2.1.4.4, the adapted mesh for sub-interval $[t_i,\, t_{i+1}]$ is built from the intersection of the n_k spatial Hessians. This ensures that the worst, *i.e.* the smallest size is prescribed at each vertex, which guarantees the control of the spatial error for the whole sub-interval. In the case of fully space-time mesh adaptation, the prescribed temporal size must be the same for all the vertices as global time-stepping is mandatory. Therefore, for each sample k, all temporal sizes must be truncated to the minimal temporal size:

$$\tau_{i,k} = \min_{\mathbf{x}\in\Omega} \tau_{i,k}(\mathbf{x})\,.$$

The temporal Hessian of the solution u_{tt} is computed for each sample inside the fluid solver as described in Appendix A.1. Thus, n_k temporal sizes $(\tau_{i,k})_{k\in[\![1,n_k]\!]}$ are obtained.

However, as the spatial mesh must remain the same during the whole sub-interval $[t_i,\, t_{i+1}]$, a single time step value must be used in the truncation algorithm to correct spatial sizes. Indeed, if different time steps are used depending on t, n_k different corrected spatial meshes will be generated: this is in contradiction with the main objective of the fixed-point algorithm which is to limit the number of re-meshings. So, we must find a way to get only one time step τ_i from the $(\tau_{i,k})_{k\in[\![1,n_k]\!]}$. Due to the local normalization term, the biggest the prescribed temporal size, the smallest the associated spatial sizes. Therefore, the most constraining temporal size, *i.e.* the biggest one, is chosen among the n_k temporal size associated with the n_k samples. The truncated time step for $[t_i,\, t_{i+1}]$ therefore reads:

$$\tau_i = \max_{k\in[\![1,n_k]\!]} \min_{\mathbf{x}\in\Omega} \tau_{i,k}(\mathbf{x})\,.$$

In terms of metric temporal eigenvalue, this writes $m_{t,i} \geq (m_t)_{i,thres.},\ i \in [\![1,\, n_{adap}]\!]$, with:

$$\begin{aligned}(m_t)_{i,thres.} &= (\tau_i)^{-2} = \min_{k\in[\![1,n_k]\!]} \max_{\mathbf{x}\in\Omega} (m_t)_{i,k}(\mathbf{x},\, t) \\ &= D_{\mathbf{L^pL^\infty}} \min_{k\in[\![1,n_k]\!]} \max_{\mathbf{x}\in\Omega} \left(\left[\prod_{l=1}^{n} \gamma_{l,i,max}(\mathbf{x}) \right] (u_{tt})_{i,k}(\mathbf{x}) \right)^{-\frac{1}{2p+n+1}} (u_{tt})_{i,k}(\mathbf{x})\,,\end{aligned} \tag{2.22}$$

and $(\,\gamma_{l,i,max}\,)_{l\in[\![1,n]\!]}$ are the eigenvalues of $|H_{i,max}|$.

The algorithm to compute the time-advancing space-time metric $\overline{\mathcal{M}}_{i,\,\mathbf{L^pL^\infty}}$ is based on the one described in Section 1.4.4 for the truncation of steady multi-scale metrics. However, the additional time dimension must now be taken into account and is *always* truncated to τ_i.
The main difference as compared to the steady truncation algorithm is that the metric threshold for the temporal size depends on the current value of the space-time Hessian, which changes at

each non-linear loop iteration. Consequently, this truncation threshold must be updated at each new loop of the truncation procedure. The algorithm is therefore highly non-linear and the convergence generally slower than in the steady case.

At the moment, we have encountered several difficulties with this algorithm. The main problem comes from the computation of temporal threshold $m_{t,i,thres.}$. Indeed, the max operator in Expression (2.22) makes the value of $m_{t,i,thres.}$ very dependent of the computation of temporal Hessians u_{tt}. If the numerical approximation fails only at one vertex and generates a very high value, it will spoil the whole spatial mesh.
In the steady case, this problem does not occur as the spatial Hessian computed at one vertex impacts the mesh generation only in the neighborhood of this vertex. Moreover, the anomalies of the numerical spatial Hessian are attenuated thanks to the metric gradation process [Alauzet 2010a]. On the contrary, in the case of the space-time metric, a single anomaly on u_{tt} can have a negative influence on the whole mesh, notably if the maximal value of u_{tt} on Ω is overestimated.
Figure 2.11 illustrate the numerical behavior of $\max_{\mathbf{x}\in\Omega} |u_{tt}(\mathbf{x}, t)|$ on a two-dimensional adaptive city blast simulation.
These curves show that numerically, the time evolution of $\max_{\mathbf{x}\in\Omega} u_{tt}(\mathbf{x}, t)$ exhibits strong oscillations which prevent the use of such numerical quantity as a global truncation parameter for the spatial metric.

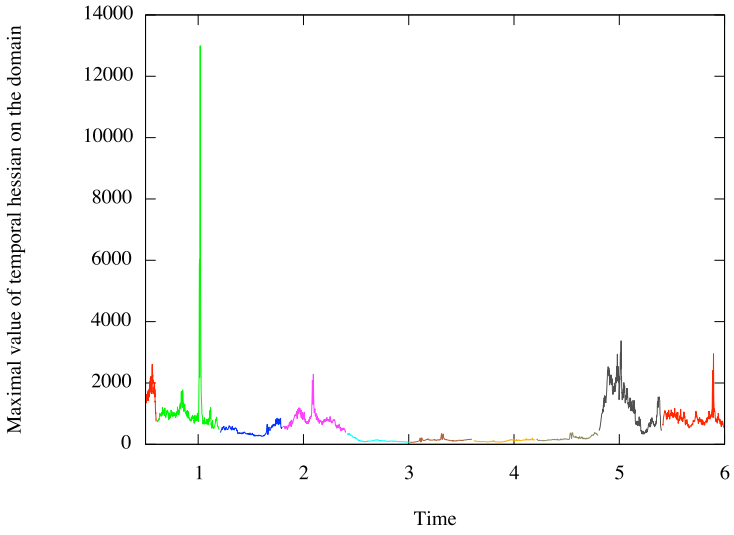


Figure 2.11: Time evolution of $\max_{\mathbf{x}\in\Omega} u_{tt}(\mathbf{x}, t)$, the maximal value of u_{tt} on Ω.

A fully space-time strategy. For this new strategy, the influence of temporal Hessian on the optimal continuous space-time mesh is taken into account. To this aim, the spatial part $\mathcal{M}^s$ of the optimal space-time metric $\mathcal{M}^{st}$ is first rewritten under Form (2.17). This is primordial to guarantee that the maximal value of the spatial error is controlled on the considered adaptation sub-interval, see Section 2.3.1.

We thus look for H^* such that:

$$\mathcal{M}^s \;=\; D_{\mathbf{L}^p}\left[\,|u_{tt}|\,\det|H^s_u|\,\right]^{-\frac{1}{2p+n+1}}|H^s_u| \;=\; D_{\mathbf{L}^p}\,\det|H^*|^{-\frac{1}{2p+n}}\,|H^*|\,.$$

The following $|H^*|$ fits:

$$H^* \;=\; \left[|u_{tt}|^{-(2p+n)}\det|H^s_u|\right]^{\frac{1}{2p\,(2p+n+1)}}\,|H^s_u|\,.$$

Indeed, with this expression of H^*, we get:

$$\begin{aligned}
& D_{\mathbf{L}^p}\,\det|H^*|^{-\frac{1}{2p+n}}\,|H^*| \\
= \;& D_{\mathbf{L}^p}\left[\det\det\left[|u_{tt}|^{-(2p+n)}\det|H^s_u|\right]^{\frac{1}{2p\,(2p+n+1)}}|H^s_u|\right]^{-\frac{1}{2p+n}}\left[|u_{tt}|^{-(2p+n)}\det|H^s_u|\right]^{\frac{1}{2p\,(2p+n+1)}}|H^s_u| \\
= \;& D_{\mathbf{L}^p}\left[|u_{tt}|^{-\frac{n(2p+n)}{2p\,(2p+n+1)}}\det|H^s_u|^{\frac{n}{2p\,(2p+n+1)}}\,\det|H^s_u|\right]^{-\frac{1}{2p+n}}\left[|u_{tt}|^{-\frac{2p+n}{2p\,(2p+n+1)}}\det|H^s_u|^{\frac{1}{2p\,(2p+n+1)}}\right]|H^s_u| \\
= \;& D_{\mathbf{L}^p}\left[\,|u_{tt}|\,\right]^{\frac{n}{2p\,(2p+n+1)}-\frac{2p+n}{2p\,(2p+n+1)}}\left[\det|H^s_u|\,\right]^{-\frac{n}{2p\,(2p+n+1)(2p+n)}+\frac{1}{2p\,(2p+n+1)}-\frac{1}{2p+n}}|H^s_u| \\
= \;& D_{\mathbf{L}^p}\left[\,|u_{tt}|\,\right]^{-\frac{2p}{2p\,(2p+n+1)}}\left[\det|H^s_u|\,\right]^{\frac{-n+2p+n-2p(2p+n+1)}{2p\,(2p+n+1)(2p+n)}}|H^s_u| \\
= \;& D_{\mathbf{L}^p}\left[\,|u_{tt}|\,\det|H^s_u|\,\right]^{-\frac{1}{2p+n+1}}|H^s_u| \;=\; \mathcal{M}^s\,.
\end{aligned}$$

Then, the mesh is adapted for adaptation sub-interval $[t_i,\,t_{i+1}]$ using:

$$\begin{aligned}
\mathcal{M}^{full.\,st}_{\mathbf{L^pL}^\infty,i}(\mathbf{x},\,t) \;&=\; D_{\mathbf{L^pL}^\infty}\left(\,\det|H^*_{max,i}(\mathbf{x},\,t)|\,\right)^{-\frac{1}{2p+n}}|H^*_{max,i}(\mathbf{x},\,t)| \\
\text{with}\quad H^*_{max,i} \;&=\; \bigcap_{k=1}^{n_k} H^*_{i,k} \;=\; \bigcap_{k=1}^{n_k} H^*(\mathbf{x},\,t_{i,k})\,,
\end{aligned} \tag{2.23}$$

and the normalization constant is computed as:

$$D_{\mathbf{L^pL}^\infty} \;=\; (N^{st})^{\frac{2}{n+1}}\left[\sum_{i=1}^{n_k}\int_{t_i}^{t_{i+1}}\int_{\Omega}\left(|u_{tt}|\,\det|H^s_{max,i}|\right)^{\frac{p}{2p+n+1}}\mathrm{d}\mathbf{x}\,\mathrm{d}t\right]^{-\frac{2}{n+1}}.$$

Note that this quantity is not equal to:

$$D_{\mathbf{L^pL}^\infty} \;\neq\; (N^{st})^{\frac{2}{n+1}}\left[\sum_{i=1}^{n_k}\int_{t_i}^{t_{i+1}}\int_{\Omega}\left(\det|H^*_{max,i}|\right)^{\frac{p}{2p+n}}\left(\tau^{opt}\right)^{-1}\mathrm{d}\mathbf{x}\,\mathrm{d}t\right]^{-\frac{2}{n+1}},$$

except if mesh adaptation is performed at each solver time step, *i.e.* no Hessian intersection is done.

In the following, we demonstrate that this metric enables to control the global space-time error in $\mathbf{L}^p(\Omega \times [0,\, T])$ norm. Two important assumptions are made:

- (H_1): The local temporal interpolation error e^t is proportional to the local spatial one under CFL condition. This has been proved for the case of the linear advection equation in Remark 8,

- (H_2): The adaptation is performed at each solver time step, which enables to get rid of the intersection operator.

According to Hypothesis (H_1), if a CFL type condition is enforced, the following inequality holds:

$$e^t(\mathbf{x},\, t) = C\, e^s(\mathbf{x},\, t)\,.$$

Therefore, the global temporal error in $\mathbf{L}^p$ norm writes:

$$E^t_{\mathbf{L}^p(\Omega\times[0,\,T])} = \left(\int_0^T \int_\Omega \left[e^t(\mathbf{x},\, t)\right]^p \,\mathrm{d}\mathbf{x}\,\mathrm{d}t \right)^{\frac{1}{p}} = C\, E^s_{\mathbf{L}^p(\Omega\times[0,\,T])}\,.$$

So, if $E^s_{\mathbf{L}^p(\Omega\times[0,\,T])}$ is controlled, the whole global space-time error is controlled in $\mathbf{L}^p(\Omega \times [0,\, T])$ norm.

Now, let us assume Metric (2.23) is used to adapt the mesh. According to Hypothesis (H_2), the adaptation is performed at each solver time-advancing step and intersected Hessian $H^*_{max,i}$ reduces to H^*_i. The prescribed spatial sizes are then locally given by:

$$h_i \;=\; D_{\mathbf{L^p}}^{-\frac{1}{2}} \left[|u_{tt}|\, \det|H^s_u|\right]^{\frac{1}{2(2p+n+1)}} \gamma_i^{-\frac{1}{2}}\,. \tag{2.24}$$

Substituting h_i by Expression (2.24) in the local spatial error, we get:

$$\begin{aligned} e^s(\mathbf{x},\, t) \;&=\; \sum_{i=1}^n h_i^2(\mathbf{x},\, t)\gamma_i(\mathbf{x},\, t) \;=\; n\, D_{\mathbf{L^p}}^{-1} \left(|u_{tt}|\, \det|H^s_u|\right)^{\frac{1}{2p+n+1}} \\ &=\; n\, \left(N^{st}\right)^{-\frac{2}{n+1}} \left(\int_0^T \int_\Omega \left(|u_{tt}|\, \det|H^s_u|\right)^{\frac{p}{2p+n+1}} \mathrm{d}\mathbf{x}\,\mathrm{d}t \right)^{\frac{2}{n+1}} \left(|u_{tt}|\, \det|H^s_u|\right)^{\frac{1}{2p+n+1}}. \end{aligned}$$

The global space-time spatial error in $\mathbf{L}^p(\Omega \times [0,\, T])$ norm for this continuous space-time mesh

then writes:

$$
\begin{aligned}
E^s_{\mathbf{L}^p(\Omega\times[0,T])} &= \left(\int_0^T\int_\Omega [e^s(\mathbf{x},\,t)]^p\;\mathrm{d}\mathbf{x}\,\mathrm{d}t\right)^{\frac{1}{p}} \\
&= n\left(N^{st}\right)^{-\frac{2}{n+1}}\left(\int_0^T\int_\Omega \left(|u_{tt}|\,\det|H^s_u|\right)^{\frac{p}{2p+n+1}}\mathrm{d}\mathbf{x}\,\mathrm{d}t\right)^{\frac{2}{n+1}}\left(\int_0^T\int_\Omega \left(|u_{tt}|\,\det|H^s_u|\right)^{\frac{p}{2p+n+1}}\mathrm{d}\mathbf{x}\,\mathrm{d}\right. \\
&= n\left(N^{st}\right)^{-\frac{2}{n+1}}\left(\int_0^T\int_\Omega \left(|u_{tt}|\,\det|H^s_u|\right)^{\frac{p}{2p+n+1}}\mathrm{d}\mathbf{x}\,\mathrm{d}t\right)^{\frac{2p+n+1}{p(n+1)}}.
\end{aligned}
$$

Thanks to Minkowski inequality, the following result holds:

$$
E^{st}_{\mathbf{L}^p(\Omega\times[0,T])} = ||e^s+e^t||_{\mathbf{L^p}(\Omega\times[0,T])} \le ||e^s||_{\mathbf{L^p}(\Omega\times[0,T])} + ||e^t||_{\mathbf{L^p}(\Omega\times[0,T])} = E^s_{\mathbf{L^p}} + E^t_{\mathbf{L^p}}\,,
$$

and finally, we obtain:

$$
E^{st}_{\mathbf{L}^p(\Omega\times[0,T])} \;\le\; C\,\frac{n+1}{n}\,E^s_{\mathbf{L^p}(\Omega\times[0,T])} \;=\; (n+1)\,\left(N^{st}\right)^{-\frac{2}{n+1}}\left(\int_0^T\int_\Omega \left(|u_{tt}|\det|H^s_u|\right)^{\frac{p}{2p+n+1}}\mathrm{d}\mathbf{x}\,\mathrm{d}t\right)^{\frac{2p+n+1}{p(n+1)}}.
$$

This shows that under Hypothesis (H_1) and (H_2), the global space-time error in $\mathbf{L}^p(\Omega\times[0,\,T])$ norm is asymptotically converged at order 2. Indeed, the space-time mesh is of dimension 4, so the order is the real α such that:

$$
E^{st}_{\mathbf{L}^p(\Omega\times[0,T])} \;\le\; C\,\left(N^{st}\right)^{-\frac{\alpha}{n+1}}.
$$

In practice, the mesh is not adapted at each iteration. However, due to the properties of the intersection operator detailed in Section 2.3.1, metric $\mathcal{M}^{full.st}_{\mathbf{L}^p\mathbf{L}^\infty,i}$ leads to a space-time mesh for sub-interval $[t_i,\,t_{i+1}]$ which is sub-optimal, but which still guarantees the control of the global space-time interpolation error at order 2.

2.4 Numerical illustrations and comparison with the former algorithm

In all the examples, the flow is modeled by the compressible Euler equations. The Euler system is solved by means of a Finite Volume technique on unstructured tetrahedral meshes. The proposed scheme is vertex-centered and uses a particular edge-based formulation with upwind elements, see Section 5.2. Appropriate β-schemes are adopted for the variable extrapolation which gives us a low diffusion second-order space-accurate scheme in the non-linear case. The MUSCL type method is combined with a generalization of the Superbee limiter with three entries to guarantee

the TVD property of the scheme. An explicit time stepping algorithm is used by means of a 4-stages, 3-order strong-stability-preserving (SSP) Runge-Kutta scheme which allows us to use a CFL coefficient up to 2.

All simulations have been run on a 8-processors 64-bits *MacPro* ® with an *IntelCore2*® chipset with a clock speed of 2.8 GHz with 32 Gb of RAM.

Quantifying mesh anisotropy. In three dimensions, mesh anisotropy can be quantified by two different indicators: the anisotropic ratios and the anisotropic quotients. Deriving these quantities for an element relies on the fact that there always exists a unique metric tensor for which this element is unit. If $\mathcal{M}_K$ denotes the metric tensor associated with element K, solving the following linear system provides $\mathcal{M}_K$:

$$(S)\ \begin{cases} \ell^2_{\mathcal{M}_K}(\mathbf{e}_1) = 1 \\ \vdots \\ \ell^2_{\mathcal{M}_K}(\mathbf{e}_6) = 1\,, \end{cases}$$

where $(\mathbf{e}_i)_{i=1,6}$ is the list of edges of element K and $\ell^2_{\mathcal{M}_K}(\mathbf{e}_i) = \mathbf{e}_i^T\,\mathcal{M}_K\,\mathbf{e}_i$. (S) admits a unique solution as soon as the volume of K is not null. Once $\mathcal{M}_K$ is computed, the anisotropic ratios and the quotients associated with element K are simply given by

$$r = \sqrt{\frac{\min\limits_{l\in[\![1,3]\!]}\lambda_l}{\max\limits_{l\in[\![1,3]\!]}\lambda_l}} = \frac{\max\limits_{l\in[\![1,3]\!]} h_l}{\min\limits_{l\in[\![1,3]\!]} h_l}, \quad \text{and} \quad \kappa = \frac{\max\limits_{l\in[\![1,3]\!]} h_l^3}{h_1h_2h_3},$$

where $(\lambda_l)_{l\in[\![1,3]\!]}$ are the eigenvalues of $\mathcal{M}_K$ and $(h_l)_{l\in[\![1,3]\!]}$ are the corresponding sizes. The anisotropic ratio stands for the maximum elongation of a tetrahedron by comparing two main directions. The anisotropic quotient represents the overall anisotropic ratio of a tetrahedron taking into account all the possible directions. This quotient can be considered as a measure of the overall gain in three dimensions of *an anisotropic adapted mesh* as compared to *an isotropic one*. This gain is of course even greater when compared to a uniform mesh.

2.4.1 Three-dimensional Double-Mach reflection simulation

The first 3D simulation to illustrate the efficiency of the proposed approach is the double Mach reflection test case proposed in [Woodward 1984]. In this problem, a planar shock wave collides with a straight compressive wedge and reflects over it as a Mach reflection. When the shock hits the sloping wall, a complicated shock reflection occurs. A triple point at which four discontinuities meet typifies the Mach reflection. The four discontinuities are the incident, the reflected and the Mach stem shock waves and the slipstream. Experiments indicate that the Mach stem (the front shock wave) appears to be straight or to develop either a concave or convex shape. A curved Mach stem occurs when the front of the curled slipstream catches up with the Mach stem, which, as a consequence, is pushed forward and exhibits a convex shape.

Numerically, the critical component of the result which differentiates between methods is the appearance of the dense jet along the wall, which is sensitive to numerical diffusion, and the prediction of a straight or curved Mach stem.

Iteration	nv	nt	nf	h_{min}	ratio	quotient
Initial Unif.	8 346	35 978	9 012	$5.e^{-2}$	1.8 (4)	2.6 (10)
10	256 351	1 448 810	49 554	$7.5e^{-4}$	11 (78)	66 (2272)
20	248 687	1 398 138	53 228	$7.5e^{-4}$	11 (82)	67 (3095)
30	235 095	1 310 082	57 864	$7.5e^{-4}$	11 (78)	68 (4943)

Table 2.2: Mesh statistics for the 3D double mach reflection problem.

Here, the problem is initialized by sending a Mach 10 shock wave into a reflecting wall that is inclined by an angle of 30°. In the region behind the shock wave the following initial conditions are considered:

$$\rho = 8, \quad \mathbf{v} = (8.25, 0, 0), \quad p = 116.5\,,$$

whereas the ambient air conditions are:

$$\rho = 1.4, \quad \mathbf{v} = (0, 0, 0), \quad p = 1\,.$$

Following the study of [Li 1999], these initial conditions lead to a convex Mach stem.

As regards mesh adaptation for this example, we split the simulation time frame into 30 adaptation sub-intervals. For each sub-interval, we consider 21 samples of the solution to build the metric. Five fixed-point iterations have been done to converge the non-linear problem of mesh adaptation. The space-time interpolation error on the local sound speed variable is controlled in $\mathbf{L}^\infty - \mathbf{L}^2$ norm. We start from an initial uniform very coarse mesh containing 8 346 vertices and 35 978 tetrahedra. At the end of the simulation, the final adapted mesh for the last sub-interval contains 235 095 vertices and 1 310 082 tetrahedra, see Figure 2.12. The final solution in Figure 2.12 shows that the dense jet along the wall has been accurately captured and, as expected, a Mach stem convex shape is obtained with only 235 095 vertices in three dimensions.

The mesh adaptation for the whole sub-interval is clearly illustrated. Indeed, the mesh refinement along band-shaped regions, which is typical of the fixed-point algorithm, is clearly visible. These band-shaped areas correspond to the evolution zone of the physical phenomena during an adaptation sub-interval and account for the control of the space-time interpolation error during the whole simulation. In Table 2.2, statistics of three meshes, corresponding to adaptation sub-intervals 10, 20 and 30, are given. We notice that the mesh accuracy is almost 100 times better than with the initial mesh allowing us to accurately capture the shocks and the slipstream. As regards the amount of anisotropy for this simulation, an average anisotropic ratio of 11 and a mean anisotropic quotient of almost 70 are obtained. This quotient measures the overall gain as compared to an isotropic mesh adaptation.

The simulation total CPU time is 8h55m. It took 1min30s to compute the first solution on the initial mesh and 1h56min to compute the last solution (at fixed point iteration 5) on the last series of adapted meshes. For the whole simulation, 7h of the CPU time has been spent in the triple - Solver / Metric Computation / Solution Interpolation - and 1h55min for the couple - Metric Gradation / Mesh Adaptation.

2.4.2 Three-dimensional blast in a city

In the second example, we consider a purely three-dimensional blast problem in a complex geometry representing a city. The city size is $85\,m \times 70\,m \times 70\,m$. In this simulation, shock waves interact with each other and are reflected by the buildings. The city geometry is the same as in [Alauzet 2007]. Initially, the ambient air is at rest:

$$\rho = 1, \quad \mathbf{v} = (0,0,0), \quad e = 2.5\,.$$

A "blast-like" initialization is considered inside a half-sphere of radius $r = 2.5\,m$ around $x = (42,53,0)$:

$$\rho = 10, \quad \mathbf{v} = (0,0,0), \quad e = 25\,,$$

The density of the flow is chosen as sensor variable for our mesh adaptation process. The space-time interpolation error on the sensor is controlled in $\mathbf{L}^\infty - \mathbf{L}^2$ norm. The time frame was split into 40 adaptation sub-intervals and 5 fixed-point iterations were used to converge the non-linear mesh adaptation problem. For each sub-interval, we consider 21 samples of the solution to build the metric. The desired accuracy was set to reach a space-time complexity equal to 4 millions. We start from an initial uniform mesh containing 99 255 vertices and 549 128 tetrahedra with an accuracy of $35\,cm$.

Figure 2.13 shows the iso-surfaces and the iso-values of the final density at sub-interval 20, 30 and 40. It points out the complexity and the unpredictable behavior of the physical phenomena with a large number of shock waves interacting with the geometry. Thanks to multi-scale mesh adaptation, all shock waves are automatically captured by the adaptation process and properly refined. Again, the mesh adaptation for the whole sub-interval is clearly illustrated. Indeed, the mesh refinement along band-shaped regions, which corresponds to the zone in which physical phenomena evolves during an adaptation sub-interval, are visible. At the end of the simulation, the final adapted mesh for the last sub-intervals contains 185 148 vertices and 1 027 537 tetrahedra, see Figure 2.14. Its accuracy is about $10\,cm$ whereas the accuracy required at the beginning of the simulation, when the energy of the blast is maximal, is close to $1\,cm$. In Table 2.3, statistics of four meshes, at sub-intervals 10, 20, 30 and 40, are given. We notice that the mesh accuracy is almost 10 to 30 times better than with the initial mesh allowing us, to precisely capture the solution. As regards the amount of anisotropy for this simulation, an average anisotropic ratio between 8 and 15 and a mean anisotropic quotient between 50 and 160 are obtained. The anisotropic quotient measures the overall gain as compared an isotropic mesh adaptation, here almost 100. The gain is of course even greater when compared to a uniform mesh.

The simulation total CPU time is 4h32min. It took 9min to compute the first solution on the initial mesh and 40min to compute the last solution (at fixed point iteration 5) on the last series of adapted meshes. For the whole simulation, 2h52min of the total CPU time has been devoted to the triple - Solver / Metric Computation / Solution Interpolation - and 1h40min to the couple - Metric Gradation / Mesh Adaptation.

Iteration	nv	nt	nf	h_{min}	ratio	quotient
Initial Unif.	99 255	549 128	35 420	35cm	2 (32)	4 (540)
10	305 027	1 746 040	42 486	4cm	15 (105)	164 (6804)
20	225 829	1 275 931	45 000	6cm	12 (72)	122 (3380)
30	189 858	1 057 022	48 594	9cm	9 (76)	77 (2890)
40	185 148	1 027 537	50 250	11cm	8 (71)	56 (2813)

Table 2.3: Mesh statistics for the 3D city blast problem.

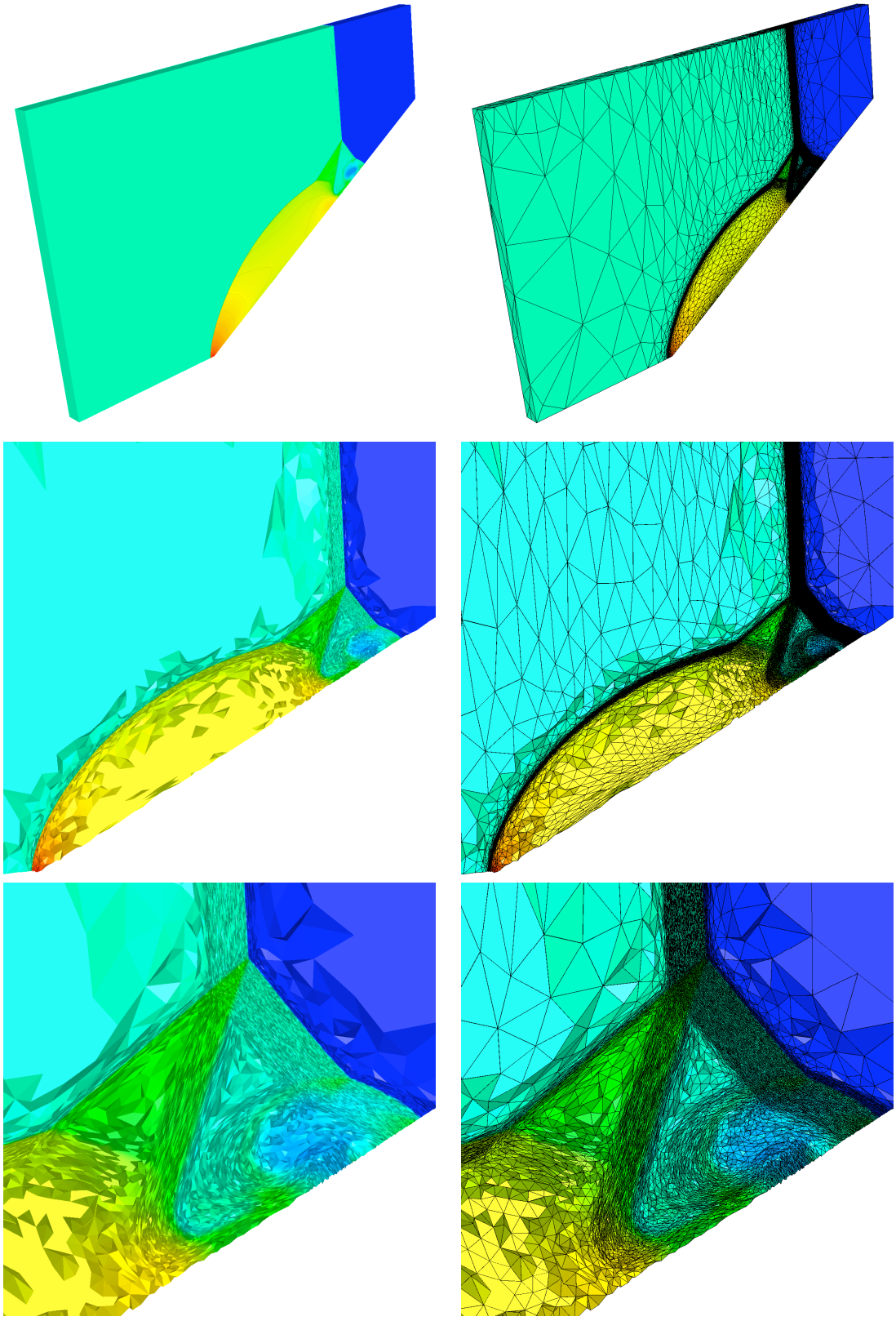


Figure 2.12: Three dimensional double Mach reflection simulation. Left, final solution isovalues and, right, last adapted mesh. From top to bottom, on the surface, in the volume and a close up view when the front of the curled slipstream catches up with the Mach stem.

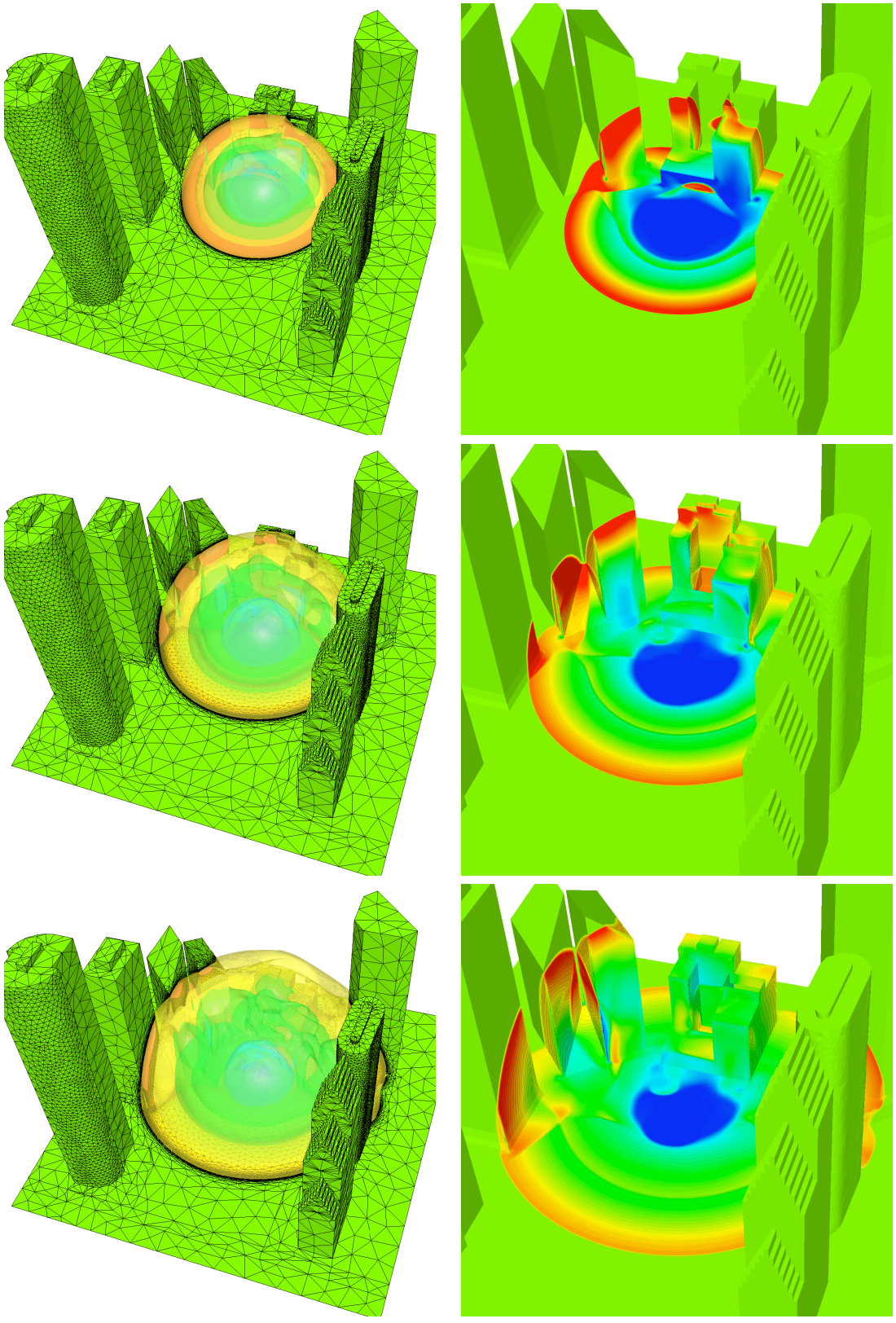


Figure 2.13: Three dimensional blast in a city simulation. Left, density iso-surfaces and, right, density iso-values on the surface. From top to bottom, solutions at sub-intervals 20, 30 and 40, respectively.

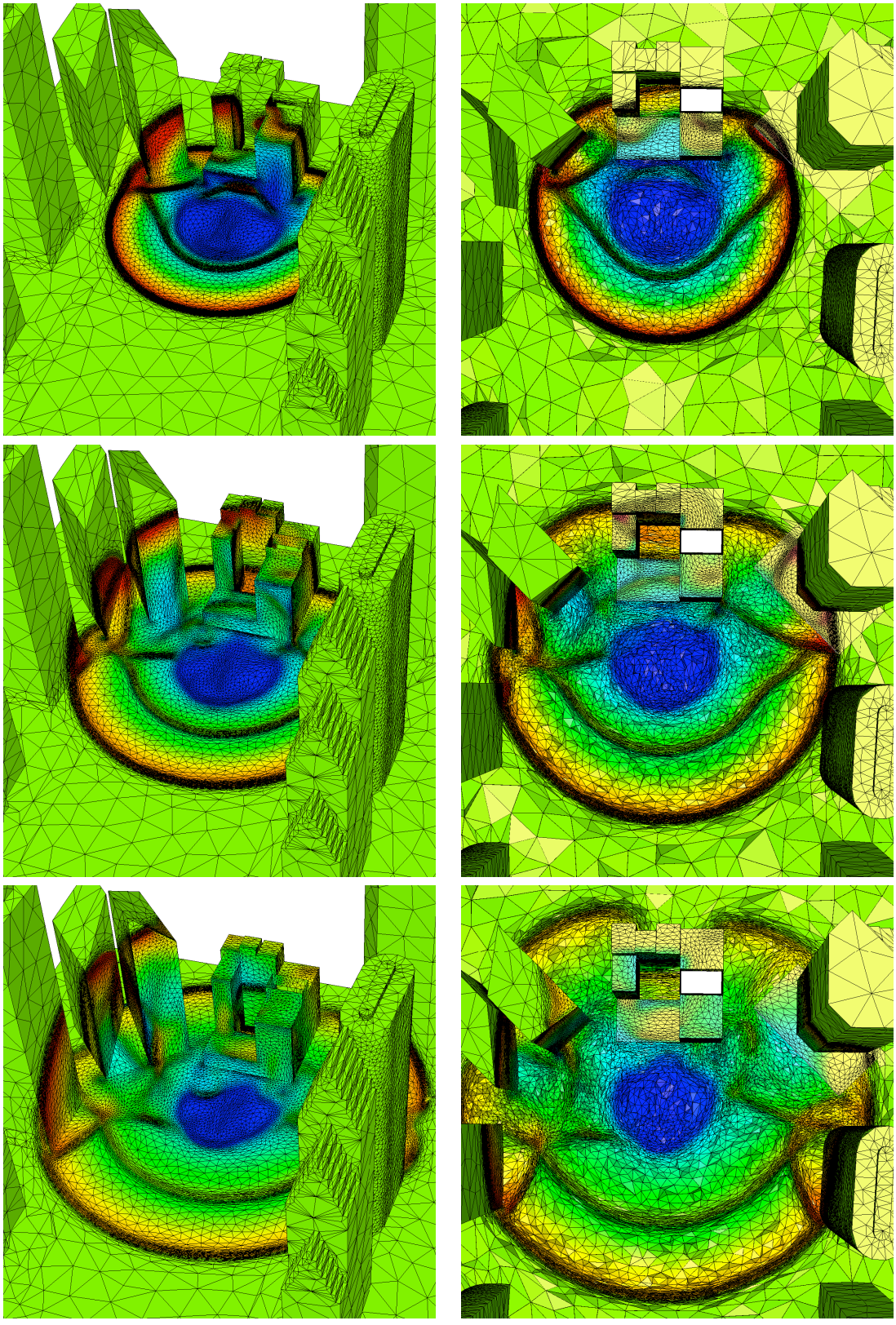


Figure 2.14: Three dimensional blast in a city simulation. Left, adapted surface meshes and, right, cut in the adapted volume meshes. From top to bottom, adapted meshes corresponding to sub-intervals 20, 30 and 40.

Chapter 3

Unsteady metric-based mesh adaptation for moving mesh simulations

Contents

3.1 The ALE metric 108
3.2 Analytical examples 111
3.2.1 Tests cases description 111
3.2.2 Results analysis. 115
3.3 Extension of the fixed-point algorithm to moving mesh simulations . . 117

This chapter deals with metric-based anisotropic mesh adaptation in the context of simulations involving moving geometries.
The problem is that direct extension of the fixed-point algorithm described in Section 2.3 to moving mesh simulations does not take into account the movement of the mesh. Indeed, the local error estimate from which the optimal metric is deduced does not take into account the local deformation of the mesh which necessarily impacts the local errors.
The final objective of this chapter is to extend the fixed-point adaptation algorithm to moving mesh simulations and to guarantee, just like in the fixed-mesh case, a control of the interpolation error during a whole adaptation sub-interval in a moving mesh context.

Now, let us mathematically describe the framework of this study. Computational domain $\Omega_h = \Omega_h(t)$ is time-dependent and generally, $\Omega_h(t^k) \neq \Omega_h(t^{k+1})$ between two instants t^k and t^{k+1} in $[0, T]$. As we want to adapt the mesh to a solution defined on a deforming domain, the deformation of the mesh must be taken into account into the metric.

To this aim, we first concentrate on a problem involving only two instants t^k and t^{k+1}. The question to be answered is the following:

> How to build a mesh of the domain at t^k which, once deformed from t^k to t^{k+1}, is adapted to the solution u^{k+1} at t^{k+1}?

The resolution of this problem led us to introduce a new optimal Arbitrary-Lagrangian-Eulerian (ALE) metric, which involves the gradient of the mesh transformation between t^k and t^{k+1}. The efficiency of the ALE metric is illustrated on several two and three-dimensional analytical examples. Next, the extension of the fixed-point algorithm to moving mesh simulations using this new metric is explained.

Notations. The following notations will be used in the sequel:

- Ω^k and Ω^{k+1} denote the spatial domain at t^k and t^{k+1}, respectively,
- ∇^k denotes the gradient operator[1] performed on domain Ω^k,
- $H^{k+1}[\cdot]$ denotes the Hessian operator performed on domain Ω^{k+1},
- $\mathcal{M}^{k+1}_{\mathbf{L^p}}[\cdot]$ denotes the $\mathbf{L^p}$ optimal metric operator calculated on Ω^{k+1}.

We also note: $\mathcal{C}\left(\mathcal{M}^{k+1}_{\mathbf{L^p}}\left[u^{k+1}\right]\right) = N^{k+1}$. As operators $\mathcal{M}^{k+1}_{\mathbf{L^p}}[\cdot]$ and $H^{k+1}[\cdot]$ are always applied to solution u^{k+1} at t^{k+1}, the following abusive notations will be used: $\mathcal{M}^{k+1}_{\mathbf{L^p}}$ will hold for $\mathcal{M}^{k+1}_{\mathbf{L^p}}\left[u^{k+1}\right]$ and H^{k+1} for $H^{k+1}\left[u^{k+1}\right]$.

3.1 The ALE metric

Optimal ALE metric. Even if $\Omega^k \neq \Omega^{k+1}$ in general, we assume that these two spatial domains can be mapped one onto the other, which means there exists a mapping $\boldsymbol{\phi}$ such that:

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

and, as $\boldsymbol{\phi}$ is a diffeomorphism, we have, for any infinitesimal vector $\mathrm{d}\mathbf{x}^k \in \Omega^k$:

$$\mathrm{d}\mathbf{x}^{k+1} = \left[\nabla^k \boldsymbol{\phi}(\mathbf{x}^k)\right]^T \mathrm{d}\mathbf{x}^k\,. \tag{3.1}$$

Mapping $\boldsymbol{\phi}$ and mesh displacement field $\mathbf{d}$ are linked by the following relation:

$$\mathbf{x}^{k+1} = \boldsymbol{\phi}(\mathbf{x}^k) = \mathbf{x}^k + \mathbf{d}(\mathbf{x}^k) \quad \Longrightarrow \quad \nabla^k \boldsymbol{\phi}(\mathbf{x}^k) = \mathcal{I}_n + \nabla^k \mathbf{d}(\mathbf{x}^k),\ \forall\, \mathbf{x}^k \in \Omega^k$$

For the purpose of simplicity, sensor function u is assumed to be scalar, the extension to vectorial functions being straightforward.
Finally, we note $\widehat{H^{k+1}}$ the Hessian of u^{k+1} computed on Ω^{k+1} and transported on domain Ω^k. This mathematically writes:

$$\begin{array}{rccl} \widehat{H^{k+1}}: & \Omega^k & \longrightarrow & \mathbb{R} \\ & \mathbf{x}^k & \longmapsto & H^{k+1}\left[u^{k+1}\right]\left(\boldsymbol{\phi}\left(\mathbf{x}^k\right)\right) = \widehat{H^{k+1}}(\mathbf{x}^k)\,. \end{array}$$

Figure 3.1 illustrates the meaning of operator $\widehat{\cdot}$ in one dimension.

[1] Here, the gradient is not the Jacobian, *i.e.* for an arbitrary vector field $\mathbf{f} = (f_1, \ldots, f_n)$, its gradient matrix is $\nabla \mathbf{f} = \left(\frac{\partial f_j}{\partial x_i}\right)_{ij}$

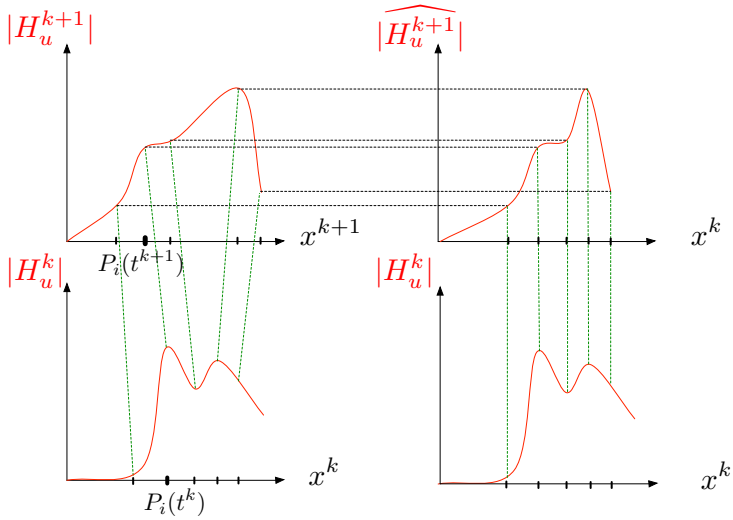


Figure 3.1: Effect of operator $\widehat{\cdot}$ on a one dimensional scalar Hessian function $|H_u|$.

For our moving mesh simulations, topology changes can occur but no vertex addition or suppression is allowed, see Part II. Consequently, the mesh complexity remains constant in time, *i.e.* the complexity of the metric field defined on Ω^k must be the same as the one of the metric field defined on Ω^{k+1}.

In the sequel, the following result is demonstrated:

Theorem 5 (Optimal ALE $\mathbf{L^p}$ metric)**.** *Let metric $\mathcal{M}_{\mathbf{L^p}}^{ALE}$ be defined on Ω^k by:*

$$\begin{aligned}\mathcal{M}_{\mathbf{L^p}}^{ALE}(\mathbf{x}^k) &= D_{\mathbf{L^p}}^{ALE}\left[\det\left(\widehat{H^{k+1}}(\mathbf{x}^k)\right)\right]^{-\frac{1}{2p+n}}\left(\nabla^k\boldsymbol{\phi}(\mathbf{x}^k)\cdot\widehat{H^{k+1}}(\mathbf{x}^k)\cdot\nabla^k\boldsymbol{\phi}^T(\mathbf{x}^k)\right)\\ &= \left(\frac{N^{k+1}}{\displaystyle\int_{\Omega^k}\left[\det H^*\right]^{\frac{p}{2p+n}}\mathrm{d}\mathbf{x}^k}\right)^{\frac{2}{n}}\left\{\det\left(H^*\right)\right\}^{-\frac{1}{2p+n}}H^*\end{aligned} \tag{3.2}$$

with

$$\begin{aligned}H^* &= \left[\det\nabla^k\boldsymbol{\phi}(\mathbf{x}^k)\right]^{\frac{1}{p}}\left(\nabla^k\boldsymbol{\phi}(\mathbf{x}^k)\cdot\widehat{H^{k+1}}(\mathbf{x}^k)\cdot\nabla^k\boldsymbol{\phi}^T(\mathbf{x}^k)\right)\\ D_{\mathbf{L^p}}^{ALE} &= \left(N^{k+1}\right)^{\frac{2}{n}}\left(\int_{\Omega^{k+1}}\left[\det\left(H^{k+1}(\mathbf{x}^{k+1})\right)\right]^{\frac{p}{2p+n}}\mathrm{d}\mathbf{x}^{k+1}\right)^{-\frac{2}{n}}\\ &= \left(N^{k+1}\right)^{\frac{2}{n}}\left(\int_{\Omega^k}\left[\det\left(\widehat{H^{k+1}}(\mathbf{x}^k)\right)\right]^{\frac{p}{2p+n}}|\det\nabla^k\boldsymbol{\phi}|\,\mathrm{d}\mathbf{x}^k\right)^{-\frac{2}{n}}\end{aligned} \tag{3.3}$$

The following properties hold:

i) Let us assume metric $\mathcal{M}_{\mathbf{L^p}}^{k,ALE}$ *is used to generate a unit mesh* $\mathcal{H}^k$ *of* Ω^k *and let us denote by* $\mathcal{H}^{k+1}$ *the mesh of* Ω^{k+1} *which is the image of mesh* $\mathcal{H}^k$ *by mapping* $\boldsymbol{\phi}$*. Then, mesh* $\mathcal{H}^{k+1}$ *is optimal to control the interpolation error in* $\mathbf{L^p}$ *norm of sensor* u^{k+1} *on* Ω^{k+1}*.*

ii) Metric $\mathcal{M}_{\mathbf{L^p}}^{k,ALE}$ *has the same complexity* N^{k+1} *as metric* $\mathcal{M}_{\mathbf{L^p}}^{k+1}$*.*

iii) Except for very specific case $\nabla^k\boldsymbol{\phi} = \mathcal{I}_n$*, the mesh is a priori not adapted to* u^k*.*

Proof of i). According to the metric-based mesh adaptation theory for steady problems, the optimal metric in $\mathbf{L^p}$ norm for u^{k+1} is $\mathcal{M}_{\mathbf{L^p}}^{k+1}$ as defined in (1.22). Thus, an optimal mesh of Ω^{k+1} adapted to u^{k+1} can be built by generating a unit mesh $\mathcal{H}^{k+1}$ with respect to $\mathcal{M}_{\mathbf{L^p}}^{k+1}$:

$$1 = (\mathbf{e}^{k+1})^T\, \mathcal{M}_{\mathbf{L^p}}^{k+1}\, \mathbf{e}^{n+1}, \quad \text{for each } \mathbf{e}^{k+1} \text{ of mesh } \mathcal{H}^{k+1}\,. \tag{3.4}$$

For any arbitrary edge $\mathbf{e}^k$ of $\mathcal{H}^k$ having $\mathbf{e}^{k+1}$ as image by $\boldsymbol{\phi}$ in $\mathcal{H}^{k+1}$, we write:

$$\widehat{\mathbf{e}^{k+1}}(\mathbf{x}^k) = \mathbf{e}^{k+1}\Big(\boldsymbol{\phi}(\mathbf{x}^k)\Big) = \Big[\nabla^k\boldsymbol{\phi}(\mathbf{x}^k)\Big]^T\, \mathbf{e}^k(\mathbf{x}^k)\,. \tag{3.5}$$

As we are only interested in controlling the prevailing term of the interpolation error, we can use the above relation, which is true at first order, in the demonstration.
The idea of this proof is to unravel how Condition (3.4) writes when transposed onto mesh $\mathcal{H}^k$. For any arbitrary edge $\mathbf{e}^k$ of $\mathcal{H}^k$ having $\mathbf{e}^{k+1}$ as image in $\mathcal{H}^{k+1}$, we write, using the definition of operator $\mathcal{M}_{\mathbf{L^p}}^{k+1}$ deduced from Relation (1.22):

$$\begin{aligned}
1 &= \Big[\mathbf{e}^{k+1}\big(\boldsymbol{\phi}(\mathbf{x}^k)\big)\Big]^T \cdot \mathcal{M}_{\mathbf{L^p}}^{k+1}\big(\boldsymbol{\phi}(\mathbf{x}^k)\big) \cdot \mathbf{e}^{k+1}\big(\boldsymbol{\phi}(\mathbf{x}^k)\big) \\
&= \Big[\widehat{\mathbf{e}^{k+1}}(\mathbf{x}^k)\Big]^T \cdot \mathcal{M}_{\mathbf{L^p}}^{k+1}\big(\boldsymbol{\phi}(\mathbf{x}^k)\big) \cdot \widehat{\mathbf{e}^{k+1}}(\mathbf{x}^k) \\
&= \Big(\Big[\nabla^k\boldsymbol{\phi}(\mathbf{x}^k)\Big]^T \cdot \mathbf{e}^k(\mathbf{x}^k)\Big)^T \cdot \mathcal{M}_{\mathbf{L^p}}^{k+1}\big(\boldsymbol{\phi}(\mathbf{x}^k)\big) \cdot \Big(\Big[\nabla^k\boldsymbol{\phi}(\mathbf{x}^k)\Big]^T \cdot \mathbf{e}^k(\mathbf{x}^k)\Big) \\
&= \Big[\mathbf{e}^k(\mathbf{x}^k)\Big]^T \cdot \Bigg\{ (N^{k+1})^{\frac{2}{n}} \Bigg(\int\limits_{\Omega^{k+1}} \Big\{\det H^{k+1}(\mathbf{x}^{k+1})\Big\}^{\frac{p}{2p+n}}\, \mathrm{d}\mathbf{x}^{k+1} \Bigg)^{-\frac{2}{n}} \\
&\qquad\qquad \times \Big\{\det \widehat{H^{k+1}}(\mathbf{x}^k)\Big\}^{-\frac{1}{2p+n}}\, \nabla^k\boldsymbol{\phi}(\mathbf{x}^k) \cdot \widehat{H^{k+1}}(\mathbf{x}^k) \cdot \nabla^k\boldsymbol{\phi}^T(\mathbf{x}^k) \Bigg\} \cdot \mathbf{e}^k(\mathbf{x}^k)\,.
\end{aligned}$$

If we create a unit mesh of Ω^k with respect to metric $\mathcal{M}_{\mathbf{L^p}}^{\mathrm{ALE}}$, the mesh generator will enforce:

$$\Big[\mathbf{e}^k(\mathbf{x}^k)\Big]^T \cdot \mathcal{M}_{\mathbf{L^p}}^{ALE}(\mathbf{x}^k) \cdot \mathbf{e}^k(\mathbf{x}^k) = 1, \quad \text{for all edge } \mathbf{e}^k \text{ of } \mathcal{H}^k\,.$$

Rewriting the above calculus upside down, we get the following implication:

$$\Big[\mathbf{e}^k(\mathbf{x}^k)\Big]^T \cdot \mathcal{M}_{\mathbf{L^p}}^{\mathrm{ALE}}(\mathbf{x}^k) \cdot \mathbf{e}^k(\mathbf{x}^k) = 1 \Longrightarrow \Big[\widehat{\mathbf{e}^{k+1}}\Big]^T \cdot \mathcal{M}_{\mathbf{L^p}}^{k+1}(\,\boldsymbol{\phi}(\mathbf{x}^k)\,) \cdot \widehat{\mathbf{e}^{k+1}} = 1\,.$$

Therefore, the deformed mesh is unit for the optimal metric associated with sensor u^{k+1}, meaning that it is optimal to control the interpolation error in $\mathbf{L^p}$ norm of the sensor at t^{k+1}. ■

Proof of ii). Using $\det(\alpha\mathcal{A}) = \alpha^n \det\mathcal{A}$, which is true for any scalar α and any matrix $\mathcal{A}$, we get:

$$\begin{aligned}
\mathcal{C}\left(\mathcal{M}^{\mathrm{ALE}}_{\mathbf{L^p}}\right) &= \int_{\Omega^k} \left[\det\left(\mathcal{M}^{\mathrm{ALE}}_{\mathbf{L^p}}(\mathbf{x}^k)\right)\right]^{\frac{1}{2}} \mathrm{d}\mathbf{x}^k \\
&= \left(D^{\mathrm{ALE}}_{\mathbf{L^p}}\right)^{\frac{n}{2}} \left(\int_{\Omega^k} \left[\det\left(\nabla^k\boldsymbol{\phi}(\mathbf{x}^k)\right)\right] \left[\det\left(\widehat{H^{k+1}}(\mathbf{x}^k)\right)\right]^{-\frac{n}{2(2p+n)}} \left[\det\widehat{H^{k+1}}(\mathbf{x}^k)\right]^{\frac{1}{2}} \mathrm{d}\mathbf{x}^k \right) \\
&= N^{k+1} \left(\int_{\Omega^k} \left[\det\left(\nabla^k\boldsymbol{\phi}\right)\right] \left[\det\left(\widehat{H^{k+1}}(\mathbf{x}^k)\right)\right]^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x}^k \right)^{-1} \\
&\qquad \times \left(\int_{\Omega^k} \left[\det\left(\nabla^k\boldsymbol{\phi}(\mathbf{x}^k)\right)\right]^{\frac{n}{2p+n}} \left[\det\left(\widehat{H^{k+1}}(\mathbf{x}^k)\right)\right]^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x}^k \right) \\
&= N^{k+1} \\
&= \mathcal{C}\left(\mathcal{M}^{k+1}_{\mathbf{L^p}}\left[u^{k+1}\right]\right) . \ \blacksquare
\end{aligned}$$

3.2 Analytical examples

Theorem 5 has been validated on several two- and three-dimensional analytical test cases. A detailed analysis of these examples in terms of mesh quality is provided to assess the practical efficiency of this theorem.

3.2.1 Tests cases description

Two-dimensional examples. The considered domain is $\Omega = [-1,\,1] \times [-1,\,1]$. We define the following four analytical functions, the graphs of which are depicted in Figure 3.2.

$$\begin{aligned}
u_0^{k+1}(x,y) &= x^2 + y^2 \\
u_1^{k+1}(x,y) &= \begin{cases} 0.01\sin(50xy) & \text{if } |xy| \le \dfrac{\pi}{50} \\ \sin(50xy) & \text{if } |xy| \le \dfrac{2\pi}{50} \end{cases} \\
u_2^{k+1}(x,y) &= 0.1\sin(50x) + \arctan\left(\frac{0.1}{\sin(5y) - 2x}\right) \\
u_3^{k+1}(x,y) &= \arctan\left(\frac{0.1}{\sin(5y) - 2x}\right) + \arctan\left(\frac{0.5}{\sin(3y) - 7x}\right)
\end{aligned}$$

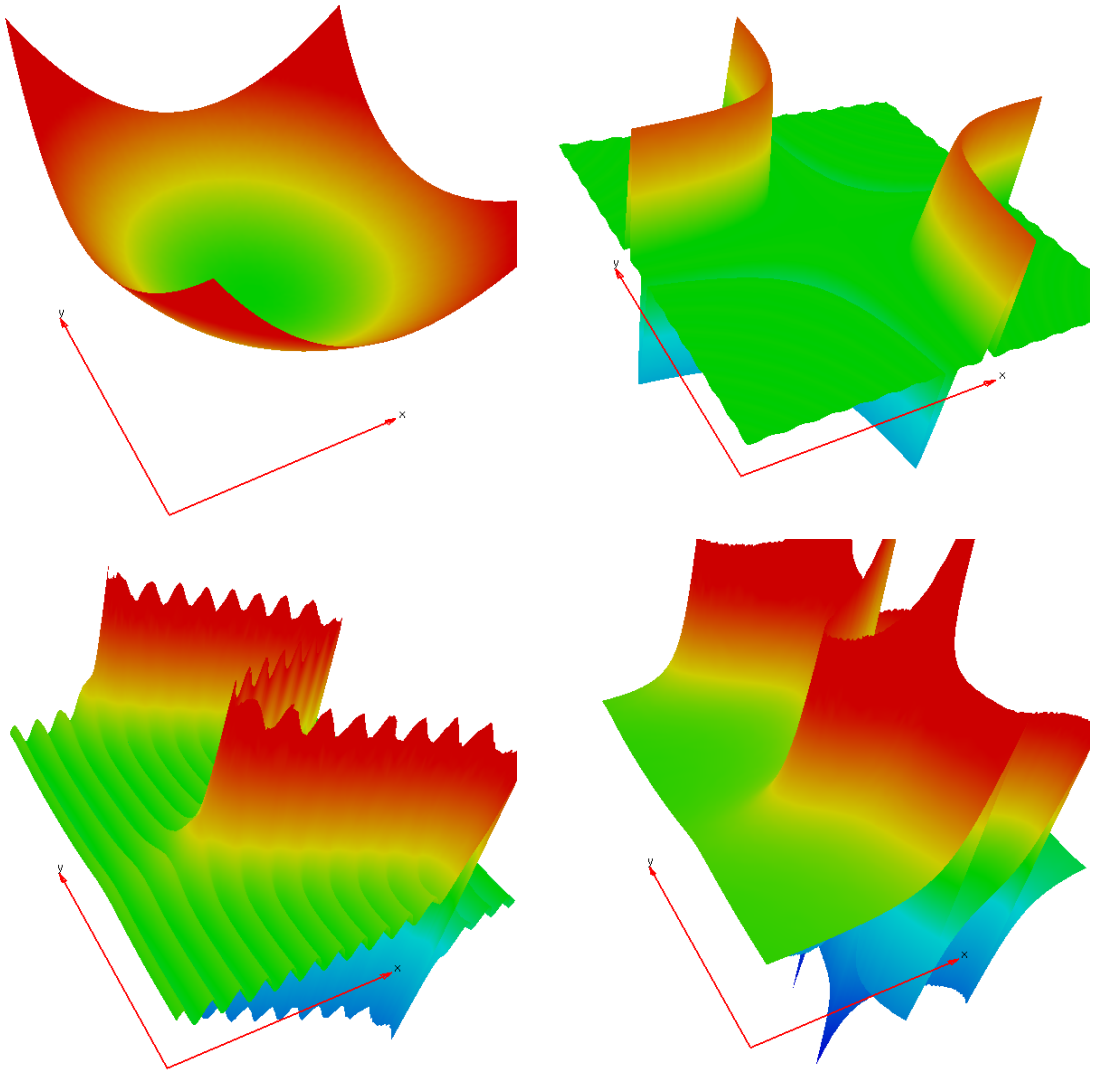


Figure 3.2: Representation of analytical functions u_0^{k+1} (top left), u_1^{k+1} (top right), u_2^{k+1} (bottom left) and u_3^{k+1} (bottom right).

Two analytical displacements are also introduced:

$$\mathbf{d}_1(x,y) = \begin{bmatrix} \begin{cases} -0.3\,(x+1)\left(y^2-1\right)\exp\left(-5x^2\right), & \text{if } x \geq 0 \\ 0.3\,(x-1)\left(y^2-1\right)\exp\left(-5x^2\right), & \text{if } x < 0 \end{cases} \\ \\ \begin{cases} -0.3\left(x^2-1\right)(y+1)\exp\left(-5y^2\right), & \text{if } y \geq 0 \\ 0.3\left(x^2-1\right)(y-1)\exp\left(-5y^2\right), & \text{if } y < 0 \end{cases} \end{bmatrix}$$

$$\mathbf{d}_2(x,y) = \begin{bmatrix} 0.5\left(x^2-1\right)\left(y^2-1\right) \\ 0 \end{bmatrix}$$

Note that these displacements are such that the mesh remains fixed on the domain boundaries. This is mandatory as moving mesh vertices on the boundary is a hard task. Actually, in this specific case, the boundary is piecewise straight so we could move boundary vertices without damaging the geometry description. But in the standard case, boundaries are curved and moving vertices on the discrete boundaries requires either the knowledge of the CAD patches used to design the geometry or the ability to rebuilt an accurate continuous surface model of the boundary from its discretization. In any case, this represents a lot of work. We therefore limit ourselves to mesh displacements satisfying a Dirichlet condition on the domain boundaries: $\mathbf{d}_{|\partial\Omega} = \mathbf{0}$.

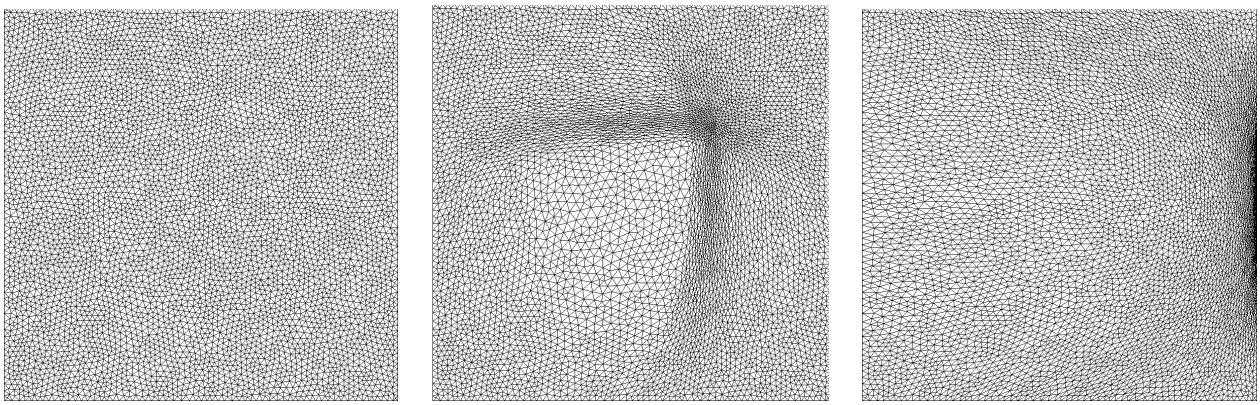


Figure 3.3: Initial mesh (left), moved mesh after displacement $\mathbf{d}_1$ (middle) and $\mathbf{d}_2$ (right).

Three dimensional examples. For the three-dimensional analytical examples, the domain is $\Omega = [-1, 1]^3$. The following analytical sensor function at t^{n+1} are considered:

$$
\begin{aligned}
u_4^{k+1}(x, y, z) &= x^2 + y^2 + z^2 \\
u_5^{k+1}(x, y, z) &= \begin{cases} 0.01 \sin(50xy) & \text{if } |xy| \leq \dfrac{\pi}{50} \\ \sin(50xy) & \text{if } |xy| \leq \dfrac{2\pi}{50} \end{cases}
\end{aligned}
$$

along with these two analytical displacements:

$$\mathbf{d}_3(x,y) = \begin{bmatrix} 0 \\ \begin{cases} -0.3\,(y+1)\left(z^2-1\right)\exp\left(-5y^2\right), & \text{if } x \geq 0 \\ 0.3\,(y-1)\left(z^2-1\right)\exp\left(-5y^2\right), & \text{if } x < 0 \end{cases} \\ \begin{cases} -0.3\left(y^2-1\right)(z+1)\exp\left(-5z^2\right), & \text{if } y \geq 0 \\ 0.3\left(y^2-1\right)(z-1)\exp\left(-5z^2\right), & \text{if } y < 0 \end{cases} \end{bmatrix}$$

$$\mathbf{d}_4(x,y) = \frac{0.8}{\sqrt{2}}\,(x^2-1)(y^2-1)(z^2-1)\begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

Functions u_4^{k+1} and u_5^{k+1} are depicted in Figure 3.4. Note that function u_5^{k+1} exhibits features of

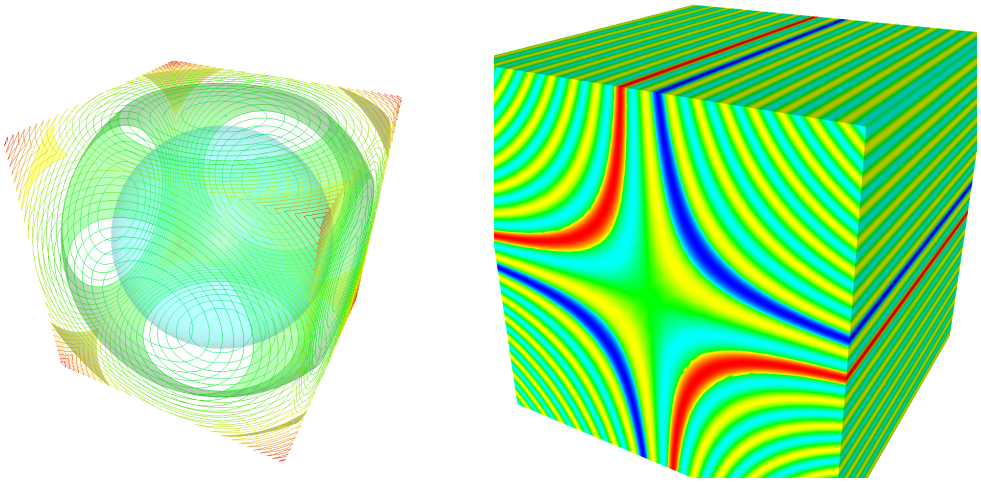


Figure 3.4: Iso-surfaces of three-dimensional quadratic analytical function u_4^{k+1} (left) and iso-values of u_5^{k+1} (right).

different scales, small oscillations of amplitude 0.01 and large oscillations of amplitude 1, which makes it especially suitable for the validation of the multi-scale mesh adaptation process.

Protocol. All these computations start with a uniform mesh $\mathcal{H}^0$. The target complexity is set to $N^{k+1} = 7000$ for two-dimensional examples and to $N^{k+1} = 70000$ for three-dimensional examples. We want to control the interpolation error on u^{k+1} in $\mathbf{L^1}$ norm. The loop described in Figure 3.5 is performed for each target functional/prescribed displacement combination.

The gradient $\nabla^k\mathbf{d} = (\nabla^k d^1, \nabla^k d^2, \nabla^k d^3)$ of displacement $\mathbf{d} = (d^1, d^2, d^3)$ is computed as follows:

$$\nabla^k\mathbf{d}(P_i) = \frac{\sum\limits_{K\,\in\,\mathrm{Ball}(P_i)} |K|\,\nabla^k\mathbf{d}_{|K}}{\sum\limits_{K\,\in\,\mathrm{Ball}(P_i)} |K|},$$

For $m = 1, \ldots, n_{adap}$

$\mathbf{d}_i^k = \texttt{ComputeDisplacement}(\mathcal{H}_i^k)$
$\nabla^k \boldsymbol{\phi}_i = \texttt{ComputeTransformationGradient}(\mathcal{H}_i^k, \mathbf{d}_i^k)$
$\mathcal{H}_i^{k+1} = \texttt{MoveMesh}(\mathcal{H}_i^n, \mathbf{d}_i^k)$
$u_i^{k+1} = \texttt{ComputeTargetSensor}(\mathcal{H}_i^{k+1})$
$\mathcal{M}_{\mathbf{L}^p,i}^{k,ALE} = \texttt{ComputeALEMetric}(\mathcal{H}_i^{k+1}, u_i^{k+1}, \nabla^k \boldsymbol{\phi}_i)$
$\mathcal{H}_{i+1}^k = \texttt{AdaptMesh}(\mathcal{H}_i^k, \mathcal{M}_{\mathbf{L}^p,i}^{k,ALE})$

End for.

Figure 3.5: Procedure applied on two and three dimensional analytical test cases.

with

$$(\nabla^k \mathbf{d}_j)_{|_K} = \frac{1}{2|K|} \sum_{l=0}^{2} d^j(P_l) \overline{\boldsymbol{\eta}}_l \text{ in 2D} , \quad (\nabla^k \mathbf{d}^j)_{|_K} = \frac{1}{6|K|} \sum_{l=0}^{3} d^j(P_l)\, \overline{\boldsymbol{\eta}}_l \text{ in 3D} ,$$

and we have noted Ball(P_i) the ball of vertex P_i, and $\overline{\boldsymbol{\eta}}_l$ the non-normalized inward normal of face l of element K.

3.2.2 Results analysis.

The adapted mesh of Ω_h^k obtained for each functional $u^{k+1} = u_j$ when mesh displacement is $\mathbf{d}_i$ is noted $\mathcal{H}_i^k(u_j)$. These meshes are unit with respect to metric $\mathcal{M}_{\mathbf{L}^1}^{\text{ALE}}$ defined by Formula (3.2). According to the above developments, we expect the image mesh $\mathcal{H}_i^{k+1}(u_j)$ of $\mathcal{H}_i^k(u_j)$ obtained when the vertices of $\mathcal{H}_i^k(u_j)$ are moved according to $\mathbf{d}_i$ to be optimal for the control of the interpolation error of u^{k+1} in $\mathbf{L}^1$ norm. To check this, the resulting image mesh $\mathcal{H}_i^{k+1}(u_j)$ obtained by applying our new methodology when $\mathcal{H}_i^k(u_j)$ is moved according to $\mathbf{d}_i$, is compared to the optimal adapted mesh $\mathcal{H}_{\mathbf{L}^1}(u_j)$ obtained with direct steady multi-scale adaptation, see Chapter 1.

For each function $u^{k+1} = u_j$ and each displacement $\mathbf{d}_i$, the progressive transformation of the mesh while its vertices are moved along displacement field $\mathbf{d}_i$ is shown. For two-dimensional analytical functions, the adapted deformed mesh is described by a sequence of 6 images at 6 different times $t^k + \alpha\left(t^{k+1} - t^k\right)$, with $\alpha \in \{0,\ 13/60,\ 25/60,\ 37/60,\ 43/60,\ 1\}$, see Figures 3.8, 3.9 for u_0, see Figures 3.11, 3.12 for u_1, see Figures 3.14 and 3.15 for u_2 and finally, see Figures 3.17 and 3.18. For three-dimensional test cases, the adapted deformed mesh is described by a sequence of 6 images at times $t^k + \alpha\left(t^{k+1} - t^k\right)$, with $\alpha \in \{0,\ 3/20,\ 7/20,\ 11/20,\ 15/20,\ 1\}$. As can be seen on all these examples, meshes $\mathcal{H}_i^k(u_j)$ are indeed transformed into a nearly optimal mesh for sensor u^{k+1} at t^{k+1}, in two dimensions as well as in three dimensions, see Figure.

The quality of final meshes obtained by moving mesh adaptation is compared to the one obtained when directly performing a standard steady mesh adaptation on u_j. A detailed comparison

between the mesh obtained by direct adaptation and the final meshes obtained after $\mathcal{H}_i^k(u_j)$ is moved of $\mathbf{d}_i$ is given for each test case associated with each displacement. This has been done by computing the quality of all the elements of these two meshes with respect to optimal $\mathbf{L}^1$ metric $\mathcal{M}_{\mathbf{L}^1}$. $\mathcal{M}_{\mathbf{L}^1}$ that has been used to generate $\mathcal{H}_{\mathbf{L}^1}$. The quality criterium is the following, where $(\mathbf{e}_l)_l$ denote the edges of the considered element K:

$$Q_{\mathcal{M}}(K) = \frac{\sqrt{3}}{216} \frac{\left(\sum_{l=1}^{6} \ell^2_{\mathcal{M}}(\mathbf{e_i})\right)^{\frac{3}{2}}}{|\,K\,|_{\mathcal{M}}}.$$

If $Q_{\mathcal{M}}(K)$ is near from 1, then element K is almost regular with respect to $\mathcal{M}_{\mathbf{L}^1}$. On the contrary, the biggest $Q_{\mathcal{M}}(K)$, the worst the quality. For each analytical sensor u^{k+1}, the quality comparison between classically adapted mesh $\mathcal{H}_{\mathbf{L}^1}$ and the images of ALE-adapted meshes $\mathcal{H}_i^{k+1}(u_j)$ for each $\mathbf{d}_i$ are provided, along with detailed quality histograms.
All these results are gathered at the end of this chapter.

First, it can be observed that the qualities of the meshes $\mathcal{H}_{\mathbf{L}^1}(u_j)$ obtained by direct adaptation are excellent. This is even more striking for 3D dimensional examples: for functional u_5, the average quality of directly adapted mesh $\mathcal{H}_{\mathbf{L}^1}(u_5)$ is 1.22 and the worst quality is 8.31. This demonstrated the efficiency of the 3D anisotropic adaptive re-mesher, even on strongly anisotropic functions like u_5.

Two-dimensional test cases are very convincing. The moving mesh applied on quadratic sensor u_0 shows the consistency of the method as we manage to a get a uniform mesh at t^{k+1}, as expected. The mesh movement is indeed taken into account: the moving mesh metric tends to reduce mesh sizes in some areas, anticipating the future stretching of these regions thanks to the gradient of $\mathbf{d}_i$. The study of the meshes qualities shows that meshes that has been generated at t^k thanks to the ALE metric are almost perfectly adapted to the corresponding analytical sensor $u^{k+1} = u_j$ once they have been moved.
For analytical functions u_2 and u_3 some bad elements are generated. These bad element are located near the domain boundary in regions of high anisotropy. This is due to the constraint imposed by the "no vertex displacement on the boundary" condition, which locally hinders the generation of the optimal anisotropic mesh near the boundary.

Three-dimensional examples are also very demonstrative. First, when applied to quadratic 3D sensor u_4 the expected uniform final mesh is obtained after ALE-adapted mesh $\mathcal{H}_i^k(u_4)$ at t^k as been moved of $\mathbf{d}_i$. This is observed for "pseudo-2D" displacement $\mathbf{d}_3$ but also for more generic displacement $\mathbf{d}_4$. Regarding u_5, the quality of ALE-adapted meshes $\mathcal{H}_3^{k+1}(u_5)$ and $\mathcal{H}_4^{k+1}(u_5)$ are also very good with an average quality of 1.24 and 1.76, respectively.
However, some bad elements for metric $\mathcal{H}_i^{k+1}(u_5)$ appear: 25 and 180 elements have a quality greater than 10 for $\mathcal{H}_3^{k+1}(u_5)$ and $\mathcal{H}_4^{n+1}(u_5)$, respectively. One of the reason to explain these meshes degradation may be that Relation (3.5) is only true at first order and higher order terms might be necessary when the transformation is complex, which is the case here. Another reason can be the lack of regularity of the ALE metric, which makes it really hard for the re-mesher to build a unit mesh with respect to this metric. To facilitate the task of the mesh generator, we have applied a gradation process on the metric [Alauzet 2010a] to reduce size shocks: this can have a non negligible impact on the image mesh. Nevertheless, the amplitude of displacement

$\mathbf{d}_2$ is large, much more than what is encountered in the moving mesh fixed-point algorithm, thus obtaining such quality results at the end of the mesh displacement is really satisfactory.

3.3 Extension of the fixed-point algorithm to moving mesh simulations

Only few things need to be modified to extend the fixed-point algorithm of Section 2.3 to moving mesh simulations. First, it is important to understand that in the case of moving mesh simulations, physical quantities are not computed at fixed locations (x, y, z) in space, but are associated to moving vertices. Thus, during the sampling stage, each vertex P is followed along its trajectory $\gamma_P(t)$ between t_i and t_{i+1}. At each sampling time $t = t_{i,k}$, the solution associated with this moving vertex $u_{i,k}(P)$ is saved, along with the mesh displacement $\mathbf{d}_{i,k}(P)$ between $P(t_i)$ and $P(t_{i,k})$, see Figure 3.6. All this is gathered to compute an ALE Hessian $|H^*_{i,max}|$ on

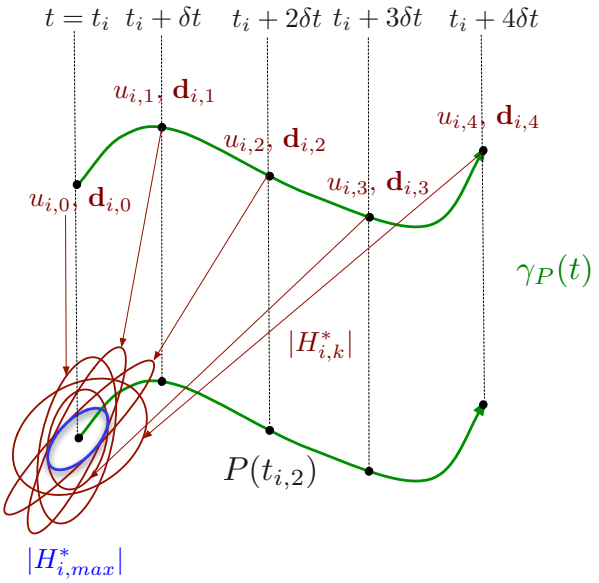


Figure 3.6: Moving vertex trajectory, solution and displacement sampling and sub-interval ALE metric computation.

mesh $\mathcal{H}(t_i)$. Note here that writing the ALE metric under Form 2.17 enable to apply the results of Section 2.3.1, *i.e* guarantees the control of the worst spatial global interpolation error $\mathbf{L}^p$ norm on each adaptation sub-interval. Of course, **this requires that the number of vertices of the mesh remains the same between** t_i **and** t_{i+1}, which is the case for the ALE method we chose, see Part II.

We are now sure that the mesh, even if it is not optimal, will be adapted to the solution during its whole movement between t_i and t_{i+1}. Except for the computation of the gradient of the transformation and the expression of the metric, the structure of the fixed-point algorithm remains the same as in the fixed-mesh case and is described in Figure 3.7.

Applications of this algorithm to moving mesh CFD computations will be presented in Part II, in two and three dimensions.

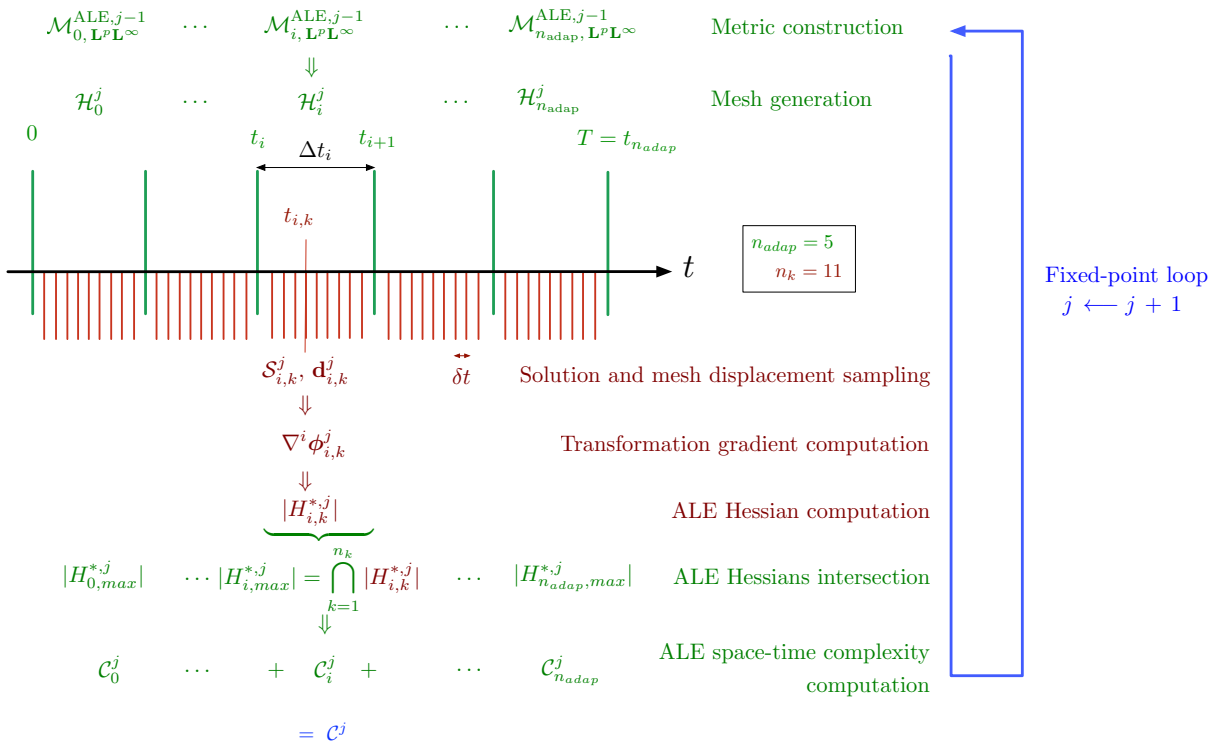


Figure 3.7: Spatial error-driven space-time fixed-point algorithm extended to moving mesh simulations.

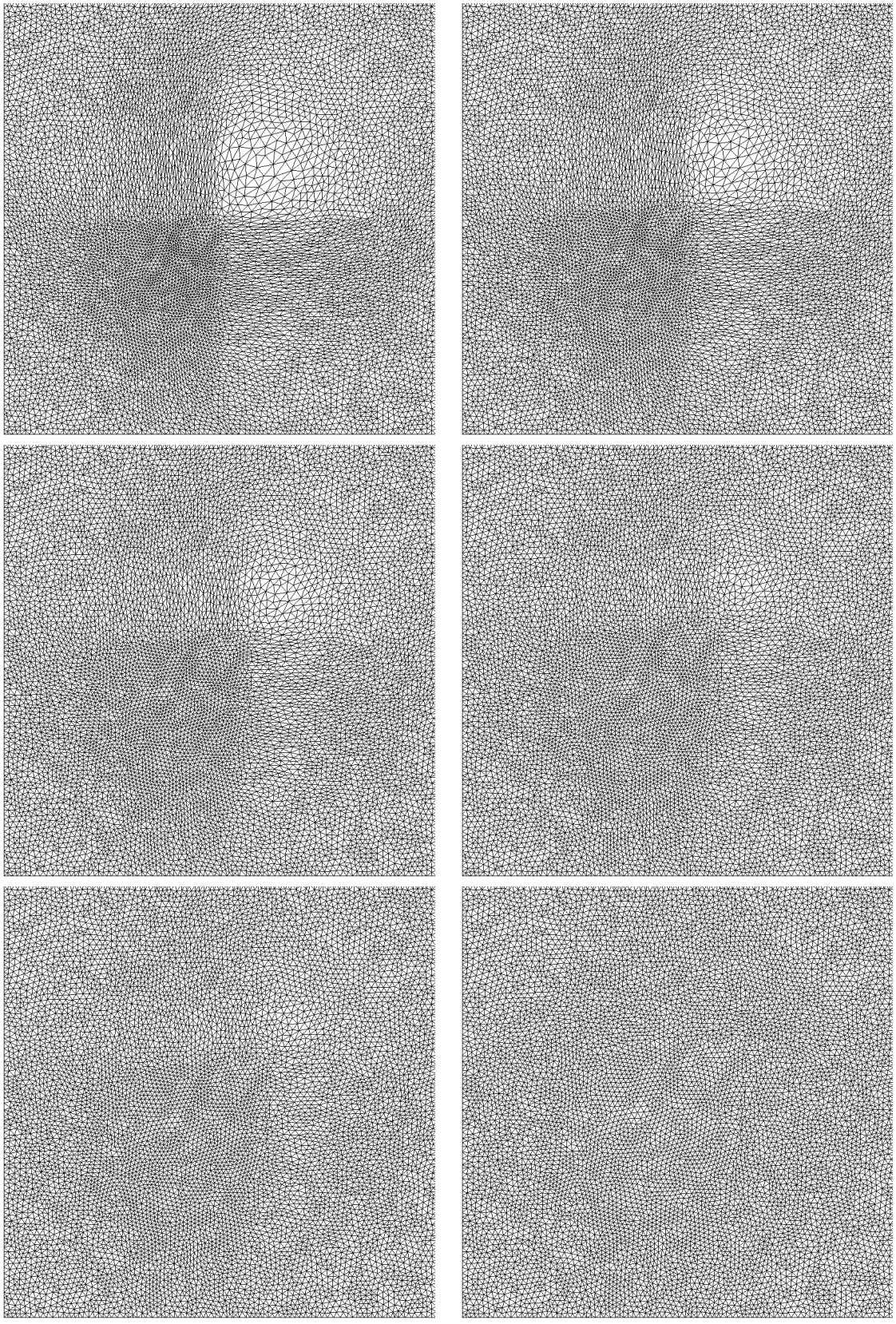


Figure 3.8: Moving-mesh metric-based mesh adaptation in $\mathbf{L}^1$ norm applied to analytical function $u^{k+1} = u_0$ for prescribed mesh displacement $\mathbf{d}_1$. Deformation of the ALE-adapted mesh from t^k to t^{k+1}. Final mesh is adapted to u_0

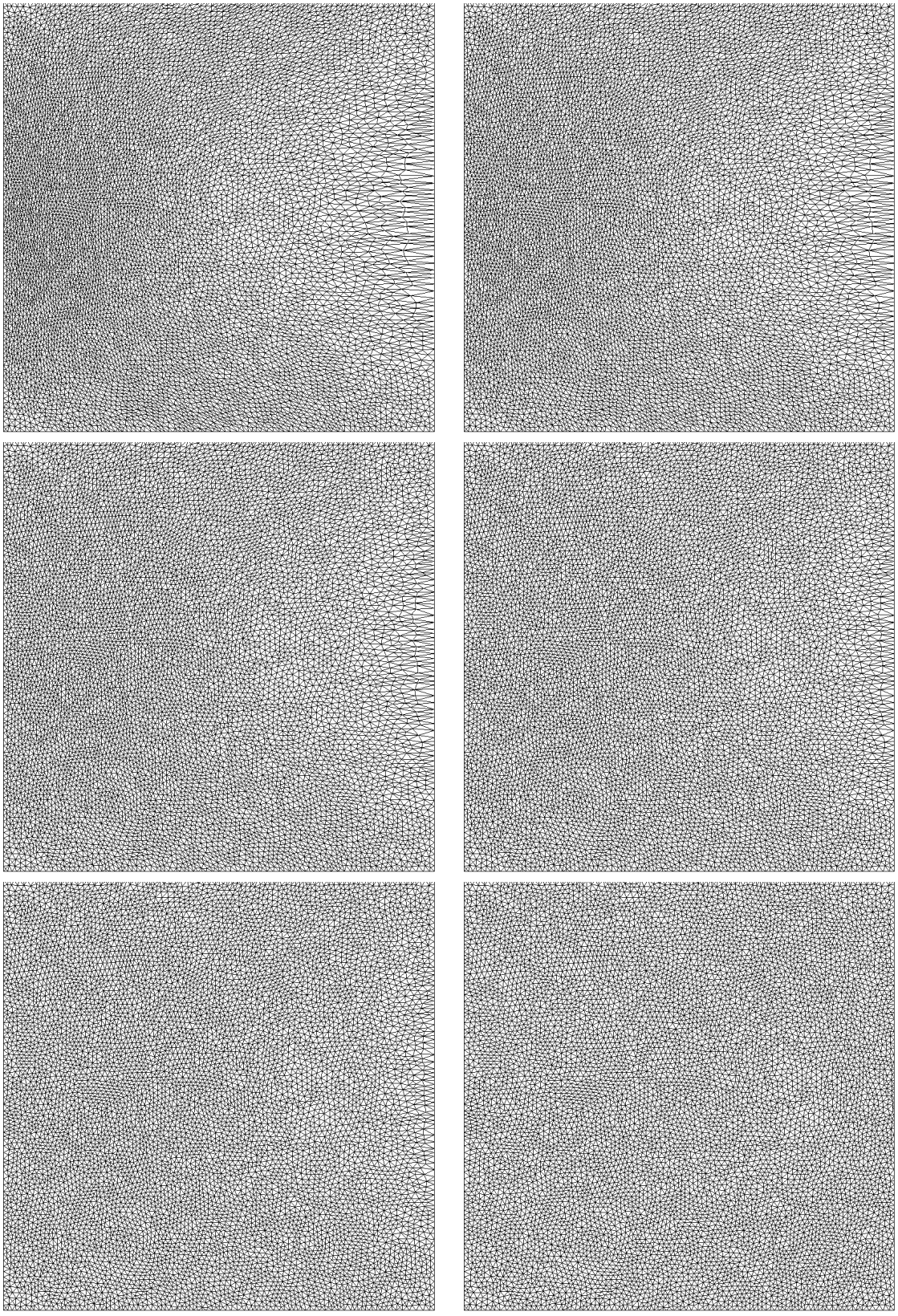
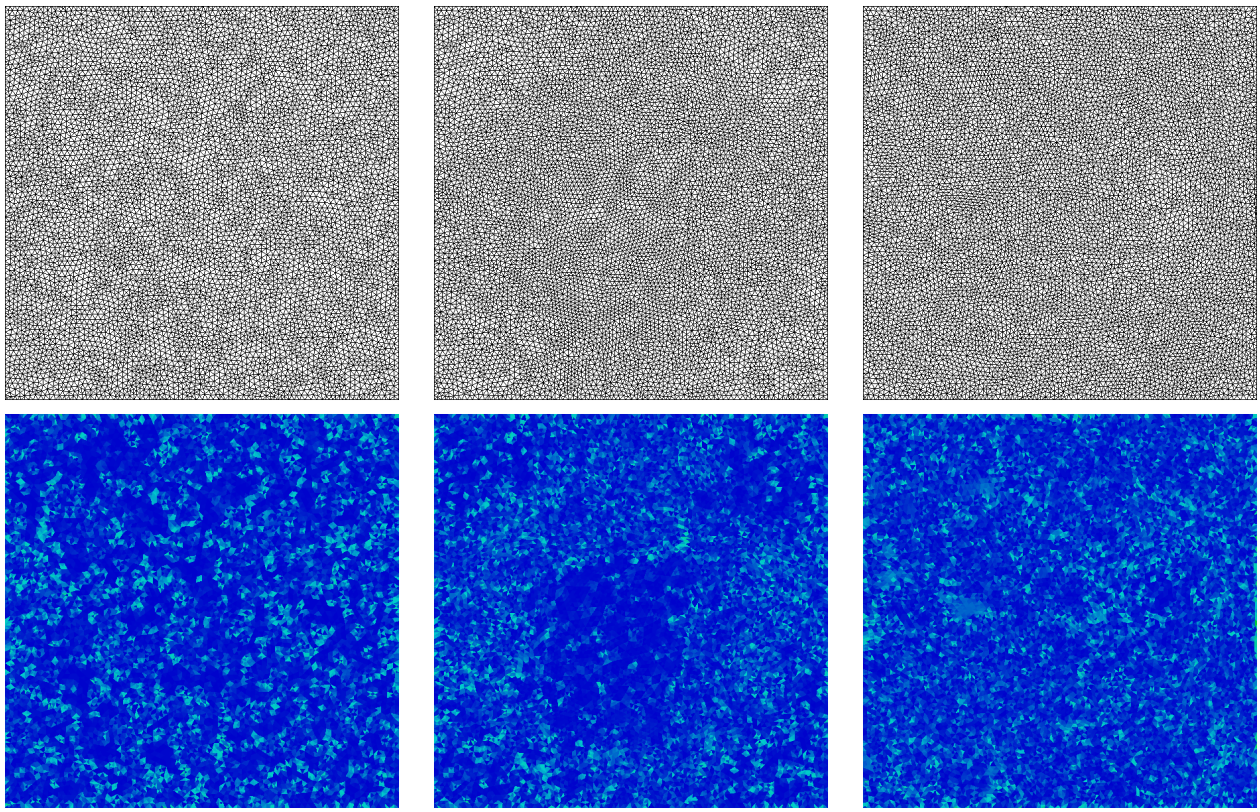


Figure 3.9: Moving-mesh metric-based mesh adaptation in $\mathbf{L}^1$ norm applied to analytical function $u^{k+1} = u_0$ for prescribed mesh displacement $\mathbf{d}_2$. Deformation of the ALE-adapted mesh from t^k to t^{k+1}. Final mesh is adapted to u_0



	$\mathcal{H}_{\mathbf{L}^1}(u_0)$		$\mathcal{H}_1^{k+1}(u_0)$ (with $\mathbf{d}_1$)		$\mathcal{H}_2^{k+1}(u_0)$ (with $\mathbf{d}_2$)	
Number of vertices	7652		8633		8860	
Number of triangles	14939		16898		17360	
Average quality	1.071505		1.064457		1.070742	
Best quality	1.000001		1.000002		1.000005	
Worst quality	1.789929		1.614584		2.163519	
$1.00 < Q < 2.00$	14939	100.00 %	16898	100.00 %	17358	99.99 %
$2.00 < Q < 3.00$	0	0.00 %	0	0.00 %	2	0.01 %

Figure 3.10: Results obtained on u_0 with true adaptation (left), using moving mesh adaptation with $\mathbf{d}_1$ (middle) and $\mathbf{d}_2$ (right). Top: final meshes. Bottom: elements quality in the target metric at t^{k+1}.

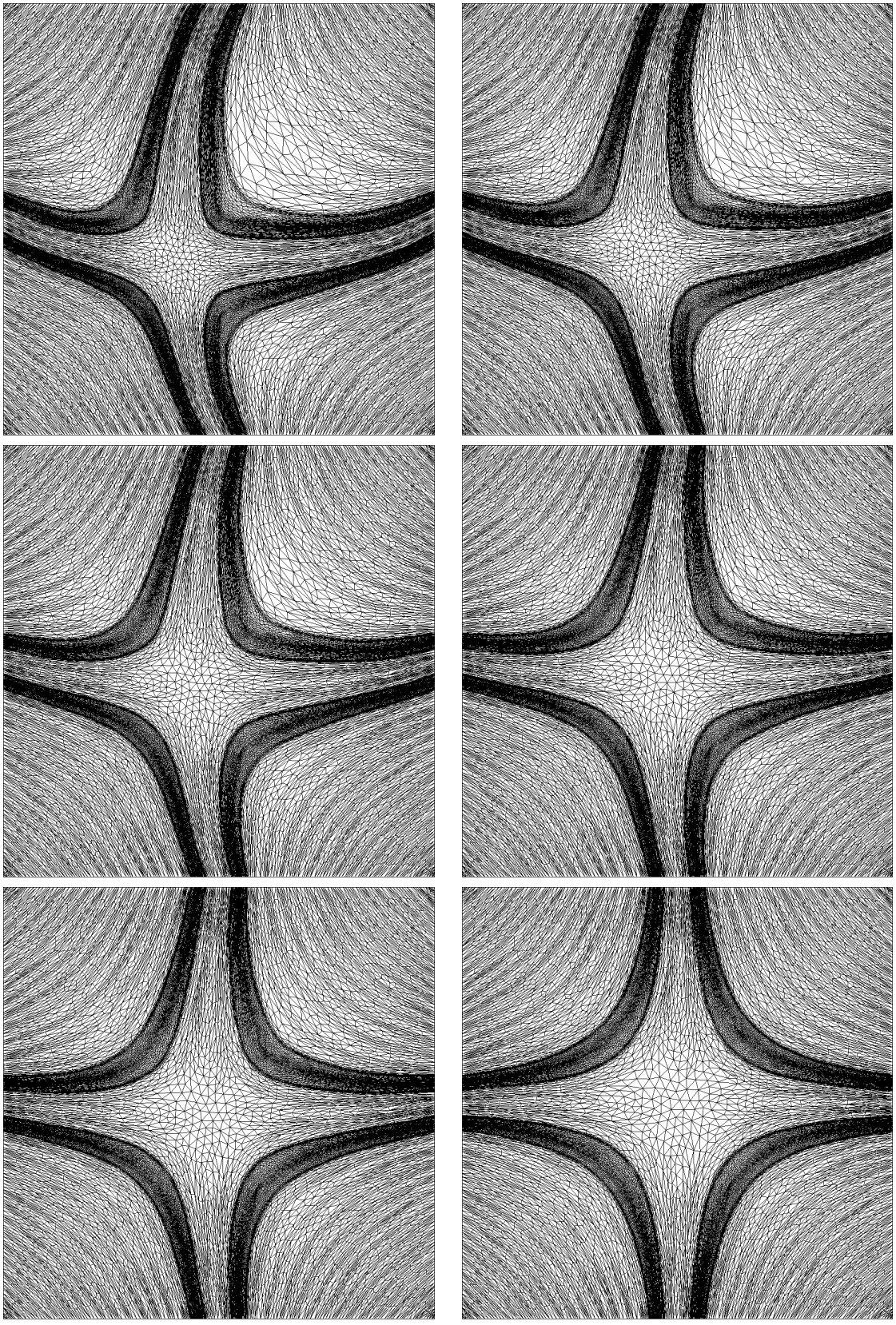


Figure 3.11: Moving-mesh metric-based mesh adaptation in $\mathbf{L}^1$ norm applied to analytical function $u^{k+1} = u_1$ for prescribed mesh displacement $\mathbf{d}_1$. Deformation of the ALE-adapted mesh from t^k to t^{k+1}. Final mesh is adapted to u_1.

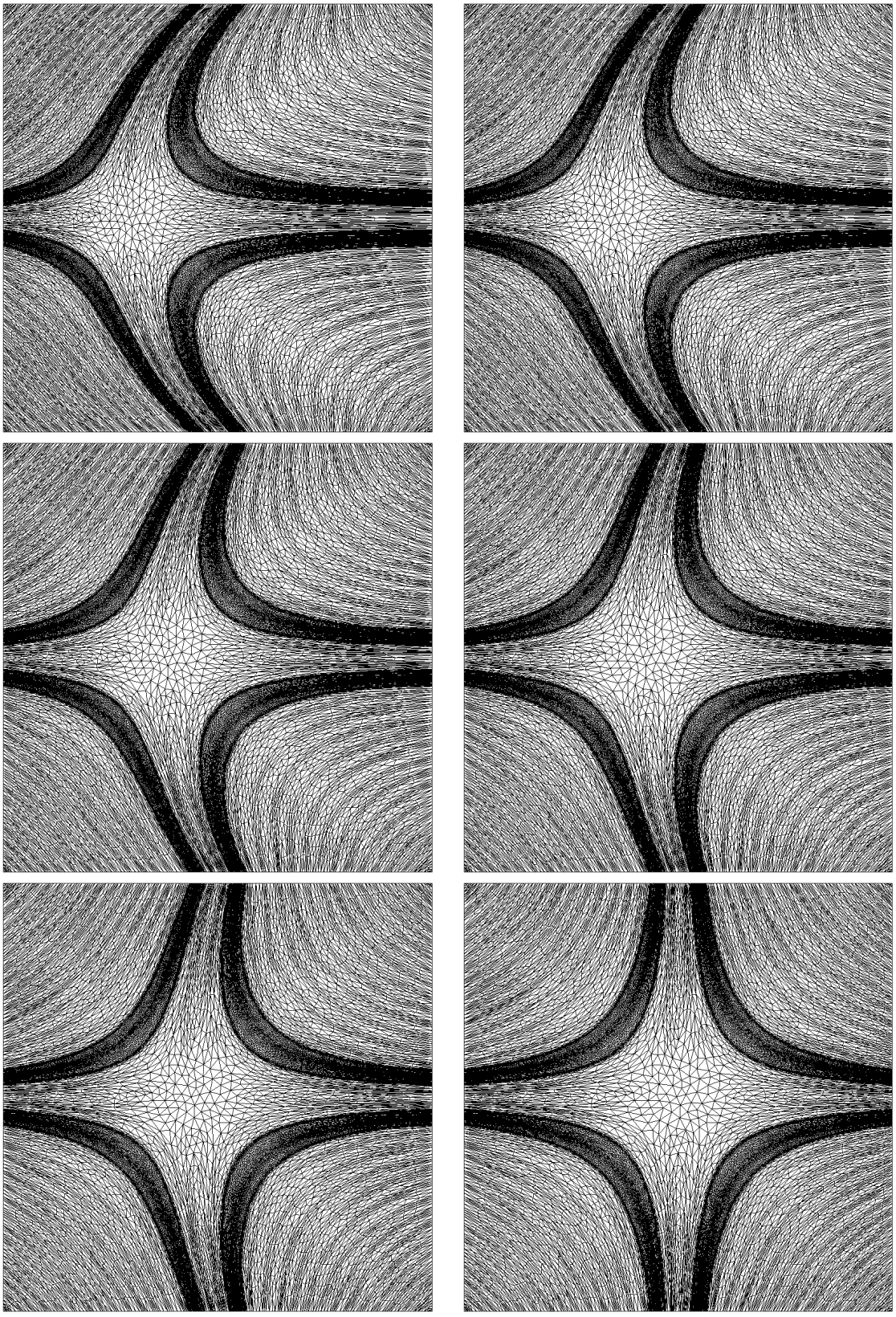
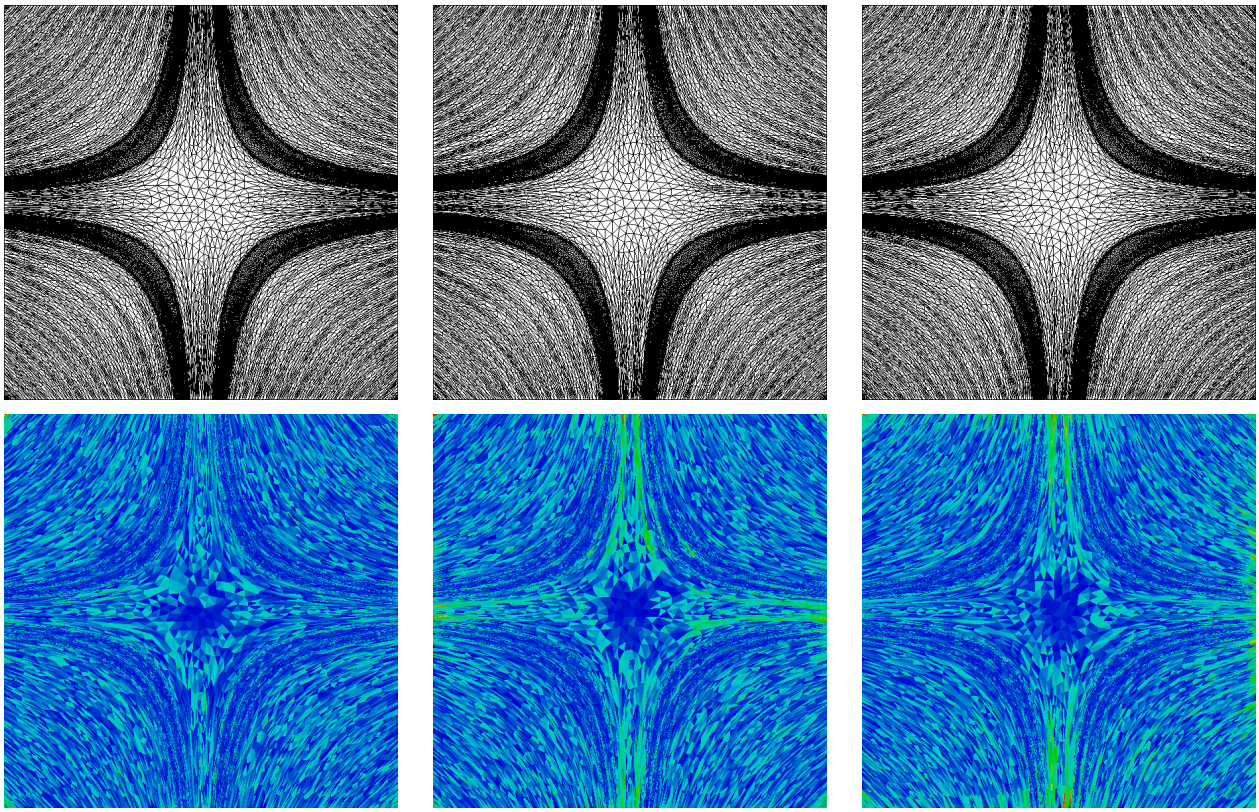


Figure 3.12: Moving-mesh metric-based mesh adaptation in $\mathbf{L}^1$ norm applied to analytical function $u^{k+1} = u_1$ for prescribed mesh displacement $\mathbf{d}_2$. Deformation of the ALE-adapted mesh from t^k to t^{k+1}. Final mesh is adapted to u_1.



	$\mathcal{H}_{\mathbf{L}^1}(u_1)$		$\mathcal{H}_1^{k+1}(u_1)$ (with $\mathbf{d}_1$)		$\mathcal{H}_2^{k+1}(u_1)$ (with $\mathbf{d}_2$)	
Number of vertices	11035		11498		11612	
Number of triangles	21168		22075		22306	
Average quality	1.135740		1.173252		1.152363	
Best quality	1.000013		1.000007		1.000001	
Worst quality	7.415330		18.712967		16.252627	
$1.00 < Q < 2.00$	21129	99.82 %	21835	98.91 %	21997	98.61 %
$2.00 < Q < 3.00$	34	0.16 %	207	0.94 %	249	1.12 %
$3.00 < Q < 4.00$	4	0.02 %	25	0.11 %	45	0.20 %
$4.00 < Q < 5.00$	1	0.00 %	3	0.01 %	5	0.02 %
$5.00 < Q < 10.00$	0	0.00 %	4	0.02 %	9	0.04 %
$10.00 < Q < 100.00$	0	0.00 %	1	0.00 %	1	0.00 %

Figure 3.13: Results obtained on u_1 with true adaptation (left) and using moving-mesh adaptation with prescribed displacement $\mathbf{d}_1$ (middle) and $\mathbf{d}_2$ (right). Top: final meshes. Bottom: elements quality in the target metric at t^{k+1}. Worse elements are located near the boundary and in strongly anisotropic areas, as in classical adaptation.

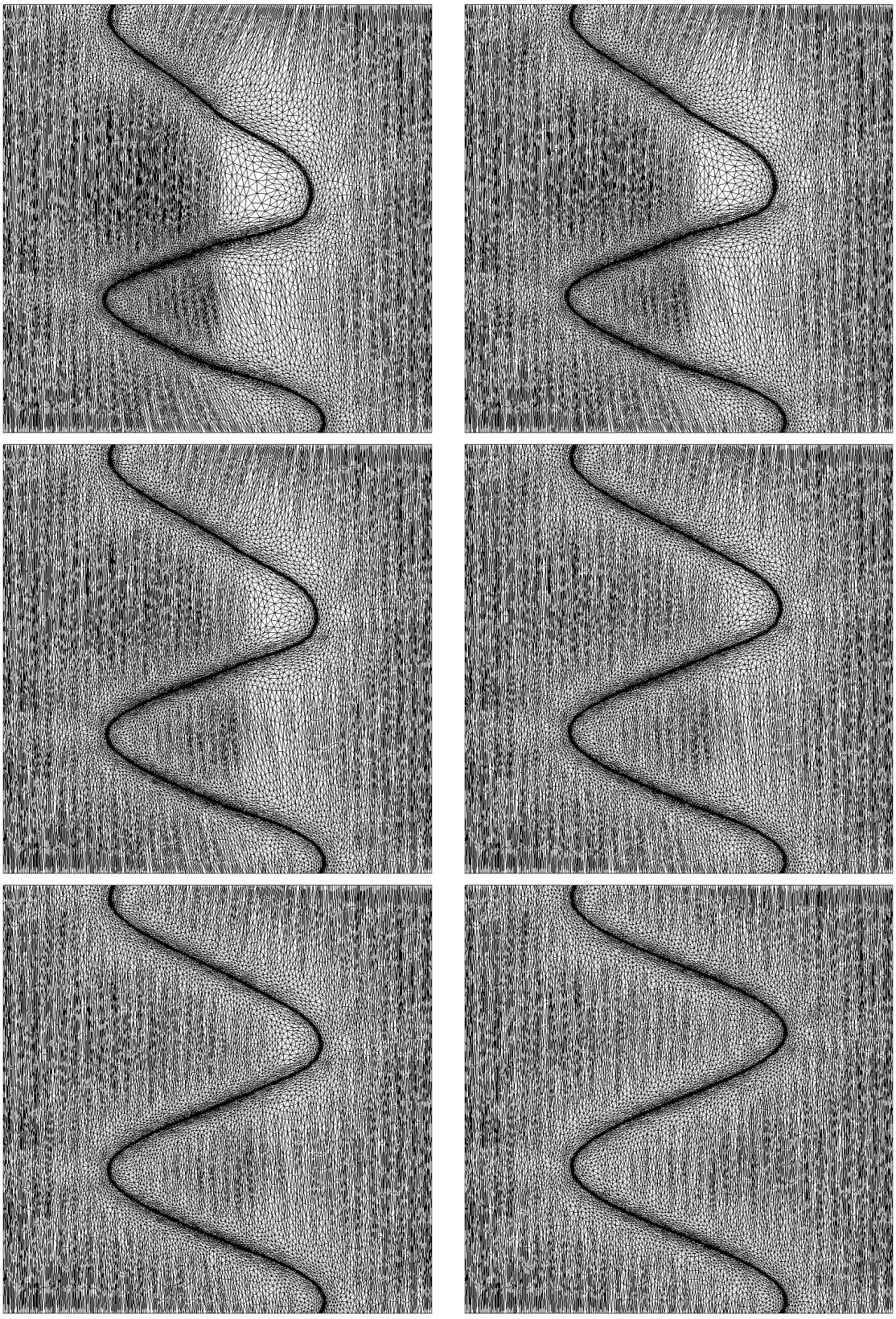


Figure 3.14: Moving-mesh metric-based mesh adaptation in $\mathbf{L}^1$ norm applied to analytical function $u^{k+1} = u_2$ for prescribed mesh displacement $\mathbf{d}_1$. Deformation of the ALE-adapted mesh from t^k to t^{k+1}. Final mesh is adapted to u_2.

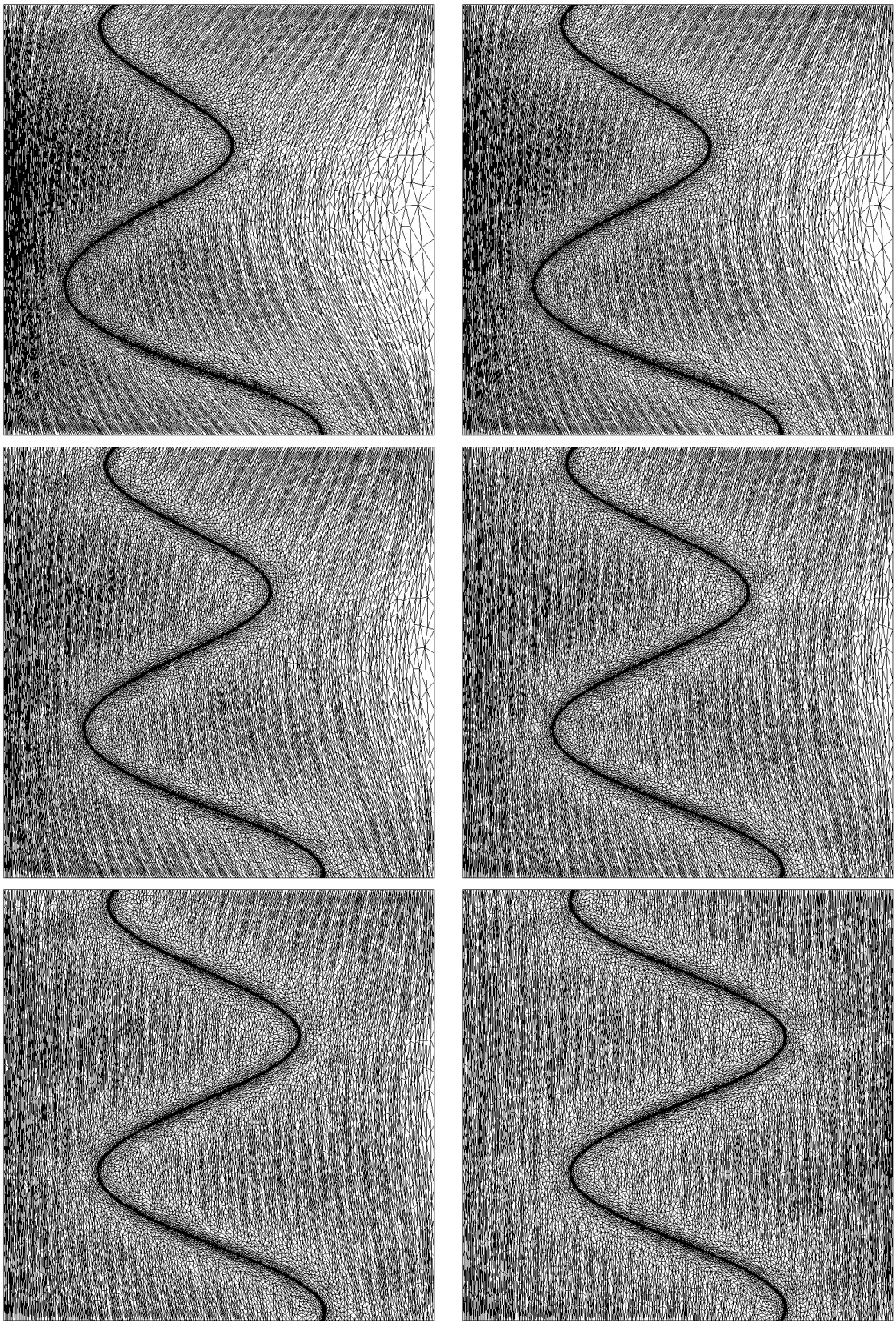
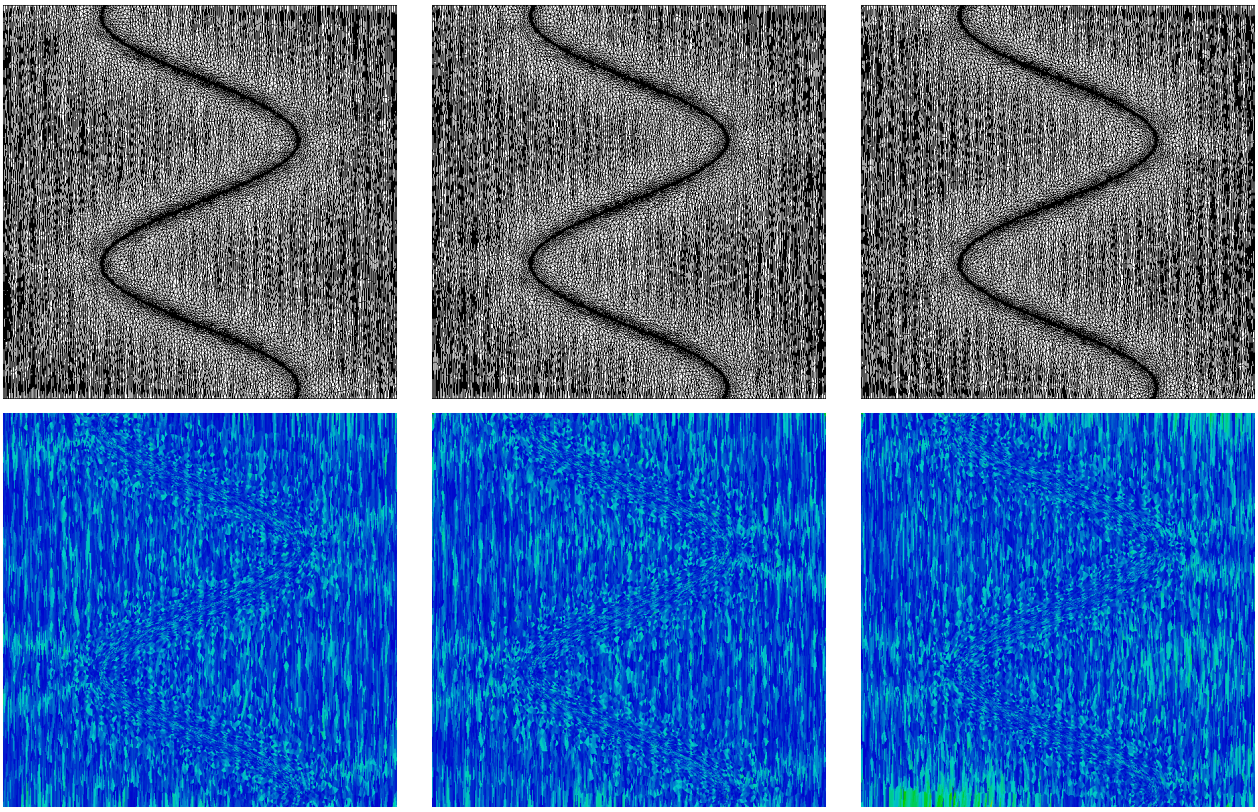


Figure 3.15: Moving-mesh metric-based mesh adaptation in $\mathbf{L}^1$ norm applied to analytical function $u^{k+1} = u_2$ for prescribed mesh displacement $\mathbf{d}_2$. Deformation of the ALE-adapted mesh from t^k to t^{k+1}. Final mesh is adapted to u_2.



	$\mathcal{H}_{\mathbf{L}^1}(u_2)$		$\mathcal{H}_1^{k+1}(u_2)$ (with $\mathbf{d}_1$)		$\mathcal{H}_2^{k+1}(u_2)$ (with $\mathbf{d}_2$)	
Number of vertices	10279		10386		10628	
Number of triangles	20125		20342		20819	
Average quality	1.084487		1.088273		1.097077	
Best quality	1.000003		1.000003		1.000005	
Worst quality	2.192020		2.501825		3.301007	
$1.00 < Q < 2.00$	20122	99.99 %	20334	99.96 %	20779	99.81 %
$2.00 < Q < 3.00$	3	0.01 %	8	0.04 %	39	0.19 %
$3.00 < Q < 4.00$	0	0.00 %	0	0.00 %	1	0.00 %

Figure 3.16: Results obtained on u_2 with true adaptation (left), using moving-mesh adaptation with prescribed displacement $\mathbf{d}_1$ (middle) and $\mathbf{d}_2$ (right). Top: Final meshes. Bottom: Elements quality in the target metric at t^{k+1}.

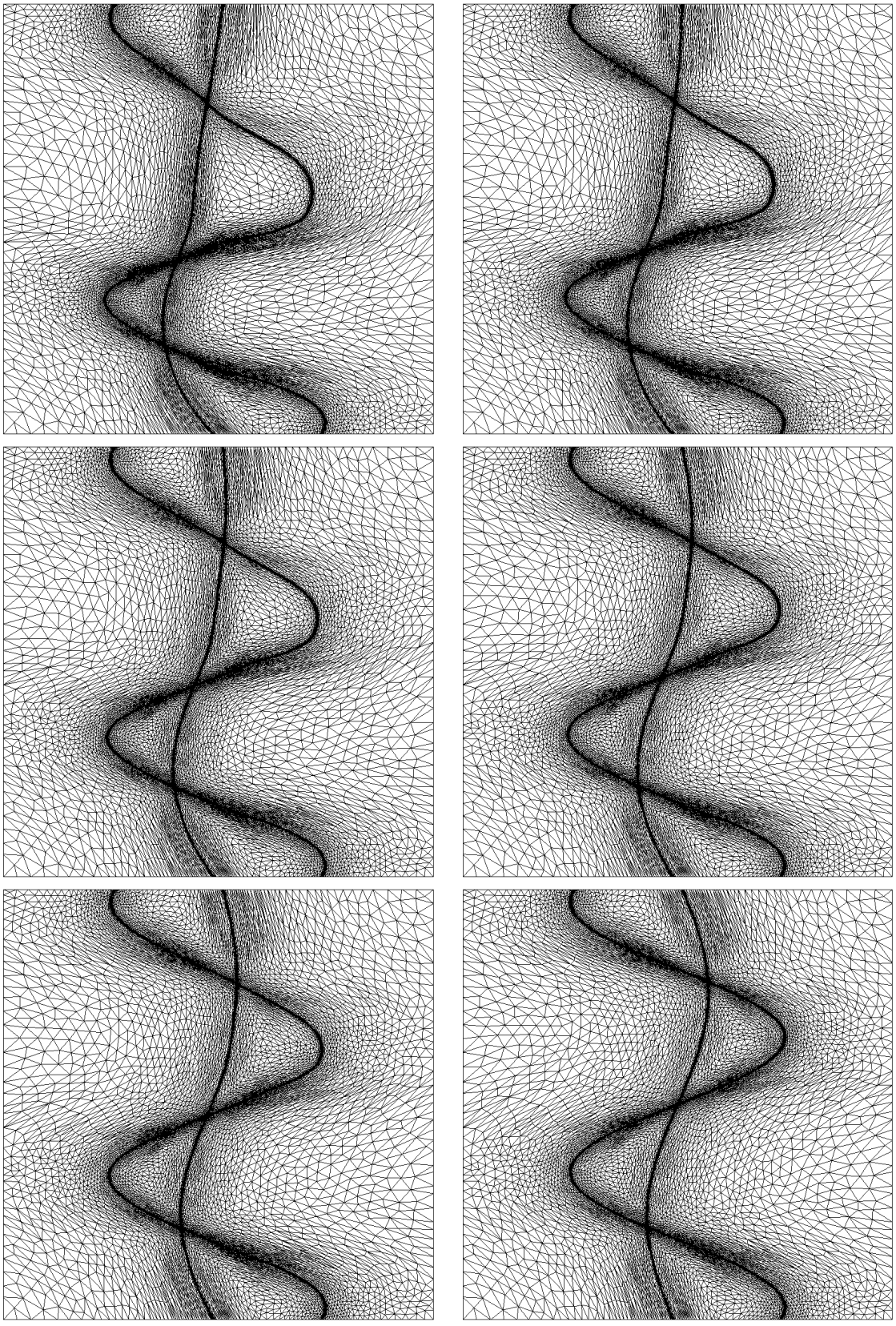


Figure 3.17: Moving-mesh metric-based mesh adaptation in $\mathbf{L}^1$ norm applied to analytical function $u^{k+1} = u_3$ for prescribed mesh displacement $\mathbf{d}_1$. Deformation of the ALE-adapted mesh from t^k to t^{k+1}. Final mesh is adapted to u_3.

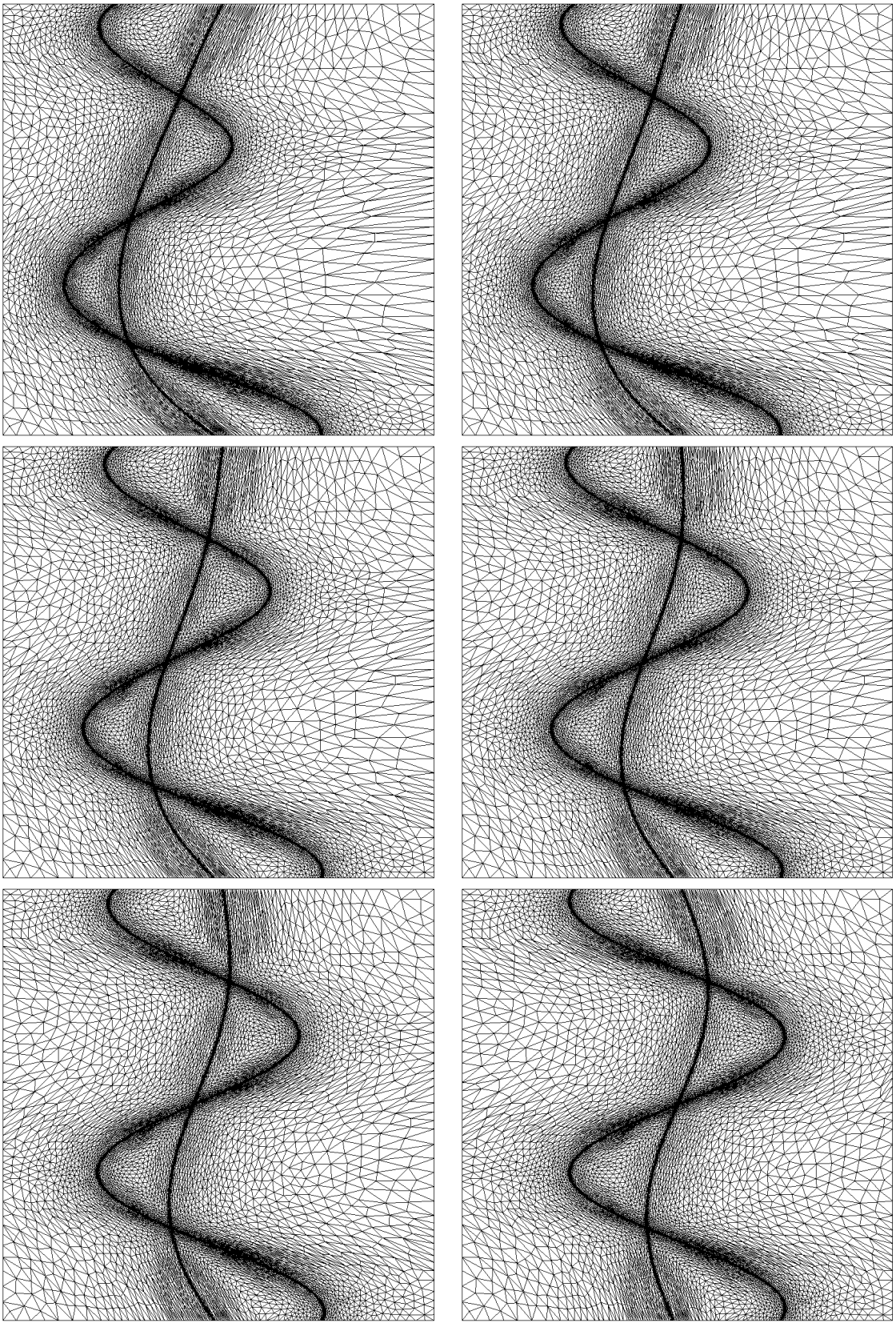
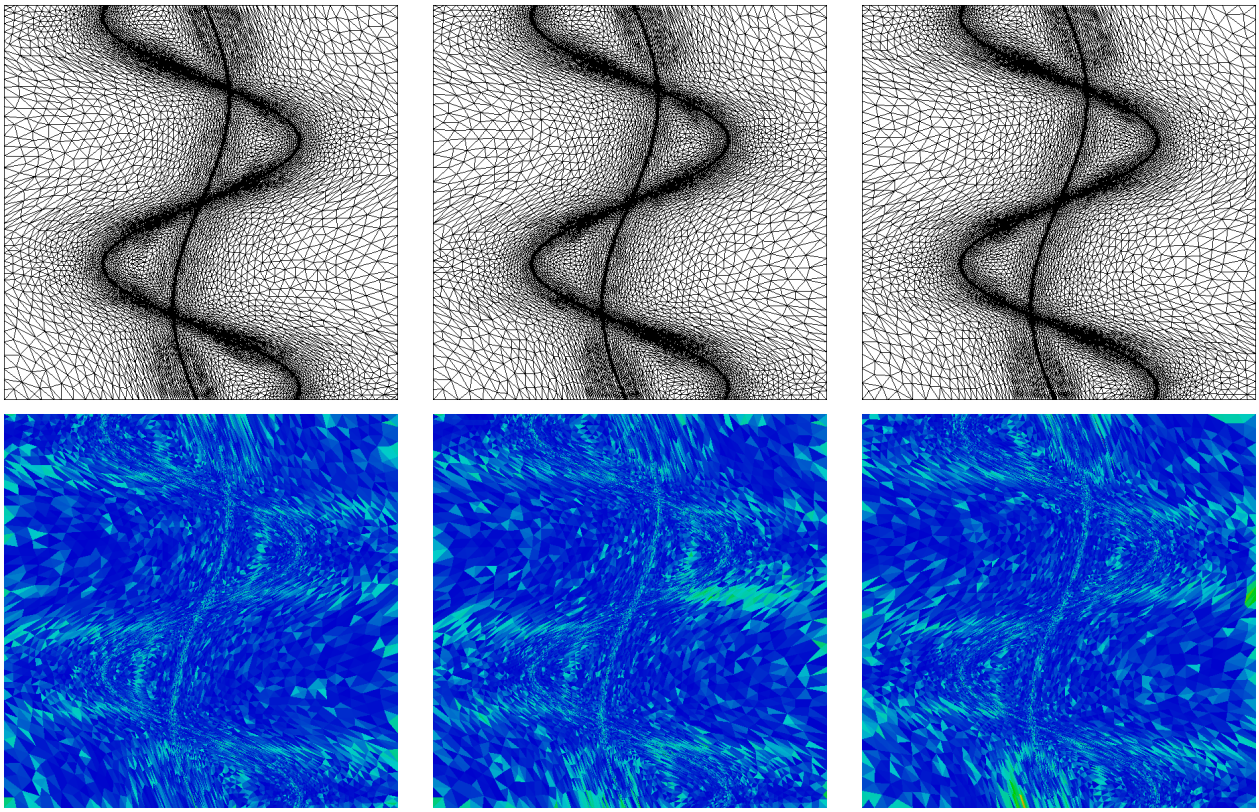


Figure 3.18: Moving-mesh metric-based mesh adaptation in $\mathbf{L}^1$ norm applied to analytical function $u^{k+1} = u_3$ for prescribed mesh displacement $\mathbf{d}_2$. Deformation of the ALE-adapted mesh from t^k to t^{k+1}. Final mesh is adapted to u_3.



	$\mathcal{H}_{\mathbf{L}^1}(u_3)$		$\mathcal{H}_1^{k+1}(u_3)$ (with $\mathbf{d}_1$)		$\mathcal{H}_2^{k+1}(u_3)$ (with $\mathbf{d}_2$)	
Number of vertices	10444		10484		10377	
Number of triangles	20568		20654		20437	
Average quality	1.098263		1.106087		1.102665	
Best quality	1.000004		1.000002		1.000002	
Worst quality	3.260156		2.994176		4.123858	
$1.00 < Q < 2.00$	20543	99.88 %	20611	99.79 %	20389	99.77 %
$2.00 < Q < 3.00$	24	0.12 %	43	0.21 %	44	0.22 %
$3.00 < Q < 4.00$	1	0.00 %	0	0.00 %	3	0.01 %
$4.00 < Q < 5.00$	0	0.00 %	0	0.00 %	1	0.00 %

Figure 3.19: Results obtained on u_3 with true adaptation (left), using moving-mesh adaptation with prescribed displacement $\mathbf{d}_1$ (middle) and $\mathbf{d}_2$ (right). Top: Final meshes. Bottom: Elements quality in the target metric at t^{k+1} and quality histograms. Worse elements are located near the boundary and in strongly anisotropic areas, as in classical adaptation.

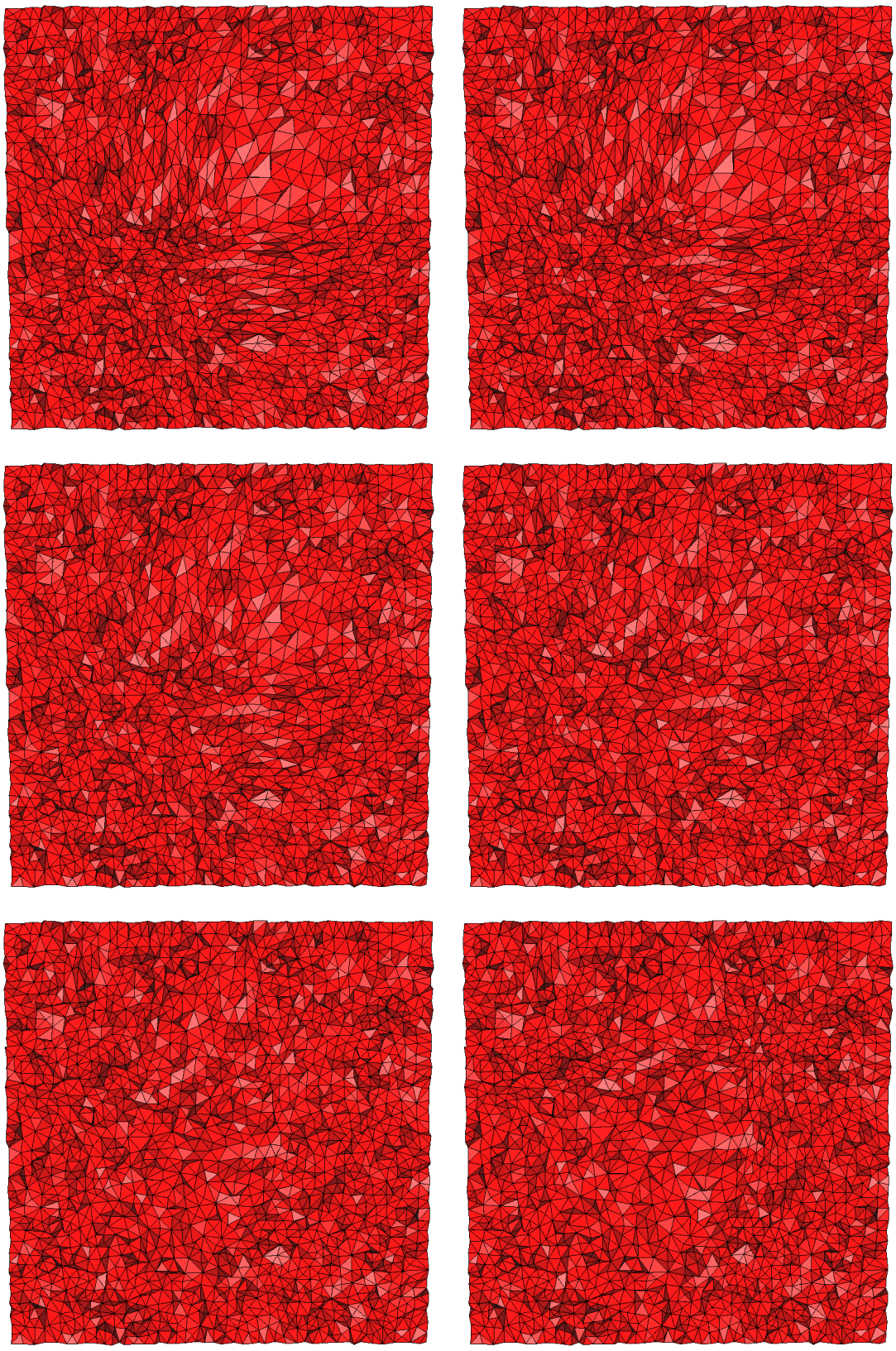


Figure 3.20: Snapshots of the deformation of the 3D moving-mesh adapted anisotropic meshes obtained for sensor function u_4 and for displacement $\mathbf{d}_3$. The resulting deformed mesh at t^{k+1} is optimal for u^{k+1}.

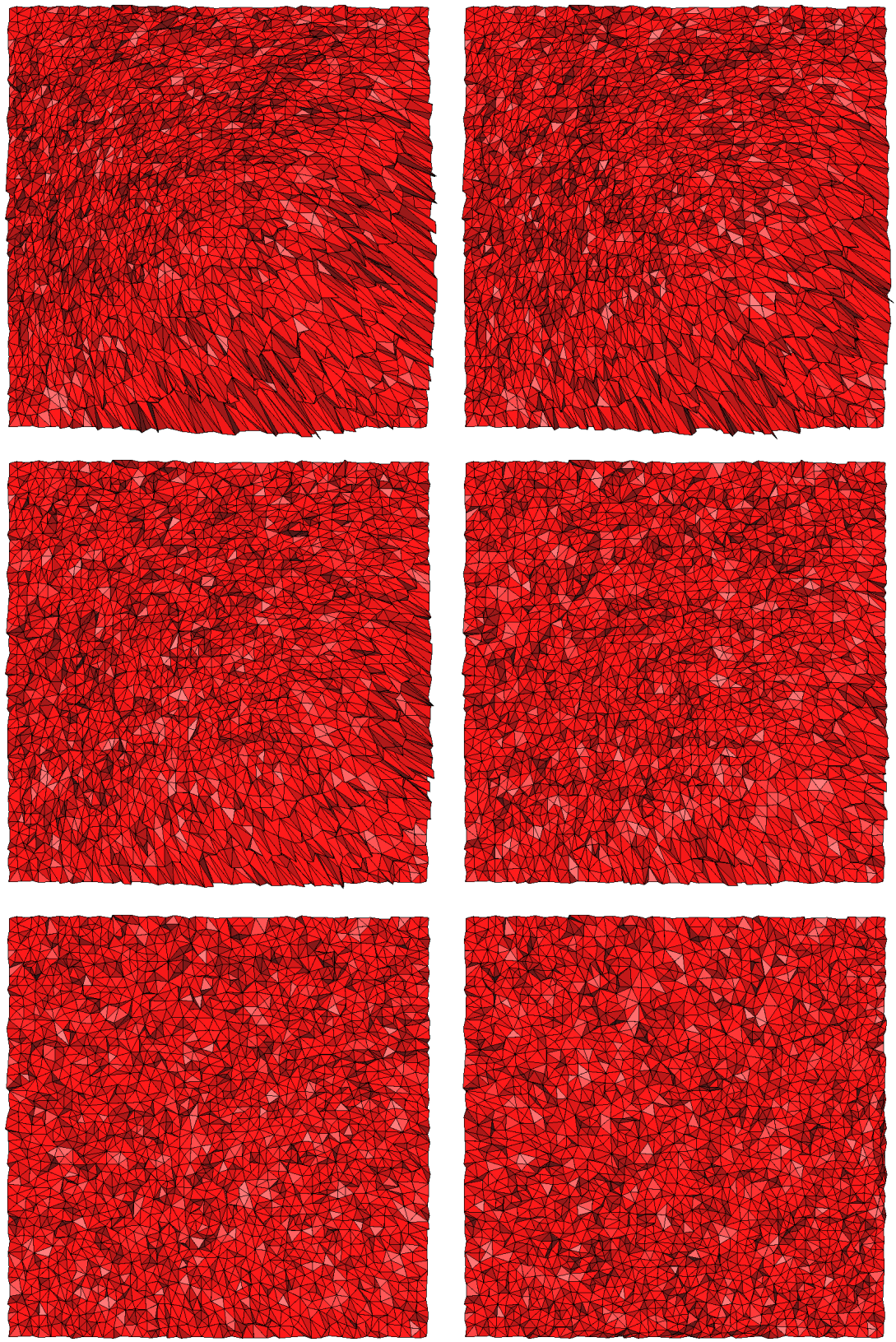


Figure 3.21: Snapshots of the deformation of the 3D moving-mesh adapted anisotropic meshes obtained for sensor function u_4 and for displacement $\mathbf{d}_4$. The resulting deformed mesh at t^{k+1} is optimal for u^{k+1}.

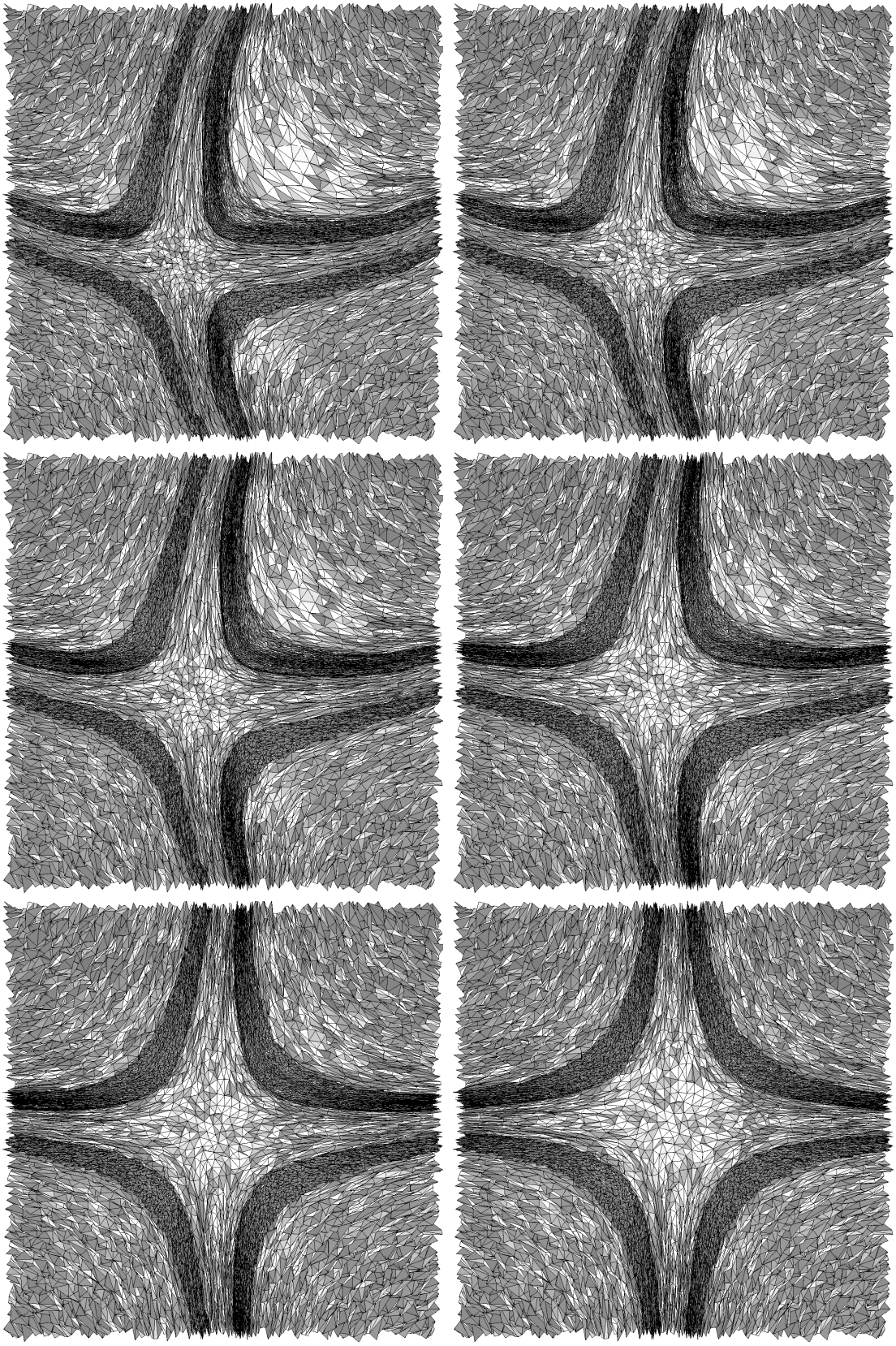


Figure 3.22: Snapshots of the deformation of the 3D moving-mesh adapted anisotropic meshes obtained for sensor function u_5 and for displacement $\mathbf{d}_3$. The resulting deformed mesh at t^{k+1} is optimal for $u^{k+1} = u_5$.

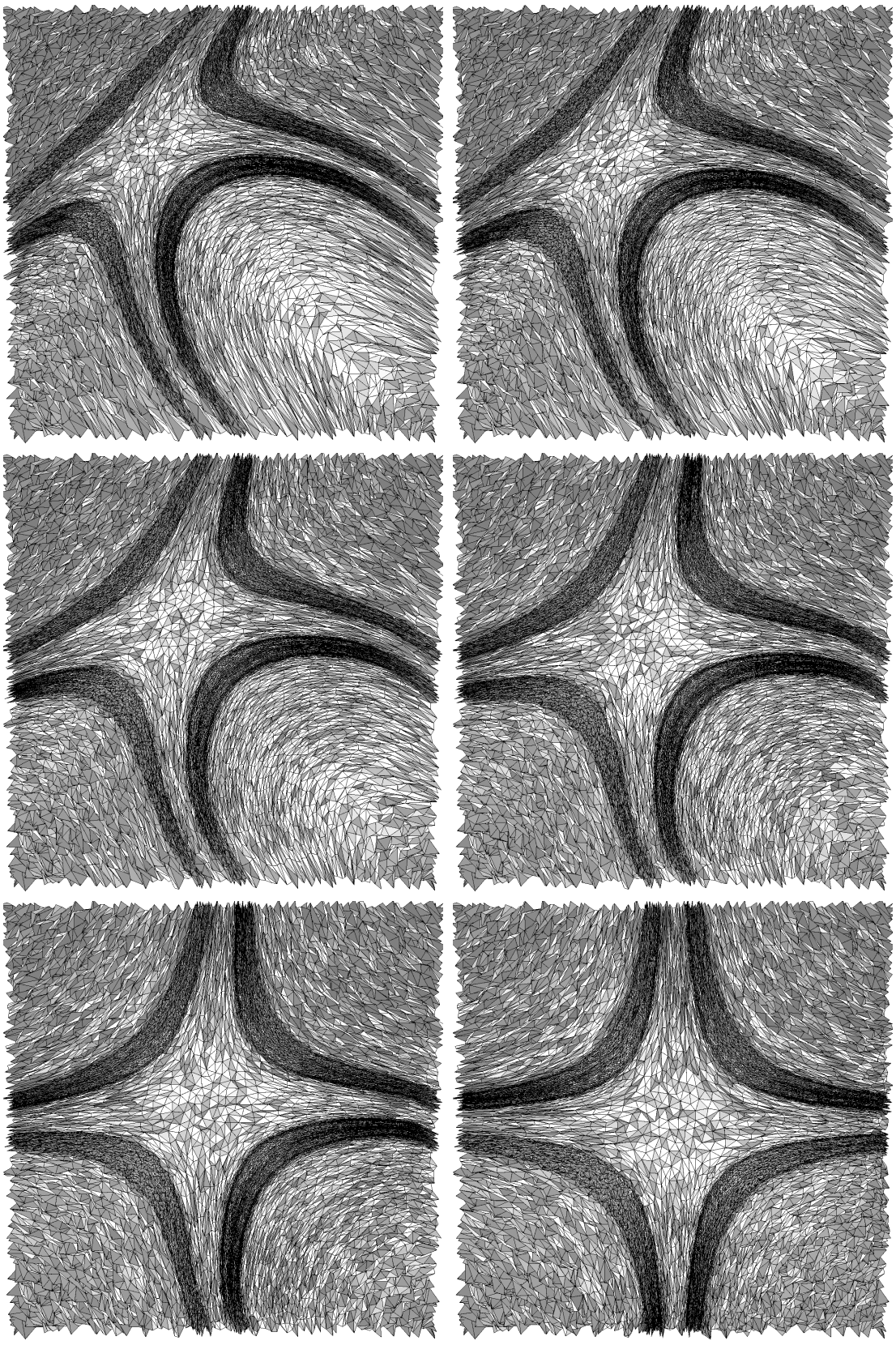


Figure 3.23: Snapshots of the deformation of the 3D moving-mesh adapted anisotropic meshes obtained for sensor function u_5 and for displacement $\mathbf{d}_4$. The resulting deformed mesh at t^{k+1} is optimal for $u^{k+1} = u_5$.

	$\mathcal{H}_{\mathbf{L}^1}(u_4)$		$\mathcal{H}_3^{k+1}(u_4)$ (with $\mathbf{d}_3$)		$\mathcal{H}_4^{k+1}(u_4)$ (with $\mathbf{d}_4$)	
Number of vertices	39459		73732		94213	
Number of tetrahedra	212509		406066		523538	
Average quality	1.316004		1.281694		1.255889	
Best quality	1.001474		1.000355		1.000518	
Worst quality	3.405595		3.860160		9.312094	
$1.00 < Q < 2.00$	211473	99.51%	404990	99.74%	521826	99.67%
$2.00 < Q < 3.00$	1029	0.48%	1060	0.26%	1584	0.30%
$3.00 < Q < 4.00$	7	0.00%	16	0.00%	102	0.02%
$4.00 < Q < 5.00$					17	0.00%
$5.00 < Q < 10.00$					9	0.00%

Figure 3.24: Quality statistics of the mesh $\mathcal{H}_{\mathbf{L}^1}(u_4)$ obtained by direct adaptation on sensor u_4 (left), of the images at t^{k+1} of the ALE-adapted meshes $\mathcal{H}_3^{k+1}(u_4)$ (middle) and $\mathcal{H}_4^{k+1}(u_4)$ (right) obtained for displacements $\mathbf{d}_3$ and $\mathbf{d}_4$, respectively.

	$\mathcal{H}_{\mathbf{L}^1}(u_5)$		$\mathcal{H}_3^{k+1}(u_5)$ (with $\mathbf{d}_3$)		$\mathcal{H}_4^{k+1}(u_5)$ (with $\mathbf{d}_4$)	
Number of vertices	144961		149160		279748	
Number of tetrahedra	765520		788653		1566807	
Average quality	1.222534		1.244649		1.759577	
Best quality	1.000899		1.000526		1.001712	
Worst quality	8.308210		448.0803		201.5625	
$1.00 < Q < 2.00$	763980	99.80 %	784041	99.42 %	1175914	75.05 %
$2.00 < Q < 3.00$	1256	0.16 %	4033	0.51 %	302909	19.33 %
$3.00 < Q < 4.00$	153	0.02 %	279	0.04 %	66066	4.22 %
$4.00 < Q < 5.00$	83	0.01 %	141	0.02 %	15699	1.00 %
$5.00 < Q < 10.00$	48	0.01 %	134	0.02 %	6039	0.39 %
$10.00 < Q < 100.00$	0	0.00 %	24	0.00 %	173	0.01 %
$100.00 < Q < 1000.00$	0	0.00 %	1	0.00 %	7	0.00 %

Figure 3.25: Quality statistics of the mesh $\mathcal{H}_{\mathbf{L}^1}(u_5)$ obtained by direct adaptation on sensor u_5 (left), of the images at t^{k+1} of the ALE-adapted meshes $\mathcal{H}_3^{k+1}(u_5)$ (middle) and $\mathcal{H}_4^{k+1}(u_5)$ (right) obtained for displacements $\mathbf{d}_3$ and $\mathbf{d}_4$, respectively.

Conclusion

In this first part, several novelties have been shown:

- the truncation algorithm for the steady multi-scale optimal metric,
- the extension of multi-scale metric based mesh adaptation to unsteady simulations, from a theoretical (error analysis) and a practical (global fixed-point algorithm) point of view,
- the computation of three-dimensional CFD flows on highly *anisotropic* three-dimensional meshes,
- the extension of multi-scale metric-based mesh adaptation to moving-mesh simulations with two and three dimensional illustrations.

This thesis has therefore permitted some interesting advances, which are really promising for industrial computations, especially those involving moving geometries.

Future work will certainly focus on the following aspects:

- validation of the fully space-time optimal metric on three-dimensional complex geometries,
- three-dimensional convergence studies for unsteady simulations both for fixed and moving mesh simulations.

Part II

Extension of anisotropic metric-based mesh adaptation to Arbitrary-Lagrangian-Eulerian simulations

Introduction

The growing expectations of the industrial world for simulations involving moving geometries have been given a boost to this research field for about ten years. Methods to handle this kind of problems are not new as articles dating back to the 60's can be found on the subject in two dimensions [Trulio 1961]. However, the lack of efficient computing resources as well as the lack of three-dimensional visualization tools have curbed the advances in this field.

In the 90's, work on these issues has resumed thanks to the arrival of computers with good performances and affordable even for small structures. As it is often the case in numerics, (static/dynamic) aerodynamics has taken the lead by proposing feasible real-life problems involving moving geometries, notably with aeroelastic computations [Tijdeman 1980, Batina 1990, Lesoinne 1993]. Moving mesh simulations are now used in many research fields:

Ballistics : missile release [Hassan 2007], missile deflection [Murman 2003],

Biomedical : aortic valves [Astorino 2009], cardiovascular system [Formaggia 2009],

Aeronautics : rotorcrafts [Dindar 2000], [Baum 1993a], store ejection [Baum 1995b], aircraft canopy trajectory [Baum 1997a], fuel-tank separation from an F-16 [Baum 1997b]

Blast studies : on a battle-field tank [Baum 1991], inside a Boeing-747 [Baum 1993b], in the World Trade Center [Baum 1995a], on a truck [Baum 1996],

Turbo-machinery : high-pressure turbine [Shyam 2010], mixed-flow pump [Zhu 1998], stirred reactors [Bakker 1997], simple turbine [Saksono 2007],

Transports : TGV tunnel-entry [Mestreau 1993], ship propeller [Compere 2010], 2D airbag deployment and balloon inflation [Saksono 2007], and even insect flights [Pivkin 2005].

Two main alternatives exist to handle moving geometries:

Body-fitted methods : The computational mesh respects all the boundaries of the geometries and follows bodies in their movement. There are two alternatives:

Chimera approach [Esnault 1985]: In this approach, each moving geometry is associated with a dedicated sub-grid, and these sub-grids can overlap each other. Each sub-grid moves in a rigid manner to follow its associated moving geometry. Interpolation techniques enable to compute the solution in overlapping regions.
Pros: flexible, only rigid mesh movement (no mesh deformation and thus no quality degradation),
Contras: not automatic, solution interpolation spoils accuracy and CPU time, cannot handle all cases.

Single-mesh methods : A single mesh is used which follows all the geometries evolving in the computational domain. As opposed to Chimera approaches, the mesh undergoes deformations and mesh quality generally degrades. To preserve mesh quality either *local* or *global* re-meshing steps are steadily performed.
Pros: accuracy, automaticity,
Cons: require efficient mesh movement techniques, frequent re-meshings required thus CPU time consuming.

Embedded/immersed methods [Löhner 2008]: An alternative is to use grids that are not body-conforming, and simply "embed" the objects in them. Various methods such as level-set methods enable to track the time evolution of the objects boundaries. Techniques of this kind are also known as immersed, fictitious domain, mortar element or Cartesian methods. The treatment of points and elements in the vicinity of the embedded triangulations or bodies is modified in such a way that the required kinetic or kinematic boundary conditions are properly imposed, for example using Lagrange multipliers.
Pros: automaticity, handle very complex geometries, no mesh movement issues,
Cons: less accuracy, boundary conditions, boundary layers, can be CPU time consuming for moving surface and Fluid-Structure Interaction computations.

Embedded methods enable to get rid of moving mesh constraints but at the price of a lower accuracy, especially for compressible flow simulations, see [Löhner 2007]. Therefore, a body-fitted methodology has been retained in this Thesis.

On the solver side, a common framework for such computations is the Arbitrary-Lagrangian-Eulerian (ALE) framework. It enables to rewrite the physical equations at stake taking into account the displacement of the mesh, which follows moving geometries. This is also the one adopted here.

This second part is built as follows.
First, moving mesh issues are addressed, and notably, it is demonstrated that moving three-dimensional complex geometries with large displacements is feasible using only vertex displacements and connectivity changes. This is new and presents several advantages over usual techniques for which the number of vertices varies in time. Notably, it facilitates the handling of data structures on the solver part, thus favorably impacting CPU time. It also answers the scarcely addressed question of spoiling interpolation errors in moving mesh simulations. Eventually, it is consistent with the moving mesh adaptation framework developed in Chapter 3, which requires that the complexity of the mesh remains constant between two adaptations.
The last chapter deals with numerical resolution issues when dynamic meshes are considered. Specific problematics linked to ALE simulations are detailed, notably DGCL time integration schemes. One of the novelties of this chapter lies in the description of a new methodology extending well-known ALE schemes to changing-connectivity meshes.

Chapter 4

Moving mesh strategies

Contents

4.1 Inner vertices movement prescription 145
4.1.1 Internal movement equation 145
4.1.2 Elasticity-dedicated mesh strategy 146
4.1.3 Local material properties 147
4.2 Mesh optimization 148
4.2.1 Laplacian smoothing 149
4.2.2 Edge/face swapping 149
4.3 Moving mesh time steps 151
4.3.1 Elasticity time step 151
4.3.2 Geometric time step 152
4.4 Numerical illustrations 152
4.4.1 Two dimensional tests cases 152
4.4.2 Three dimensional tests cases 155

In our case of body-fitted simulations, moving mesh issues represent one of the main challenges for simulations involving large displacements of the geometries, especially when non-uniform or adapted meshes are used. Indeed, the moving mesh algorithm must fulfill the following requirements:

- it must be very efficient as it is called at each solver time step,
- it must preserve the validity of the mesh (*i.e* it should not create reversed elements). Remeshing must indeed remain occasional because this operation, if constantly repeated, becomes costly and spoils the solution accuracy due to the interpolation stage it requires. It might even happen that generating a new mesh is not possible, especially in three dimensions,
- it must preserve elements quality to maintain the solution accuracy and acceptable time steps,
- the moving algorithm must handle small mesh size regions, shears and large movements,
- the movement must preserve the (anisotropic) adaptation, if any.

So far, for body-fitted simulations using a single mesh, two different methods are usually adopted to handle large displacements moving mesh simulations.
The first one consists in moving the mesh as much as possible keeping the topology fixed. Mesh is moved and the equations are solved in a fully ALE manner until the quality of the mesh becomes too bad. The domain is then globally re-meshed with the current geometry configuration, data structures are rebuilt, the current solution is interpolated on the newly generated mesh and the ALE computation resumes. One asset of this method is that meshing and solver aspects are entirely decoupled. Another advantage is that, between two re-meshings, the computation is "fully ALE", *i.e.* free from any interpolation error. However, the number of global re-meshings can become very large, especially when elements undergo shearing, which is unfortunately often the case in real life simulations. For example, in [Saksono 2007], one of the turbo-machinery simulations is done by globally re-meshing at *every* solver iteration![1] These very frequent re-meshings can lead to prohibitive CPU times and can even sometimes fail in three-dimensions, during the tricky boundary recovery phase performed by the mesh generator. The last drawback - but not the least - is that pushing mesh movement at fixed-topology to its limit results in very badly shaped elements, which can negatively impact the accuracy of the solution and also slows down the computation as the solver time step is controlled by the mesh minimal altitude through a CFL condition, see Section 2.1.2. For instance, this strategy is the one retained in [Murman 2003] on structured hexahedral meshes and in [Saksono 2007] on two-dimensional simplicial meshes.
The second approach aims at maintaining the best possible mesh quality while moving using several local re-meshing operations such as vertex addition, vertex collapsing, connectivity change and vertex displacements. This strategy enables to maintain an acceptable mesh quality and avoids the data structures frequent global reconstructions needed with the previous method. However, it induces a great number of interpolation steps. This is even worse if the solution interpolation is performed on the fly after each re-meshing operation. Besides, even if global data structures reconstruction is avoided, local re-meshing using various meshing operations requires fully dynamical data structures inside the solver, which can be costly to update and which can generate memory "holes" and cache-misses. Illustrations of this method can be found in [Compere 2010, Bruchon 2009, Dobrzynski 2008].

In this thesis, we have adopted a completely new philosophy for moving mesh ALE simulations as compared to existing methodologies. Our opinion is that **moving three-dimensional complex geometries with large displacements is possible using only vertex displacements and connectivity changes**. Vertex addition or collapsing is not mandatory and can remain occasional. For us, this perspective seems very promising for two main reasons:

CPU time: Limiting the number of authorized meshing operations comes to limit the number of interactions between meshing and numerical aspects. Notably, it enables to save a considerable amount of CPU time, on the meshing side of course, but also on the solver side as it limits changes in data structures,

[1] "The severity of the mesh deformation at the tip of the blades and at the corner of the base of the blade allows only a single ALE step after each re-meshing step".

Accuracy: It gives new perspectives to get rid of most of the spoiling interpolation errors due to excessive re-meshing. Indeed, limiting the number of meshing operations and keeping the same number of vertices during the simulation would allow to stay "fully ALE" throughout the simulation, provided that an ALE formulation is found to handle connectivity changes, see Section 5.4,

Moving mesh adaptation: The extension of the fixed-point algorithm to moving mesh simulations (see Section 3.3), which guarantees the control of the global interpolation error on each sub-interval in $\mathbf{L}^\infty - \mathbf{L}^p$ norm, is valid only if the number of vertices - the complexity of the mesh - remains constant between two mesh adaptations. Therefore, and at least between two re-meshings in the fixed-point algorithm, the number of vertices must remain constant, which precludes the use of vertices addition/collapse while moving.

This conviction has led us to focus on moving mesh techniques to numerically demonstrate the feasibility of this approach. These techniques are detailed in this chapter. The efficiency and the gain in terms of CPU time are *clearly* demonstrated on various three dimensional geometry movements.

4.1 Inner vertices movement prescription

4.1.1 Internal movement equation

In moving domain simulations, the whole mesh must move in order to follow the geometry movement while keeping a valid mesh (*i.e* mesh movement should not create reversed elements). The problem is the following: knowing the displacement of the vertices located on the moving boundaries, find a displacement of the inner vertices respecting the above criterium.
Two alternatives are generally considered: the spring-analogy methods [Batina 1990] and the elasticity methods [Lynch 1980, Baker 1999]. As stated in [Yang 2007], and despite various attempts to improve them (truss analogy [Farhat 1998], torsional spring methods, ...), spring-analogy techniques have shown less robust than elasticity-based methods, and also tend to deteriorate the quality of the mesh at a faster rate especially when considering large-displacements. Besides, elasticity-based techniques enable vertices to go round moving bodies instead of bumping into them, thus playing in favor of quality.

Consequently, the second approach has been retained and the inner vertices movement is obtained by solving an *elasticity-like equation* with a $\mathbf{P}^1$ Finite Element method, as suggested in [Lynch 1980]:

$$\operatorname{div}(\mathcal{S}(\mathcal{E})) = 0\,, \quad \text{with } \mathcal{E} = \frac{\nabla \mathbf{d} + \nabla \mathbf{d}^T}{2}\,, \tag{4.1}$$

where $\mathcal{S}$ and $\mathcal{E}$ are respectively the constraint and the deformation tensors and $\mathbf{d}^{els} = (d_1, d_2, d_3)^T$ is the Lagrangian elasticity displacement field. The constraint tensor follows the linear elasticity behavior law, where ν is the Poisson ratio, E the Young modulus of the material and λ, μ are the Lamé coefficients:

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

Dirichlet boundary conditions are used and the displacement of vertices located on the domain boundary is strongly enforced in the linear system. Note that the vertices located on the domain boundaries cannot slip on these boundary surfaces. Indeed, moving mesh vertices on the boundary is a hard task: boundaries are generally curved and moving vertices on discrete boundaries requires either the knowledge of the CAD patches used to design the geometry or the ability to rebuilt an accurate continuous surface model of the boundary from its discretization. In any case, this represents a lot of work. We therefore limit ourselves to mesh displacements satisfying a Dirichlet condition on the domain boundaries (non-slipping condition).

In our context, ν is typically chosen of the order of 0.48. This leads to a very soft, nearly incompressible material. Actually, this value for ν corresponds to a nearly "ill-posed" problem. Note that the closer ν is to 0.5, the harder it is to converge the Finite Element linear system, especially in three-dimensions. $\nu = 0.48$ appears as a good trade-off between material softness and the preservation of the iterative linear system resolution algorithm efficiency.
The assembly of the Finite Element matrix, along with other technical aspects, is detailed in Appendix B. The Finite Element system is solved by a GMRES algorithm coupled with an ILU pre-conditioner. This resolution strategy is clearly not optimal as it does not take into account the symmetry of the linear elasticity matrix, but improvements in the resolution is currently underway (new resolution algorithm and p-thread parallelization of this algorithm).

4.1.2 Elasticity-dedicated mesh strategy

The resolution is performed on an elasticity-dedicated mesh, which is generally chosen uniform and much coarser than the one used to solve the Euler equations. The displacement computed on the elasticity-dedicated mesh is then interpolated on the finer adapted Euler mesh using a $\mathbf{P}^2$-Lagrangian scheme (see Section 1.2.3.3), which seems sufficiently accurate considering the intrinsic smoothness of the solution of the above elasticity problem. This strategy is illustrated in Figure 4.1. This elasticity-dedicated mesh strategy enables a significant reduction of CPU time:

- It reduces the size of the elasticity linear system to be solved, thus reducing CPU time and storage (the extra-storage of the elasticity mesh and of its resolution is negligible compared to the reduction of the elasticity matrix size).
- It avoids restructuring the Finite Element elasticity matrix each time a connectivity change is performed in the mesh used for the fluid resolution.
- As already said, we intend to couple moving mesh simulations and anisotropic mesh adaptation. But it would be very hard to solve the FE elasticity system on an anisotropic mesh adapted on the compressible flow due to the creation of an artificial extra-stiffness hindering the convergence of the linear system. With this strategy, the property of the elasticity mesh can be chosen independently of the physics of the flow. Actually, this strategy is mandatory if highly-anisotropic metric-based mesh adaptation is introduced in these moving mesh simulations.

Even if the elasticity mesh must be moved and sometimes optimized along with the computational mesh, the additional cost is negligible as compared to the gain in terms of CPU for the

h!

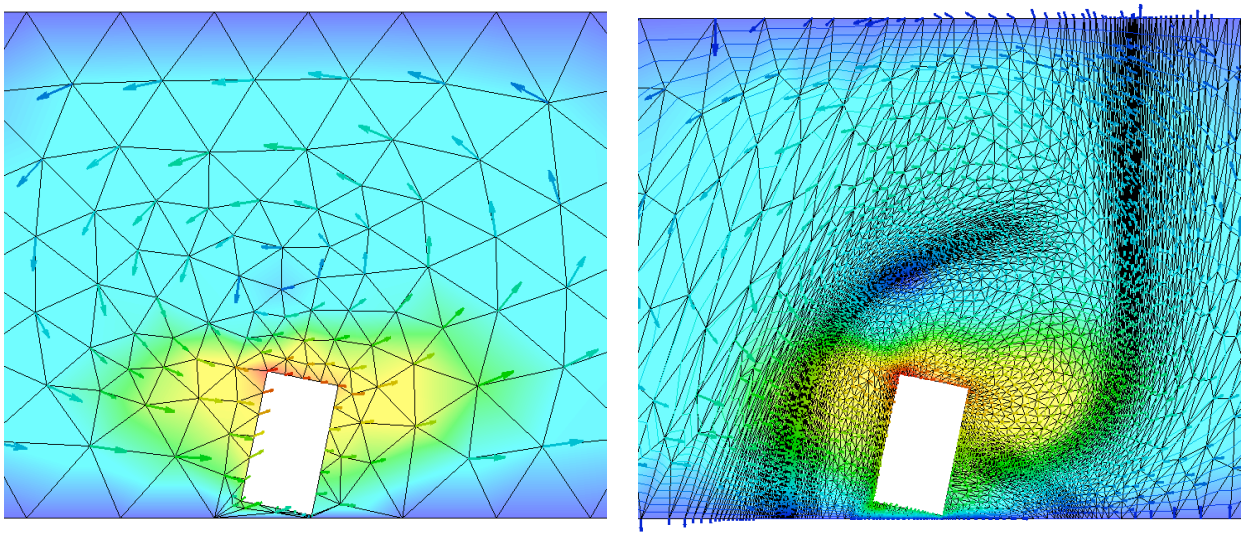


Figure 4.1: Left, solution of the elasticity system on the elasticity dedicated, coarse mesh. Right, interpolation of the solution from the back, coarse mesh to the computational, fine one. Note that the elasticity mesh is uniform, which prevents the resolution of the linear elasticity system on the highly anisotropic computational mesh.

linear elasticity resolution.

Indeed, on the one hand, the size of the elasticity linear system is reduced as the elasticity mesh is generally coarser. On the other hand, the efficiency of the GMRES algorithm is known to be considerably damaged when the regularity of the mesh is insufficient.

4.1.3 Local material properties

Another advantage of elasticity-like methods is that they offer the opportunity to adapt the local material properties of the mesh, especially its softness, according to the distortion and efforts born by each element and to the quality of the element. In our case, for efficiency purpose, only basic tools are used such as rigidifying certain regions, which actually reduces the general problem of moving arbitrary geometries in a mesh to the much simpler problem of moving simpler objects (spheres, boxes, convex objects) encompassing them.

Jacobian-based stiffening. Following [Stein 2003], the way the Jacobian of the transformation from the reference element to the current element is accounted for in the Finite Element matrix assembly is modified. Classically, the $\mathbf{P}^1$ Finite Element formulation of the linear elasticity matrix leads to the evaluation of quantities of the form:

$$\int_K s \frac{\partial \varphi_J}{\partial x_k} \frac{\partial \varphi_I}{\partial x_l} \, \mathrm{d}x = |K| \, s \frac{\partial \varphi_J}{\partial x_k} \frac{\partial \varphi_I}{\partial y_l} = |K| \, s \frac{1}{6|K|} (\overline{\boldsymbol{\eta}}_J)_k \frac{1}{6|K|} (\overline{\boldsymbol{\eta}}_I)_l = \frac{s}{36|K|} (\overline{\boldsymbol{\eta}}_J)_k (\overline{\boldsymbol{\eta}}_I)_l \,,$$

where s is either λ, μ or $\lambda + 2\mu$ and $\overline{\boldsymbol{\eta}}_I = ((\overline{\boldsymbol{\eta}}_I)_x, (\overline{\boldsymbol{\eta}}_I)_y, (\overline{\boldsymbol{\eta}}_I)_z)$ is the inward non-normalized normal opposite to vertex I in tetrahedron K. In [Stein 2003], the above quantity is replaced by:

$$\int_K s \frac{\partial \varphi_J}{\partial x_l} \frac{\partial \varphi_I}{\partial x_n} \left(\frac{|\hat{K}|}{|K|} \right)^{\chi} \mathrm{d}x \,,$$

where $\chi > 0$ is the stiffening power and $\hat{K}$ is the reference element. With a $\mathbf{P}^1$ approximation, this quantity equals:

$$|K|\, s \frac{1}{6|K|} (\overline{\boldsymbol{\eta}}_J)_k \frac{1}{6|K|} (\overline{\boldsymbol{\eta}}_I)_l \left(\frac{|\hat{K}|}{|K|} \right)^{\chi} = s \left(\frac{|\hat{K}|}{|K|} \right)^{\chi} \frac{1}{36|K|} (\overline{\boldsymbol{\eta}}_J)_k (\overline{\boldsymbol{\eta}}_I)_l \,.$$

This technique therefore comes to locally multiply λ and μ by a factor proportional to $|K|^{-\chi}$. With $\chi = 0$, the method reduces back to an elasticity model with no Jacobian-based stiffening. χ determines the degree by which the smaller elements are rendered stiffer than the larger ones.

Finally, in some situations - geometries involving sharp corners, mesh movement with shearing - we have found it useful to rigidify a certain number of elements layers around moving objects. The number of layers to be rigidified for each object is provided by user, and elements belonging to these layers are automatically found using a topological algorithm. Elements belonging to a rigid layer are moved in a completely rigid manner using the angular and the translation displacement of the object the considered boundary layer is associated with.

4.2 Mesh optimization

Mesh quality tends to decrease while the mesh is moving. Therefore, regular optimization phases must be performed to maintain a quality compatible with the solution desired accuracy and maintain a solver time step which is reasonably large[2]. The quality of the mesh in three dimensions is measured in terms of element shape by the following quality function:

$$Q(K) = \frac{\sqrt{3}}{216} \frac{\left(\sum_{i=0}^{5} ||\mathbf{e}_i||^2 \right)^{\frac{3}{2}}}{|K|} \quad \in [1, +\infty] \,.$$

$Q(K) = 1$ corresponds to a perfectly regular element while a high value of $Q(K)$ indicates a nearly degenerated element. Only Laplacian smoothing and swaps are used to optimize the mesh, which means that the number of vertices remains constant throughout the simulation. These optimizations are performed very often to preserve the best possible quality of the mesh and avoid global re-meshings. A quality threshold Q_{target} is prescribed by user and only tetrahedra with a quality exceeding this threshold are considered during the optimization phase. Tetrahedra are sorted by decreasing quality, which enables to deal in priority with the worst of them.

[2]The solver time step is linked to the mesh smallest altitude through a CFL type condition.

4.2.1 Laplacian smoothing

Laplacian smoothing consists in relocating each vertex inside its ball. For each tetrahedron K_j of the ball of P_i, the face of K_j opposite to vertex P_i is noted F_j. An optimal position P_j^{opt} for P_i is proposed by each face F_j of the ball:

$$P_j^{opt} = G_j + \sqrt{\frac{2}{3}} L_j^{moy} \frac{\mathbf{n_j}}{||\mathbf{n_j}||}, \quad \text{with } L_j^{moy} = \frac{1}{3}\Big[\ell_0(F_j) + \ell_1(F_j) + \ell_2(F_j)\Big],$$

where G_j is the gravity center of face F_j and $\mathbf{n}_j$ is the inward normal to face F_j relative to element K_j, see Figure 4.2. The final optimal position is computed as a weighted average of

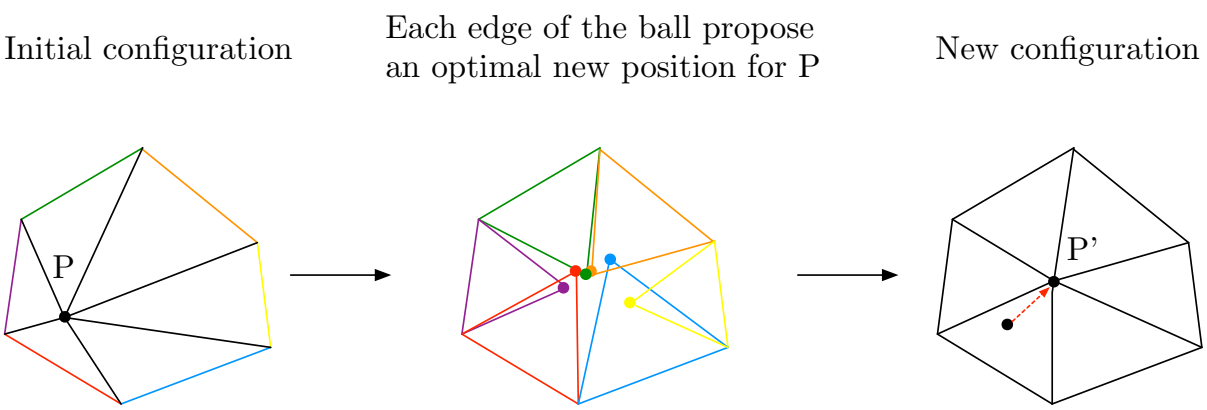


Figure 4.2: Laplacian smoothing in two dimensions. Each element of the ball of considered vertex P_i suggests an optimal position for P_i. The resulting new optimal position for P_i is computed as a weighted average of all these proposed locations.

these optimal positions, the weight coefficient associated with F_j depending on the quality of K_j:

$$P_i^{opt} = \frac{\sum\limits_{K_j \in \text{Ball}(P_i)} \max\left(Q(K_j),\, Q_{max}\right)\, P_j^{opt}}{\sum\limits_{K_j \in \text{Ball}(P_i)} \max\left(Q(K_j),\, Q_{max}\right)},$$

and Q_{max} is a parameter to be defined. In this way, an element of the ball is all the more influent as 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. Otherwise, successive relaxed positions $P_i^{new} = P_i + \alpha \overrightarrow{P_i P_i^{opt}}$, with progressively decreasing values of α are tested.

4.2.2 Edge/face swapping

In two dimensions, edge swapping - or swap - is a rather simple topological operation which consists in flipping an edge shared by two triangles, see Figure 4.3 (top left). In two dimensions, this operation changes neither the number of triangles nor the number of edges of the mesh, guaranteeing a constant number of entities throughout the computation. The generalization of this operation in three dimensions is a little bit more complicated. Let **e** be an internal edge.

Its shell (see Conventions, page 3 for the definition) is represented in black in Figure 4.3 (right). From the shell, a non-planar *pseudo-polygon* - shown in blue on Figure 4.3 - can be defined.

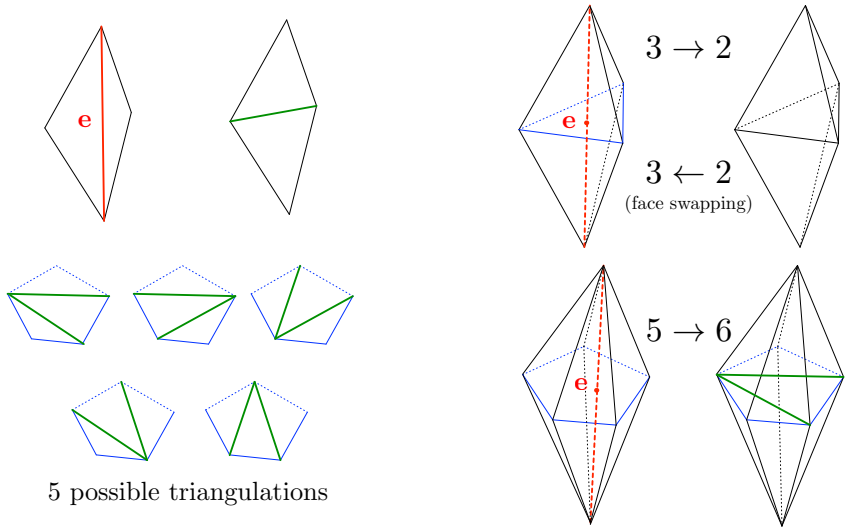


Figure 4.3: Top left, the swap operation in two dimensions. Top right, edge swap of type $3 \to 2$ and face swap $2 \to 3$. Bottom left, the five possible triangulations of the pseudo-polygon for a shell having five elements. Bottom right, an example of $5 \to 6$ edge swap. For all these figures, shells are in black, old edges are in red, new edges in green and the pseudo-polygon is in blue.

Performing a three-dimensional swap of edge **e** comes to define a triangulation of this pseudo-polygon. The number of possible *topological* triangulations depends on the cardinal of the shell n_{shell} and is given by a combinatorial argument:

$$n_{combi} = \frac{(2n_{shell} - 4)!}{(n_{shell} - 1)!\,(n_{shell} - 2)!}\,.$$

The maximal number of triangulations in function of n_{shell} is given in Table 4.1. The five

n_{shell}	3	4	5	6	7	8	9	10	11	12	13
n_{combi}	1	2	5	14	42	132	429	1430	4862	16796	58786
n_{tri}	1	4	10	20	35	56	84	120	165	220	...

Table 4.1: Number of topologically different triangulations n_{combi} that are valid as a function of the number of vertices n_{shell} in the pseudo-polygon related to one edge. Number of different triangles n_{tri} in each possible triangulation. The validity of the triangulations is not considered (only the topological aspect is taken into account)

possible triangulations for pseudo-polygon with $n_{shell} = 5$ vertices are shown on Figure 4.3, bottom left. Swapping edge **e** consists in suppressing this edge from the mesh and in creating all the tetrahedra having one of the extremities of **e** as vertex and one triangle of the pseudo-polygon triangulation as face, see [Frey 2008]. The different kinds of swaps are designated by $n^{old}_{shell} \to n^{new}$, n^{new} being the number of new tetrahedra created by the swap. Figure 4.3

depicts an edge swap of type $3 \rightarrow 2$ (top right) and one of type $5 \rightarrow 6$ (bottom right). Face swapping consists in suppressing face F of a tetrahedron K and to create an edge linking the two vertices opposite to F in K and in the neighboring tetrahedra to K through F, respectively. Face swapping can be viewed as a reversed $3 \rightarrow 2$ swap and results in the creation of three new tetrahedra, see Figure 4.3, top right.

Three-dimensional swaps are coded using swap and rejection tables, which avoids a considerable amount of useless tests and allows to test most plausible swaps first. Face swapping $2 \rightarrow 3$ has been implemented, along with edge swaps of type $3 \rightarrow 2$, $4 \rightarrow 4$, $5 \rightarrow 6$, $6 \rightarrow 8$ and $7 \rightarrow 10$. Swaps are first simulated and, for each edge/face to be swapped, only the best swap is retained. Except for the $4 \rightarrow 4$, all the other three dimensional swaps change the number of tetrahedra of the mesh: this number decreases for the $3 \rightarrow 2$ and increases for all the other combinations. The number of edges varies the same way during the simulation.New tetrahedra/edges are added at the end of the table and old ones are just marked as deleted. A packing procedure is regularly applied on these tables to re-fill holes created by "phantom" tetrahedra/edges and restore the contiguity of elements in memory.
Another key to perform efficient swaps in the moving mesh context is to authorize a slight quality degradation. Of course, when mesh adaptation is used, the optimization routines must be adapted because all the geometric quantities involved must be computed in the metric field, see Section 1.2.1.1.

4.3 Moving mesh time steps

4.3.1 Elasticity time step

The elasticity system is not solved at each solver iteration to reduce CPU time. It is rather solved at specific times, the trajectory of each inner vertex being considered as piecewise linear along its associated elasticity displacement vector between two elasticity resolutions. Note that vertices located on moving objects are always moved exactly, *i.e.* according to the FSI solution or the analytical prescription of the angular and translation displacement of the moving object they are associated with at the current time.

Several criteria can be used to determine when the elasticity problem must be solved.
The simplest choice is to solve the elasticity system only every $m \geq 1$ iterations but this can be problematic if a sharp change occurs in the trajectory of some inner vertex. Besides, Yang and Mavriplis [Yang 2005] advise against this kind of artifices, arguing that a loss of temporal accuracy can stem from the lack of regularity of the vertices numerical trajectories: "For problems where the mesh motion is governed by a set of partial differential equations, achieving high temporal accuracy requires a smooth $\mathbf{x}(t)$ variation, which implies the specification of a smooth boundary motion in time, and convergence of the mesh motion equations at **each time step** to a suitable tolerance, which can be assumed to be of the same order as the convergence tolerance used for the implicit flow solver. The common practice of only partially converging the mesh motion equation in order to reduce computational costs, under the assumption that a valid mesh with positive cell volumes is all that is required at each time step may thus result in the loss of higher-order temporal accuracy".

A clever way to proceed would consist in adapting the elasticity time step according to the smoothness of the vertices trajectories, the most winding trajectory driving the adaptation: if all the points have a linear trajectory, the elasticity time step can be taken very large whereas the elasticity must be solved at each iteration when one of the trajectories exhibits abrupt direction changes. With this technique, the number of elasticity resolutions could be reduced while maintaining the regularity of the trajectories. This adaptive elasticity time step has not yet been implemented but this will be done very soon.
Eventually, the interval between two elasticity resolutions can be reduced when bad qualities are anticipated.

4.3.2 Geometric time step

A good restriction to be imposed to the mesh movement is that vertices cannot cross too many elements on a single move. Therefore, a geometric parameter CFL^{geom} is introduced to control the number of stages used to perform the mesh displacement between t and $t + \mathrm{d}t^{solver}$. If CFL^{geom} is greater than one, vertices are allowed to cross more than an element of the backward mesh in a single move. The smallest this number, the highest the number of moving steps performed to reach final destination. As each moving step is preceded by an optimization procedure, cutting a large displacement in several smaller displacements by reducing the geometric CFL enables to ease mesh movement.
Practically, if the time interval between two elasticity resolutions is $\mathrm{d}t^{els}$, and if, for this elasticity time step, a displacement $\mathbf{d}^{els}$ has been computed at vertex, an artificial elasticity speed can be computed for vertex P_i:

$$v_i^{els} = \frac{||\mathbf{d}_i^{els}||}{\mathrm{d}t^{els}}\,.$$

Let $h_{min,i}$ denote the smallest altitude among all the elements belonging to the ball of P_i. The minimal number of stages necessary to perform mesh movement is given by:

$$N_{mov} = \max\left(1,\, \lceil CFL^{geom} \min_{i\in[\![1,\,N_v]\!]} \frac{h_{min,i}}{v_i^{els}} \rceil^{-1}\right)$$

and the maximal geometric time step is:

$$\mathrm{d}t^{geom} = \frac{\mathrm{d}t^{els}}{N_{mov}}\,.$$

Finally, the effective time step for the mesh movement is:

$$\mathrm{d}t^{mov} = \min\left(\mathrm{d}t^{geom},\, \mathrm{d}t^{solver}\right)\,.$$

4.4 Numerical illustrations

4.4.1 Two dimensional tests cases

4.4.1.1 Rotating blades

This academic example is very illustrative of the powerfulness of the swap operation as soon as the mesh undergoes shearing movements. A pump made of six blades rotates. Here, we are only

interested in the mesh deformation. The pump is put into a rigid circular area (in green) to ease mesh movement, as described in Section 4.1.3. Figure 4.4 shows the progressive deterioration of the mesh during the rotation movement in the case where swapping is not allowed. For this case, elements inside the green area are moved in a rigid manner and external elements are moved using the elasticity equation. After only a quarter of turn, elements located at the boundary between the two areas are highly stretched and the movement finally ends as the volume of these elements tends to 0. The repartition of the shearing on the several layers of elements by the elasticity is not sufficient to avoid re-meshing. On the contrary, Figure 4.5 shows the same test case when swapping is allowed. In this case, it is possible to impose a null displacement to external (pink) elements. None of the vertices actually moves. Only swaps are performed and there are done only in the layer at the junction between the static (pink) and the rigid (green) areas. This enables to keep the initial, presumably excellent, quality of the initial mesh throughout the simulation. An infinite number of turns can be performed without deteriorating at all the quality of the mesh. Note that this phenomenon had already been observed in [Zhu 1998].

4.4.1.2 Falling wedge

This test case is inspired by the simulation of the impact of a falling wedge on water surface, described in [Xu 1998, Oger 2006, Allain 2009]. It demonstrates in two dimensions that edge swapping allows large mesh displacements, without adding or suppressing vertices. Here again, we are only interested in mesh movement issues. The wedge is put inside a disk of rigid elements to avoid mesh deterioration near sharp angles. Figure 4.6 shows the mesh deformation while the wedge is falling. Only swaps and vertices displacements have been used, no vertex is added or suppressed. As can be seen on Figure 4.6, we are able to move the wedge throughout the computational domain without any global re-meshing. Moreover, we have been able to move until only one element layer remains between the disk containing the wedge and the bottom wall, which demonstrates the robustness of the swapping algorithm. Figure 4.7 is a close-up view of the mesh near the falling wedge. It illustrates how the swap enables to open the way to the falling geometry.

4.4.1.3 Shocks between moving disks

This test case illustrates the usefulness of the swap operation when moving objects are getting close to each other. Four disks are falling following piecewise linear "zigzag" trajectories. The directions of their trajectories change each time they collide with another of the falling disks or with side walls. Figure 4.8 shows the movement of the mesh for this test case. Again, only swaps and vertex displacements is permitted. No global re-meshing has been performed for this simulation. If swap is forbidden, 62 re-meshings are necessary to perform the complete simulation, which gives an idea of the gain that can be expected with edge swapping, in terms of both CPU time and interpolation errors. Moreover, edge swapping is crucial in contacts areas between disks, as it enables to handle a distance between geometry of one mesh edge.
Of course, the mesh quality is not as good as it is when global re-meshings are performed, mainly because with our strategy, vertex movements on the domain boundaries are forbidden, and the initial boundary discretization is not adapted anymore to the position of the geometries once

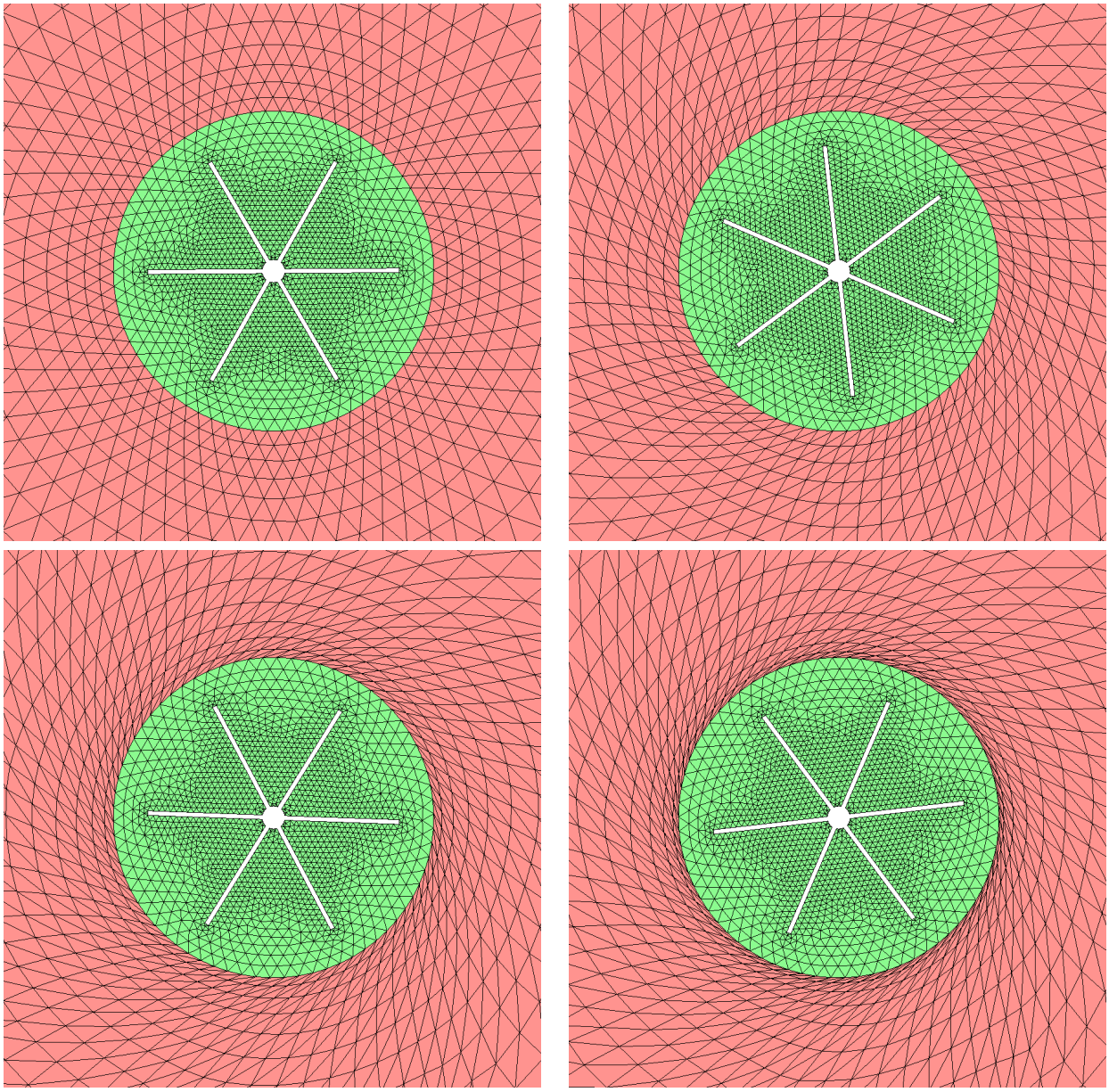


Figure 4.4: Mesh movement around rotating blades when swapping is not allowed. Mesh keeps on deteriorating during the movement due to the strong shearing undergone by elements located at the boundary between green (rigid) and pink areas. Mesh movement finally stops as the quality of the elements degrades to $+\infty$.

they have moved. However, the interest of this test case is to demonstrate the feasibility of using only edge swapping to handle contacts. Three or four re-meshings, by repositioning the vertices on the boundaries, would certainly be enough to maintain a good mesh quality.

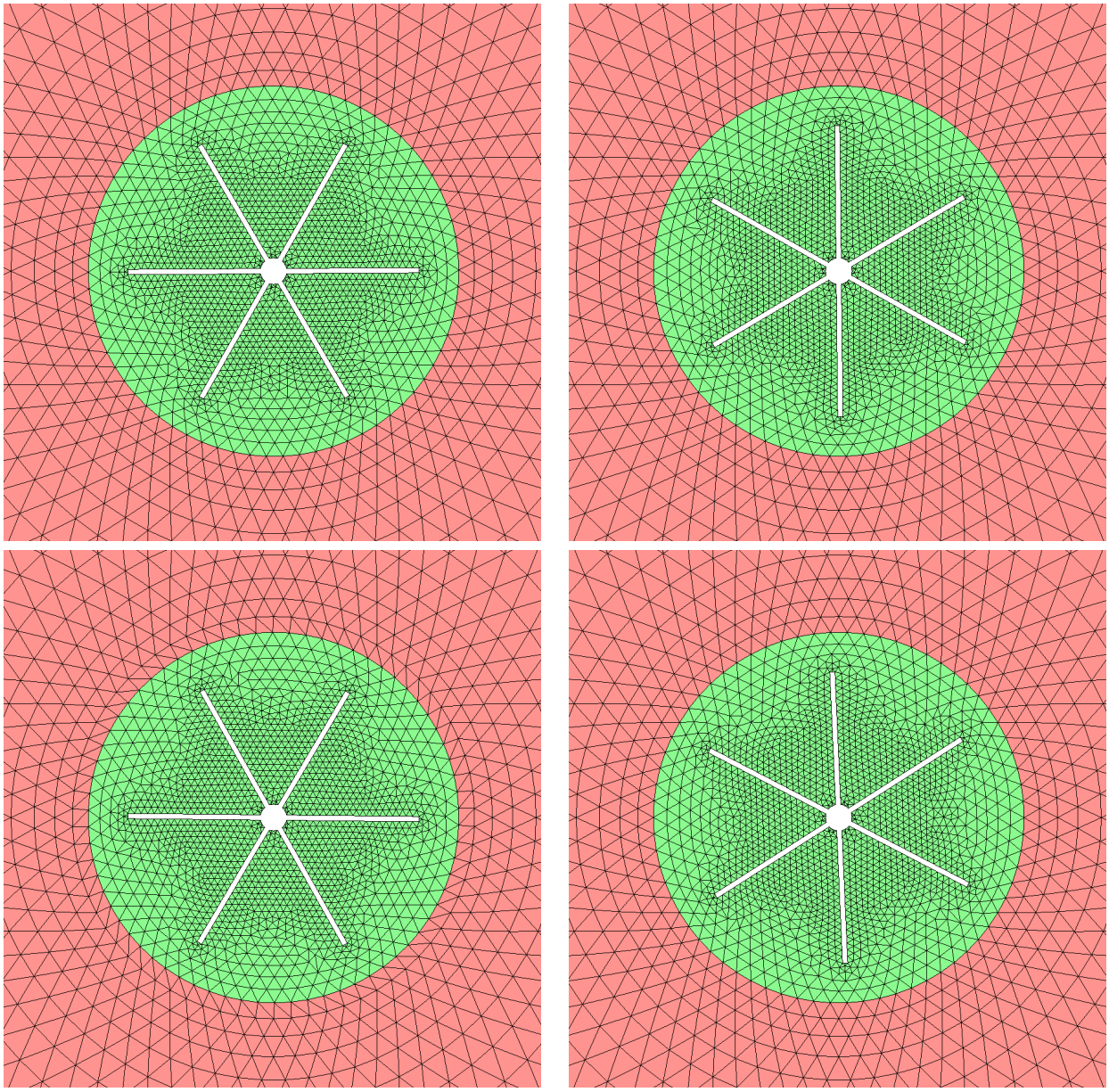


Figure 4.5: Mesh movement around a rotating blade when swap is allowed. No vertex is actually moving. Only swaps are performed in the layer at the junction between static (pink) and rigid (green) areas. An infinite number of turns can be performed while maintaining the excellent quality of the initial mesh.

4.4.2 Three dimensional tests cases

Several analytical examples on academic geometries have been addressed to demonstrate the feasibility of performing three-dimensional moving mesh simulation while keeping the number of vertices constant. This means that large displacements can be handled **without any re-meshing** using only vertices displacements and edges/faces swaps. Here, four examples of

Case	Average Q_{end}	$1 < Q_{end} < 2$	Worst Q_{end}	Worst Q_{all}	# of swaps	CPU
Cube	1.4452	98.37%	3.99	6.14	386 702	4m43s
Plate	1.4502	97.70%	5.19	5.60	269 533	2m29s
Cylinder	1.5053	93.71%	4.29	6.68	1 049 348	5m45s
Two cylinders	1.5000	95.58%	5.57	198.81	861 228	11m05

Table 4.2: Mesh statistics and CPU time for the four test cases.

three-dimensional moving geometries are detailed: a moving rotating cube, a moving rolling plate, a swirling cylinder and two cylinders interpenetrating. All these examples have been run in serial on a 64-bits MacPro with an IntelCore2 chipsets and a clock speed of 2.8 GHz with 32 Gb of RAM.

4.4.2.1 A moving three-dimensional rotating cube

We translate through the domain a rotating cube in a mesh made of 34 567 vertices and 188 847 tetrahedra. The cube movement is shown in Figure 4.9. Twenty-five elasticity systems have been solved during the movement and a total of 251 moving steps have been done. In Table 4.2, we observe that at the end of the move, the quality of the mesh is excellent with an average quality of 1.4452, 98.37% of the elements with a quality less than 2 while the worst quality is 3.99. The worst element created during the whole movement has a quality of 6.14. The quality is thus excellent throughout the simulation. A total of 386 702 swaps have been performed, which represents 1 540 swaps per moving steps. As regards CPU time, it took 4m43s to move the cube, 45% of the CPU time being spent in the elasticity resolution, which will soon be parallelized.

4.4.2.2 An moving three-dimensional rolling plate

In this second example, we translate through the domain a rolling plate in a mesh having 14 376 vertices and 76 049 tetrahedra. The geometry is anisotropic and exhibits sharp angles. Moreover, its rolling movement tends to create shearing inside the mesh. The plate movement is shown in Figures 4.10 and 4.11. Twenty-five elasticity systems have been solved during the movement and a total of 544 moving steps have been done. In Table 4.2, we note that at the end of the move, the quality of the mesh is excellent with an average quality of 1.4502, 97.70% of the elements with a quality less than 2 and a worst quality of 5.19. Again, the quality is excellent throughout the simulation as the worst element generated during the whole movement has a quality of 5.60. A total of 269 533 swaps have been performed, which represents 495 swaps per moving steps. As regards CPU time, it took 2m29s to move the plate, 35% of the CPU time being spent in the elasticity resolution.

4.4.2.3 Swirling cylinder

In this example, a cylinder is swirling inside a domain performing two turns. This displacement mainly involves shears inside the mesh. This test case is thus very appropriate to illustrate the efficiency of the topological swap operation. The initial mesh is made of 25 135 vertices

and 139 540 tetrahedra. The cylinder rotation is depicted in Figure 4.12. Thirty-six elasticity systems have been solved during the movement and a total of 299 moving steps have been done. In Table 4.2, we note that at the end of the move, the quality of the mesh is excellent with an average quality of 1.5053, 93.71% of the elements having a quality less than 2 and the worst quality being 4.29. The quality remains excellent throughout the simulation as the worst created element during the whole movement has a quality of 6.68. A total of 1 049 348 swaps have been performed which represents 3 509 swaps per moving steps. The number of swaps per move is quite large due to the shear movement impacting a large part of the domain. As regards CPU time, it took 5m45s to rotate the cylinder, 27% of the CPU time being spent in the elasticity resolution.

4.4.2.4 Interpenetrating cylinders

The last example is more challenging. Two cylinders are interpenetrating each other and, during this movement, there is only one layer of elements between both cylinders, *i.e.* the two cylinders boundaries are connected by an internal edge. The layer of elements separating the two cylinders hence undergoes strong shearing, see Figure 4.14. This movement is shown in Figure 4.13. The initial mesh is made of 34 567 vertices and 188 847 tetrahedra. One hundred elasticity systems have been solved during the movement and a total of 614 moving steps have been done. In Table 4.2, we note that at the end of the move the quality of the mesh is excellent with an average quality of 1.5, 95.58% of the elements with a quality less than 2 and a worst quality of 5.57. But, during the movement badly-shaped elements appear, the worst of them having a quality equal to 198.81. However, the optimization process is able to get rid of these elements to finally achieve an excellent quality. A total of 861 228 swaps have been performed, which represents 1 402 swaps per moving steps. As regards CPU time, it took 11m05s to perform the displacement, 57% of the CPU time being spent in the elasticity resolution.

Conclusion

In this chapter, the feasibility of moving various two- and three-dimensional objects with large displacements has been demonstrated numerically. Our strategy, with consists in limiting the number of meshing operations allowed, leads to very satisfying results that are definitely comparable to other strategies [Compere 2010, Dobrzynski 2008] in terms of mesh quality, and which seems more efficient in terms of CPU time. For example, in [Compere 2010], the authors precise the CPU times they observed on a propeller test case. The simulation lasts $T = 1\,s$ and the moving time step is $\mathrm{d}t^{mov} = 0.001T\,s$. The number of nodes is averagely $N_v = 55\,000$. For this number of nodes, the global procedure takes about $50\,s$ per time step, thus $50000\,s$ are needed to complete the whole simulation, which leads to approximately 139 hours (5.79 days) of CPU time for 1000 moves. These results must be compared with CPU time 11m05s obtained with $N_v = 34\,567$ nodes and 614 moving steps on the interpenetrating cylinders.

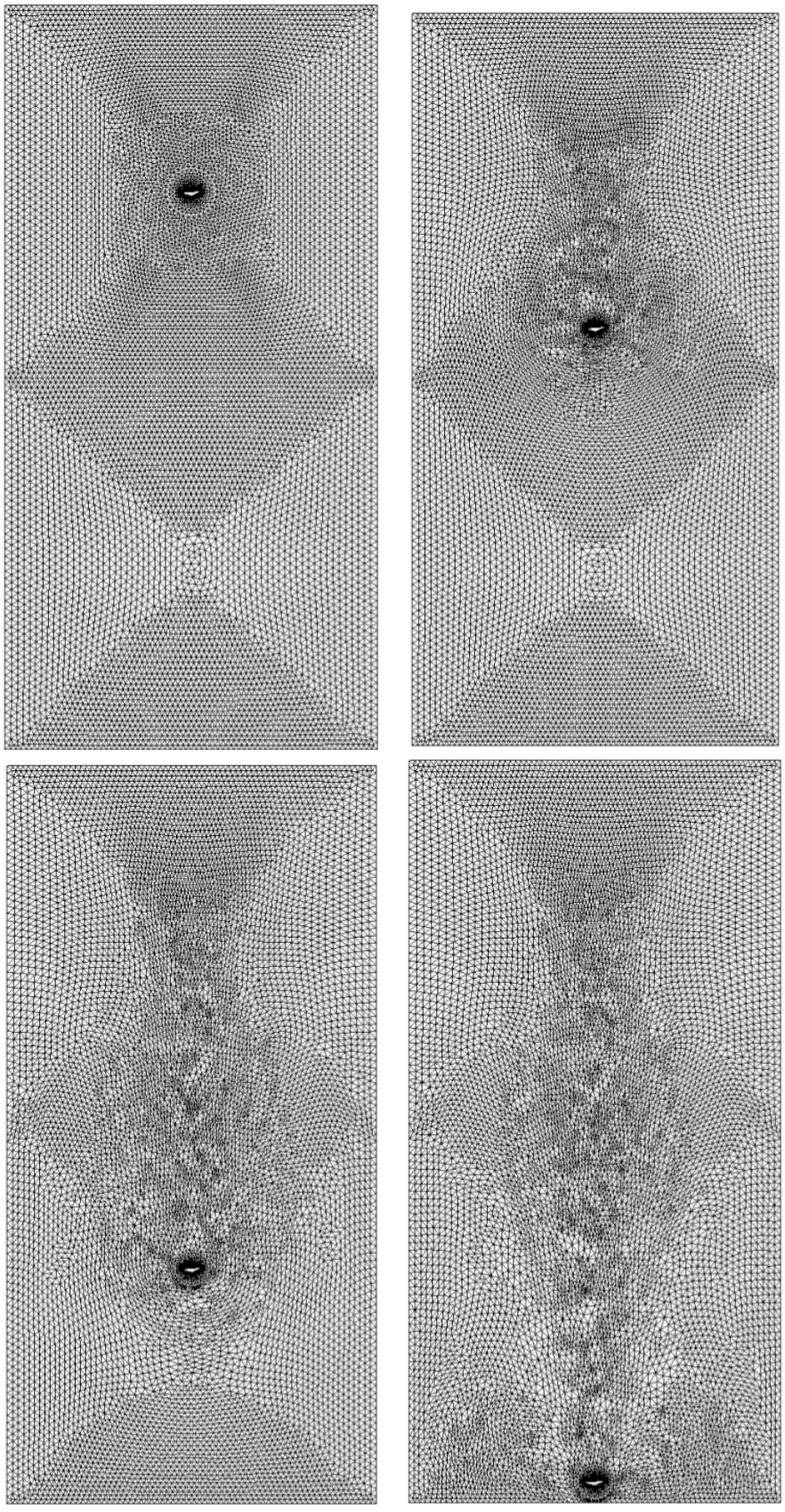


Figure 4.6: Movement of the mesh around a falling wedge enclosed inside a rigid-elements disk. It is possible to cross the whole computational domain using only swaps and vertex displacements.

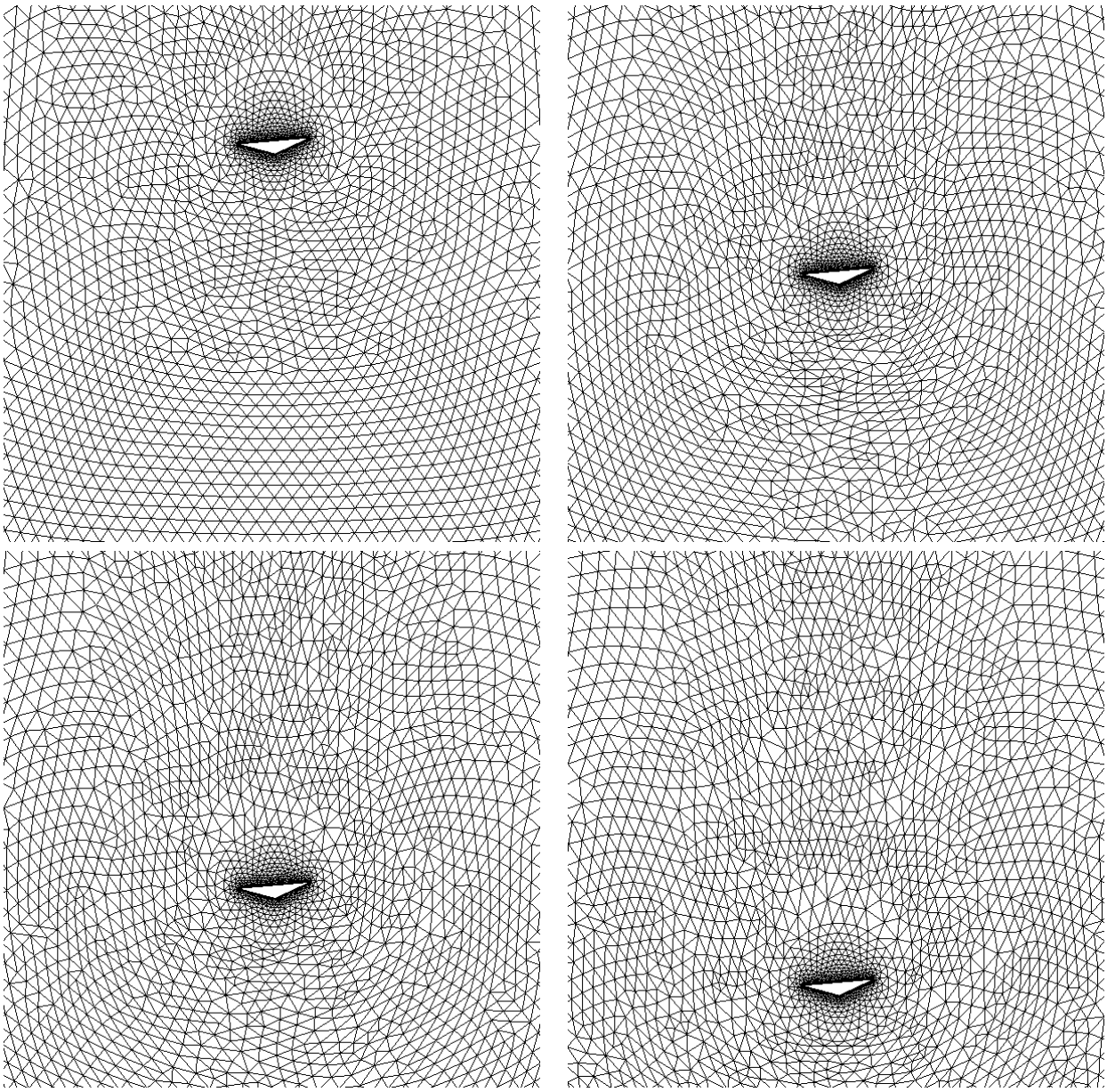


Figure 4.7: Close-up view near the falling wedge. Edge swapping enable to open the way to the falling geometry.

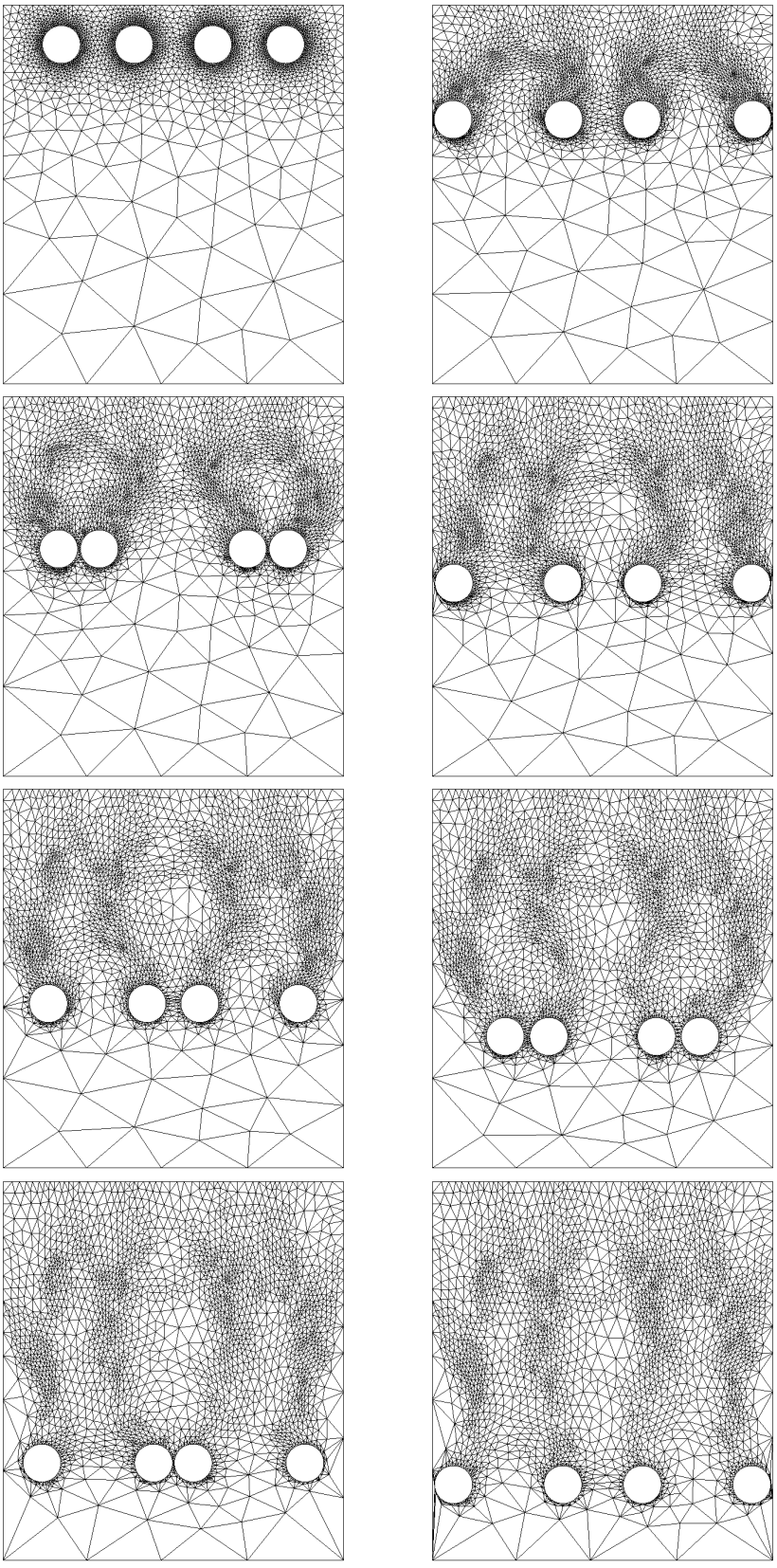


Figure 4.8: Four falling disks colliding with each other.

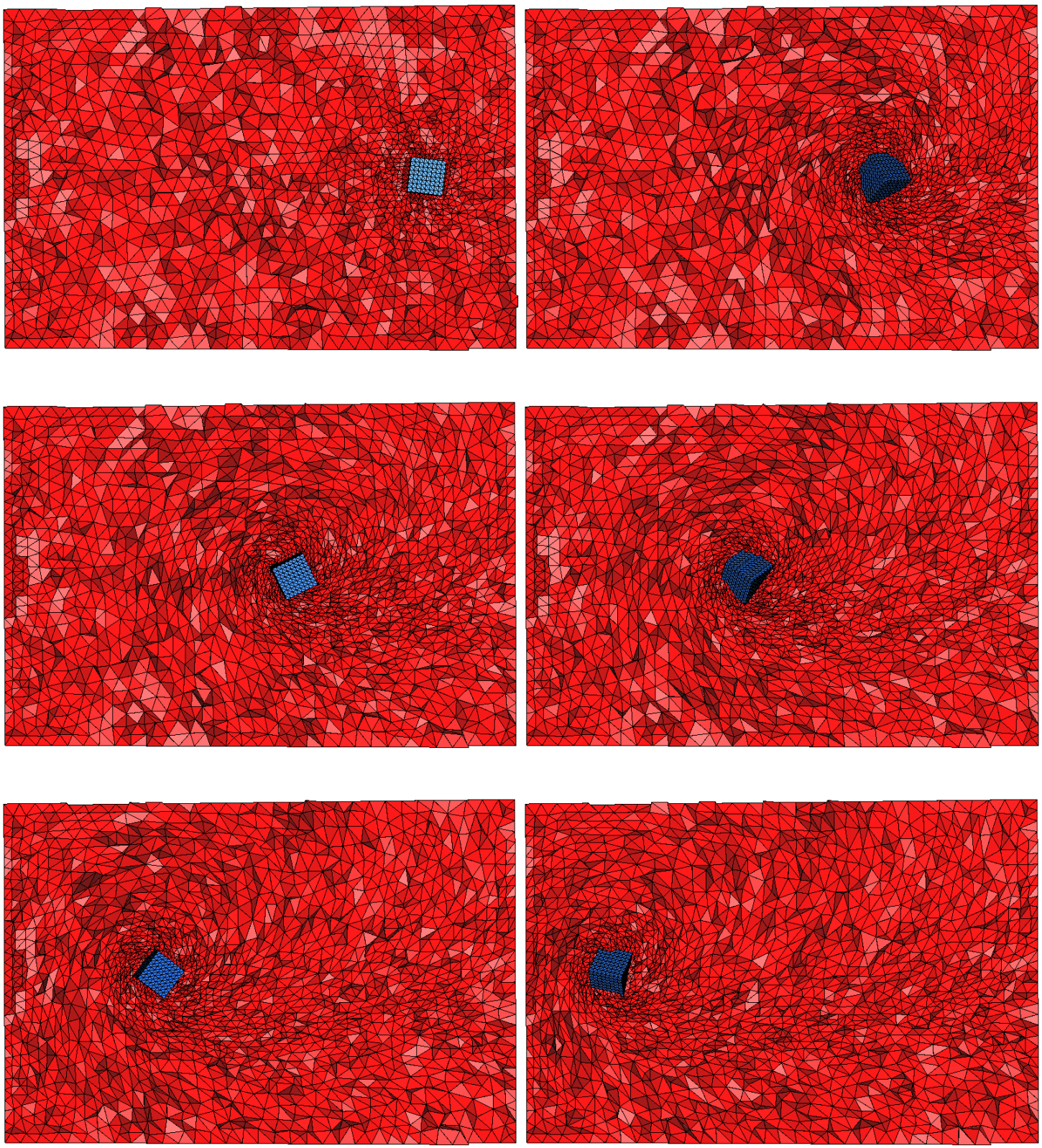


Figure 4.9: Snapshots of a moving rotating cube in three dimensions.

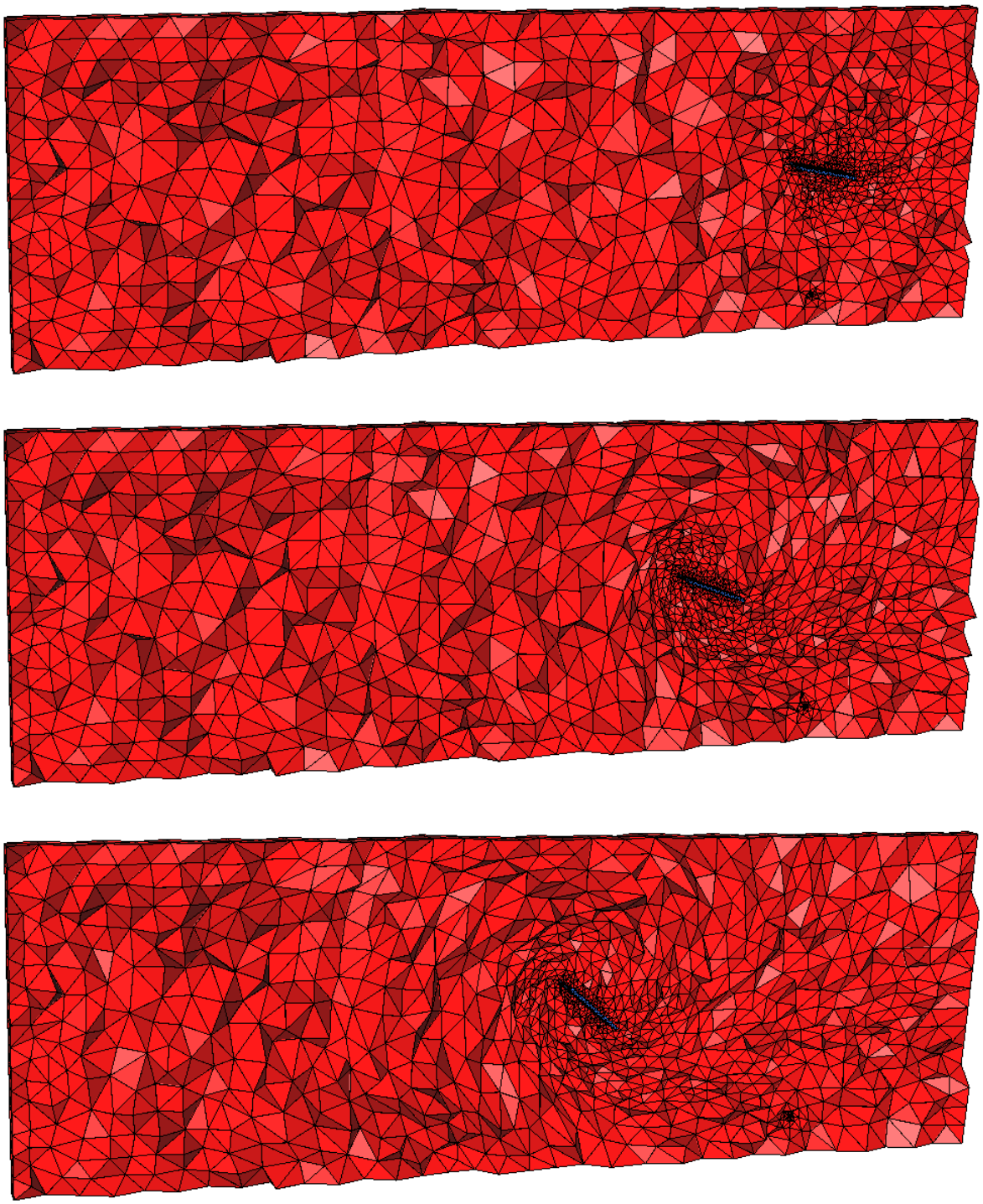


Figure 4.10: Snapshots of a moving rolling plate (1).

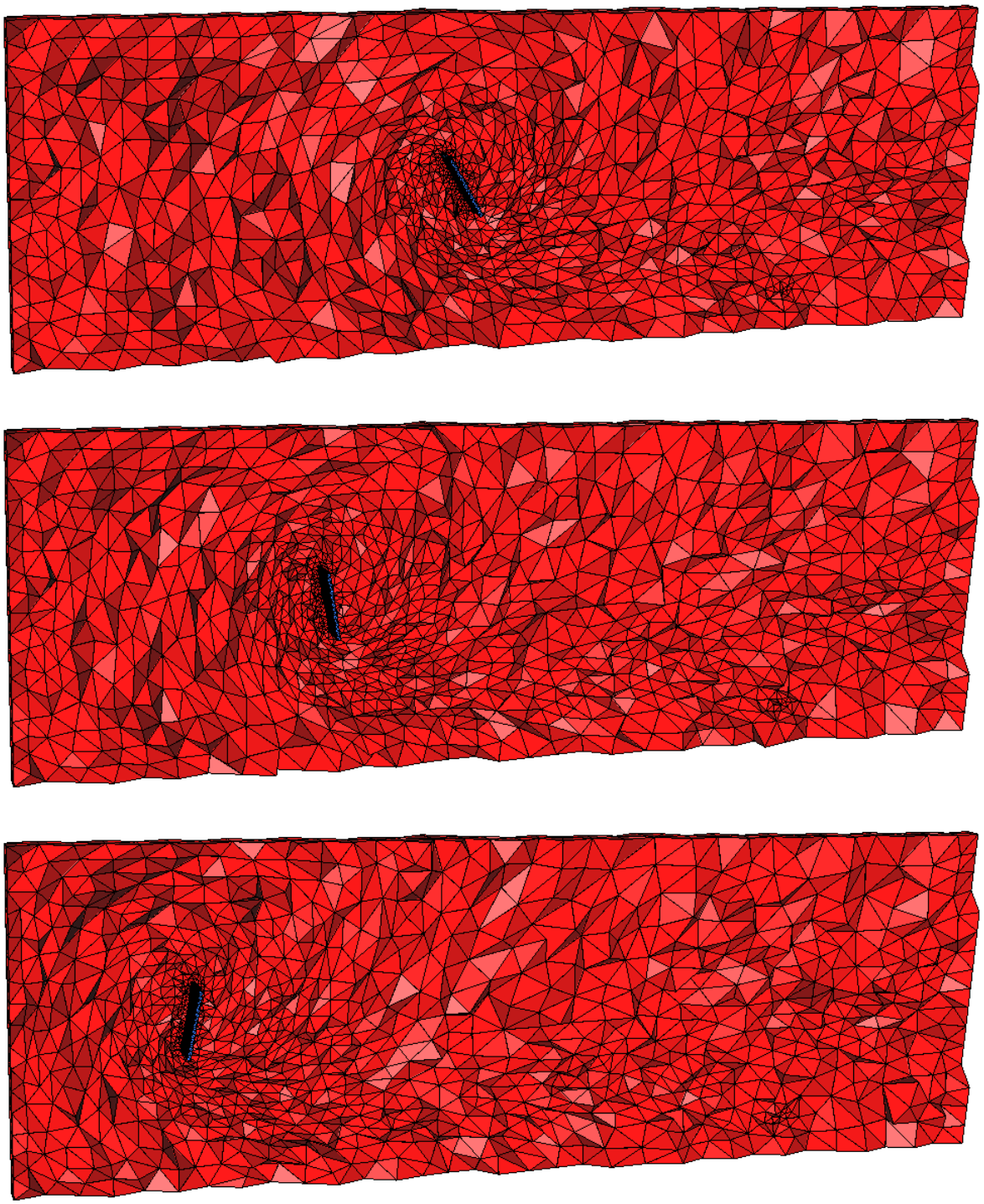


Figure 4.11: Snapshots of a moving rolling plate (2).

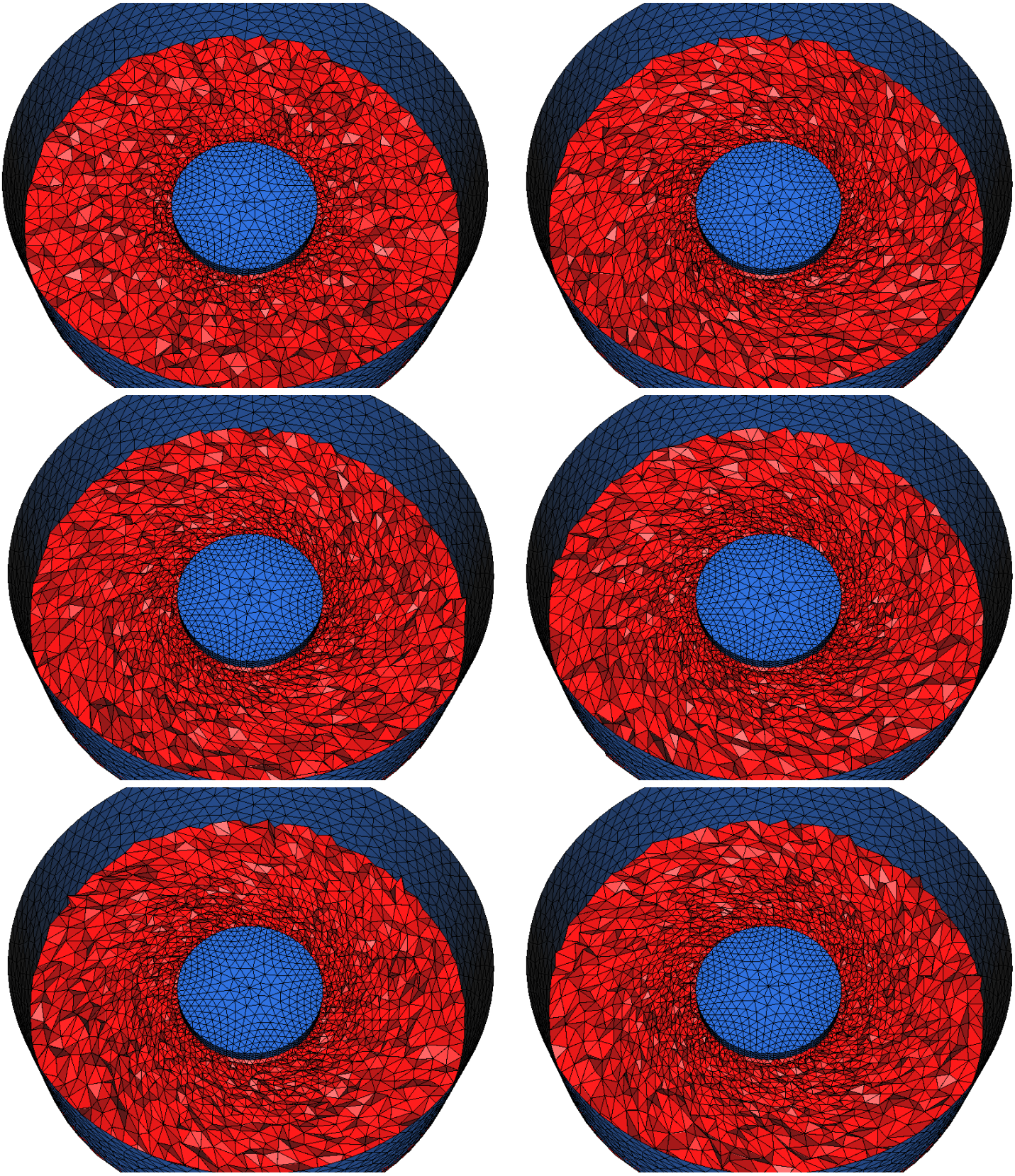


Figure 4.12: Snapshots of a swirling cylinder. From left to right and from top to bottom, the cylinder after a rotation of 0, 0.4π, 0.8π, 1.2π, 1.6π and 2π.

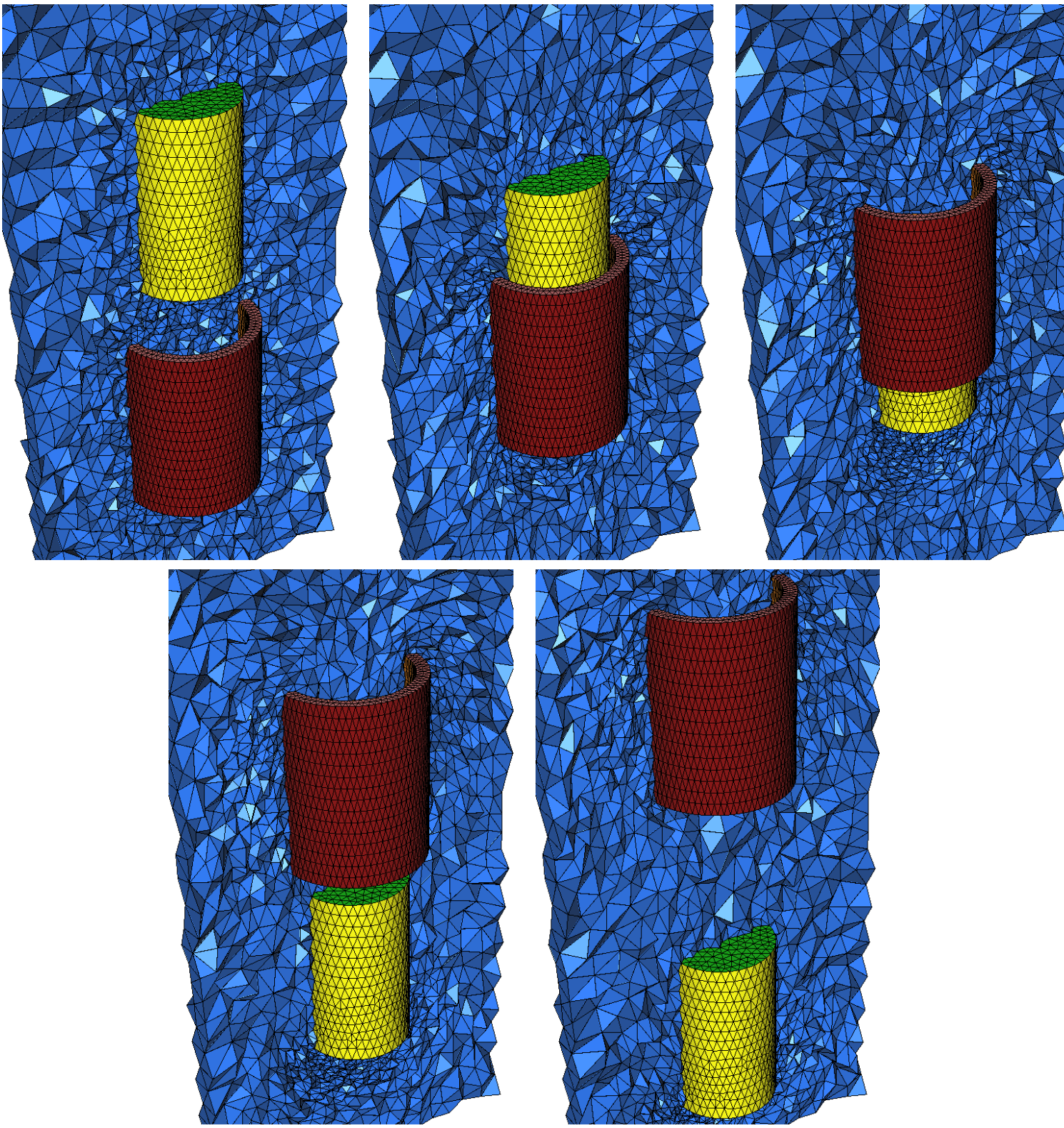


Figure 4.13: Snapshots of two cylinders interpenetrating.

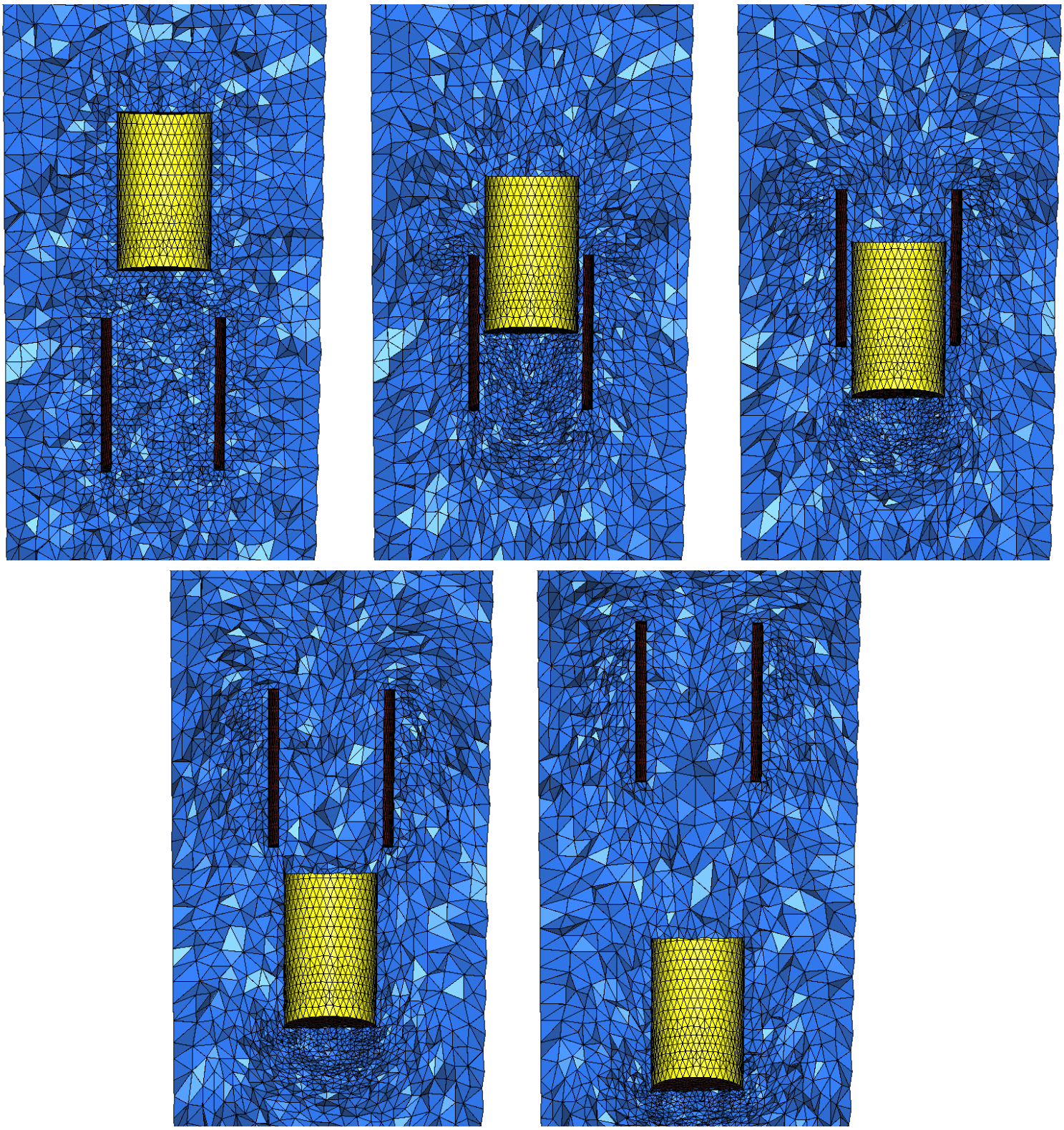


Figure 4.14: Snapshots of two cylinders interpenetrating. Only one layer of elements separates the two geometries and undergoes strong shearing.

Chapter 5

The Arbitrary-Lagrangian-Eulerian solver

Contents

5.1 Modelization of the physical problem 168
5.1.1 The Arbitrary-Lagrangian-Eulerian framework 168
5.1.2 The Euler equations in the Arbitrary-Lagrangian-Eulerian formulation . . 169
5.2 Spatial discretization of the ALE Euler equations 170
5.2.1 Finite-Volume edge-based formulation 171
5.2.2 High-order schemes 174
5.2.3 Mirror state boundary conditions 176
5.3 ALE specific issues regarding time discretization 178
5.3.1 The GCL law 178
5.3.2 Accuracy preserving and DGCL temporal schemes 178
5.3.3 Runge-Kutta Strong-Stability-Preserving schemes 181
5.3.4 DGCL RKSSP schemes for moving mesh simulations. 182
5.3.5 Practical computation of the areas swept 187
5.4 A new changing-topology ALE formulation 189
5.4.1 Problematics 189
5.4.2 Our approach 190
5.4.3 Swapping evanescent cell and volume redistribution 190
5.4.4 Notations 192
5.4.5 Swept areas computations 194
5.4.6 The changing-topology ALE schemes 200
5.4.7 Generalizations 204
5.4.8 Scheme analysis 208
5.5 Fluid Structure Interactions issues 212
5.5.1 Description of the moving objects 212
5.5.2 Movement of the geometries 214
5.5.3 Resolution of the rigid bodies equations and loose coupling 217
5.6 Numerical results 221
5.6.1 Turbo-machinery 221
5.6.2 Pitching NACA0012 airfoil 223
5.6.3 Blast test case 224

This chapter introduces the Arbitrary-Lagrangian-Eulerian solver. Section 5.1 recalls the basics of the ALE formulation and provides the expression of the Euler system of equations in this context. Section 5.2 details the spatial schemes used for our computations. In Section 5.3, ALE specific issues regarding time discretization are explained and several Runge-Kutta Strong-Stability-Preserving schemes enforcing their associated Discrete Geometric Conservation Law (DGCL) are detailed. Then, Section 5.4 introduces a new variable-topology ALE scheme enabling the use of swapping operations during the optimization phase, see Chapter 4, in a fully ALE manner. This scheme enables to relax the strong fixed-topology constraint imposed by the classical ALE formulation. Finally, Section 5.5 describes the resolution of the rigid body dynamics.

5.1 Modelization of the physical problem

5.1.1 The Arbitrary-Lagrangian-Eulerian framework

In this section, we detail how the Euler equations read in the ALE framework.

Lagrangian description. The Lagrangian method follows the particles of the continuum medium in their movement. This is done by permanently keeping a track of the particles: each of them can be mapped at each time onto a constant initial reference configuration $R_{\mathbf{X}} = (\Omega_{\mathbf{X}}, \mathbf{X})$. When this framework is used to solve the equations of fluid or solid mechanics, each vertex is constantly attached to a particle and moves with the same velocity as the one of the fluid $\mathbf{u} = \frac{\partial \mathbf{x}}{\partial t}|_{\mathbf{X}}$. This formulation is especially suited for solid mechanics computations and has some great assets. Notably, because of the perfect coincidence between particles and vertices throughout the simulation, there is no convective term in Lagrangian computations. Moreover, these methods deal with interfaces problems in a natural manner. However, especially for fluid simulations involving vorticity and shears, large mesh displacements occur, which often results in bad vertices repartition and distorted meshes. This can lead to a loss of accuracy and even the end of the computation.

Eulerian description. The Eulerian viewpoint is classically used to perform fluid dynamics computations. The basic idea is to look at the time evolution of the physical quantities at stake passing through a fixed point of space. In this description, the mesh remains fixed and the fluid particles move with respect to this fixed mesh. The velocity $\mathbf{u}(\mathbf{x}, t)$ computed at some position $\mathbf{x}$ thus corresponds to the instantaneous velocity of the point-wise particle which is located at coordinates $\mathbf{x}$ at t. Therefore, the Eulerian description only involves variables having an instantaneous significance in a fixed region of space. In this case, the reference configuration is updated at each time to the current configuration $R_{\mathbf{x}} = (\Omega_{\mathbf{x}}, \mathbf{x})$. Contrary to the Lagrangian description case, this framework enables the computation of complex fluid flows, involving high vorticity areas and shears. However, it is done at the price of an additional complexity in the equations: as particles move with respect to the mesh, non-linear convective terms appear. Another drawback is the difficulty arising when one wants to follow mobile interfaces or boundaries.

ALE description. ALE descriptions were first proposed for the generalization of Finite Difference or Finite Volume schemes to moving meshes. Trulio [Trulio 1961], Noh [Noh 1964], Franck and Lazarus [Franck 1964] and Hirt [Hirt 1974] were pioneers in this field. The description was transposed in the Finite Element context by Donea [Donea 1977], Belytschko [Belytschko 1978] and Hugues [Hugues 1978].
The basic idea of ALE is that there is a priori no reason for which the movement of the nodes should remain either fix (Eulerian description) or should follow exactly the fluid particles (Lagrangian formulation). On the contrary, the vertices movement and the particles movement can be totally de-correlated. To do this, an intermediate reference configuration $\Omega_\xi = \Omega_\xi(t)$ is introduced. The important thing is that the new reference configuration can evolve with time, but not necessarily by following the particles like in the purely Eulerian case. If $\boldsymbol{\phi}_t : \Omega_\xi(t) \to \Omega_x(t)$ is the mapping between the ALE reference configuration and the current domain at t, the velocity of the mesh is defined by :

$$\boldsymbol{w}(\xi, t) := \frac{\partial \boldsymbol{\phi}_t}{\partial t}|_\xi(\xi, t)\,,$$

which represents the instantaneous velocity of the points of the domain.
For an arbitrary function $\mathbf{g} = \mathbf{g}(\mathbf{x}, t)$, we can write two times Reynolds transport theorem [Belytschko 2000], first taking $R_\mathbf{X}$ and second taking R_ξ as reference configurations:

$$\frac{\partial}{\partial t}|_\mathbf{X} \left(\int_{C(t)} \mathbf{g}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} \right) = \int_{C_{\mathrm{co}} \equiv C(t)} \frac{\partial \mathbf{g}\,(\mathbf{x}, t)}{\partial t}|_\mathbf{x}\, \mathrm{d}\mathbf{x} + \int_{\partial C_{\mathrm{co}} \equiv \partial C(t)} \mathbf{g}(\mathbf{x}, t)\, (\mathbf{u} \cdot \mathbf{n})\, \mathrm{d}\boldsymbol{s}$$

$$\frac{\partial}{\partial t}|_\xi \left(\int_{C(t)} \mathbf{g}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} \right) = \int_{C_{\mathrm{co}} \equiv C(t)} \frac{\partial \mathbf{g}\,(\mathbf{x}, t)}{\partial t}|_\mathbf{x}\, \mathrm{d}\mathbf{x} + \int_{\partial C_{\mathrm{co}} \equiv \partial C(t)} \mathbf{g}(\mathbf{x}, t)\, (\boldsymbol{w} \cdot \mathbf{n})\, \mathrm{d}\boldsymbol{s}$$

By subtraction, we get:

$$\frac{\partial}{\partial t}|_\mathbf{X} \left(\int_{C(t)} \mathbf{g}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} \right) = \frac{\partial}{\partial t}|_\xi \left(\int_{C(t)} \mathbf{g}(\mathbf{x}, t)\, \mathrm{d}\mathbf{x} \right) + \int_{\partial C_{\mathrm{co}} \equiv \partial C(t)} \mathbf{g}(\mathbf{x}, t)\, ((\mathbf{u} - \boldsymbol{w}) \cdot \mathbf{n})\, \mathrm{d}\boldsymbol{s}$$

and the notation $C_{\mathrm{co}} \equiv C(t)$ denotes the region in the physical space that coincides with $C(t)$ at instant t.

5.1.2 The Euler equations in the Arbitrary-Lagrangian-Eulerian formulation

The Euler equations are a classical model for inviscid compressible fluids. In the continuous medium representation, they are obtained by writing the conservation of mass, momentum and energy of a moving particle occupying a closed moving volume $C(t)$ and using Reynolds theorem. Assuming that the gas is perfect, inviscid and that there is no thermal diffusion, the Euler equations in the ALE framework read, for any arbitrary closed volume $C(t)$ of boundary $\partial C(t)$:

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

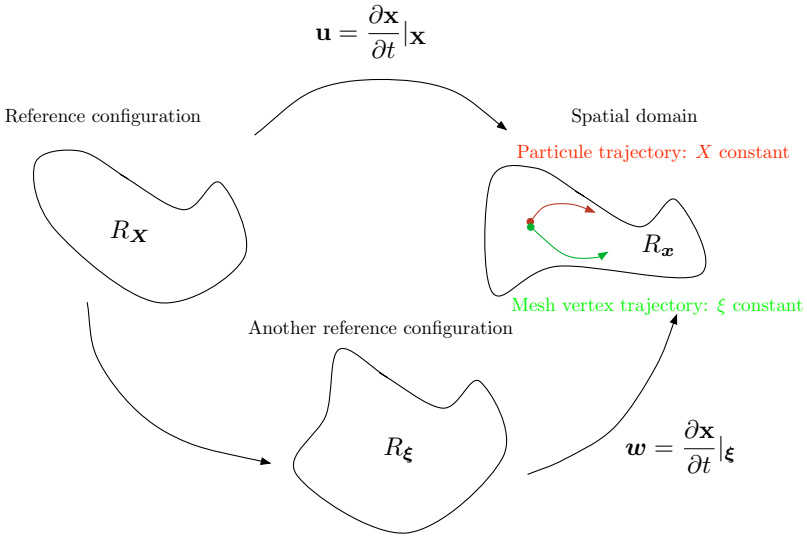


Figure 5.1: Reference configurations and physical domain.

$$\text{where} \begin{cases} W = (\rho, \rho\mathbf{u}, \rho e)^T \text{ is the conservative variables vector} \\ \mathcal{F}(W) = (\rho\mathbf{u},\ \rho u_x\mathbf{u} + p\mathbf{e}_x,\ \rho u_y\mathbf{u} + p\mathbf{e}_y,\ \rho u_z\mathbf{u} + p\mathbf{e}_z,\ \rho\mathbf{u}h) \text{ is the flux tensor} \\ \mathbf{F}(W) = \mathcal{F}(W)\cdot\mathbf{n} = (\rho u^{nor},\ \rho u_x u^{nor} + pn_x,\ \rho u_y u^{nor} + pn_y,\ \rho u_z u^{nor} + pn_z,\ \rho e\eta + pu^{nor})^T \\ \mathbf{F}_{ext} = (0,\ \rho\,\mathbf{f}_{ext},\ \rho\mathbf{u}\cdot\mathbf{f}_{ext})^T \text{ is the external forces vector}, \end{cases}$$

and we have noted ρ the volume mass of the fluid, p its local pressure, $\mathbf{u} = (u_x, u_y, u_z)$ its Eulerian velocity, $u^{nor} = \mathbf{u}\cdot\mathbf{n}$, $q = \|\mathbf{u}\|$, ε the internal energy per unit mass, $e = 0.5\,q^2 + \varepsilon$ the total energy per unit mass, $h = e + p/\rho$ the enthalpy per unit mass of the flow, $\mathbf{f}_{ext}$ the resultant of the volume external forces applied on the particle and $\mathbf{n}$ the outward normalized normal to the interface $\partial C(t)$ of $C(t)$.
Note that we have only considered Newtonian fluids the behavior law of which writes $\mathcal{C} = -p\,\mathcal{I}_n + 2\mu\Xi = -p\,\mathcal{I}_n$ if the fluid is inviscid, with μ the kinematic viscosity and Ξ the deformation tensor of the fluid. It is important to understand that all the above physical quantities are ALE variables, *i.e.* they are functions of t and of ξ, the considered particle coordinates vector with respect to the current ALE reference configuration.

5.2 Spatial discretization of the ALE Euler equations

In this section, we first describe the chosen spatial scheme when the topology of the mesh remains fixed.

5.2.1 Finite-Volume edge-based formulation

Domain Ω is discretized by a tetrahedral unstructured mesh Ω_h. The vertex-centered Finite Volume formulation consists in associating with each vertex P_i of the mesh and at each time t a control volume or Finite Volume cell, denoted $C_i(t)$. The dual Finite Volume cell mesh is built by the rule of medians, see Figure 5.2. The common boundary $\partial C_{ij}(t) = \partial C_i(t) \cap \partial C_j(t)$ between two neighboring cells $C_i(t)$ and $C_j(t)$ is decomposed into several triangular interface facets (bi-segments in two dimensions). The normal flux $\mathbf{F}_{ij}(t)$ along each cell interface is taken constant (not in time but in space), just like the approximation of the solution W_{ij} on the interface.
Rewriting System (5.1) for $C(t) = C_i(t)$ and using the above assumptions, we get the following semi-discretization at P_i:

$$\frac{\partial \left(|\, C_i(t) \,| W_i(t) \right)}{\partial t}|_\xi + \sum_{P_j \in \mathrm{Ball}(P_i)} |\, \partial C_{ij}(t) \,|\, \mathbf{\Phi}_{ij} \left(W_i(t),\, W_j(t),\, \boldsymbol{\eta}_{ij}(t),\, \sigma_{ij}(t) ||\boldsymbol{\eta}_{ij}(t)|| \right) = 0\,, \quad (5.1)$$

- $W_i(t)$ is the mean value of state W in cell C_i at t,
- $\mathrm{Ball}(P_i)$ is the set of all neighboring vertices of P_i,
- $\boldsymbol{\eta}_{ij}$ is the outward non-normalized normal (with respect to cell C_i) of cell interface ∂C_{ij},
- $\mathbf{F}_{ij}(t) = \mathcal{F}(W_{ij}(t)) \cdot \mathbf{n}_{ij}(t)$ is an approximation of the physical flux through $\partial C_{ij}(t)$[1]
- $\sigma_{ij}(t)$ is an approximation of the normal velocity of cell interface $\partial C_{ij}(t)$,
- $\mathbf{\Phi}_{ij}\left(W_i(t), W_j(t), \mathbf{n}_{ij}(t), \sigma_{ij}(t)\right) \approx \mathbf{F}_{ij}(t) - W_{ij}(t)\sigma_{ij}(t)$ is the numerical flux function used to approximate the flux at cell interface $\partial C_{ij}(t)$.

The computation of the convective fluxes is performed mono-dimensionally in the direction normal to each Finite Volume cell interface. Consequently, the numerical calculation of flux function $\mathbf{\Phi}_{ij}$ at interface ∂C_{ij} can be achieved by the resolution at each time step of a one-dimensional Riemann problem in direction $\mathbf{n}_{ij} = \mathbf{n}$ (from left to right) with initial values $W_L = W_i$ on the left of the interface and $W_R = W_j$ on the right. The normal speed to the interface is temporarily noted σ for clarity reasons.
Different kinds of approximate Riemann solvers can be used.

Roe solver. In the Roe approach, the upwinding term is defined through the Jacobian matrix of $\mathbf{F}$:

$$\mathcal{A}(W) \;=\; \frac{\partial \left(\mathcal{F}(W) \cdot \mathbf{n} \right)}{\partial W} \;=\; \frac{\partial \mathbf{F}(W)}{\partial W}\,.$$

The eigenvalues of $\mathcal{A}$ are real and given by u^{nor}, $u^{nor} + c$ and $u^{nor} - c$, where $u^{nor} = \mathbf{u} \cdot \mathbf{n}$ and c is the celerity or local sound speed of the fluid. In the context of the Euler equations, the hyperbolic flux is homogeneous of order one, leading to:

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

[1] Our convention is that the flux is positive if it goes in the same direction as the normal. Thus, $\mathbf{F}_{ij} > 0$ means that the flux goes from cell C_i to C_j. If $\sigma_{ij} > 0$, geometrical flux $-W_{ij}(t)\sigma_{ij}$ is negative and therefore oriented from C_j to C_i, which means that cell C_i steals mass from C_j,

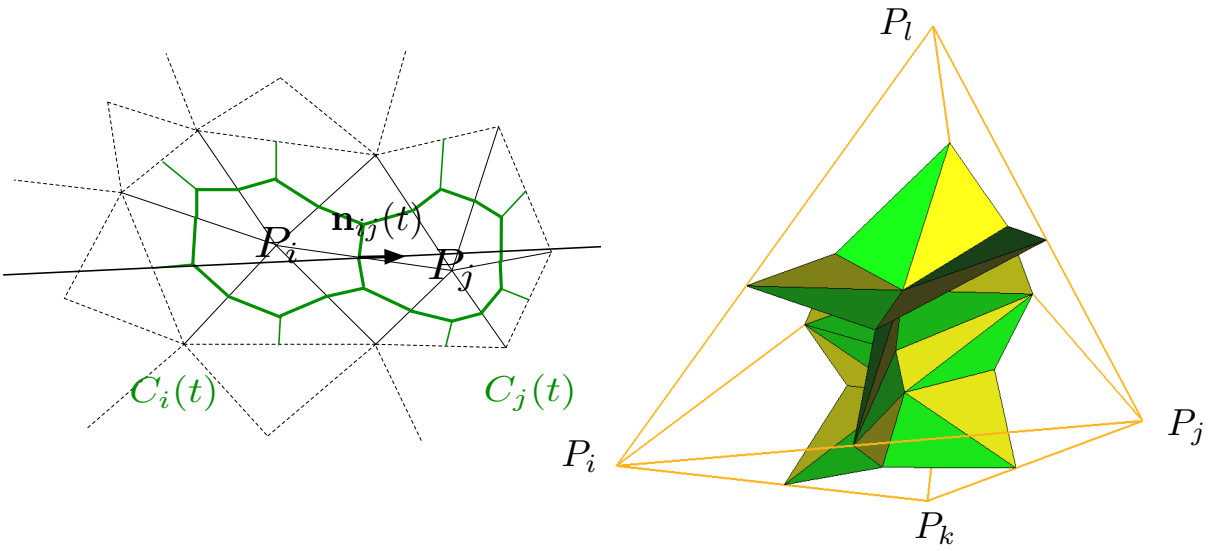


Figure 5.2: Left, Finite-Volume median cells in two dimensions. Right, Finite-Volume median cells in three dimensions inside two neighboring tetrahedra. Note that median cells lead to coplanar quadrangular facets.

that enables non-oscillatory conservative schemes to be built. If we note $\mathbf{F}_i = \mathcal{F}(W_i)\cdot\mathbf{n}$ and $\mathbf{F}_j = \mathcal{F}(W_j)\cdot\mathbf{n}$, the Roe flux function $\mathbf{\Phi}^{\text{Roe}}$ is given by:

$$\mathbf{\Phi}^{\text{Roe}}(W_i,\, W_j,\, \sigma_{ij}||\boldsymbol{\eta}_{ij}||,\, \boldsymbol{\eta}_{ij}) \;=\; \frac{F_i + F_j}{2} - \sigma_{ij}\,\frac{W_i + W_j}{2} \;+\; |\tilde{\mathcal{A}}(W_i,\, W_j) - \sigma_{ij}\mathcal{I}_n|\frac{W_i - W_j}{2}$$

where $\tilde{\mathcal{A}}$ is the Jacobian of $\mathbf{F}$ evaluated for the Roe average variables. For diagonalizable tensor $\tilde{\mathcal{A}} = \mathcal{P}\mathcal{D}\mathcal{P}^{-1}$, we have noted $|\,\tilde{\mathcal{A}}\,| = \mathcal{P}|\,\mathcal{D}\,|\mathcal{P}^{-1}$ where $|\,\mathcal{D}\,| = \text{diag}\left(|\tilde{\lambda}_1|,\, \ldots,\, |\tilde{\lambda}_{n+2}|\right)$. The Roe averages are given by:

$$\tilde{\rho} = \sqrt{\rho_i \rho_j}\,, \quad \tilde{\mathbf{u}} = \frac{\sqrt{\rho_i}\,\mathbf{u}_i + \sqrt{\rho_j}\,\mathbf{u}_j}{\sqrt{\rho_i} + \sqrt{\rho_j}} \quad \text{and} \quad \tilde{h} = \frac{\sqrt{\rho_i}\,h_i + \sqrt{\rho_j}\,h_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}$$

from which we get the Roe average sound speed: $\tilde{c}^2 = (\gamma - 1)\,(\tilde{h} - \frac{1}{2}\tilde{q}^2)$
However, the Roe scheme is not positive, which means that negative sound speeds, densities or energies can appear. It is neither entropic. Indeed, as this scheme is based on a linearization procedure, an expansion may be replaced by a non-entropic shock wave, notably at possible sonic points $q = c$ in expansion fans. At these points, the numerical viscosity becomes zero and the scheme might be unstable. Therefore, much attention should be paid while using the Roe scheme.

Harten-Lax-van Leer Contact wave (HLLC) solver. The methodology provided in [Harten 1983, Batten 1997] can be extended to the Euler equations in their ALE formulation. We still want to find an approximate value of the flux passing through moving interface ∂C_{ij} during a time τ. Let us first recall that the solution of this local Riemann problem is made of two acoustic waves of phase speeds $S_i = S_L$, $S_j = S_R$ which can be either shocks or expansion

fans and of a contact wave of phase speed S_M[2]. This is a classical result of the one-dimensional Euler equations theory. Normally, we should distinguish between four to six different regions depending on whether S_L (resp. S_R) is a shock wave or an expansion fan, see Figure 5.3 (left). The idea of the HLLC scheme is to simplify the complete Riemann problem by considering only four constant states $W_i = W_L$, $W_i^* = W_L^*$, $W_j^* = W_R^*$, $W_j = W_R$, see Figure 5.3 (right). In the case of moving meshes, four different situations can occur depending on the movement of the interface. The HLLC flux is described by three waves phase velocities:

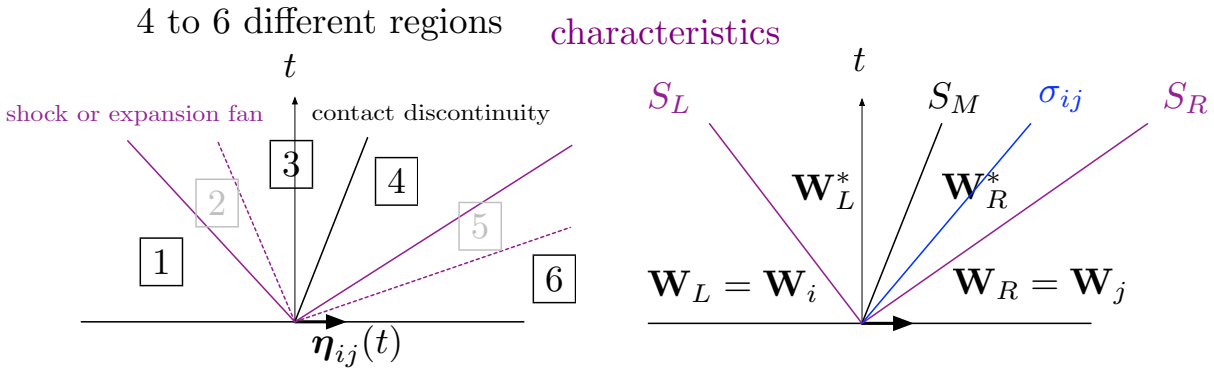


Figure 5.3: Left, characteristic lines of the exact solution for the Riemannian problem in direction $\mathbf{n}_{ij}$. Right, approximation of the solution with HLLC scheme, using only four approximate states.

$$S_i = S_L = \min u_L^{nor} - c_L,\ \tilde{u}^{nor} - \tilde{c} \quad \text{and} \quad S_j = S_R = \max u_R^{nor} + c_R,\ \tilde{u}^{nor} + \tilde{c}$$

$$S_M = \frac{\rho_R u_R^{nor}(c_R - u_R^{nor}) - \rho_L u_L^{nor}(c_L - u_L^{nor}) + p_L - p_R}{\rho_R(c_R - u_R^{nor}) - \rho_L(c_L - u_L^{nor})}$$

and two approximate states:

$$W_i^* = \begin{cases} \rho_i^* = \rho_i \dfrac{S_i - u_i^{nor}}{S_i - S_M} \\ p_i^* = p^* = \rho_i\,(u_i^{nor} - S_i)\,(u_i^{nor} - S_M) + p_L \\ (\rho\,\mathbf{u})_i^* = \dfrac{(S_i - u_i^{nor})\,\rho\,\mathbf{u}_i + (p^* - p_i)\,\mathbf{n}}{S_i - S_M} \\ (\rho\,e)_i^* = \dfrac{(S_i - \eta_i)\,\rho\,e_i - p_i u_i^{nor} + p^* S_M}{S_i - S_M} \end{cases} \qquad W_j^* = \begin{cases} \rho_j^* = \rho_j \dfrac{S_j - u_j^{nor}}{S_j - S_M} \\ p_j^* = p^* = \rho_j\,\left(u_j^{nor} - S_j\right)\left(u_j^{nor} - S_M\right) + p_j \\ (\rho\,\mathbf{u})_j^* = \dfrac{\left(S_j - u_j^{nor}\right)\rho\,\mathbf{u}_j + (p^* - p_j)\,\mathbf{n}}{S_j - S_M} \\ (\rho\,e)_j^* = \dfrac{(S_j - \eta_j)\,\rho\,e_j - p_R u_j^{nor} + p^* S_M}{S_j - S_M} \end{cases}.$$

If we note $\sigma = \sigma_{ij}$, the HLLC flux through moving interface ∂C_{ij} is finally given by:

$$\mathbf{\Phi}^{\text{Hllc}}(W_i, W_j, \boldsymbol{\eta}_{ij}, \sigma||\boldsymbol{\eta}_{ij}||) = \begin{cases} \mathbf{F}_i - \sigma W_L & \text{if } S_i - \sigma > 0 \\ \mathbf{F}_i^* - \sigma W_i^* & \text{if } S_i - \sigma \leq 0 < S_M - \sigma \\ \mathbf{F}_j^* - \sigma W_j^* & \text{if } S_M - \sigma \leq 0 \leq S_j - \sigma \\ \mathbf{F}_j - \sigma W_j & \text{if } S_j - \sigma < 0 \end{cases}$$

The HLLC approximate Riemann solver has the following properties. It automatically:

[2] $_L$ for left, $_R$ for right and $_M$ for middle

i) satisfies the entropy inequality,

ii) resolves isolated contacts exactly,

iii) resolves isolated shocks exactly,

iv) and preserves positivity.

5.2.2 High-order schemes

The previous formulation reaches at best first-order spatial accuracy. Fortunately, higher-order extensions are possible using a MUSCL type technique. The 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 [Van Leer 1972]. The idea is to use extrapolated values W_{ij} and W_{ji} of W at interface ∂C_{ij} to evaluate the flux, see Figure 5.4.
The following approximation is performed:

$$\mathbf{\Phi}_{ij} = \mathbf{\Phi}(W_{ij}, W_{ji}, \boldsymbol{\eta}_{ij}, \sigma_{ij}||\boldsymbol{\eta}_{ij}||)\,,$$

where W_{ij} and W_{ji} are linearly interpolated state values on each side of the interface:

$$W_{ij} \;=\; W_i + \frac{1}{2}\,(\nabla W)_{ij}\cdot\overrightarrow{P_iP_j}\,, \quad \text{and} \quad W_{ji} \;= W_j + \frac{1}{2}\,(\nabla W)_{ji}\cdot\overrightarrow{P_iP_j}\,. \tag{5.2}$$

Contrary to the original MUSCL approach, the approximate "slopes" $(\nabla W)_{ij}$ and $(\nabla W)_{ji}$ are defined for each edge using a combination of centered, upwind and nodal gradients.

Centered gradients. The centered gradient, which is related to edge $\mathbf{e}_{ij}$, is defined as:

$$(\nabla W)^C_{ij}\cdot\overrightarrow{P_iP_j} = W_j - W_i\,.$$

Upwind/Downwind gradients. Upwind and downwind gradients related to edge $\mathbf{e}_{ij}$ are computed using the upstream and downstream tetrahedra associated with edge $\mathbf{e}_{ij}$. These tetrahedra are respectively denoted K_{ij} and K_{ji} and are described in Figure 5.4. K_{ij} (resp. K_{ji}) is the unique tetrahedron of the ball of P_i (resp. P_j) the opposite face of which is crossed by the straight line prolongating edge $\mathbf{e}_{ij}$. Upwind and downwind gradients are then defined respectively for vertices P_i and P_j as:

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

where $\nabla W_{|_K} = \sum_{P\in K}(\nabla\phi_P \otimes W_P)$ is the $\mathbf{P}^1$-Galerkin gradient on element K.

Parametrized nodal gradients are built by introducing the β-scheme:

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

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 $\beta = 1$.

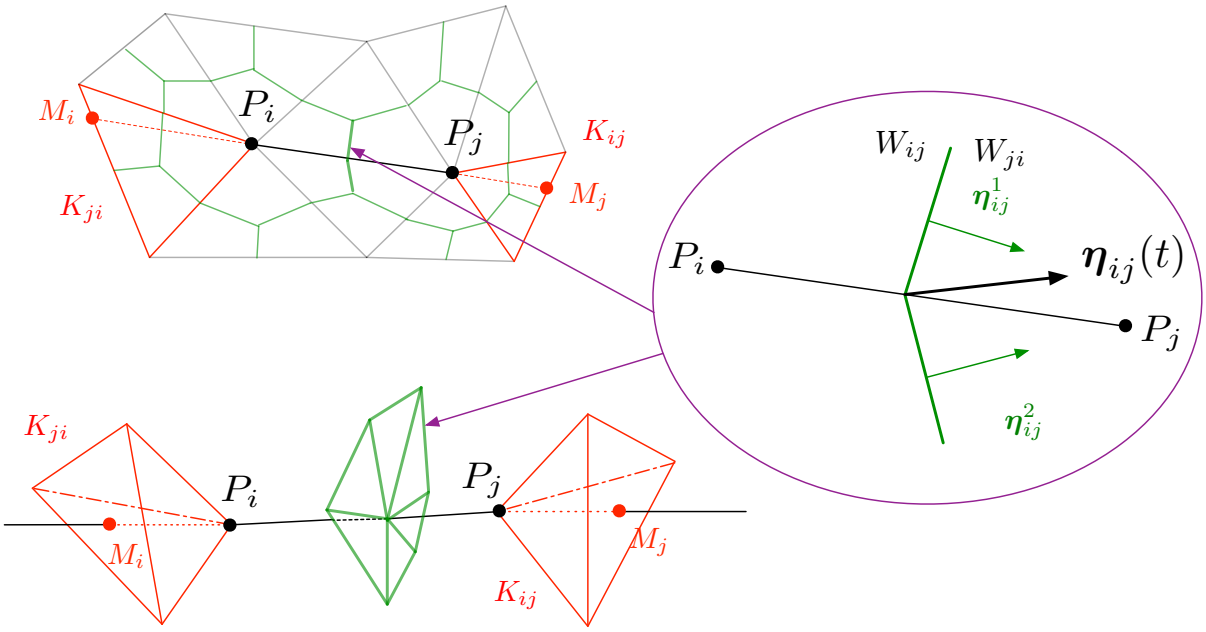


Figure 5.4: *Top left, upwind/downwind triangles K_{ij} and K_{ji} and upwind/downwind points M_i and M_j associated with edge $\mathbf{e}_{ij}$, in two dimensions. Bottom left, upwind/downwind tetrahedra K_{ij} and K_{ji} and upwind/downwind points M_i and M_j associated with edge $\mathbf{e}_{ij}$, in three dimensions. Right, extrapolated values W_{ij} and W_{ji} used at cells interfaces (in green) in the MUSCL approach.*

Numerical dissipation of fourth-order: V4-scheme. The most accurate β-scheme is obtained for $\beta = 1/3$. Indeed, it can be demonstrated that this scheme is third-order for the two-dimensional linear advection problem on structured triangular 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. These high-order gradients are given by:

$$(\nabla W)_{ij}^{V4} = \frac{2}{3}\,(\nabla W)_{ij}^{C} + \frac{1}{3}\,(\nabla W)_{ij}^{U} \quad \text{and} \quad (\nabla W)_{ji}^{V4} = \frac{2}{3}\,(\nabla W)_{ij}^{C} + \frac{1}{3}\,(\nabla W)_{ij}^{D}\,.$$

Numerical dissipation of sixth-order: V6-scheme. An even less dissipative scheme has been proposed in [Debiez 2000]. It is a more complex linear combination of gradients using centered, upwind and nodal $\mathbf{P}^1$-Galerkin gradients. The nodal $\mathbf{P}^1$-Galerkin gradient of P_i is related to cell C_i and is computed by averaging the gradients of all the tetrahedra having P_i as a vertex:

$$(\nabla W)_i^N = \nabla W_{|_{P_i}} = \frac{1}{4\,|\mathrm{Ball}(P_i)|} \sum_{K\,\in \mathrm{Ball}(P_i)} \left(|K| \nabla W_{|_K}\right).$$

Upwind/downwind nodal gradients. The upwind (resp. downwind) nodal gradients $(\nabla W)_i^{UN} = \nabla W_{|_{M_i}}$ (resp. $(\nabla W)_j^{DN} = \nabla W_{|_{M_j}}$) are computed by linear interpolation of the nodal gradients attached to the three vertices of the face containing M_i (resp. M_j), where M_i and M_j have been defined above in Figure 5.4.

A sixth-order dissipation scheme on structured meshes for the linear advection equation can be obtained by considering the following high-order gradient:

$$\begin{aligned}(\nabla W)_{ij}^{V6} &= (\nabla W)_{ij}^{V4} - \frac{1}{30}\left((\nabla W)_{ij}^{U} - 2\,(\nabla W)_{ij}^{C} + (\nabla W)_{ij}^{D}\right) \\ &\quad - \frac{2}{15}\left((\nabla W)_{i}^{HO} - 2\,(\nabla W)_{i}^{N} + (\nabla W)_{j}^{N}\right), \\ (\nabla W)_{ji}^{V6} &= (\nabla W)_{ij}^{V4} - \frac{1}{30}\left((\nabla W)_{ij}^{D} - 2\,(\nabla W)_{ij}^{C} + (\nabla W)_{ij}^{U}\right) \\ &\quad - \frac{2}{15}\left((\nabla W)_{j}^{HO} - 2\,(\nabla W)_{j}^{N} + (\nabla W)_{i}^{N}\right).\end{aligned}$$

On general meshes and non-linear equations, this scheme is of order two in space but with a sixth-order numerical dissipation.

Gradient limiter. The previous MUSCL schemes are not monotone. Therefore, limiting functions must be coupled with the previous high-order gradient evaluations to guarantee the TVD property of the scheme. To this aim, the gradient of Relation (5.2) is replaced by a limited gradient denoted $\left(\nabla W^{lim}\right)_{ij}$. Here, we will always consider a three entries limiter introduced in [Cournède 2006b], which is a generalization of the SuperBee limiter:

$$\left(\nabla W^{lim}\right)_{ij} = Lim\left(\left(\nabla W^{D}\right)_{ij}, \left(\nabla W^{C}\right)_{ij}, \left(\nabla W^{V4/V6}\right)_{ij}\right)$$

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

5.2.3 Mirror state boundary conditions

Mirror state boundary conditions consist in imposing slipping boundary conditions in a weak manner by prescribing a flux rather than directly enforcing a specific value for the variables on the boundary. As we are interested in the Euler equations, the fluid is inviscid and the physically consistent boundary condition on the moving bodies is a slipping boundary condition, *i.e.* $(\mathbf{u}\cdot\mathbf{n})_{|\partial B} = \sigma_{|\partial B}$. Mirror state $\overline{W}$ associated with boundary state W is an imaginary state, virtually defined on the other side of the boundary and such that the extrapolated value of W on the boundary satisfies $(\mathbf{u}\cdot\mathbf{n})_{|\partial B} = \sigma_{|\partial B}$. This extrapolated value is defined as:

$$W_{|\partial B} = \frac{1}{2}\left(W + \overline{W}\right).$$

Let us consider a Finite Volume cell C_i touching object B. We note $\mathbf{n}_{if}$ the outward normalized normal to boundary facet f and σ_{if} its normal speed, see Figure 5.5. The ALE mirror state $\overline{W}_{if}$ of state W_i on the other side of boundary interface f of cell C_i is:

$$\overline{W}^{if} = \left(\overline{\rho}_i = \rho_i,\, \overline{\mathbf{u}}_i = \mathbf{u}_i - 2\left(\mathbf{u}_i\cdot\mathbf{n}_{if} - \sigma_{if}\right)\mathbf{n}_{if},\, \overline{\varepsilon}_i = \varepsilon_i\right)^T.$$

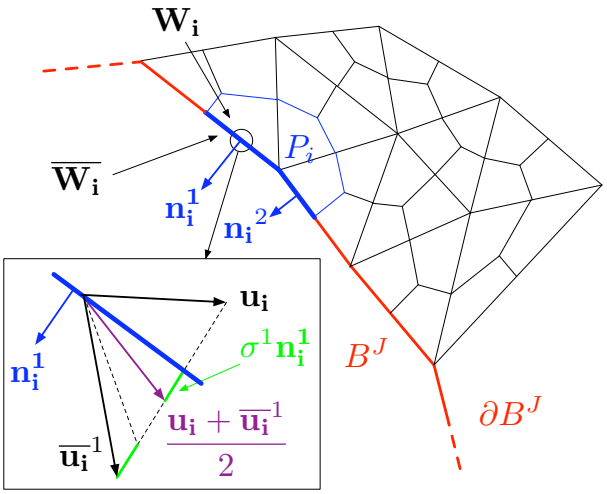


Figure 5.5: Mirror state $\overline{W}^{if}$ across boundary interface f of cell C_i.

Note that with this definition, we indeed have $\mathbf{u}_{if} \cdot \mathbf{n}_{if} = \sigma_{if}$.

In the sequel, we will note $u_{if}^{nor} = \mathbf{u}_i \cdot \mathbf{n}_{if}$.

Roe mirror state. For a Roe approximate Riemann solver, the boundary flux for the numerical fluxes between a boundary state and its mirror state across boundary interface f of cell C_i can be compute by hand and is given by:

$$\mathbf{\Phi}^{\text{Roe}}\left(W_i,\, \boldsymbol{\eta}_{if},\, \sigma_{if}||\boldsymbol{\eta}_{if}||\right) = ||\boldsymbol{\eta}_{if}|| \begin{pmatrix} 0 \\ \left[p_i + \rho_i\left(u_{if}^{nor} - \sigma_{if}\right)^2 + \rho_i \tilde{c}_i\left(u_{if}^{nor} - \sigma_{if}\right)\right]\mathbf{n}_{if} \\ p_i\sigma_{if} + \rho_i\sigma_{if}\left(u_{if}^{nor} - \sigma_{if}\right)\left(u_{if}^{nor} - \sigma_{if} + \tilde{c}_i\right) \end{pmatrix}$$

$$\text{with } \tilde{c}_i = (\gamma - 1)\left(h_i - \sigma_{if}u_{if}^{nor} - \frac{\left(u_{if}^{tan}\right)^2}{2} + \frac{(\sigma_{if})^2}{2}\right)$$

and $u_{if}^{tan} = ||\mathbf{u} - u_{if}^{nor}\mathbf{n}_{if}||$ the norm of the component of $\mathbf{u}$ tangent to the boundary.

HLLC mirror state. Regarding the HLLC scheme, we obtain:

$$\mathbf{\Phi}^{\text{Hllc}}\left(W_i,\, \boldsymbol{\eta}_{if},\, \sigma_{if}||\boldsymbol{\eta}_{if}||\right) = ||\boldsymbol{\eta}_{if}|| \begin{pmatrix} 0 \\ p_i^*\mathbf{n}_{if} \\ p_i^*\sigma_{if} \end{pmatrix}$$

where

$$p_i^* = \rho_i\left(u_{if}^{nor} - \sigma_{if}\right)\max c_i, \tilde{c}_i + u_{if}^{nor} - \sigma_{if} + p_i\,.$$

Details about the computation of these boundary fluxes are given in Appendix D.

Other boundary conditions (inflow, outflow, reactors, symmetry, transmitting) are applied on fixed boundaries and are therefore not specific to moving mesh computations. The reader is referred to [Godlewski 1996, Hirsch 1988b] for more details on this subject.

5.3 ALE specific issues regarding time discretization

In this section, we first recall an important issue which is specific to ALE computations: the numerical enforcement of the so-called Geometric Conservation Law (GCL) by the chosen temporal scheme. We next detail the genuine approach developed by Mavriplis and Yang on this subject. Finally, we detail how to build RKSSP schemes that are DGCL while preserving the order of accuracy of the fixed-mesh scheme they come from.

5.3.1 The GCL law

As already explained, ALE formulations have been designed to decouple the movement of the mesh from any physical description of the fluid. However, doing this imposes to check that the movement of the mesh is not responsible for any artificial alteration of the physical phenomena at stake. 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 (5.1) is written for a constant state, assuming that there is no external force, *i.e.* $\mathbf{F}_{ext} = 0$, we get, for any arbitrary closed volume $C = C(t)$ of boundary $\partial C(t)$:

$$\frac{\partial \left(|C(t)|\right)}{\partial t}_{|\boldsymbol{\xi}} - \int_{\partial C(t)} (\boldsymbol{w} \cdot \mathbf{n}) \, \mathrm{ds} \;=\; 0\,. \tag{5.3}$$

Note that the physical flux vanishes thanks to the closed nature of $C(t)$, which enables to write:

$$\int_{\partial C(t)} \mathbf{n} \, \mathrm{ds} \;=\; 0\,.$$

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. This relation is trivially integrated into:

$$|\,C(t+\tau)\,| \,-\, |\,C(t)\,| \;=\; \int_t^{t+\tau} \int_{\partial C(t)} (\boldsymbol{w} \cdot \mathbf{n}) \, \mathrm{d}\boldsymbol{s} \, \mathrm{d}t, \quad \text{with } t \text{ and } t + \mathrm{d}t \,\in\, [0,\, T]\,, \tag{5.4}$$

for any arbitrary closed volume $C = C(t)$ of boundary $\partial C(t)$. From a geometrical viewpoint, this relation simply states that the algebraic variation of the volume of C between two instants equals the algebraic area swept by its boundary during the same time.

5.3.2 Accuracy preserving and DGCL temporal schemes

A DGCL property for each temporal scheme. The continuous GCL relation raises several important questions: should Relation (5.3) be satisfied at the discrete level? What are the effects of respecting this law at a discrete level on the consistency, stability or accuracy of the scheme? How can the continuous GCL be enforced at a discrete level? In the following, we will restrict ourselves to Finite Volume schemes but the GCL and its importance in moving mesh Finite Element and Discontinuous Galerkin simulations is also a very active field of research

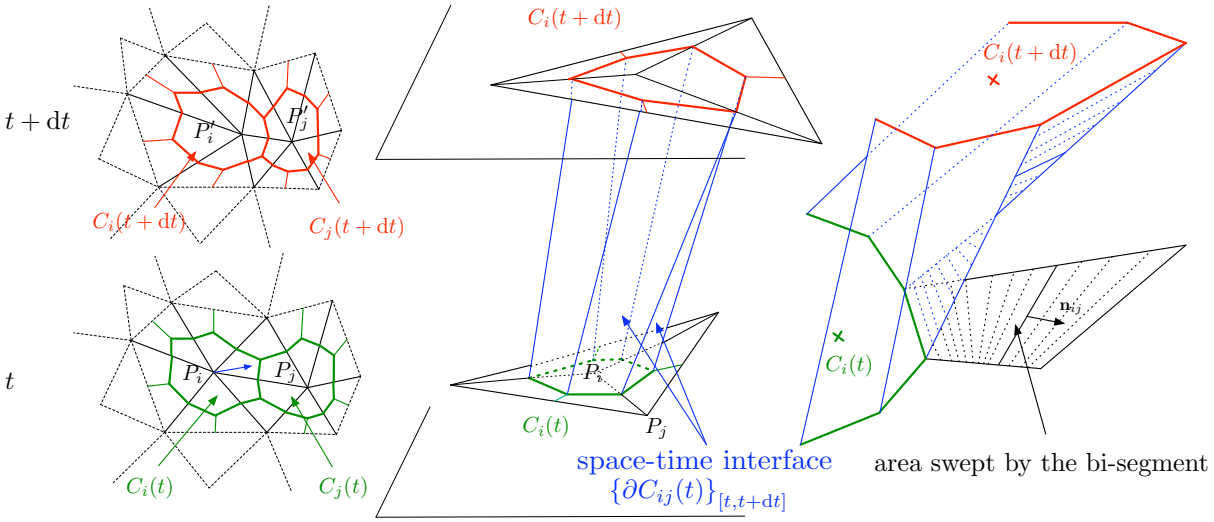


Figure 5.6: Areas swept by the interfaces of a two-dimensional Finite Volume cell during τ.

[Formaggia 1999, Boffi 2004, Etienne 2009]. Thomas and Lombard [Thomas 1979] were the first to emphasize the importance of this law for moving mesh simulations at the end of the 70's in a Finite Difference context. Since then, the subject has often proved controversial. Some people considered it as completely useless whereas others thought it had a strong influence on the quality of the scheme. It took a long time before a consensual line started to take shape. The reader is referred to [Etienne 2009] for a very clear and detailed state of the art on the subject.
However, there are currently two things almost everybody agrees about:

i) The GCL serves as an additional constraint controlling the way the computation of the geometrical parameters is performed. Indeed, we *a priori* ignore how to compute geometrical parameters σ and $\mathbf{n}$ appearing in Relation (5.1). For instance, should one take $\mathbf{n}$ at t^n? at t^{n+1}? Or take a kind of averaged normal? The GCL can help answer this question.

ii) Enforcing the GCL at the discrete level is mandatory for:

Accuracy : satisfying an appropriate DGCL is a sufficient condition for a numerical scheme to be at least first-order time-accurate on moving meshes [Guillard 2000]. However, it is not a necessary condition to preserve the accuracy order of the numerical scheme observed on fixed-mesh.

Non-linear stability : satisfying the corresponding DGCL is a necessary and sufficient condition for a numerical scheme to preserve the non-linear stability of its fixed grid counterpart. This has been proved for different implicit and explicit schemes and for various spatial discretization methods [Formaggia 1999, Farhat 2001, Formaggia 2004b, Boffi 2004, Cournède 2006a],

Energy conservation : using a DGCL numerical scheme ensures that the work of forces on fluid particles is accurately transformed into total fluid energy variation at the discrete

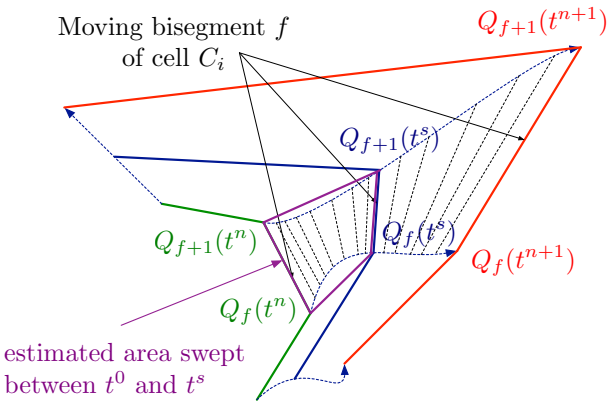


Figure 5.7: Exact swept area v.s estimated swept area.

level, including at domain boundaries [Dervieux 2010]. This point is crucial notably for computations involving moving pistons.

An exception can be made if sufficiently small time steps are used but, in practice, it is very time consuming and it is hard to know what "sufficiently small" means.

In the sequel, we detail Mavriplis-Yang approach to extend well-known temporal schemes to ALE simulations while enforcing the GCL. This method is illustrated on several optimal RKSSP schemes.

Mavriplis-Yang approach: σ as unique degree of freedom. Since 2005, Mavriplis and Yang [Yang 2005, Mavriplis 2006, Yang 2007] have renewed the way of thinking the GCL. The objective was initially to further clarify the link between the DGCL nature of a temporal scheme and the preservation of its order of accuracy. The originality of Mavriplis-Yang approach consists in defining precisely which ALE geometrical parameters are true degrees of freedom and which are not, which had not been done to our knowledge until then. In contrast with the other approaches [Lesoinne 1996, Koobus 1999, Nkonga 2000], they consider that the times and the configurations at which fluxes are evaluated do not constitute a new degree of freedom to be set thanks to the ALE scheme, and neither distinguish between different frameworks. To maintain the design accuracy of the fixed-mesh temporal integration, the moment at which the geometrical parameters (upwind/downwind elements, normals,...) 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 normal mesh speed σ. Incidentally, Mavriplis and Yang implicitly underline that $\boldsymbol{w}$ is never involved alone but only hidden in the term $\sigma||\boldsymbol{\eta}||$, which represents the instantaneous algebraic area swept.
Actually, this method comes to evaluate the areas swept by cells interfaces in a way which is consistent with the time discretization of the vertices trajectories. Indeed, the definition of the vertices trajectories, which are *a priori* curved, increases while the accuracy of the time integration increases. To maintain the order of accuracy, the computation of swept areas must take into account this better representation of vertices trajectories, see Figure 5.7. Practically speaking, the interfaces normal speeds are found by simply rewriting the scheme for a constant

discrete solution, which leads to a small linear system easily invertible by hand. This procedure is detailed in the next section for several RKSSP schemes. Any fixed-mesh scheme can be extended to moving mesh simulations thanks to this methodology, and the resulting scheme is naturally DGCL. Even if this has not been proven theoretically, the expected order of convergence has also been observed numerically for several schemes designed with this method [Yang 2005].

5.3.3 Runge-Kutta Strong-Stability-Preserving schemes

Runge-Kutta methods are famous multi-stages methods to integrate ODEs. In the following, n_s is the number of stages of the considered scheme and p its accuracy order. For a Runge-Kutta scheme to be of order p, its coefficients must satisfy several relations which can be found either by the study of the truncation error of the scheme or using the much more elegant theory of Butcher graphs [Butcher 1987, Lambert 1991]. The order of these schemes is always lower than the number of stages $p \leq n_s$. Besides, it has been proved that there exists no explicit Runge-Kutta scheme of order greater than 5.

Definition. When people got interested in the numerical resolution of hyperbolic partial differential equations, notably the Euler equations, they started to seek among the huge family of Runge-Kutta schemes for the schemes satisfying an additional property called the Strong-Stability-Preserving (SSP) property.
A Runge-Kutta scheme is said to be Strong-Stability-Preserving (SSP) [Shu 1988, Gottlieb 1998, Spiteri 2003, Kraaijevanger 1991, Ferracina 2005] if the following relation holds:

$$|\, W^{n+1} \,| \leq |\, W^n \,|\,,$$

$|\cdot|$ being here a chosen semi-norm. This semi-norm is classically the Total Variation Diminishing (TVD) semi-norm:

$$|\, W^{n+1} - W^n \,|_{TVD} = \sum_{i=1}^{N} |\, W_{i+1}^n - W_i^n \,|\,.$$

Indeed, under certain hypothesis, the solution of the Euler equations is TVD, *i.e.* the solution cannot exhibit new local extrema or minima during its time evolution. Maintaining this feature for the discrete solution is also of crucial importance. More details about RKSSP schemes can be found in Appendix C.

Notations. In the sequel, we note RKSSP(n_s, p) the n_s-stages RKSSP scheme of order p. The following notations are also adopted:

$$\mathbf{F}_i^s = \sum_{f=0}^{n_{ball,i}-1} \mathbf{\Phi}(W_i^s,\, \boldsymbol{\eta}_{if}^s,\, \sigma_{if}^s\, ||\boldsymbol{\eta}_{if}^s||)\,,$$

with $\begin{cases} n_{ball,i} & \text{the number of elements in the ball of vertex } P_i\,, \\ \boldsymbol{\eta}_{if}^s & \text{the outward } \textbf{non-normalized} \text{ normal to facet (or bi-segment) } f \text{ of cell } C_i^s\,, \\ \sigma_{if}^s & \text{the normal speed of facet } f \text{ of cell } C_i^s\ . \end{cases}$

Superscript notation X^s indicates that the considered quantity is the resulting quantity 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, see Figure 5.8 (left). Coefficients $(c_s)_{0\leq s\leq n_s}$ indicate the relative position in time of the current Runge-Kutta configuration. In other words, c_s is such that the time of the current Runge-Kutta configuration is:

$$t^s = t^n + c_s\,\tau\,, \quad \text{with } \tau \;=\; t^{n+1} - t^n\,.$$

Finally, we note A^s_{if} the area swept by facet f of cell C_i between the initial Runge-Kutta configuration at $t = t^0 = t^n$ and the s^{th} one at $t = t^s$.

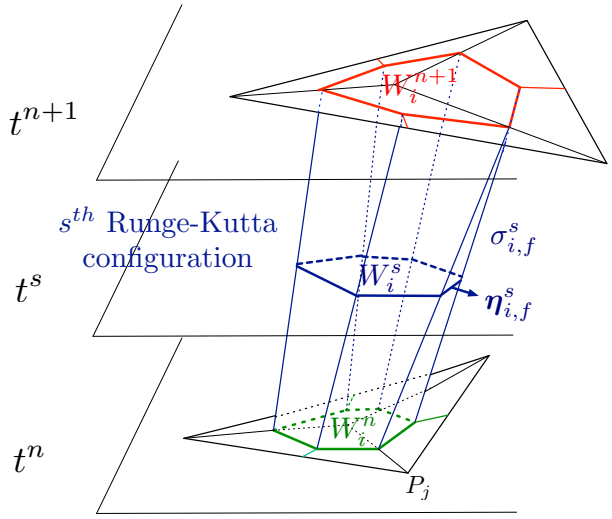


Figure 5.8: Left, s^{th} Runge-Kutta configuration and the geometrical parameters for bi-segment f of cell C_i at stage s. Right, estimation of the area swept by the bi-segment between the initial and the current Runge-Kutta configurations.

5.3.4 DGCL RKSSP schemes for moving mesh simulations.

For each RKSSP scheme, its Butcher representation (left) and its Shu-Osher [Shu 1988] formulation (right) are provided. Mavriplis-Yang method is first detailed for the RKSSP(4, 3) scheme, and the results obtained for the other optimal schemes are then listed.

5.3.4.1 An example of Mavriplis-Yang method: RKSSP(4, 3)

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

For this scheme to be DGCL, it must preserve a constant solution $W_i = W_0$. In this specific case, our conservative variable is $\mathbf{Y_i} = |C_i|W_0$ and the purely physical fluxes vanish, leading to:

$$\mathbf{F}_i^s \;=\; -\,W_0 \sum_{f=0}^{n_{ball,i}-1} ||\boldsymbol{\eta}_{if}^s||\sigma_{if}^s\,.$$

Therefore, the scheme writes:

$$\begin{aligned}
|C_i^0| &= |C_i^n| \\
|C_i^1| - |C_i^0| &= \sum_{f=0}^{n_{ball,i}-1} A_{if}^1 = \frac{\tau}{2}\sum_{f=0}^{n_{ball,i}-1} ||\boldsymbol{\eta}_{if}^0||\sigma_{if}^0 \\
|C_i^2| - |C_i^0| &= \sum_{f=0}^{n_{ball,i}-1} A_{if}^2 = \frac{\tau}{2}\sum_{f=0}^{n_{ball,i}-1} \left(||\boldsymbol{\eta}_{if}^0||\sigma_{if}^0 + ||\boldsymbol{\eta}_{if}^1||\sigma_{if}^1\right) \\
|C_i^3| - |C_i^0| &= \sum_{f=0}^{n_{ball,i}-1} A_{if}^3 = \frac{\tau}{6}\sum_{f=0}^{n_{ball,i}-1} \left(||\boldsymbol{\eta}_{if}^0||\sigma_{if}^0 + ||\boldsymbol{\eta}_{if}^1||\sigma_{if}^1 + ||\boldsymbol{\eta}_{if}^2||\sigma_{if}^2\right) \\
|C_i^4| - |C_i^0| &= \sum_{f=0}^{n_{ball,i}-1} A_{if}^4 = \frac{\tau}{2}\sum_{f=0}^{n_i-1} \left(\frac{1}{3}||\boldsymbol{\eta}_{if}^0||\sigma_{if}^0 + \frac{1}{3}||\boldsymbol{\eta}_{if}^1||\sigma_{if}^1 + \frac{1}{3}||\boldsymbol{\eta}_{if}^2||\sigma_{if}^2 + ||\boldsymbol{\eta}_{if}^3||\sigma_{if}^3\right).
\end{aligned}$$

A necessary and natural condition for the above relations to be satisfied is to have, for each facet (or bi-segment in 2D) f of each Finite Volume cell C_i:

$$\begin{aligned}
&\begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \\ A_{if}^4 \end{pmatrix} = \tau \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\|\sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\|\sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\|\sigma_{if}^2 \\ \|\boldsymbol{\eta}_{if}^3\|\sigma_{if}^3 \end{pmatrix} \\
\iff &\begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\|\sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\|\sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\|\sigma_{if}^2 \\ \|\boldsymbol{\eta}_{if}^3\|\sigma_{if}^3 \end{pmatrix} = \frac{1}{\tau} \begin{pmatrix} 2 & 0 & 0 & 0 \\ -2 & 2 & 0 & 0 \\ 0 & -2 & 6 & 0 \\ 0 & 0 & -2 & 2 \end{pmatrix} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \\ A_{if}^4 \end{pmatrix}.
\end{aligned} \tag{5.5}$$

Therefore, normal speed σ_{if}^s at stage s of facet f of cell C_i must be updated as follows in the Runge-Kutta process:

$$\begin{aligned}
\sigma_{if}^0 &= \frac{2A_{if}^1}{\tau||\boldsymbol{\eta}_{if}^0||}, \\
\sigma_{if}^1 &= \frac{-2A_{if}^1 + 2A_{if}^2}{\tau||\boldsymbol{\eta}_{if}^1||}, \\
\sigma_{if}^2 &= \frac{-2A_{if}^2 + 6A_{if}^3}{\tau||\boldsymbol{\eta}_{if}^2||}, \\
\sigma_{if}^3 &= \frac{-2A_{if}^3 + 2A_{if}^4}{\tau||\boldsymbol{\eta}_{if}^3||}.
\end{aligned}$$

Practical computation of the A_{if}^s is detailed in Section 5.3.5.

5.3.4.2 Other schemes

RKSSP(3, 3)**.**

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

For each Finite Volume facet f and each Finite Volume cell C_i, we have:

$$\begin{aligned} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \end{pmatrix} &= \tau \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\| \sigma_i^{k,0} \\ \|\boldsymbol{\eta}_{if}^1\| \sigma_i^{k,1} \\ \|\boldsymbol{\eta}_{if}^2\| \sigma_i^{k,2} \end{pmatrix} \\ \iff \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\| \sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\| \sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\| \sigma_{if}^2 \end{pmatrix} &= \tfrac{1}{\tau} \begin{pmatrix} 1 & 0 & 0 \\ -1 & 4 & 0 \\ 0 & -1 & \frac{3}{2} \end{pmatrix} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \end{pmatrix}. \end{aligned} \tag{5.6}$$

RKSSP(2, 2)**.**

Butcher representation	Shu-Osher representation	$t^s = t^n + c_s\,\tau$
$\mathbf{Y_i}^0 = \mathbf{Y_i}^n$		
$\mathbf{Y_i}^1 = \mathbf{Y_i}^0 + \tau\,\mathbf{F}_i^0$		$c_1 = 1$
$\mathbf{Y_i}^2 = \mathbf{Y_i}^0 + \frac{\tau}{2}\left(\mathbf{F}_i^0 + \mathbf{F}_i^1\right)$	$\mathbf{Y_i}^2 = \frac{1}{2}\mathbf{Y_i}^0 + \frac{1}{2}\mathbf{Y_i}^1 + \frac{\tau}{2}\mathbf{F}_i^1$	$c_2 = 1$

For each Finite Volume facet f and each Finite Volume cell C_i, we have:

$$\begin{aligned} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \end{pmatrix} &= \tau \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \|\mathbf{n}_{if}^0\| \sigma_{if}^0 \\ \|\mathbf{n}_{if}^1\| \sigma_{if}^1 \end{pmatrix} \\ \iff \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\| \sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\| \sigma_{if}^1 \end{pmatrix} &= \frac{1}{\tau} \begin{pmatrix} 1 & 0 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \end{pmatrix}. \end{aligned} \tag{5.7}$$

RKSSP(3, 2)

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

For each Finite Volume facet f and each Finite Volume cell C_i, we have:

$$\begin{aligned} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \end{pmatrix} &= \tau \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix} \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\| \sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\| \sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\| \sigma_{if}^2 \end{pmatrix} \\ \iff \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\| \sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\| \sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\| \sigma_{if}^2 \end{pmatrix} &= \frac{1}{\tau} \begin{pmatrix} 2 & 0 & 0 \\ -2 & 2 & 0 \\ 0 & -2 & 3 \end{pmatrix} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \end{pmatrix}. \end{aligned} \tag{5.8}$$

RKSSP(4, 2)

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

For each Finite Volume facet f and each Finite Volume cell C_i, we have:

$$\begin{aligned} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \\ A_{if}^4 \end{pmatrix} &= \tau \begin{pmatrix} \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\| \sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\| \sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\| \sigma_{if}^2 \\ \|\boldsymbol{\eta}_{if}^3\| \sigma_{if}^3 \end{pmatrix} \\ \iff \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\| \sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\| \sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\| \sigma_{if}^2 \\ \|\boldsymbol{\eta}_{if}^3\| \sigma_{if}^3 \end{pmatrix} &= \frac{1}{\tau} \begin{pmatrix} 3 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 0 & -3 & 3 & 0 \\ 0 & 0 & -3 & 4 \end{pmatrix} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \\ A_{if}^4 \end{pmatrix}. \end{aligned} \tag{5.9}$$

RKSSP(5, 2)

Butcher representation	Shu-Osher representation	$t^s = t^n + c_s\,\tau$
$\mathbf{Y_i}^0 = \mathbf{Y_i}^n$		
$\mathbf{Y_i}^1 = \mathbf{Y_i}^0 + \frac{\tau}{4}\mathbf{F}_i^0$		$c_1 = \frac{1}{4}$
$\mathbf{Y_i}^2 = \mathbf{Y_i}^0 + \frac{\tau}{4}\left(\mathbf{F}_i^0 + \mathbf{F}_i^1\right)$	$\mathbf{Y_i}^2 = \mathbf{Y_i}^1 + \frac{\tau}{4}\mathbf{F}_i^1$	$c_2 = \frac{1}{2}$
$\mathbf{Y_i}^3 = \mathbf{Y_i}^0 + \frac{\tau}{4}\left(\mathbf{F}_i^0 + \mathbf{F}_i^1 + \mathbf{F}_i^2\right)$	$\mathbf{Y_i}^3 = \mathbf{Y_i}^2 + \frac{\tau}{4}\mathbf{F}_i^2$	$c_3 = \frac{3}{4}$
$\mathbf{Y_i}^4 = \mathbf{Y_i}^0 + \frac{\tau}{4}\left(\mathbf{F}_i^0 + \mathbf{F}_i^1 + \mathbf{F}_i^2 + \mathbf{F}_i^3\right)$	$\mathbf{Y_i}^4 = \mathbf{Y_i}^3 + \frac{\tau}{4}\mathbf{F}_i^3$	$c_4 = 1$
$\mathbf{Y_i}^5 = \mathbf{Y_i}^0 + \frac{\tau}{5}\left(\mathbf{F}_i^0 + \mathbf{F}_i^1 + \mathbf{F}_i^2 + \mathbf{F}_i^3 + \mathbf{F}_i^4\right)$	$\mathbf{Y_i}^5 = \frac{1}{5}\mathbf{Y_i}^0 + \frac{4}{5}\mathbf{Y_i}^4 + \frac{\tau}{5}\mathbf{F}_i^4$	$c_5 = 1$

For each Finite Volume facet f and each Finite Volume cell C_i, we have:

$$\begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \\ A_{if}^4 \\ A_{if}^5 \end{pmatrix} = \tau \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{pmatrix} \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\|\sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\|\sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\|\sigma_{if}^2 \\ \|\boldsymbol{\eta}_{if}^3\|\sigma_{if}^3 \\ \|\boldsymbol{\eta}_{if}^4\|\sigma_{if}^4 \end{pmatrix}$$

$$\iff \begin{pmatrix} \|\boldsymbol{\eta}_{if}^0\|\sigma_{if}^0 \\ \|\boldsymbol{\eta}_{if}^1\|\sigma_{if}^1 \\ \|\boldsymbol{\eta}_{if}^2\|\sigma_{if}^2 \\ \|\boldsymbol{\eta}_{if}^3\|\sigma_{if}^3 \\ \|\boldsymbol{\eta}_{if}^4\|\sigma_{if}^4 \end{pmatrix} = \frac{1}{\tau} \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 0 & -4 & 4 & 0 & 0 \\ 0 & 0 & -4 & 4 & 0 \\ 0 & 0 & 0 & -4 & 5 \end{pmatrix} \begin{pmatrix} A_{if}^1 \\ A_{if}^2 \\ A_{if}^3 \\ A_{if}^4 \\ A_{if}^5 \end{pmatrix}. \tag{5.10}$$

5.3.5 Practical computation of the areas swept

We assume that facet f of cell C_i is associated with edge $\mathbf{e}_{ij}$ of element $K = (P_i, P_j, P_k, P_l)$. G is the center of gravity of K and $\boldsymbol{w}_G$ is the mesh velocity at G. The algebraic area swept by cells interface between the initial and the current Runge-Kutta stage s are computed using the formula given in [Nkonga 1993].
In the sequel, we note $(\boldsymbol{w}_G)_{if}^s$ the velocity of the gravity center of the considered facet and $\tilde{\boldsymbol{\eta}}_{if}^s$ a pseudo non-normalized normal used to compute the area swept by this facet. **This pseudo normal should not be mistaken with the facet (bi-segment) normal itself.** It is only define for the computation of swept areas and plays no other role.

2D case. Let us assume the considered interface f is associated with edge $\mathbf{e}_{ij}$ and belongs to triangle $K = (P_i, P_j, P_k)$. In two dimensions,

$$A_{if}^s \;=\; c_s\,\tau\,(\boldsymbol{w}_G)_{if}^s \cdot \tilde{\boldsymbol{\eta}}_{if}^s\,,$$

$$\text{with} \quad \left\{ \begin{array}{l} \tilde{\boldsymbol{\eta}}_{i,f}^s \;=\; \dfrac{1}{2}\left(\boldsymbol{\eta}_{if}^0 + \boldsymbol{\eta}_{if}^s\right) \;=\; \dfrac{1}{12}\begin{pmatrix} 2\,\left(y_k^0 + y_k^s\right) - \left(y_j^0 + y_j^s\right) - \left(y_i^0 + y_i^s\right) \\ -2\,\left(x_k^0 + x_k^s\right) + \left(x_j^0 + x_j^s\right) + \left(x_i^0 + x_i^s\right) \end{pmatrix}, \\ \\ (\boldsymbol{w}_G)_{if}^s \;=\; 5\boldsymbol{w}_i^s \;+\; 5\boldsymbol{w}_j^s \;+\; 2\boldsymbol{w}_k^s \;=\; \dfrac{1}{12\,c_s\,\tau}\begin{pmatrix} 5\,\left(x_i^s - x_i^0\right) + 5\,\left(x_j^s - x_j^0\right) + 2\,\left(x_k^s - x_k^0\right) \\ 5\,\left(y_i^s - y_i^0\right) + 5\,\left(y_j^s - y_j^0\right) + 2\,\left(y_k^s - y_k^0\right) \end{pmatrix}. \end{array} \right.$$

3D case. Let us assume the considered facet f is associated with edge $\mathbf{e}_{ij}$ and belongs to tetrahedron $K = (P_i, P_j, P_k, P_l)$. In the following, if G_k denotes the gravity center of face $F_k = (P_i, P_j, P_l)$ of tetrahedron K and G_l the gravity center of face $F_l = (P_i, P_j, P_k)$:

$$A_{if}^s \;=\; c_s\,\tau\,(\boldsymbol{w}_G)_{if}^s \cdot \tilde{\boldsymbol{\eta}}_{if}^s$$

$$\text{with} \quad \left\{ \begin{array}{l} \tilde{\boldsymbol{\eta}}_{if}^s = \dfrac{1}{4}\mathbf{GG_k^0} \wedge \mathbf{GG_l^s} + \dfrac{1}{4}\mathbf{GG_k^s} \wedge \mathbf{GG_l^0} + \dfrac{1}{2}\mathbf{GG_k^0} \wedge \mathbf{GG_l^0} + \dfrac{1}{2}\mathbf{GG_k^s} \wedge \mathbf{GG_l^s}\,, \\ \\ (\boldsymbol{w}_G)_{if}^s = \dfrac{1}{36\,c_s\,\tau}\,\left(13\boldsymbol{w}_i^s + 13\boldsymbol{w}_j^s + 5\boldsymbol{w}_k^s + 5\boldsymbol{w}_l^s\right) \\ \\ \quad = \frac{1}{36\,c_s\,\tau}\begin{pmatrix} 13\,\left(x_i^s - x_i^0\right) + 13\,\left(x_j^s - x_j^0\right) + 5\,\left(x_k^s - x_k^0\right) + 5\,\left(x_l^s - x_l^0\right) \\ 13\,\left(y_i^s - y_i^0\right) + 13\,\left(y_j^s - y_j^0\right) + 5\,\left(y_k^s - y_k^0\right) + 5\,\left(y_l^s - y_l^0\right) \end{pmatrix} \end{array} \right.,$$

and we compute:

$$\mathbf{GG_k^0} = \frac{1}{12}\begin{pmatrix} x_i^0 + x_j^0 - 3x_k^0 + x_l^0 \\ y_i^0 + y_j^0 - 3y_k^0 + y_l^0 \\ z_i^0 + z_j^0 - 3z_k^0 + z_l^0 \end{pmatrix}, \quad \mathbf{GG_k^s} = \frac{1}{12}\begin{pmatrix} x_i^s + x_j^s - 3x_k^s + x_l^s \\ y_i^s + y_j^s - 3y_k^s + y_l^s \\ z_i^s + z_j^s - 3z_k^s + z_l^s \end{pmatrix},$$

$$\mathbf{GG_l^0} = \frac{1}{12}\begin{pmatrix} x_i^0 + x_j^0 + x_k^0 - 3x_l^0 \\ y_i^0 + y_j^0 + y_k^0 - 3y_l^0 \\ z_i^0 + z_j^0 + z_k^0 - 3z_l^0 \end{pmatrix} \quad \text{and} \quad \mathbf{GG_l^s} = \frac{1}{12}\begin{pmatrix} x_i^s + x_j^s + x_k^s - 3x_l^s \\ y_i^s + y_j^s + y_k^s - 3y_l^s \\ z_i^s + z_j^s + z_k^s - 3z_l^s \end{pmatrix}.$$

Implementation issues. We can notice that, for these RKSSP schemes, the computation of σ_{if}^s depends on A_{if}^s and A_{if}^{s+1}, which are the areas swept by interface f of C_i between the initial RK stage and the current s and next $s+1$ Runge-Kutta stages, respectively. Therefore, for each element, the areas swept by its associated interfaces at previous Runge-Kutta step must be stored (three values in two dimensions, six values in three dimensions) to compute geometrical parameter σ_{if}^s at next RK stage.

Note that, in accordance with Mavriplis-Yang approach, the cells normals $\boldsymbol{\eta}_{if}^s$ used to compute numerical flux $\boldsymbol{\Phi}(W_i^s, \boldsymbol{\eta}_{if}^s, ||\boldsymbol{\eta}_{if}^s||\sigma_{if}^s)$ between stages s and $s+1$ are those computed when the

mesh is in its configuration at $t^s = t^n + c_s\,\tau$. This is also true for the upwind/downwind elements if high-order spatial schemes (see Section 5.2.2) are used: these elements are those computed on the current Runge-Kutta configuration. Again, they are imposed by the Runge-Kutta scheme.

It is also important to use the Shu-Osher representation of the scheme while practically computing the numerical flux. With this formulation, we just have to store the previous cells interfaces normals while we would have had to store the interfaces normals of several previous stages if we had used the equivalent Butcher representation.

5.4 A new changing-topology ALE formulation

5.4.1 Problematics

Usually, when the topology of the mesh must be changed between two time steps, local or global interpolation procedures are used to get the solution on the new mesh. It is standard practice to use a simple $\mathbf{P}^1$-Lagrangian projection, even if an increasing number of research teams try to investigate and improve the accuracy and conservativity properties of this interpolation step [Alauzet 2010c, Venkatakrishnan 2007, Margolin 2004]. However, the effects of these repeated interpolations are still not well understood, especially when they are performed locally on the fly after each topology change.

Our opinion is that having an ALE scheme compatible with meshing operations would get us rid of this spoiling projection step and would better fit into the global ALE framework. However, as already mentioned, as soon as Fluid-Structure Interaction problems are considered, time advancing strategies seem mandatory because the computational domain is part of the unknown of the problem. In this context, topology changes are not known in advance and are performed on the fly depending on the evolution of the moving mesh. In this context, designing an ALE procedure handling all kinds of topological operations is far from straightforward, for the reasons detailed below.

Data structures: The mesh data structures will be much more complex and dynamic. The number of edges and elements is the same in two dimensions but varies in three dimensions if connectivity changes occur. If edges collapsing and splitting are authorized, the number of vertices also evolves in time. So does the number of cells.

Four dimensional meshes: The generation of four dimensional space-time meshes is required, linking Finite Volume cells at t^n with cells at t^{n+1} (see Figure 5.9). To our knowledge, such type of meshes are actually not easily generated. This also implies that a four-dimensional mesh must be generated at each time step, which means that four-dimensional mesh generator must be especially efficient to maintain reasonable CPU times. Eventually, these space-time meshes must be hybrid. Indeed, to be coherent with the classical ALE framework, the four-dimensional mesh involves cell-based pseudo-prismatic elements (prismatic-elements with twisted space-time interfaces) in regions were the connectivity of the mesh does not change and tetrahedra when connectivity changes occur, see Figure 5.9.

General ALE schemes: A new fully ALE scheme must be written, which handles all kinds of topological changes.

To our knowledge, only a few attempts to do this can be found in the literature, see for instance [Isola 2010, Meriaux 2003, Kucharik 2008, Olivier 2011].

5.4.2 Our approach

In our opinion, such a general approach cannot be considered at the moment as it would imply the rewriting of most of the Finite Volume ALE code and would necessitate the inclusion of a four-dimensional mesh generator inside the solver. As a first step, we therefore chose to simplify the problem following the line described below.

- First, we focus on **changing-topology meshes only involving swaps** (see definition in Section 4.2.2), *i.e.* vertex addition or suppression is not allowed. This choice is due to the powerfulness of this tool and the fact that it is especially appropriate to handle shears and large deformation movements. With this strategy, only the number of tetrahedra and edges varies in three dimensions. The number of vertices remains fixed. In two dimensions, the number of elements does not even change and neither does the number of edges.

- Second, to fit at best the existing Finite Volume framework, we impose that only the geometrical parameters associated with moving/swapped edges are impacted by topology changes. More precisely, the Mavriplis-Yang approach is adopted, which means that the only parameters that are allowed to be corrected due to edge swapping are the edges normal velocities.

- Third, we impose that once a swap has been performed, all the edges touching the swapped edge are set as blocked until the next optimization step. This clearly simplifies the management of data structures, which remains pseudo-static. But above all, it is absolutely necessary to be able to virtually build a space-time mesh between two solver time steps and thus to define an ALE formulation handling edge swapping.
  Note that this is not as restrictive as it may seem at first glance. Indeed, imagine an edge has been blocked due to a previous swap in its neighborhood while one would have wished to swap it. It is generally not a problem as this swap might be done at the next step on a configuration which is generally not too different from the current one.

- Finally, the space-time mesh is generated implicitly, *i.e.* its entities are never stored in practice but they virtually exist through the way our new scheme works.

Only the two-dimensional case has been implemented. Note also that for the sake of clarity, only the case of the RKSSP(1, 1) time scheme is addressed in a first time. However, extension to other RKSSP schemes will be discussed at the end of this section.

5.4.3 Swapping evanescent cell and volume redistribution

Three-dimensional space-time representation of edge swapping. The swap operation is considered as a time continuous process during which some Finite Volume cells interfaces appear and other collapse. The idea is to consider a three-dimensional mesh linking the cells affected by the swap at t^n and those at t^{n+1}, the time being the third dimension. In the classical ALE framework, all the space-time interfaces are quadrangular and generally twisted, see Figure 5.9

(left). If a swap occurs, the space-time mesh in the swap region exhibits four tetrahedral elements in three dimensions, the faces of which are planar and made of triangles, see Figure 5.9 (right). Note that in this case, space-time interfaces are straight, *i.e.* the bi-segments normals keep the same direction while moving along these space-time triangular faces between two time steps.

Evanescent Finite-Volume cell. By cutting the three-dimensional space-time mesh at an intermediate time between t^n and t^{n+1}, for instance $t^{n+\frac{1}{2}}$, we see that an **evanescent cell** is created just after t^n and vanishes at t^{n+1}. Consequently, some fluxes are exchanged between this evanescent cell and its surrounding real cells between the two configurations. The cells present at $t^{n+\frac{1}{2}}$ are depicted in Figure 5.10. In Figure 5.12 (right), we have represented the directions of the purely geometrical fluxes entering and going out of this evanescent cell on a specific configuration.

Main ideas. Following the above considerations, we have to find a way of correcting the ALE geometrical parameters of the involved edges so as to take these new volume exchanges into account. The main idea is to take all the volume exchanges occurring during the swap into

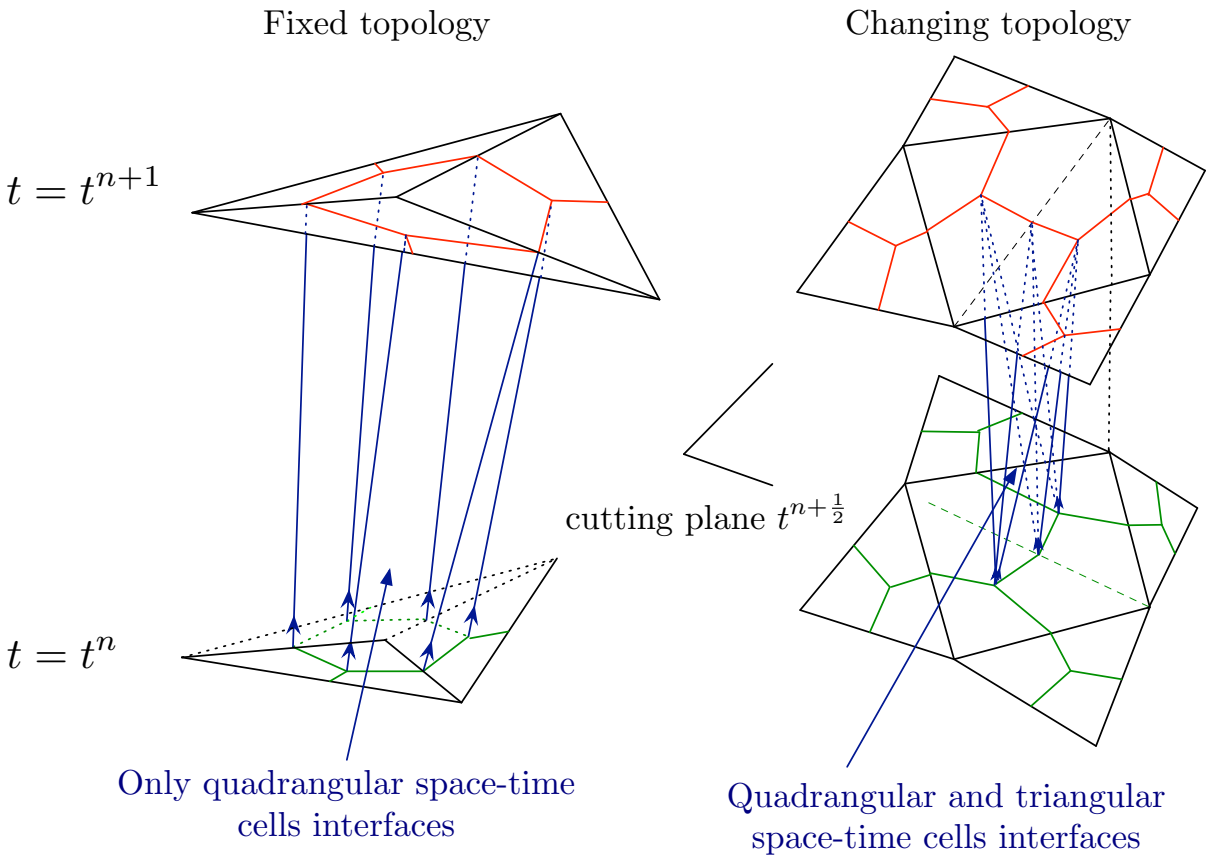


Figure 5.9: Left, space-time interfaces in the fixed-topology case. Finite Volume space-time interfaces are quadrangular and twisted. Right, space-time interpretation of the edge swapping operation and its effect on space-time Finite Volume cells interfaces (in blue). The four tetrahedral space-time elements are depicted in blue lines.

account only by attributing adequate DGCL parameters to the six edges involved, *i.e.* the four bordering edges plus the old and the new edges.
For instance, let us consider two cells connected by one of the bordering edges, for instance C_0 and C_3 which are connected by edge $\mathbf{e}_{03}$. During the swap, these cells can exchange volume either directly through the movement of the associated bordering bi-segment b_{ij} or though the evanescent cell which collects some volume from the four cells and redistribute this volume between the four cells. For instance, in Figure 5.12, arrows in violet, which represent mass exchanges, show that cell C_3 may receive mass from C_0 directly through edge $\mathbf{e}_{03}$ (left figure) but can also get some mass from C_0 through the evanescent cell (right figure). The idea is to take into account the volume exchanged through the evanescent cell by correcting the geometrical parameters associated with edge $\mathbf{e}_{03}$. Corrected parameters will be noted with superscript *. Once these corrections are performed, everything works exactly as they do in the classical ALE framework, *i.e.* these geometrical parameters are used to compute Euler ALE fluxes.

5.4.4 Notations

A crucial remark on relevant parameters. One essential remark made in [Mavriplis 2006] is that, in the classical ALE framework, the interface normal velocity σ_{ij} associated with an edge $\mathbf{e}_{ij}$ never appears alone when computing the mass exchange between two cells but always appears under product $A_{ij} = \tau\sigma_{ij}||\boldsymbol{\eta}_{ij}||$. For instance, when the Roe numerical flux is used, the

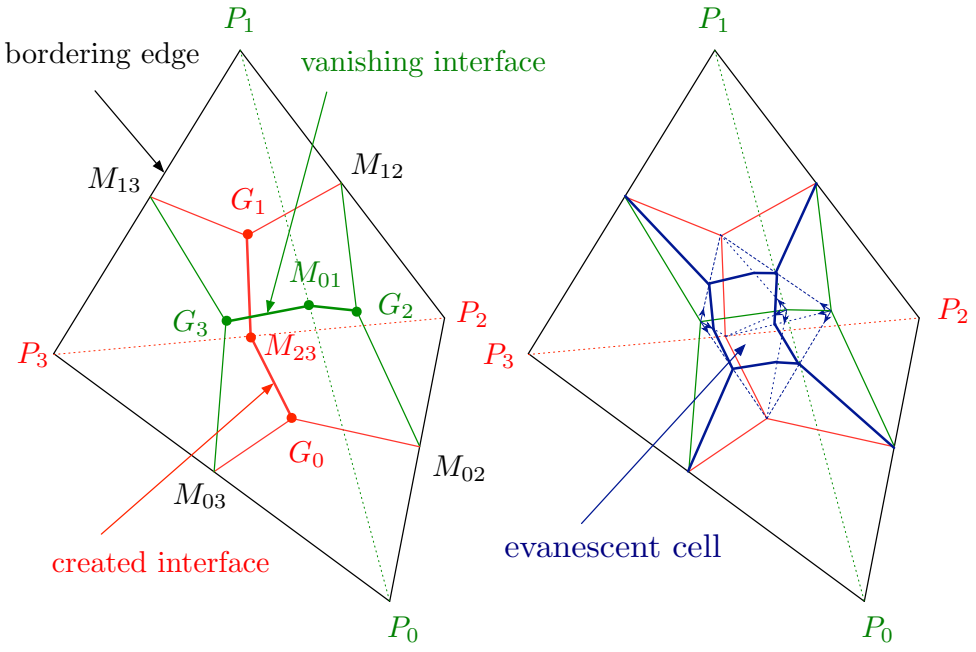


Figure 5.10: Left, effect of edge swapping on Finite Volume cells interfaces. The old edge is represented with a dotted green line and the new one with a dotted red line. The old Finite Volume bi-segments are in green and the new ones in red. Right, representation of the evanescent cell obtained at intermediate configuration $t^{n+\frac{1}{2}}$.

algebraic mass transferred from cell C_i to cell C_j is given by:

$$\begin{aligned}
& \tau||\boldsymbol{\eta}_{ij}||\boldsymbol{\Phi}^{\text{Roe}}(W_i,\, W_j,\, \sigma_{ij},\, \boldsymbol{\eta}_{ij}) \\
= \;& \tau||\boldsymbol{\eta}_{ij}||\frac{F_i+F_j}{2} - \tau\sigma_{ij}||\boldsymbol{\eta}_{ij}||\,\frac{W_i+W_j}{2} \;+\; \tau||\boldsymbol{\eta}_{ij}||\,|\tilde{\mathcal{A}}(W_i,\, W_j) - \sigma_{ij}\mathcal{I}_n|\frac{W_i-W_j}{2} \\
= \;& \tau||\boldsymbol{\eta}_{ij}||\frac{F_i+F_j}{2} - \tau\sigma_{ij}||\boldsymbol{\eta}_{ij}||\,\frac{W_i+W_j}{2} \;+\; \left|\tau||\boldsymbol{\eta}_{ij}||\tilde{\mathcal{A}}(W_i,\, W_j) - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij}\mathcal{I}_n\right|\frac{W_i-W_j}{2} \\
= \;& \tau||\boldsymbol{\eta}_{ij}||\frac{F_i+F_j}{2} - A_{ij}\,\frac{W_i+W_j}{2} \;+\; \left|\tau||\boldsymbol{\eta}_{ij}||\tilde{\mathcal{A}}(W_i,\, W_j) - A_{ij}\mathcal{I}_n\right|\frac{W_i-W_j}{2} \\
= \;& M^{\text{Roe}}(W_i,\, W_j,\, A_{ij},\, \boldsymbol{\eta}_{ij},\, \tau)
\end{aligned}$$

Similarly, if the approximate HLLC Riemann solver is chosen, the algebraic mass exchanged between C_i and C_j during τ is:

$$\tau||\boldsymbol{\eta}_{ij}||\Phi^{\text{HLLC}}(W_i,\, W_j,\, \sigma_{ij},\, \boldsymbol{\eta}_{ij}) = \tau||\boldsymbol{\eta}_{ij}||\begin{cases} \mathbf{F}_i - \sigma_{ij}W_i & \text{if } S_i - \sigma_{ij} > 0 \\ \mathbf{F}_i^* - \sigma_{ij}W_i^* & \text{if } S_i - \sigma_{ij} \le 0 < S_M - \sigma_{ij} \\ \mathbf{F}_j^* - \sigma_{ij}W_j^* & \text{if } S_M - \sigma_{ij} \le 0 \le S_j - \sigma_{ij} \\ \mathbf{F}_j - \sigma_{ij}W_j & \text{if } S_j - \sigma_{ij} < 0 \end{cases}$$

$$= \begin{cases} \tau||\boldsymbol{\eta}_{ij}||\mathbf{F}_i - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij}W_i & \text{if } \tau||\boldsymbol{\eta}_{ij}||S_i - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij} > 0 \\ \tau||\boldsymbol{\eta}_{ij}||\mathbf{F}_i^* - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij}W_i^* & \text{if } \tau||\boldsymbol{\eta}_{ij}||S_i - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij} \le 0 < \tau||\boldsymbol{\eta}_{ij}||S_M - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij} \\ \tau||\boldsymbol{\eta}_{ij}||\mathbf{F}_j^* - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij}W_j^* & \text{if } \tau||\boldsymbol{\eta}_{ij}||S_M - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij} \le 0 \le \tau||\boldsymbol{\eta}_{ij}||S_l - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij} \\ \tau||\boldsymbol{\eta}_{ij}||\mathbf{F}_j - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij}W_j & \text{if } \tau||\boldsymbol{\eta}_{ij}||S_j - \tau||\boldsymbol{\eta}_{ij}||\sigma_{ij} < 0 \end{cases}$$

$$= \begin{cases} \tau||\boldsymbol{\eta}_{ij}||\mathbf{F}_i - A_{ij}W_i & \text{if } \tau||\boldsymbol{\eta}_{ij}||S_i - A_{ij} > 0 \\ \tau||\boldsymbol{\eta}_{ij}||\mathbf{F}_i^* - A_{ij}W_i^* & \text{if } \tau||\boldsymbol{\eta}_{ij}||S_i - A_{ij} \le 0 < \tau||\boldsymbol{\eta}_{ij}||S_M - A_{ij} \\ \tau||\boldsymbol{\eta}_{ij}||\mathbf{F}_j^* - A_{ij}W_j^* & \text{if } \tau||\boldsymbol{\eta}_{ij}||S_M - A_{ij} \le 0 \le \tau||\boldsymbol{\eta}_{ij}||S_l - A_{ij} \\ \tau||\boldsymbol{\eta}_{ij}||\mathbf{F}_j - A_{ij}W_j & \text{if } \tau||\boldsymbol{\eta}_{ij}||S_j - A_{ij} < 0 \end{cases}$$

$$= M^{\text{HLLC}}(W_i,\, W_j,\, A_{ij},\, \boldsymbol{\eta}_{ij},\, \tau)\,.$$

Quantity A_{ij} represents the area swept by interface I_{ij} associated with edge $\mathbf{e}_{ij}$ between t^n and t^{n+1}, and $\tau = t^{n+1} - t^n$ is the solver time step.
This means that the relevant geometrical parameters for ALE computations are not σ_{ij} and $\boldsymbol{\eta}_{ij}$ as it is often suggested, but rather A_{ij} and $\boldsymbol{\eta}_{ij}$.

The second important remark is that the choice of normal $\boldsymbol{\eta}_{ij}$, and especially its direction, has no influence on the value of the mass exchanged between C_i and C_j due to the geometrical deformation of the mesh. Indeed, this purely geometrical exchange writes $M_{ij} = -A_{ij}W$, and is independent of the direction of $\boldsymbol{\eta}_{ij}$. The normal only influences the physical flux computation as $\mathbf{F} = \mathcal{F}\cdot\mathbf{n}$. In other words, geometrical fluxes are not directional, contrary to physical ones. These simple remarks are of crucial importance for the understanding of our new ALE formulation. Actually, our approach comes to consider mass exchanges rather than fluxes.

Notations for the four bordering edges. For each of the four bordering edges $\mathbf{e}_{02}$, $\mathbf{e}_{03}$, $\mathbf{e}_{12}$ and $\mathbf{e}_{23}$, the bi-segment located inside the swap area is noted $f = b_{ij} \in \{1, 2\}$. The normal speed of bi-segment b_{ij} of **bordering** edge $\mathbf{e}_{ij}$ is noted $\sigma_{ij,b_{ij}}$. $\overline{b}_{ij}$ is the other bi-segment associated

with bordering edge $\mathbf{e}_{ij}$, which is located outside the swap area. The interface associated with a bordering edge $\mathbf{e}_{ij}$ is noted I_{ij}. For this interface, geometrical parameters are noted as follows:

- According to Mavriplis-Yang approach, the **non-normalized** normal to bi-segment $f = \left\{b_{ij}, \overline{b}_{ij}\right\}$ associated with interface I_{ij} is the one computed on the initial Runge-Kutta configuration $\boldsymbol{\eta}^n_{ij,f}$,
- The global **non-normalized** normal associated with I_{ij} is: $\boldsymbol{\eta}^n_{ij} = \boldsymbol{\eta}^n_{ij,b_{ij}} + \boldsymbol{\eta}^n_{ij,\overline{b}_{ij}}$,
- The area swept by bordering bi-segments $f = \{b_{ij}, \overline{b}_{ij}\}$ associated with interface I_{ij} are noted $A_{ij,f}$, with $(i,\, j) \in \{(0,\, 2),\, (0,\, 3),\, (1,\, 2),\, (1,\, 3)\}$,
- The total area swept by interface I_{ij} is: $A_{ij} = A_{ij,b_{ij}} + A_{ij,\overline{b}_{ij}}$. If σ_{ij} is the DGCL normal velocity associated with I_{ij} in the fixed-topology ALE framework, we have, for RKSSP(1, 1):

$$A_{ij} = \tau\sigma_{ij}||\boldsymbol{\eta}^n_{ij}||\,.$$

Notations inside the mixing area. Two interfaces, noted I_0 and I_1 are now associated with old green edge $\mathbf{e}_{01}$ and two interfaces I_2 and I_3 with new red edge $\mathbf{e}_{23}$, instead of just one. Each of these interfaces has its own ALE parameters that will be used to compute the ALE flux crossing them. We also define new parameters for the four interfaces of this evanescent cell, each interface being made of two bi-segments $f = 1$ and $f = 2$, see Figure 5.11.

- The area swept by bi-segment $f = \{1, 2\}$ associated with interface I_i is noted $A_{i,f}$, with $i \in [\![0, 3]\!]$,
- The total area swept by interface I_i is: $A_i = A_{i,1} + A_{i,2}$,
- The **non-normalized** normal to bi-segment $f = \{1, 2\}$ associated with interface I_i is noted $\boldsymbol{\eta}_{i,f}$,
- The global **non-normalized** normal associated with I_i is : $\boldsymbol{\eta}_i = \boldsymbol{\eta}_{i,1} + \boldsymbol{\eta}_{i,2}$.

Due to the specific geometric configuration of the evanescent cell, see Figure 5.11 (right), the reversed Thales theorem applies and we have the following relations, for all $t \in [t^n,\, t^{n+1}]$:

$$\begin{array}{lll} \boldsymbol{\eta}_{0,1}(t) = -\boldsymbol{\eta}_{1,1}(t)\,, & \boldsymbol{\eta}_{0,2}(t) = -\boldsymbol{\eta}_{1,2}(t)\,, & ||\boldsymbol{\eta}_0(t)|| = ||\boldsymbol{\eta}_1(t)||\,, \\ \boldsymbol{\eta}_{2,1}(t) = -\boldsymbol{\eta}_{3,1}(t)\,, & \boldsymbol{\eta}_{2,2}(t) = -\boldsymbol{\eta}_{3,2}(t)\,, & ||\boldsymbol{\eta}_3(t)|| = ||\boldsymbol{\eta}_2(t)||\,. \end{array} \tag{5.11}$$

5.4.5 Swept areas computations

In two dimensions, we recall (see Section 5.3.5) that the area swept by a moving bi-segment f associated with interface I is computed as:

$$A_{I,f} = \tau\,(\boldsymbol{w}_G)_{I,f} \cdot \tilde{\boldsymbol{\eta}}_{I,f}\,, \quad \text{with} \quad \tilde{\boldsymbol{\eta}}_{I,f} = \frac{1}{2}\left(\boldsymbol{\eta}^n_{I,f} + \boldsymbol{\eta}^{n+1}_{I,f}\right)\,,$$

where $(\boldsymbol{w}_G)_{I,f}$ is the velocity of the center of gravity of bi-segment f associated with interface I. It is important to understand that normals $\tilde{\boldsymbol{\eta}}_{I,f}$ are pseudo-normals which are used only to compute the areas swept by the bi-segments. It must not be mistaken for normals to bi-segments taken at t^n, $\boldsymbol{\eta}^n_{I,f}$, that will be used for the computation of ALE fluxes.

Areas swept by bi-segments inside the mixing area. First, pseudo-normals $\tilde{\boldsymbol{\eta}}_{ij,f}$ associated with the bi-segments located in the mixing area are computed. For interfaces I_0 and I_1, the normals are of null norm at t^{n+1}. Moreover, Relations (5.11) are true in particular for pseudo-normals. Therefore, we get:

$$\tilde{\boldsymbol{\eta}}_{0,1} = \frac{\boldsymbol{\eta}^n_{0,1}}{2}, \quad \tilde{\boldsymbol{\eta}}_{0,2} = \frac{\boldsymbol{\eta}^n_{0,2}}{2}, \quad \tilde{\boldsymbol{\eta}}_{1,1} = \frac{\boldsymbol{\eta}^n_{1,1}}{2} = -\frac{\boldsymbol{\eta}^n_{0,1}}{2}, \quad \text{and} \quad \tilde{\boldsymbol{\eta}}_{1,2} = \frac{\boldsymbol{\eta}^n_{1,2}}{2} = -\frac{\boldsymbol{\eta}^n_{0,2}}{2}.$$

Similarly, for interfaces I_2 and I_3, the normals are of null norm at t^n. Therefore, we get:

$$\tilde{\boldsymbol{\eta}}_{2,1} = \frac{\boldsymbol{\eta}^{n+1}_{2,1}}{2}, \quad \tilde{\boldsymbol{\eta}}_{2,2} = \frac{\boldsymbol{\eta}^{n+1}_{2,2}}{2}, \quad \tilde{\boldsymbol{\eta}}_{3,1} = \frac{\boldsymbol{\eta}^{n+1}_{3,1}}{2} = -\frac{\boldsymbol{\eta}^{n+1}_{2,1}}{2} \quad \text{and} \quad \tilde{\boldsymbol{\eta}}_{3,2} = \frac{\boldsymbol{\eta}^{n+1}_{3,2}}{2} = -\frac{\boldsymbol{\eta}^{n+1}_{2,2}}{2}.$$

In the sequel, we compute the areas $A_{I,f}$ swept by vanishing and appearing bi-segments. The following notations are used:

$$\boldsymbol{w}_i = \frac{\mathbf{x}^{n+1}_{P_i} - \mathbf{x}^n_{P_i}}{\tau} \quad \text{and} \quad \boldsymbol{w}_{P^n \to Q^{n+1}} = \frac{\mathbf{x}^{n+1}_{Q} - \mathbf{x}^n_{P}}{\tau}.$$

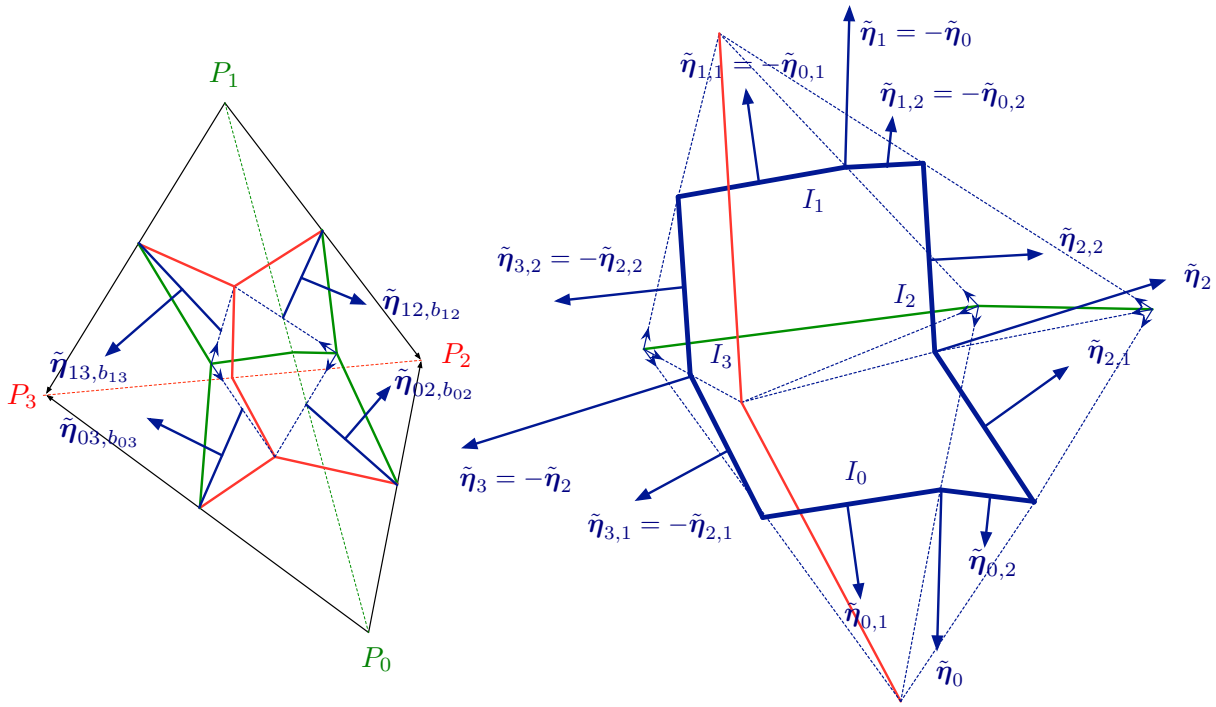


Figure 5.11: Notations. Left, pseudo-normals $\tilde{\boldsymbol{\eta}}_{ij,b_{ij}}$ associated with the four bordering inner bi-segments associated with each bordering edge. Right, pseudo-normals $\tilde{\boldsymbol{\eta}}_{i,f}$ used to compute the areas swept by vanishing or created bi-segments inside the mixing area.

Hence, the four algebraic swept areas by the **four vanishing bi-segments** are computed as follows, see Figure 5.13 (middle):

$$
\begin{aligned}
A_{0,1} &= \tau \boldsymbol{w}_{(G_3+M_{01})/2\to G_0} \cdot \tilde{\boldsymbol{\eta}}_{0,1}\,, \quad \text{with} \quad \boldsymbol{w}_{(G_3+M_{01})/2\to G_0} = \frac{1}{12}\left(-\boldsymbol{w}_0 - 5\boldsymbol{w}_1 + 4\boldsymbol{w}_2 + 2\boldsymbol{w}_3\right)\,,\\
A_{0,2} &= \tau \boldsymbol{w}_{(M_{01}+G_2)/2\to G_0} \cdot \tilde{\boldsymbol{\eta}}_{0,2}\,, \quad \text{with} \quad \boldsymbol{w}_{(M_{01}+G_2)/2\to G_0} = \frac{1}{12}\left(-\boldsymbol{w}_0 - 5\boldsymbol{w}_1 + 2\boldsymbol{w}_2 + 4\boldsymbol{w}_3\right)\,,\\
A_{1,1} &= \tau \boldsymbol{w}_{(G_3+M_{01})/2\to G_1} \cdot \tilde{\boldsymbol{\eta}}_{1,1}\,, \quad \text{with} \quad \boldsymbol{w}_{(G_3+M_{01})/2\to G_1} = \frac{1}{12}\left(-5\boldsymbol{w}_0 - \boldsymbol{w}_1 + 4\boldsymbol{w}_2 + 2\boldsymbol{w}_3\right)\,,\\
A_{1,2} &= \tau \boldsymbol{w}_{(M_{01}+G_2)/2\to G_1} \cdot \tilde{\boldsymbol{\eta}}_{1,2}\,, \quad \text{with} \quad \boldsymbol{w}_{(M_{01}+G_2)/2\to G_1} = \frac{1}{12}\left(-5\boldsymbol{w}_0 - \boldsymbol{w}_1 + 2\boldsymbol{w}_2 + 4\boldsymbol{w}_3\right)\,.
\end{aligned}
\tag{5.12}
$$

Pseudo-normals are computed using the coordinates of vertices P_0, P_1, P_2 and P_3:

$$
\tilde{\boldsymbol{\eta}}_{0,1} = \frac{\boldsymbol{\eta}^n_{0,1}}{2} = \frac{1}{2}\begin{pmatrix} 2y_3^n - y_0^n - y_1^n \\ -2x_3^n + x_0^n + x_1^n \end{pmatrix}\,, \quad \tilde{\boldsymbol{\eta}}_{0,2} = \frac{\boldsymbol{\eta}^n_{0,2}}{2} = \frac{1}{2}\begin{pmatrix} 2y_2^n - y_0^n - y_1^n \\ -2x_2^n + x_0^n + x_1^n \end{pmatrix}\,.
$$

The four algebraic areas swept by the **four appearing bi-segments** are computed as follows,

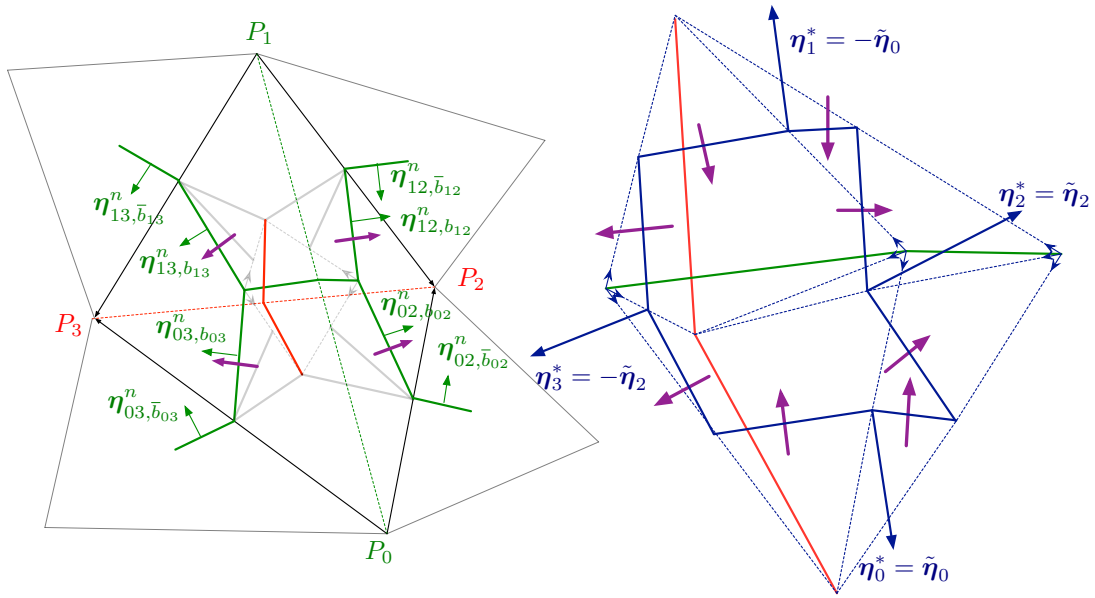


Figure 5.12: Geometrical mass exchanges induced by edge swapping (violet). Left, geometrical mass exchanges through the four bordering edges. According to Mavriplis-Yang approach, the effective normals used to compute the global fluxes through these edges are the ones computed on the initial Runge-Kutta configuration. Right, geometrical mass exchanges between the evanescent cell and its four neighboring Finite-Volume cells. As explained in Section 5.4.6, the effective normals $\boldsymbol{\eta}^*_i$ associated with these evanescent interfaces can be chosen arbitrarily.

see Figure 5.13 (right):

$$
\begin{aligned}
A_{2,1} &= \tau \boldsymbol{w}_{G_2\to(G_0+M_{23})/2}\cdot\tilde{\boldsymbol{\eta}}_{2,1}\,, \quad \text{with} \quad \boldsymbol{w}_{G_2\to(G_0+M_{23})/2} = \frac{1}{12}\left(-2\boldsymbol{w}_0-4\boldsymbol{w}_1+5\boldsymbol{w}_2+\boldsymbol{w}_3\right)\,,\\
A_{2,2} &= \tau \boldsymbol{w}_{G_2\to(M_{23}+G_1)/2}\cdot\tilde{\boldsymbol{\eta}}_{2,2}\,, \quad \text{with} \quad \boldsymbol{w}_{G_2\to(M_{23}+G_1)/2} = \frac{1}{12}\left(-4\boldsymbol{w}_0-2\boldsymbol{w}_1+5\boldsymbol{w}_2+\boldsymbol{w}_3\right)\,,\\
A_{3,1} &= \tau \boldsymbol{w}_{G_3\to(G_0+M_{23})/2}\cdot\tilde{\boldsymbol{\eta}}_{3,1}\,, \quad \text{with} \quad \boldsymbol{w}_{G_3\to(G_0+M_{23})/2} = \frac{1}{12}\left(-2\boldsymbol{w}_0-4\boldsymbol{w}_1+\boldsymbol{w}_2+5\boldsymbol{w}_3\right)\,,\\
A_{3,2} &= \tau \boldsymbol{w}_{G_3\to(M_{23}+G_1)/2}\cdot\tilde{\boldsymbol{\eta}}_{3,2}\,, \quad \text{with} \quad \boldsymbol{w}_{G_3\to(M_{23}+G_1)/2} = \frac{1}{12}\left(-4\boldsymbol{w}_0-2\boldsymbol{w}_1+\boldsymbol{w}_2+5\boldsymbol{w}_3\right)\,.
\end{aligned}
\tag{5.13}
$$

The associated pseudo-normals are:

$$
\tilde{\boldsymbol{\eta}}_{2,1} = \frac{\boldsymbol{\eta}_{2,1}^{n+1}}{2} = \frac{1}{2}\begin{pmatrix} 2y_0^{n+1}-y_2^{n+1}-y_3^{n+1} \\ -2x_0^{n+1}+x_2^{n+1}+x_3^{n+1}\end{pmatrix}, \quad \tilde{\boldsymbol{\eta}}_{2,2} = \frac{\boldsymbol{\eta}_{2,2}^{n+1}}{2} = \frac{1}{2}\begin{pmatrix} 2y_1^{n+1}-y_2^{n+1}-y_3^{n+1} \\ -2x_1^{n+1}+x_2^{n+1}+x_3^{n+1}\end{pmatrix}.
$$

Note that the evanescent nature of the middle cell is contained in formula:

$$
(A_{0,1}+A_{0,2}+A_{1,1}+A_{1,2})+(A_{2,1}+A_{2,2}+A_{3,1}+A_{3,2}) \;=\; 0 \Longleftrightarrow A_0+A_1+A_2+A_3 \;=\; 0\,.
$$

From now on, we note $a^+ = \max(0,\,a)$ (resp. $a^- = \min(0,\,a)$) the positive (resp. negative)

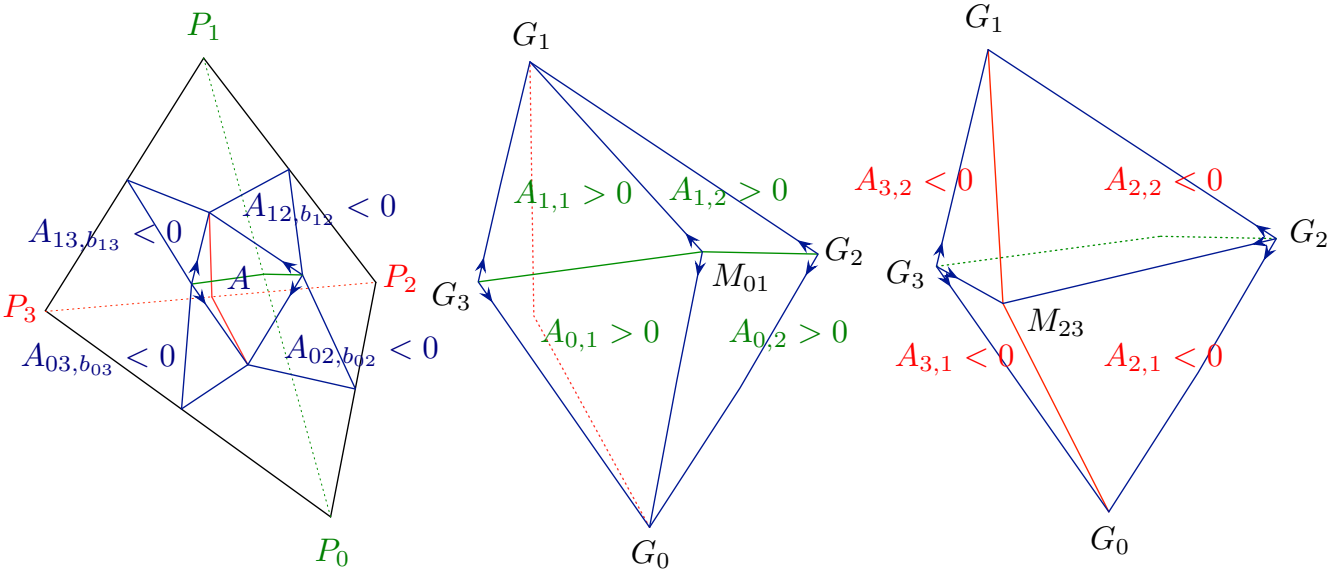


Figure 5.13: Left, areas swept by the four inner bordering bi-segments during the swap operation. Middle, areas swept by vanishing bi-segments associated with evanescent interfaces I_i inside the mixing area during the edge swapping operation. Right, areas swept by appearing bi-segments associated with evanescent interfaces I_i inside the mixing area during the edge swapping operation. In this very specific case, $A_{0,1}$, $A_{0,2}$, $A_{1,1}$ and $A_{1,2}$ are positive while $A_{2,1}$, $A_{2,2}$, $A_{3,1}$ and $A_{3,2}$ are negative.

part of a real value a. Classically, we have:

$$a^+ \geq 0\,, \quad -a^- \geq 0\,, \quad a = a^+ + a^-\,, \quad |a| = \frac{a^+ - a^-}{2}\,, \quad \text{and} \quad a^+ a^- = 0\,. \tag{5.14}$$

Then, using Relations (5.14), we get:

$$\begin{aligned} A_0 + A_1 + A_2 + A_3 \;&=\; 0 \Longleftrightarrow A_0^+ + A_0^- + A_1^+ + A_1^- + A_2^+ + A_2^- + A_3^+ + A_3^- \;=\; 0 \\ \Longleftrightarrow A_0^+ + A_0^- + A_1^+ + A_1^- \;&=\; -\left(A_2^+ + A_2^- + A_3^+ + A_3^-\right) = A\,. \end{aligned} \tag{5.15}$$

Area A is depicted in Figure 5.13. To finish with, we define the total volume eaten and redistributed by the evanescent cell during the swap:

$$M = A_0^+ + A_1^+ + A_2^+ + A_3^+ = -\left(A_0^- + A_1^- + A_2^- + A_3^-\right) \;>\; 0\,.$$

Areas swept bi-segments associated with the four bordering edges. The areas swept by these bordering bi-segments are represented in Figure 5.13.

$$A_{12,b_{12}} = \tau\, \boldsymbol{w}_{(M_{12}^n + G_3^n)/2 \to (M_{12}^{n+1} + G_0^{n+1})/2} \cdot \tilde{\boldsymbol{\eta}}_{12,b_{12}}\,,$$

$$\text{with} \quad \boldsymbol{w}_{(M_{12}^n + G_3^n)/2 \to (M_{12}^{n+1} + G_0^{n+1})/2} = \frac{1}{24\tau}\left(5\boldsymbol{w}_1 + 5\boldsymbol{w}_2 + 2\boldsymbol{w}_{P_0^n \to P_3^{n+1}}\right)$$

$$\text{and} \quad \tilde{\boldsymbol{\eta}}_{12,b_{12}} = \frac{1}{2}\left(\boldsymbol{\eta}_{12,b_{12}}^n + \boldsymbol{\eta}_{12,b_{12}}^{n+1}\right) = \frac{1}{12}\begin{pmatrix} -2\left(y_0^n + y_3^{n+1}\right) + \left(y_1^n + y_1^{n+1}\right) + \left(y_2^n + y_2^{n+1}\right) \\ 2\left(x_0^n + x_3^{n+1}\right) - \left(x_1^n + x_1^{n+1}\right) - \left(x_2^n + x_2^{n+1}\right) \end{pmatrix},$$

$$A_{13,b_{13}} = \tau\, \boldsymbol{w}_{(M_{13}^n + G_2^n)/2 \to (M_{13}^{n+1} + G_0^{n+1})/2} \cdot \tilde{\boldsymbol{\eta}}_{13,b_{13}}\,,$$

$$\text{with} \quad \boldsymbol{w}_{(M_{13}^n + G_2^n)/2 \to (M_{13}^{n+1} + G_0^{n+1})/2} = \frac{1}{24\tau}\left(5\boldsymbol{w}_1 + 5\boldsymbol{w}_3 + 2\boldsymbol{w}_{P_0^n \to P_2^{n+1}}\right)$$

$$\text{and} \quad \tilde{\boldsymbol{\eta}}_{12,b_{12}} = \frac{1}{2}\left(\boldsymbol{\eta}_{13,b_{13}}^n + \boldsymbol{\eta}_{13,b_{13}}^{n+1}\right) = \frac{1}{12}\begin{pmatrix} -2\left(y_0^n + y_2^{n+1}\right) + \left(y_1^n + y_1^{n+1}\right) + \left(y_3^n + y_3^{n+1}\right) \\ 2\left(x_0^n + x_2^{n+1}\right) - \left(x_1^n + x_1^{n+1}\right) - \left(x_3^n + x_3^{n+1}\right) \end{pmatrix},$$

$$A_{02,b_{02}} = \tau\, \boldsymbol{w}_{(M_{02}^n + G_3^n)/2 \to (M_{02}^{n+1} + G_1^{n+1})/2} \cdot \tilde{\boldsymbol{\eta}}_{02,b_{02}}\,,$$

$$\text{with} \quad \boldsymbol{w}_{(M_{02}^n + G_3^n)/2 \to (M_{02}^{n+1} + G_1^{n+1})/2} = \frac{1}{24\tau}\left(5\boldsymbol{w}_0 + 5\boldsymbol{w}_2 + 2\boldsymbol{w}_{P_1^n \to P_3^{n+1}}\right)$$

$$\text{and} \quad \tilde{\boldsymbol{\eta}}_{02,b_{02}} = \frac{1}{2}\left(\boldsymbol{\eta}_{02,b_{02}}^n + \boldsymbol{\eta}_{02,b_{02}}^{n+1}\right) = \frac{1}{12}\begin{pmatrix} -2\left(y_1^n + y_3^{n+1}\right) + \left(y_0^n + y_0^{n+1}\right) + \left(y_2^n + y_2^{n+1}\right) \\ 2\left(x_1^n + x_3^{n+1}\right) - \left(x_0^n + x_0^{n+1}\right) - \left(x_2^n + x_2^{n+1}\right) \end{pmatrix},$$

$$A_{03,b_{03}} = \tau \boldsymbol{w}_{(M_{03}^n + G_2^n)/2 \to (M_{03}^{n+1} + G_1^{n+1})/2} \cdot \tilde{\boldsymbol{\eta}}_{03,b_{03}}\,,$$

$$\text{with} \quad \boldsymbol{w}_{(M_{03}^n + G_2^n)/2 \to (M_{03}^{n+1} + G_1^{n+1})/2} = \frac{1}{24\tau}\left(5\boldsymbol{w}_0 + 5\boldsymbol{w}_3 + 2\boldsymbol{w}_{P_1^n \to P_2^{n+1}}\right)$$

$$\text{and} \quad \tilde{\boldsymbol{\eta}}_{03,b_{03}} = \frac{1}{2}\left(\boldsymbol{\eta}_{03,b_{03}}^n + \boldsymbol{\eta}_{03,b_{03}}^{n+1}\right) = \frac{1}{12}\begin{pmatrix} -2\left(y_1^n + y_2^{n+1}\right) + \left(y_0^n + y_0^{n+1}\right) + \left(y_3^n + y_3^{n+1}\right) \\ 2\left(x_1^n + x_2^{n+1}\right) - \left(x_0^n + x_0^{n+1}\right) - \left(x_3^n + x_3^{n+1}\right) \end{pmatrix}.$$

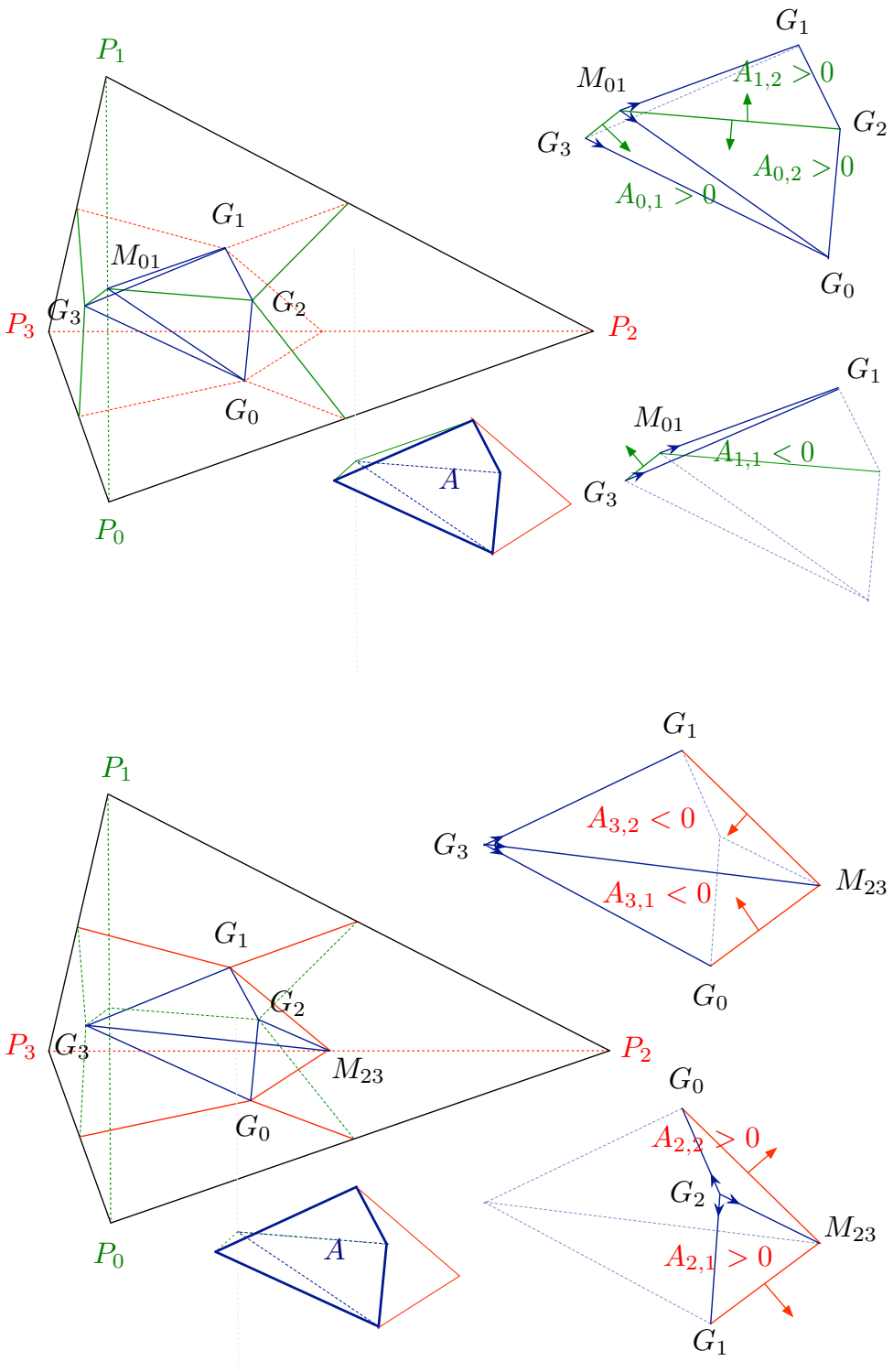


Figure 5.14: A trickier swapping configuration. Top, areas swept by vanishing bi-segments inside the mixing area. Bottom, areas swept by appearing bi-segments inside the mixing area. If blue arrows and bi-segments normals evolve in the same direction (their scalar product is positive), the swept area is positive. On the contrary, if they have opposite evolution directions, it is negative.

Illustrations. Figure 5.13 shows these swept areas and their signs on the configuration depicted in Figure 5.11. By way of example, for the configuration described in Figures 5.12 and 5.13, $A_1 > 0$, $A_0 > 0$, $A_3 < 0$ and $A_2 < 0$. As A_0 and A_1 are of the same sign, all the mass taken from cell C_0 (resp. C_1) is fully redistributed to C_2 and C_3 and nothing is given to C_1 (resp. C_0). In other words, there is no mass exchange between C_0 and C_1. Similarly, as A_2 and A_3 are of the same sign, all the mass given to cell C_2 (resp C_3) comes from cells C_0 and C_1 and nothing comes from C_3 (resp. C_2). There is actually no mass exchange between C_2 and C_3 during the swap. However, this case is a very special one and does not correspond to what will be generally encountered in practice.

Figure 5.14 shows a trickier and more realistic configuration, with its associated swept areas and their respective signs. In this case, $A_0 > 0$, $A_1 > 0$, $A_2 > 0$ and $A_3 < 0$. As $A_2 > 0$ and $A_3 < 0$ are of opposite signs, there is a mass exchange between C_2 and C_3. More specifically, C_3 takes some mass from C_2 during the swap.

5.4.6 The changing-topology ALE schemes

Mavriplis-Yang approach. As recalled in Section 5.4.4, the relevant ALE geometrical parameters associated with interface I_{ij} are not σ_{ij} and $\boldsymbol{\eta}_{ij}$, but rather A_{ij} and $\boldsymbol{\eta}_{ij}$. According to [Yang 2005], the configuration on which flux normal $\boldsymbol{\eta}_{ij}$ must be computed is dictated by the chosen time integration scheme. In our case, the RKSSP(1, 1) is used, which means that the effective normals - the one that must be used to calculate Euler fluxes - are those computed on the initial Runge-Kutta configuration. The only quantities that can be adjusted to take the mass exchanges through the evanescent cell into account are swept areas A_{ij}.
The idea is thus to correct geometrical parameters A_{ij} associated with the six involved edges -the four bordering edges plus the old swapped edge and the new one - to take the volume exchanges though the evanescent cell into account. To this aim, a detailed mass balance analysis is performed. The corrected relevant DGCL geometrical parameter is noted A^*_{ij}.

Volume exchanges through the evanescent cell. M represents the total volume taken from the four cells by the evanescent cell during the swap. Thus, A_j^+/M represents the fraction of this total collected volume which has been stolen from cell C_j. Indeed, if A_j is positive, then cell C_j is losing volume because the geometrical flux between cell C_j and the evanescent cell writes $-A_j^+ W < 0$.
M is also the total volume redistributed by the evanescent cell to the four cells during the swap. Thus A_j^-/M represents the fraction of this total redistributed volume which is given to cell C_j. Indeed, if A_j is negative, then cell C_j is gaining volume from the evanescent cell since the geometrical flux between cell C_j and the evanescent cell writes $-A_j^- W > 0$.

Then, the algebraic volume A_{ij}^{mix} exchanged between cells C_i and C_j through the evanescent

cell, see Figure 5.12 (right), is equal to:

$$A_{ij}^{mix} = \underbrace{\frac{A_j^+}{M}}_{\substack{\text{fraction of volume}\\ \text{taken from } C_j\\ \text{w.r.t total volume}\\ \text{collected by}\\ \text{evanescent cell}}} \underbrace{A_i^-}_{\substack{\text{which is}\\ \text{recuperated by } C_i}} - \underbrace{\frac{A_i^+}{M}}_{\substack{\text{fraction of volume}\\ \text{taken from } C_i\\ \text{w.r.t total volume}\\ \text{collected by}\\ \text{evanescent cell}}} \underbrace{A_j^-}_{\substack{\text{which is}\\ \text{recuperated by } C_j}} .$$

A_{ij}^{mix} is negative if C_i gives volume to C_j and positive otherwise.

Volume exchanges through bordering edges. For the four bordering edges, the volume exchanges due to the displacement of the bordering bi-segments, see Figure 5.12 (left), must also be taken into account. It is equal to:

$$A_{ij}^{border} = A_{ij,b_{ij}} + A_{ij,\bar{b}_{ij}} = A_{ij} .$$

Global volume exchanges for the six involved edges. We deduce that the global volume exchanged by two cells C_i and C_j writes:

$$A_{ij}^* \;=\; A_{ij}^{mix} + A_{ij}^{border} \;=\; A_{ij} + \frac{A_j^+}{M} A_i^- - \frac{A_i^+}{M} A_j^- .$$

As edges $\mathbf{e}_{01}$ and $\mathbf{e}_{23}$ are not bordering edges, they are not associated with any bordering bi-segment and $A_{01} = A_{23} = 0$. Finally, the corrected ALE parameters $\left(A_{ij}^*, \boldsymbol{\eta}_{ij}^n\right)$ associated with the six edges are given by:

$$\begin{aligned}
\text{Edge } \mathbf{e}_{02}: &\quad \left(A_{02}^* = A_{02} + \frac{(A_2^+ A_0^- - A_2^- A_0^+)}{M}, \quad \boldsymbol{\eta}_{02}^n\right),\\
\text{Edge } \mathbf{e}_{03}: &\quad \left(A_{03}^* = A_{03} + \frac{(A_3^+ A_0^- - A_3^- A_0^+)}{M}, \quad \boldsymbol{\eta}_{03}^n\right),\\
\text{Edge } \mathbf{e}_{12}: &\quad \left(A_{12}^* = A_{12} + \frac{(A_2^+ A_1^- - A_2^- A_1^+)}{M}, \quad \boldsymbol{\eta}_{12}^n\right),\\
\text{Edge } \mathbf{e}_{13}: &\quad \left(A_{13}^* = A_{13} + \frac{(A_3^+ A_1^- - A_3^- A_1^+)}{M}, \quad \boldsymbol{\eta}_{13}^n\right),\\
\text{Edge } \mathbf{e}_{01}: &\quad \left(A_{01}^* = \frac{(A_1^+ A_0^- - A_1^- A_0^+)}{M}, \quad \boldsymbol{\eta}_{01}^n\right),\\
\text{Edge } \mathbf{e}_{23}: &\quad \left(A_{23}^* = \frac{(A_3^+ A_2^- - A_3^- A_2^+)}{M}, \quad \boldsymbol{\eta}_{23}^n\right),
\end{aligned} \tag{5.16}$$

and we have noted:

$$M \;=\; A_0^+ + A_1^+ + A_2^+ + A_3^+ \;=\; -\left(A_0^- + A_1^- + A_2^- + A_3^-\right) > 0\,.$$

Implementation issues. The integration of this new scheme inside an existing ALE Finite Volume solver is relatively easy. In two dimensions, only the edges structure has to be modified. Practically speaking, the old and the new edges coexist in the mesh between t^n and t^{n+1}. Some additional space is added at the end of the edges structure to handle the coexistence of old and new edges due to edge swappings performed during the optimization phase. In two dimensions, it is sufficient to allocate 20% more space than what is needed to store the initial number of edges. When a swap occurs, the new edge is added at the end of the table.

Then, corrected ALE geometric parameters are attributed to the six edges involved during the swap. Even if, as already mentioned, σ_{ij} and $\boldsymbol{\eta}_{ij}$ are not the most relevant parameters to write the changing-topology ALE scheme, it may be useful to define these quantities for the six edges in order to fit existing ALE Finite Volume solvers and to avoid adding a new field to the edge structure. For the four bordering edges $\mathbf{e}_{02}$, $\mathbf{e}_{03}$, $\mathbf{e}_{12}$ and $\mathbf{e}_{13}$, this is easily done by defining a corrected normal velocity σ_{ij}^* as:

$$\sigma_{ij}^* = \frac{A_{ij}^*}{\tau||\boldsymbol{\eta}_{ij}^n||}\,.$$

There is also no problem for edge $\mathbf{e}_{01}$:

$$\sigma_{01}^* = \frac{A_{01}^*}{\tau||\boldsymbol{\eta}_{01}^n||}\,.$$

On the contrary, this definition seems problematic when considering edge $\mathbf{e}_{23}$. Indeed, in this case, $||\boldsymbol{\eta}_{23}^n||$ is zero at time t^n because edge $\mathbf{e}_{23}$ does not exist at this time. Defining σ_{23} as described above is impossible as it would imply a division by 0.
However, as explained in Section 5.4.4, the geometrical flux through interface I_{ij} is not directly dependent of the choice of the normal $\boldsymbol{\eta}_{ij}$ used in the approximate Riemannian solver. This geometrical flux depends only on the solution state W^n and on A_{ij}, which in turn depends only on $\boldsymbol{\eta}_{ij}^n$ and $\boldsymbol{\eta}_{ij}^{n+1}$. Therefore, the choice of the configuration on which $\boldsymbol{\eta}_{ij}$ is computed is of no importance for the geometrical flux. This means that we have the right to define:

$$\sigma_{23}^* = \frac{A_{23}^*}{\tau||\boldsymbol{\eta}_{23}^n||}\,,$$

even if $||\boldsymbol{\eta}_{23}^n|| = 0$, because this quantity will always be multiplied by $||\boldsymbol{\eta}_{23}^n||$ when computing geometrical mass exchanges.
Note that although the choice of the configuration on which $\boldsymbol{\eta}_{ij}$ is computed is of no importance for geometrical mass exchanges computation, it has a strong influence on the computed physical mass exchanges through I_{23}. Indeed, as $||\boldsymbol{\eta}_{23}^n|| = 0$, the physical flux through I_{23} is null in the case of the RKSSP(1, 1) time scheme.
Finally, it is possible to define pseudo-normal velocities for evanescent interfaces I_0, I_1, I_2 and I_3:

$$\sigma_i = \frac{A_i}{\tau||\boldsymbol{\eta}_i^*||}\,,$$

where $\boldsymbol{\eta}_i^*$ is an arbitrarily chosen normal. Indeed, the choice of this normal has no influence on the scheme because σ_i will always be multiplied by $||\boldsymbol{\eta}_i^*||$ while computing effective mass exchanges. For instance, we can take $\boldsymbol{\eta}_i^* = \tilde{\boldsymbol{\eta}}_i$. This leads to:

$$\sigma_0 = 2\frac{A_0}{\tau||\boldsymbol{\eta}_{01}^n||}, \qquad \sigma_1 = 2\frac{A_1}{\tau||\boldsymbol{\eta}_{01}^n||}$$

$$\sigma_2 = 2\frac{A_2}{\tau||\boldsymbol{\eta}_{23}^{n+1}||} \quad \text{and} \quad \sigma_3 = 2\frac{A_3}{\tau||\boldsymbol{\eta}_{23}^{n+1}||}.$$

With the above definitions, the corrected normal velocities associated with the six edges involved during the edge swapping are:

$$\begin{aligned}
A_{ij}^* &= A_{ij} + \frac{A_j^+}{M}A_i^- - \frac{A_i^+}{M}A_j^- \\
\Rightarrow \quad \sigma_{ij}^* &= \frac{A_{ij}}{\tau||\boldsymbol{\eta}_{ij}^n||} + \frac{A_j^+}{\tau||\boldsymbol{\eta}_{ij}^n||M}A_i^- - \frac{A_i^+}{\tau||\boldsymbol{\eta}_{ij}^n||M}A_j^- \\
&= \frac{\tau\sigma_{ij}||\boldsymbol{\eta}_{ij}^n||}{\tau||\boldsymbol{\eta}_{ij}^n||} + \frac{\tau\sigma_j^+||\boldsymbol{\eta}_j^*||}{\tau||\boldsymbol{\eta}_{ij}^n||M}\left(\tau\sigma_i^-||\boldsymbol{\eta}_i^*||\right) - \frac{\tau\sigma_i^+||\boldsymbol{\eta}_i^*||}{\tau||\boldsymbol{\eta}_{ij}^n||M}\left(\tau\sigma_j^-||\boldsymbol{\eta}_j^*||\right) \\
&= \sigma_{ij} + \left(\sigma_j^+\sigma_i^- - \sigma_j^-\sigma_i^+\right)\frac{\tau||\boldsymbol{\eta}_i^*||\,||\boldsymbol{\eta}_j^*||}{||\boldsymbol{\eta}_{ij}^n||M}.
\end{aligned}$$

Finally, the following parameters are used in the classical ALE Finite Volume solver:

$$\begin{aligned}
\text{Edge } \mathbf{e}_{02}: &\quad \left(\sigma_{02}^* = \sigma_{02} + \left(\sigma_2^+\sigma_0^- - \sigma_2^-\sigma_0^+\right)\frac{||\boldsymbol{\eta}_0^*||||\boldsymbol{\eta}_2^*||}{||\boldsymbol{\eta}_{02}^n||\dot{M}}, \quad \boldsymbol{\eta}_{02}^n\right), \\
\text{Edge } \mathbf{e}_{03}: &\quad \left(\sigma_{03}^* = \sigma_{03} + \left(\sigma_3^+\sigma_0^- - \sigma_3^-\sigma_0^+\right)\frac{||\boldsymbol{\eta}_0^*||||\boldsymbol{\eta}_3^*||}{||\boldsymbol{\eta}_{03}^n||\dot{M}}, \quad \boldsymbol{\eta}_{03}^n\right), \\
\text{Edge } \mathbf{e}_{12}: &\quad \left(\sigma_{12}^* = \sigma_{12} + \left(\sigma_2^+\sigma_1^- - \sigma_2^-\sigma_1^+\right)\frac{||\boldsymbol{\eta}_1^*||||\boldsymbol{\eta}_2^*||}{||\boldsymbol{\eta}_{12}^n||\dot{M}}, \quad \boldsymbol{\eta}_{12}^n\right), \\
\text{Edge } \mathbf{e}_{13}: &\quad \left(\sigma_{13}^* = \sigma_{13} + \left(\sigma_3^+\sigma_1^- - \sigma_3^-\sigma_1^+\right)\frac{||\boldsymbol{\eta}_1^*||||\boldsymbol{\eta}_3^*||}{||\boldsymbol{\eta}_{13}^n||\dot{M}}, \quad \boldsymbol{\eta}_{13}^n\right), \\
\text{Edge } \mathbf{e}_{01}: &\quad \left(\sigma_{01}^* = \left(\sigma_1^+\sigma_0^- - \sigma_0^-\sigma_0^+\right)\frac{||\boldsymbol{\eta}_0^*||||\boldsymbol{\eta}_1^*||}{||\boldsymbol{\eta}_{01}^n||\dot{M}}, \quad \boldsymbol{\eta}_{01}^n\right), \\
\text{Edge } \mathbf{e}_{23}: &\quad \left(\sigma_{23}^* = \left(\sigma_3^+\sigma_2^- - \sigma_3^-\sigma_2^+\right)\frac{||\boldsymbol{\eta}_2^*||||\boldsymbol{\eta}_3^*||}{||\boldsymbol{\eta}_{23}^n||\dot{M}}, \quad \boldsymbol{\eta}_{23}^n\right),
\end{aligned} \tag{5.17}$$

and we have noted:

$$\dot{M} \;=\; \frac{M}{\tau} \;=\; \frac{A_0^+ + A_1^+ + A_2^+ + A_3^+}{\tau} \;=\; -\,\frac{A_0^- + A_1^- + A_2^- + A_3^-}{\tau} > 0\,.$$

$\dot{M}$ has the dimension of a sweeping velocity.

Once geometrical parameters have been defined, these edges are treated exactly like the others, *i.e.* a complete (geometrical plus physical) ALE flux is computed across them, with the chosen Riemannian approximate solver. Once the fluxes between t^n and t^{n+1} have been computed, the old edge is removed and replaced by the new one. Note that with this strategy, the extremities of the edges involved in the swap operation can also move during the edge swapping. It will automatically be taken into account by Scheme (5.19).
As explained in Section 5.4.2, once a swap has been performed, all the bordering edges are blocked and cannot be swapped anymore until the next solver time step.

5.4.7 Generalizations

5.4.7.1 Extension to other time schemes

The extension to multi-step integration schemes like the implicit Backward Differentiation Formula schemes seems quiet easy, even if we have not tested it yet.
As regards the extension of this new scheme to ALE DGCL Runge-Kutta schemes of Section 5.3.2, we again follow [Yang 2005], which means that the normal used to compute the solution at Runge-Kutta stage $s+1$ is the one computed on the s^{th} Runge-Kutta configuration. Thus, the following "relevant" ALE DGCL parameters $(A_{ij}^{*,s},\, \boldsymbol{\eta}_{ij}^s)$ are proposed for the six edges at the s^{th} Runge-Kutta stage:

$$\begin{aligned}
\text{Edge } \mathbf{e}_{02}: &\quad \left(A_{02}^{*,s} = A_{02}^s + \frac{(A_2^{+,s}A_0^{-,s} - A_2^{-,s}A_0^{+,s})}{M^s},\quad \boldsymbol{\eta}_{02}^s\right),\\
\text{Edge } \mathbf{e}_{03}: &\quad \left(A_{03}^{*,s} = A_{03}^s + \frac{(A_3^{+,s}A_0^{-,s} - A_3^{-,s}A_0^{+,s})}{M^s},\quad \boldsymbol{\eta}_{03}^s\right),\\
\text{Edge } \mathbf{e}_{12}: &\quad \left(A_{12}^{*,s} = A_{12}^s + \frac{(A_2^{+,s}A_1^{-,s} - A_2^{-,s}A_1^{+,s})}{M^s},\quad \boldsymbol{\eta}_{12}^s\right),\\
\text{Edge } \mathbf{e}_{13}: &\quad \left(A_{13}^{*,s} = A_{13}^s + \frac{(A_3^{+,s}A_1^{-,s} - A_3^{-,s}A_1^{+,s})}{M^s},\quad \boldsymbol{\eta}_{13}^s\right),\\
\text{Edge } \mathbf{e}_{01}: &\quad \left(A_{01}^{*,s} = \frac{\left(A_1^{+,s}A_0^{-,s} - A_1^{-,s}A_0^{+,s}\right)}{M^s},\quad \boldsymbol{\eta}_{01}^s\right),\\
\text{Edge } \mathbf{e}_{23}: &\quad \left(A_{23}^{*,s} = \frac{\left(A_3^{+,s}A_2^{-,s} - A_3^{-,s}A_2^{+,s}\right)}{M^s},\quad \boldsymbol{\eta}_{23}^s\right),
\end{aligned} \tag{5.18}$$

where

$$\begin{cases} A_{ij}^s & \text{is the area swept by } I_{ij} \text{ between the } \textbf{initial} \text{ and the } s^{th} \text{ RK configuration}\,, \\ A_{i}^s & \text{is the area swept by } I_{i} \text{ between the } \textbf{initial} \text{ and the } s^{th} \text{ RK configuration}\,, \\ \boldsymbol{\eta}_{ij}^s & \text{is the non-normalized normal to } I_{ij} \text{ computed on the } s^{th} \text{ RK configuration}\,, \\ M^s & = A_0^{+,s} + A_1^{+,s} + A_2^{+,s} + A_3^{+,s} = -\left(A_0^{-,s} + A_1^{-,s} + A_2^{-,s} + A_3^{-,s}\right) > 0\,. \end{cases}$$

Swept areas computation. Areas swept by the bi-segments between the initial Runge-Kutta configuration at t^n and current one at $t^s = t^n + c_s\tau$ are computed using the following relation:

$$A_{I,f}^s = c_s\tau\,(\boldsymbol{w}_G)_{I,f}^s \cdot \tilde{\boldsymbol{\eta}}_{I,f}^s\,.$$

$(\boldsymbol{w}_G)_{I,f}^s$ is velocity of the center of gravity of the bi-segment f of interface I computed between the initial and the current Runge-Kutta configuration. It is equal to:

$$(\boldsymbol{w}_G)_{I,f}^s = \frac{\mathbf{x}_{G_{I,f}}^s - \mathbf{x}_{G_{I,f}}^0}{c_s\tau} = c_s\frac{\mathbf{x}_{G_{I,f}}^{n+1} - \mathbf{x}_{G_{I,f}}^n}{c_s\tau} = \frac{\mathbf{x}_{G_{I,f}}^{n+1} - \mathbf{x}_{G_{I,f}}^n}{\tau} = (\boldsymbol{w}_G)_{I,f}\,.$$

$\tilde{\boldsymbol{\eta}}_{I,f}^s$ is the pseudo-normal of the considered bi-segment computed between the initial and the end of the current Runge-Kutta configuration. For bordering bi-segments, it is equal to:

$$\tilde{\boldsymbol{\eta}}_{I,f} = \frac{1}{2}\left(\boldsymbol{\eta}_{ij,f}^0 + \boldsymbol{\eta}_{ij,f}^s\right) = \frac{1}{2}\left(\boldsymbol{\eta}_{ij,f}^n + \boldsymbol{\eta}_{ij,f}^n + c_s\left(\boldsymbol{\eta}_{ij,f}^{n+1} - \boldsymbol{\eta}_{ij,f}^n\right)\right) = \frac{1}{2}\left((2-c_s)\boldsymbol{\eta}_{ij,f}^n + c_s\boldsymbol{\eta}_{ij,f}^{n+1}\right)\,.$$

In particular, for vanishing bi-segments, we get:

$$\tilde{\boldsymbol{\eta}}_0^s = -\frac{2-c_s}{2}\boldsymbol{\eta}_{01}^n \quad \text{and} \quad \tilde{\boldsymbol{\eta}}_1^s = \frac{2-c_s}{2}\boldsymbol{\eta}_{01}^n\,,$$

and for appearing bi-segments:

$$\tilde{\boldsymbol{\eta}}_2^s = -\frac{c_s}{2}\boldsymbol{\eta}_{23}^{n+1} \quad \text{and} \quad \tilde{\boldsymbol{\eta}}_3^s = \frac{c_s}{2}\boldsymbol{\eta}_{23}^{n+1}\,.$$

Note that this is consistent with the expressions found for RKSSP(1, 1), for which $c_s = c_1 = 1$.

$(\sigma, \boldsymbol{\eta})$ ***formulation.*** Instead of using "relevant" geometrical parameters $(A_{ij}^{*,s}, \boldsymbol{\eta}_{ij}^s)$ to compute the solution at stage $s+1$, it is possible to use another set of ALE DGCL geometrical parameters

$(\sigma_{ij}^{*,s},\, \boldsymbol{\eta}_{ij}^{s})$ defined as:

$$
\begin{aligned}
\text{Edge } \mathbf{e}_{02}: &\quad \left(\sigma_{02}^{*,s} = \sigma_{02}^{s} + \left(\sigma_2^{+,s}\sigma_0^{-,s} - \sigma_2^{-,s}\sigma_0^{+,s}\right)\frac{||\boldsymbol{\eta}_0^*||||\boldsymbol{\eta}_2^*||}{||\boldsymbol{\eta}_{02}^s||\dot{M}^s}\,, \quad \boldsymbol{\eta}_{02}^s\right),\\
\text{Edge } \mathbf{e}_{03}: &\quad \left(\sigma_{03}^{*,s} = \sigma_{03}^{s} + \left(\sigma_3^{+,s}\sigma_0^{-,s} - \sigma_3^{-,s}\sigma_0^{+,s}\right)\frac{||\boldsymbol{\eta}_0^{*,s}||||\boldsymbol{\eta}_3^{*,s}||}{||\boldsymbol{\eta}_{03}^s||\dot{M}^s}\,, \quad \boldsymbol{\eta}_{03}^s\right),\\
\text{Edge } \mathbf{e}_{12}: &\quad \left(\sigma_{12}^{*,s} = \sigma_{12}^{s} + \left(\sigma_2^{+,s}\sigma_1^{-,s} - \sigma_2^{-,s}\sigma_1^{+,s}\right)\frac{||\boldsymbol{\eta}_1^{*,s}||||\boldsymbol{\eta}_2^{*,s}||}{||\boldsymbol{\eta}_{12}^s||\dot{M}^s}\,, \quad \boldsymbol{\eta}_{12}^s\right),\\
\text{Edge } \mathbf{e}_{13}: &\quad \left(\sigma_{13}^{*,s} = \sigma_{13}^{s} + \left(\sigma_3^{+,s}\sigma_1^{,s} - \sigma_3^{-,s}\sigma_1^{+,s}\right)\frac{||\boldsymbol{\eta}_1^{*}||||\boldsymbol{\eta}_3^{*}||}{||\boldsymbol{\eta}_{13}^s||\dot{M}^s}\,, \quad \boldsymbol{\eta}_{13}^s\right),\\
\text{Edge } \mathbf{e}_{01}: &\quad \left(\sigma_{01}^{*,s} = \left(\sigma_1^{+,s}\sigma_0^{-,s} - \sigma_0^{-,s}\sigma_1^{+,s}\right)\frac{||\boldsymbol{\eta}_0^{*}||||\boldsymbol{\eta}_1^{*,s}||}{||\boldsymbol{\eta}_{01}^s||\dot{M}^s}\,, \quad \boldsymbol{\eta}_{01}^s\right),\\
\text{Edge } \mathbf{e}_{23}: &\quad \left(\sigma_{23}^{*,s} = \left(\sigma_3^{+,s}\sigma_2^{-,s} - \sigma_3^{-,s}\sigma_2^{+,s}\right)\frac{||\boldsymbol{\eta}_2^{*,s}||||\boldsymbol{\eta}_3^{*,s}||}{||\boldsymbol{\eta}_{23}^s||\dot{M}^s}\,, \quad \boldsymbol{\eta}_{23}^s\right),
\end{aligned}
\tag{5.19}
$$

and we have noted:

$$
\dot{M}^s = \frac{M^s}{\tau} = \frac{A_0^{+,s} + A_1^{+,s} + A_2^{+,s} + A_3^{+,s}}{\tau} = -\frac{A_0^{-,s} + A_1^{-,s} + A_2^{-,s} + A_3^{-,s}}{\tau} > 0\,.
$$

$\dot{M}^s$ has the dimension of a sweeping velocity. $\boldsymbol{\eta}_i^{*,s}$ are arbitrarily chosen normals associated with evanescent interfaces I_i, which have no influence on the global flux computation. They are only defined to fit the former ALE framework which uses σ and $\boldsymbol{\eta}$ as geometrical parameters. The following choice can be made:

$$
\boldsymbol{\eta}_0^{*,s} = \tilde{\boldsymbol{\eta}}_0^{*,s} = -\frac{2-c_s}{2}\boldsymbol{\eta}_{01}^n\,, \quad \boldsymbol{\eta}_1^{*,s} = \tilde{\boldsymbol{\eta}}_1^{*,s} = \frac{2-c_s}{2}\boldsymbol{\eta}_{01}^n
$$

$$
\boldsymbol{\eta}_2^{*,s} = \tilde{\boldsymbol{\eta}}_2^{*,s} = -\frac{c_s}{2}\boldsymbol{\eta}_{23}^n \quad \text{and} \quad \boldsymbol{\eta}_3^{*,s} = \tilde{\boldsymbol{\eta}}_3^{*,s} = \frac{c_s}{2}\boldsymbol{\eta}_{23}^n\,.
$$

The definition of the σ_i^s's depends on this choice and is given by:

$$
\sigma_i^s = \frac{A_i^s}{\tau||\boldsymbol{\eta}_i^{*,s}||}\,.
$$

Finally, note that the volume of the evanescent cell at t^s is given by:

$$
|C^{evanes.}|^s = A_0^s + A_1^s + A_2^s + A_3^s\,.
$$

This is coherent with equality $A_0 + A_1 + A_2 + A_3 = 0$ found in the case of the RKSSP(1, 1) time integration because in this case, there is only one stage and $|C^{evanes.}|^1 = |C^{evanes.}|^{n+1} = 0$.

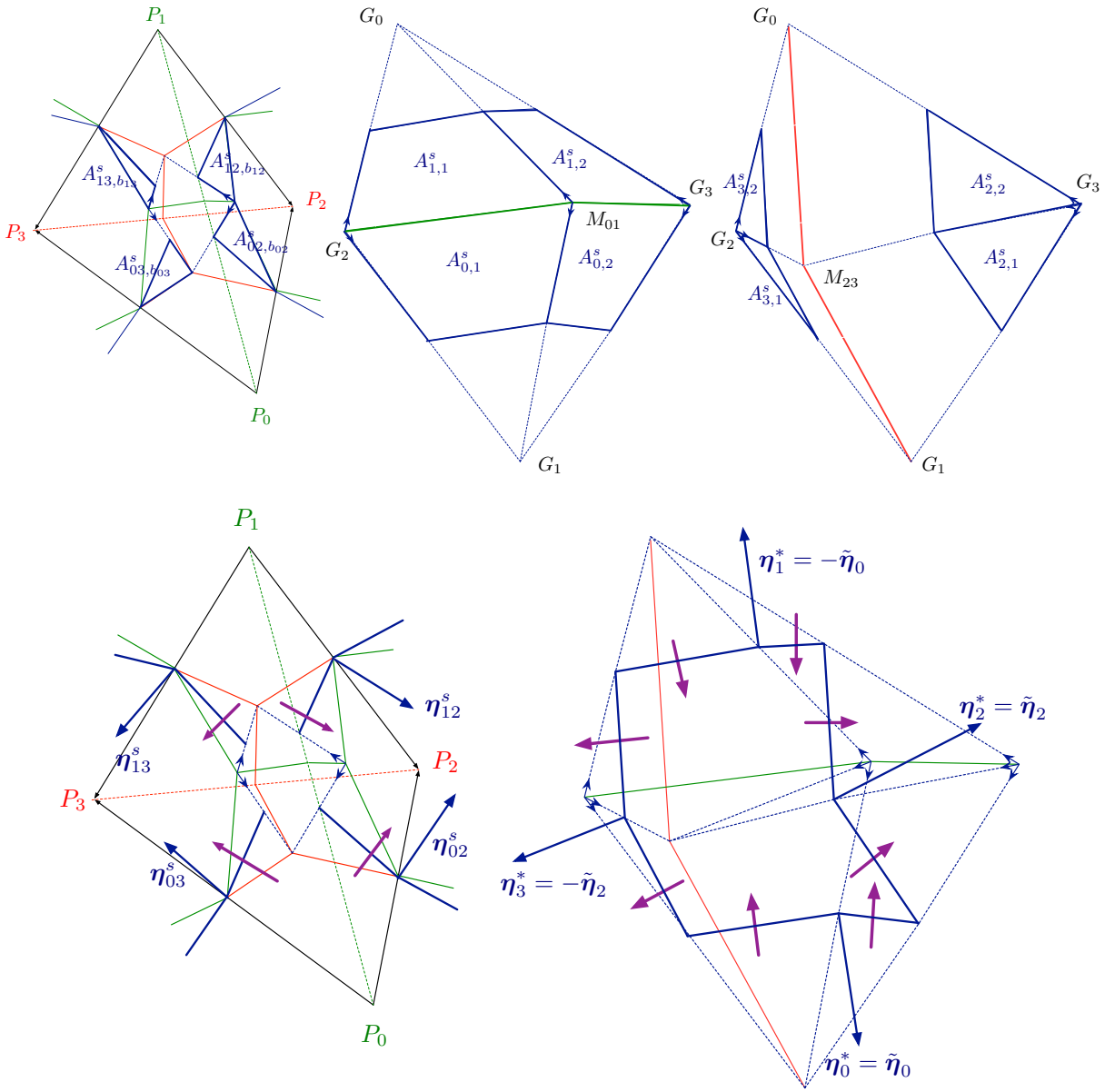


Figure 5.15: Swept areas computations during the swap when a multi-stage Runge-Kutta scheme is used. Top, areas swept by the four inner bordering bi-segments (left), by vanishing bi-segments (middle) and by appearing bi-segments (right) between the initial and the current Runge-Kutta configuration during the swap inside the mixing area. Bottom, geometrical mass exchanges induced by edge swapping (violet) through the four bordering edges (left) and through the evanescent cell (right) between the initial and the current Runge-Kutta configuration.

5.4.7.2 Extension to three dimensions

The generalization of this scheme to three dimensional edge/face swapping is really hard but some promising investigations are currently underway.

5.4.8 Scheme analysis

5.4.8.1 DGCL property

Case RKSSP(1,1). Let us first focus on the case where a RKSSP(1, 1) time integration is used. In this case, the scheme obtained using the geometrical parameters described above is of course DGCL as it exactly computes geometrical mass exchanges. However, we provide here a mathematical demonstration of this property.
To prove that our new scheme is DGCL with respect to the RKSSP(1, 1) time integration scheme, we demonstrate that any constant state solution $W = W_0$ is preserved. Let us consider the edge swapping that transforms edge $\mathbf{e}_{01}$ into $\mathbf{e}_{23}$, see Figure 5.11. If the solution at instant t^n is constant in space and equal to W^n, the integral of the physical mass exchanges around non-boundary cell C_0 is null. So, only the geometrical mass exchanges remain. The mass of cell C_0 after edge swapping at t^{n+1} is:

$$
\begin{aligned}
|C_0|^{n+1}W_0^{n+1} &= |C_0|^n W_0^n + \tau\sigma_{01}^*||\boldsymbol{\eta}_{01}^n||W_1^n + \tau\sigma_{02}^*||\boldsymbol{\eta}_{02}^n||W_2^n + \tau\sigma_{03}^*||\boldsymbol{\eta}_{03}^n||W_3^n \\
&= |C_0|^n W^n + \tau \left(\begin{array}{lll}
 & & \left(\sigma_1^+\sigma_0^- - \sigma_1^-\sigma_0^+\right)\dfrac{||\boldsymbol{\eta}_0^*||\,||\boldsymbol{\eta}_1^*||}{\dot{M}||\boldsymbol{\eta}_{01}^n||}||\boldsymbol{\eta}_{01}^n|| \\
+ \quad \sigma_{02}||\boldsymbol{\eta}_{02}^n|| & + & \left(\sigma_2^+\sigma_0^- - \sigma_2^-\sigma_0^+\right)\dfrac{||\boldsymbol{\eta}_0^*||\,||\boldsymbol{\eta}_2^*||}{\dot{M}||\boldsymbol{\eta}_{02}^n||}||\boldsymbol{\eta}_{02}^n|| \\
+ \quad \sigma_{03}||\boldsymbol{\eta}_{03}^n|| & + & \left(\sigma_3^+\sigma_0^- - \sigma_3^-\sigma_0^+\right)\dfrac{||\boldsymbol{\eta}_0^*||\,||\boldsymbol{\eta}_3^*||}{\dot{M}||\boldsymbol{\eta}_{03}^n||}||\boldsymbol{\eta}_{03}^n||
\end{array} \right) W^n
\end{aligned}
$$

Using the fact that, according to Relations (5.11), $||\boldsymbol{\eta}_0^*|| = ||\boldsymbol{\eta}_1^*||$ and $||\boldsymbol{\eta}_2^*|| = ||\boldsymbol{\eta}_3^*||$, we deduce that:

$$
\begin{aligned}
|C_0|^{n+1}W_0^{n+1} = \Bigg(|\mathcal{C}_0|^n + \tau\sigma_{02}||\boldsymbol{\eta}_{02}^n|| + \tau\sigma_{03}||\boldsymbol{\eta}_{03}^n|| + \tau\sigma_0^- \frac{||\boldsymbol{\eta}_0^*||}{\dot{M}} \left[\left(\sigma_2^+ + \sigma_3^+\right)||\boldsymbol{\eta}_2^*|| + \sigma_1^+||\boldsymbol{\eta}_0^*|| \right] \\
+ \tau\sigma_0^+ \frac{||\boldsymbol{\eta}_0^*||}{\dot{M}} \left[-\left(\sigma_2^- + \sigma_3^-\right)||\boldsymbol{\eta}_2^*|| - \sigma_1^-||\boldsymbol{\eta}_0^*|| \right] \Bigg) W^n \,.
\end{aligned}
$$

Using Relation (5.14), *i.e.* $\sigma_0^+\sigma_0^- = 0$, we can add the terms marked in red in the following expression without changing anything:

$$|C_0|^{n+1}W_0^{n+1} = \Bigg(|\mathcal{C}_0|^n + \tau\sigma_{02}||\boldsymbol{\eta}_{02}^n|| + \tau\sigma_{03}||\boldsymbol{\eta}_{03}^n|| + \tau\sigma_0^- \frac{||\boldsymbol{\eta}_0^*||}{\dot{M}}\Big[\left(\sigma_2^+ + \sigma_3^+\right)||\boldsymbol{\eta}_2^*|| + \left(\sigma_1^+ + \sigma_0^+\right)||\boldsymbol{\eta}_0^*||\Big]$$
$$+\tau\sigma_0^+ \frac{||\boldsymbol{\eta}_0^*||}{\dot{M}}\Big[-\left(\sigma_2^- + \sigma_3^-\right)||\boldsymbol{\eta}_2^*|| - \left(\sigma_1^- + \sigma_0^-\right)||\boldsymbol{\eta}_0^*||\Big]\Bigg) W^n\,.$$

Using Relation (5.15), we then obtain:

$$|C_0|^{n+1}W_0^{n+1} = \Bigg(|\mathcal{C}_0|^n + \tau\sigma_{02}||\boldsymbol{\eta}_{02}^n|| + \tau\sigma_{03}||\boldsymbol{\eta}_{03}^n||$$
$$+\tau\underbrace{\left(\sigma_0^- + \sigma_0^+\right)}_{=\sigma_0}\frac{||\boldsymbol{\eta}_0^*||}{\dot{M}}\underbrace{\Big[\left(\sigma_2^+ + \sigma_3^+\right)||\boldsymbol{\eta}_2^*|| + \left(\sigma_1^+ + \sigma_0^+\right)||\boldsymbol{\eta}_0^*||\Big]}_{=\dot{M}}\Bigg)W^n$$
$$= \Big(|\mathcal{C}_0|^n + \tau\sigma_{02}||\boldsymbol{\eta}_{02}^n|| + \tau\sigma_{03}||\boldsymbol{\eta}_{03}^n|| + \tau\sigma_0||\boldsymbol{\eta}_0^*||\Big)W^n$$
$$= \Big(|\mathcal{C}_0|^n + A_{02} + A_{03} + A_0\Big)W^n \;=\; |C_0|^{n+1}W^n\,.$$

Therefore, $W_0^{n+1} = W^n = W_0^n$. The same demonstration can be done for the four cells. The scheme therefore preserves the constant solution and, as a consequence, is DGCL for RKSSP(1, 1).

Case RKSSP(s,p). The mass of cell C_0 after edge swapping at $t^s = t^n + c_s\tau$ is:

$$|C_0|^s W_0^s = |C_0|^n W_0^n + c_s\tau\sigma_{01}^{*,s}||\boldsymbol{\eta}_{01}^s||W_1^n + c_s\tau\sigma_{02}^{*,s}||\boldsymbol{\eta}_{02}^s||W_2^n + c_s\tau\sigma_{03}^{*,s}||\boldsymbol{\eta}_{03}^s||W_3^n$$
$$= |C_0|^n W^n + c_s\tau\Bigg(\begin{array}{lllll} & & + & \left(\sigma_1^{+,s}\sigma_0^{-,s} - \sigma_1^{-,s}\sigma_0^{+,s}\right)\dfrac{||\boldsymbol{\eta}_0^{*,s}||\,||\boldsymbol{\eta}_1^{*,s}||}{\dot{M}^s||\boldsymbol{\eta}_{01}^s||}||\boldsymbol{\eta}_{01}^s|| \\ + & \sigma_{02}^s||\boldsymbol{\eta}_{02}^s|| & + & \left(\sigma_2^{+,s}\sigma_0^{-,s} - \sigma_2^{-,s}\sigma_0^{+,s}\right)\dfrac{||\boldsymbol{\eta}_0^{*,s}||\,||\boldsymbol{\eta}_2^{*,s}||}{\dot{M}^s||\boldsymbol{\eta}_{02}^s||}||\boldsymbol{\eta}_{02}^s|| \\ + & \sigma_{03}^s||\boldsymbol{\eta}_{03}^s|| & + & \left(\sigma_3^{+,s}\sigma_0^{-,s} - \sigma_3^{-,s}\sigma_0^{+,s}\right)\dfrac{||\boldsymbol{\eta}_0^{*,s}||\,||\boldsymbol{\eta}_3^{*,s}||}{\dot{M}^s||\boldsymbol{\eta}_{03}^s||}||\boldsymbol{\eta}_{03}^s|| \end{array}\Bigg)W^n$$

Using Relations (5.11) at $t = t^s/2$, we have in particular $||\boldsymbol{\eta}_0^{*,s}|| = ||\boldsymbol{\eta}_1^{*,s}||$ and $||\boldsymbol{\eta}_2^{*,s}|| = ||\boldsymbol{\eta}_3^{*,s}||$ and we deduce that:

$$|C_0|^s W_0^s = \Bigg(|\mathcal{C}_0|^n + c_s\tau\sigma_{02}^s||\boldsymbol{\eta}_{02}^s|| + c_s\tau\sigma_{03}^s||\boldsymbol{\eta}_{03}^s|| + c_s\tau\sigma_0^{-,s}\frac{||\boldsymbol{\eta}_0^{*,s}||}{\dot{M}^s}\left[\left(\sigma_2^{+,s}+\sigma_3^{+,s}\right)||\boldsymbol{\eta}_2^{*,s}|| + \sigma_1^{+,s}||\boldsymbol{\eta}_0^{*,s}||\right]$$
$$+ c_s\tau\sigma_0^{+,s}\frac{||\boldsymbol{\eta}_0^{*,s}||}{\dot{M}^s}\left[-\left(\sigma_2^{-,s}+\sigma_3^{-,s}\right)||\boldsymbol{\eta}_2^{*,s}|| - \sigma_1^{-,s}||\boldsymbol{\eta}_0^{*,s}||\right]\Bigg) W^n\,.$$

Using Relation (5.14), *i.e.* $\sigma_0^{+,s}\sigma_0^{-,s} = 0$, we can add the terms marked in red in the following expression without changing anything:

$$|C_0|^s W_0^s = (|\mathcal{C}_0|^s + c_s\tau\sigma_{02}^s||\boldsymbol{\eta}_{02}^s|| + c_s\tau\sigma_{03}^s||\boldsymbol{\eta}_{03}^s||$$
$$+c_s\tau\sigma_0^{-,s}\frac{||\boldsymbol{\eta}_0^{*,s}||}{\dot{M}^s}\left[\left(\sigma_2^{+,s}+\sigma_3^{+,s}\right)||\boldsymbol{\eta}_2^{*,s}|| + \left(\sigma_1^{+,s}+\sigma_0^{+,s}\right)||\boldsymbol{\eta}_0^{*,s}||\right]$$
$$+c_s\tau\sigma_0^{+,s}\frac{||\boldsymbol{\eta}_0^{*,s}||}{\dot{M}^s}\left[-\left(\sigma_2^{-,s}+\sigma_3^{-,s}\right)||\boldsymbol{\eta}_2^{*,s}|| - \left(\sigma_1^{-,s}+\sigma_0^{-,s}\right)||\boldsymbol{\eta}_0^{*,s}||\right]\Bigg) W^n\,.$$

Using Relation (5.15), we then obtain:

$$|C_0|^s W_0^s = \Bigg(|\mathcal{C}_0|^n + c_s\tau\sigma_{02}^s||\boldsymbol{\eta}_{02}^s|| + c_s\tau\sigma_{03}^s||\boldsymbol{\eta}_{03}^s||$$
$$+ c_s\tau\underbrace{\left(\sigma_0^{-,s}+\sigma_0^{+,s}\right)}_{=\sigma_0^s}\frac{||\boldsymbol{\eta}_0^{*,s}||}{\dot{M}^s}\underbrace{\left[\left(\sigma_2^{+,s}+\sigma_3^{+,s}\right)||\boldsymbol{\eta}_2^{*,s}|| + \left(\sigma_1^{+,s}+\sigma_0^{+,s}\right)||\boldsymbol{\eta}_0^{*,s}||\right]}_{=\dot{M}^s}\Bigg) W^n$$
$$= \Bigg(|\mathcal{C}_0|^n + c_s\tau\sigma_{02}^s||\boldsymbol{\eta}_{02}^s|| + c_s\tau\sigma_{03}^s||\boldsymbol{\eta}_{03}^s|| + c_s\tau\sigma_0^s||\boldsymbol{\eta}_0^{*,s}||\Bigg) W^n$$
$$= \left(|\mathcal{C}_0|^n + A_{02}^s + A_{03}^s + A_0^s\right) W^n \;=\; |C_0|^s W^n\,.$$

Therefore, $W_0^s = W^n = W_0^n$. The same demonstration can be done for the four cells. The scheme therefore preserves the constant solution and, as a consequence, is DGCL for any RKSSP(n_s, p) scheme.

Conservativity. It may seem surprising that:

$$|C_0|^s + |C_1|^s + |C_2|^s + |C_3|^s \neq |C_0|^n + |C_1|^n + |C_2|^n + |C_3|^n\,.$$

Indeed, if solution W^n is constant, we have proved that the scheme is DGCL and thus W^s is also constant. The above inequality therefore rewrites:

$$|C_0|^s W_0^s + |C_1|^s W_1^s + |C_2|^s W_2^s + |C_3|^s W_3^s \neq |C_0|^n W_0^n + |C_1|^n W_1^n + |C_2|^n W_2^n + |C_3|^n W_3^n\,,$$

which could suggest that our scheme is not conservative.
However, no mass has been lost. It has simply been transferred to the evanescent cell which is "implicitly" taken into account and which is not associated with any vertex of the scheme. Even if the evanescent cell is not created explicitly in the code (with a new data structure for instance), it nevertheless exists and its existence should be taken into account when computing the discrete total mass at an intermediate configuration. If the evanescent cell is taken into account in the mass balance, conservativity is recovered:

$$\begin{aligned} & |C_0|^s W_0^s + |C_1|^s W_1^s + |C_2|^s W_2^s + |C_3|^s W_3^s + |C^{evanes.}|^s W_{evanes.}^s \\ = \; & |C_0|^n W_0^n + |C_1|^n W_1^n + |C_2|^n W_2^n + |C_3|^n W_3^n \,. \end{aligned}$$

Of course, when the final Runge-Kutta stage is reached, the total mass present at t^n has been redistributed and the conservativity is explicitly recovered:

$$\begin{aligned} & |C_0|^{n+1} W_0^{n+1} + |C_1|^{n+1} W_1^{n+1} + |C_2|^{n+1} W_2^s + |C_3|^{n+1} W_3^{n+1} + \underbrace{|C^{evanes.}|^{n+1}}_{=0} W_{evanes.}^{n+1} \\ = \; & |C_0|^{n+1} W_0^n + |C_1|^n W_1^{n+1} + |C_2|^n W_2^n + |C_3|^n W_3^n \,. \end{aligned}$$

Actually, our scheme is even more than conservative, it is *locally* conservative.

5.4.8.2 Accuracy order

As being DGCL is a sufficient condition for a scheme to be at least first-order time-accurate on moving meshes according to [Guillard 2000], our changing-topology RKSSP ALE schemes are at least of order one in time. Moreover, as we have followed Mavriplis-Yang approach, the changing-topology version of ALE RKSSP(s, p) time schemes with $p > 1$ are expected to be of order p in time.

Regarding spatial accuracy, the accuracy order can be enhanced using the MUSCL technique coupled with a limiter, therefore guaranteeing that this truly ALE scheme is intrinsically TVD, contrary to the classical projection approach which requires a repairing step. If low dissipation is desired, one must determine how to compute upwind/downwind gradients which are necessary for the V4/V6 schemes. Still following [Yang 2005], it is clear that preserving the expected order of accuracy in time imposes that the upwind/downwind elements are computed on the current Runge-Kutta configuration, *i.e.* on the mesh at t^s.

Once the upwind/downwind triangles are found, the V4/V6 schemes is naturally extended for our changing-topology ALE formulation. Therefore, this scheme is of order two in space, with low numerical dissipation.

5.4.8.3 $\mathbf{P}^1$-preservation

Contrary to some conservative interpolation schemes, like the one described in Section 1.2.3, our approach is not $\mathbf{P}^1$ preserving, *i.e.* it does not preserve a $\mathbf{P}^1$ solution. However, the classical Finite Volume ALE formulation neither preserves linear solutions. Therefore, it seems natural not to require such property for the changing topology formulation.

5.5 Fluid Structure Interactions issues

5.5.1 Description of the moving objects

In this work, the ALE formulation is used to perform computations involving bodies interacting with a surrounding fluid. Bodies are assumed to be **rigid**, of constant mass and **homogenous**, i.e their mass is uniformly distributed in their volume. Moving bodies are made of only one connected component and their topology remains the same during the whole movement, *i.e.* bodies do not split into different parts.

Movement prescription. The movement of a body can be described in two different ways:

As a data: the movement of the geometries is prescribed analytically as a data of the problem. Practically, the user provides two functions of time: the translational and rotational displacement of the object.

As an unknown: the position and orientation of the object is part of the unknowns. A Fluid-Structure Interaction (FSI) problem between the fluid and rigid bodies has to be solved along with the Euler equations.

We now detail the equations governing the movement of the objects in the latter case. Each rigid body B is fully described by:

- $\partial B = \boldsymbol{s}(u, v)$ (resp. $\boldsymbol{\gamma}(u)$) the parametrized surface (resp. curve in two dimensions) defining its boundary,
- $\boldsymbol{\eta} = \boldsymbol{\eta}(\boldsymbol{s}(u, v))$ the outward non-normalized normal to object boundary ∂B,
- $\mathbf{x}_G = (x_G(t),\, y_G(t),\, z_G(t))$ the position of its gravity center,
- $\boldsymbol{\theta} = \boldsymbol{\theta}(t)$ its angular displacement vector: the direction of $\boldsymbol{\theta}$ gives the rotation axis while its norm θ represents the angular displacement when a projection on the plane normal to $\boldsymbol{\theta}$ is performed,
- $\boldsymbol{\omega} = \dfrac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t}$ its angular speed vector,
- m its mass assumed to be constant,
- $\mathcal{J}_G$ its $n \times n$ matrix of inertia computed at G.

Matrix of inertia. The matrix of inertia of a solid is symmetric and depends only on the shape and physical nature of the solid object:

$$\mathcal{J}_G = \begin{pmatrix} \displaystyle\int_B \rho\left((y-y_G)^2 + (z-z_G)^2\right)\,\mathrm{d}\mathbf{x} & \displaystyle -\int_B \rho\,(x-x_G)\,(y-y_G)\,\mathrm{d}\mathbf{x} & \displaystyle -\int_B \rho\,(x-x_G)\,(z-z_G)\,\mathrm{d}\mathbf{x} \\ \displaystyle -\int_B \rho\,(x-x_G)\,(y-y_G)\,\mathrm{d}\mathbf{x} & \displaystyle\int_B \rho\left((x-x_G)^2 + (z-z_G)^2\right)\,\mathrm{d}\mathbf{x} & \displaystyle -\int_B \rho\,(y-y_G)\,(z-z_G)\,\mathrm{d}\mathbf{x} \\ \displaystyle -\int_B \rho\,(x-x_G)\,(z-z_G)\,\mathrm{d}\mathbf{x} & \displaystyle -\int_B \rho\,(y-y_G)\,(z-z_G)\,\mathrm{d}\mathbf{x} & \displaystyle\int_B \rho\left((x-x_G)^2 + (y-y_G)^2\right)\,\mathrm{d}\mathbf{x} \end{pmatrix}$$

And, as the objects are assumed to be homogeneous, the local volume mass of the object is constant $\rho = \dfrac{m}{V}$ with V the volume of B , and we get:

$$\mathcal{J}_G = \rho \begin{pmatrix} \int_B \left((y-y_G)^2 + (z-z_G)^2\right) \mathrm{d}\mathbf{x} & -\int_B (x-x_G)(y-y_G)\,\mathrm{d}\mathbf{x} & -\int_B (x-x_G)(z-z_G)\,\mathrm{d}\mathbf{x} \\ -\int_B (x-x_G)(y-y_G)\,\mathrm{d}\mathbf{x} & \int_B \left((x-x_G)^2 + (z-z_G)^2\right) \mathrm{d}\mathbf{x} & -\int_B (y-y_G)(z-z_G)\,\mathrm{d}\mathbf{x} \\ -\int_B (x-x_G)(z-z_G)\,\mathrm{d}\mathbf{x} & -\int_B (y-y_G)(z-z_G)\,\mathrm{d}\mathbf{x} & \int_B \left((x-x_G)^2 + (y-y_G)^2\right) \mathrm{d}\mathbf{x} \end{pmatrix}$$

Here are a few useful examples for $\mathcal{J}_G$.

Spheric ball of mass m and radius R: $\mathcal{J} = \dfrac{2mR^2}{5}\mathcal{I}_n$.

Solid cylinder of axis $\mathbf{a}$ *of radius R, height h and mass m*: the matrix of inertia is diagonal in any orthonormal basis $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ including $\mathbf{a}$.

$$\mathcal{J} = \left(\ \mathbf{a}\ \middle|\ \mathbf{b}\ \middle|\ \mathbf{c}\ \right) \begin{pmatrix} \dfrac{mR^2}{2} & 0 & 0 \\ 0 & \dfrac{1}{12}m\left(3R^2+h^2\right) & 0 \\ 0 & 0 & \dfrac{1}{12}m\left(3R^2+h^2\right) \end{pmatrix} \left(\ \mathbf{a}\ \middle|\ \mathbf{b}\ \middle|\ \mathbf{c}\ \right)^T .$$

Solid parallelepiped of mass m with height h, width w and depth d: if $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ are respectively the height, width and depth axis of the parallelepiped,

$$\mathcal{J} = \left(\ \mathbf{a}\ \middle|\ \mathbf{b}\ \middle|\ \mathbf{c}\ \right) \begin{pmatrix} \dfrac{1}{12}m\left(w^2+d^2\right) & 0 & 0 \\ 0 & \dfrac{1}{12}m\left(h^2+d^2\right) & 0 \\ 0 & 0 & \dfrac{1}{12}m\left(h^2+w^2\right) \end{pmatrix} \left(\ \mathbf{a}\ \middle|\ \mathbf{b}\ \middle|\ \mathbf{c}\ \right)^T$$

Axis of inertia Let us consider a solid particle of object B occupying volume $\mathrm{d}V$ and located at coordinates vector $\mathbf{x}$. The distance d between the particle and the axis of rotation passing through body gravity center G and collinear to arbitrary direction $\mathbf{a}$ is $d = ||\mathbf{x}-\mathbf{x}_G - ((\mathbf{x}-\mathbf{x}_G)\cdot\mathbf{a})\ \mathbf{a}||$. $\mathbf{a}$ is assumed to be normalized. By using formula $J_{(G\mathbf{a})} = md^2$ and some simple vector algebra, it can be seen that the moment of inertia of this particle relative to axis of rotation $(G\mathbf{a})$ is:

$$J_{(G\mathbf{a})} = \rho \mathrm{d}V \left(||\mathbf{x}-\mathbf{x}_G||^2 - [(\mathbf{x}-\mathbf{x}_G)\cdot\mathbf{a}]^2\right) ,$$

and a simple computation gives:

$$
\begin{aligned}
&||\mathbf{x}-\mathbf{x}_G||^2 - [(\mathbf{x}-\mathbf{x}_G)\cdot\mathbf{a}]^2 \\
&= \left(a_x^2+a_y^2+a_z^2\right)\left((x-x_G)^2+(y-y_G)^2+(z-z_G)^2\right) - \left(a_x(x-x_G)+a_y(y-y_G)+a_z(z-z_G)\right)^2 \\
&= a_x^2\left[(y-y_G)^2+(z-z_G)^2\right] + a_y^2\left[(x-x_G)^2+(z-z_G)^2\right] + a_z^2\left[(x-x_G)^2+(y-y_G)^2\right] \\
&\qquad - 2a_xa_y(x-x_G)(y-y_G) - 2a_ya_z(y-y_G)(z-z_G) - 2a_za_x(z-z_G)(x-x_G)\,.
\end{aligned}
$$

This is a quadratic form in $\mathbf{a}$, which leads to a tensor formula for the moment of inertia relative to axis $(G\mathbf{a})$:

$$
\mathrm{d}J_{(G\mathbf{a})} = \rho\,\mathrm{d}V\ (a_x,a_y,a_z)\begin{pmatrix} (y-y_G)^2+(z-z_G)^2 & -(x-x_G)(y-y_G) & -(x-x_G)(z-z_G) \\ -(y-y_G)(x-x_G) & (x-x_G)^2+(z-z_G)^2 & -(y-y_G)(z-z_G) \\ -(z-z_G)(x-x_G) & -(z-z_G)(y-y_G) & (x-x_G)^2+(y-y_G)^2 \end{pmatrix}\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}.
$$

The moment of inertia relative to axis $(G\mathbf{a})$ is therefore given by:

$$
\mathrm{d}J_{(G\mathbf{a})}(\mathbf{x}) \;=\; \mathbf{a}^T\,\mathrm{d}\mathcal{J}_G(\mathbf{x})\,\mathbf{a}\,.
$$

By integrating on all the volume of the body, the following relation between the matrix of inertia at point G and the moment of inertia relative to any axis passing through G is found:

$$
J_{(G\mathbf{a})} \;=\; \mathbf{a}^T\,\mathcal{J}_G(\mathbf{x})\,\mathbf{a}\,.
$$

The matrix of inertia is Symmetric Positive Definite and the spectral theorem applies: it has three different eigen-directions, called principal axes of inertia. Note that a symmetry axis of the object is also a principal axis of the matrix of inertia.

5.5.2 Movement of the geometries

We assume that bodies are only submitted to gravity and fluid pressure forces. The Euler equations for solid dynamics in an inertial frame then read:

$$
\left\{\begin{array}{lll} m\dfrac{\mathrm{d}^2\mathbf{x}_G}{\mathrm{d}t^2} &= \mathbf{F}_{ext} &= \displaystyle\int_{\partial B} p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s} \;+\; m\mathbf{g} \\ \dfrac{\mathrm{d}^2\mathcal{J}_G\boldsymbol{\theta}}{\mathrm{d}t^2} &= \mathbf{M}_G\left(\mathbf{F}_{ext}\right) &= \displaystyle\int_{\partial B}\left[(\boldsymbol{s}-\mathbf{x}_G)\wedge p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\right]\mathrm{d}\boldsymbol{s}\,, \end{array}\right. \tag{5.20}
$$

and we have used the fact that the kinetic moment of the gravity forces is zero because these forces are applied at gravity center G of the object.

5.5.2.1 Two dimensional case

In two dimensions, the angular displacement and angular speed vectors remain collinear to $\mathbf{e}_z$ during the whole simulation, thus $\boldsymbol{\theta}=\theta\mathbf{e}_z$ and $\boldsymbol{\omega}=\omega\mathbf{e}_z$. This means that we only need one equation on the kinetic moment as $\boldsymbol{\omega}$ is fully described by a scalar. Besides, $\mathbf{e}_z$ is also a symmetry

axis for the object and thus, it is a principal axis of inertia. In this context, the equation on the kinetic moment is projected on axis $(G\mathbf{e}_z)$:

$$\frac{\mathrm{d}^2\left(\mathcal{J}_G\boldsymbol{\theta}\right)}{\mathrm{d}t^2}\cdot\mathbf{e}_z = \frac{\mathrm{d}^2\left(\mathcal{J}_G\boldsymbol{\theta}\cdot\mathbf{e}_z\right)}{\mathrm{d}t^2} = \frac{\mathrm{d}^2\left(\mathbf{e}_z^T\cdot\left(\mathcal{J}_G\cdot\theta\mathbf{e}_z\right)\right)}{\mathrm{d}t^2} = \frac{\mathrm{d}^2\left[\theta\left(\mathbf{e}_z^T\cdot\mathcal{J}_G\cdot\mathbf{e}_z\right)\right]}{\mathrm{d}t^2}$$

$$= \left(\mathbf{e}_z^T\cdot\mathcal{J}_G\cdot\mathbf{e}_z\right)\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = J_{(Gz)}\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}$$

In this case, the dynamics fundamental relations applied to B simply read:

$$\left\{\begin{array}{lll} m\dfrac{\mathrm{d}^2x_G}{\mathrm{d}t^2} & = \mathbf{F}_{ext}\cdot\mathbf{e}_x & = \left(\displaystyle\int_{\partial B} p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_x + m\mathbf{g}\cdot\mathbf{e}_x \\ m\dfrac{\mathrm{d}^2y_G}{\mathrm{d}t^2} & = \mathbf{F}_{ext}\cdot\mathbf{e}_y & = \left(\displaystyle\int_{\partial B} p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_y + m\mathbf{g}\cdot\mathbf{e}_y \\ J_{(Gz)}\dfrac{\mathrm{d}^2\theta}{\mathrm{d}t^2} & = \mathbf{M}_G\left(\mathbf{F}_{ext}\right)\cdot\mathbf{e}_z & = \left(\displaystyle\int_{\partial B}\left(\boldsymbol{s}-\mathbf{x}_G\right)\wedge p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_z \end{array}\right. \tag{5.21}$$

and the kinetic moment of the gravity forces is zero when computed at body's gravity center G. This means that a system of Ordinary Differential Equation (ODE) in the unknown functions of time $(\mathbf{x}_G,\,\theta)$ must be solved for each body.

Additional constraint. If one point A of object B must remain fixed, the moment equation must be rewritten at fixed point A. Actually, A becomes the new center of reduction of the dynamic and action torsors. The system to be solved now reads:

$$\left\{\begin{array}{ll} \dfrac{\mathrm{d}x_A}{\mathrm{d}t} = 0, \quad \dfrac{\mathrm{d}y_A}{\mathrm{d}t} = 0 & \\ J_A\dfrac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = \mathbf{M}_A\left(\mathbf{F}_{ext}\right)\cdot\mathbf{e}_z & \\ \qquad = \left(\displaystyle\int_{\partial B}\left(\boldsymbol{s}-\mathbf{x_A}\right)\wedge p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_z + m\left(\displaystyle\int_{\partial B}\left(\boldsymbol{s}-\mathbf{x_A}\right)\wedge\mathbf{g}\,\mathrm{d}\boldsymbol{s}\right)\cdot\mathbf{e}_z & \end{array}\right. \tag{5.22}$$

with $\mathbf{M}_A$ the kinetic moment of the external forces applied on ∂B computed at point A and J_A the moment of inertia of the object related to axis (Az). According to Huygens theorem, we have:

$$J_{(Az)} \;=\; J_{(Gz)} \;+\; m\,\mathrm{dist}\left((Gz),\,(Az)\right)^2$$

and $\mathrm{dist}\left((Gz),\,(Az)\right) = \|\overrightarrow{GA}\|$ is the distance between axes (Gz) and (Az).

5.5.2.2 Three dimensional case

Things are much more complex in three dimensions. Indeed, the number of parameters needed to describe the dynamics of a rigid body now equals six, hence the name 6-DOF problem (6-Degrees-of-Motion) used for this class of problems. This means that, contrary to the two-dimensional case, System (5.20) must be fully utilized to compute the movement. Moreover, the directions of $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are not constant in time anymore, and none of the principal axes of inertia keeps a constant direction. This means that:

$$\frac{\mathrm{d}^2\left(\mathcal{J}_G\boldsymbol{\theta}\right)}{\mathrm{d}t^2} \;\neq\; \mathcal{J}_G\frac{\mathrm{d}^2\boldsymbol{\theta}}{\mathrm{d}t^2}\,,$$

as $\mathcal{J}_G$, written in the canonical basis, is not constant in time anymore.

To solve these difficulties, the idea is to rewrite Euler fundamental relations in a moving frame of reference attached to the body. A natural orthonormal basis in this frame is formed by the three principal axes $(\mathbf{e_1}(t), \mathbf{e_2}(t), \mathbf{e_3}(t))$ of the body. In this frame, the matrix of inertia is constant in time and, as we have chosen a good basis, it is also diagonal:

$$\mathcal{J}_{G|(\mathbf{e_1}(t),\,\mathbf{e_2}(t),\,\mathbf{e_3}(t))} \;=\; \mathrm{diag}\left(J_1^b,\, J_2^b,\, J_3^b\right), \quad \text{with } J_1^b,\, J_2^b,\, J_3^b \text{ constant in time}. \tag{5.23}$$

In the sequel, we note $\mathcal{R} = \mathcal{R}(t)$ the matrix enabling to pass from the description in basis $(\mathbf{e_1}(t),\, \mathbf{e_2}(t),\, \mathbf{e_3}(t))$ to the description in fixed canonical basis $(\mathbf{e}_x,\, \mathbf{e}_y,\, \mathbf{e}_z)$:

$$\begin{aligned}
\mathbf{e_1}(t) &= r_{1x}(t)\mathbf{e}_x + r_{1y}(t)\mathbf{e}_y + r_{1z}(t)\mathbf{e}_z \\
\mathbf{e_2}(t) &= r_{2x}(t)\mathbf{e}_x + r_{2y}(t)\mathbf{e}_y + r_{2z}(t)\mathbf{e}_z \\
\mathbf{e_3}(t) &= r_{3x}(t)\mathbf{e}_x + r_{3y}(t)\mathbf{e}_y + r_{3z}(t)\mathbf{e}_z
\end{aligned}$$

$$\mathcal{R} = \mathcal{R}(t) = \begin{array}{c} \left(\begin{array}{ccc} r_{1x}(t) & r_{2x}(t) & r_{3x}(t) \\ r_{1y}(t) & r_{2y}(t) & r_{3y}(t) \\ r_{1z}(t) & r_{2z}(t) & r_{3z}(t) \end{array}\right) \\ \mathbf{e_1}(t) \quad \mathbf{e_2}(t) \quad \mathbf{e_3}(t) \end{array} \begin{array}{c} \mathbf{e}_x \\ \mathbf{e}_y \\ \mathbf{e}_z \end{array}$$

As the two basis are orthonormal, $\mathcal{R}(t)$ is always an unitary matrix, *i.e.* $\mathcal{R}(t)\mathcal{R}(t)^T = \mathcal{I}_3,\ \forall\ t$.

Derivation of vectors written in the body frame. An arbitrary vector $\mathbf{v}^b(t)$ described in frame $(\mathbf{e}_1(t),\, \mathbf{e}_2(t),\, \mathbf{e}_3(t))$ associated with the body has a description $\mathbf{v}(t)$ in the fixed inertial frame. The link between these two representations is given by relation:

$$\mathbf{v}(t) \;=\; \mathcal{R}(t)\,\mathbf{v}^b(t)\,.$$

As transformation $\mathcal{R}(t)$ itself is time dependent, in general :

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t}(t) \neq \mathcal{R}(t)\frac{\mathrm{d}\mathbf{v}^b}{\mathrm{d}t}(t)\,.$$

The movement of the body frame must be taken into account while derivating:

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t}(t) \;=\; \frac{\mathrm{d}\mathcal{R}(t)}{\mathrm{d}t}\mathbf{v}^b(t) + \mathcal{R}(t)\frac{\mathrm{d}\mathbf{v}^b(t)}{\mathrm{d}t} \;=\; \frac{\mathrm{d}\mathcal{R}(t)}{\mathrm{d}t}\mathcal{R}^T\mathbf{v}(t) + \mathcal{R}(t)\frac{\mathrm{d}\mathbf{v}^b(t)}{\mathrm{d}t} \;=\; \omega(t)\wedge\mathbf{v}(t) + \mathcal{R}(t)\frac{\mathrm{d}\mathbf{v}^b(t)}{\mathrm{d}t}\,.$$

Indeed, by derivating Relation $\mathcal{R}^T(t)\mathcal{R}(t) \;=\; \mathcal{I}_3$, we get:

$$\frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t)\mathcal{R}^T(t) \;+\; \mathcal{R}(t)\frac{\mathrm{d}\mathcal{R}^T}{\mathrm{d}t}(t) \;=\; 0$$

$$\Rightarrow\quad \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t)\mathcal{R}^T(t) \;=\; -\left(\frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t)\mathcal{R}(t)^T\right)^T \quad (\textit{ i.e. } \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t)\mathcal{R}(t)^T \text{ is skew-symmetric})$$

$$\Rightarrow\quad \exists\,\boldsymbol{\omega} \;=\; \boldsymbol{\omega}(t) \in \mathbb{R}^3,\ \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t)\mathcal{R}^T(t) \;=\; \left(\begin{array}{ccc} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{array}\right) \;=\; \boldsymbol{\omega}(t)\wedge\cdot\,,$$

with $\boldsymbol{\omega}(t) = (\omega_1(t), \omega_2(t), \omega_3(t))$. The first term $\boldsymbol{\omega}(t) \wedge \cdot$ accounts for the movement of the body frame (its rotation) as compared to the inertial frame. $\boldsymbol{\omega}(t)$ is the instantaneous angular speed of the body frame as compared to the inertial frame, written in the canonical, fixed basis $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$. We note $\boldsymbol{\omega}^b$ this same angular speed written in moving basis $(\mathbf{e_1}(t), \mathbf{e_2}(t), \mathbf{e_3}(t))$ attached to the body. We have $\boldsymbol{\omega}(t) = \mathcal{R}(t)\boldsymbol{\omega}^b(t)$. Then, using the invariance property of the cross-product, we have, for any arbitrary vector $\mathbf{v}$:

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t}(t) = \left(\mathcal{R}(t)\boldsymbol{\omega}^b(t)\right) \wedge \mathbf{v} + \mathcal{R}(t)\frac{\mathrm{d}\mathbf{v}^b(t)}{\mathrm{d}t} = \mathcal{R}(t)\left(\boldsymbol{\omega}^b(t) \wedge \mathbf{v}^b(t) + \frac{\mathrm{d}\mathbf{v}^b(t)}{\mathrm{d}t}\right). \tag{5.24}$$

Rewriting System (5.20). Using Formula (5.24) to rewrite $\frac{\mathrm{d}(\mathcal{J}\boldsymbol{\omega})}{\mathrm{d}t}$ in System (5.20), we get:

$$\frac{\mathrm{d}\left(\mathcal{J}_G\boldsymbol{\omega}\right)}{\mathrm{d}t} = \mathbf{M}_G\left(\mathbf{F}_{ext}\right) \Rightarrow \mathcal{R}(t)\left(\boldsymbol{\omega}^b \wedge \mathcal{J}_G^b\boldsymbol{\omega}^b + \frac{\mathrm{d}\left(\mathcal{J}_G^b\boldsymbol{\omega}^b\right)}{\mathrm{d}t}\right) = \mathcal{R}(t)\mathbf{M}_G^b\left(\mathbf{F}_{ext}^b\right)$$

$$\Rightarrow \boldsymbol{\omega}^b \wedge \mathcal{J}_G^b\boldsymbol{\omega}^b + \frac{\mathrm{d}\left(\mathcal{J}_G^b\boldsymbol{\omega}^b\right)}{\mathrm{d}t} = \mathbf{M}_G^b\left(\mathbf{F}_{ext}^b\right) \Rightarrow \boldsymbol{\omega}^b \wedge \mathcal{J}_G^b\boldsymbol{\omega}^b + \mathcal{J}_G^b\frac{\mathrm{d}\boldsymbol{\omega}^b}{\mathrm{d}t} = \mathbf{M}_G^b\left(\mathbf{F}_{ext}^b\right),$$

and $\mathcal{J}_G^b$ written in the moving body frame is invariant. Term $\boldsymbol{\omega}^b \wedge \mathcal{J}^b\boldsymbol{\omega}^b$ can be developped as:

$$\begin{pmatrix} 0 & -\omega_3^b & \omega_2^b \\ \omega_3^b & 0 & -\omega_1^b \\ -\omega_2^b & \omega_1^b & 0 \end{pmatrix}\begin{pmatrix} J_1^b & 0 & 0 \\ 0 & J_2^b & 0 \\ 0 & 0 & J_3^b \end{pmatrix}\begin{pmatrix} \omega_1^b \\ \omega_2^b \\ \omega_3^b \end{pmatrix} = \begin{pmatrix} \left(J_3^b - J_2^b\right)\omega_2^b\,\omega_3^b \\ \left(J_1^b - J_3^b\right)\omega_3^b\,\omega_1^b \\ \left(J_2^b - J_1^b\right)\omega_1^b\,\omega_2^b \end{pmatrix},$$

which finally leads to the following system of equations for the dynamics of rigid bodies:

$$\begin{pmatrix} J_1^b & 0 & 0 \\ 0 & J_2^b & 0 \\ 0 & 0 & J_3^b \end{pmatrix}\begin{pmatrix} \dot{\omega}_1^b \\ \dot{\omega}_2^b \\ \dot{\omega}_3^b \end{pmatrix} + \begin{pmatrix} \left(J_3^b - J_2^b\right)\omega_2^b\,\omega_3^b \\ \left(J_1^b - J_3^b\right)\omega_3^b\,\omega_1^b \\ \left(J_2^b - J_1^b\right)\omega_1^b\,\omega_2^b \end{pmatrix} = \mathbf{M}_G^b\left(\mathbf{F}_{ext}^b\right)$$

$$\Leftrightarrow \begin{pmatrix} \dot{\omega}_1^b \\ \dot{\omega}_2^b \\ \dot{\omega}_3^b \end{pmatrix} = \begin{pmatrix} \frac{J_2^b - J_3^b}{J_1^b}\,\omega_2^b\,\omega_3^b \\ \frac{J_3^b - J_1^b}{J_2^b}\,\omega_3^b\,\omega_1^b \\ \frac{J_1^b - J_2^b}{J_3^b}\,\omega_1^b\,\omega_2^b \end{pmatrix} + \begin{pmatrix} \frac{M_{G,1}^b}{J_1^b} \\ \frac{M_{G,2}^b}{J_2^b} \\ \frac{M_{G,3}^b}{J_3^b} \end{pmatrix} \tag{5.25}$$

with $\mathbf{M}_G^b\left(\mathbf{F}_{ext}^b\right)$ given by:

$$\mathbf{M}_G^b\left(\mathbf{F}_{ext}^b\right) = \mathcal{R}(t)^T\mathbf{M}_G\left(\mathbf{F}_{ext}\right). \tag{5.26}$$

5.5.3 Resolution of the rigid bodies equations and loose coupling

In this section, the resolution of the FSI problem in three-dimensions is explained.
The equation governing the position of the rigid object gravity center is quiet easy to solve as it is linear.

$$m\frac{\mathrm{d}^2\mathbf{x}_G}{\mathrm{d}t^2} = \int_{\partial B} p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\,\mathrm{d}\boldsymbol{s} + m\mathbf{g}. \tag{5.27}$$

The kinetic moment equations are more difficult to handle. Equation (5.25) describing the movement of three-dimensional moving objects is a non-linear ODE system of the second-order in θ. In order to solve this problem, it must first be rewritten as a system of equations of greater size but of the first order:

$$\begin{pmatrix} \dot{\theta}_1^b \\ \dot{\theta}_2^b \\ \dot{\theta}_3^b \\ \ddot{\theta}_1^b \\ \ddot{\theta}_2^b \\ \ddot{\theta}_3^b \end{pmatrix} = \begin{pmatrix} \dot{\theta}_1^b \\ \dot{\theta}_2^b \\ \dot{\theta}_3^b \\ \frac{J_2^b - J_3^b}{J_1^b} \dot{\theta}_2^b \dot{\theta}_3^b \\ \frac{J_3^b - J_1^b}{J_2^b} \dot{\theta}_3^b \dot{\theta}_1^b \\ \frac{J_1^b - J_2^b}{J_3^b} \dot{\theta}_1^b \dot{\theta}_2^b \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{M_{G,1}^b}{J_1^b} \\ \frac{M_{G,2}^b}{J_2^b} \\ \frac{M_{G,3}^b}{J_3^b} \end{pmatrix}$$

However, $\mathbf{M}_G^b$ cannot be computed directly as the moving frame associated with the moving object is not known. Indeed:

$$\begin{aligned} & \mathbf{M}_G^b \left(\mathbf{F}_{ext}^b \right) = \mathcal{R}(t)^T \mathbf{M}_G \left(\mathbf{F}_{ext} \right) \\ \Leftrightarrow \quad & \begin{pmatrix} M_{G,1}^b \\ M_{G,2}^b \\ M_{G,3}^b \end{pmatrix} = \begin{pmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{pmatrix} \begin{pmatrix} M_{G,1} \\ M_{G,2} \\ M_{G,3} \end{pmatrix} = \begin{pmatrix} r_{11} M_{G,1} + r_{21} M_{G,2} + r_{31} M_{G,3} \\ r_{12} M_{G,1} + r_{22} M_{G,2} + r_{32} M_{G,3} \\ r_{13} M_{G,1} + r_{23} M_{G,2} + r_{33} M_{G,3} \end{pmatrix} \end{aligned}$$

Therefore, the orientation of this moving frame as compared to fixed frame ($\mathbf{e}_x$, $\mathbf{e}_y$, $\mathbf{e}_z$) is also an unknwon of the problem and is fully described by matrix $\mathcal{R}(t)$. In Section 5.5.2.2, we have already shown that:

$$\frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t)\mathcal{R}^T(t) = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \Leftrightarrow \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t) = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \mathcal{R}(t)$$

$$\Leftrightarrow \quad \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t) = \left[\mathcal{R}(t) \begin{pmatrix} 0 & -\omega_3^b & \omega_2^b \\ \omega_3^b & 0 & -\omega_1^b \\ -\omega_2^b & \omega_1^b & 0 \end{pmatrix} \mathcal{R}(t)^T \right] \mathcal{R}(t) = \mathcal{R}(t) \begin{pmatrix} 0 & -\omega_3^b & \omega_2^b \\ \omega_3^b & 0 & -\omega_1^b \\ -\omega_2^b & \omega_1^b & 0 \end{pmatrix}.$$

This linear system can be detailed coefficient by coefficient and we get the following nine first-order differential equations on the coefficients of $\mathcal{R}$:

$$\frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t}(t) = \mathcal{R}(t) \begin{pmatrix} 0 & -\dot{\theta}_3^b & \dot{\theta}_2^b \\ \dot{\theta}_3^b & 0 & -\dot{\theta}_1^b \\ -\dot{\theta}_2^b & \dot{\theta}_1^b & 0 \end{pmatrix} \tag{5.28a}$$

$$\Leftrightarrow \begin{pmatrix} \dot{r_{11}} & \dot{r_{12}} & \dot{r_{13}} \\ \dot{r_{21}} & \dot{r_{22}} & \dot{r_{23}} \\ \dot{r_{31}} & \dot{r_{32}} & \dot{r_{33}} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} 0 & -\dot{\theta}_3^b & \dot{\theta}_2^b \\ \dot{\theta}_3^b & 0 & -\dot{\theta}_1^b \\ -\dot{\theta}_2^b & \dot{\theta}_1^b & 0 \end{pmatrix}$$

$$\Leftrightarrow \left\{ \begin{array}{rclrclrcl} \dot{r_{11}} &=& \dot{\theta}_3^b r_{12} - \dot{\theta}_2^b r_{13} & \dot{r_{21}} &=& \dot{\theta}_3^b r_{22} - \dot{\theta}_2^b r_{23} & \dot{r_{31}} &=& \dot{\theta}_3^b r_{32} - \dot{\theta}_2^b r_{33} \\ \dot{r_{12}} &=& -\dot{\theta}_3^b r_{11} + \dot{\theta}_1^b r_{13} & \dot{r_{22}} &=& -\dot{\theta}_3^b r_{21} + \dot{\theta}_1^b r_{23} & \dot{r_{32}} &=& -\dot{\theta}_3^b r_{31} + \dot{\theta}_1^b r_{33} \\ \dot{r_{13}} &=& \dot{\theta}_2^b r_{11} - \dot{\theta}_1^b r_{12} & \dot{r_{23}} &=& \dot{\theta}_2^b r_{21} - \dot{\theta}_1^b r_{22} & \dot{r_{33}} &=& \dot{\theta}_2^b r_{31} - \dot{\theta}_1^b r_{32}\,, \end{array} \right.$$

and $\dot{\boldsymbol{\theta}^b} = \frac{\mathrm{d}\boldsymbol{\omega}^b}{\mathrm{d}t}$. Finally, a larger system of size 15×15 must be solved and it writes:

$$\begin{pmatrix} \dot{r_{11}} \\ \dot{r_{12}} \\ \dot{r_{13}} \\ \dot{r_{21}} \\ \dot{r_{22}} \\ \dot{r_{23}} \\ \dot{r_{31}} \\ \dot{r_{32}} \\ \dot{r_{33}} \\ \dot{\theta}_1^b \\ \dot{\theta}_2^b \\ \dot{\theta}_3^b \\ \ddot{\theta}_1^b \\ \ddot{\theta}_2^b \\ \ddot{\theta}_3^b \end{pmatrix} = \begin{pmatrix} \dot{\theta}_3^b r_{12} - \dot{\theta}_2^b r_{13} \\ -\dot{\theta}_3^b r_{11} + \dot{\theta}_1^b r_{13} \\ \dot{\theta}_2^b r_{11} - \dot{\theta}_1^b r_{12} \\ \dot{\theta}_3^b r_{22} - \dot{\theta}_2^b r_{23} \\ -\dot{\theta}_3^b r_{21} + \dot{\theta}_1^b r_{23} \\ \dot{\theta}_2^b r_{21} - \dot{\theta}_1^b r_{22} \\ \dot{\theta}_3^b r_{32} - \dot{\theta}_2^b r_{33} \\ -\dot{\theta}_3^b r_{31} + \dot{\theta}_1^b r_{33} \\ \dot{\theta}_2^b r_{31} - \dot{\theta}_1^b r_{32} \\ \dot{\theta}_1^b \\ \dot{\theta}_2^b \\ \dot{\theta}_3^b \\ \left[\left(J_2^b - J_3^b\right) \dot{\theta}_2^b \dot{\theta}_3^b + r_{11} M_{G,1} + r_{21} M_{G,2} + r_{31} M_{G,3} \right] / J_1^b \\ \left[\left(J_3^b - J_1^b\right) \dot{\theta}_3^b \dot{\theta}_1^b + r_{12} M_{G,1} + r_{22} M_{G,2} + r_{32} M_{G,3} \right] / J_2^b \\ \left[\left(J_1^b - J_2^b\right) \dot{\theta}_1^b \dot{\theta}_2^b + r_{13} M_{G,1} + r_{23} M_{G,2} + r_{33} M_{G,3} \right] / J_3^b \end{pmatrix} \tag{5.29}$$

Let $\mathbf{H}$ be a vectorial, non linear function of $\mathbb{R}^{15}$ defined by:

$$\mathbf{H}: \quad \mathbb{R}^{15} \longrightarrow \mathbb{R}^{15}$$

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{pmatrix} \longmapsto \mathbf{H}(\mathbf{X}) = \begin{pmatrix} x_{15}x_2 - x_{14}x_3 \\ -x_{15}x_1 + x_{13}x_3 \\ x_{14}x_1 - x_{13}x_2 \\ x_{15}x_5 - x_{14}x_6 \\ -x_{15}x_4 + x_{13}x_6 \\ x_{14}x_4 - x_{13}x_5 \\ x_{15}x_8 - x_{14}x_9 \\ -x_{15}x_7 + x_{13}x_9 \\ x_{14}x_7 - x_{13}x_8 \\ x_{13} \\ x_{14} \\ x_{15} \\ \left[\left(J_2^b - J_3^b\right) x_{14}\, x_{15} + x_1 M_{G,1}(\mathbf{X}, \mathbf{F}_{ext}) + x_4 M_{G,2}(\mathbf{X}, \mathbf{F}_{ext}) + x_7 M_{G,3}(\mathbf{X}, \mathbf{F}_{ext})\right] / J_1^b \\ \left[\left(J_3^b - J_1^b\right) x_{15}\, x_{13} + x_2 M_{G,1}(\mathbf{X}, \mathbf{F}_{ext}) + x_5 M_{G,2}(\mathbf{X}, \mathbf{F}_{ext}) + x_8 M_{G,3}(\mathbf{X}, \mathbf{F}_{ext})\right] / J_2^b \\ \left[\left(J_1^b - J_2^b\right) x_{13}\, x_{14} + x_3 M_{G,1}(\mathbf{X}, \mathbf{F}_{ext}) + x_6 M_{G,2}(\mathbf{X}, \mathbf{F}_{ext}) + x_9 M_{G,3}(\mathbf{X}, \mathbf{F}_{ext})\right] / J_3^b \end{pmatrix}$$

Then, System (5.29) can be summarize into:

$$\mathbf{V^b} = \begin{pmatrix} \mathbf{r} \\ \boldsymbol{\theta}^b \\ \boldsymbol{\omega}^b = \dot{\boldsymbol{\theta}}^b \end{pmatrix}, \quad \dot{\mathbf{V}}^b = \begin{pmatrix} \dot{\mathbf{r}} \\ \dot{\boldsymbol{\theta}}^b \\ \dot{\boldsymbol{\omega}}^b = \ddot{\boldsymbol{\theta}}^b \end{pmatrix}, \quad \dot{\mathbf{V}}^b(t) = \mathbf{H}(\mathbf{V}^b(t)),$$

with

$$\mathbf{r}(t) = (r_{11}(t),\, r_{12}(t),\, r_{13}(t),\, r_{21}(t),\, r_{22}(t),\, r_{23}(t),\, r_{31}(t),\, r_{32}(t),\, r_{33}(t))^T .$$

Explicit coupling. 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 (5.27-5.29) is performed using the same RKSSP scheme as the one used to advance the fluid numerical solution. The coupling is loose and explicit as the external forces and moments applying on rigid objects are computed on the current configuration.

Moving vertices belonging to object boundaries. Finally, the displacement of a vertex P of coordinates vector $\mathbf{x}$ which is attached to the object and which moves with the same rigid movement is given by:

$$\begin{aligned} \mathbf{d} &= (\mathbf{x}\cdot\boldsymbol{\theta}_u)\boldsymbol{\theta}_u + \cos\theta\ [\mathbf{x} - (\mathbf{x}\cdot\boldsymbol{\theta}_u)\boldsymbol{\theta}_u] + \sin\theta\ [\boldsymbol{\theta}_u \wedge (\mathbf{x} - (\mathbf{x}\cdot\boldsymbol{\theta}_u)\boldsymbol{\theta}_u)] - \mathbf{x} \\ &= (\mathbf{x}\cdot\boldsymbol{\theta}_u)\boldsymbol{\theta}_u + \cos\theta\ [\mathbf{x} - (\mathbf{x}\cdot\boldsymbol{\theta}_u)\boldsymbol{\theta}_u] + \sin\theta\ (\boldsymbol{\theta}_u \wedge \mathbf{x}) - \mathbf{x} \end{aligned}$$

This formula can be interpreted geometrically as shown on Figure 5.16.

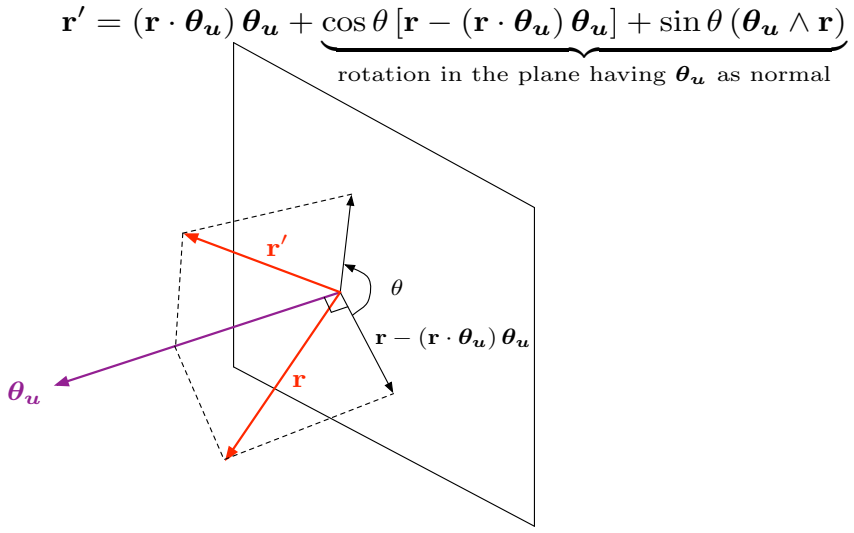


Figure 5.16: Three-dimensional rotation of an angle θ around an axis of direction $\boldsymbol{\theta}_u$.

5.6 Numerical results

5.6.1 Turbo-machinery

To show the usefulness and efficiency of the swap operation, we performed a simplified turbo-machinery simulation. A fluid is emitted radially from a central hub of radius $r_0 = 0.3$. The emitted fluid applies a pressure on the blades, which makes them rotate quicker and quicker. By centrifugal effects, the fluid is propelled toward the exterior. The initial conditions, written in cylindrical coordinates $(\mathbf{e}_r, \mathbf{e}_\theta)$ are as follows:

$$\rho = \rho(r) = \frac{r_0}{r}\rho(r_0) = \frac{r_0}{r}\rho_{in}, \quad \rho\mathbf{u} = \rho(r)q_{unif}\mathbf{e}_r, \quad \rho e = \frac{p_{unif}}{\gamma - 1} + \frac{1}{2}\rho(r)q_{unif}^2$$

with $\rho_{in} = \rho(r_0) = 1$ the inflow density on the hub, $p_{unif} = 1$ the initial uniform pressure and $q_{unif} = ||\mathbf{u}_{unif}||$ the initial uniform radial velocity. This initial state ensures that we initially have a constant, uniform radial flow. Indeed, if we take a ring delimited by radius r_0 and $r > r_0$, all the mass entering the ring across the inner circle is expelled through the outer circle, which means that the initial state satisfies the stationary conservation equation $\mathrm{div}\,(\rho\,\mathbf{u}) = 0$.

$$\int_{\theta=0}^{2\Pi} \rho(r)\mathbf{u}(u)\cdot\mathbf{e}_r\, r\mathrm{d}\theta = \int_{\theta=0}^{2\Pi} \rho(r_0)\mathbf{u}(r_0)\cdot\mathbf{e}_r\, r_0\mathrm{d}\theta$$

The details of the geometry are given in Figure 5.17. An inflow boundary condition is imposed on the hub, a slipping boundary condition on the blades and a transmitting boundary condition on the bounding box.

The initial mesh is shown on Figure 5.17. Several sub-domains are defined: sub-domain ① is linked to the hub and is set to be static, sub-domain ② is linked to the blades and so turns while sub-domain ③ is simply the rest of the computational domain and is also set to be static.

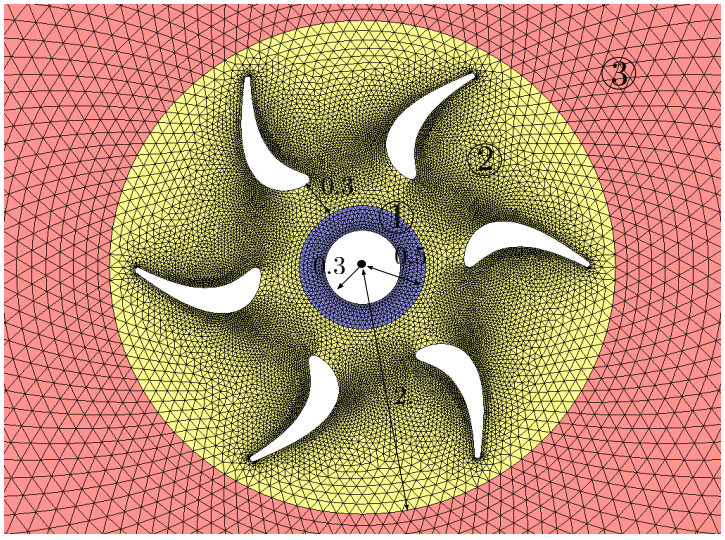


Figure 5.17: The test case geometry and the associated mesh. The three sub-domains of the mesh are represented in violet, yellow and pink, respectively.

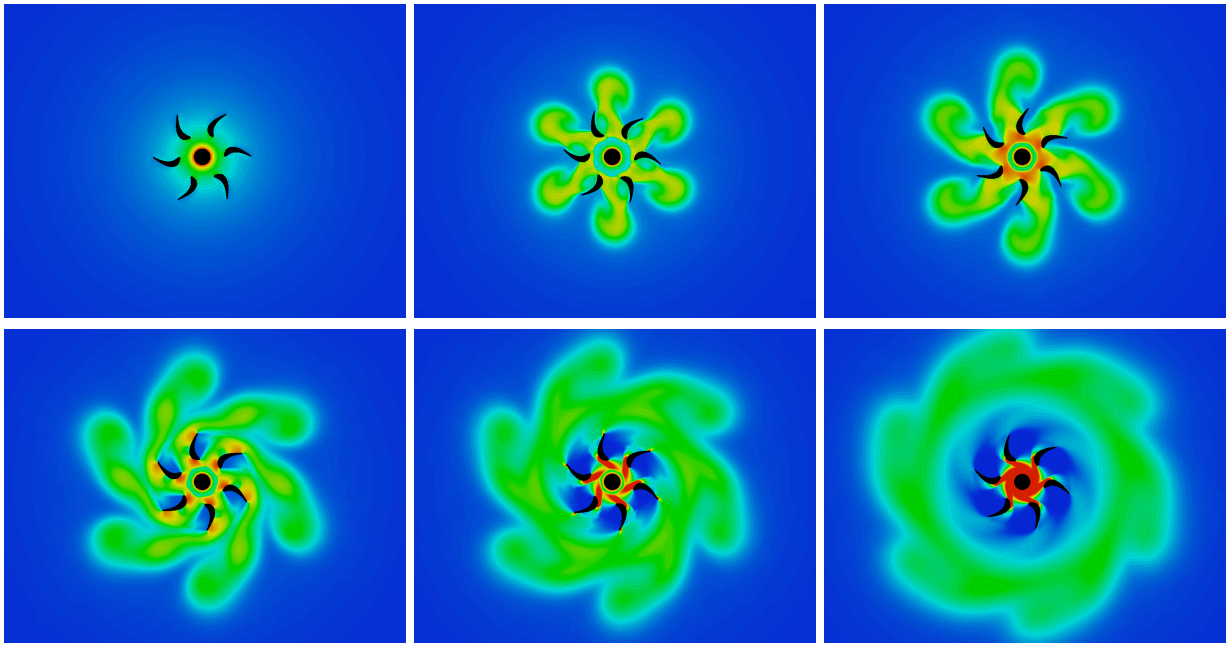


Figure 5.18: The density of the fluid obtained by resolution of the complete fluid-rigid-body interaction problem, using our changing-topology ALE scheme. Solution at a-dimensioned time $t = 0.05$, 4, 6, 9, 12 and 14.

This simulation constitutes an excellent example of the efficiency of the swap when the mesh is sheared. Indeed, if we forbid the use of the swap operation, the mesh deteriorates very quickly because while trying to follow the blades movement, the elements progressively stretch until the minimal altitude of the mesh is so small that the simulation cannot advance in time anymore. Indeed, the CFL condition

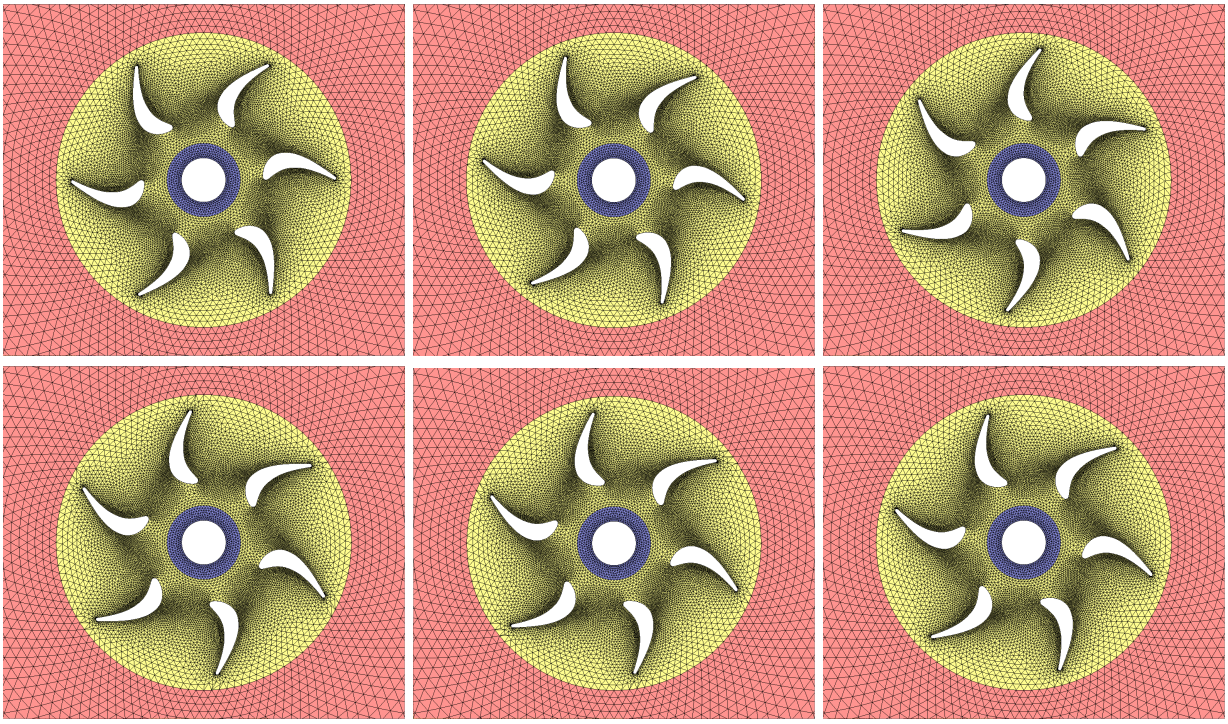


Figure 5.19: The variable-topology mesh at a-dimensioned time $t = 0.05$, 4, 6, 9, 12 and 14 (about 6 turns). Final mesh and solution are the obtained without a single global re-meshing, and thus without any interpolation.

makes the solver time step tend to 0. In our simulation, the initial mesh has an excellent quality and during the movement, only the layers of elements separating moving sub-domain 1 from the two other fixed sub-domains change in time. The vertices attached to the blades are the only ones to move in time, and, as they do it in a completely rigid manner, the initial quality of the mesh is preserved, without any remeshing, see Figure 5.19. Therefore, **the excellent quality of the initial mesh has been maintained throughout the computation and the simulation can evolve as long as desired**. Besides, we note that the changing-topology ALE scheme works well and gives a result which at least seems in accordance with physical intuition, Figure 5.18. The conservativity has also been positively checked.

5.6.2 Pitching NACA0012 airfoil

A pitching NACA airfoil (AGARD CT5 test case) has been simulated with our changing-topology ALE solver. Mesh adaptation has been performed using the extension of metric-based multi-scale mesh adaptation to moving mesh problems described in Chapter 3. The angle of attack α of the NACA is prescribed analytically by:

$$\alpha(t) = \alpha(t_0) + \max \alpha \, \sin(\kappa\, t), \text{ with: } \alpha(t_0) = 0.016^\circ, \ \max(\alpha) = 2.51^\circ \text{ and } \kappa = 0.1628 \text{ rad}/s\,.$$

The period of the movement is $T_\kappa = 2\pi/\kappa = 38.5945\, s$. The inflow Mach is 0.755. The simulation is of spatial order 2, with a V4 scheme. The RKSSP(3,3) scheme has been chosen for the temporal discretization. Figure 5.20 shows the results obtained for this simulation. While the angle of attack increases to its maximum, the upper shock wave moves towards the trailing edge and becomes sharper.

On the contrary, the shock located on the intrados moves toward the leading edge and its amplitude diminishes. Once the maximum value of the angle of attack has been reached, it decreases and the shock wave located on the extrados moves back to the leading edge while its amplitude is dropping. The intrados shock wave moves to the trailing edge and becomes sharper and sharper. The same phenomenon occurs for the second-half of the period.

All the elements located inside a disk containing the airfoil have been stiffened, see Section 4.1.3. All the vertices located inside this rigid disk have therefore the same pitching movement as the NACA. This rigidifying process allows to preserve the quality of the initial mesh and reduces to its bare minimum the CPU time devoted to mesh movement and optimization. Figure 5.20 shows how the shock wave always evolves inside an adapted strip-shaped area of the mesh.

The adaptation is anisotropic and the density of the flow is taken as sensor. While the angle of attack increases toward an extremum, the adaptation process naturally tends to privilege more and more the side of the airfoil on which the shock wave becomes sharper. On the contrary, when the angle of attack approaches zero, the flow is nearly symmetric and both shocks have a similar amplitude. Therefore, the adaptation effort is balanced between the intrados and the extrados.

5.6.3 Blast test case

The changing-topology ALE scheme coupled with anisotropic metric-based mesh adaptation was tested on a two-dimensional blast test case proposed in [Baum 1989]. A very strong shock wave at Mach 10 impacts a rectangular object, which is blown up. The object is first maintained fixed at the lower right corner and released when the vertical speed exceeds a given threshold. The results obtained by performing an $\mathbf{L}^2$ anisotropic adaptation on the density of the flow are shown in Figure 5.21.

First, the ALE formulation of the swap turns out to be very useful in this simulation to handle the mesh shearing between the ground and the bottom of the object when it is released. Second, the ALE fixed-point algorithm really enables to take the mesh movement into account in the adaptation process. Indeed, the reflected shock wave not only evolves inside the adapted strip-shaped area, but the strip-shaped area itself moves toward the shock wave. This ALE formulation of the swap, coupled with new efficient moving mesh technique, enables to re-mesh only when the fixed-point algorithm requires it and not because we are forced to do it due to our inability to move the mesh properly. To finish with, let us give some CPU times. This blast test case has been run without mesh adaptation using on the one hand the global re-meshing strategy (*i.e.* the domain is re-meshed each time the quality of the mesh exceeds a prescribed threshold) and on the other hand using our changing-topology strategy. The global re-meshing strategy has required 32 remeshings while not a single global remeshing has been performed with our strategy. In terms of CPU time, the global re-meshing strategy was more than two times slower than our strategy, with the same quality threshold.

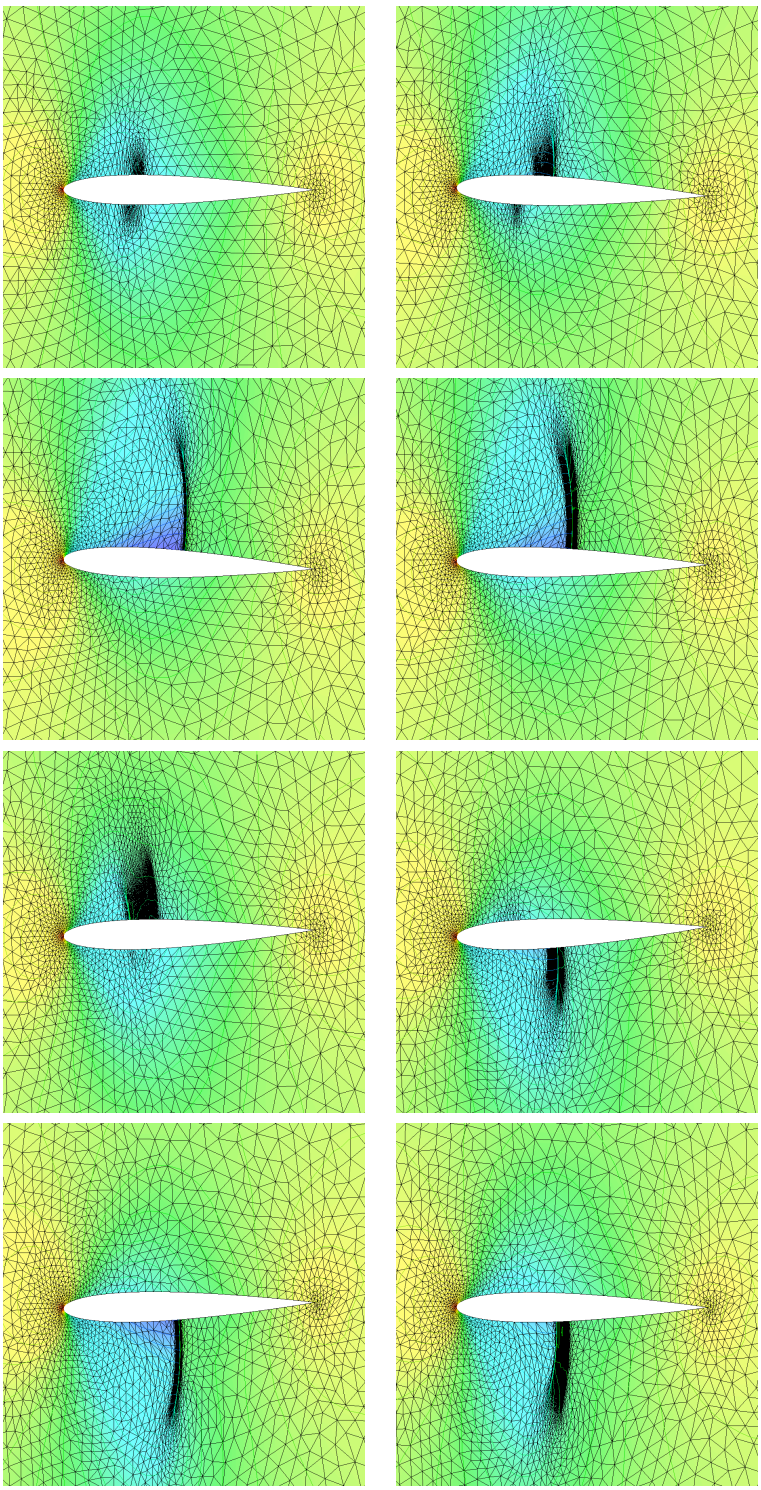


Figure 5.20: Anisotropic unsteady adaptation around a pitching NACA 0012 airfoil when the area surrounding the NACA is rigidified.

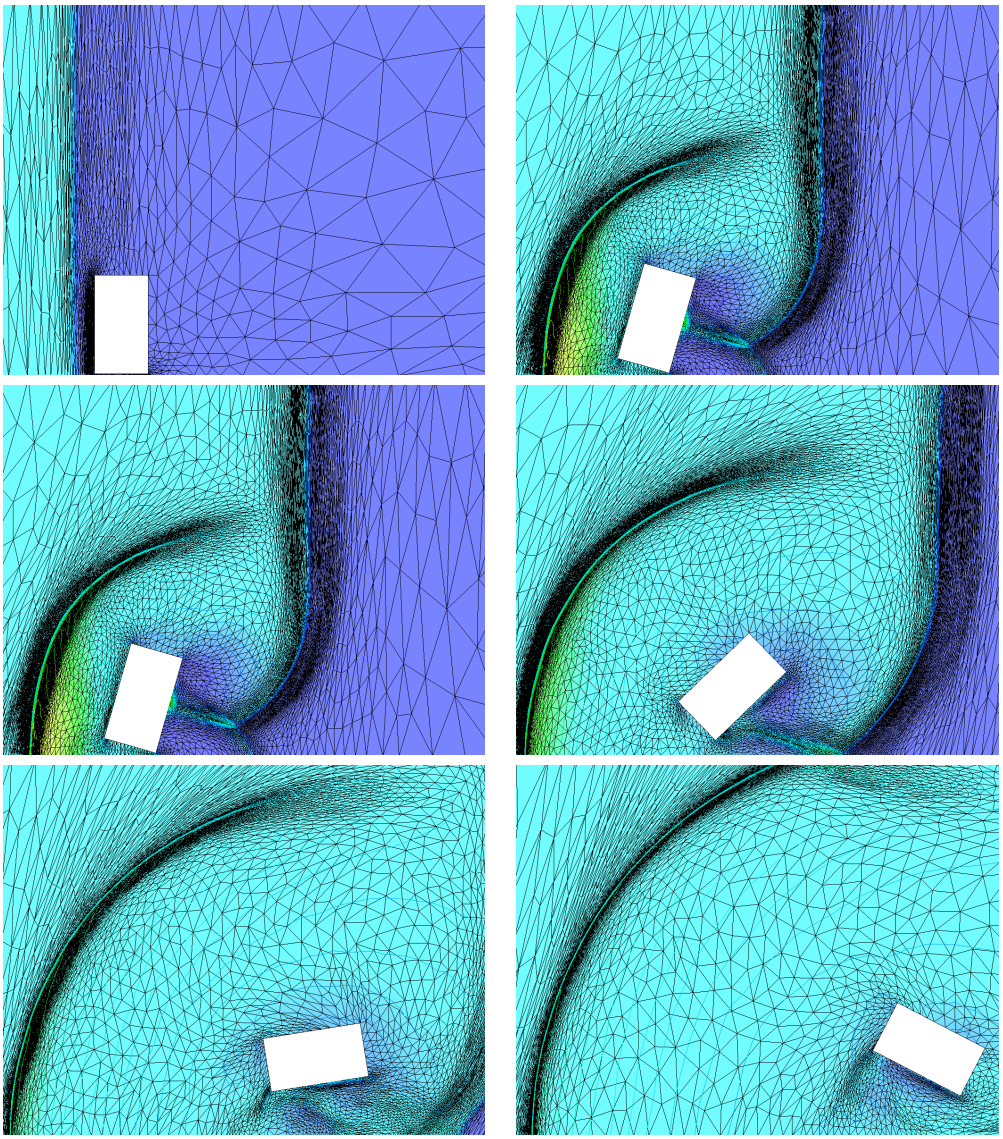


Figure 5.21: Moving mesh anisotropic mesh adaptation on a blast test case. The solution evolves inside the adapted region and the adapted region moves towards the solution. No re-meshing has been needed inside an adaptation sub-interval.

Conclusion

In this second part, a new moving mesh framework, which can be seen at the balance between local and global re-meshing methods, has been introduced. This framework, which allows only a limited number of meshing operations and maintain a constant number of vertices, is consistent with the moving mesh adaptation theory developed in Chapter 3. It has been proved on various two- and three-dimensional numerical examples that, although this strategy only permits swap and vertices relocation, large mesh displacements can be performed. One advantage of this strategy, and not the least, is that it considerably reduces CPU time, both on the solver and on the meshing side.
Second, as our moving mesh strategy allows topology changes, an extension of the classical ALE framework to variable-topology moving meshes in two dimensions has been designed. This formulation, is DGCL with respect to RKSSP(1,1) and has been validated on several two-dimensional tests cases. Incidentally, the Mavriplis-Yang approach, which enables to create ALE time-integration scheme enforcing a DGCL property while maintaining their expected order of accuracy has been introduced. It has been applied to optimal RKSSP schemes. Full ALE and Fluid-Structure Interaction methodologies have also been detailed.
Eventually, the first results coupling multi-scale metric based mesh adaptation and ALE simulations have been presented.

Future work will certainly focus on 3D simulations. Regarding three-dimensional computations, the fixed-topology ALE solver, the re-meshing software, the moving mesh adaptation loop and the 6-DOF Fluid-Structure Interaction resolution are currently implemented. It remains to introduce dynamic topological structures into the solver (tetrahedra and edges) and to implement a three-dimensional conservative scheme to handle topology changes. This last point will certainly be the most difficult as the extension of our two-dimensional changing-topology ALE scheme to three-dimensions is not at all straightforward. A first option will consist in performing local conservative interpolation using the algorithm described in [Alauzet 2010c]. With this compromise, three-dimensional ALE simulations coupling the new moving-mesh global fixed-point algorithm (Chapter II-Chapter III), the moving mesh techniques described in Chapter IV and a new 3D scheme handling topology changes (conservative interpolation or better) may soon be available.
Further validation of our two-dimensional changing-topology ALE scheme, notably as compared to standard and conservative interpolation strategies, would also deserve some time. Finally, its extension to higher-order time-integration schemes, notably the family of RKSSP schemes, would also be a good point.

CONCLUSION AND PERSPECTIVES

Conclusion

This thesis has presented several novelties regarding the application of metric-based mesh adaptation to unsteady simulations, for both fixed and moving computational domains. It has initiated a new research track inside the INRIA-GAMMA project.
The problematics raised by mesh adaptation, unsteadiness and mesh movement have been spotted and analyzed in the details, at all the levels of the resolution loop, *i.e.* solver, meshing and interpolation phases.
Three main improvements of the transient fixed-point mesh adaptation algorithm described in [Alauzet 2007] have been suggested. First, a new space-time error estimator has been designed, which guarantees the equi-distribution of the complete **local** space-time error. On the contrary, the former algorithm uses only the spatial part of the **local** space-time error. This work actually extends the powerful continuous mesh framework described in Chapter 1 to unsteady computations. Second, the control of the **global** space-time interpolation error, *i.e.* on the whole domain and during the whole time frame, has been clarified. Notably, the issue of meshing inconsistencies has been solved using a global normalization constant in the metric. The notion of $\mathbf{L}^p - \mathbf{L}^\infty$ error control has been clarified thanks to a detailed analysis of the intersection procedure. Third, the expression of the optimal metric to handle moving mesh simulations has been deduced. It has lead to a theortically well-founded extension of the fixed-point mesh-adaptation strategy to moving mesh simulations, which has demonstrated its efficiency in practice. This ALE metric takes the movement of the mesh into account in the error estimate.

In the perspective of metric-based multi-scale mesh adaptation, and to remain in accordance with the fixed-point algorithm requirements, a new moving mesh framework has then been introduced, which can be seen as halfway between global re-meshing methods and local re-meshing methods. Global re-meshing methods allow only vertex displacements and ask for the re-meshing of the whole domain as soon as the mesh quality exceeds a prescribed threshold. On the contrary, local re-meshing methods tend to preserve the mesh quality while moving, using various local meshing operations. Our strategy consists in trying to preserve the mesh quality like local re-meshing methods, but using only a limited number of meshing operations: only edge/face swapping and vertex displacements are permitted. This moving strategy enables to keep the number of vertices constant, which is in accordance with moving mesh optimal metric theory developed in Chapter 3. But this strategy is at the same time sufficiently flexible and efficient to handle large displacements involving various type of movements and of geometries, as illustrated in Chapter 4 with various two- and three-dimensional moving mesh simulations. It is also very effective in terms of CPU time as compared to the other recent methods.
Finally, the ALE solver has been adapted to handle this new moving mesh framework. In particular, our moving mesh strategy, which allows mesh topology changes, is more flexible than what the classical ALE framework permits. Therefore, a new changing-topology ALE scheme has been introduced, only in two dimensions for the moment, which is consistent with the fixed-topology framework and enforce a DGCL property. First results coupling metric-based multi-scale mesh adaptation with ALE simulations and implementing this new scheme have been presented in two dimensions.

Future work on this subject will mostly consist in consolidating these advances, notably by emphasizing the gain induced by multi-scale metric-based mesh adaptation for unsteady simulations. To this aim, convergence studies both for fixed and moving-mesh computations will be realized, with the intention

to demonstrate numerically that the expected second-order accuracy (if schemes which are high-order in time and second-order in space are used) is indeed recovered.
Mesh movements techniques will also need some enhancements, notably to remain efficient even in the presence of mesh anisotropy, for instance when moving bodies cross highly-anisotropic areas of the mesh.
A lot of work remains regarding changing-topology ALE schemes, notably regarding their extension to three-dimensional computations. Several ideas are currently under study. Validation and comparison with existing methods (standard and conservative interpolation) also lacks.
Finally, several additional features will be implemented in the ALE solver, such as the handling of body contacts, which requires the implementation of a re-distanciation algorithm. At long term, deformable structures and a fully consistent Fluid-Structure Interaction coupling will also probably be implemented.

Perspectives

A wide variety of perspectives stem from this work.

First of all, the coupling between metric-based mesh adaptation and mesh movement could be reinforced by introducing the metric field inside the moving mesh equation. With such kind of strategy, the movement prescribed to inner vertices would at the same time enforce the prescribed boundary displacement of moving bodies, but would also move inner vertices depending on the interesting features of the flow. In this case, the gradient of the metric field might certainly be involved. Actually, the moving equation could be reformulated under the form of a global optimization problem tending to enforce moving boundary conditions and target metric prescriptions.
More generally, other kinds of moving mesh equations can be investigated. Elasticity-like analogy might be enhanced in a anisotropic mesh adaptation perspective using other behavior laws, for instance linear anisotropic behavior laws, involving a target metric field. Another idea is to use the metric field inside moving mesh equations inspired by electro-magnetic analogy. The "magnetic" analogy might enable to get a better control on the elements orientation depending on the prescribed metric field. Such strategies, if they prove sufficiently efficient, might represent a first step toward moving mesh unsteady adaptation. The idea would be to use mesh movement to adapt the mesh during unsteady simulations with fixed geometry. This is not a completely novel idea but, by enabling swapping operations and by enhancing the moving mesh equation to fit metric-based mesh adaptation, moving mesh unsteady adaptation might be improved in terms of efficiency and accuracy.

Another interesting research perspective is the extension of unsteady goal-oriented metric-based mesh adaptation to moving mesh simulations. Goal-oriented metric-based mesh adaptation has already proved its efficiency for pseudo-steady simulations [Loseille 2010c] and its extension for unsteady problems will certainly get the same successful results within the months to come, [Belme 2010]. It then seems natural to extend these goal-oriented mesh adaptation techniques to moving mesh simulations, using the advances provided by this thesis. An efficient goal-oriented moving-mesh metric-based mesh adaptation strategy would certainly arouse industrials interest. For instance, for Fluid-Structure Interaction, the efforts applied to moving objects could be taken as quantity of interest, thus leading to a better predictions of objects displacements submitted to flow forces (missile release, seat ejection, objects blown by explosions...).

Appendix A

Temporal Hessian computation

A.1 Scheme to compute temporal Hessians

Scheme for $k \geq 1$**.** For $k \geq 1$, the temporal Hessian of the solution at t^k is approximated at each vertex P_i of coordinates vector $\mathbf{x}_i$ by the following Finite Difference scheme:

$$\frac{\partial^2 u_h}{\partial t^2}(P_i,\, t^k) \;\approx\; \frac{\left(\nabla_\tau^C u\right)^{k+\frac{1}{2}}(P_i) - \left(\nabla_\tau^C u\right)^{k-\frac{1}{2}}(P_i)}{\dfrac{t^{k+1}-t^{k-1}}{2}} \approx \frac{2}{t^{k+1}-t^{k-1}}\left(\frac{u_i^{k+1}-u_i^k}{t^{k+1}-t^k} - \frac{u_i^k - u_i^{k-1}}{t^k - t^{k-1}}\right)$$

$$\frac{\partial^2 u_h}{\partial t^2}(\mathbf{x}_i,\, t^k) \approx 2\,\frac{\left(t^k - t^{k-1}\right)\,u_i^{k+1} \;+\; \left(t^{k+1}-t^k\right)\,u_i^{k-1} \;-\; \left(t^{k+1}-t^{k-1}\right)\,u_i^k}{\left(t^k - t^{k-1}\right)\left(t^{k+1}-t^k\right)\left(t^{k+1}-t^{k-1}\right)}\,. \tag{A.1}$$

This means that the solution obtained at previous iteration u^{k-1} and current iteration u^k must be stored when computing second-order derivatives in time. Fortunately, u^k is already stored as it is used for Runge-Kutta time advancing. Previous and current time steps must also be stored.

Scheme for $k = 1$**.** The above computation cannot be used at initial time t^0. First, the following approximation is used:

$$\frac{\partial u_h}{\partial t}\left(P_i, t^0\right) \approx \left(\nabla_\tau^D u\right)_i^0 \approx \frac{u_i^1 - u_i^0}{t^1 - t^0}\,,$$

and the gradient at t^1 is approximated using a $\mathbf{L^2}$ projection:

$$\begin{aligned}\frac{\partial u_h}{\partial t}\left(P_i, t^1\right) \;&\approx \frac{\left(t^1 - t^0\right)\,\left(\nabla_\tau^C u\right)_i^{\frac{1}{2}} \;+\; \left(t^2 - t^1\right)\,\left(\nabla_\tau^C u\right)_i^{\frac{3}{2}}}{\left(t^2 - t^1\right) \;+\; \left(t^1 - t^0\right)} \\ &= \frac{\left(t^1 - t^0\right)\,\dfrac{u_i^1 - u_i^0}{t^1 - t^0} \;+\; \left(t^2 - t^1\right)\,\dfrac{u_i^2 - u_i^1}{t^2 - t^1}}{\left(t^2 - t^1\right) \;+\; \left(t^1 - t^0\right)} = \frac{u_i^2 - u_i^0}{t^2 - t^0}\,.\end{aligned}$$

The temporal Hessian is then approximated by:

$$\frac{\partial^2 u_h}{\partial t^2}(P_i,\, t^0) \approx \frac{u_t\left(P_i, t^1\right) - u_t\left(P_i, t^0\right)}{t^1 - t^0} \approx \frac{\left(t^1 - t^0\right)\,u_i^2 \;-\; \left(t^2 - t^0\right)\,u_i^1 \;+\; \left(t^2 - t^1\right)\,u_i^0}{\left(t^2 - t^0\right)\left(t^1 - t^0\right)^2}\,. \tag{A.2}$$

Clement interpolation. If the previous approximation is used like this independently for each vertex, the numerical temporal Hessian obtained is non-smooth. To smoothen this approximation, a $\mathbb{P}^0$ Clement-interpolator [Clément 1975] is used.

$$\Pi_0\left(\frac{\partial^2 u_h}{\partial t^2}\right)_{|C_i} = \frac{\displaystyle\sum_{K\,\in\,\mathrm{Ball}(i)} |K| \int_K \frac{\partial^2 u_h}{\partial t^2}}{\displaystyle\sum_{K\,\in\,\mathrm{Ball}(i)} |K|}\,.$$

Besides, u_h is piecewise linear and can be decomposed on the $\mathbf{P}^1$ Finite Element basis $(\phi_i)_{i \in [\![1, N_v]\!]}$ as:

$$u_h(\mathbf{x}, t) = \sum_{i=1}^{N_v} u_i(t)\, \phi_i(\mathbf{x})\,.$$

Therefore, as the mesh is assumed to be fixed in time, the temporal Hessian of u_h is equal to:

$$\frac{\partial^2 u_h}{\partial t^2}(\mathbf{x}, t) = \sum_{i=1}^{N_v} \frac{\mathrm{d}^2 u_i}{\mathrm{d}t^2}(t)\, \phi_i(\mathbf{x})\,,$$

and is also piecewise linear. As a consequence, the integral of $\dfrac{\partial^2 u_h}{\partial t^2}$ on an element K is computed using a first-order Gauss quadrature formula:

$$\int_K \frac{\partial^2 u_h}{\partial t^2}(\mathbf{x}, t)\,\mathrm{d}\mathbf{x} = \frac{|K|}{n}\left(\sum_{P_j \in K} \frac{\partial^2 u_h}{\partial t^2}(P_j, t)\right) = |K| \frac{\partial^2 u_h}{\partial t^2}_{|K}(t)\,. \tag{A.3}$$

Finally, temporal Hessian value at vertex P_i is approximated by:

$$\Pi_0\left(\frac{\partial^2 u_h}{\partial t^2}\right)_{|C_i}(t^k) = \frac{1}{n} \frac{\sum\limits_{K \in \mathrm{Ball}(i)} |K| \dfrac{\partial^2 u_h}{\partial t^2}_{|K}(t^k)}{\sum\limits_{K \in \mathrm{Ball}(i)} |K|}\,,$$

and $\dfrac{\partial^2 u_h}{\partial t^2}(\mathbf{x}_l, t^k)$ in Formula (A.3) is evaluated by Finite Difference Scheme (A.1-A.2).

A.2 Numerical illustration

Figure A.3 shows the time evolution of u_{tt} at six different vertices represented on Figure A.1. In Figure A.1, peak values of the temporal Hessian correspond to the passing of one of a shock wave through the considered point.

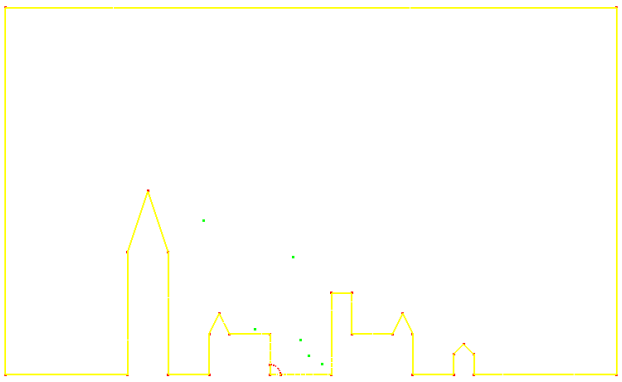


Figure A.1: The six vertices for which the time evolution of u_{tt} has been computed.

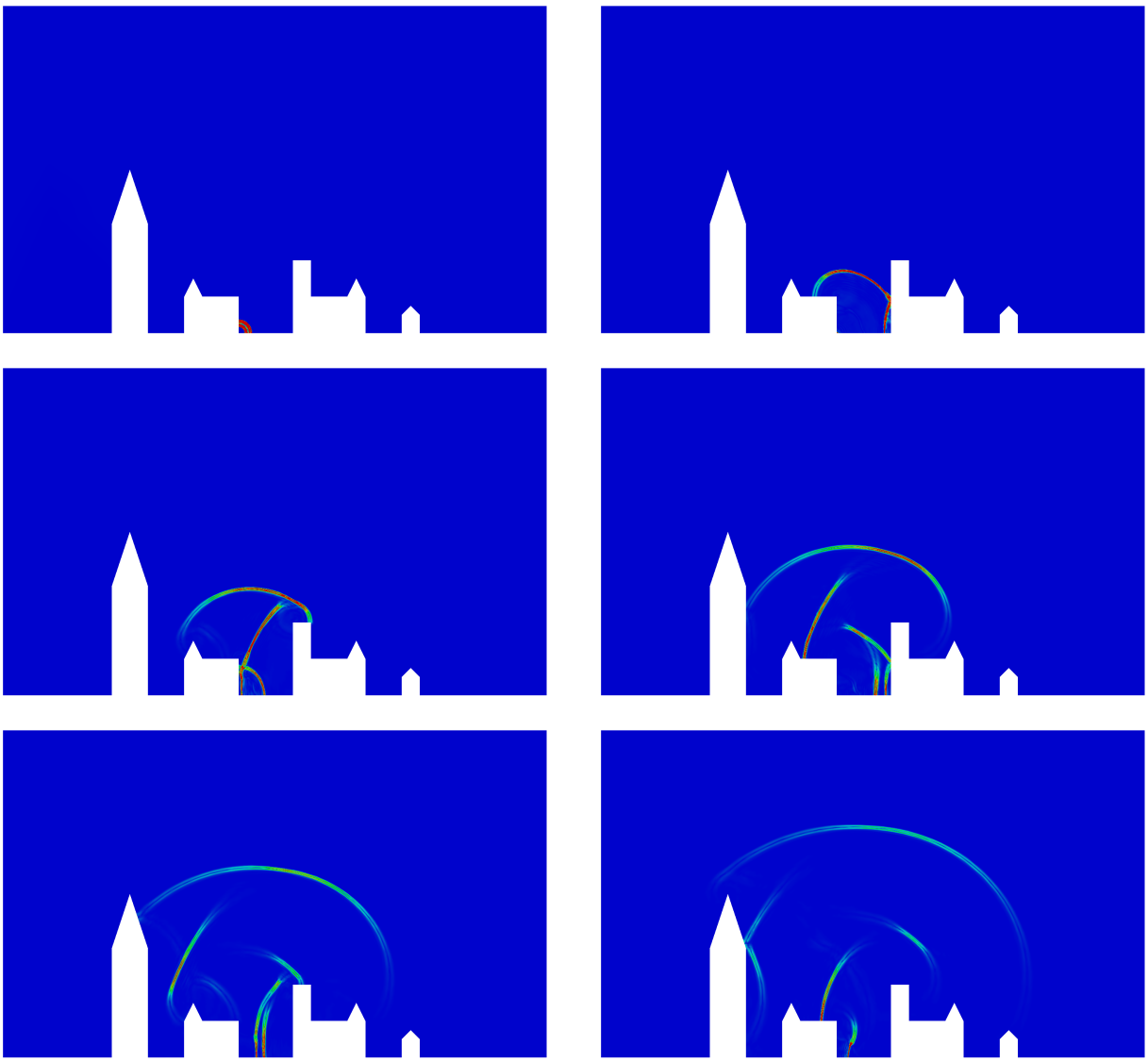


Figure A.2: From left to right and up to bottom, numerical temporal Hessian computed for a two-dimensional adaptive city blast simulation at a-dimensioned times $t = 0.03$, 1, 2, 3, 4 and $t = 5$.

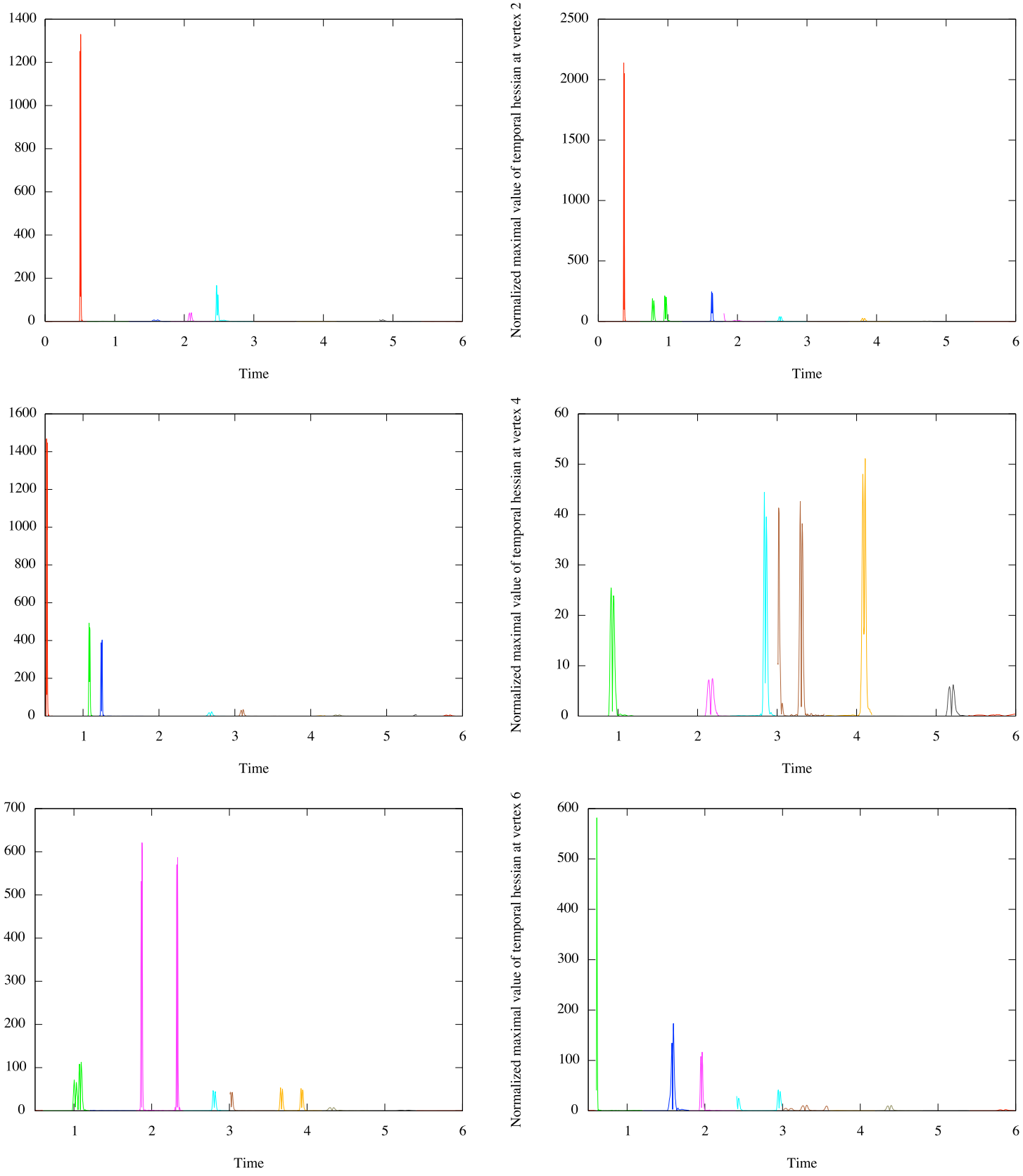


Figure A.3: Temporal Hessian at the six different points during a two dimensional adaptive city blast simulation.

Appendix B

The elasticity system resolution

B.1 Problem formulation

We solve the three-dimensional *linear elasticity equation*:

$$\operatorname{div} \mathcal{S}(\mathcal{E}) = 0\,, \tag{B.1}$$

where $\mathcal{S}$ and $\mathcal{E}$ are respectively the constiffnesst and the deformation tensors. The deformation tensor is defined by the compatibility relation:

$$\mathcal{E} = \frac{\nabla \mathbf{d} + \nabla \mathbf{d}^T}{2}\,,$$

with $\mathbf{d} = (d_1, d_2, d_3)^T$ the Lagrangian displacement of the vertices. The constiffnesst tensor follows the linear elasticity behavior law:

$$\mathcal{S}(\mathcal{E}) = \lambda \operatorname{trace}(\mathcal{E}) I_3 + 2\,\mu\,\mathcal{E}\,, \tag{B.2}$$

where λ and μ are the Lamé coefficients. The Lamé coefficients can also be expressed as a function of the Young modulus E and the Poisson ratio ν:

$$\begin{aligned} \mathrm{E} &= \frac{\mu\,(3\lambda + 2\mu)}{\lambda + \mu} && \text{and} && \nu = \frac{\lambda}{2\,(\lambda + \mu)} \\ \text{or} \qquad \lambda &= \frac{\mathrm{E}\,\nu}{(1+\nu)(1-2\nu)} && \text{and} && \mu = \frac{\mathrm{E}}{2(1+\nu)}\,. \end{aligned}$$

The following properties regarding the elasticity coefficients are deduced from thermodynamic considerations:

$$\begin{aligned} \lambda + \frac{2}{3}\mu \geq 0 \qquad & \text{and} \qquad \mu > 0 \\ \mathrm{E} > 0 \qquad & \text{and} \qquad -1 < \nu < \frac{1}{2}\,. \end{aligned}$$

Physically, it is very difficult to give a meaning to λ. μ, which is also named "shear modulus" or "Coulomb modulus", is defined as the ratio of the shear stress to the shear strain and is generally expressed in GPa.

Young modulus E is a measure of the stiffness of an isotropic elastic material. It is the ration between the stress and the stretch of a solid particle in a given spatial directional. In the case of an isotropic material, which is the assumption made in this Thesis, it is the same in all directions. The higher the Young modulus, the stiffer the material. For instance, natural rubber has a small stiffness, comprised between 0.01 and 0.1 GPa (10^9 Pascal). Aluminum and steel have a Young modulus of 70 and 210 GPa, respectively, in standard conditions. For diamond, the value is 1220 GPa. For short-glass-fiber-reinforced polyamides, E lowers to 10 GPa [Launay 2011].

The Poisson ratio ν is the ratio, measured on a stretched solid particle, of the contraction or transverse stiffness (perpendicular to the applied load), to the extension or axial stiffness (in the direction of the applied load). When a sample cube of a material is stretched in one direction, it tends to contract (or

theoretically, expand if $\nu < 0$) in the other two directions perpendicular to the load direction. Conversely, when a sample of material is compressed in one direction, it tends to expand (or theoretically, contract) in the other two directions. This phenomenon is called the Poisson effect. The Poisson ratio ν is a measure of this Poisson effect. Even if negative Poisson coefficients are tolerated by the thermodynamics theory, it has never been observed in practice for any homogeneous material. Except for some specific structural materials, like sponges or trusses which can not actually be considered as homogeneous, the value of nu is generally strictly greater than 0. One of the smallest value of ν is obtained for the beryllium (Be), one of the component of emeralds with $\nu \approx 0.03$. A perfectly incompressible material deformed elastically at small stiffness would have a Poisson ratio exactly equal to 0.5. It is the case, for example, of "perfect" rubber. Most steels, when used within their design limits (before yield), exhibit values of about 0.3, increasing to 0.5 for post-yield deformation (which occurs largely at constant volume). This is due to the fact that plastic deformations leads to local incompressibility properties. Actually, the apparent Poisson coefficient tends to 0.5 but the intrinsic one keeps the same value.

B.2 Finite-Element discretization

We note:

$$V = H^1(\Omega) = \{u \in L^2(\Omega) \text{ such that } \nabla_{\boldsymbol{x}} u \in L^2(\Omega) \text{ in the distribution sense } \}$$

$$V_0 = H_0^1(\Omega) = \{u \in H^1(\Omega) \text{ such that } u_{|\partial\Omega} = 0\}$$

The variational formulation of the linear elasticity equation (B.1) is:

$$\text{Find } \boldsymbol{d} \in V^n\,,\ \int_\Omega \mathcal{S}(\mathcal{E}) : \nabla \boldsymbol{v}\, \mathrm{d}\Omega + BdyTerms = \int_\Omega \mathbf{f}\boldsymbol{v}\,, \quad \forall \boldsymbol{v} \in (V_0)^n \tag{B.3}$$

where the operator ":" is understood as $A : B = \sum_{i,j} a_{ij}\, b_{ij}$. Test functions $\boldsymbol{v}$ are chosen in $(V_0)^n$ because we only consider Dirichlet boundary conditions, i.e we always impose the displacement on the boundary and never impose the stiffness on the boundary. Thus, the boundary terms $BdyTerms$ cancel.

2D case Let us first deal with the two dimensional case. Under matrix form, we have:

$$\mathcal{S}(\mathcal{E}) = \begin{pmatrix} \lambda\,(\dfrac{\partial d_1}{\partial x} + \dfrac{\partial d_2}{\partial y}) + 2\mu \dfrac{\partial d_1}{\partial x} & \mu\,(\dfrac{\partial d_1}{\partial y} + \dfrac{\partial d_2}{\partial x}) \\ \mu\,(\dfrac{\partial d_1}{\partial y} + \dfrac{\partial d_2}{\partial x}) & \lambda\,(\dfrac{\partial d_1}{\partial x} + \dfrac{\partial d_2}{\partial y}) + 2\mu \dfrac{\partial d_2}{\partial y} \end{pmatrix}.$$

We choose a polynomial approximation space:

$$V_h = \{u \in V \text{ such that } u_{|\partial\Omega} \in P^k\,, \quad \forall\, K \in \mathcal{H}\}$$

and we set $s = \lambda + 2\mu$. Then, the discrete variational formulation reads:

$$\text{Find } \boldsymbol{d}_h = (d_1,\, d_2)^T \in V_h^2 \text{ such that}\,, \quad \forall \boldsymbol{v}_h = (v_1,\, v_2)^T \in V_h^2 :$$

$$\begin{aligned} \int_{\Omega_h} & \left(s\frac{\partial d_1}{\partial x} + \lambda \frac{\partial d_2}{\partial y}\right)\frac{\partial v_1}{\partial x} + \mu\left(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x}\right)\frac{\partial v_1}{\partial y} \\ + & \mu\left(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x}\right)\frac{\partial v_2}{\partial x} + \left(\lambda\frac{\partial d_1}{\partial x} + s\frac{\partial d_2}{\partial y}\right)\frac{\partial v_2}{\partial y} = \int_{\Omega_h} \mathbf{f}_h \boldsymbol{v}_h\,, \end{aligned}$$

From now on, we note $d_{J,k}$ the unknown value of the k^{th} component of displacement $\boldsymbol{d}_h$ at vertex P_J. We then decompose each component of displacement $\boldsymbol{d}_h$ on the Finite Element basis $(\varphi_J)_{J\in[\![1,N_v]\!]}$:

$$\boldsymbol{d}_h(\mathbf{x}) = \left(\sum_J d_{J,1}\varphi_J(\mathbf{x})\ ,\ \sum_J d_{J,2}\varphi_J(\mathbf{x})\right)^T \quad \text{and} \quad \frac{\partial \boldsymbol{d}_h}{\partial x_k}(\mathbf{x}) = \left(\sum_J d_{J,1}\frac{\partial\varphi_J}{\partial x_k}(\mathbf{x})\ ,\ \sum_J d_{J,2}\frac{\partial\varphi_J}{\partial x_k}(\mathbf{x})\right)^T .$$

We obtain the following discrete approximate problem:

$$\text{Find } \Xi = \left\{(d_{I,1}, d_{I,2})^T\right\}_{I=1\dots N} \text{ such that, } \forall \boldsymbol{v}_h = (v_1, v_2)^T \in V_h^2\text{:}$$

$$\begin{aligned}
&\sum_J d_{J,1}\int_{\Omega_h}\left(s\frac{\partial\varphi_J}{\partial x}\frac{\partial v_1}{\partial x} + \mu\frac{\partial\varphi_J}{\partial y}\frac{\partial v_1}{\partial y} + \mu\frac{\partial\varphi_J}{\partial y}\frac{\partial v_2}{\partial x} + \lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial v_2}{\partial y}\right)\\
+ &\sum_J d_{J,2}\int_{\Omega_h}\left(\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial v_1}{\partial x} + \mu\frac{\partial\varphi_J}{\partial x}\frac{\partial v_1}{\partial y} + \mu\frac{\partial\varphi_J}{\partial x}\frac{\partial v_2}{\partial x} + s\frac{\partial\varphi_J}{\partial y}\frac{\partial v_2}{\partial y}\right)\\
= &\int_{\Omega_h}\mathbf{f}_h\boldsymbol{v}_h
\end{aligned}$$

We now split the discrete variational equation into the following system by choosing respectively as test functions $\boldsymbol{v} = (\varphi_I, 0)^T$ and $\boldsymbol{v} = (0, \varphi_I)^T$, with I the index of a vertex which is not on the boundary, so that φ_I belongs to V_h:

$$\begin{aligned}
\sum_J d_{J,1}\int_{\Omega_h}\left(s\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x} + \mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}\right) &+ \sum_J d_{J,2}\int_{\Omega_h}\left(\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial x} + \mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y}\right)\\
&= \sum_J f_{J,1}\int_{\Omega_h}\varphi_J\varphi_I\,, \quad \text{for } (\varphi_I, 0)^T \in V_h^2\\
\sum_J d_{J,1}\int_{\Omega_h}\left(\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial x} + \lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y}\right) &+ \sum_J d_{J,2}\int_{\Omega_h}\left(\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x} + s\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}\right)\\
&= \sum_J f_{J,2}\int_{\Omega_h}\varphi_J\varphi_I\,, \quad \text{for } (0, \varphi_I)^T \in V_h^2
\end{aligned}$$

Therefore, we have to solve the following linear system:

$$A\Xi = F$$

where A is a block matrix having 2×2 block A_{IJ} at block index (I, J) given by:

$$A_{IJ} = \begin{pmatrix} \displaystyle\int_{\Omega_h} s\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x} + \mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y} & \displaystyle\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial x} + \mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y}\\ \displaystyle\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial x} + \lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y} & \displaystyle\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x} + s\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}\end{pmatrix},$$

and F is a block vector having F_I as 2-vector at index I:

$$F_I = \begin{pmatrix} \displaystyle\sum_J f_{J,1}\int_{\Omega_h}\varphi_J\varphi_I\\ \displaystyle\sum_J f_{J,2}\int_{\Omega_h}\varphi_J\varphi_I\end{pmatrix}.$$

$$\begin{pmatrix} A_{11} & \vdots & & \\ & \vdots & & \\ \cdots & A_{IJ} & \cdots & \cdots \\ & \vdots & & \end{pmatrix} \begin{pmatrix} d_{1,1} \\ d_{1,2} \\ \vdots \\ d_{J,1} \\ d_{J,2} \\ \vdots \end{pmatrix} = \begin{pmatrix} F_{1,1} \\ F_{1,2} \\ \vdots \\ F_{I,1} \\ F_{I,2} \\ \vdots \end{pmatrix}$$

Now, let us detail how to compute the integral terms in block A_{IJ}. Classically, in the FEM method, the shape functions are defined element by element as polynomial functions. As the shape functions are designed such that a vertex only interacts with the vertices of its ball (see Conventions page 3), the matrix assembly is done in one shot with a loop on triangles. For example, the term $\int_{\Omega_h} s \frac{\partial \varphi_J}{\partial x}\frac{\partial \varphi_I}{\partial x}$ in the A_{IJ} matrix is computed as:

$$\int_{\Omega_h} s \frac{\partial \varphi_J}{\partial x}\frac{\partial \varphi_I}{\partial x} = \sum_{K\in\Omega_h} \int_K s \frac{\partial \varphi_J}{\partial x}\frac{\partial \varphi_I}{\partial x} = \sum_{K\in \mathrm{Ball}(P_I)} \int_K s \frac{\partial \varphi_J}{\partial x}\frac{\partial \varphi_I}{\partial x}$$

In the most general case of a P^k approximation, we use a mapping from the reference element $\hat{K}$ onto the current element to compute the integrals on K in A_{IJ}. Chosen reference triangle is $\hat{K} = (\hat{P}_0, \hat{P}_1, \hat{P}_2)$, with $\hat{P}_0 = (0,0)$, $\hat{P}_1 = (1,0)$ and $\hat{P}_2 = (0,1)$. In two dimensions, the Jacobian matrix of the mapping from the reference triangle to the current triangle $K = (P_0, P_1, P_2)$ is given by:

$$B_K = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{e_2} & \boldsymbol{e_1} \end{pmatrix}$$

$$B_K^{-T} = \frac{1}{\det B_K} \begin{pmatrix} y_2 - y_0 & y_0 - y_1 \\ x_0 - x_2 & x_1 - x_0 \end{pmatrix} = \frac{1}{2|K|} \begin{pmatrix} \overline{\boldsymbol{\eta_1}} & \overline{\boldsymbol{\eta_2}} \end{pmatrix}$$

We note with a hat the quantities defined on the reference triangle. If $f = f(x)$, then we change the variable x into $\hat{x}$ thanks to the mapping and we thus have the differentiation formula:

$$\nabla_{\hat{x}} \hat{f}(\hat{x}) = B_K^T \nabla_x f(x)\,.$$

Now, we can use the above consideration to compute terms of the form $\int_K \frac{\partial \varphi_{J,j}}{\partial x_l}\frac{\partial \varphi_{I,m}}{\partial x_n}$ with $1 \le k, l, m, n \le 2$ by performing a variable change in the integral:

$$\int_K \frac{\partial \varphi_{J,k}}{\partial x_l}\frac{\partial \varphi_{I,m}}{\partial x_n}\,\mathrm{d}x = \det B_K \int_{\hat{K}} \left[B_K^{-T}\nabla_{\hat{x}}\hat{\varphi}_{J,k}\right]_l \left[B_K^{-T}\nabla_{\hat{x}}\hat{\varphi}_{I,m}\right]_n \mathrm{d}\hat{x}$$

(note that vertices P_I and P_J belong to K, otherwise the integral is zero)
In the case of a $\mathbb{P}^1$ approximation of $\mathbf{d}$, all this simplifies a lot. Indeed, the shape functions and their gradients on the reference triangle $\hat{K}$ are given by:

$$\begin{array}{ll} \hat{\varphi}_0(x,y) = 1 - x - y & \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}_0 = \begin{pmatrix} -1, & -1 \end{pmatrix}^T \\ \hat{\varphi}_1(x,y) = x & \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}_1 = \begin{pmatrix} 1, & 0 \end{pmatrix}^T \\ \hat{\varphi}_2(x,y) = y & \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}_2 = \begin{pmatrix} 0, & 1 \end{pmatrix}^T \end{array} \tag{B.4}$$

and remembering that $\overline{\boldsymbol{\eta_0}} + \overline{\boldsymbol{\eta_1}} + \overline{\boldsymbol{\eta_2}} = \mathbf{0}$, their gradients on the current element are given by:

$$\begin{aligned}
& \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}(\hat{\boldsymbol{x}}) = B_K^T \nabla_{\boldsymbol{x}}\varphi(\boldsymbol{x}) \\
\Leftrightarrow \quad & \nabla_{\boldsymbol{x}}\varphi(\boldsymbol{x}) = B_K^{-T} \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}(\hat{\boldsymbol{x}}) \\
\Rightarrow \quad & \begin{cases}
\nabla_{\boldsymbol{x}}\varphi_0(\boldsymbol{x}) = \dfrac{1}{\det B_K} \begin{pmatrix} \overline{\boldsymbol{\eta_1}} & \overline{\boldsymbol{\eta_2}} \end{pmatrix} \begin{pmatrix} -1 \\ -1 \end{pmatrix} & = \dfrac{1}{2\,|K|}\overline{\boldsymbol{\eta_0}} \\
\nabla_{\boldsymbol{x}}\varphi_1(\boldsymbol{x}) = \dfrac{1}{\det B_K} \begin{pmatrix} \overline{\boldsymbol{\eta_1}} & \overline{\boldsymbol{\eta_2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} & = \dfrac{1}{2\,|K|}\overline{\boldsymbol{\eta_1}} \\
\nabla_{\boldsymbol{x}}\varphi_2(\boldsymbol{x}) = \dfrac{1}{\det B_K} \begin{pmatrix} \overline{\boldsymbol{\eta_1}} & \overline{\boldsymbol{\eta_2}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} & = \dfrac{1}{2\,|K|}\overline{\boldsymbol{\eta_2}}
\end{cases}
\end{aligned} \tag{B.5}$$

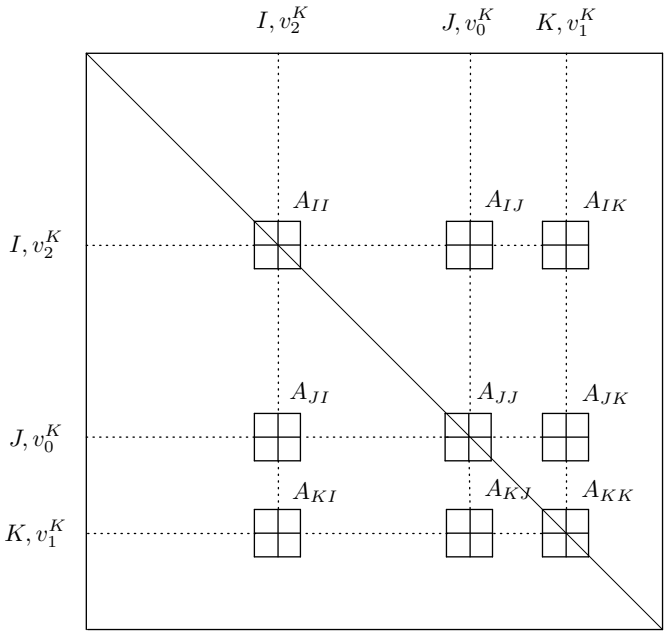
We immediately see that in the case of a $\mathbb{P}_1$ approximation, there is no need to calculate B_K^{-T} for each triangle to perform the integrals. Indeed, the gradients of the shape functions are constant on each triangle and the integral computation is trivially given by:

$$\int_K \frac{\partial\varphi_J}{\partial x_l}\frac{\partial\varphi_I}{\partial x_n}\mathrm{d}x = |K|\frac{\partial\varphi_J}{\partial x_l}\frac{\partial\varphi_I}{\partial x_n}$$

For example, if vertex I and J have respectively $I = 2$ and $J = 0$ as local index in triangle K, we have:

$$\int_K \frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y}\mathrm{d}x = |K|\frac{\partial\varphi_0}{\partial x}\frac{\partial\varphi_2}{\partial y} = \frac{|K|}{4|K|^2}(\overline{\boldsymbol{\eta_0}})_x(\overline{\boldsymbol{\eta_2}})_y$$

Thus, triangle K contributes to the following sub-blocks:



Note that the elasticity matrix is symmetric and that a triangle K contributes to $9 \times 4 = 36$ boxes of the matrix (but we only need to fill $3 \times 4 + 3 \times 3 = 21$ boxes as the matrix is symmetric).

3D case The stiffness tensor reads:

$$\mathcal{S}(\mathcal{E}) = \begin{pmatrix} s\,\frac{\partial d_1}{\partial x} + \lambda\,\frac{\partial d_2}{\partial y} + \lambda\,\frac{\partial d_3}{\partial z} & \mu\,(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x}) & \mu\,(\frac{\partial d_1}{\partial z} + \frac{\partial d_3}{\partial x}) \\ \mu\,(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x}) & \lambda\,\frac{\partial d_1}{\partial x} + s\,\frac{\partial d_2}{\partial y} + \lambda\,\frac{\partial d_3}{\partial z} & \mu\,(\frac{\partial d_2}{\partial z} + \frac{\partial d_3}{\partial y}) \\ \mu\,(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x}) & \mu\,(\frac{\partial d_2}{\partial z} + \frac{\partial d_3}{\partial y}) & \lambda\,\frac{\partial d_1}{\partial x} + \lambda\,\frac{\partial d_2}{\partial y} + s\,\frac{\partial d_3}{\partial z} \end{pmatrix}$$

Then, the variational formulation is given by:

Find $\boldsymbol{d}_h = (d_1,\, d_2,\, d_3)^T \in V_h^3$ such that, $\quad \forall\, \boldsymbol{v}_h = (v_1,\, v_2,\, v_3)^T \in V_h^3$:

$$\int_{\Omega_h} \begin{pmatrix} s\,\frac{\partial d_1}{\partial x} + \lambda\,\frac{\partial d_2}{\partial y} + \lambda\,\frac{\partial d_3}{\partial z} & \mu\,(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x}) & \mu\,(\frac{\partial d_1}{\partial z} + \frac{\partial d_3}{\partial x}) \\ \mu\,(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x}) & \lambda\,\frac{\partial d_1}{\partial x} + s\,\frac{\partial d_2}{\partial y} + \lambda\,\frac{\partial d_3}{\partial z} & \mu\,(\frac{\partial d_2}{\partial z} + \frac{\partial d_3}{\partial y}) \\ \mu\,(\frac{\partial d_1}{\partial y} + \frac{\partial d_2}{\partial x}) & \mu\,(\frac{\partial d_2}{\partial z} + \frac{\partial d_3}{\partial y}) & \lambda\,\frac{\partial d_1}{\partial x} + \lambda\,\frac{\partial d_2}{\partial y} + s\,\frac{\partial d_3}{\partial z} \end{pmatrix} : \begin{pmatrix} \frac{\partial v_1}{\partial x} & \frac{\partial v_1}{\partial y} & \frac{\partial v_1}{\partial z} \\ \frac{\partial v_2}{\partial x} & \frac{\partial v_2}{\partial y} & \frac{\partial v_2}{\partial z} \\ \frac{\partial v_3}{\partial x} & \frac{\partial v_3}{\partial y} & \frac{\partial v_3}{\partial z} \end{pmatrix}$$

$$= \int_{\Omega_h} \mathbf{f_h} \boldsymbol{v}_h$$

By developing the above expression and reorganizing the different terms, we get:

$$\begin{aligned} & \int_{\Omega_h} s\frac{\partial d_1}{\partial x}\frac{\partial v_1}{\partial x} + \int_{\Omega_h} \mu\frac{\partial d_1}{\partial y}\frac{\partial v_1}{\partial y} + \int_{\Omega_h} \mu\frac{\partial d_1}{\partial z}\frac{\partial v_1}{\partial z} \quad + \int_{\Omega_h} \mu\frac{\partial d_1}{\partial y}\frac{\partial v_2}{\partial x} + \int_{\Omega_h} \lambda\frac{\partial d_1}{\partial x}\frac{\partial v_2}{\partial y} \\ & \qquad\qquad + \int_{\Omega_h} \mu\frac{\partial d_1}{\partial z}\frac{\partial v_3}{\partial x} + \int_{\Omega_h} \lambda\frac{\partial d_1}{\partial x}\frac{\partial v_3}{\partial z} \\ + & \int_{\Omega_h} \lambda\frac{\partial d_2}{\partial y}\frac{\partial v_1}{\partial x} + \int_{\Omega_h} \mu\frac{\partial d_2}{\partial x}\frac{\partial v_1}{\partial y} \qquad + \int_{\Omega_h} \mu\frac{\partial d_2}{\partial x}\frac{\partial v_2}{\partial x} + \int_{\Omega_h} s\frac{\partial d_2}{\partial y}\frac{\partial v_2}{\partial y} + \int_{\Omega_h} \mu\frac{\partial d_2}{\partial z}\frac{\partial v_2}{\partial z} \\ & \qquad\qquad + \int_{\Omega_h} \mu\frac{\partial d_2}{\partial z}\frac{\partial v_3}{\partial y} + \int_{\Omega_h} \lambda\frac{\partial d_2}{\partial y}\frac{\partial v_3}{\partial z} \\ + & \int_{\Omega_h} \mu\frac{\partial d_2}{\partial z}\frac{\partial v_1}{\partial y} + \int_{\Omega_h} \lambda\frac{\partial d_2}{\partial y}\frac{\partial v_1}{\partial z} \qquad + \int_{\Omega_h} \lambda\frac{\partial d_3}{\partial z}\frac{\partial v_2}{\partial y} + \int_{\Omega_h} \mu\frac{\partial d_3}{\partial y}\frac{\partial v_2}{\partial z} \\ & \qquad\qquad + \int_{\Omega_h} \mu\frac{\partial d_3}{\partial x}\frac{\partial v_3}{\partial x} + \int_{\Omega_h} \mu\frac{\partial d_3}{\partial y}\frac{\partial v_3}{\partial y} + \int_{\Omega_h} s\frac{\partial d_3}{\partial z}\frac{\partial v_3}{\partial z} \\ = & \int_{\Omega_h} \mathbf{f}_h \boldsymbol{v}_h\,, \qquad\qquad \forall\, \boldsymbol{v}_h = (v_1,\, v_2,\, v_3) \in V_h^3\,. \end{aligned}$$

From now on, we note $d_{J,k}$ the unknown value of the k^{th} component of displacement $\boldsymbol{d}_h$ at vertex P_J. We decompose each component of displacement $\boldsymbol{d}_h$ on the Finite Element basis:

$$\begin{aligned} \boldsymbol{d}_h(\mathbf{x}) &= (\textstyle\sum_J d_{J,1}\varphi_J(\mathbf{x})\,,\ \sum_J d_{J,2}\varphi_J(\mathbf{x})\,,\ \sum_J d_{J,3}\varphi_J(\mathbf{x}))^T\,, \\ \boldsymbol{f}_h(\mathbf{x}) &= (\textstyle\sum_J f_{J,1}\varphi_J(\mathbf{x})\,,\ \sum_J f_{J,2}\varphi_J(\mathbf{x})\,,\ \sum_J f_{J,3}\varphi_J(\mathbf{x}))^T\,. \end{aligned}$$

We then reintroduce this decomposition in B.2:

$$\begin{aligned}
&\sum_J d_{J,1}\left(\int_{\Omega_h} s\frac{\partial\varphi_J}{\partial x}\frac{\partial v_1}{\partial x}+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial v_1}{\partial y}+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial v_1}{\partial z}\right.\\
&\qquad\left.+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial v_2}{\partial x}+\int_{\Omega_h}\lambda\frac{\partial\,\varphi_J}{\partial x}\frac{\partial v_2}{\partial y}\right.\\
&\qquad\left.+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial v_3}{\partial x}+\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial v_3}{\partial z}\right)\\
+\;&\sum_J d_{J,2}\left(\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial v_1}{\partial x}+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial v_1}{\partial y}\right.\\
&\qquad\left.+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial v_2}{\partial x}+\int_{\Omega_h}s\frac{\partial\varphi_J}{\partial y}\frac{\partial v_2}{\partial y}+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial v_2}{\partial z}\right.\\
&\qquad\left.+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial v_3}{\partial y}+\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial v_3}{\partial z}\right)\\
+\;&\sum_J d_{J,3}\left(\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial v_1}{\partial y}+\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial v_1}{\partial z}\right.\\
&\qquad\left.+\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial z}\frac{\partial v_2}{\partial y}+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial v_2}{\partial z}\right.\\
&\qquad\left.+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial v_3}{\partial x}+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial v_3}{\partial y}+\int_{\Omega_h}s\frac{\partial\varphi_J}{\partial z}\frac{\partial v_3}{\partial z}\right)\\
=\;&\sum_J f_{J,1}\int_{\Omega_h}\varphi_J v_1\;+\;\sum_J f_{J,2}\int_{\Omega_h}\varphi_J v_2\;+\;\sum_J f_{J,3}\int_{\Omega_h}\varphi_J v_3\quad\forall\,\boldsymbol{v}_h\;=\;(v_1,\,v_2,\,v_3)\,\in\,V_h^3\,.
\end{aligned}$$

We can now split this formulation in three different parts corresponding to the three lines of the linear system $A\Xi = F$ associated with the three components of vertex I. This is done by choosing as test functions $\boldsymbol{v_h} = (\varphi_I,\,0,\,0)$, $\boldsymbol{v_h} = (0,\,\varphi_I,\,0)$ and $\boldsymbol{v_h} = (0,\,0,\,\varphi_I)$, respectively. We get:

$$\begin{aligned}
&\sum_J d_{J,1}\left(\int_{\Omega_h} s\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x}\;+\;\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}\;+\;\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial z}\right)+\;\sum_J d_{J,2}\left(\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial x}+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y}\right)\\
+\;&\sum_J d_{J,3}\left(\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial x}\;+\;\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial z}\right)\qquad\qquad=\;\sum_J f_{J,1}\int_{\Omega_h}\varphi_J\varphi_I\,,\\
&\sum_J d_{J,1}\left(\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial x}\;+\;\int_{\Omega_h}\lambda\frac{\partial\,\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y}\right)+\;\sum_J d_{J,2}\left(\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x}\;+\;\int_{\Omega_h}s\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}\;+\;\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial z}\right)\\
+\;&\sum_J d_{J,3}\left(\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial y}\;+\;\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial z}\right)\qquad=\;\sum_J f_{J,2}\int_{\Omega_h}\varphi_J\varphi_I\,,
\end{aligned}$$

$$\sum_J d_{J,1}\left(\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial x}+\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial z}\right)+\sum_J d_{J,2}\left(\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial y}+\int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial z}\right)$$

$$+\sum_J d_{J,3}\left(\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x}+\int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}+\int_{\Omega_h}s\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial z}\right)=\sum_J f_{J,3}\int_{\Omega_h}\varphi_J\varphi_I\,.$$

Therefore, we have to solve the following linear system:

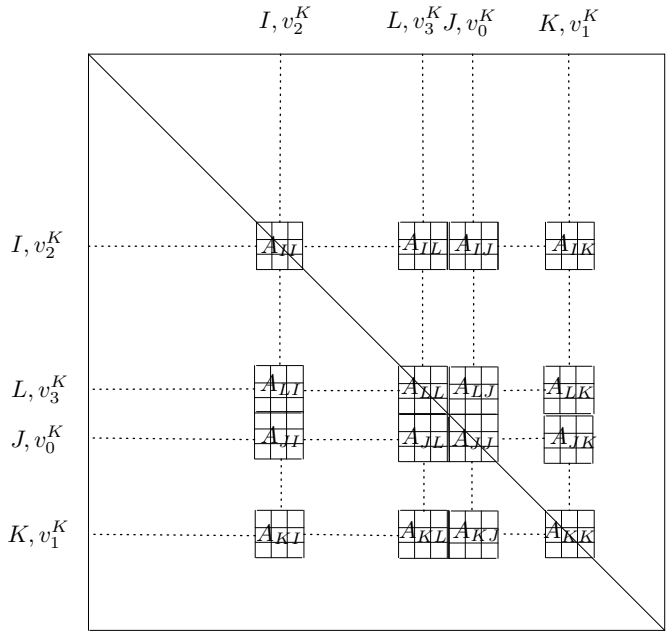
$$A\Xi=F$$

where A is a block matrix having 3×3 block A_{IJ} at block indices (I,J) given by:

$$A_{IJ}=\begin{pmatrix}\int_{\Omega_h}s\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x}+\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}+\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial z} & \int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial x}+\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y} & \int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial x}+\lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial z}\\ \int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial x}+\lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial y} & \int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x}+s\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}+\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial z} & \int_{\Omega_h}\lambda\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial y}+\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial z}\\ \int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial x}+\lambda\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial z} & \int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial y}+\lambda\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial z} & \int_{\Omega_h}\mu\frac{\partial\varphi_J}{\partial x}\frac{\partial\varphi_I}{\partial x}+\mu\frac{\partial\varphi_J}{\partial y}\frac{\partial\varphi_I}{\partial y}+s\frac{\partial\varphi_J}{\partial z}\frac{\partial\varphi_I}{\partial z}\end{pmatrix}$$

and F is a block vector having F_I as 3-vector at index I:

$$F_I=\left(\sum_J f_{J,1}\int_{\Omega_h}\varphi_J\varphi_I,\quad\sum_J f_{J,2}\int_{\Omega_h}\varphi_J\varphi_I,\quad\sum_J f_{J,3}\int_{\Omega_h}\varphi_J\varphi_I\right)^T.$$



Note that the elasticity matrix is symmetric and that a tetrahedron K contributes to $16 \times 9 = 144$ boxes of the matrix (but we only need to fill $6 \times 9 + 4 \times 6 = 78$ boxes as the matrix is symmetric).
As for the two dimensional case, we now detail the computation of the integral terms. $\hat{K} = \left(\hat{P}_0, \hat{P}_1, \hat{P}_2, \hat{P}_3\right)$ with $\hat{P}_0 = (0,0,0)$, $\hat{P}_1 = (1,0,0)$, $\hat{P}_2 = (0,1,0)$ and $\hat{P}_3 = (0,0,1)$ is taken as the reference tetrahedron. The gradients of the P^1 shape functions (i.e the barycentrics) on the reference tetrahedron are given by:

$$\begin{array}{ll} \hat{\varphi}_0 = 1 - x_0 - x_1 - x_2\,, & \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}_0 = \begin{pmatrix} -1, & -1, & -1 \end{pmatrix}^T \\ \hat{\varphi}_1 = x_0 & \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}_1 = \begin{pmatrix} 1, & 0, & 0 \end{pmatrix}^T, \\ \hat{\varphi}_2 = x_1\,, & \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}_2 = \begin{pmatrix} 0, & 1, & 0 \end{pmatrix}^T, \\ \hat{\varphi}_3 = x_2\,, & \nabla_{\hat{\boldsymbol{x}}}\hat{\varphi}_3 = \begin{pmatrix} 0, & 0, & 1 \end{pmatrix}^T. \end{array} \tag{B.6}$$

The Jacobian matrix of the mapping from $\hat{K}$ to the current tetrahedron K is given by:

$$B_K = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{e_0} & \boldsymbol{e_1} & \boldsymbol{e_2} \end{pmatrix}$$

$$B_K^{-T} = \frac{1}{\det B_K} \begin{pmatrix} \overline{\boldsymbol{\eta_1}} & \overline{\boldsymbol{\eta_2}} & \overline{\boldsymbol{\eta_3}} \end{pmatrix} = \frac{1}{6|K|} \begin{pmatrix} \overline{\boldsymbol{\eta_1}} & \overline{\boldsymbol{\eta_2}} & \overline{\boldsymbol{\eta_3}} \end{pmatrix},$$

and the normals are defined in Section . As in 2D, we can now compute the gradients of the shape functions on the current tetrahedron:

$$\nabla_{\boldsymbol{x}}\varphi_0(\boldsymbol{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_0}}, \quad \nabla_{\boldsymbol{x}}\varphi_1(\boldsymbol{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_1}}, \quad \nabla_{\boldsymbol{x}}\varphi_2(\boldsymbol{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_2}}, \quad \nabla_{\boldsymbol{x}}\varphi_3(\boldsymbol{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_3}}\,.$$

Boundary conditions. We only use Dirichlet boundary conditions and strongly impose the displacement on the domain boundaries. This means that if vertex P_I is on the boundary, we set $A_{II} = \mathcal{I}_n$, $A_{IJ} = 0\ \forall J \neq I$ and $F_I = \begin{pmatrix} d_{I,1}, & d_{I,2}, & d_{I,3} \end{pmatrix}^T$ in the linear system, with $\begin{pmatrix} d_{I,1}, & d_{I,2}, & d_{I,3} \end{pmatrix}^T$ the imposed displacement at vertex I.
The displacement of the boundaries can be prescribed in different ways:

Numerically: by an external solution, for example the results of the FSIs resolution, see Section 5.5.3

Analytically: by prescription of a translation vector and a rotation vector.

Appendix C

Strong-Stability-Preserving Runge-Kutta schemes

C.1 Runge-Kutta schemes

C.1.1 General formulation

The following first order Ordinary Differential Equation is considered:

$$\dot{Y}(t) = H(t, Y(t)),\ t \in [a,\, b] \subset \mathbb{R} \tag{C.1}$$

for which $Y \in C^1([a,\, b])$ is either a scalar or vectorial function of t and H is either a scalar or vectorial function.
Let $\left(t^k\right)_{k \in [\![0,\, N_{ite}]\!]}$ be a uniform time discretization of $[a,\, b]$. We note $\tau = t^k - t^{k-1}$. $Y(t^k)$ is the approximated solution of Problem (C.1) at $t = t^k$.
A Runge-Kutta method involving n_s stages can be written under general form:

$$\begin{aligned} Y(t^{k+1}) &= Y^k + \tau \sum_{s=1}^{n_s} b_s\, H_s \\ \text{and} \quad H_s &= H(t^k + c_s\tau, Y^s + \tau \sum_{l=0}^{n_s} a_{sl} H_l), \quad s \in [\![0,\, n_s]\!]\,, \end{aligned}$$

where $(a_{sl})_{0 \leq s,l \leq n_s}$, $(b_s)_{0 \leq s \leq n_s}$ and $(c_s)_{0 \leq s \leq n_s}$ are called the Butcher coefficients of the Runge-Kutta scheme. These coefficients are sometimes represented using a Butcher array:

$$\begin{array}{c|ccc} c_0 & a_{0,0} & \dots & a_{0,n_s} \\ \vdots & \vdots & & \vdots \\ c_{n_s} & a_{n_s,0} & \dots & a_{n_s,n_s} \\ \hline & b_0 & \dots & b_{n_s} \end{array} \tag{C.2}$$

The minimum conditions to be satisfied by these coefficients are:

$$\begin{aligned} c_s &= \sum_{l=0}^{n_s} a_{sl} \quad \text{(exact prediction for the special equation } \dot{Y} = 1) \\ 1 &= \sum_{s=0}^{n_s} b_s \quad \text{(consistency condition)} \end{aligned} \tag{C.3}$$

If these relations are checked, then Runge-Kutta schemes can be rewritten:

$$\begin{aligned} Y^{n+1} &= Y^n + \tau \sum_{s=0}^{n_s} b_s\, H\left(t^n + c_s\tau,\, Y^s\right) \\ \text{with} \quad Y^s &= Y^n + \tau \sum_{l=0}^{n_s} a_{sl}\, H\left(t^n + c_l\tau,\, Y^l\right), \end{aligned} \tag{C.4}$$

and Y^s can be interpreted as a predictor for the the value of the solution Y at $t^s = t^n + c_s\tau$.

C.1.2 Explicit and implicit schemes

In the following, the following notations are adopted:

$$A = \begin{pmatrix} a_{00} & \dots & a_{0,n_s} \\ \vdots & & \vdots \\ a_{n_s,0} & \dots & a_{n_s,n_s} \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} b_0 \\ \vdots \\ b_{n_s} \end{pmatrix}. \tag{C.5}$$

If $(j >= i \Rightarrow a_{ij} = 0)$, *i.e.* A is lower triangular, then the scheme is explicit. Otherwise, it is implicit. For an explicit Runge-Kutta scheme, its associated Butcher array has the following form:

$$\begin{array}{c|cccc} c_0 & 0 & & \dots & 0 \\ c_1 & a_{11} & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots & 0 \\ c_{n_s} & a_{n_s,0} & \dots & a_{n_s,n_s-1} & 0 \\ \hline & b_0 & & \dots & b_{n_s} \end{array} \tag{C.6}$$

From now on, Problem (C.1) is assumed to be autonomous, *i.e.* H does not depend directly on t. As a consequence, explicit Runge-Kutta schemes write:

$$\begin{aligned} &Y(t^{k+1}) = Y^{n_s} = Y^0 + \tau \sum_{s=0}^{n_s-1} b_s\, H(Y^s), \quad b_{n_s} = a_{n_s,n_s} \\ &\text{and} \quad Y^s = Y^0 + \tau \sum_{l=0}^{s-1} a_{sl}\, H(Y^s) \quad \text{with} \quad Y^0 = Y(t^k)\,. \end{aligned}$$

C.1.3 Shu-Osher representation

We now introduce two new set of parameters (α_{sl}) and (β_{sl}) with $s \in [\![1,\, n_s]\!]$ and $l \in [\![0,\, n_s - 2]\!]$ such that:

$$\sum_{l=0}^{s-1} \alpha_{sl} = 1 \quad \forall\, 1 \le s \le n_s \quad \text{and} \quad \beta_{sl} = a_{sl} - \sum_{k=l+1}^{n_s-1} \alpha_{sk}\, a_{kl}\,, \quad \forall s \in [\![1,\, n_s]\!]\,.$$

Runge-Kutta explicit schemes can then be reformulated using these coefficients as a convex combination of forward Euler steps:

$$\begin{aligned} Y(t^{k+1}) = Y^{n_s} &= Y^0 + \tau \sum_{s=0}^{n_s-1} b_s\, H(Y^s) \\ Y^s = \sum_{l=0}^{s-1} \left[\alpha_{sl} Y^l + \tau\, \beta_{sl} H(Y^l)\right] &= \sum_{l=0}^{s-1} \alpha_{sl} \left[Y^l + \tau\, \frac{\beta_{sl}}{\alpha_{sl}} H(Y^l)\right] \quad \text{(Shu-Osher representation)}\,. \end{aligned} \tag{C.7}$$

The proof is given below:

$$\begin{aligned} Y^s &= Y^0 + \tau \sum_{l=0}^{s-1} a_{sl} H(Y^l) = \left(\sum_{l=0}^{s-1} \alpha_{sl}\right) Y^0 + \tau \sum_{l=0}^{s-1} a_{sl}\, H(Y^l) \\ Y^s &= \alpha_{s0} Y^0 + \sum_{l=1}^{s-1} \alpha_{sl} \underbrace{\left(Y^l - \tau \sum_{k=0}^{l-1} a_{lk}\, H(Y^k)\right)}_{=Y^0} + \tau \sum_{l=0}^{s-1} a_{sl}\, H(Y^l) \\ Y^s &= \sum_{l=0}^{s-1} \left[\alpha_{sl} Y^l + \tau \underbrace{\left(a_{sl} - \sum_{k=l+1}^{s-1} \alpha_{sk} a_{kl}\right)}_{\beta_{sl}} H(Y^l)\right] \quad \text{(manipulation on sums)}\,. \end{aligned}$$

C.2 Order of accuracy

Theorem 6 ($n_s \geq p$)**.** *The order of an explicit n_s-stages Runge-Kutta method is less than or equal to its number of stages n_s ($p \leq n_s$)*

Theorem 7. *There is no explicit n_s-stages Runge-Kutta method of order n_s if $n_s \geq 5$*

The following relationships must be satisfied by the scheme coefficients for the scheme to be of order p:

Order	1	2	3	4
Conditions	$\sum_{s=0}^{n_s-1} b_s = 1$	$\sum_{s=0}^{n_s-1} b_s c_s = \frac{1}{2}$	$\sum_{s=0}^{n_s-1} b_s c_s^2 = \frac{1}{3}$ $\sum_{s=0}^{n_s-1} b_s \sum_{l=0}^{s-1} a_{sl} c_l = \frac{1}{6}$	$\sum_{s=0}^{n_s-1} b_s c_s^3 = \frac{1}{4}$ $\sum_{s=0}^{n_s-1} b_s c_s \sum_{l=0}^{s-1} a_{sl} c_l = \frac{1}{8}$ $\sum_{s=0}^{n_s-1} b_s \sum_{l=0}^{s-1} a_{sl} c_l^2 = \frac{1}{12}$ $\sum_{s=0}^{n_s-1} b_s \sum_{l=0}^{s-1} a_{sl} \sum_{k=1}^{l-1} a_{lk} c_k = \frac{1}{24}$

(C.8)

These relations can be demonstrated by truncation error study (Taylar expansion) or in a much more beautiful way using Butcher graph theory (rooted-trees), [Butcher 1987].

C.3 Linear stability results

Let us first consider the following linear problem, with λ constant:

$$\dot{Y}(t) = \lambda Y(t)\,, \quad \text{i.e.} \quad H(Y) = \lambda Y(t)\,.$$

In this very special case, Runge-Kutta methods (explicit or implicit) read:

$$Y(t^{k+1}) = Y^0 + \tau \sum_{s=0}^{n_s-1} b_s H_s\,, \quad Y^0 = Y(t^k)$$
$$\text{with} \quad H_s = H(Y^0 + \tau \sum_{l=0}^{s-1} a_{sl} H_l) = \lambda \left(Y^0 + \tau \sum_{l=0}^{s-1} a_{sl} H_l \right)\,.$$

For $s \in [\![0,\, n_s - 1]\!]$:

$$H_s = \lambda \left(Y^0 + \tau \sum_{l=0}^{n_s-1} a_{sl} H_l \right) \quad \Rightarrow \quad \left(H_s - \lambda\tau \sum_{l=0}^{n_s-1} a_{sl} H_l \right) = \lambda Y^0 \quad \Rightarrow \quad (\mathcal{I}_{n_s} - \lambda\tau A)\,\mathbf{K} = \lambda Y^0 \mathbf{e}$$
$$\Rightarrow \quad \mathbf{K} = Y^0 \lambda\,(\mathcal{I}_{n_s} - \lambda\tau A)^{-1}\,\mathbf{e}\,,$$

where we have noted:

$$\mathbf{e} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \quad \text{and} \quad \mathbf{K} = \begin{pmatrix} H_0 \\ \vdots \\ H_s \end{pmatrix}\,.$$

Then, we get:

$$Y^{n_s} = Y^0 + \tau \mathbf{b}^T \cdot \mathbf{K} = Y^0 \left(1 + \tau\lambda \mathbf{b}^T \left(\mathcal{I}_{n_s} - \lambda\tau A\right)^{-1} \mathbf{e}\right) = Y^0\, R(\tau\lambda) \in \mathbb{C}\,.$$

$R(\tau\lambda)$ is a complex number and can be rewritten as:

$$R(h\lambda) \;=\; \frac{\det\left(\mathcal{I}_{n_s} - \tau\lambda A + \tau\lambda \mathbf{e}\cdot\mathbf{b}^T\right)}{\det\left(\mathcal{I}_{n_s} - \tau\lambda A\right)}\,.$$

If we use an explicit method, the denominator is equal to 1 because $\mathbb{A}$ is triangular inferior with only zeros on its diagonal. The stability factor is thus polynomial in λh. The stability region will never contain the whole $Re(\lambda h) < 0$ semi-plane.
If we use an implicit Runge-Kutta method, the stability factor is genereally rationnal, which tends to bring more stability to the scheme.

C.4 Non-linear stability results

Non-linear stability definition. A scheme is said to be Strong-Stability-Preserving (SSP) if, in a given semi-norm $|\cdot|$, we have $|W^{k+1}| \leq |W^k|$, with W^k the numerical solution vector at time t^k. This notion is linked to the more classical concept of contractivity. The semi-norm is classically the TVD norm appearing in the study of hyperbolic PDEs.

$$|W^k|_{TVD} = \sum_{i=1}^{N} |W^k_{i+1} - W^k_i|$$

Indeed, under certain hypothesis, the true solution of this kind of equations is TVD, *i.e.* the solution cannot exhibit new local extrema or minima during its time evolution.

Theorem 8. *If the Forward Euler (FE) method is SSP under CFL restriction $\tau \leq \tau_{FE}$, then the Runge-Kutta method*

$$Y(t^{k+1}) = Y^{n_s} \tag{C.9}$$

$$Y^s = \sum_{l=0}^{s-1} \left[\alpha_{sl} Y^l + \beta_{sl}\tau H(Y^l)\right] = \sum_{l=0}^{s-1} \alpha_{sl}\left[Y^l + \frac{\beta_{sl}}{\alpha_{sl}}\tau H(Y^l)\right] \tag{C.10}$$

$$Y_0 = Y^n \tag{C.11}$$

with $\beta_{sl} \geq 0$ is SSP provided $\tau \leq CFL\tau_{FE}$ where CFL is the CFL coefficient of the SSP Runge-Kutta scheme given by:

$$CFL = \min_{s,l} \frac{\alpha_{sl}}{\beta_{sl}}\,.$$

C.5 SSP optimal schemes (i.e maximal CFL schemes)

C.5.1 Linear case with $n_s = p$ [Gottlieb 2001]

The following linear differential equation is considered, with L a linear operator:

$$\dot{Y}(t) = LY(t)$$

Theorem 9. *Consider the family of n_s-stages, p-order Runge-Kutta Strong-Stability-Preserving methods with non-negative coefficients $\alpha_{i,k}$, $\beta_{i,k}$. The best (larger) CFL attainable with such methods is the one dictated by the Forward Euler scheme.*

For the special case of **linear** equations, we can exhibit n_s-stages Runge-Kutta schemes which are n_s-order accurate and SSP with $CFL = 1$.

$$\begin{aligned}
Y^s &= Y^{s-1} + \tau L Y^{s-1} \\
Y^{n_s} &= \sum_{s=0}^{n_s-2} \alpha_{n_s,s} Y^s + \alpha_{n_s,n_s-1}\left(Y^{n_s-1} + \tau L Y^{n_s-1}\right) \\
\alpha_{n_s,s} &= \frac{1}{s}\alpha_{n_s-1,s-1}, \quad s \in [\![1, n_s-2]\!] \\
\alpha_{n_s,n_s-1} &= \frac{1}{n_s!}, \quad \alpha_{n_s,0} = 1 - \sum_{s=1}^{n_s-1} \alpha_{n_s,s}
\end{aligned}$$

Order n_s	$\alpha_{n_s,0}$	$\alpha_{n_s,1}$	$\alpha_{n_s,2}$	$\alpha_{n_s,3}$	$\alpha_{n_s,4}$	$\alpha_{n_s,5}$	$\alpha_{n_s,6}$	$\alpha_{n_s,7}$
1	1							
2	$\frac{1}{2}$	$\frac{1}{2}$						
3	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{6}$					
4	$\frac{3}{8}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{24}$				
5	$\frac{11}{30}$	$\frac{3}{8}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{1}{120}$			
6	$\frac{53}{144}$	$\frac{11}{30}$	$\frac{3}{16}$	$\frac{1}{18}$	$\frac{1}{48}$	$\frac{1}{720}$		
7	$\frac{103}{280}$	$\frac{53}{144}$	$\frac{11}{60}$	$\frac{3}{48}$	$\frac{1}{72}$	$\frac{1}{240}$	$\frac{1}{5040}$	
8	$\frac{2119}{5760}$	$\frac{103}{280}$	$\frac{53}{288}$	$\frac{11}{180}$	$\frac{1}{64}$	$\frac{1}{360}$	$\frac{1}{1440}$	$\frac{1}{40320}$

These schemes are low-storage, in the sense that only the previous state need to be stored to compute the current state. These schemes are no more n_s-order accurate if the equation is non-linear.

More generally, it is proved in [Spiteri 2002] that:

Theorem 10 (Optimal RKSSP CFL for linear problems)**.** *The optimal CFL coefficient of an n_s-stage, p-order RKSSP method applied to a linear, constant coefficient problem is $n_s - p + 1$*

C.5.2 Non-linear case with $n_s = p$

The coefficient CFL for the Runge-Kutta schemes with $n_s = p$ is such that $CFL \leq 1$. Indeed, it is already the case in the simple linear case (see above), so it is all the more the case in the non-linear case. So, if we find a scheme satisfying the n_s-order conditions and for which $CFL = 1$, then it is optimal. Gottlieb and Shu [Gottlieb 1998] give a direct and technical proof of this result for $n_s = 2, 3, 4$. The proof is based on the accuracy relationships satisfied by the scheme coefficients. From these relations, a general form of the Runge-Kutta coefficients is then found, usually involving unknown parameters. Finally, a case study is performed to prove that it is impossible to have $CFL > 1$ with these families of coefficient for $n_s = 2, 3, 4$, which exactly means that these schemes are optimal.

Order 2 RKSSP(2,2) An optimal second-order SSP Runge-Kutta method is given by:

$$Y^0 = Y(t^k) \tag{C.12}$$
$$Y^1 = Y^0 + \tau H(Y^0) \tag{C.13}$$
$$Y(t^{k+1}) = Y^2 = \frac{1}{2}Y^0 + \frac{1}{2}Y^1 + \frac{1}{2}\tau H(Y^1) \tag{C.14}$$

$$\alpha_{10} = 1 \quad \beta_{10} = 1 \tag{C.15}$$
$$\alpha_{20} = \frac{1}{2} \quad \alpha_{21} = \frac{1}{2} \quad \beta_{20} = 0 \quad \beta_{21} = \frac{1}{2} \tag{C.16}$$
$$\text{and} \quad CFL = \min_{ij} \frac{\alpha_{ij}}{\beta_{ij}} = \min\left(\frac{\alpha_{10}}{\beta_{10}}, \frac{\alpha_{20}}{\beta_{20}}, \frac{\alpha_{21}}{\beta_{21}}\right) = 1 \tag{C.17}$$

Thus the CFL for RKSSP(2,2) equals 1, as expected.

Order 3 An optimal third-order SSP Runge-Kutta method is given by:

$$Y^0 = Y(t^k) \tag{C.18}$$
$$Y^1 = Y^0 + \tau H(Y^0) \tag{C.19}$$
$$Y^2 = \frac{3}{4}Y^0 + \frac{1}{4}Y^1 + \frac{1}{4}\tau H(Y^1) \tag{C.20}$$
$$Y(t^{k+1}) = Y^3 = \frac{1}{3}Y^0 + \frac{2}{3}Y^2 + \frac{2}{3}\tau H(Y^2) \tag{C.21}$$

$$\alpha_{10} = 1 \quad \beta_{10} = 1 \tag{C.22}$$
$$\alpha_{20} = \frac{3}{4} \quad \alpha_{21} = \frac{1}{4} \quad \beta_{20} = 0 \quad \beta_{21} = \frac{1}{4} \tag{C.23}$$
$$\alpha_{30} = \frac{1}{3} \quad \alpha_{31} = 0 \quad \alpha_{32} = \frac{2}{3} \quad \beta_{20} = 0 \quad \beta_{21} = 0 \quad \beta_{22} = \frac{2}{3} \tag{C.24}$$
$$\text{and} \quad CFL = \min_{ij} \frac{\alpha_{ij}}{\beta_{ij}} = \min\left(\frac{\alpha_{10}}{\beta_{10}}, \frac{\alpha_{20}}{\beta_{20}}, \frac{\alpha_{21}}{\beta_{21}}, \frac{\alpha_{30}}{\beta_{30}}, \frac{\alpha_{31}}{\beta_{31}}, \frac{\alpha_{32}}{\beta_{32}}\right) = 1 \tag{C.25}$$

Thus the CFL for RKSSP(3, 3) equals 1, as expected.

Order 4 A negative result:

Theorem 11. *No four-stages fourth-order Runge-Kutta Strong-Stability-Preserving scheme exists with positive β_{ij} coefficients (i.e with no adjoint operator computations)*

C.5.3 Non-linear case with $n_s \geq p$

Theorem 12 (Order 1 schemes)**.** *: The optimal n_s-stage RKSSP method of order 1 with $\beta_{ij} \geq 0$ has a CFL coefficient equal to n_s and can be represented under the form:*

$$\alpha_{sl} = \begin{cases} 1 & \text{if} \quad l = s-1\,, \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \quad \beta_{sl} = \begin{cases} \frac{1}{n_s} & \text{if} \quad l = s-1\,, \\ 0 & \text{otherwise} \end{cases} \quad , \quad \forall\, s \in [\![1, n_s]\!]\,.$$

Theorem 13 (Order 2 schemes)**.** *: The optimal n_s-stage RKSSP method of order 2 with $\beta_{ij} \geq 0$ has*

CFL coefficient $n_s - 1$ *and can be represented under the form:*

$$\alpha_{sl} = \begin{cases} 1 & \text{if} \quad l = s-1\,, \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \quad \beta_{sl} = \begin{cases} \frac{1}{s-1} & l = s-1\,, \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, s \in [\![1,\, n_s - 1]\!]$$

$$\alpha_{n_s l} = \begin{cases} \frac{1}{n_s} & \text{if} \quad l = 0\,, \\ \frac{n_s - 1}{n_s} & \text{if} \quad l = n_s - 1\,, \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \quad \beta_{n_s l} = \begin{cases} \frac{1}{n_s} & \text{if} \quad l = n_s - 1 \\ 0 & \text{otherwise} \end{cases} \qquad \text{for} \quad s = n_s\,.$$

Proofs of these theorems rely on the following lemma:

Theorem 14. *If a method of the Shu-Osher form with* $\alpha_{sl}, \beta sl \geq 0$ *has a CFL coefficient* $CFL > m > 0$*, then* $0 \leq a_{sl} < \frac{1}{m}, \quad \forall l \in [\![0,\, s-1]\!] \quad ands \in [\![1,\, n_s]\!]$

The above lemma, combined with the accuracy relations of Section C.2 are used to prove that it is impossible to have $CFL > n_s$ for an n_s-stage RKSSP method of order 1 and $CFL > n_s - 1$ for an n_s-stage SSPRK method of order 2. As the proposed RKSSP methods are exactly such that $CFL = n_s$ for order 1 and $CFL = n_s - 1$ for order 2, we deduce that they are optimal in the sense of the maximal CFL coefficient.

First-order accurate RKSSP optimal schemes.

Stages	α_{ij}	β_{ij}	CFL coefficient
1	1	1	1
2	$\begin{matrix} 1 & \\ 0 & 1 \end{matrix}$	$\begin{matrix} \frac{1}{2} & \\ 0 & \frac{1}{2} \end{matrix}$	2
3	$\begin{matrix} 1 & & \\ 0 & 1 & \\ 0 & 0 & 1 \end{matrix}$	$\begin{matrix} \frac{1}{3} & & \\ 0 & \frac{1}{3} & \\ 0 & 0 & \frac{1}{3} \end{matrix}$	3

(C.26)

Second-order accurate RKSSP optimal schemes.

Stages	α_{ij}	β_{ij}	CFL coefficient
2	$\begin{matrix}1 & \\ \frac{1}{2} & \frac{1}{2}\end{matrix}$	$\begin{matrix}1 & \\ 0 & \frac{1}{2}\end{matrix}$	1
3	$\begin{matrix}1 & & \\ 0 & 1 & \\ \frac{1}{3} & 0 & \frac{2}{3}\end{matrix}$	$\begin{matrix}\frac{1}{2} & & \\ 0 & \frac{1}{2} & \\ 0 & 0 & \frac{1}{3}\end{matrix}$	2
4	$\begin{matrix}1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ \frac{1}{4} & 0 & 0 & \frac{3}{4}\end{matrix}$	$\begin{matrix}\frac{1}{3} & & & \\ 0 & \frac{1}{3} & & \\ 0 & 0 & \frac{1}{3} & \\ 0 & 0 & 0 & \frac{1}{4}\end{matrix}$	3
5	$\begin{matrix}1 & & & & \\ 0 & 1 & & & \\ 0 & 0 & 1 & & \\ 0 & 0 & 0 & 1 & \\ \frac{1}{5} & 0 & 0 & 0 & \frac{4}{5}\end{matrix}$	$\begin{matrix}\frac{1}{4} & & & & \\ 0 & \frac{1}{4} & & & \\ 0 & 0 & \frac{1}{4} & & \\ 0 & 0 & 0 & \frac{1}{4} & \\ 0 & 0 & 0 & 0 & \frac{1}{5}\end{matrix}$	4

(C.27)

RKSSP(5,3) and RKSSP(5,4) are obtained by Spiteri and Ruuth [Spiteri 2003] through the numerical resolution of a non-linear programming problem.

Third-order accurate RKSSP optimal schemes.

Stages	α_{ij}	β_{ij}	CFL coefficient
3	$\begin{matrix}1 & & \\ \frac{3}{4} & \frac{1}{4} & \\ \frac{1}{3} & 0 & \frac{2}{3}\end{matrix}$	$\begin{matrix}1 & & \\ 0 & \frac{1}{4} & \\ 0 & 0 & \frac{2}{3}\end{matrix}$	1
4	$\begin{matrix}1 & & & \\ 0 & 1 & & \\ \frac{2}{3} & 0 & \frac{1}{3} & \\ 0 & 0 & 0 & 1\end{matrix}$	$\begin{matrix}\frac{1}{2} & & & \\ 0 & \frac{1}{2} & & \\ 0 & 0 & \frac{1}{6} & \\ 0 & 0 & 0 & \frac{1}{2}\end{matrix}$	2

(C.28)

For clarity purpose, scheme RKSSP(5,3) is given under a different form. The Shu-Osher (α_{sl}) coefficients

are given by:

s	α_{s0}	α_{s1}	α_{s2}	α_{s3}	α_{s4}
1	1				
2	0	1			
3	0.56656131914033	0	0.43343868085967		
4	0.0929948344413	0.00002090369620	0	0.90698426185967	
5	0.00736132260920	0.20127980325145	0.00182955389682	0	0.78952932024253

,

and the (β_{sl}) write:

s	β_{s0}	β_{s1}	β_{s2}	β_{s3}	β_{s4}
1	0.37726891511710				
2	0	0.37726891511710			
3	0	0	0.16352294089771		
4	0.00071997378654	0	0	0.34217696850008	
5	0.00277719819460	0.00001567934613	0	0	0.29786487010104

The maximal CFL coefficient for this scheme is $CFL = 2.65062919294483$.

Fourth-order accurate RKSSP optimal scheme. For clarity purpose, scheme RKSSP(5,3) is given under a different form.

s	α_{s0}	α_{s1}	α_{s2}	α_{s3}	α_{s4}
1	1				
2	0.44437049406734	0.55562950593266			
3	0.62010185138540	0	0.37989814861460		
4	0.17807995410773	0	0	0.82192004589227	
5	0.00683325884039	0	0.51723167208978	0.12759831133288	0.34833675773694

s	β_{s0}	β_{s1}	β_{s2}	β_{s3}	β_{s4}
1	0.39175222700392				
2	0	0.36841059262959			
3	0	0	0.25189177424738		
4	0	0	0	0.54497475021237	
5	0	0	0	0.08460416338212	0.22600748319395

The maximal CFL coefficient for this scheme is $CFL = 1.50818004975927$.

Remark 11. *Note that the "classical" fourth-order Runge-Kutta scheme is not Strong-Stability-Preserving. Indeed, this scheme writes:*

$$Y^0 = Y(t^k)\,, \tag{C.29}$$
$$Y^1 = Y^0 + \frac{\tau}{2} H(Y^0)\,, \tag{C.30}$$
$$Y^2 = Y^0 + \frac{\tau}{2} H(Y_1)\,, \tag{C.31}$$
$$Y^3 = Y^0 + \frac{\tau}{2} H(Y_2)\,, \tag{C.32}$$
$$Y(t^{k+1}) = Y^4 = Y^0 + \frac{\tau}{6} f(Y^0) + \frac{\tau}{3} H(Y_1) + \frac{\tau}{3} H(Y_2) + \frac{\tau}{6} H(Y_3)\,. \tag{C.33}$$

Hence, its Shu-Osher coefficients are:

$$\begin{array}{llllll} \alpha_{10}=1 & & & & & \beta_{10}=\frac{1}{2} \\ \alpha_{20}=1 & \alpha_{21}=0 & & & & \beta_{20}=0 \quad \beta_{21}=\frac{1}{2} \\ \alpha_{30}=1 & \alpha_{31}=0 & \alpha_{32}=0 & & \textit{and} & \beta_{30}=0 \quad \beta_{31}=0 \quad \beta_{32}=1 \\ \alpha_{40}=1 & \alpha_{41}=0 & \alpha_{42}=0 & \alpha_{43}=0 & \beta_{40}=\frac{1}{6} & \beta_{41}=\frac{1}{3} \quad \beta_{42}=\frac{1}{3} \quad \beta_{43}=\frac{1}{6} \end{array} \tag{C.34}$$

Then, we see that:

$$CFL = \min_{sl} \frac{\alpha_{sl}}{\beta_{sl}} = \frac{\alpha_{21}}{\beta_{21}} = \frac{\alpha_{32}}{\beta_{32}} = \frac{\alpha_{41}}{\beta_{41}} = \frac{\alpha_{42}}{\beta_{42}} = \frac{\alpha_{43}}{\beta_{43}} = 0\,,$$

which means that this scheme is not SSP.

C.6 Implicit Backward Difference Formulae (BDF) methods

The backward differentiation formula (BDF) is a family of implicit methods for the numerical integration of ordinary differential equations. They are linear multistep methods that, for a given function and time, approximate the derivative of that function using information from already computed times, thereby increasing the accuracy of the approximation. Backward Differentiation Formula methods are implicit and, as such, require the solution of non-linear equations at each step. Typically, a modified Newton's method is used to solve these non-linear equations.

C.6.1 First-order accurate implicit Backward Difference Formula

$$Y(t^{k+1}) - Y(t^k) = H(t^(k), Y(t^{k+1}))\,.$$

C.6.2 Second-order accurate implicit Backward Difference Formula

For variable time steps, the second-order accurate BDF scheme writes:

$$\frac{1+2r}{1+r}Y(t^{k+1}) + (-1-r)\,Y(t^k) + \frac{r^2}{1+r}Y(t^{k-1}) = H(t^{k+1}, Y(t^{k+1})), \quad \text{where} = \frac{\tau^k}{\tau^{k-1}}\,.$$

For constant time steps, *i.e.* $r = 1$, the above expression reduces to:

$$\frac{3}{2}Y(t^{k+1}) - 2Y(t^k) + \frac{1}{2}Y(t^{k-1}) = H(t^{k+1}, Y(t^{k+1}))\,.$$

C.6.3 Higher-order accurate implicit Backward Difference Formula

Writing and solving appropriate Differential Algebraic equations leads to the following coefficients for higher-order BDF schemes:

Order	α_0	α_1	α_2	α_3	α_4	α_5	α_6
1	1	-1					
2	$\frac{3}{2}$	-2	$\frac{1}{2}$				
3	$\frac{11}{6}$	-3	$\frac{3}{2}$	$-\frac{1}{3}$			
4	$\frac{25}{12}$	-4	3	$-\frac{4}{3}$	$-\frac{1}{4}$		
5	$\frac{137}{60}$	-5	5	$-\frac{10}{3}$	$\frac{5}{4}$	$-\frac{1}{5}$	
6	$\frac{49}{20}$	-6	$\frac{15}{2}$	$-\frac{20}{3}$	$\frac{15}{4}$	$-\frac{6}{5}$	$\frac{1}{6}$

(C.35)

BDF- based methods are especially appreciated for their stability properties. The six schemes given above are stable along the whole negative real axis. Note that for order higher than 7, the instability region intersects the negative real axis, making it and any higher order BDF methods of little interest.

Appendix D

ALE mirror state slipping boundary conditions

D.1 Preliminary: Jacobian matrix in three dimensions

The Euler system in the ALE framework writes, for any arbitrary closed volume $C(t)$ of boundary $\partial C(t)$:

$$\frac{\partial}{\partial t}|_{\boldsymbol{\xi}} \left(\int_{C(t)} W \, \mathrm{d}\mathbf{x} \right) \; + \; \int_{\partial C(t)} \left(\mathbf{F} - W \; (\boldsymbol{w} \cdot \mathbf{n}) \right) \, \mathrm{d}\boldsymbol{s} = \int_{C(t)} \mathbf{f_{ext}} \, \mathrm{d}\mathbf{x} \, ,$$

and

$$\left\{ \begin{array}{ll} W & = (\rho, \, \rho \mathbf{u}, \, \rho e)^T \text{ is the conservative variables vector} \\ \mathbf{F}\,(W) & = \mathcal{F}(W) \cdot \mathbf{n} \text{ is the normal flux vector} \\ \mathbf{f_{ext}} & = (0, \, \rho \, \mathbf{f_{ext}}, \, \rho \mathbf{u} \cdot \mathbf{f_{ext}})^T \text{ is the external volumes forces vector} \end{array} \right.$$

In the sequel, $\boldsymbol{\eta}$ is an outward non-normalized normal vector to $C(t)$ and $\mathbf{n}$ its associated normalized vector. Notation $u^{nor} = \mathbf{u} \cdot \mathbf{n}$ is used.

The flux vector $\mathbf{F}$ may be decomposed as $\mathbf{F}(W) = F_1(W) \, \mathbf{e}_x + F_2(W) \, \mathbf{e}_y + F_3(W) \, \mathbf{e}_z$ with

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u w \\ (\rho e + p) u \end{pmatrix}, \; F_2(W) = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v w \\ (\rho e + p) v \end{pmatrix} \text{ and } F_3(W) = \begin{pmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + p \\ (\rho e + p) w \end{pmatrix} .$$

These vectors may be expressed only in terms of conservative variables:

$$F_1(W) = \begin{pmatrix} \rho u \\ \dfrac{(\rho u)^2}{\rho} + p \\ \dfrac{\rho u \, \rho v}{\rho} \\ \dfrac{\rho u \, \rho w}{\rho} \\ (\rho e + p) \dfrac{\rho u}{\rho} \end{pmatrix}, \; F_2(W) = \begin{pmatrix} \rho v \\ \dfrac{\rho u \, \rho v}{\rho} \\ \dfrac{(\rho v)^2}{\rho} + p \\ \dfrac{\rho v \, \rho w}{\rho} \\ (\rho e + p) \dfrac{\rho v}{\rho} \end{pmatrix} \text{ and } F_3(W) = \begin{pmatrix} \rho w \\ \dfrac{\rho u \, \rho w}{\rho} \\ \dfrac{\rho v \, \rho w}{\rho} \\ \dfrac{(\rho w)^2}{\rho} + p \\ (\rho e + p) \dfrac{\rho w}{\rho} \end{pmatrix} .$$

where p is written $p = (\gamma - 1)\rho e - \dfrac{\gamma - 1}{2} \dfrac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{\rho}$.

Normal flux for a normal vector $\mathbf{n}$ writes:

$$\mathbf{F}(W) \; = \; \mathcal{F}(W) \cdot \mathbf{n} \; = \; \begin{pmatrix} \rho u^{nor} \\ \rho u u^{nor} + p n_x \\ \rho v u^{nor} + p n_y \\ \rho w u^{nor} + p n_z \\ (\rho e + p) \, u^{nor} \end{pmatrix} . \tag{D.1}$$

With the above expression, we are now able to differentiate these vectors with respect to the conservative variables. The Jacobian of p with respect to the conservative variable, after simplifications, is:

$$\frac{\partial p(W)}{\partial W} = (\gamma-1)\,{}^t\left(\frac{q^2}{2}\,,\,-u\,,\,-v\,,\,-w\,,\,1\right) \tag{D.2}$$

where $q^2 = u^2+v^2+w^2$. After simplifications, the Jacobian reads:

$$\frac{\partial F_1(W)}{\partial W} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{(\gamma-1)}{2}q^2-u^2 & -(\gamma-3)u & -(\gamma-1)v & -(\gamma-1)w & \gamma-1 \\ -uv & v & u & 0 & 0 \\ -uw & w & 0 & u & 0 \\ (\gamma-1)uq^2-\gamma ue & \gamma e-\frac{\gamma-1}{2}\left(2u^2+q^2\right) & -(\gamma-1)uv & -(\gamma-1)uw & \gamma u \end{pmatrix},$$

$$\frac{\partial F_2(W)}{\partial W} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ -uv & v & u & 0 & 0 \\ \frac{(\gamma-1)}{2}q^2-v^2 & -(\gamma-1)u & -(\gamma-3)v & -(\gamma-1)w & \gamma-1 \\ -vw & 0 & w & v & 0 \\ (\gamma-1)vq^2-\gamma vE & -(\gamma-1)uv & \gamma e-\frac{\gamma-1}{2}\left(2v^2+q^2\right) & -(\gamma-1)vw & \gamma v \end{pmatrix},$$

$$\frac{\partial F_3(W)}{\partial W} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ -uw & w & 0 & u & 0 \\ -vw & 0 & w & v & 0 \\ \frac{(\gamma-1)}{2}q^2-w^2 & -(\gamma-1)u & -(\gamma-1)v & -(\gamma-3)w & \gamma-1 \\ (\gamma-1)wq^2-\gamma wE & -(\gamma-1)uw & -(\gamma-1)vw & \gamma e-\frac{\gamma-1}{2}\left(2w^2+q^2\right) & \gamma w \end{pmatrix}.$$

Note that we have $(\gamma-1)q^2-\gamma e = (\gamma-1)\dfrac{q^2}{2}-h$.
Thus, we deduce:

$$\mathcal{A}(W) = \frac{\partial\left(\mathbf{F}(W)\cdot\mathbf{n}\right)}{\partial W}$$

$$= \begin{pmatrix} 0 & n_x & n_y & n_z & 0 \\ \frac{(\gamma-1)}{2}q^2 n_x - u\,u^{nor} & u^{nor}-(\gamma-2)un_x & u\,n_y-(\gamma-1)vn_x & u\,n_z-(\gamma-1)wn_x & (\gamma-1)n_x \\ \frac{(\gamma-1)}{2}q^2 n_y - vu^{nor} & v\,n_x-(\gamma-1)un_y & u^{nor}-(\gamma-2)vn_y & v\,n_z-(\gamma-1)wn_y & (\gamma-1)n_y \\ \frac{(\gamma-1)}{2}q^2 n_z - wu^{nor} & w\,n_x-(\gamma-1)un_z & w\,n_y-(\gamma-1)vn_z & u^{nor}-(\gamma-2)wn_z & (\gamma-1)n_z \\ u^{nor}\left((\gamma-1)q^2-\gamma e\right) & \left(\frac{p}{\rho}+e\right)n_x-(\gamma-1)uu^{nor} & \left(\frac{p}{\rho}+e\right)n_y-(\gamma-1)vu^{nor} & \left(\frac{p}{\rho}+e\right)n_z-(\gamma-1)wu^{nor} & \gamma u^{nor} \end{pmatrix}.$$

$\mathcal{A}$ is diagonalizable and is decomposed as $\mathcal{A} = \mathcal{T}\,\mathcal{D}\,\mathcal{T}^{-1}$. Matrices $\mathcal{D}$, $\mathcal{T}$ and $\mathcal{T}^{-1}$ are:

$$\mathcal{D} = \begin{pmatrix} u^{nor} & 0 & 0 & 0 & 0 \\ 0 & u^{nor} & 0 & 0 & 0 \\ 0 & 0 & u^{nor} & 0 & 0 \\ 0 & 0 & 0 & u^{nor}+c & 0 \\ 0 & 0 & 0 & 0 & u^{nor}-c \end{pmatrix}, \tag{D.3}$$

$$\mathcal{T} = \frac{1}{c^2}\begin{pmatrix} n_x & n_y & n_z & \frac{1}{2} & \frac{1}{2} \\ un_x & un_y - n_z & un_z + n_y & \frac{u+cn_x}{2} & \frac{u-cn_x}{2} \\ vn_x + n_z & vn_y & vn_z - n_x & \frac{v+cn_y}{2} & \frac{v-cn_y}{2} \\ wn_x - n_y & wn_y + n_x & wn_z & \frac{w+cn_z}{2} & \frac{w-cn_z}{2} \\ \frac{q^2}{2}n_x + vn_z - wn_y & \frac{q^2}{2}n_y + wn_x - un_z & \frac{q^2}{2}n_z + un_y - vn_x & \frac{h+cu^{nor}}{2} & \frac{h-cu^{nor}}{2} \end{pmatrix}, \tag{D.4}$$

$$\mathcal{T}^{-1} = \begin{pmatrix} (c^2 - \frac{(\gamma-1)\,q^2}{2})n_x + (wn_y - vn_z)c^2 & (\gamma-1)\,un_x & (\gamma-1)\,vn_x + c^2 n_z & (\gamma-1)\,wn_x - c^2 n_y & -(\gamma-1)\,n_x \\ (c^2 - \frac{(\gamma-1)\,q^2}{2})n_y + (un_z - wn_x)c^2 & (\gamma-1)\,un_y - c^2 n_z & (\gamma-1)\,vn_y & (\gamma-1)\,wn_y + c^2 n_x & -(\gamma-1)\,n_y \\ (c^2 - \frac{(\gamma-1)\,q^2}{2})n_z + (vn_x - un_y)c^2 & (\gamma-1)\,un_z + c^2 n_y & (\gamma-1)\,vn_z - c^2 n_x & (\gamma-1)\,wn_z & -(\gamma-1)\,n_z \\ \frac{(\gamma-1)\,q^2}{2} - c\,u^{nor} & cn_x - (\gamma-1)\,u & cn_y - (\gamma-1)\,v & cn_z - (\gamma-1)\,w & \gamma-1 \\ \frac{(\gamma-1)\,q^2}{2} + c\,u^{nor} & -cn_x - (\gamma-1)\,u & -cn_y - (\gamma-1)\,v & -cn_z - (\gamma-1)\,w & \gamma-1 \end{pmatrix}, \tag{D.5}$$

where $h = \dfrac{c^2}{\gamma-1} + \dfrac{q^2}{2}$ is the enthalpy per unit mass.

D.2 ALE mirror state description

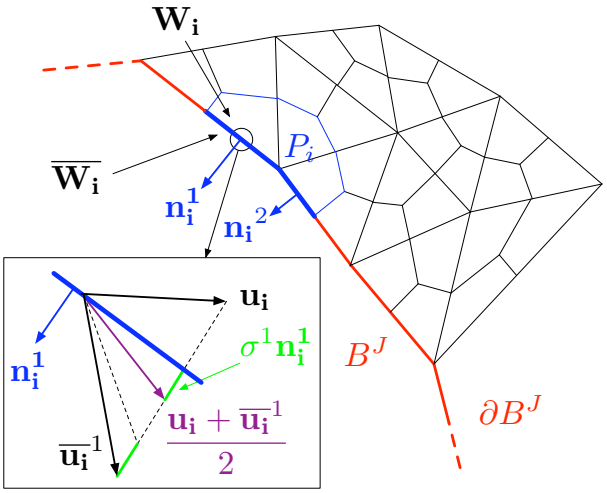


Figure D.1: Mirror state across boundary interface k of cell C_i.

Mirror state $\overline{W}$ associated with boundary state W is an imaginary state, virtually defined on the other side of the boundary and such that the extrapolated value of $W_{|\partial B} = (W + \overline{W})/2$ on the boundary satisfies:

$$(u^{nor})_{|\partial B} \;=\; \sigma_{|\partial B}\,.$$

We consider a Finite Volume cell C_i touching a moving object B, and we note $\mathbf{n}_i^k$ the outward normalized normal to the k^{th} boundary facet and σ_i^k its normal speed, see Figure 5.5. The ALE mirror state associated with state $W_{i,k}$ on the other side of boundary interface k of cell C_i is defined by:

$$\left\{ \begin{array}{l} \overline{\rho}_i \;=\; \rho_i \,, \\ \overline{\mathbf{u}}_i^k \;=\; \mathbf{u}_i^k - 2\left(\mathbf{u}_i^k.\mathbf{n}_i^k - \sigma_i^k\right)\mathbf{n}_i^k \,, \quad . \\ \overline{\varepsilon}_i = \varepsilon_i \,. \end{array} \right.$$

Note that with this definition, we indeed have $\left(\mathbf{u}_i^k\right)_{|\partial B} \cdot \left(\mathbf{n}_i^k\right)_{|\partial B} = \left(\sigma_i^k\right)_{\partial B}$.
For convenience, indices $\frac{k}{i}$ are dropped in the sequel. The squared norm of the fluid velocity associated with mirror state $\overline{W}$ reads:

$$\begin{array}{rl} \overline{q}^2 & = ||\overline{\mathbf{u}}||^2 = ||\mathbf{u} - 2(u^{nor})\mathbf{n} + 2\sigma\mathbf{n}||^2 \\ & = ||\mathbf{u} - 2u^{nor}\mathbf{n}||^2 + (2\sigma)^2 - 4\sigma\mathbf{u} - (2u^{nor}\mathbf{n})\cdot\mathbf{n} \\ & = q^2 + 4\sigma(\sigma - u^{nor})\,. \end{array}$$

The total energy per unit mass of the mirror state hence reads:

$$\overline{e} = \overline{\varepsilon} + \frac{1}{2}\overline{q}^2 = \varepsilon + \frac{1}{2}q^2 - 2\sigma(\sigma - u^{nor}) = e - 2\sigma(\sigma - u^{nor})\,.$$

The conservative mirror state finally writes:

$$\overline{W} = \left(\begin{array}{l} \rho \\ \rho\left(u - 2\left(u^{nor} - \sigma\right)n_x\right) \\ \rho\left(v - 2\left(u^{nor} - \sigma\right)n_y\right) \\ \rho\left(w - 2\left(u^{nor} - \sigma\right)n_z\right) \\ \rho\left(e - 2\sigma\left(u^{nor} - \sigma\right)\right) \end{array} \right) . \tag{D.6}$$

Other quantities relative to mirror state $\overline{W}$ can also be computed:

$$\overline{p} \;=\; (\gamma - 1)\,\overline{\rho\varepsilon} \;=\; p\,,$$

$$\overline{h} \;=\; \overline{e} + \frac{\overline{p}}{\overline{\rho}} = h - 2\sigma\left(u^{nor} - \sigma\right)\,,$$

and finally,

$$\overline{c} \;=\; (\gamma - 1)\left(\overline{h} - \frac{q^2}{2}\right) \;=\; (\gamma - 1)\left(\overline{\varepsilon} + \frac{\overline{p}}{\overline{\rho}}\right) \;=\; c\,.$$

The average Roe state $\tilde{W}$ between W and its mirror state $\overline{W}$ is defined by:

$$\tilde{W} \;=\; \left\{ \begin{array}{l} \tilde{\rho} \;=\; \sqrt{\rho\overline{\rho}} \;=\; \rho \\ \\ \tilde{\mathbf{u}} \;=\; \dfrac{\sqrt{\rho}\mathbf{u} + \sqrt{\overline{\rho}}\overline{\mathbf{u}}}{\rho + \overline{\rho}} \;=\; \dfrac{\overline{\mathbf{u}} + \mathbf{u}}{2} \;=\; \mathbf{u} - \left(u^{nor} - \sigma\right)\mathbf{n} \\ \\ \tilde{h} \;=\; \dfrac{\sqrt{\rho}h + \sqrt{\overline{\rho}}\overline{h}}{\sqrt{\rho} + \sqrt{\overline{\rho}}} \;=\; \dfrac{h + \overline{h}}{2} \;=\; h - \sigma\left(u^{nor} - \sigma\right)\,. \end{array} \right.$$

Thus, we get:

$$\tilde{q}^2 \;=\; ||\tilde{\mathbf{u}}||^2 = ||\underbrace{\left(\mathbf{u} - u^{nor}\mathbf{n}\right)}_{\text{orthogonal to } \mathbf{n}} + (\sigma\mathbf{n})||^2 \;=\; ||\mathbf{u} - u^{nor}\mathbf{n}||^2 \;+\; \sigma^2$$

Finally, if we note $\mathbf{u}^{tan} = \mathbf{u} - u^{nor}\mathbf{n}$, the celerity associated with Roe average state between W and $\overline{W}$ writes:

$$\begin{aligned}\tilde{c} &= (\gamma-1)\left(\tilde{h} - \frac{\tilde{q}^2}{2}\right) = (\gamma-1)\left(h - \sigma\,(u^{nor}-\sigma) - \frac{||\mathbf{u}^{tan}||^2}{2} - \frac{\sigma^2}{2}\right)\\ &= (\gamma-1)\left(h - \sigma\,u^{nor} - \frac{||\mathbf{u}^{tan}||^2}{2} + \frac{\sigma^2}{2}\right).\end{aligned}$$

Note that this Roe average state $\tilde{W}$ also satisfies:

$$\tilde{\mathbf{u}}\cdot\mathbf{n} = \sigma\,. \tag{D.7}$$

To end this section, we compute several quantities useful for the upcoming calculus:

$$W - \overline{W} = \begin{pmatrix} 0 \\ 2\rho\,(u^{nor}-\sigma)\,n_x \\ 2\rho\,(u^{nor}-\sigma)\,n_y \\ 2\rho\,(u^{nor}-\sigma)\,n_z \\ 2\rho\sigma\,(u^{nor}-\sigma) \end{pmatrix}. \tag{D.8}$$

To finish with, we give:

$$\mathbf{F}\,(W) - \sigma W = \begin{pmatrix} \rho(u^{nor}-\sigma) \\ \rho u(u^{nor}-\sigma) + pn_x \\ \rho v(u^{nor}-\sigma) + pn_y \\ \rho w(u^{nor}-\sigma) + pn_z \\ \rho e\,(u^{nor}-\sigma) + pu^{nor} \end{pmatrix},$$

and, for mirror state $\overline{W}$, we replace variables associated with W by the ones associated with $\overline{W}$ in the above expression:

$$\begin{aligned}\mathbf{F}\left(\overline{W}\right) - \sigma\overline{W} &= \begin{pmatrix} \rho(-u^{nor}+2\sigma-\sigma) \\ \rho\,[u+2(-u^{nor}+2\sigma-\sigma)n_x]\,[-u^{nor}+2\sigma-\sigma] + pn_x) \\ \rho\,[v+2(-u^{nor}+2\sigma-\sigma)n_y]\,[-u^{nor}+2\sigma-\sigma] + pn_y) \\ \rho\,[w+2(-u^{nor}+2\sigma-\sigma)n_z]\,[-u^{nor}+2\sigma-\sigma] + pn_z) \\ \rho\,[e-2\sigma\,(u^{nor}-\sigma)]\,(-u^{nor}+2\sigma-\sigma) + p(-u^{nor}+2\sigma) \end{pmatrix}\\ &= \begin{pmatrix} -\rho(u^{nor}-\sigma) \\ -\rho u(u^{nor}-\sigma) + 2\rho(u^{nor}-\sigma)^2 n_x + pn_x \\ -\rho v(u^{nor}-\sigma) + 2\rho(u^{nor}-\sigma)^2 n_y + pn_y \\ -\rho w(u^{nor}-\sigma) + 2\rho(u^{nor}-\sigma)^2 n_z + pn_z \\ -(\rho e+p)\,(u^{nor}-\sigma) + 2p\sigma + 2\rho\sigma(u^{nor}-\sigma)^2) \end{pmatrix}\\ &= -\left[\mathbf{F}\,(W) - \sigma W\right] + 2\begin{pmatrix} 0 \\ \rho(u^{nor}-\sigma)^2 n_x + pn_x \\ \rho(u^{nor}-\sigma)^2 n_y + pn_y \\ \rho(u^{nor}-\sigma)^2 n_z + pn_z \\ p\sigma + \rho\sigma(u^{nor}-\sigma)^2 \end{pmatrix}.\end{aligned} \tag{D.9}$$

D.3 Roe ALE mirror state slipping boundary condition

The Roe ALE numerical flux function $\mathbf{\Phi}^{\mathrm{Roe}}$ computed between states W and $\overline{W}$ is given by:

$$\mathbf{\Phi}^{\mathrm{Roe}}\left(W,\,\overline{W},\,\mathbf{n},\,\sigma||\boldsymbol{\eta}||\right) = ||\boldsymbol{\eta}||\left(\frac{\mathbf{F}(W)+\mathbf{F}(\overline{W})}{2} - \sigma\,\frac{W+\overline{W}}{2} + |\tilde{\mathcal{A}} - \sigma\mathcal{I}_n|\,\frac{|W-\overline{W}|}{2}\right),$$

where $\tilde{\mathcal{A}} = \mathcal{A}(\tilde{W}, \mathbf{n})$ is the Jacobian matrix of normal continuous flux $\mathbf{F}$ evaluated for Roe average state $\tilde{W}$ between W and $\overline{W}$.
We have:

$$\tilde{\mathcal{A}} - \sigma\mathcal{I}_n = \tilde{\mathcal{T}} \left(\tilde{\mathcal{D}} - \sigma\mathcal{I}_n \right) \tilde{\mathcal{T}}^{-1} ,$$

where $\tilde{\mathcal{D}}$, $\tilde{\mathcal{T}}$ and $\tilde{\mathcal{T}}^{-1}$ are computed by evaluating Matrices (D.3), (D.4) and (D.5) for average Roe state $\tilde{W}$ between W and $\overline{W}$.

D.3.1 Upwinding term computation

First, we focus on the computation of the upwinding term appearing in Roe ALE numerical flux:

$$|\tilde{\mathcal{A}} - \sigma\mathcal{I}_n| \frac{|W - \overline{W}|}{2} .$$

According to Relation (D.7), Roe average state between W and $\overline{W}$ satisfies $\tilde{u}^{nor} = \sigma$, which leads to:

$$\begin{aligned} \tilde{\mathcal{D}} - \sigma\mathcal{I}_n &= \tilde{\mathcal{D}}' = \operatorname{diag}\left(\tilde{u}^{nor} - \sigma,\, \tilde{u}^{nor} - \sigma,\, \tilde{u}^{nor} - \sigma,\, \tilde{u}^{nor} - \sigma + \tilde{c},\, \tilde{u}^{nor} - \sigma - \tilde{c}\right) \\ &= \operatorname{diag}\left(\tilde{d}'_1 = 0,\, \tilde{d}'_2 = 0,\, \tilde{d}'_3 = 0,\, \tilde{d}'_4 = \tilde{c}, \tilde{d}'_5 = -\tilde{c}\right) . \end{aligned}$$

Vector $W - \overline{W}$, given by Relation (D.8), can be decomposed on the eigen-basis $(\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_5)$ of $\tilde{\mathcal{A}} - \sigma\mathcal{I}_n$, which is the same as the one of $\tilde{\mathcal{A}}$:

$$W - \overline{W} = \sum_{k=1}^{5} \alpha_k \tilde{\mathbf{r}} ,$$

and :

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{pmatrix} = \tilde{T}^{-1} \begin{pmatrix} 0 \\ 2\rho\left(u^{nor} - \sigma\right) n_x \\ 2\rho\left(u^{nor} - \sigma\right) n_y \\ 2\rho\left(u^{nor} - \sigma\right) n_z \\ 2\rho\sigma\left(u^{nor} - \sigma\right) \end{pmatrix} = 2\rho\left(u^{nor} - \sigma\right) \begin{pmatrix} \left(\gamma - 1\right)\left(\tilde{u}^{nor} - \sigma\right) n_x \\ \left(\gamma - 1\right)\left(\tilde{u}^{nor} - \sigma\right) n_y \\ \left(\gamma - 1\right)\left(\tilde{u}^{nor} - \sigma\right) n_z \\ \tilde{c} - \left(\gamma - 1\right)\left(\tilde{u}^{nor} - \sigma\right) \\ -\tilde{c} - \left(\gamma - 1\right)\left(\tilde{u}^{nor} - \sigma\right) \end{pmatrix} = 2\rho\left(u^{nor} - \sigma\right) \begin{pmatrix} 0 \\ 0 \\ 0 \\ \tilde{c} \\ -\tilde{c} \end{pmatrix} .$$

For the "viscous" part of the flux, using Expression (D.8), we get:

$$|\tilde{\mathcal{A}} - \sigma\mathcal{I}_n| \frac{|W - \overline{W}|}{2} = \sum_{k=1}^{5} |\tilde{d}'_k| \alpha_k \tilde{\mathbf{r}}_k .$$

As $\tilde{d}'_1 = \tilde{d}'_2 = \tilde{d}'_3 = 0$, the above summation simplifies to:

$$|\tilde{\mathcal{A}} - \sigma\mathcal{I}_n| \frac{|W - \overline{W}|}{2} = \tilde{c}\alpha_4\tilde{\mathbf{r}}_4 + \tilde{c}\alpha_5\tilde{\mathbf{r}}_5 = 2\rho\left(u^{nor} - \sigma\right)\tilde{c}^2\left(\tilde{\mathbf{r}}_4 - \tilde{\mathbf{r}}_5\right) . .$$

$\tilde{\mathbf{r}}_4$ and $\tilde{\mathbf{r}}_5$ are respectively the fourth and last column of $\tilde{\mathcal{T}}$, and we get, according to Expression (D.4):

$$|\tilde{\mathcal{A}} - \sigma\mathcal{I}_n| \, |W - \overline{W}| = 2\rho\left(u^{nor} - \sigma\right)\tilde{c}^2 \left[\begin{pmatrix} \frac{1}{2} \\ \frac{\tilde{u} + \tilde{c}n_x}{2} \\ \frac{\tilde{u} + \tilde{c}n_y}{2} \\ \frac{\tilde{u} + \tilde{c}n_z}{2} \\ \frac{\tilde{h} + \tilde{c}\tilde{u}^{nor}}{2} \end{pmatrix} - \begin{pmatrix} \frac{1}{2} \\ \frac{\tilde{u} - \tilde{c}n_x}{2} \\ \frac{\tilde{u} - \tilde{c}n_y}{2} \\ \frac{\tilde{u} - \tilde{c}n_z}{2} \\ \frac{\tilde{h} - \tilde{c}\tilde{u}^{nor}}{2} \end{pmatrix} \right] = 2\rho\left(u^{nor} - \sigma\right)\tilde{c}^2 \begin{pmatrix} 0 \\ \tilde{c}n_x \\ \tilde{c}n_y \\ \tilde{c}n_z \\ \tilde{c}\underbrace{\tilde{u}^{nor}}_{=\sigma} \end{pmatrix}$$

D.3.1.1 Centered term computation

We are now interested in the computation of the centered term:

$$\frac{\mathbf{F}(W)+\mathbf{F}(\overline{W})}{2} - \sigma\frac{W+\overline{W}}{2}\,.$$

According to Relation (D.9), we immediately get:

$$\frac{\mathbf{F}(W)+\mathbf{F}(\overline{W})}{2} - \sigma\frac{W+\overline{W}}{2} = \frac{1}{2}\left[(\mathbf{F}(W)-\sigma W)+\left(\mathbf{F}(\overline{W})-\sigma\overline{W}\right)\right] = \begin{pmatrix} 0 \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2\right)n_x \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2\right)n_y \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2\right)n_z \\ \sigma p+\rho\sigma\left(u^{nor}-\sigma\right)^2 \end{pmatrix}$$

Finally, the complete Roe flux is obtained by adding the upwind and the centered parts:

$$\begin{aligned}
\mathbf{\Phi}^{\text{Roe}}\left(W,\,\overline{W},\,\mathbf{n},\,\sigma||\boldsymbol{\eta}||\right) &= ||\boldsymbol{\eta}||\left(\frac{\mathbf{F}(W)+\mathbf{F}(\overline{W})}{2} - \sigma\frac{W+\overline{W}}{2} + |\tilde{\mathcal{A}}-\sigma\mathcal{I}_n|\frac{|W-\overline{W}|}{2}\right) \\
&= ||\boldsymbol{\eta}||\begin{pmatrix} 0 \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2\right)n_x \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2\right)n_y \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2\right)n_z \\ \sigma p+\rho\sigma\left(u^{nor}-\sigma\right)^2 \end{pmatrix} + \rho||\boldsymbol{\eta}||\left(u^{nor}-\sigma\right)\begin{pmatrix} 0 \\ \tilde{c}n_x \\ \tilde{c}n_y \\ \tilde{c}n_z \\ \tilde{c}\sigma \end{pmatrix} \\
&= ||\boldsymbol{\eta}||\begin{pmatrix} 0 \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2+\rho\tilde{c}\left(u^{nor}-\sigma\right)\right)n_x \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2+\rho\tilde{c}\left(u^{nor}-\sigma\right)\right)n_y \\ \left(p+\rho\left(u^{nor}-\sigma\right)^2+\rho\tilde{c}\left(u^{nor}-\sigma\right)\right)n_z \\ p\sigma+\rho\sigma\left(u^{nor}-\sigma\right)\left(u^{nor}-\sigma+\tilde{c}\right) \end{pmatrix},
\end{aligned}$$

where

$$\tilde{c} = (\gamma-1)\left(h-\sigma u^{nor}-\frac{||\mathbf{u}^{tan}||^2}{2}+\frac{\sigma^2}{2}\right) \quad \text{and} \quad \mathbf{u}^{tan} = \mathbf{u}-u^{nor}\mathbf{n}\,.$$

If the boundary condition is actually enforced, *i.e.* $u^{nor}=\sigma$, this flux writes:

$$\mathbf{\Phi}^{\text{Roe}}\left(W,\,\overline{W},\,\mathbf{n},\,\sigma||\boldsymbol{\eta}||\right) = ||\eta||\begin{pmatrix} 0 \\ p\mathbf{n} \\ p\sigma \end{pmatrix}.$$

D.4 HLLC ALE mirror state boundary condition

The HLLC flux between state W and its associated mirror state $\overline{W}$ is described by the following three waves phase velocities:

$$S = \min u^{nor}-c,\,\tilde{u}^{nor}-\tilde{c}\,, \quad \overline{S} = \max \overline{u}^{nor}+\overline{c},\,\tilde{u}^{nor}+\tilde{c} \quad \text{and} \quad S_M = \frac{\overline{u}^{nor}(\overline{c}-\overline{u}^{nor})-\rho u^{nor}(c-u^{nor})+p-\overline{p}}{\overline{\rho}(\overline{c}-\overline{u}^{nor})-\rho(c-u^{nor})}\,,$$

and two approximate states W^* and $\overline{W}^*$:

$$W^* = \begin{cases} \rho^* = \rho \dfrac{S - u^{nor}}{S - S_M} \\ p^* = \rho \left(u^{nor} - S\right)\left(u^{nor} - S_M\right) + p \\ (\rho \mathbf{u})^* = \dfrac{\left(S - u^{nor}\right)\rho \mathbf{u} + \left(p^* - p\right)\mathbf{n}}{S - S_M} \\ (\rho e)^* = \dfrac{\left(S - u^{nor}\right)\rho e - p u^{nor} + p^* S_M}{S - S_M} \end{cases} \qquad \overline{W}^* = \begin{cases} \overline{\rho}^* = \overline{\rho} \dfrac{\overline{S} - \overline{u}^{nor}}{\overline{S} - S_M} \\ \overline{p}^* = p^* = \overline{\rho}\left(\overline{u}^{nor} - \overline{S}\right)\left(\overline{u}^{nor} - S_M\right) + \overline{p} \\ (\overline{\rho}\overline{\mathbf{u}})^* = \dfrac{\left(\overline{S} - \overline{u}^{nor}\right)\overline{\rho}\overline{\mathbf{u}} + \left(p^* - \overline{p}\right)\mathbf{n}}{\overline{S} - S_M} \\ (\overline{\rho e})^*_R = \dfrac{\left(\overline{S} - \overline{u}^{nor}\right)\overline{\rho e} - \overline{p} u^{nor} + p^* S_M}{\overline{S} - S_M} \end{cases}$$

The HLLC flux through the interface is given by:

$$\mathbf{\Phi}^{\text{Hllc}}\left(W, \overline{W}, \mathbf{n}, \sigma||\boldsymbol{\eta}||\right) = ||\boldsymbol{\eta}|| \begin{cases} \mathbf{F}(W) - \sigma W & \text{if } S - \sigma > 0 \\ \mathbf{F}(W^*) - \sigma W^* & \text{if } S - \sigma \leq 0 < S_M - \sigma \\ \mathbf{F}(\overline{W}^*) - \sigma \overline{W}^* & \text{if } S_M - \sigma \leq 0 \leq \overline{S} - \sigma \\ \mathbf{F}(\overline{W}) - \sigma \overline{W} & \text{if } \overline{S} - \sigma < 0 \end{cases}$$

According to Relation (D.7), $\tilde{u}^{nor} = \sigma$ and using the definition of mirror state $\overline{W}$ associated with W:

$$\overline{\rho} = \rho, \quad \overline{p} = p \quad \text{and} \quad \overline{u}^{nor} = -u^{nor} + 2\sigma\,,$$

we have:

$$S = \min\left(u^{nor} - c,\, \tilde{u}^{nor} - \tilde{c}\right) = \min\left(u^{nor} - c,\, \sigma - \tilde{c}\right)$$

$$\text{and} \quad \overline{S} = \max\left(\overline{u}^{nor} + \overline{c}, \tilde{u}^{nor} - \tilde{c}\right) = \max\left(-u^{nor} + 2\sigma + c, \tilde{c} + \sigma\right)\,.$$

In any case, we have $\overline{S} = 2\sigma - S$. Indeed:

$$\begin{aligned} \text{If } u^{nor} - c \leq \sigma - \tilde{c} \quad \text{then} \quad & -u^{nor} + c \geq \tilde{c} - \sigma \\ \Rightarrow \quad & -u^{nor} + c + 2\sigma \geq \tilde{c} + \sigma \\ \Rightarrow \quad & S = u^{nor} - c \quad \text{and} \quad \overline{S} = 2\sigma - u^{nor} + c \\ \Rightarrow \quad & \overline{S} = 2\sigma - S\,, \\ \text{If } u^{nor} - c \geq \sigma - \tilde{c} \quad \text{then} \quad & -u^{nor} + c \leq \tilde{c} - \sigma \\ \Rightarrow \quad & -u^{nor} + c + 2\sigma \leq \tilde{c} + \sigma \\ \Rightarrow \quad & S = \sigma - \tilde{c} \quad \text{and} \quad \overline{S} = \sigma + \tilde{c} \\ \Rightarrow \quad & \overline{S} = 2\sigma - S\,. \end{aligned}$$

Besides, the middle wave speed writes:

$$S_M = \frac{\overline{\rho u}^{nor}\left(\overline{S}-\overline{u}^{nor}\right)-\rho u^{nor}\left(S-u^{nor}\right)-\left(\overline{p}-p\right)}{\overline{\rho}\left(\overline{S}-\overline{u}^{nor}\right)-\rho\left(S-u^{nor}\right)}$$

$$= \frac{\rho\left(-u^{nor}+2\sigma\right)\left(2\sigma-S+u^{nor}-2\sigma\right)-\rho u^{nor}\left(S-u^{nor}\right)-\overbrace{\left(\overline{p}-p\right)}^{0}}{\rho\left(2\sigma-S+u^{nor}-\sigma\right)-\rho\left(S-u^{nor}\right)} = \frac{2\rho\sigma\left(u^{nor}-S\right)}{2\rho\left(u^{nor}-S\right)} = \sigma\,.$$

We now compute the fluxes involved in the HLLC solver.

$$\mathbf{F}\left(W^*\right)-\sigma W^* = \begin{pmatrix} \rho^*(u^{nor})^* \\ \rho^* u^*(u^{nor})^* + p^* n_x \\ \rho^* v^*(u^{nor})^* + p^* n_y \\ \rho^* w^*(u^{nor})^* + p^* n_z \\ \left(\rho^* e^* + p^*\right)(u^{nor})^* \end{pmatrix} - \begin{pmatrix} \sigma\rho^* \\ \sigma\rho^* u^* \\ \sigma\rho^* v^* \\ \sigma\rho^* w^* \\ \sigma\left(\rho^* e^* + p^*\right) \end{pmatrix}. \tag{D.10}$$

Taking the dot product of $\rho^*\mathbf{u}^*$ and $\mathbf{n}$, we compute:

$$\rho^*(u^{nor})^* = \frac{(S-u^{nor})\rho u^{nor}+(p^*-p)}{S-\sigma}\,,$$

and using the expression of ρ^* and p^*, we deduce:

$$(u^{nor})^* = \frac{(S-u^{nor})\rho u^{nor}+(p^*-p)}{\rho\left(S-u^{nor}\right)} = \frac{(S-u^{nor})\rho u^{nor}+\rho(u^{nor}-S)(u^{nor}-S_M)}{\rho\left(S-u^{nor}\right)}$$

$$= u^{nor}-(u^{nor}-S_M) = S_M = \sigma\,.$$

Substituting $(u^{nor})^*$ by its value in Expression (D.10), we get:

$$\mathbf{F}\left(W^*\right)-\sigma W^* = \begin{pmatrix} 0 \\ p^* n_x \\ p^* n_y \\ p^* n_z \\ p^*\sigma \end{pmatrix},$$

with

$$p* = \rho\left(u^{nor}-S\right)\left(u^{nor}-\sigma\right)+p = \rho\left(u^{nor}-\sigma\right)\max\left(c,\tilde{c}+u^{nor}-\sigma\right)+p\,.$$

A similar computation leads to:

$$\mathbf{F}\left(\overline{W}^*\right)-\sigma\overline{W}^* = \begin{pmatrix} 0 \\ \overline{p}^* n_x \\ \overline{p}^* n_y \\ \overline{p}^* n_z \\ \overline{p}^*\sigma \end{pmatrix},$$

with

$$\overline{p}^* = \overline{\rho}\left(\overline{u}^{nor}-\overline{S}\right)\left(\overline{u}^{nor}-\sigma\right)+\overline{p}$$

$$= \rho\left(-u^{nor}+2\sigma-(2\sigma-S)\right)\left(-u^{nor}+2\sigma-\sigma\right)+p = \rho\left(-u^{nor}+S\right)\left(-u^{nor}+\sigma\right)+p = p^*\,.$$

D.5 ALE mirror slipping boundary conditions coupled with DGCL Runge-Kutta schemes

2D case. Let $\mathbf{e} = (P_0,\, P_1)$ be a boundary edge. It is made of two Finite Volume interfaces: $I_0 = (P_0,\, M)$ associated with cell C_0 and $I_1 = (M,\, P_1)$ with C_1, see Figure (D.2). The area swept by boundary interface

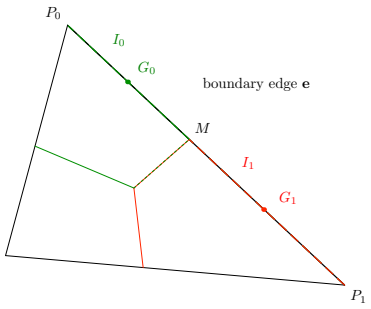


Figure D.2: A boundary edge and its two associated boundary Finite-Volume interfaces.

I_i between the initial and the current Runge-Kutta configuration s is:

$$A_i^s = \quad = c_s\tau\, \boldsymbol{w}_{G_i}^s \cdot \tilde{\boldsymbol{\eta}}_i^s\,,$$

where $\boldsymbol{w}_{G_i}^s$ is the velocity of the center of gravity G_i of interface I_i and $\tilde{\boldsymbol{\eta}}_{ij}^s$ is the pseudo-normal associated with I_i, computed between the initial and the current Runge-Kutta configurations.
However, the two interfaces are colinear, which means that their pseudo-normals have the same direction. Moreover, as median cells are used, their pseudo-normals have also the same norm, which is equal to half of the norm of the pseudo-normal to edge $\mathbf{e}$.

$$\tilde{\boldsymbol{\eta}}_i^s = \frac{1}{2}\tilde{\boldsymbol{\eta}}_{\mathbf{e}} = \frac{1}{4}\left(\boldsymbol{\eta}_e^0 + \boldsymbol{\eta}_e^s\right)\,.$$

In the sequel, the following notation is used:

$$\boldsymbol{w}_i^s \;=\; \frac{\mathbf{x}_i^s - \mathbf{x}_i^0}{c_s\tau}$$

We easily compute the gravity centers velocities $\boldsymbol{w}_{G_i}^s$:

$$\boldsymbol{w}_{G_i}^s \;=\; \frac{1}{2}\left(\boldsymbol{w}^s(P_i) + \boldsymbol{w}^s(M)\right) \;=\; \frac{1}{4}\left(3\boldsymbol{w}_i^s + \boldsymbol{w}_j^s\right)$$

and thus,

$$A_i^s \;=\; \frac{c_s\tau}{4}\left[3\boldsymbol{w}_i^s + \boldsymbol{w}_j^s\right] \cdot \frac{1}{4}\left(\boldsymbol{\eta}_e^0 + \boldsymbol{\eta}_e^s\right)\,.$$

The following ALE DGCL parameters,

$$\sigma_{i,\mathbf{e}}^s \;=\; \frac{A_i^s}{c_s\tau||\boldsymbol{\eta}_{\mathbf{e}}^n||}\,, \quad \boldsymbol{\eta}_i^n = \boldsymbol{\eta}_{\mathbf{e}}^n,$$

are finally used to compute boundary mass exchange between C_i and the outside through I_i, for instance using mirror states:

$$M_{i,\mathbf{e}}^{s,boundary} = c_s\tau\, \sigma_{i,\mathbf{e}}^s ||\boldsymbol{\eta}_{\mathbf{e}}^n||\, \Phi(W_i^s,\, \overline{W}_i^s,\, \sigma_{i,\mathbf{e}}^s,\, \boldsymbol{\eta}_{\mathbf{e}}^n)\,.$$

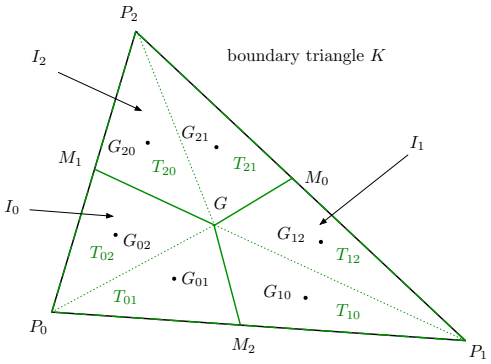


Figure D.3: A boundary triangle and its three associated boundary Finite-Volume interfaces. Each of these interfaces I_i is made of two sub-triangles T_{ij} and T_{ik}.

3D case. Let $K = (P_0, P_1, P_2)$ be a boundary triangle. It is made of three quadrangular Finite Volume interfaces: $I_0 = (P_0, M_2, G, M_1)$ associated with cell C_0, $I_1 = (P_1, M_0, G, M_2)$ with C_1 and $I_2 = (P_2, M_1, G, M_0)$ with C_2. Each interface I_i is made of two sub-triangles, noted T_{ij} and T_{ik} with $j, k \neq i$, see Figure (D.3).

The area swept by boundary interface I_i between the initial and the current Runge-Kutta configuration s is:

$$A_i^s = A_{T_{ij}}^s + A_{T_{ik}}^s \;=\; c_s\tau\, \boldsymbol{w}_{G_{ij}}^s \cdot \tilde{\boldsymbol{\eta}}_{ij}^s \;+\; c_s\tau\, \boldsymbol{w}_{G_{ik}}^s \cdot \tilde{\boldsymbol{\eta}}_{ik}^s \,,$$

where $\boldsymbol{w}_{G_{ij}}^s$ is the velocity of the center of gravity G_{ij} of triangle T_{ij} and $\tilde{\boldsymbol{\eta}}_{ij}^s$ is the pseudo-normal associated with T_{ij}, computed between the initial and the current Runge-Kutta configurations.

However, the six triangles T_{ij} are coplanar, which means that their pseudo-normals have the same direction. Moreover, as median cells are used, their pseudo-normals have also 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:

$$\tilde{\boldsymbol{\eta}}^s \;=\; \frac{1}{6}\tilde{\boldsymbol{\eta}}_K^s \;=\; \frac{1}{6}\left[\frac{1}{4}\overrightarrow{P_0P_1^0}\wedge\overrightarrow{P_0P_2^s} + \frac{1}{4}\overrightarrow{P_0P_1^s}\wedge\overrightarrow{P_0P_2^0} + \frac{1}{2}\overrightarrow{P_0P_1^0}\wedge\overrightarrow{P_0P_2^0} + \frac{1}{2}\overrightarrow{P_0P_1^s}\wedge\overrightarrow{P_0P_2^s}\right]$$

$$A_i^s \;=\; c_s\tau\left[\boldsymbol{w}_{G_{ij}}^s + \boldsymbol{w}_{G_{ik}}^s\right]\cdot\tilde{\boldsymbol{\eta}}^s \,,$$

In the sequel, the following notation is used:

$$\boldsymbol{w}_i^s \;=\; \frac{\mathbf{x}_i^s - \mathbf{x}_i^0}{c_s\tau}$$

We easily compute the gravity centers velocities $\boldsymbol{w}_{G_{ij}}^s$:

$$\boldsymbol{w}_{G_{ij}}^s \;=\; \frac{1}{18}\left(11\boldsymbol{w}_i^s + 5\boldsymbol{w}_j^s + 2\boldsymbol{w}_k^s\right)$$

and thus,

$$A_i^s \;=\; \frac{c_s\tau}{18}\left[22\boldsymbol{w}_i^s + 7\boldsymbol{w}_j^s + 7\boldsymbol{w}_k^s\right]\cdot\frac{1}{24}\left[\overrightarrow{P_0P_1^0}\wedge\overrightarrow{P_0P_2^s} + \overrightarrow{P_0P_1^s}\wedge\overrightarrow{P_0P_2^0} + 2\overrightarrow{P_0P_1^0}\wedge\overrightarrow{P_0P_2^0} + 2\overrightarrow{P_0P_1^s}\wedge\overrightarrow{P_0P_2^s}\right].$$

The following ALE DGCL parameters,

$$\sigma_{i,K}^s \;=\; \frac{A_i^s}{c_s\tau||\boldsymbol{\eta}_K^n||}\,, \quad \boldsymbol{\eta}_i^n = \boldsymbol{\eta}_K^n,$$

are finally used to compute boundary mass exchange between C_i and the outside through I_i, for instance using mirror states:

$$M_{i,K}^{s,boundary} = c_s \tau \, \sigma_{i,K}^s ||\boldsymbol{\eta}_K^n|| \, \Phi(W_i^s, \, \overline{W}_i^s, \, \sigma_{i,K}^s, \, \boldsymbol{\eta}_K^n) \, .$$

Bibliography

[Alauzet 2003a] F. Alauzet. *Adaptation de maillage anisotrope en trois dimensions. Application aux simulations instationnaires en Mécanique des Fluides.* PhD thesis, Université Montpellier II, Montpellier, France, 2003. (in French). 7, 8, 14, 70

[Alauzet 2003b] F. Alauzet and P.J. Frey. *Estimateur d'erreur géométrique et métrique anisotropes pour l'adaptation de maillage. Partie I : aspects théoriques.* RR-4759, INRIA, March 2003. (in French). 35, 36

[Alauzet 2006] F. Alauzet, A. Loseille, A. Dervieux and P.J. Frey. *Multi-dimensional continuous metric for mesh adaptation.* In Proceedings of the 15th International Meshing Roundtable, pages 191–214. Springer, 2006. 17

[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. 17, 60, 70, 71, 74, 89, 91, 102, 229

[Alauzet 2010a] F. Alauzet. *Size gradation control of anisotropic meshes.* Finite Elem. Anal. Des., vol. 46, pages 181–202, 2010. 96, 116

[Alauzet 2010b] F. Alauzet and A. Loseille. *High Order Sonic Boom Modeling by Adaptive Methods.* J. Comp. Phys., vol. 229, pages 561–593, 2010. 13, 17, 48

[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. 27, 189, 227

[Alauzet 2011] 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, Orlando, FL, USA, Jan 2011. 30

[Allain 2009] O. Allain, D. Guégan and F. Alauzet. *Studying the impact of unstructured mesh adaptation on free surface flow simulations.* In Proceedings of the ASME 28th International Conference on Ocean, Offshore and Arctic Engineering, 2009. 17, 60, 153

[Anderson 1982] D.A. Anderson and M.M. Rai. *The Use of Solution Adaptive Grids in Solving Partial Differential Equations.* Applied Mathematics and Computation, vol. 10-11, pages 317–338, 1982. 67

[Arsigny 2006] V. Arsigny, P. Fillard, X. Pennec and N. Ayache. *Log-Euclidean Metrics for Fast and Simple Calculus on Diffusion Tensors.* Magn. Reson. Med., vol. 56, no. 2, pages 411–421, 2006. 36, 37

[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. Comp., vol. 31, no. 6, pages 4041–4065, 2009. 60, 141

[Azarenok 2008] B.N. Azarenok. *Variational Method for Adaptive Mesh Generation.* Computational Mathematics and Mathematical Physics, vol. 48, no. 5, pages 786–804, 2008. 68

[Baines 1995] M. J. Baines. Moving Finite Elements. Monographs On Numerical Analysis. Oxford Science Publications, 1995. 66

[Baker 1997] T.J. Baker. *Mesh adaptation strategies for problems in fluid dynamics.* Finite Elem. Anal. Des., vol. 25, pages 243–273, 1997. 16

[Baker 1999] T. Baker and P Cavallo. *Dynamic Adaptation for Deforming Tetrahedral Meshes.* AIAA Journal, vol. 19, pages 2699–3253, 1999. 145

[Bakker 1997] A. Bakker, R. D. Laroche, M. H. Wang and R. V. Calabrese. *Sliding Mesh Simulation of Laminar Flow in Stirred Reactors.* Chem. Engnr. Resrch. Design., vol. 75, no. 1, pages 42–44, 1997. 141

[Batina 1990] J. Batina. *Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes.* AIAA Journal, vol. 28, no. 8, pages 1381–1388, 1990. 141, 145

[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. 172

[Baum 1989] J.D. Baum and R. Löhner. *Numerical Simulation of a Shock-Elevated Box Interaction Using an Adaptive Finite Element Shock Capturing Scheme.* In AIAA-89-0653, 1989. 224

[Baum 1991] J.D. Baum and R. Löhner. *Numerical Simulation of a Shock Interaction with a Modern Battlefield Tank.* In AIAA-91-1666, 1991. 141

[Baum 1993a] J.D. Baum and R. Löhner. *Numerical Simulation of a Pilot/Seat Ejection from an F-16.* In AIAA-93-0783, 1993. 141

[Baum 1993b] J.D. Baum, H. Luo and R. Löhner. *Numerical Simulation of a Blast Inside a Boeing 747.* In AIAA-93-3091, 1993. 60, 141

[Baum 1995a] J.D. Baum, H. Luo and R. Löhner. *Numerical Simulation of a Blast in the World Trade Center.* In AIAA-95-0085, 1995. 60, 141

[Baum 1995b] J.D. Baum, H. Luo and R. Löhner. *Validation of a new ALE Adaptive, Unstructured Moving Body Methodology for Multi-Store Ejection Simulations.* In AIAA-95-1792, 1995. 141

[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. 141

[Baum 1997a] J.D. Baum, R. Löhner, T.J Marquette and H. Luo. *Numerical Simulation of Aircraft Canopy Trajectory.* In AIAA-97-0166, 1997. 60, 141

[Baum 1997b] J.D. Baum, H. Luo, R. Löhner, E. Goldberg and A. Feldhun. *Application of Unstructured Adaptive Moving Body Methodology to the Simulation of Fuel Tank Separation from a F16 C/D Fighter.* In AIAA-97-1885, 1997. 141

[Baum 2003] J.D. Baum, E. Mestreau, H. Luo, R. Löhner, D. Pelessone and C. Charman. *Modeling Structural Response to Blast Loading Using a Coupled CFD/CSD Methodology.* In Proceedings of the Design and Analysis of Protective Structures Against Impact/Impulsive/Shock Loads (DAPSIL) Conference, Tokyo, Japan, December 2003. 60

[Belhamadia 2004] Y. Belhamadia, A. Fortin and E. Chamberland. *Three-dimensional anisotropic mesh adaptation for phase change problems.* J. Comp. Phys., vol. 201, pages 753–770, 2004. 17

[Belme 2010] A. Belme, A. Dervieux and F. Alauzet. *Fully Anisotropic Goal-Oriented Mesh Adaptation for Unsteady Flows.* In ECCOMAS CFD Conference, Lisbon, Portugal, June 2010. 91, 230

[Belytschko 1978] T. Belytschko, J.M. Kennedy and D.F. Schoeberle. *Quasi-Eulerian Finite Element Formulation for Fluid-Structure Interaction.* In Proc. Joint ASME/CSME Pressure Vessels and Piping Conference ASME, 1978. 169

[Belytschko 2000] T. Belytschko, W.K. Liu and B. Moran. Nonlinear Finite Elements for Continua and Structures. John Wiley & Sons Ltd, 2000. 169

[Boffi 2004] D. Boffi and L. Gastaldi. *Stability and Geometric Conservation Laws For ALE Formulations.* Comput. Methods Appl. Mech. Engrg., vol. 193, no. 42-44, pages 4717–4739, 2004. 179

[Bottasso 2004] C.L. Bottasso. *Anisotropic mesh adaption by metric-driven optimization.* Int. J. Numer. Meth. Engng, vol. 60, pages 597–639, 2004. 7, 17

[Brackbill 1982] J.U. Brackbill and J.S. Saltzmann. *Adaptive Zoning for Singular Problems in Two Dimensions.* J. Comp. Phys., vol. 46, pages 342–368, 1982. 67

[Brackbill 1993] J.U. Brackbill. *An Adaptive Grid with Directional Control.* J. Comp. Phys., vol. 108, pages 38–50, 1993. 68

[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. 17, 144

[Buscaglia 1997] G.C. Buscaglia and E.A. Dari. *Anisotropic Mesh Optimization and its Application in Adaptivity.* Int. J. Numer. Meth. Engng, vol. 40, pages 4119–4136, 1997. 16

[Butcher 1987] J. C. Butcher. The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods. Wiley-Interscience, New York, NY, 1987. 181, 249

[Cao 2002] W. Cao, W. Huang and R.D. Russell. *A Moving Mesh Method based on the Geometric Conservation Law.* SIAM J. Sci. Comp., vol. 47, pages 118–142, 2002. 66

[Carlson 1998a] N. Carlson and K. Miller. *Design and Application of a Gradient-Weighted Moving Finite Element Code, Part I: in 1-D.* SIAM J. Sci. Comp., vol. 19, pages 728–765, 1998. 66

[Carlson 1998b] N. Carlson and K. Miller. *Design and Application of a Gradient-Weighted Moving Finite Element Code, Part II: in 2-D.* SIAM J. Sci. Comp., vol. 19, pages 766–798, 1998. 66

[Castro-Díaz 1997] M.J. Castro-Díaz, F. Hecht, B. Mohammadi and O. Pironneau. *Anisotropic Unstructured Mesh Adaptation for Flow Simulations.* Int. J. Numer. Meth. Fluids, vol. 25, pages 475–491, 1997. 13, 16, 40, 45

[Ceniceros 2001] H.D. Ceniceros and T.H. Hou. *An Efficient Dynamically Adaptive Mesh for Potentially Singular Solutions.* J. Comp. Phys., vol. 172, pages 609–639, 2001. 68

[Chacón 2006] L. Chacón and G. Lapenta. *A Fully Implicit, Non-Linear Adaptve Grid Srategy.* J. Comp. Phys., vol. 212, pages 703–717, 2006. 68

[Ciarlet 1978] P.G. Ciarlet. The finite element method for elliptic problems. North-Holland, Amsterdam, 1978. 41

[Clément 1975] Ph. Clément. *Approximation by finite element functions using local regularization.* Revue Française d'Automatique, Informatique et Recherche Opérationnelle, vol. R-2, pages 77–84, 1975. 31, 233

[Compère 2007] G. Compère, E. Marchandise and J.-F. Remacle. *Transient adaptivity applied to two-phase incompressible flows.* J. Comp. Phys., vol. 227, pages 1923–1942, 2007. 17, 60

[Compere 2010] G. Compere, 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, no. 7, pages 843–867, 2010. 17, 70, 141, 144, 157

[Coudière 2002] Y. Coudière, B. Palmerio, A. Dervieux and D. Leservoisier. *Accuracy barriers in mesh adaptation.* RR-4528, INRIA, aug 2002. 48, 49

[Coupez 2000] T. Coupez. *Génération de maillages et adaptation de maillage par optimisation locale.* Revue Européenne des Éléments Finis, vol. 9, pages 403–423, 2000. 17

[Cournède 2006a] 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. 179

[Cournède 2006b] P.H. Cournède, B. Koobus and A. Dervieux. *Positivity Statements for a Mixed-Element-Volume Scheme on Fixed and Moving Grids.* Europ. J. Comp. Mech., vol. 15, no. 7-8, pages 767–798, 2006. 78, 176

[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. 69

[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, no. 6, pages 89–118, 2000. 76, 78, 175

[Dervieux 2010] A. Dervieux, C. Farhat, B. Koobus and M. Vázquez. *Total energy conservation in ALE schemes for compressible flows.* Europ. J. Comp. Mech., vol. 19, no. 4, pages 337–363, 2010. 180

[Di 2007] Y. Di, R. Li and T. Tang. *A General Moving Mesh Framework in 3D and its Application for Simulating the Mixture of Multi-Phase Flows.* Communications in Computational Physics, vol. 3, no. 3, pages 582–602, 2007. 68

[Dindar 2000] M. Dindar, M.S. Shephard, J.E. Flaherty and K. Jansen. *Adaptive CFD analysis for rotorcraft aerodynamics.* Comput. Methods Appl. Mech. Engrg, vol. 189, pages 1055–1076, 2000. 141

[Dobrzynski 2008] C. Dobrzynski and P.J. Frey. *Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations.* In Proceedings of the 17th International Meshing Roundtable, pages 177–194. Springer, 2008. 17, 144, 157

[Dompierre 1997] J. Dompierre, M.G. Vallet, M. Fortin, Y. Bourgault and W.G. Habashi. *Anisotropic mesh adaptation: towards a solver and user independent CFD.* In AIAA 35th Aerospace Sciences Meeting and Exhibit, AIAA-1997-0861, Reno, NV, USA, Jan 1997. 16

[Donea 1977] J. Donea, S. Guiliani and J.P. Halleux. *An Arbitrary Lagrangian-Eulerian Finite Element Method for Transient Dynamic Fluid-Structure Interactions.* In Trans. 4th Int. Conf. on Structural Mechanics in Reactor Technology, Paper B1/2, California, USA, 1977. 169

[Dvinsky 1991] A.S. Dvinsky. *Adaptive Grid Generation from Harmonic Maps on Riemannian Manifolds.* J. Comp. Phys., vol. 95, pages 450–476, 1991. 68

[Esnault 1985] O. Esnault, M. Boileau, R. Vicqueli and al. *A 3D Chimera Grid Embedding Technique.* In AIAA Paper No. 85-1523, 7th AIAA Computational Fluid Dynamics Conference, 1985. 141

[Esnault 2011] O. Esnault, M. Boileau, R. Vicqueli and al. *A Method to Accelerate LES Explicit Solvers Using Local Time-Stepping.* In 48th AIAA Aerospace Sciences Meeting, Orlando, FL, June 2011. 63

[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. 179

[Farhat 1998] C. Farhat, C. Degand, B. Koobus and M.Lesoinne. *Torsional Spring for Two-Dimensional Dynamic Unstructured Fluid Meshes.* Comput. Methods Appl. Mech. Engrg., vol. 163, no. 1-4, pages 231–245, 1998. 145

[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. 179

[Farhat 2003] C. Farhat and P. Geuzaine. *Design and Analysis of Robust ALE Time-Integrators for the Solution of Unsteady Flow Problems on Moving Grids.* Comput. Methods Appl. Mech. Engrg., vol. 193, no. 39-41, pages 4073–4095, September 2003. 60

[Farrell 2009] P.E. Farrell, M. Piggott, C.C. Pain and G.J. Gorman. *Conservative interpolation between unstructured meshes via supermesh construction.* Comput. Methods Appl. Mech. Engrg., vol. 198, no. 33-36, pages 2632–2642, 2009. 30

[Ferracina 2005] L. Ferracina and M.N. Spijker. *Stepsize Restrictions for the Total-Variation-Diminishing Property in General Runge-Kutta Methods.* Appl. Numer. Math., vol. 53, no. 2, pages 26–279, 2005. 181

[Formaggia 1999] L. Formaggia and F. Nobile. *A Stability Analysis for the Arbitrary Lagrangian Eulerian Formulation with Finite Elements.* East-West J. Numer. Math., vol. 7, no. 2, pages 105–131, 1999. 179

[Formaggia 2001] L. Formaggia and S. Perotto. *New anisotropic a priori error estimates.* Numer. Math., vol. 89, pages 641–667, 2001. 17

[Formaggia 2004a] L. Formaggia, S. Micheletti and S. Perotto. *Anisotropic mesh adaptation in computational fluid dynamics: Application to the advection-diffusion-reaction and the Stokes problems.* Appl. Numer. Math., vol. 51, no. 4, pages 511–533, 2004. 17

[Formaggia 2004b] L. Formaggia and F. Nobile. *Stability Analysis of Second-Order Time Accurate Schemes for ALE-FEM.* Comput. Methods Appl. Mech. Engrg., vol. 193, no. 1, pages 4097–4116, 2004. 179

[Formaggia 2009] L. Formaggia, A. Quarteroni and A. Veneziani. Cardiovascular Mathematics - modelling and simulation of the circulatory system, volume 1. Springer - Modelling, Simulations and Applications, 2009. 141

[Fortin 1996] M. Fortin, M.-G. Vallet, J. Dompierre, Y. Bourgault and W.G. Habashi. *Anisotropic mesh adaptation : theory, validation and applications.* In Proceedings of ECCOMAS CFD, 1996. 16

[Franck 1964] R.M. Franck and R.B. Lazarus. *Mixed Eulerian-Lagrangian Methods.* Methods in Computational Physics, Volume 3, Fundamental Methods in Hydrodynamics, vol. 3, 1964. 169

[Frey 2001] P.J. Frey. `Yams`*, A fully automatic adaptive isotropic surface remeshing procedure.* RT-0252, INRIA, November 2001. 17

[Frey 2005] P.J. Frey and F. Alauzet. *Anisotropic mesh adaptation for* CFD *computations.* Comput. Methods Appl. Mech. Engrg., vol. 194, no. 48-49, pages 5068–5082, 2005. 13, 17, 40, 45

[Frey 2008] P.J. Frey and P.L. George. Mesh generation: Application to Finite Elements. ISTE Ltd and John Wiley & Sons, 2nd édition, 2008. 27, 150

[Gear 1971] C. William Gear. Numerical initial value problems in ordinary differential equations [by] c. william gear. Prentice-Hall Englewood Cliffs, N.J.„ 1971. 70

[George 1991] P.L. George, F. Hecht and M.G. Vallet. *Creation of internal points in Vornoi's type method. Control adaptation.* Adv. Eng. Software, vol. 13, no. 5-6, pages 303–312, 1991. 16, 25

[George 2003] P.L George. `Gamanic3d`, *an adaptive anisotropic tetrahedral mesh generator*. RT-0252, INRIA, November 2003. 17

[Godlewski 1996] E. Godlewski and P-A. Raviart. Numerical Approximation of Hyperbolic Systems of Conservation Laws. Springer-Verlag, New-York, 1996. 177

[Godunov 1967] S.K. Godunov and G.P. Prokopov. *On Computation of Conformal Transformations and Construction of Difference Meshes.* USSR Zh. Vychisl. Mat. Mat. Fiz., vol. 7, no. 209, pages 1031–1059, 1967. 67

[Gottlieb 1998] S. Gottlieb and C. Shu. *Total Variation Diminishing Runge-Kutta Schemes.* Math. of Comp., vol. 67, no. 221, pages 73–85, 1998. 181, 251

[Gottlieb 2001] S. Gottlieb, C. Shu and E. Tadmor. *Strong-Stability-Preserving High-Order Time Discretization Methods.* SIAM J. Numer. Anal., vol. 43, no. 1, pages 89–112, 2001. iii, 250

[Gruau 2005] C. Gruau and T. Coupez. *3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric.* Comput. Methods Appl. Mech. Engrg., vol. 194, no. 48-49, pages 4951–4976, 2005. 7, 17

[Guégan 2010] D. Guégan, O. Allain, A. Dervieux and F. Alauzet. *An L^∞-L^p mesh adaptive method for computing unsteady bi-fluid flows.* Int. J. Numer. Meth. Engng, vol. 84, no. 11, pages 1376–1406, 2010. 7, 17, 60, 70

[Guillard 2000] H. Guillard and C. Farhat. *On the Significance of the Geometric Conservation Law for Flow Computations on Moving Meshes.* Comput. Methods Appl. Mech. Engrg., vol. 190, no. 11-12, pages 1467–1482, 2000. 179, 211

[Harten 1983] A. Harten, P. Lax and B.V. Leer. *On Upstream Differencing and Godunov-type Schemes for Hyperbolic Conservation Laws.* SIAM Review, vol. 25, no. 1, pages 35–61, 1983. 75, 172

[Hassan 2007] O. Hassan, K.A. Sorensen, 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, pages 1243–1266, 2007. 141

[Hecht 1997] F. Hecht and B. Mohammadi. *Mesh adaptation by metric control for multi-scale phenomena and turbulence.* In 35th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-1997-0859, Reno, NV, USA, Jan 1997. 7, 16

[Hecht 1998] F. Hecht. `BAMG`*: bidimensional Anisotropic Mesh Generator.* Available from http://www-rocq.inria.fr/gamma/cdrom/www/bamg/eng.htm, INRIA-Rocquencourt, France, 1998. 17

[Hirsch 1988a] C. Hirsch. Numerical computation of internal and external flows - volume 1: Fundamentals of numerical discretization. Wiley Interscience Series in Numerical Methods in Engineering. John Wiley & Sons Ltd, Chichester, England, 1988. 78

[Hirsch 1988b] C. Hirsch. Numerical computation of internal and external flows - volume 2: Computational methods for inviscid and viscous flows. Wiley Interscience Series in Numerical Methods in Engineering. John Wiley & Sons Ltd, Chichester, England, 1988. 78, 177

[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, pages 227–253, 1974. reprinted in JCP vol. 135 (1997), pp. 203–2016. 169

[Huang 1999] W. Huang and R.D. Russell. *Moving Mesh Strategy based upon a Gradient Flow Equation for Two Dimensional Problems.* SIAM J. Sci. Comp., vol. 20, no. 2, pages 998–1115, 1999. 68

[Huang 2001] W. Huang. *Variational Mesh Adaptation: Isotropy and Equidistribution.* J. Comp. Phys., vol. 174, pages 903–924, 2001. 68

[Huang 2005] W. Huang. *Metric tensors for anisotropic mesh generation.* J. Comp. Phys., vol. 204, no. 2, pages 633–665, 2005. 17, 40

[Hugues 1978] T.J.R. Hugues, W.K. Liu and T.K. Zimmermann. *Lagrangian-Eulerian Finite Element Formulation for Incompressible Viscous Flows.* In U.S.-Japan Seminar on Interdisciplinary Finite Element Analysis, Cornell Univ, Ithaca, NY, 1978. 169

[Isola 2010] D. Isola, A. Guardone and G. Quaranta. *An ALE Scheme Without Interpolation for Moving Domain with Adaptive Grids.* In Proceedings of 40th the Fluid Dynamics Conference and Exhibit, volume AIAA 2010-4439, Chicago, IL, 28 June - 1 July 2010. 190

[Jones 2006] W.T. Jones, E.J. Nielsen and M.A. Park. *Validation of 3D Adjoint Based Error Estimation and Mesh Adaptation for Sonic Boom Reduction.* In 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2006-1150, Reno, NV, USA, Jan 2006. 17

[Kavetski 2002] Dmitri Kavetski, Philip Binning and Scott W. Sloan. *Adaptive Backward Euler Time Stepping with Truncation Error Control for Numerical Modeling of Unsaturated Fluid Flow.* Int. J. Numer. Meth. Engng, vol. 53, no. 6, pages 1301–1322, 2002. 70

[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. 180

[Korhonen 2008] T Korhonen and P Tavi. *Automatic Time-Step Adaptation of the Forward Euler Method in Simulation of Models of Ion Channels and Excitable Cells and Tissue.* Simulation Modeling Practice and Theory, vol. 16, no. 6, pages 639–644, 2008. 70

[Kraaijevanger 1991] J.F.B.M. Kraaijevanger. *Contractivity of Runge-Kutta Methods.* BIT Numer. Math.., vol. 31, no. 3, pages 482–528, 1991. 181

[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. 190

[Lambert 1991] J. D. Lambert. Numerical Methods for Ordinary Differential Systems: the Initial Value Problem. Wiley-Interscience, New York, NY, 1991. 181

[Langseth 2000] J. O. Langseth and R.J. LeVeque. *A Wave Propagation Method for Three-Dimensional Hyperbolic Conservation Laws.* J. Comp. Phys., vol. 165, no. 1, pages 126–166, November 2000. 60

[Laug 2003] P. Laug and H. Bourochaki. `BL2D-V2`, *Mailleur bidimensionnel adaptatif.* RT-0275, INRIA, 2003. 17

[Launay 2011] A. Launay, M. H. Maitournam, Y. Marco, I. Raoult and F. Szmytka. *Cyclic behaviour of short glass fiber reinforced polyamide: experimental study and constitutive equations.* International Journal of Plasticity, vol. (submitted to publication), Feb 2011. 237

[Lesage 2007] A.-C. Lesage, O. Allain and A. Dervieux. *On level set modelling of bi-fluid capillary flow.* Int. J. Numer. Meth. Fluids, vol. 53, no. 8, pages 1297–1314, 2007. 60

[Leservoisier 2001] D. Leservoisier, P.L. George and A. Dervieux. *Métrique continue et optimisation de maillage.* RR-4172, INRIA, April 2001. (in French). 13

[Lesoinne 1993] M. Lesoinne and C. Farhat. *Stability Analysis of Dynamic Meshes for Transient Aeroelastic Compu- tations.* In AIAA 93-3325, July 1993. 141

[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. 180

[Li 1999] H. Li and G. Ben-Dor. *Analysis of double-Mach-reflection wave configurations with convexly curved Mach stems*. Shock Waves, vol. 9, pages 319–326, 1999. 101

[Li 2005] X. Li, M.S. Shephard and M.W. Beal. *3D anisotropic mesh adaptation by mesh modification*. Comput. Methods Appl. Mech. Engrg., vol. 194, no. 48-49, pages 4915–4950, 2005. 17

[Löhner 1989] R. Löhner. *Adaptive Remeshing for Transient Problems*. Comput. Methods Appl. Mech. Engrg., vol. 75, pages 195–214, 1989. 16

[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. 16

[Löhner 1992] R. Löhner and J.D. Baum. *Adaptive H-Refinement on 3-D Unstructured Grids for Transient Problems*. Int. J. Numer. Meth. Fluids, vol. 14, no. 12, pages 1407–1419, 1992. 69

[Löhner 2001] R. Löhner. Applied CFD techniques. An introduction based on finite element methods. John Wiley & Sons, Ltd, New York, 2001. 27

[Löhner 2007] R. Löhner, S. Appanaboyina and J. Cebral. *Comparison of Body-Fitted, Embedded and Immersed 3D Euler Predictions for Blast Loads on Columns*. In 45th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 2007. 142

[Löhner 2008] R. Löhner. Applied Computational Fluid Dynamics Techniques: an Introduction based on Finite Elements Methods - 2nd Edition. John Wiley & Sons Ltd, Chichester, England, 2008. 142

[Loseille 2007] A. Loseille, A. Dervieux, P.J. Frey and F. Alauzet. *Achievement of global second-order mesh convergence for discontinuous flows with adapted unstructured meshes*. In 37th AIAA Fluid Dynamics Conference and Exhibit, AIAA-2007-4186, Miami, FL, USA, Jun 2007. 48

[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). 35

[Loseille 2009] A. Loseille and R. Löhner. *On 3D anisotropic local remeshing for surface, volume, and boundary layers*. In Proceedings of the 18th International Meshing Roundtable, pages 611–630. Springer, 2009. 17

[Loseille 2010a] A. Loseille and F. Alauzet. *Continuous mesh framework. Part I: well-posed continuous mesh and interpolation error models*. SIAM J. Numer. Anal., 2010. Accepted for publication. 8, 13, 37, 41, 83

[Loseille 2010b] A. Loseille and F. Alauzet. *Continuous mesh framework. Part II: validations and applications*. SIAM J. Numer. Anal., 2010. Accepted for publication. 8, 43, 47, 48, 49, 73, 74, 82, 88

[Loseille 2010c] A. Loseille and F. Alauzet. *Fully Anisotropic Goal-Oriented Mesh Adaptation for 3D Steady Euler Equations*. J. Comp. Phys., vol. 229, no. 8, pages 2866–2897, 2010. 7, 13, 17, 74, 230

[Loseille 2010d] A. Loseille and R. Löhner. *Adaptive Anisotropic Simulations in Aerodynamics*. In AIAA 2010-169, 48th AIAA Aerospace Sciences Meeting, Orlando, FL, USA, Jan 2010. 73

[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. 65

[Lozinski 2009] A. Lozinski, M. Picasso and V. Prachittham. *An Anisotropic Error Estimator for the Crank-Nicholson Method: Application to a Parabolic Problem.* Journal of Computational and Applied Mathematics, vol. 233, no. 4, pages 1139–1154, 2009. 69

[Lynch 1980] D.R. Lynch and K. O'Neill. *Elastic Grid Deformation for Moving Boundary Problems in Two Space Dimensions.* In Proceedings of the 3rd International Conference on Finite Elements in Water Resources - Volume 2, pages 7111–7120, Missippi University, Oxford, May 19-23 1980. 145

[Mani 2009] K. Mani and D.J. Mavriplis. *Spatially Non-uniform Time-Step Adaptation in Unsteady Flow Problems.* In 21st Century Challenges in Computational Engineering and Science. International symposium on the occasion of the 75th birthday of Antony Jameson, Princeton, NJ, November 2009. 70

[Mani 2010] K. Mani and D.J. Mavriplis. *Spatially Non-Uniform Time-Step Adaptation for Functional Outputs in Unsteady Flow Problems.* J. Comp. Phys., vol. 229, no. 2, pages 415–440, January 2010. 70

[Margolin 2004] L.G. Margolin and M. Shashkov. *Remapping, Recovery and Repair on a Staggered Grid.* Comput. Methods Appl. Mech. Engrg., vol. 193, pages 4139–4155, 2004. 189

[Mavriplis 1990] D.J. Mavriplis. *Adaptive mesh generation for viscous flows using Delaunay triangulation.* J. Comp. Phys., vol. 90, pages 271–291, 1990. 16

[Mavriplis 2006] D. 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. 180, 192

[Meriaux 2003] M. Meriaux and S. Piperno. *Adaptation Dynamique de Maillage pour les Lois de Conservation Hyperboliques en Une Dimension.* INRIA, Rocquencourt, France, 2003. Research Report RR-4696. 190

[Mesri 2008] Y. Mesri, W. Zerguine, H. Digonnet, L. Silva and T. Coupez. *Dynamic Parallel Adaption for Three Dimensional Unstructured Meshes: Application to Interface Tracking.* In Proceedings of the 17th International Meshing Roundtable, pages 195–212. Springer, 2008. 60

[Mestreau 1993] E. Mestreau, R. Löhner and S. Aita. *TGV Tunnel-Entry Simulation using a Finite Element Code with Automatic Remeshing.* In AIAA-93-0890, 1993. 141

[Micheletti 2008] S. Micheletti and S. Perotto. *Anisotropic Mesh Adaption for Time-Dependent Problems.* Int. J. Numer. Meth. Fluids, vol. 58, no. 9, pages 1009–1015, 2008. 69

[Miller 1981] K. Miller and R.N. Miller. *Moving Finite Elements I.* SIAM J. Numer. Anal., vol. 18, pages 1019–1032, 1981. 66

[Moser 1965] J. Moser. *On the Volume Elements of a Manifold.* Trans. AMS, vol. 120, pages 286–294, 1965. 66

[Murman 2003] S. Murman, M. Aftosmis and M. Berger. *Simulation of 6-DOF Motion with Cartesian Method.* In Proceedings of the 41th AIAA Conference, Reno, NV, 6-9 January 2003. 141, 144

[Nkonga 1993] B. Nkonga and H. Guillard. *Godunov Type Methods on Non-Structured Meshes for Three-Dimensional Moving Boundary Problems.* INRIA, Sophia Antipolis, France, 1993. Research Report RR-1893. 187

[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. 180

[Noh 1964] W.F. Noh. *CEL:A Time-Dependent Two-Space Dimensional Coupled Eulerian-Lagrangian Code.* Methods in Computational Physics, Volume 3, Fundamental Methods in Hydrodynamics, vol. 3, 1964. 169

[Oger 2006] G. Oger, M. Doring, B. Alessandrini and P. Ferrant. *Two-Dimensional SPH simulations of Wedge Water Entries.* J. Comp. Phys., vol. 213, no. 2, pages 803–822, April 2006. 153

[Olivier 2011] G. Olivier and F. Alauzet. *A New Changing-Topology ALE Scheme for Moving Mesh Unsteady Simulations.* In 49th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2011, Orlando, FL, USA, Jan 2011. 190

[Pain 2001] C.C Pain, A.P. Humpleby, C.R.E. de Oliveira and A.J.H. Goddard. *Tetrahedral mesh optimization and adaptivity for steady-state and transient finite element calculations.* Comput. Methods Appl. Mech. Engrg., vol. 190, pages 3771–3796, 2001. 17

[Peraire 1987] J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz. *Adaptive Remeshing for Compressible Flow Computations.* J. Comput. Phys., vol. 72, pages 449–466, 1987. 16

[Peraire 1992] J. Peraire, J. Peiro and K. Morgan. *Adaptive Remeshing for Three-Dimensional Compressible Flow Computations.* J. Comput. Phys., vol. 103, pages 269–285, 1992. 16

[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. 17, 69

[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. 69

[Pivkin 2005] I.V. Pivkin, E. Hueso, R. Weinstein and al. *Simulation and Visualization of Air Flow Around Bat Wings During Flight.* In In Proceedings of the International Conference on Computational Science, pages 689–694, Atlanta, USA, 2005. 141

[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. 69

[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. 17

[Richtmyer 1967] R.D. Richtmyer and K.W. Morton. Difference Methods for Initial-Value Problems. Interscience Publishers, New York, 1967. 78

[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. 141, 144

[Selmin 1992] V. Selmin and L. Formaggia. *Simulation of hypersonic flows on unstructured grids.* Int. J. Numer. Meth. Engng, vol. 34, pages 569–606, 1992. 16

[Shu 1988] C.W. Shu and S. Osher. *Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes.* J. Comp. Phys., vol. 77, no. 2, pages 439–471, 1988. 181, 182

[Shyam 2010] V. Shyam, A. Ameri, D. Luk and J-P. Chen. *Three-Dimensional Unsteady Simulation of a Modern High Pressure Turbine Stage Using Phase Lag Periodicity: Analysis of Flow and Heat Transfer*. Glenn Research Center, Cleveland, Ohio, February 2010. NASA/TM-2010-216064. 60, 141

[Spiteri 2002] R. Spiteri and S. Ruuth. *Two barriers on strong-stability-preserving time discretization methods*. SIAM J. Sci. Comput., vol. 17, no. 1-4, pages 211–220, 2002. 251

[Spiteri 2003] R. Spiteri and S. 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, 2003. doi:10.1023/A:1015156832269. 181, 254

[Stein 2003] K. Stein, T. Tezduyar and R. Benney. *Mesh Moving Techniques for Fluid-Structure Interactions with Large Displacements*. Journ. Appl. Mech., vol. 70, pages 58–63, 2003. 147, 148

[Stück 2009] A. Stück, F.F. Camelli and R. Löhner. *Adjoint-Based Design of Shock Mitigation Devices*. Int. J. Numer. Meth. Fluids, vol. 189, no. 1, pages 1055–1076, 2009. AIAA-09-3801. 60

[Tam 2000] A. Tam, D. Ait-Ali-Yahia, M.P. Robichaud, M. Moore, V. Kozel and W.G. Habashi. *Anisotropic mesh adaptation for 3D flows on structured and unstructured grids*. Comput. Methods Appl. Mech. Engrg., vol. 189, pages 1205–1230, 2000. 17

[Thomas 1979] P.D. Thomas and C.K. Lombard. *Geometric Conservation Law and its Application to Flow Computations on Moving Grids*. AIAA Journal, vol. 17, no. 10, pages 1030–1037, 1979. 179

[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. 67

[Tijdeman 1980] H. Tijdeman and R. Seebass. *Transonic Flows Past Oscillating Airfoils*. Comput. Methods Appl. Mech. Engrg., vol. 12, pages 181–222, 1980. 141

[Trulio 1961] J. Trulio and K. Trigger. *Numerical Solution of the One-Dimensional Hydrodynamic Equations in an Arbitrary Time-Dependent Coordinate System*. University of California Lawrence Radiation Laboratory, Berkeley, CA, 1961. Technical Report UCLR-6522. 141, 169

[Vallet 1992] M.-G. Vallet. *Génération de maillages éléments finis anisotropes et adaptatifs*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 1992. 13

[Van 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. 174

[Venditti 2003] D.A. Venditti and D.L. Darmofal. *Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows*. J. Comput. Phys., vol. 187, no. 1, pages 22–46, 2003. 17

[Venkatakrishnan 2007] V. Venkatakrishnan and D. Mavriplis. *Implicit Method for the Computation of Unsteady Flows on Unstructured Grids*. Int. J. Numer. Meth. Fluids, vol. 56, pages 1359–1365, 2007. 189

[Winslow 1963] A. Winslow. *Equipotential Zoning of Two Dimensional Meshes*. Lawrence Livemore Laboratory, California, USA, 1963. Technical Report UCRL-7312. 67

[Woodward 1984] P. Woodward and P. Collela. *The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks*. J. Comp. Phys., vol. 54, no. 1, pages 115–173, April 1984. 60, 100

[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. 69

[Xu 1998] L. Xu, A.W. Troesch and R. Peterson. *Assymetric Hydrodynamic Impact and Dynamic Response of Vessels*. In OMAE98-0382 Proceedings of the ASME 17th International Conference on Offshore Mechanics and Arctic Engineering, 1998. 153

[Yang 2005] Z. Yang and D. Mavriplis. *Unstructured Dynamic Meshes with Higher-Order Time Integration Schemes for the Unsteady Navier-Stokes Equations*. In Proceedings of the 41^th AIAA Conference, 10-13 January 2005. 151, 180, 181, 200, 204, 211

[Yang 2007] Z. Yang and D. Mavriplis. *Higher-Order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes*. AIAA Journal, vol. 45, no. 1, pages 138–150, 2007. 145, 180

[Zhu 1998] M. Zhu, T. Fusegi, A. Tabbal and al. *A Tetrahedral Finite-Element Method For Fast 3-D Numerical Modeling and Entire Simulation of Unsteady Turbulent Flow in a Mixed-Flow Pump*. In Proceedings of the FEDSM'98, ASME Fluids Engineering Division Summer Meeting FEDSM98-4879, Washington, DC, 21-25 June 1998. 141, 153

[Zienkiewicz 1994] O.C. Zienkiewicz and J. Wu. *Automatic directional refinement in adaptive analysis of compressible flows*. Int. J. Numer. Meth. Engng, vol. 37, pages 2189–2210, 1994. 16

List of notations

$E^s_{\mathbf{L}^\infty}(t)$ global spatial interpolation error in $\mathbf{L}^\infty$ norm.

$|H_{i,\mathbf{max}}|$ maximal (intersected) Hessian associated with adaptation sub-interval i.

$\mathcal{R}$ matrix of the eigenvectors.

$e^s(\mathbf{x}, t)$ local spatial interpolation error.

n_{ptfx} number of fixed-point iterations for the fixed-point algorithm .

n_s number of Runge-Kutta stages.

$(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$ canonical basis of $\mathbf{R}^n$.

$(\mathbf{e}_x, \mathbf{e}_y)$ canonical basis in 2D.

$(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ canonical basis in 3D.

B a moving rigid body.

CFL^{geom} geometric CFL number.

$D_{\mathbf{L}^\infty\mathbf{L}^\infty}$ $\mathbf{L}^\infty - \mathbf{L}^\infty$ metric global normalization constant.

E Young modulus for elasticity.

H^s_u spatial Hessian matrix of scalar function u.

$H^s_{i,k}$ spatial Hessian matrix of u at sampling time $t_{i,k}$.

$J_{(G\mathbf{a})}$ body moment of inertia w.r.t axis of direction $\mathbf{a}$ passing through G.

K simplicial mesh element.

N^{ptfx} prescribed spatial complexity for each fixed-point mesh.

N_e number of edges.

N_t number of elements.

N_v number of vertices.

N_{ite} number of solver iterations.

P_i mesh vertex.

Q_{target} maximal quality threshold.

R_h smooth reconstruction operator.

W Euler conservative state vector (dim $n + 2$).

$[0, T]$ simulation time frame.

$[t_i, t_{i+1}]$ adaptation sub-interval for the fixed-point algorithm .

Λ diagonal matrix of the eigenvalues of $\mathcal{M}$.

Ω_h computational domain.

Ω physical domain.

Π_c Clément interpolation operator.

$\Pi_{\mathbf{L}^2}$ $\mathbf{L}^2$ projection operator.

Π_h $\mathbf{P}^1$ Lagrangian interpolation operator.

Π an interpolation operator.

$\boldsymbol{\eta}$ non-normalized outward normal.

$\boldsymbol{\omega}$ body angular speed vector.

$\boldsymbol{\phi}$ mapping between domain Ω^k and Ω^{k+1} .

$\boldsymbol{\theta}$ body angular displacement vector.

$\boldsymbol{w}$ instantaneous mesh velocity.

$\cdot^T$ tensor transposition operator.

χ stiffening power for elasticity assembly.

δt time between two samples in the fixed-point algorithm .

$\mathcal{E}$ deformation tensor.

$\mathbf{f}_{ext}$ resultant of the external volume forces applied to a moving body.

$\frac{\mathrm{D}}{\mathrm{D}t}$ Lagrangian derivative (following particle trajectories) .

γ heat capacity ratio.

$\mathbf{g}$ gravity vector.

$\hat{K}$ reference element.

κ_i anisotropic quotient $\frac{h_i^n}{\prod_{j=1}^{n} h_j}$.

λ_t space-time metric time eigenvalue.

λ first Lamé coefficient.

$(\Omega, \mathbb{R}^n, \mathbf{M})$ Riemannian space.

$(\Omega, \mathbb{R}^n)$ affine Euclidian space.

$(\beta_j(P))_{j \in [\![1, n+1]\!]}$ barycentric coordinates .

$\mathbf{M}$ metric field.

$\mathbf{d}^{els}$ displacement field prescribed by elasticity resolution.

$\mathbf{d}$ mesh displacement field .

$\mathbf{e_t}$ space-time metric eigenvector corresponding to time direction.

$\mathbf{e}_{ij}$ mesh internal or boundary edge.

$\mathbf{n}$ normalized outward normal.

$\mathbf{u}$ fluid Eulerian velocity.

$\mathbf{x}_G$ body gravity center coordinates vector.

$\mathbf{x}$ coordinate vector.

$\mathcal{A}$ Jacobian tensor of the Euler fluxes.

$\mathcal{C}$ continuous mesh complexity.

$\mathcal{E}_{\mathcal{M}}$ unit ball of metric tensor $\mathcal{M}$.

$\mathcal{H}$ simplicial mesh.

$\mathcal{I}_n$ identity matrix of $\mathbb{R}^{n\times n}$.

$\mathcal{J}_G$ body matrix of inertia computed at point G.

$\mathcal{L}$ a spatial differential operator.

$\mathcal{M}^{st}$ space-time optimal metric.

$\mathcal{M}^{s}$ spatial part of the space-time metric.

$\mathcal{M}$ metric tensor.

$\mathcal{S}$ strain tensor.

$\mathrm{d}t^{els}$ time step between two elasticity resolution.

$\mathbf{M}_G\left(\mathbf{F}_{ext}\right)$ kinetic moment of external volume forces computed at G.

ξ mesh position field w.r.t a determined initial configuration.

μ two Lamé coefficient.

ν Poisson ratio for elasticity.

$\overline{W}$ mirror state vector (dim $n+2$) for weak slipping boundary conditions.

$\overline{\boldsymbol{\eta}}$ non-normalized inward normal.

$\overline{\mathbf{n}}$ normalized inward normal.

$\partial\Omega_h$ computational domain boundary.

$\partial\Omega$ physical domain boundary.

ϕ a scalar weight function.

ρ fluid volume mass or density.

τ solver time step.

$\mathbf{Ball}(P_i)$ ball of vertex P_i.

$\mathbf{Neigh}(K)$ set of neighboring elements of element K.

$\mathbf{Shell}(\mathbf{e})$ shell of edge $\mathbf{e}$.

θ norm of body angular displacement vector.

$\tilde{W}$ Roe average state vector (dim $n+2$).

ε internal energy of the fluid per unit mass.

μ fluid kinematic viscosity.

a wave speed norm.

c local sound speed or celerity in the fluid.

d continuous mesh density.

$e^{st}_{\mathcal{M}}(\mathbf{x},t)$ local total space-time interpolation error associated with continuous mesh $\mathbf{M}$.

e fluid total energy per unit mass.

h_{max} maximal prescribed mesh size.

h fluid enthalpy per unit mass.

h typical mesh size.

$m_{t,i,thres.}$ temporal eigenvalue threshold for adaptation sub-interval i.

m_{thres} metric minimal threshold for the steady truncation algorithm.

m body mass (constant).

n_k number of samples in an adaptation sub-interval for the fixed-point algorithm .

n_{adap} number of adaptation sub-intervals for the fixed-point algorithm .

n_{ball} number of elements in the ball of a vertex.

n spatial dimension.

p fluid pressure.

q fluid velocity norm $q = ||\mathbf{u}||$.

$t_{i,k}$ time of sample number k in adaptation sub-interval i.

u^{nor} fluid normal velocity $u^{nor} = \mathbf{u} \cdot \mathbf{n}$.

u scalar solution or sensor function.

$\mathcal{S}_{i,k}$ solution sample at $t_{i,k}$.

$|C_i|$ volume of the dual cell associated with P_i.

Acronyms

AGARD Advisory Group for Aerospace Research and Development.

ALE Arbitrary-Lagrangian-Eulerian.

BDF Backward Differentiation Formula.

CAD Computer-Aided Design.

CFD Computational Fluid Dynamics.

CFL Courant-Friedrichs-Lewy.

CPU Central Processing Unit.

DG Discontinuous Galerkin.

DGCL Discrete Geometric Conservation Law.

DOF Degrees of Motion.

FD Finite Difference.

FE Finite Element.

FSI Fluid-Structure Interaction.

FV Finite Volume.

GAMMA Génération Automatique de Maillage et Méthodes d'Adaptation.

GCL Geometric Conservation Law.

GMRES Generalized Minimal RESidual method.

HLLC Harten-Lax-van Leer Contact wave.

INRIA Institut National de Recherche en Informatique et Automatique.

MMPDE Moving Mesh Partial Differential Equation.

MUSCL Monotone Upstream-Centered Schemes for Conservation Laws.

NACA National Advisory Committee for Aeronautics.

ODE Ordinary Differential Equation.

PDE Partial Differential Equation.

RAM Random Access Memory.

RK Runge-Kutta.

RKSSP Runge-Kutta Strong-Stability-Preserving.

SPD Symmetric Positive Definite.

SSP Strong-Stability-Preserving.

TVD Total Variation Diminishing.

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

Abstract: This thesis deals with time-evolving simulations involving fixed or moving geometries. Growing expectations of industrials regarding this kind of simulations are currently observed, and most of them would like such computations to be performed in their research centers on a daily basis, which is clearly not the case at the moment. This works attempts to partly fulfill this demand, and notably intends to improve the accuracy of these simulations as well as their efficiency in terms of CPU time.
Anisotropic metric-based mesh adaptation strategies, which have now reached a certain level of maturity on steady problems, offers good perspectives to enhance time-evolving simulations, but their extension in this context is far from straightforward. As for their application to moving mesh simulations, only few attempts can be listed so far and only a minority address complex three-dimensional real-life problems. This study proposes several novelties on these questions, notably the extension of multi-scale anisotropic metric based mesh adaptation to unsteady problems, for both fixed and moving domains. Besides, mainly for CPU reduction purpose, a genuine strategy has been adopted to handle moving mesh simulations. It is notably demonstrated in practice that it is possible to move three dimensional complex objects undergoing large displacements using only connectivity changes and vertex movements, which comes to keep the number of vertices of the moving mesh constant throughout the simulation. Limiting the number of mesh operations allowed enable to considerably reduce CPU time as time is saved both on the meshing and on the solver parts. Finally, a new scheme extending the classical fixed-topology Arbitrary-Lagrangian-Eulerian framework to variable -topology moving meshes is proposed and its validity has been assessed on two dimensional test cases. All these methods have been applied to Computational Fluid Dynamics simulations governed by the Euler compressible fluid equations around complex geometries in two and three dimensions.

Keywords: Unsteady CFD simulations, moving mesh, ALE, metric-based mesh adaptation, anisotropy, variable topology