3D anisotropic mesh adaptation for Reynolds Averaged Navier-Stokes simulations.

Loic Frazza

Sorbonne Université

École doctorale de Sciences Mathématiques de Paris Centre - ED 386

3D anisotropic mesh adaptation for Reynolds Averaged Navier-Stokes simulations.

Par Loïc Frazza

Thèse de doctorat de Mathématiques Appliquées

Dirigée par Frédéric Alauzet et Adrien Loseille

Présentée et soutenue le 18 Décembre 2018

devant le jury composé de:

Rapporteurs :	Professeur	David Darmofal	-	MIT
	Professeur	Nicolas Gourdain	-	ISAE
Directeurs de thèse :	DR	Frédéric Alauzet	-	INRIA Saclay
	CR	Adrien Loseille	-	INRIA Saclay
Examinateurs :	Professeur	Dominique Pelletier	-	Polytechnique Montréal
	HDR	Frédéric Feyel	-	SAFRAN
	Professeur	Jean-Camille Chassaing	-	Sorbonne Université

Contents

Résumé

La simulation rapide et fiable d'écoulements turbulents à l'aide de modèles RANS (Reynolds Averaged Navier Stokes) est un enjeu financier de premier plan pour de nombreuses industries. Paradoxalement, bien qu'elles soient devenues la norme aujourd'hui, la précision de ces simulations dépend fortement de la qualité du maillage utilisé. Face à l'accroissement de la complexité des géométries et des écoulements simulés, ainsi que des exigences en termes de fidélité, la génération de maillages appropriés est devenue un maillon clef de la chaine de calcul. Pourtant celle-ci repose toujours sur l'experience subjective d'un ingénieur qualifié. Au-delà de l'obstacle que cela représente pour l'automatisation de la chaine de calcul en question, il est impossible de déterminer des règles claires et universelles pour la génération d'un maillage. Ceci est d'autant plus vrai lorsqu'on est confronté à un écoulement complex.

Nous montrons dans cette thèse la capacité des schémas numériques modernes à simuler des écoulements turbulents sur des maillages totalement non-structurés générés automatiquement à l'aide de méthodes adaptatives. Nous détaillons le développement de différentes versions du modèle de Spalart-Allmaras ainsi que les choix numériques garantissant une robustesse suffisante du solver pour ne pas nécessiter de couche limite structurée.

Nous introduisons en suite l'analyse d'erreur nécessaire pour proposer different estimateurs d'erreur à la base de l'optimisation de maillage. Cette méthodologie est testée sur différents cas tests d'aérodynamique externe et de turbomachines et comparée aux méthodes traditionnelles de géneration de maillage. Nous montrons ainsi la capacité des méthodes d'adaptation de maillage à générer automatiquement des maillages adaptés optimaux pour les simulations RANS autour de géométries réalistes et complexes.

Introduction

Turbulent flows are commonly encountered in numerous industrial or research activities such as health care, environment, oil industry, transports industries... The understanding and prediction of these phenomena is often critical: the turbulent flow in arteries [Vétel 2009] or in a cardiac valve [Miron 2014], the prediction of pollutant dispersion [Sini 1996, Gousseau 2011] and generally the efficiency of several systems and processes [Liu 2008, Norton 2006]. For the particular case of aviation industry, turbulence prediction has a high impact on different aspects of the flight, such as the noise generated near the ground which is subject to always stronger regulation [Bodony 2005, Souliez 2002, Dobrzynski 2010], the maximal lift before stall or the cruise drag. These have a direct financial impact: the maximal lift before stall determines the wing area which directly impacts the cruise viscous drag. The cruise drag determines the autonomie range of the plane and thus its achievable routes and its financial yield. In turn, for a given range of flight, following the approximate Breguet range law, an increase of 1% of cruise drag leads to a loss of about 7.6% of the payload [Vassberg 2002], *e.g.* revenu.

Flight engineers are thus eager for accurate and automatic processes to minimize cruise drag [Slotnick 2014]. In this context, CFD (Computational Fluid Dynamics) has risen strong interest as it makes it possible to use various analytical optimization processes and to explore a large variety of designs [Ding 2018, Reuther 1999, Sun 2017]. Reliable numerical simulation capabilities also reduce the need for ground-based and in-flight tests that are extremely expensive to lead. For some applications, like military applications, catastrophic events simulation or some specific flight phases impossible to reproduce in wind tunnels, it is even the only source of datas [Kraft 2010]. Moreover, numerical simulations give an accessible physical insight in the flows, that may be difficult to obtain with experiment: flows in enclosed spaces like in turbomachinery applications are particularly difficult to measure. With the exponential growth of computational power, the CFD prediction capabilities have rapidly increased in the last decades, leading to a large spread of numerical simulations in various industries. From the 1970's to the 1990's, the CFD models have evolved from panel methods to linearized and non-linear potential flow, inviscid flows and Reynolds Averages Navier-Stokes (RANS) models. In particular, RANS simulations are now commonly used to simulate and predict turbulent flows and considering the prohibitive costs involved by Large Eddy Simulations (LES), they are expected to remain so for the decades to come [Larsson 2014].

Despite their success and the maturity of the technology, RANS simulations are still too expensive for some applications. Indeed, RANS computations exhibit a strong grid sensitivity that requires a proper mesh convergence [Chitale 2014, Mavriplis 2009]. In a design context, numerical uncertainties (model related, mesh related, ...) are accounted for with safe margin. These margin diminishes the efficiency of the design which again leads to less payload and revenu. Hence, computations reliability is a critical requirement and as long as modeling errors can only be reduced by changing the numerical model used, it is essential to minimize and to be able to assess the errors related to the discretization. This is generally addressed with a mesh convergence study, computing solutions on grids ranging from 10 to 200 million nodes. These large computations require large computational ressources and considering the fact that in 2018

a CPU hour of computation is priced about 0.05 euros, the costs of a single simulations rapidly grows when 10 000 cores are used. It is thus critical to achieve early mesh convergence.

The only way of achieving such goal is to carefully design the grid sequence used for these computations. Yet the mesh generation is already a difficult problem itself, and even considered to be the main obstacle in the simulation chain for some applications [Seo 2010]. Meshes are generally generated after guidelines or the intuition of a skilled engineer [Chan 2017, Vassberg 2003]. As mesh generation tools lack from robustness, the mesh generation step requires a strong implication of the user. It is thus difficult to automatize and ends up being the main bottle neck in modern computations. This lack of automation and robustness is also the main obstacle to the reliable use of mesh adaptation in extensive design optimization processes [Ranut 2014, Kim 2006, Skinner 2017]. Figure 1 shows the evolution of the geometries used in the AIAA high-lift prediction workshops from 2010 to 2018. With the increase of available computational power, all domains have followed the same complexification trend. It is now extremely complex, let alone to verify the appropriate mesh characteristics on every features of the geometry, not to mention the underlying flow field. This is why mesh generation requires to be automated.

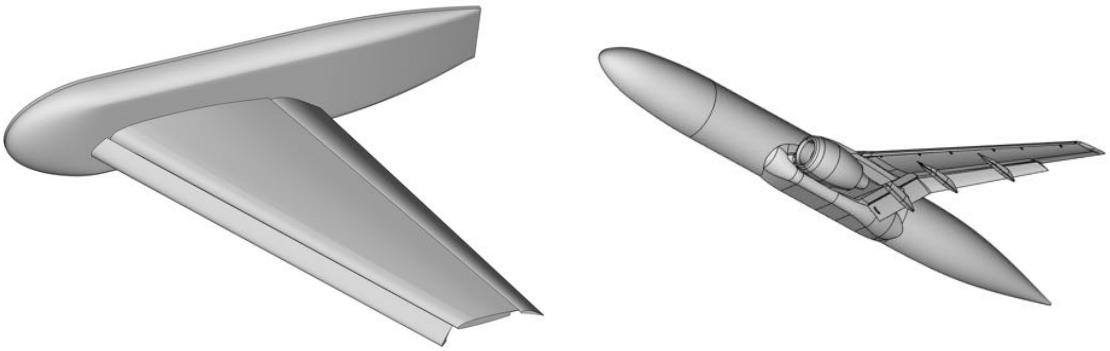


Figure 1: Trap-Wing geometry: 2010 (left) and Jaxa JSM geometry: 2018 (right).

Meanwhile, anisotropic mesh adaptation has proven to be very efficient to automatically generate optimal meshes for inviscid applications [Loseille 2010, Alauzet 2009a, Loseille 2007a, Venditti 2002b, Bourgault 2009]. The idea of mesh adaptation [Castro-Díaz 1997, Dervieux 2003, Frey 2005, Gruau 2005, Li 2003] is to use a previous computation to automatically determine the modifications to the mesh needed to optimize the overall solution or a specific output. This is done through sensors that analyzes the curent solution and prescribe modifications to the current mesh in order to balance and optimize the elements (and thus computational ressources) in the domain [Fidkowski 2011, Rogé 2008, Loseille 2010, Venditti 2000, Chen 2007, Bank 1993, Hecht 2008, Jones 2006]. Zones exhibiting steep variations of the solution or of importance for the computation of a specific output are automatically refined in order to increase the local precision of the discretization. The optimality of this approach relies on a rigorous mathematical analysis of the numerical error.

This approach has given very satisfying results on inviscid flows where fully unstructured meshes can be used, and thanks to the low expense of Euler simulations. It automatically

provides meshes adapted to the features of the flow. But, mesh adaptation has failed for years to be extended to turbulent applications due to different issues [Park 2016].

- Traditional flow solvers require a structured mesh in the boundary layers in order not to crash or simply to provide results of reasonable accuracy [Sahni 2009]. But it is a difficult task to automatically adapt the surface of the considered body and grow an appropriate structured boundary layer, coherent with the adapted volume mesh. Figure 2 illustrates this miss-match between the structured boundary layer and the adapted volume on a supersonic 3D shuttle.

- Before its generation, the very prescription of the mesh characteristics is also an open question. Although clear and efficient sensors have been developed for inviscid flows, these are inappropriate for viscous and RANS applications. A few direct extensions of the inviscid DWR estimate to RANS equations have been successfully applied to academic cases in 2D [Venditti 2003, Yano 2012b, Hu 2016, Darmofal 2013, Yano 2011] and in 3D with Adaptive Mesh Refinement (AMR) [Hartman 2010], although their optimization is based on iterative methods [Fidkowski 2011, Yano 2012b, Yano 2012a].

- Most mesh adaptation/generation tools lack robustness in the presence of complex geometries. This is an impediment for iterative mesh adaptation processes which requires the automatic generation of successive grids with no intervention of the user.

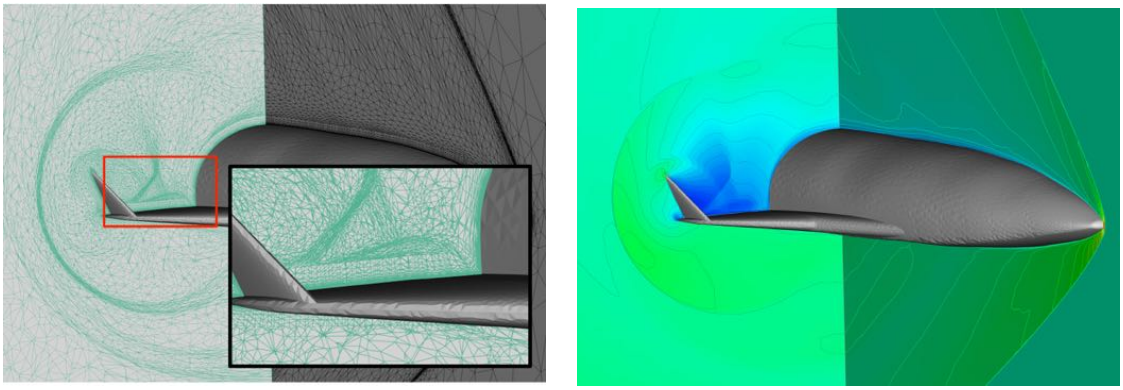


Figure 2: Illustration of a 3D shock impinging on a boundary layer. Structured boundary layer prevents the proper discretization of the shock.

Objectives

The goal of this PhD is to overcome the aforementioned issues in order to extend anisotropic mesh adaptation to RANS applications. To this end, the objectives of this PhD was to finish and validate the error analysis for viscous flows initiated in [Belme 2011]. This sensor analysis would be then used to perform mesh adaptation on RANS applications and be compared to feature-based mesh adaptations initiated in [Menier 2015]. From a technical point of view, the strategy initially foreseen was to develop a process to adapt separately the surface of the

body considered and then generate a structured boundary layer using the approach proposed in [Alauzet 2015, Alauzet 2017], before adapting the volume mesh.

Contributions

Solver robustness: At the begining of this PhD, a pre-existing implicit RANS flow solver had already been implemented but suffered from a lack of robustness, especially compared to the inviscid solver. Computations were limited to low CFL despite the implicit resolution and as many solvers, a structured discretization of the boundary layer was necessary with a specific treatment. A crucial part of this PhD has been dedicated to various developments in the flow solver, described in the first part of this thesis, in order to allow fast and faultless computations on adapted grids. We show in this thesis that contrary to common belief, finite volume flow solvers are able to simulate accurately RANS equations on fully unstructured anisotropic grids, as long as their development has been properly carried out.

Modelization: For the purpose of industrial contracts with Boeing Aircrafts and Safran Tech, we implemented additional versions of the Spalart-Allmaras turbulence model and performed the necessary implementations to deal with turbomachinery applications.

Solution correction: The assessment and control of discretization errors in simulations is essential, hence we introduced a non-linear solution correction context and implement a finite volume version in Chapter 4. This corrector has been tested on different applied cases and shown satisfactory results.

Error estimation and mesh adaptation: We corrected and finished the analysis introduced in [Belme 2011]. This methodology is extended to the general non-linear case in Chapter 6. From this analysis we derived different additional sensors further improving the predictions.

We then tested these sensors on various applied cases in 2D and 3D. In this thesis we demonstrate the viability and performances of unstructured anisotropic mesh adaptation for RANS simulations, comparing the predictions to standard mesh generation strategies on various cases.

Turbomachinery applications: We implemented a mesh adaptation strategy for periodic cases and demonstrated its viability and benefits on 2D and 3D applications, in Chapter 7.

Scientific communications

Book chapters

- **Numerical uncertainties estimation and mitigation by mesh adaptation**, F. Alauzet, A. Dervieux, L. Frazza and A. Loseille, *UMRIDA Book*, Part I, Chapter II.1.4, Edited by C. Hirsch.

Proceedings with peer review

- **Non-linear corrector for RANS equations**, L. Frazza, A. Loseille, F. Alauzet and A. Dervieux , 2018 Fluid Dynamics Conference, 2018.

- **Mesh adaptation strategies using wall functions and low-Reynolds models**, L. Frazza, A. Loseille and F. Alauzet , 2018 Fluid Dynamics Conference, 2018.
- **Comparing Anisotropic Error Estimates for the Onera M6 Wing RANS Simulations**, T. R. Michal, F. Alauzet, A. Loseille, L. Frazza, D. L. Marcum and D. S. Kamenetskiy , 55th AIAA Aerospace Sciences Meeting, 2018.
- **Comparing anisotropic adaptive strategies on the 2nd AIAA sonic boom workshop geometry**, A. Loseille, L. Frazza and F. Alauzet , 55th AIAA Aerospace Sciences Meeting, 2017.
- **Anisotropic mesh adaptation for turbomachinery applications**, L. Frazza, A. Loseille and F. Alauzet , 23rd AIAA Computational Fluid Dynamics Conference, 2017.

Communications

- **Adjoint based anisotropic mesh adaptation for turbomachinery applications**, L. Frazza, A. Loseille and F. Alauzet , Eccomas Conference, Glasgow, Scotland, 2018.
- **Anisotropic norm-oriented mesh adaptation for compressible Navier-Stokes equations**, L. Frazza, A. Loseille and F. Alauzet and A. Dervieux , Eccomas Conference, Crete, Grece, 2016.

Part I

RANS solver implementation for turbulent applications.

Introduction

Since the early days of numerical computation, many numerical schemes and approaches have been proposed, continuously improving the precision and efficiency of solvers. Meanwhile, modelization followed the same trend, evolving from panel methods in the 1970's to potential flows, inviscid flows and RANS models in the 1990's. Since, RANS computations have become in the industry the baseline for design and verification and are expected to remain the norm for decades, despite the emergence of LES and DNS methods [Slotnick 2014]. During this process, finite volume methods has become the most popular discretization for flow solvers for different reasons. Though, with high order needs, we see today a regain in popularity in finite elements methods (continuous and discontinuous Galerkin).

Indeed, finite element discretization was considered for a long time to be inappropriate for convection dominated problems, so that it is now considered a common knowledge (belief) [Chitale 2014, Pirzadeh 1996, Park 2010, Frazza 2017, Sahni 2009]. Similarly, it is a common believe that unstructured grids are improper for the computation of High Reynolds boundary layers. Many justifications have been proposed for this, like the inaccurate computation of gradients. In the facts, structured properties in the mesh significantly eases the math involved in the development of a numerical scheme, providing privileged directions in the datas to perform finite difference gradient reconstruction. Thus, such assumptions are the base of numerous schemes. These in turn perform nominally on structured grids and rather well on unstructured isotropic grids, but degenerate rapidly in presence of anisotropy without structured orthogonal elements. Hence, most of the standard flow solvers either crash or yield poor results when boundary layers are not discretized with a structured mesh [Sahni 2009]. This is generally due to an accumulation of inaccuracies in the solver, both in the residual computation and in the resolution. For example, an implicit solver with too large approximations in the linearization will be prone to divergence and be unstable.

The situation can be summed up by noting that there is somehow a constant balance between the work required in the solver and the mesh generation. Easing the development of the flow solver displaces the work requirement on the mesh generation by putting strong constraints on the quality of the mesh required. Such constraints have been one of the major limitation preventing the spread of mesh adaptation methods as it is nearly impossible to generate meshes that follows both the requirement of the flow and the solver. We can cite as a common example the treatment of the mesh near a cut trailing edge of a wing [Aubry 2009]. In this first part, we show that modern strong solvers are able to compute RANS solutions on fully unstructured grids. These solvers have in common the control and minimization of unnecessary approximations in the resolution.

The flow solver `Wolf` was initially developed as a finite-volume inviscid compressible Euler flow solver. It was then a tool to assess the capability of mesh adaptation strategies to produce reliable meshes for CFD computations. Its whole development has thus been driven by mesh adaptation. It has to be able to run several time without crashing and without any external intervention on unstructured grids with some isolated bad elements. `Wolf` is now a mixed finite

volume/finite element, vertex centered flow solver that handles unstationary and stationary flows on fixed grids and ALE simulations, inviscid and viscous laminar or RANS flows. In the scope of being able to perform adjoint-based mesh adaptation, the computation of the adjoint system for inviscid flows has been also implemented.

In this first part, we present the implementation of `Wolf` and the different parameters that will be used in our computations. We develop the various implementations that have been made to deal with the various turbulent flows simulations presented in this thesis. In particular, we introduce the choices and modifications that have been made to make it possible to perform RANS computations with `Wolf` on fully unstructured meshes. We also present the implementation of the turbulent adjoint and consideration of periodic flows to deal with turbomachinery applications.

Chapter 1

Main solver description

In this Chapter we develop the base implementation of our flow solver `Wolf` and the various choices that have been made to provide an efficient and robust resolution of the equations on fully unstructured grids. We detail the primal and adjoint solver that are used throughout this thesis and their validation. We also introduce the various implementations that have been made during this thesis to deal with various applications such as turbomachinery or low-Mach applications.

1.1 Main flow solver description

We solve the RANS equations using the Spalart-Allmaras one-equation model, which is a standard and well-documented option ($k-\omega-SST$ model is in development but not tested yet). The spatial discretization of the governing equations is based on a vertex-centered finite element-finite volume formulation, where the finite volume cells are built on unstructured meshes. Second-order space accuracy is achieved through a piecewise-linear extrapolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure with a particular edge-based formulation. Both explicit and implicit approaches are available for the time integration. During implicit simulations, a linearized system is solved at each solver iteration using an approach derived from the Lower-Upper Symmetric Gauss-Seidel (LU-SGS).

1.1.1 Modeling equations

The Reynolds Averaged Navier-Stokes (RANS) system relying on the Spalart-Allmaras model is composed of the compressible Navier-Stokes equations and the standard Spalart-Allmaras equation with no trip.

The compressible Navier-Stokes RANS equations

The compressible Navier-Stokes equations for mass, momentum and energy conservation read:

$$\left\{\begin{array}{rcl} \dfrac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) & = & 0\,, \\ \dfrac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p & = & \nabla \cdot \mathcal{T}\,, \\ \dfrac{\partial (\rho E)}{\partial t} + \nabla \cdot ((\rho E + p)\mathbf{u}) & = & \nabla \cdot (\mathcal{T} \cdot \mathbf{u}) + \nabla \cdot (\lambda \nabla T)\,, \end{array}\right. \tag{1.1}$$

where ρ denotes the density (kg/m^3), $\mathbf{u}$ the velocity (m/s), E the total energy per mass ($m^2.s^{-2}$), p the pressure (N/m^2) defiend as

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho||u||^2\right),$$

T the temperature (K) defined as

$$T = \frac{1}{c_v}\left(E - \frac{1}{2}\|u\|^2)\right),$$

λ the laminar conductivity and $\mathcal{T}$ the viscous stress tensor. Assuming the fluid to be newtonian and following the Boussinesq hypothesis $\mathcal{T}$, is defined from μ the laminar dynamic viscosity ($kg/(m.s)$) and μ_t the turbulent viscosity as:

$$\mathcal{T} = (\mu + \mu_t)\left[(\nabla \otimes \mathbf{u} + {}^t\nabla \otimes \mathbf{u}) - \frac{2}{3}\nabla.\,\mathbf{u}\,\mathbb{I}\right],$$

where (in 3D) $\mathbf{u} = (u, v, w)$ and

$$\nabla.\,\mathbf{u}\,\mathbb{I} = \begin{pmatrix} u_x + v_y + w_z & 0 & 0 \\ 0 & u_x + v_y + w_z & 0 \\ 0 & 0 & u_x + v_y + w_z \end{pmatrix},$$

where $u_x = \frac{\partial u}{\partial x}$, $u_y = \frac{\partial u}{\partial y}$, $u_z = \frac{\partial u}{\partial z}$ (idem for v and w).

The variation of nondimensionalized laminar dynamic viscosity and conductivity coefficients μ and λ as a function of the dimensional temperature T is defined by Sutherland's law:

$$\mu = \mu_\infty \left(\frac{T}{T_\infty}\right)^{\frac{3}{2}} \left(\frac{T_\infty + \mathrm{Su}}{T + \mathrm{Su}}\right) \quad \text{and} \quad \lambda = \lambda_\infty \left(\frac{T}{T_\infty}\right)^{\frac{3}{2}} \left(\frac{T_\infty + \mathrm{Su}}{T + \mathrm{Su}}\right),$$

where $\mathrm{Su} = 110$ is the Sutherland constant and the index ∞ denotes reference quantities. The relation linking μ and λ is expressed from the Prandtl laminar number:

$$\mathrm{Pr} = \frac{\mu c_p}{\lambda} \quad \text{with} \quad \mathrm{Pr} = 0.72 \quad \text{for (dry) air},$$

where C_p is the specific heat at constant pressure.

Turbulence modeling

The modelization of turbulence is described in more details in Section 2.1. We use the Spalart-Allmaras [Spalart 1992] or Menter $k - \omega - SST$ [Menter 1993] RANS turbulence models to compute an eddy viscosity μ_t and eddy thermal conductivity λ_t that are added to the laminar ones. These relie on the Boussinesq approximation to model the turbulent stress tensor as proportional to the strain rate tensor. It can be eventually amended in the case of the $SA{-}QCR$ (see Section 2.1.2) with a quadratic constitutive relation to express the turbulent tensor with respect to the strain rate and vorticity tensors.

Vector form of the RANS system

We write the RANS system for the Spalart-Allmaras model in the following (more compact) vector form:

$$W_t + F_1(W)_x + F_2(W)_y + F_3(W)_z = S_1(W)_x + S_2(W)_y + S_3(W)_z + Q(W)\,,$$

where $S_i(W)_a = \frac{\partial S_i(W)}{\partial a}$ $(i = 1,2,3, a = x,y,z)$ (idem for F). W is the nondimensionalized conservative variables vector:

$$W = (\rho, \rho u, \rho v, \rho w, \rho E, \rho\tilde{\nu})^T\,,$$

with $\tilde{\nu}$ the turbulent variable. $F(W) = (F_1(W), F_2(W), F_3(W))$ are the convective (Euler) flux functions:

$$\begin{aligned} F_1(W) &= \left(\rho u,\ \rho u^2 + p,\ \rho uv,\ \rho uw,\ u(\rho E + p), \rho u\tilde{\nu}\right)^T\,, \\ F_2(W) &= \left(\rho v,\ \rho uv,\ \rho v^2 + p,\ \rho vw,\ v(\rho E + p), \rho v\tilde{\nu}\right)^T\,, \\ F_3(W) &= \left(\rho w,\ \rho uw,\ \rho vw,\ \rho w^2 + p,\ w(\rho E + p), \rho w\tilde{\nu}\right)^T\,. \end{aligned} \tag{1.2}$$

$S(W) = (S_1(W), S_2(W), S_3(W))$ are the laminar viscous fluxes:

$$\begin{aligned} S_1(W) &= \left(0,\ \mathcal{T}_{xx},\ \mathcal{T}_{xy},\ \mathcal{T}_{xz},\ u\mathcal{T}_{xx} + v\mathcal{T}_{xy} + w\mathcal{T}_{xz} + \lambda T_x, \frac{\rho}{\sigma}(\nu + \tilde{\nu})\tilde{\nu}_x\right)^T\,, \\ S_2(W) &= \left(0,\ \mathcal{T}_{xy},\ \mathcal{T}_{yy},\ \mathcal{T}_{yz},\ u\mathcal{T}_{xy} + v\mathcal{T}_{yy} + w\mathcal{T}_{yz} + \lambda T_y, \frac{\rho}{\sigma}(\nu + \tilde{\nu})\tilde{\nu}_y\right)^T\,, \\ S_3(W) &= \left(0,\ \mathcal{T}_{xz},\ \mathcal{T}_{yz},\ \mathcal{T}_{zz},\ u\mathcal{T}_{xz} + v\mathcal{T}_{yz} + w\mathcal{T}_{zz} + \lambda T_z, \frac{\rho}{\sigma}(\nu + \tilde{\nu})\tilde{\nu}_z\right)^T\,, \end{aligned} \tag{1.3}$$

where $\mathcal{T}_{ij}$ are the components of the laminar and turbulent stress tensors $\mathcal{T}$:

$$\mathcal{T} = (\mu + \mu_t)\left[(\nabla \otimes \mathbf{u} + {}^t\nabla \otimes \mathbf{u}) - \frac{2}{3}\nabla.\,\mathbf{u}\,\mathbb{I}\right]$$

with

$$\mathcal{T}_{xx} = (\mu + \mu_t)\frac{2}{3}\left(2u_x - v_y - w_z\right), \quad \mathcal{T}_{xy} = (\mu + \mu_t)\left(u_y + v_x\right), \quad \mathcal{T}_{xz} = (\mu + \mu_t)\left(u_z + w_x\right), \quad ...$$

$Q(W)$ are the source terms, i.e. the diffusion, production and destruction terms from the Spalart-Allmaras turbulence model:

$$Q(W) = (0,\ 0,\ 0,\ 0,\ 0, \frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}\|^2 + \rho c_{b1}\tilde{S}\tilde{\nu} + c_{w1}f_w\rho\left(\frac{\tilde{\nu}}{d}\right)^2)^T\,. \tag{1.4}$$

Note that $Q = 0$ in the case of the laminar Navier-Stokes equations, unless additional source terms are added (to take into account gravity, for instance).

1.1.2 Spatial discretization

The spatial discretization of the fluid equations (1.1) is based on a vertex-centered finite element/finite volume formulation on unstructured meshes. It combines a HLLC approximate Riemann solver [Batten 1997] to compute the convective fluxes and the Galerkin centered method to evaluate the viscous terms. Second order space accuracy is achieved through a piecewise linear extrapolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure [Leer 1972] which uses a particular edge-based formulation with upwind elements. A specific slope limiter is employed to damp or eliminate spurious oscillations that may occur in the vicinity of discontinuities [Cournède 2006] (see Section 1.1.2).

Dual mesh construction

Let $\mathcal{H}$ be a mesh of domain Ω, the vertex-centered finite volume formulation consists in associating with each vertex P_i of the mesh a control volume or finite volume cell, denoted C_i. Discretized domain Ω_h (see Figure 1.3) can be written as the union of the elements or the union of the finite volume cells:

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

where N_K is the number of elements and N_V the number of vertices.

Note that the dual mesh (composed of cells) is built in a preprocessing step. Consequently, only a simplicial mesh is needed in the input. Several choices are possible to build finite volume cells. In this work, two methods were considered: median cells and containment cells.

Median cells. In 2D, this standard method consists in building cells bounded by segments of medians (so-called median cells), see Figure 1.1. In 3D, each tetrahedron is split into four hexahedra (one associated with each one of its four vertices). The eight vertices of the hexahedron associated with a point P_i are given by: (i) M_i, M_j, M_k, the middle points of the three edges incident to P_i, (ii) Gf_i, Gf_j, Gf_k, the gravity centers of the 3 faces containing P_i, (iii) G, the gravity center of the tetrahedra, and (iv) the vertex P_i considered. The cell C_i associated with vertex P_i is the union of all hexahedra of the tetrahedra surrounding P_i.

These cells enjoy a rather good robustness to distorted meshes, but they are not well-suited to stretched quasi-structured meshes. Traditional meshes for RANS computations exhibit structured boundary meshes where median cells are evenly distorted and can create spurious noise. To deal with such meshes the second method is used.

Containment cells. This method, introduced in 2D by Barth [Barth 1994] and generalized to 3D by [Gourvitch 2004], is supposed to be well-suited to discretize accurately the flow equations on highly anisotropic quasi-structured meshes (boundary layer meshes). In 2D, each triangle is split in three quadrangles, each one joining a vertex, the middle of each edge containing this vertex and the center of the containment circle of the triangle, see Figure 1.1. The center of the containment circle is the circumcenter if it is located inside the triangle, otherwise it is the center of the smallest circle containing the triangle (which, in fact, is the middle of the largest edge when the triangle has an obtuse angle). In 3D, it consists in subdividing each tetrahedron into four hexahedra cell around each vertex. The hexahedron cell vertices associated with vertex

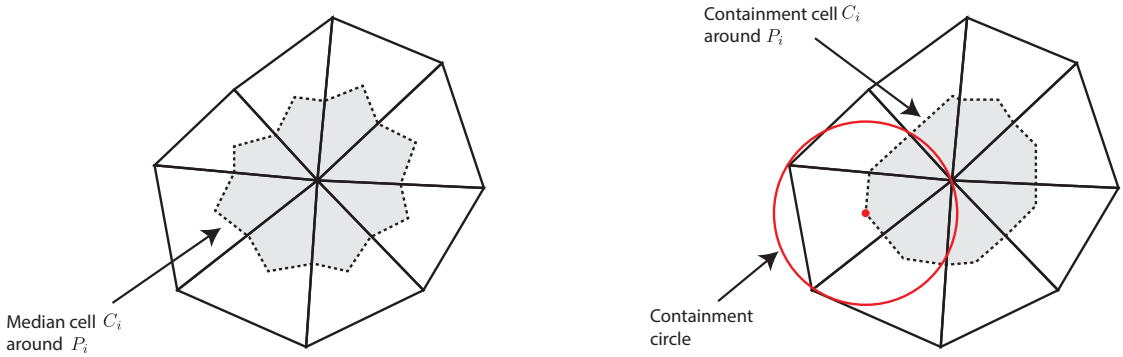


Figure 1.1: Median cell (left) and containment cell (right).

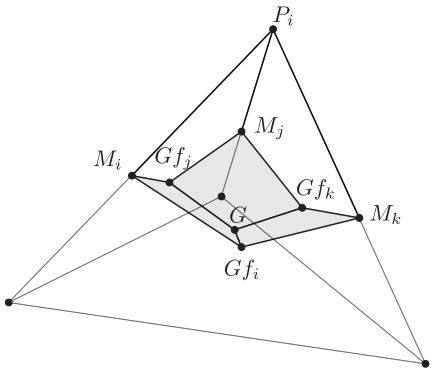


Figure 1.2: Median cell in 3D.

P_i are (i) the middle of the three edges issued from P_i, (ii) the containment circle center of the three faces containing P_i, (iii) the containment sphere center of the tetrahedron and (iv) the vertex P_i considered. The containment cell of vertex P_i is the union of all of these hexahedra cells. The containment sphere center corresponds to the sphere circumcenter if it is located inside the element, otherwise it is the center of the smallest sphere containing the tetrahedra which can be the middle of the largest edge or the center of the circumsphere containing the "largest" face.

Comparison of median and containment cells. Figure 1.3 compares the two approaches on a 2D mesh with a quasi-structured region. The same comparison in 3D is shown in Figure 1.4. These examples illustrate how the faces of containment cells are aligned with the flow direction in the presence of a boundary layer mesh, in contrast to median cells. This is why, containment cells are supposed to exhibit a better behaviour for highly stretched quasi-structured meshes.

Containment cells success and failures: Figure 1.3 shows the perfect cells obtained with containment cells on cartesian grids but such grids are seldom used. In practice containment cells are very efficient on fully structured meshes with very high aspect ratios, such as shown

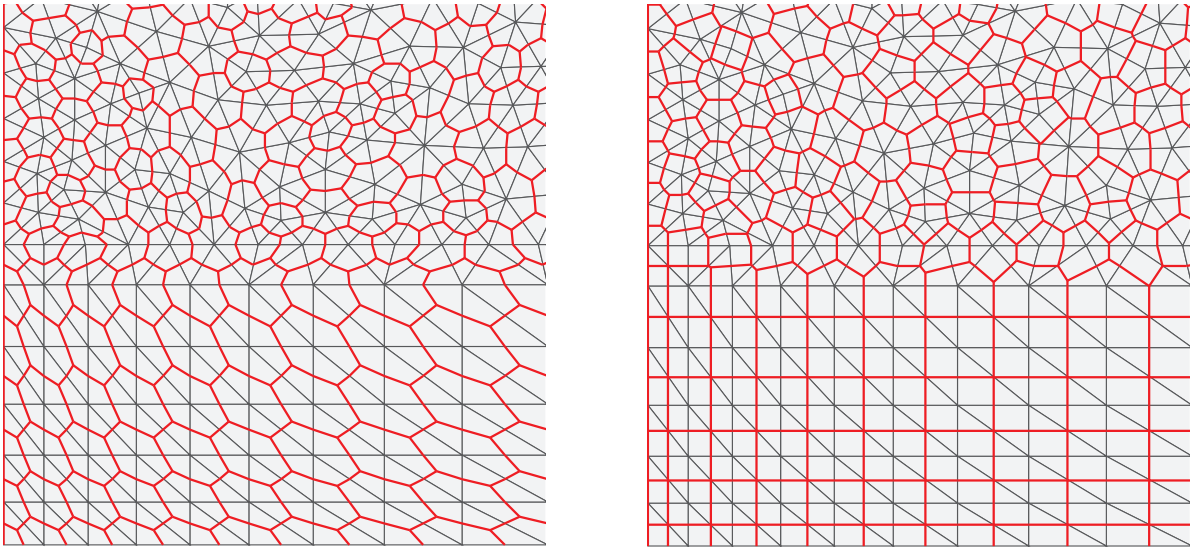


Figure 1.3: Median (left) and containment (right) dual meshes (in red) constructed on a mesh containing a quasi-structured region with a transition to a fully-unstructured one.

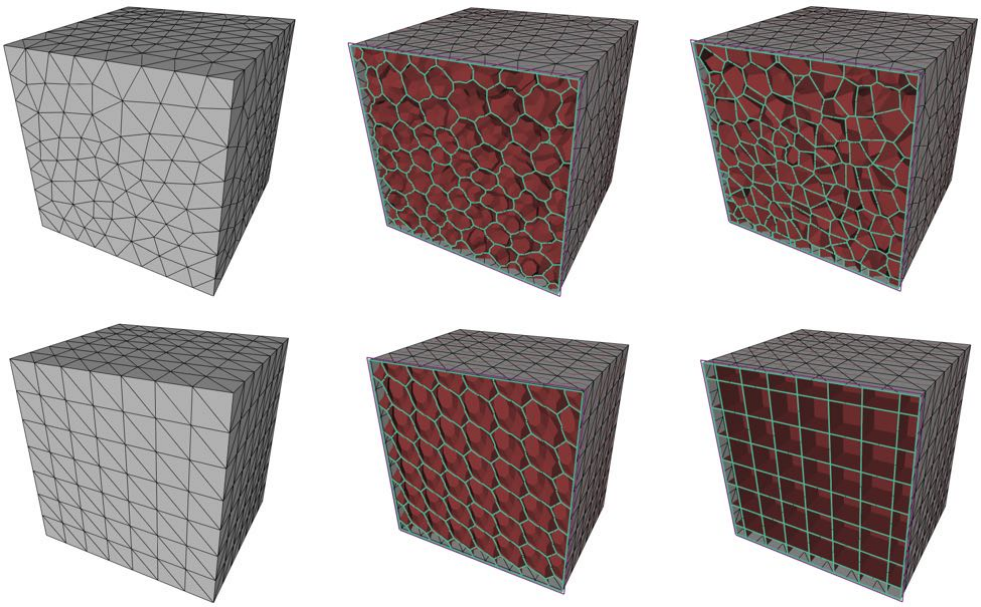


Figure 1.4: 3D median (middle) and containment (right) dual meshes constructed for an unstructured mesh (top) and a quasi-structured mesh (bottom).

in Figure 1.5, where the cells are quasi perfect. Median cells are in that case, nonetheless less stable that containment cells (require lower CFL and an exact differentiation of the viscous terms), but also yield a poor discretization of the equations and are more dissipatives.

Figure 1.6 shows the density and turbulent variable fields around a RAE2822 wing at Mach= 0.725, Reynolds= 6.5×10^6 and an angle of attack $\alpha = 2.92°$. We used here the standard Spalart-

Allamaras model, with HLLC solver, Fourth order numerical dissipation and the Piperno limiter. The linear system is inverted with SGS iterations with a precision of 0.1. We start with a uniform solution at $CFL = 0.1$, with a geometric growth of 1.02 and a maximal $CFL_{max} = 500$.

Both seem globally similar except that with median cells (left), the shock is more diffused and positioned aft of the shock predicted with containment cells (right). But a closer look at the turbulent variable $\tilde{\nu}$ field shows that median cells predict less turbulence near the trailing edge as can be seen in Figure 1.7 (bottom). Meanwhile, Figure 1.7 also shows that the velocity field (bottom) is smoother, which leads to less turbulence production. This is due to the artificial dissipation introduced by median cells. This in turn increase the drag computed by 3%.

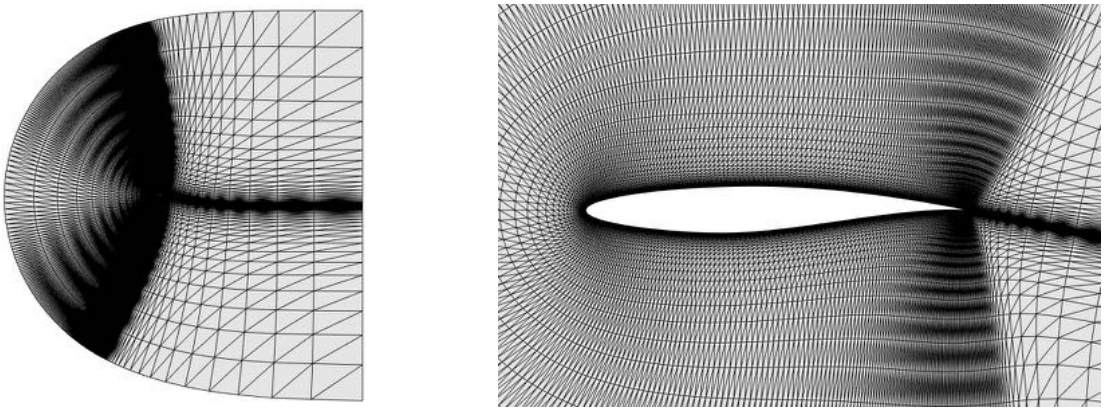


Figure 1.5: Structured mesh of an RAE2822.

Following this constatation, containment cells seemed appropriate for pseudo structured meshes with a structured boundary layer and an unstructured discretization for the outer field where containment cells were expected to behave roughly like median cells. However, containment cells tend to produce spurious noise and are more sensible to low-Mach on unstructured meshes, such as shown in Figure 1.8. The resulting density fields obtained on this mesh at Mach 0.1 and $\alpha = 2°$, with median and containment cells are shown in Figure 1.9. Both solutions are obtained with HLLC solver with no limiter and the residual is converged by 8 orders (1×10^{-8}). The noise introduced by containment cells is clearly visible.

Moreover, in order to recover structured cells in 3D, the structured mesh has to be generated from a specific subdivision of an hexahedral mesh [Gourvitch 2004]. Any permutation in the subdivision of the hexahedra leads to bad cells. It thus yields a strong constraint on the mesh generation in 3D. Finally, in a mesh adaptation context, median cells proved to be sufficient and as we were able to discretize boundary layers with unstructured meshes, we abandoned containment cells.

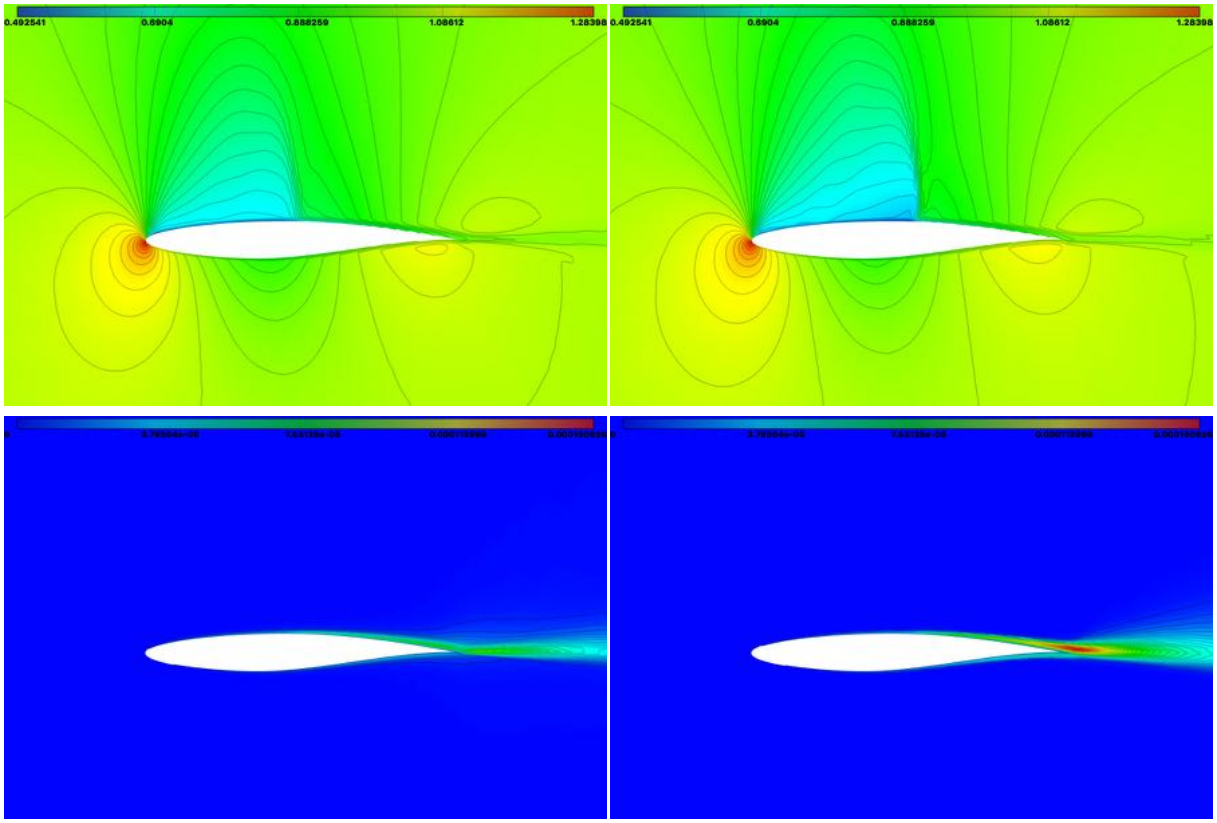


Figure 1.6: Transsonic RAE2822: density (top) and Spalart-Allmaras variable (bottom) obtained with median cells (left) and containment cells (right).

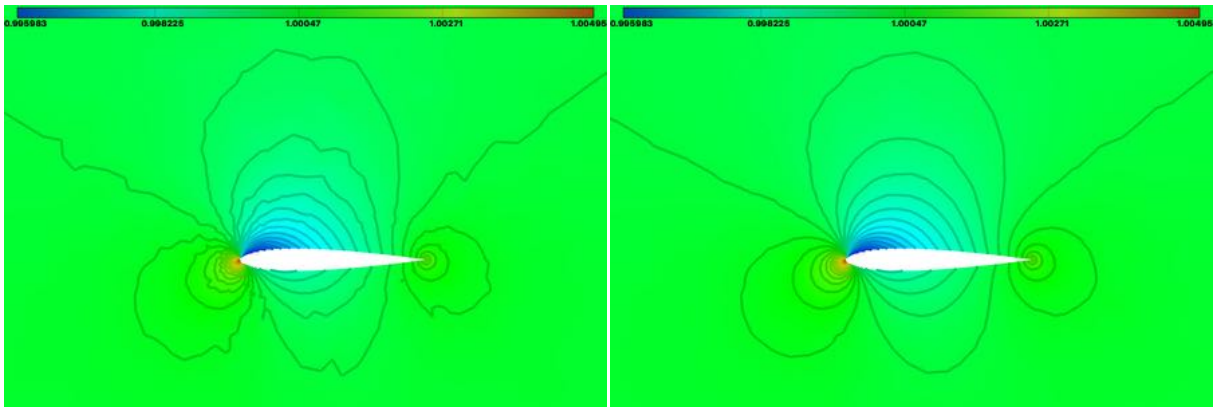


Figure 1.9: Subsonic inviscid NACA0012: density fields on unstructured mesh obtained with containment (left) and median (right) cells.

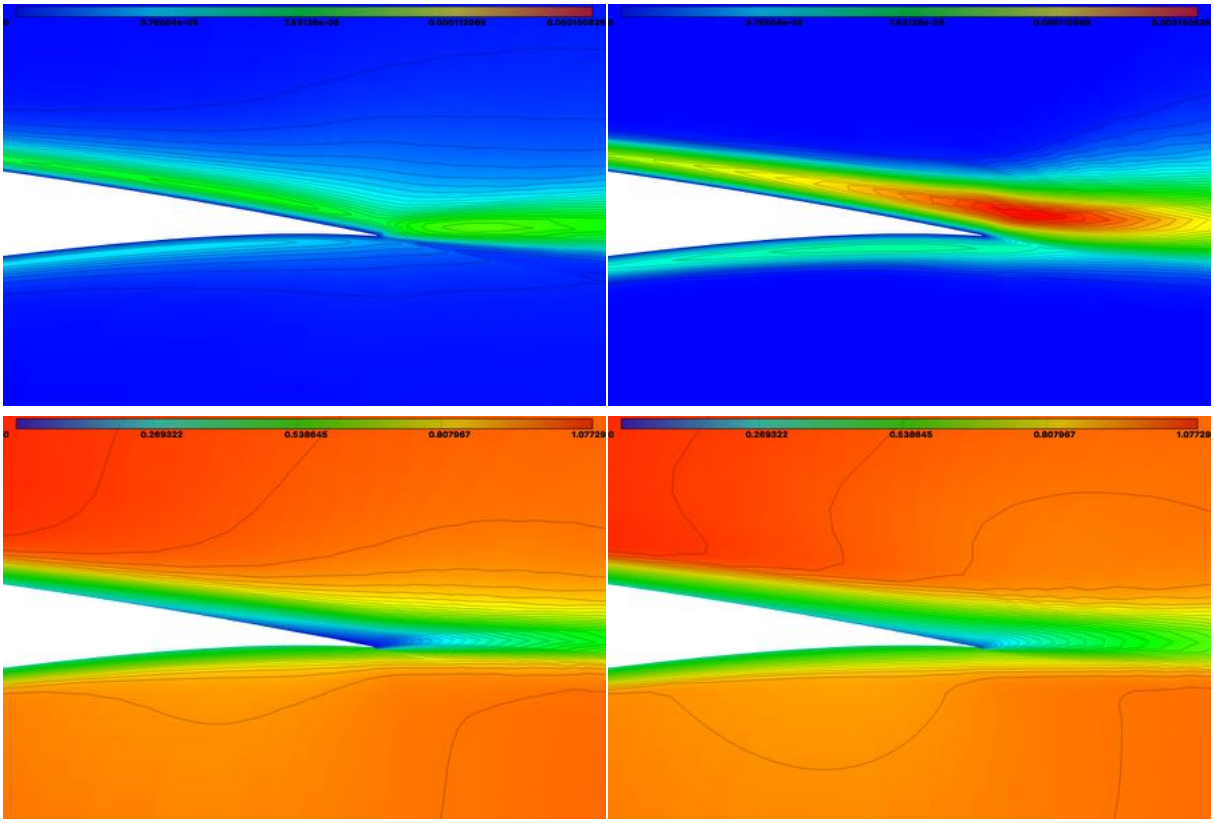


Figure 1.7: Transsonic RAE2822: zoom on the trailing edge of the Spalart-Allmaras variable (top) and velocity (bottom) fields obtained with median cells (left) and containment cells (right).

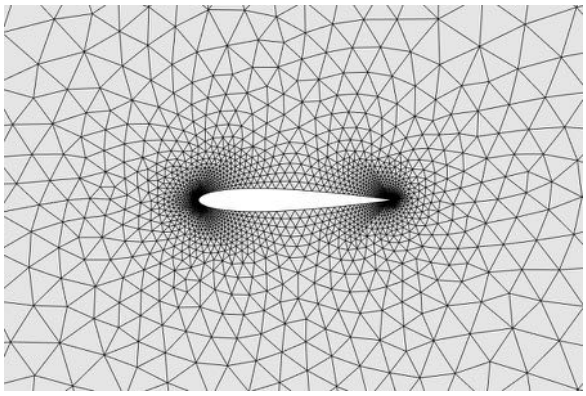


Figure 1.8: NACA0012 unstructured mesh.

Finite volume discretization

Based on a finite volume formulation, the Reynolds Averaged Navier-Stokes equations are integrated on each finite volume cell C_i (using the Green formula):

$$|C_i|\frac{dW_i}{dt} + \mathbf{F}_i = \mathbf{S}_i + \mathbf{Q}_i + \mathbf{\Gamma}_i\,, \tag{1.5}$$

where W_i is the mean value of the solution W on cell C_i, $\mathbf{F}_i$, $\mathbf{S}_i$, $\mathbf{Q}_i$ and $\mathbf{\Gamma}_i$ are respectively the numerical convective, viscous, source flux and boundary terms:

$$\mathbf{F}_i = \int_{\partial C_i} F(W)\cdot\mathbf{n}_i \mathrm{d}\gamma\,, \qquad \mathbf{S}_i = \int_{\partial C_i} S(W)\cdot\mathbf{n}_i\,\mathrm{d}\gamma\,,$$
$$\mathbf{Q}_i = \int_{C_i} Q(W)\,\mathrm{d}\Omega\,, \qquad \mathbf{\Gamma}_i = \int_{\partial C_i\cap\partial\Omega} G(W)\cdot\mathbf{n}_i\,\mathrm{d}\gamma\,,$$

where $\mathbf{n}_i$ is the outer normal to the finite volume cell surface ∂C_i, and F, S and Q are respectively the convective, viscous and source terms flux functions as defined previously in Relations (6.16), (6.17) and (1.4). $G(W)$ is a a boundary flux that depends on the type of boundary condition considered (see Section 1.1.3).

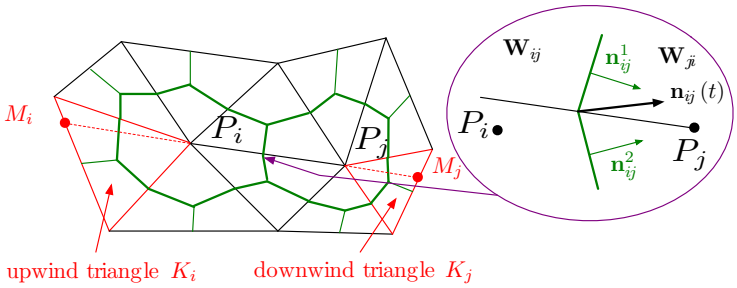


Figure 1.10: Illustration of finite volume cell construction in 2D: two neighbouring cells C_i and C_j and the upwind triangles K_i and K_j associated with edge P_iP_j.

Discretization of the convective terms. The integration of convective fluxes $\mathbf{F}$ of Equation (1.5) is done by decomposing the cell boundary into many facets ∂C_{ij}:

$$\mathbf{F}_i = \int_{\partial C_i} F(W)\cdot\mathbf{n}_i \mathrm{d}\gamma \approx \sum_{P_j\in\mathcal{V}(P_i)} F|_{\partial C_{ij}}\cdot\int_{\partial C_{ij}}\mathbf{n}_i\,\mathrm{d}\gamma\,,$$

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

$$\Phi_{ij} = \Phi_{ij}(W_i, W_j, \mathbf{n}_{ij}) = F|_{\partial C_{ij}}\cdot\int_{\partial C_{ij}}\mathbf{n}_i\,\mathrm{d}\gamma\,,$$

where $\mathbf{n}_{ij} = \displaystyle\int_{\partial C_{ij}}\mathbf{n}_i\,\mathrm{d}\gamma$. The numerical flux function approximates the hyperbolic terms on the common boundary ∂C_{ij}. We notice that the computation of the convective fluxes is performed

mono-dimensionally in the direction normal to the boundary of the finite volume cell. Therefore, the numerical calculation of the flux function Φ_{ij} at the interface ∂C_{ij} is achieved by the resolution of a one-dimensional Riemann problem in the direction of the normal $\mathbf{n}_{ij}$ by means of an approximate Riemann solver.

Godounov approximate Riemann solvers

In oder to compute the fluxes at the interface ∂C_{ij} the idea proposed by Godounov is to assume the solution W to be constant by cell, equal to its mean value $\frac{1}{|C_i|}\int_{C_i} W = W_i$ at each time step and solve the 1D Riemann problem at each face. These hyperbolic 1D problems expresse as

$$\frac{\partial W}{\partial t} + \frac{\partial F(W)}{\partial x} = 0,$$

with the initial condition at $t = 0$

$$W = W_L = W_i \text{ if } x < 0 \quad \text{and} \quad W = W_R = W_j \text{ if } x > 0,$$

with the interface ∂C_{ij} located at $x = 0$. The analytical solutions $w(x,t)$ of these 1D problems are known and it is possible to compute the flux at the interface $F(w(x,t))$. But the resolution of these 1D problems requires to determine the different shock waves, rarefaction waves and contact discontinuities involved to compute the solution $w(0,t)$ at the interface. This usually requires to solve a nonlinear problem with Newton-Raphson iterations. This process is thus extremely expensive to apply to all faces. Different "approximate" Riemann solvers have therefore been proposed in order to approach the solution of the 1D problem with more affordable computations.

Roe Solver: The idea of Roe solvers is to solve a local linearized system instead of the full non-linear one,

$$\frac{\partial W}{\partial t} + A(W)\frac{\partial W}{\partial x} = 0,$$

where A is the Jacobian of the fluxes F. If the exact Jacobian $A(W)$ (depending on W) is considered, we recover the same non-linear problem, but if an approximate constant value $\tilde{A}$ is considered, the problem is linear and can be easily solved using characteristics method. The exact solution can be expressed as a combination of the eigen vectors of $\tilde{A}$ and the flux at the interface can be expressed as

$$F|_{\partial C_{ij}} = \frac{1}{2}\left[F(W_i) + F(W_j) - |\tilde{A}| \cdot (W_i - W_j)\right].$$

where $|\tilde{A}| = P|\Delta|P^{-1}$ is the matrix obtained from the diagonalisation of $\tilde{A} = P\Delta P^{-1}$, taking the absolute values of the eigen values Δ.

Roe [Roe 1981] proposed to approach $A(W)$ with a matrix $\tilde{A}(W_L, W_R)$ that only depends on the left and right states and verifies the following constraints:

- $\tilde{A}(W_L, W_R)$ converges toward $A(W) = \frac{\partial F}{\partial W}$ as $W_L \to W$ and $W_R \to W$,
- $\tilde{A}(W_L, W_R) \cdot (W_L - W_R) = F(W_L) - F(W_R)$ for any W_L, W_R,

- The eigenvectors of $\tilde{A}$ are linearly independent.

Roe showed that such a matrix can be computed as $A(\tilde{W})$ where $\tilde{W}$ is an averaged state defined as

$$\begin{aligned}\tilde{\rho} &= \sqrt{\rho_L \rho_R},\\ \tilde{u} &= \frac{\sqrt{\rho_L}\mathbf{u}_L \cdot \mathbf{n}_{ij} + \sqrt{\rho_R}\mathbf{u}_R \cdot \mathbf{n}_{ij}}{\sqrt{\rho_L} + \sqrt{\rho_R}},\\ \tilde{H} &= \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}},\end{aligned}$$

with $H = E + p/\rho$ the enthalpy of the fluid.

Although it is obtained from mathematical and physical considerations, the term $-|\tilde{A}| \cdot (W_i - W_j)$ introduces an upwinding in the numerical scheme for each characteristic separately. This stabilize the unstable centered scheme obtained with $F|_{\partial C_{ij}} = \frac{1}{2}\left[F(W_i) + F(W_j)\right]$.

Roe-Turkel Solver: For low Mach number computations, numerical schemes using Riemann solvers are ill conditioned, the inversion of the linear system is more difficult and solutions exhibit noise. Basically, the flow becomes incompressible so that ρ is almost constant, any small disturbance in ρ has thus large effect on the solution. Meanwhile the dimensionless pressure which is of the order of the squared inverse of the Mach number, $p \approx \frac{1}{M^2}$ becomes large and extremely sensible to ρ. For low Mach computations, a discretization of the solution in primitive (entropic) variables $(p, \mathbf{u}, S)$ would be more appropriate (S being the entropy).

From a numerical point of view, as we mentioned earlier, the Roe matrix term can be seen as an upwinding stabilization. But when the Mach number gets close to 0, it can be shown [Guillard 1999, Boniface 2017, Viozat 1997] that the Roe upwinding induces a to dissipative stabilization of the momentum equation and an under-stabilization of the energie equation. This leads to oscillations in the pressure field. Expressed in primitive variables $(p, \mathbf{u}, S)$, the idea of Turkel is to rescale the pressure variations in the Roe solver, multiplying it by M^2. This numerically slows down the acoustic waves down to the flow velocity. The eigen values of the Roe-Turkel matrix are thus in primitive variables in 1D

$$\lambda_1 = \frac{1}{2}(1+\beta^2)\mathbf{u} + \sqrt{(1-\beta^2)^2\mathbf{u}^2 + 4\beta^2 c^2}, \quad \lambda_2 = \mathbf{u},$$

$$\lambda_3 = \frac{1}{2}(1+\beta^2)\mathbf{u} - \sqrt{(1-\beta^2)^2\mathbf{u}^2 + 4\beta^2 c^2},$$

where $\beta^2 = \min\left(2\frac{|\mathbf{u}|^2}{c^2}, 1\right)$ is of the order of the mach number. If $\beta = 1$ we recover the usual eigen values of Roe matrix: $\mathbf{u} - c, \mathbf{u}, \mathbf{u} + c$. When $\beta << 1$, we can see that all eigen values remain of the same order as $\mathbf{u}$. The fluxes are then rescaled back to the proper magnitudes so that the flux writes in primitive variables

$$F_e|_{\partial C_{ij}} = \frac{1}{2}\left[F_e(W_{i,e}) + F_e(W_{j,e}) - P_e^{-1}|P_e\tilde{A}_e| \cdot (W_{i,e} - W_{j,e})\right],$$

where the subscript $_e$ denotes the fluxes, vectors and matrices expressed in entropic variables. $P_e = Diag(\beta^2, 1, .., 1)$ is the preconditioning diagonal matrix used to scale the pressure.

As the initial Roe-Turkel solver is expressed in entropic variables, we have to transform it in conservative variables, using the relations

$$\frac{\partial W_e}{\partial t} = \frac{\partial W_e}{\partial W}\frac{\partial W}{\partial t} \quad \text{and} \quad \frac{\partial W_e}{\partial x} = \frac{\partial W_e}{\partial W}\frac{\partial W}{\partial x},$$

so that the Roe-Turkel flux expresses in conservatives variables

$$F|_{\partial C_{ij}} = \frac{1}{2}\left[F(W_i) + F(W_j) - \underbrace{\frac{\partial W}{\partial W_e}P_e^{-1}|P_e\tilde{A}_e|\frac{\partial W_e}{\partial W}}_{\tilde{A}^{RT}}\cdot(W_i - W_j)\right].$$

Finally, the Roe-Turkel matrix is decomposed as

$$\tilde{A}^{RT} = R\Delta R^{-1}$$

with in 2D:

$$R = \begin{bmatrix} 1 & 0 & \frac{1}{2\beta^2c^2} & \frac{1}{2\beta^2c^2} \\ \rho u & n_y & \frac{\rho u + (\lambda_2 - \beta^2\lambda_1)n_x}{2\beta^2c^2} & \frac{\rho u + (\lambda_3 - \beta^2\lambda_1)n_x}{2\beta^2c^2} \\ \rho v & -n_x & \frac{\rho v + (\lambda_2 - \beta^2\lambda_1)n_y}{2\beta^2c^2} & \frac{\rho v + (\lambda_3 - \beta^2\lambda_1)n_y}{2\beta^2c^2} \\ q^2 & -(n_xv - n_yu) & \frac{\frac{c^2}{\gamma-1} + q^2 + (\lambda_2 - \beta^2\lambda_1)\mathbf{u}\cdot\mathbf{n}}{2\beta^2c^2} & \frac{\frac{c^2}{\gamma-1} + q^2 + (\lambda_3 - \beta^2\lambda_1)\mathbf{u}\cdot\mathbf{n}}{2\beta^2c^2} \end{bmatrix} \tag{1.6}$$

$$R^{-1} = \begin{bmatrix} 1 - \frac{(\gamma-1)q^2}{c^2} & \frac{(\gamma-1)u}{c^2} & \frac{(\gamma-1)v}{c^2} & -\frac{(\gamma-1)}{c^2} \\ n_xv - n_yu & n_y & -n_x & 0 \\ \frac{s(\gamma-1)q^2 + \beta^2c^2\mathbf{u}\cdot\mathbf{n}}{0.5(\lambda_3-\lambda_2)} & -\frac{s(\gamma-1)u + \beta^2c^2n_x}{0.5(\lambda_3-\lambda_2)} & -\frac{s(\gamma-1)v + \beta^2c^2n_y}{0.5(\lambda_3-\lambda_2)} & \frac{s(\gamma-1)}{0.5(\lambda_3-\lambda_2)} \\ -\frac{r(\gamma-1)q^2 + \beta^2c^2\mathbf{u}\cdot\mathbf{n}}{0.5(\lambda_3-\lambda_2)} & \frac{r(\gamma-1)u + \beta^2c^2n_x}{0.5(\lambda_3-\lambda_2)} & \frac{r(\gamma-1)v + \beta^2c^2n_y}{0.5(\lambda_3-\lambda_2)} & -\frac{r(\gamma-1)}{0.5(\lambda_3-\lambda_2)} \end{bmatrix} \tag{1.7}$$

$$\Delta = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & \lambda_3 \end{bmatrix} \tag{1.8}$$

with

$$\begin{aligned} \lambda_1 &= |\mathbf{u}\cdot\mathbf{n}| & r &= \lambda_2 - \beta^2\lambda_1 \\ \lambda_2 &= \tfrac{1}{2}(1+\beta^2)|\mathbf{u}\cdot\mathbf{n}| + \sqrt{(1-\beta^2)^2|\mathbf{u}\cdot\mathbf{n}|^2 + 4\beta^2c^2} & s &= \lambda_3 - \beta^2\lambda_1 \\ \lambda_3 &= \tfrac{1}{2}(1+\beta^2)|\mathbf{u}\cdot\mathbf{n}| - \sqrt{(1-\beta^2)^2|\mathbf{u}\cdot\mathbf{n}|^2 + 4\beta^2c^2} & q^2 &= \frac{(\rho u)^2 + (\rho v)^2}{2} \end{aligned}$$

In 3D:

$$R = \begin{bmatrix} n_x & n_y & n_z & 0.5 & 0.5 \\ un_x & un_y - n_z & un_z + n_y & 0.5(u + n_xr) & 0.5(u + n_xs) \\ vn_x + n_z & vn_y & vn_z - n_x & 0.5(v + n_yr) & 0.5(v + n_ys) \\ wn_x - n_y & wn_y + n_x & wn_z & 0.5(w + n_zr) & 0.5(w + n_zs) \\ q^2n_x + vn_z - wn_y & q^2n_ywn_x - un_z & q^2n_z + un_y - vn_x & \frac{\mathbf{u}\cdot\mathbf{n}r + \rho E}{2} & \frac{\mathbf{u}\cdot\mathbf{n}s + \rho E}{2} \end{bmatrix} \tag{1.9}$$

$$R^{-1} = \begin{bmatrix} n_x(1-(\gamma-1)\frac{q^2}{c^2}) + (wn_y - vn_z) & (\gamma-1)\frac{u}{c^2}n_x & (\gamma-1)\frac{v}{c^2}n_x + n_z \\ n_y(1-(\gamma-1)\frac{q^2}{c^2}) + (un_z - wn_x) & (\gamma-1)\frac{u}{c^2}n_y - n_z & (\gamma-1)\frac{v}{c^2}n_y \\ n_z(1-(\gamma-1)\frac{q^2}{c^2}) + (vn_x - un_y) & (\gamma-1)\frac{u}{c^2}n_z + n_y & (\gamma-1)\frac{v}{c^2}n_z - n_x \\ (\gamma-1)\frac{q^2 s}{\beta^2 tc^2} + \frac{\mathbf{u}\cdot\mathbf{n}}{t} & -(\gamma-1)\frac{us}{\beta^2 tc^2} - \frac{n_x}{t} & -(\gamma-1)\frac{vs}{\beta^2 tc^2} - \frac{n_y}{t} \\ -(\gamma-1)\frac{q^2 r}{\beta^2 tc^2} - \frac{\mathbf{u}\cdot\mathbf{n}}{t} & (\gamma-1)\frac{ur}{\beta^2 tc^2} + \frac{n_x}{t} & \gamma-1)\frac{vr}{\beta^2 tc^2} + \frac{n_y}{t} \end{bmatrix} \tag{1.10}$$

$$\begin{bmatrix} (\gamma-1)\frac{w}{c^2}n_x - n_y & -\frac{(\gamma-1)}{c^2}n_x \\ (\gamma-1)\frac{w}{c^2}n_y + n_x & -\frac{(\gamma-1)}{c^2}n_y \\ (\gamma-1)\frac{w}{c^2}n_z & -\frac{(\gamma-1)}{c^2}n_z \\ -(\gamma-1)\frac{ws}{\beta^2 tc^2} - \frac{n_z}{t} & (\gamma-1)\frac{s}{\beta^2 tc^2} \\ (\gamma-1)\frac{wr}{\beta^2 tc^2} + \frac{n_z}{t} & -(\gamma-1)\frac{r}{\beta^2 tc^2} \end{bmatrix}$$

$$\Delta = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & 0 & \lambda_3 \end{bmatrix} \tag{1.11}$$

with

$$\lambda_1 = |\mathbf{u}\cdot\mathbf{n}| \qquad r = \lambda_2 - \beta^2\lambda_1$$
$$\lambda_2 = \frac{1}{2}(1+\beta^2)|\mathbf{u}\cdot\mathbf{n}| + \sqrt{(1-\beta^2)^2|\mathbf{u}\cdot\mathbf{n}|^2 + 4\beta^2 c^2} \qquad s = \lambda_3 - \beta^2\lambda_1$$
$$\lambda_3 = \frac{1}{2}(1+\beta^2)|\mathbf{u}\cdot\mathbf{n}| - \sqrt{(1-\beta^2)^2|\mathbf{u}\cdot\mathbf{n}|^2 + 4\beta^2 c^2} \qquad q^2 = \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{2}$$
$$t = -\frac{1}{2}\sqrt{(1-\beta^2)^2|\mathbf{u}\cdot\mathbf{n}|^2 + 4\beta^2 c^2}$$

One could note the striking difference between the 2D and 3D expressions of the matrices. This is due to the fact that the first eigen value has a multiplicity higher that 1 (of 2 in 2D and 3 in 3D). Hence it is not associated to a single eigen vector but rather to a whole sub-vector space and any basis of this space can be chosen to express R. In the 2D case, the basis has been chosen so that the first component of the second eigen vector is null. We could recover a similar expression in 3D by using linear combinations of the three first columns of R and using the fact that $n_x^2 + n_y^2 + n_z^2 = 1$.

The asymptotic analysis shows [Viozat 1997] that this formulation with $\beta^2 = \min\left(2\frac{|\mathbf{u}|^2}{c^2}, 1\right)$ introduces more numerical dissipation on the energy equation and less on the momentum one when the Mach number tends to 0, so that the scheme remains stable and not too dissipative. Yet, the β parameter can be freely chosen. In our case, we implemented two options. The β parameter is either fixed to a user defined value (generally chosen as $\beta^2 = \min\left(2\frac{|\mathbf{u}_\infty|^2}{c^2}, 1\right)$) or

Iterations	CFL	HLLC SGS iterations	RT SGS iterations
50	1.14	1	6
100	13.2	4	18
200	100	12	3
300	100	13	5
400	100	20	5
500-1700	100	20	5

Table 1.1: Low subsonic NACA0012: Number of SGS iterations required to inverse the linear system with HLLC and Roe-Turkel solvers

computed for each node as $\beta_i^2 = \min\left(2\frac{|\mathbf{u}_i|^2}{c_i^2}, 1\right)$ depending on the local Mach number. That way, the preconditioner is active only in the low Mach number regions such as boundary layers.

Figure 1.11 shows the density and pressure fields and isolines around a NACA0012. The flow is considered inviscid at a Mach number of $M = 0.01$ and angle of attack $\alpha = 2°$. Solutions are obtained with standard HLLC solver and Roe-Turkel solver with $\beta = 0.02$. We can clearly see that the HLLC solver (left) let unphysical instabilities grow in the density (top) and pressure (bottom) fields at low Mach numbers, while the Roe-Turkel solver (right) successfully stabilises the solutions.

Moreover, the system has a better numerical conditioning so that the linear systems converge faster. The present case was computed without limiters and we require the linear system to be converged down to an order of magnitude (0.1). The CFL law is geometric with a growth factor of 1.02 (CFL increases of 2% at each iteration), starting from $CFL = 0.1$ with a maximal $CFL_{\max} = 100$. Figure 1.12 shows the convergence of the density residuals throughout the simulation, in term of iterations and CPU time. Both solvers yield roughly the same decrease in the residual in term of iterations but each iteration of the HLLC solver is cheaper than Roe-Turkel solver during the pseudo-unstationary convergence (the first 200 iterations) and becomes way more expensive in the asymptotic regime. The difference lies in the number of iterations required by the SGS solver to inverse the linear system. In the asymptotic regime, HLLC solver requires more than 20 iteration (maximal number of iterations allowed) while the Roe-Turkel solver demands only 5 iterations. Table 1.1 summarizes the number of SGS iterations needed to inverse the linear system throughout the computation with HLLC and Roe-Turkel solvers.

Still, the HLLC solver enjoys better stability properties, hence we use the Roe-Turkel only for low mach simulations (below $M = 0.1$).

HLLC approximate Riemann solver. The idea of the HLLC flow solver is to consider locally a simplified Riemann problem with two intermediate states depending on the local left and right states. The simplified solution to the Riemann problem consists of a contact wave with a velocity S_M and two acoustic waves, which may be either shocks or expansion fans. The acoustic waves have the smallest and the largest velocities (S_i and S_j, respectively) of all the waves present in the exact solution. If $S_i > 0$ then the flow is supersonic from left to right and the upwind flux is simply defined from $F(W_i)$ where W_i is the state to the left of the

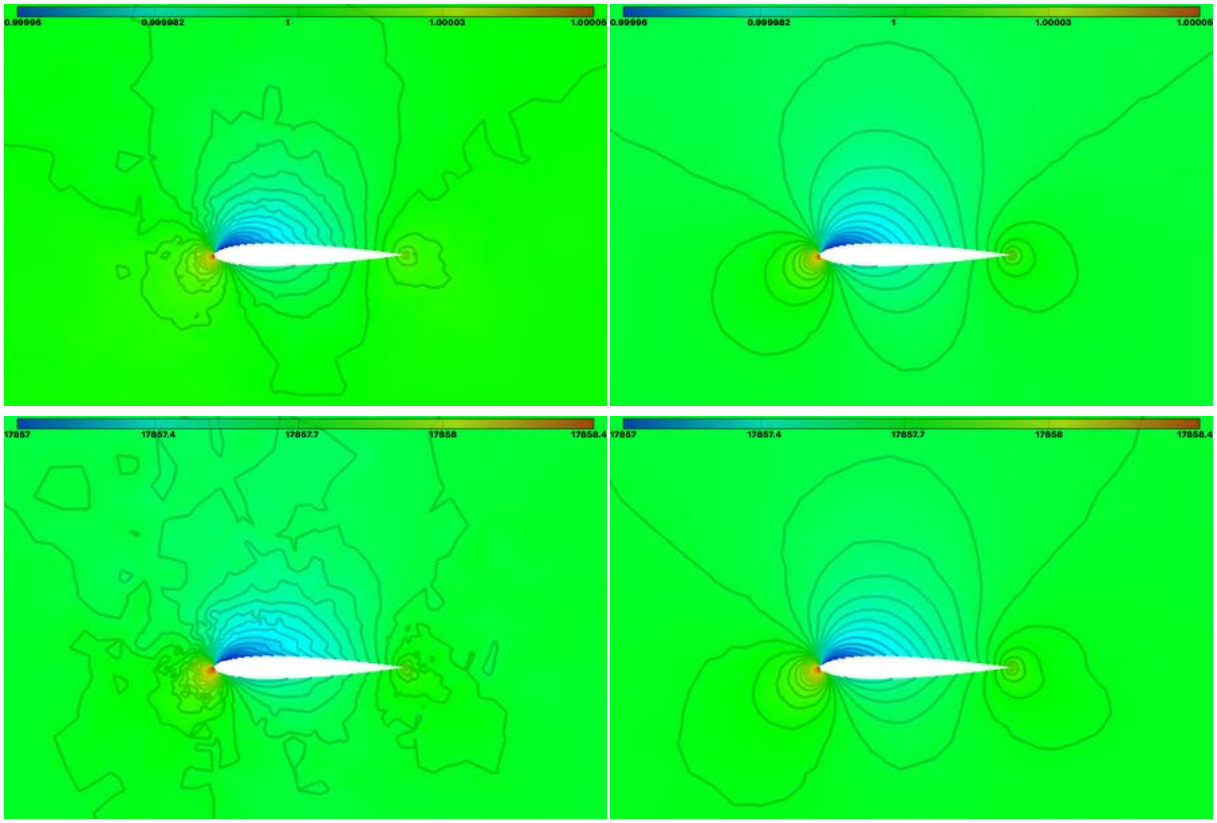
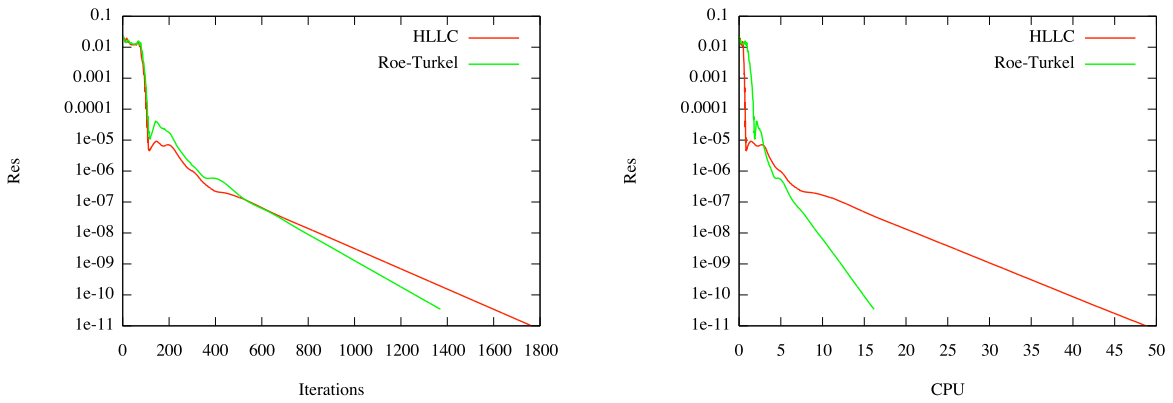


Figure 1.11: Inviscid NACA0012, $M = 0.01$, $\alpha = 2°$, computed with HLLC (left) and Roe-Turkel (right) solvers: Density (top) and Pressure (bottom) fields and isolines.



(a) (b)

Figure 1.12: Inviscid NACA0012, $M = 0.01$, $\alpha = 2°$. Convergence of density residual with respect to the number of iterations (a) and CPU time (b) using HLLC solver (red) and Roe-Turkel solver (green).

discontinuity. Similarly, if $S_j < 0$ then the flow is supersonic from right to left and the flux is defined from $F(W_j)$ where W_j is the state to the right of the discontinuity. In the more difficult subsonic case when $S_i < 0 < S_j$ we have to calculate $F(W_i^\star)$ or $F(W_j^\star)$. Consequently, the HLLC flux is given by:

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

$W_i^\star$ and $W_j^\star$ are evaluated as follows. We denote by $\eta = \mathbf{u}\cdot\mathbf{n}$. Assuming that $\eta^\star = \eta_i^\star = \eta_j^\star = S_M$, the following evaluations are proposed [Batten 1997] (the subscripts $_i$ and $_j$ are omitted for clarity):

$$W^\star = \frac{1}{S - S_M}\begin{pmatrix} \rho\,(S-\eta) \\ \rho\mathbf{u}\,(S-\eta) + (p^\star - p)\mathbf{n} \\ \rho E\,(S-\eta) + p^\star S_M - p\eta \end{pmatrix} \quad \text{where} \quad p^\star = \rho\,(S-\eta)(S_M-\eta) + p\,.$$

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

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

and the acoustic wave speeds based on the Roe average:

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

With such waves velocities, the approximate HLLC Riemann solver has the following properties. It automatically (i) satisfies the entropy inequality, (ii) resolves isolated contacts exactly, (iii) resolves isolated shocks exactly, and (iv) preserves positivity.

MUSCL extrapolation

The MUSCL type reconstruction method has been designed to increase the order of accuracy of the scheme [Leer 1972]. The idea is to use extrapolated values W_{ij} and W_{ji} instead of W_i and W_j at the interface ∂C_{ij} to evaluate the flux with an approximate Riemann solver. Note that, in the implementation, the primitive variables $(\rho, \mathbf{u}, p)$ are extrapolated to guarantee the positivity of the density and the pressure, then the conservative variables are reconstructed from these values. Thus, the gradients of the primitive variables are evaluated. However, in the following, to simplify the notation, we still denote by W the primitive variables vector. The numerical flux becomes:

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

where W_{ij} and W_{ji} are linearly extrapolated as:

$$W_{ij} \;=\; W_i + \frac{1}{2}(\nabla W)_i \cdot \overrightarrow{P_iP_j} \qquad \text{and} \qquad W_{ji} \;=\; W_j + \frac{1}{2}(\nabla W)_j \cdot \overrightarrow{P_jP_i}\,.$$

2nd-order accurate version. In contrast to the original MUSCL approach, the approximate "slopes" $(\nabla W)_{ij}$ and $(\nabla W)_{ji}$ are defined for any edge and obtained using a combination of centered, upwind and nodal gradients in order to build low dissipation second order numerical scheme [Debiez 2000]. The centered gradient, which is related to edge P_iP_j, is implicitly defined along edge P_iP_j by the relation:

$$(\nabla W)^C_{ij} \cdot \overrightarrow{P_iP_j} = W_j - W_i \qquad \text{and} \qquad (\nabla W)^C_{ij} \cdot \overrightarrow{P_jP_i} = W_i - W_j \,.$$

Upwind and downwind gradients, which are also related to edge P_iP_j, are computed according to the definition of upwind and downwind tetrahedra of edge P_iP_j. These tetrahedra are respectively denoted K_{ij} and K_{ji}. 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 line defined by the edge P_iP_j, see Figure 1.14.

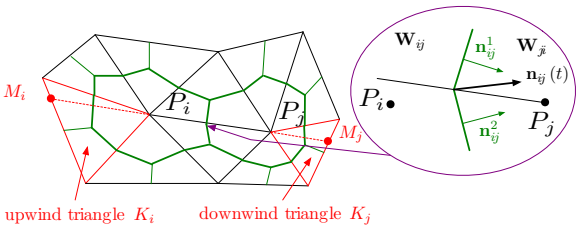


Figure 1.13: Vertex centered finite volume flux computation.

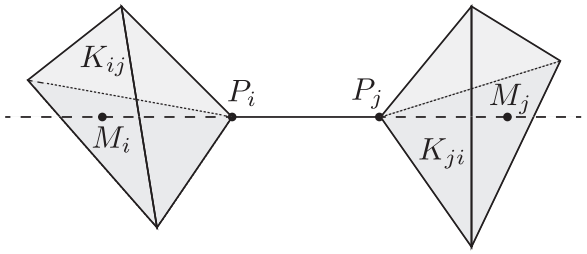


Figure 1.14: Downwind K_{ij} and upwind K_{ji} tetrahedra associated with edge P_iP_j.

Upwind and downwind gradients are then defined 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 $(\nabla W)|_K = \sum_{P\in K} W_P \nabla\phi_P|_K$ is the P_1-Galerkin gradient on tetrahedron K. Parametrized gradients are built by introducing the β-scheme:

$$\begin{aligned}(\nabla W)_i \cdot \overrightarrow{P_iP_j} &= (1-\beta)(\nabla W)^C_{ij} \cdot \overrightarrow{P_iP_j} + \beta\,(\nabla W)^U_{ij} \cdot \overrightarrow{P_iP_j} \\ (\nabla W)_j \cdot \overrightarrow{P_jP_i} &= (1-\beta)(\nabla W)^C_{ij} \cdot \overrightarrow{P_jP_i} + \beta\,(\nabla W)^D_{ij} \cdot \overrightarrow{P_jP_i}\,,\end{aligned}$$

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 $\beta = 1$.

Fourth-order numerical dissipation: V4-scheme. The most accurate β-scheme is obtained for $\beta = 1/3$ [Koren 1993, Debiez 2000]. Indeed, it can be demonstrated that this scheme is third-order for the two-dimensional linear advection on structured triangular meshes. On unstructured meshes, a second-order scheme with a fourth-order numerical dissipation is obtained. These high-order gradients are given by:

$$\begin{aligned}(\nabla W)^{V4}_i \cdot \overrightarrow{P_iP_j} &= \frac{2}{3}(\nabla W)^C_{ij} \cdot \overrightarrow{P_iP_j} + \frac{1}{3}(\nabla W)^U_{ij} \cdot \overrightarrow{P_iP_j} \\ (\nabla W)^{V4}_j \cdot \overrightarrow{P_jP_i} &= \frac{2}{3}(\nabla W)^C_{ij} \cdot \overrightarrow{P_jP_i} + \frac{1}{3}(\nabla W)^D_{ij} \cdot \overrightarrow{P_jP_i}\,.\end{aligned}$$

Sixth-order numerical dissipation: 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 P_1-Galerkin gradients. The nodal P_1-Galerkin gradient of P_i is related to cell C_i and is computed by averaging the gradients of all the tetrahedra containing vertex P_i:

$$(\nabla W)^N_{P_i} = \frac{1}{4\,|C_i|} \sum_{K \in C_i} |K| (\nabla W)|_K \,.$$

A sixth-order dissipation scheme is obtained by considering the following high-order gradient:

$$\begin{aligned}
(\nabla W)^{V6}_i \cdot \overrightarrow{P_iP_j} &= \Big[(\nabla W)^{V4}_i - \frac{1}{30}\left((\nabla W)^U_{ij} - 2\,(\nabla W)^C_{ij} + (\nabla W)^D_{ij}\right) \\
&- \frac{2}{15}\left((\nabla W)^N_{M_i} - 2\,(\nabla W)^N_{P_i} + (\nabla W)^N_{P_j}\right)\Big] \cdot \overrightarrow{P_iP_j} \\
(\nabla W)^{V6}_j \cdot \overrightarrow{P_jP_i} &= \Big[(\nabla W)^{V4}_j - \frac{1}{30}\left((\nabla W)^D_{ij} - 2\,(\nabla W)^C_{ij} + (\nabla W)^U_{ij}\right) \\
&- \frac{2}{15}\left((\nabla W)^N_{M_j} - 2\,(\nabla W)^N_{P_j} + (\nabla W)^N_{P_i}\right)\Big] \cdot \overrightarrow{P_jP_i} \,,
\end{aligned}$$

where $(\nabla W)^N_{M_i}$ and $(\nabla W)^N_{M_j}$ are the gradients at the intersection points M_i and M_j between the line defined by P_iP_j and upwind and downwind tetrahedra. These gradients are computed by linear interpolation of the nodal gradients of faces containing M_i and M_j, see Figure 1.14.

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

In this thesis, we use four limiters:

- MinMod Limiter:

$$Lim_{MM}(u,v) = \begin{cases} min(|u|,|v|) & \text{if } uv > 0 \\ 0 & \text{otherwise} \end{cases}$$

  The limited gradient is then computed using the centered and the upwind gradients as

$$(\nabla W)^{Lim}_i = Lim_{MM}(\nabla W^C_{ij}, \nabla W^U_{ij}) \quad \text{or} \quad (\nabla W)^{Lim}_j = Lim_{MM}(\nabla W^C_{ij}, \nabla W^D_{ij}) \,.$$

  The MinMod limiter cannot be used with the V4 or V6 high-order gradients.

- Van-Albada Limiter:

$$Lim_{VA}(u,v) = \begin{cases} \dfrac{(u^2+\varepsilon)v + (v^2+\varepsilon)u}{u^2+v^2+2\varepsilon} & \text{if } uv > 0 \\ 0 & \text{otherwise} \end{cases}$$

The limited gradient is then computed using the centered and the upwind gradients as

$$(\nabla W)_i^{Lim} = Lim_{VA}(\nabla W_{ij}^C, \nabla W_{ij}^U) \quad \text{or} \quad (\nabla W)_j^{Lim} = Lim_{VA}(\nabla W_{ij}^C, \nabla W_{ij}^D)\,.$$

The Van-Albada limiter cannot be used with the V4 or V6 high-order gradients.

- Piperno Limiter [Piperno 1998]: This limiter is expressed in a factorized form,

$$(\nabla W)^{Lim} = \nabla W^C \; \psi_{PI}\left(\frac{\nabla W^C}{\nabla W^{V4}}\right),$$

with

$$\psi_{PI}(R) = (\frac{1}{3} + \frac{2}{3}R)\begin{cases} \dfrac{3\frac{1}{R^2} - 6\frac{1}{R} + 19}{\frac{1}{R^3} - 3\frac{1}{R} + 18} & \text{if } R < 1 \\ 1 + (\dfrac{3}{2}\dfrac{1}{R} + 1)(\dfrac{1}{R} - 1)^3 & \text{if } R \geq 1 \end{cases}$$

where

$$R = \frac{\nabla W^C}{\nabla W^{V4}}.$$

The Piperno limiter can be only used with the V4 high-order gradient.

- Koren Limiter [Koren 1993]:

$$Lim_{KO}(u,v,w) = \begin{cases} Sgn(u)min(2|u|, 2|v|, |w|) & \text{if } uv > 0 \\ 0 & \text{otherwise} \end{cases}$$

This three-entries limiter introduced by Koren is a generalization of the Superbee limiter [Cournède 2006] and yields to the limited gradient (for P_i):

$$(\nabla W)_i^{Lim} = Lim_{KO}\left((\nabla W)_{ij}^C,\ (\nabla W)_{ij}^D,\ (\nabla W)_i^{HO}\right),$$

where the high-order gradient is either the V4 or V6 gradient.

Influence Stencil: Finally the convective part of fluxes for the cell C_i writes

$$\mathbf{F}_i = \sum_{e \in \mathcal{E}_i} \Phi^{inv}(e) = \sum_{P_j \in \mathcal{V}^1(P_i)} \Phi_{ij}^{HLLC}(W_{ij}, W_{ji}, \mathbf{n}_{ij}). \tag{1.12}$$

For a given neighbour P_j, W_{ij} and W_{ji} are computed with a MUSCL extrapolation as stated previously. In particular, as can be seen in Figure 1.14, W_{ji} depends on the centered gradient and thus on the nodal values of P_i and P_j but also on the upwind gradient. As the upwind gradient is computed with the nodal values of the upwind tetrahedron/triangle K_{ji}, W_{ji} and in turn the HLLC flux associated with the edge P_iP_j depend on the nodal values of the neighbours of P_j. The inviscid fluxes of the cell C_i and thus the residual of node P_i depend on the neighbours of P_i and the neighbours of its neighbours which forms the second order ball $\mathcal{V}^2(P_i)$.

Discretization of the viscous terms

In `Wolf`, we discretize the viscous terms using either the finite element method (FEM) or an edge based formulation. We need to evaluate

$$\mathbf{S}_i \;=\; \int_{C_i} \nabla \cdot S(W)\,\mathrm{d}\Omega \;=\; \sum_{P_j \in \mathcal{V}(P_i)} \int_{\partial C_{ij}} S(W)\cdot \mathbf{n}\,\mathrm{d}\gamma + BT,$$

where ∂C_{ij} is the common interface between cells C_i and C_j, and BT stands for the boundary terms.

Finite element method: In the FEM approach, we directly deal with the volumic integral, relying on a linear representation of the fields. Let φ_i be the $\mathbb{P}_1$ finite element basis function associated with vertex P_i in element K, we have:

$$\int_K \nabla\varphi_i\,\mathrm{d}\Omega = -\int_{\partial C_i \cap K} \mathbf{n}\,\mathrm{d}\gamma$$

and as the solution is represented on the $\mathbb{P}_1$ basis, $S(W_i)$ (which comes from a gradient) is assumed constant by parts on each element K, for example:

$$\mathcal{T}_{xy}|_K = \sum_{P_i \in K} \mu|_K \left(u_i \frac{\partial \varphi_i}{\partial y} + v_j \frac{\partial \varphi_i}{\partial x} \right),$$

where $\mu|_K$ is the mean value of μ on the element to be constant. Then we obtain

$$\begin{aligned} \sum_{P_j \in \mathcal{V}(P_i)} \int_{\partial C_{ij}} S(W)\cdot \mathbf{n}\,\mathrm{d}\gamma \;&=\; \sum_{K \ni P_i} S(W)|_K \cdot \int_{\partial C_i \cap K} \mathbf{n}\,\mathrm{d}\gamma \\ &=\; -\sum_{K \ni P_i} \int_K S(W)|_K \cdot \nabla\phi_i\,\mathrm{d}\mathbf{x}\,. \end{aligned}$$

The effective computation of the previous integral then leads to the computation of integrals of the following form:

$$\int_K \nabla\varphi_i \cdot \nabla\varphi_j\,\mathrm{d}\Omega \;=\; |K|\,\nabla\varphi_i|_K \cdot \nabla\varphi_j|_K.$$

In practice, as the nodal viscous term is decomposed as the sum of the contribution of each element containing this node, we cycle on the elements instead of the vertices to compute them. Given an element $K = (P_i, P_j, P_k, P_l)$ we have the partial flux associated with each vertex

$$\Phi^{visc}_{i,K}(W_i, W_j, W_k, W_l) = \int_{\partial C_i \cap K} S|_K \cdot \mathbf{n}\,\mathrm{d}\gamma,$$

computed as mentioned previously.

Edge-based viscous terms: In order to avoid the finite element analysis on pyramids and hexahedra for viscous terms when using hybrid meshes we implemented an edge based formulation of viscous fluxes. Although it has been developed for hybrid meshes, this approach also

works for full triangles/tetrahedral meshes. In the edge-based approach we deal with the surface integral $\int_{\partial C_{ij}} S(W_i) \cdot \mathbf{n} \, \mathrm{d}\gamma$ as for the inviscid fluxes, with a centered scheme. Namely, we have

$$S_{ij} = \int_{\partial C_{ij}} S(W) \cdot \mathbf{n} \, \mathrm{d}\gamma = |C_{ij}| \frac{1}{2} \left(S_i(W) + S_j(W) \right) \cdot \mathbf{n}_{ij}.$$

To do so, we need to compute the nodal values of $S(W)$ which involve gradients. These nodal gradients are already assembled for the MUSCL extrapolation with L_2 projection or least square minimization. We thus compute for each edge the mean value of the nodal gradients $\nabla W|_{C_{ij}}$ and the eddy and turbulent viscosity $\mu|_{C_{ij}}$ so that for example

$$\mathcal{T}_{xy}|_{C_{ij}} = \mu|_{C_{ij}} \left(\left. \frac{\partial u_i}{\partial y} \right|_{C_{ij}} + \left. \frac{\partial v_j}{\partial x} \right|_{C_{ij}} \right),$$

and the numerical viscous flux is defined as

$$\Phi_{ij}^{visc}(W_i, W_j) = |C_{ij}| S(\nabla W_i, \nabla W_j)|_{C_{ij}} \cdot \mathbf{n}_{ij}.$$

Influence Stencil: In the finite elements formulation, as each element produces a viscous flux for each of its nodes we can deduce the total viscous fluxes for a given cell/node i as:

$$\mathbf{S}_i = \sum_{K \ni P_i} \Phi_{i,K}^{visc}(W_i, W_j, W_k, W_l), \tag{1.13}$$

so that the viscous fluxes of a node only depend on the first order ball of P_i: $\mathcal{V}^1(P_i)$.

For the edge-based formulation, we have a flux for each edge that depends on the nodal gradients of both nodes of the considered edge. Hence, as for the inviscid fluxes, the viscous fluxes of the cell C_i and thus the residual of node P_i depend on the neighbours of P_i and the neighbours of its neighbours, the second order ball $\mathcal{V}^2(P_i)$.

1.1.3 Boundary Conditions

Boundary conditions in `Wolf` are imposed element-wise, *e.g.* a boundary flux is computed for each boundary edge/face and added to each of its vertices.

No-slip Boundary Condition

For no-slip boundary conditions, $\mathbf{u} = 0$ and $\tilde{\nu} = 0$ are strongly enforced at each iteration. Consistently, we impose $\Phi_\rho = 0$ at the boundary for the non-penetration condition. The energy flux is fixed according to the desired temperature behaviour: for an adiabatic wall it is null and for an isothermal wall the energy variable is enforced similarly to the velocity. Hence, for an adiabatic wall,

$$\Phi_{NoSlip} = (0, 0, 0, 0, 0)^t,$$

where the flux on the velocity component is imposed to zero for convenience. Though this also allows to compute the forces on the body through the defect residual of the momentum equations, see below for more details.

Slip Boundary Condition:

For this boundary condition we impose weakly

$$\mathbf{u} \cdot \mathbf{n} = 0\,. \tag{1.14}$$

To this end, we compute the flux Φ between the state on the boundary W and a mirror state $\overline{W}$:

$$\Phi_{Slip} = \Phi^{HLLC}(W, \overline{W}, \mathbf{n}),$$

where

$$W = \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \quad \text{and} \quad \overline{W} = \begin{pmatrix} \rho \\ \rho\mathbf{u} - 2\,\rho\,(\mathbf{u}\cdot\mathbf{n})\,\mathbf{n} \\ \rho E \end{pmatrix}.$$

If Condition (1.14) is satisfied then $W = \overline{W}$ and thus $\Phi^{HLLC}(W, \overline{W}, \mathbf{n}) = F(W)\cdot\mathbf{n}$. Moreover, if W verifies Relation (1.14), then $F(W)\cdot\mathbf{n}$ simplifies to:

$$\Phi_{Slip} = F(W)\cdot\mathbf{n} = (0, p\,\mathbf{n}, 0)^t\,.$$

Therefore, if the desired condition is satisfied, then the boundary flux reduces to its well known commonly used form.

Nevertheless, the state W on the boundary does not satisfy this condition unless it is imposed strongly which is not possible as we will no more conserve the mass. This problem is solved by computing the flux between the state and its mirror state. This flux depends on the considered approximate Riemann solver.

Supersonic/Subsonic Inlet

Subsonic and supersonic inflow boundary conditions are used to prescribe a consistent physical incoming flow with a given total pressure, total temperature and flow direction. This is especially used in enclosed geometries such as encountered in turbomachinery applications, pipes, ducts, ...

Flux prescription. In order to provide a numerically and physically consistent boundary condition, we rely on Riemann invariants across the boundary surface to compute an appropriate external state. The boundary flux is then computed with the same approximate Riemann solver (HLLC, Roe, ...) as inside the domain, using the external and the inner state.

In the supersonic case, all eigen values of the Riemann problem are positive, meaning that no information comes from the inside domain (sound waves cannot go upstream). A numerical consistent boundary condition can be provided by prescribing the full desired state outside which does not depends on the inner state

$$W_{ext} = (\rho_{ext},\ u_{ext},\ p_{ext})\,.$$

In the subsonic case, sound wave travels upstream and imposing a given state as in supersonic case leads to wave reflexions and instabilities slowing convergence. The negative Riemann invariant is thus used to compute a consistent external state. Given the inner state $(\rho, \rho u, \rho v, \rho w, \rho e)$

and the normal of the face, we can compute the normal velocity

$$\eta = \mathbf{u} \cdot \mathbf{n},$$

the boundary sound velocity

$$c = \sqrt{\gamma p/\rho},$$

the outgoing characteristic that must be conserved

$$R^- = \eta - \frac{2c}{\gamma - 1},$$

and the ingoing enthalpy, defined by our desired state

$$H_{tot} = \frac{\gamma}{\gamma - 1} R T_{tot}.$$

These quantities are conserved accros the boundary and are thus related to external state by:

$$H_{tot} = \frac{c_{ext}}{\gamma - 1} + \frac{u_{ext}^2}{2} \quad \text{and} \quad R^- = u_{ext} - \frac{2c_{ext}}{\gamma - 1},$$

which forms a set of coupled equations. We solve them by replacing u_{ext} in the first equation by its expression obtained from the second equation. We are left with a quadratic equation in c_{ext} to solve, that has two solutions. The physical root is the largest one (positive). We then deduce u_{ext} from R^- value.

External pressure and temperature are then computed from

$$p_{ext} = P_{tot} \left(1 + \frac{\gamma - 1}{2} \frac{u_{ext}^2}{c_{ext}^2}\right), \quad T_{ext} = T_{tot} \left(\frac{p_{ext}}{P_{tot}}\right)^{\frac{\gamma-1}{\gamma}},$$

and finally, the external state reads

$$W_{ext} = \left(\frac{p_{ext}}{R T_{ext}},\ u_{ext}\mathbf{n},\ p_{ext}\right).$$

Supersonic/Subsonic Outlet

Subsonic and supersonic outflow boundary conditions are used to prescribe a consistent physical outgoing flow with a given pressure. As for inflow boundary conditions we rely on Riemann invariants.

Flux prescription. In the supersonic case, no information should come from outside the domain, so the external state is chosen equal to the inner state. Thus if the outgoing flow is supersonic, no outflow pressure is effectively imposed.

In the subsonic case, outflow pressure is imposed by upwind going sound waves. The downstream traveling Riemann invariant

$$R^+ = \frac{2c}{\gamma - 1} + \mathbf{u} \cdot \mathbf{n},$$

and entropy

$$s = \frac{p}{\rho^\gamma},$$

are computed from inside the domain and assumed constant accros the frontier. As we want to impose a given pressure, we can deduce

$$\rho_{ext} = \left(\frac{p_{ext}}{s}\right)^{\frac{1}{\gamma}}, \quad c_{ext} = \sqrt{\gamma p_{ext}/\rho_{ext}},$$

and the normal velocity from the Riemann invariant

$$u_{ext} = R^+ - \frac{2c_{ext}}{\gamma - 1}.$$

As tangential velocity is advected by the flow and assumed constant, external state is defined as

$$W_{ext} = \left(\rho_{ext},\ u_{ext}\mathbf{n} + \mathbf{u_t},\ p_{ext}\right),$$

with

$$\mathbf{u_t} = \mathbf{u} - (\mathbf{u}\cdot\mathbf{n})\mathbf{n}.$$

Influence stencil. As boundary conditions fluxes are computed element-wise, they involve at most a dependency between a node and its neighbours. The total contribution of boundary terms to the flux of a given node can be summed up as

$$\mathbf{\Gamma}_i = \sum_{F^{bdy} \ni P_i} \Phi^{bdy}_{i,F^{bdy}}(W_i, W_j, W_k), \tag{1.15}$$

where F^{bdy} are the boundary elements containing the vertex P_i.

No-Slip residual aero-coefficients computation

The post processing of aero-coefficients on bodies requires the computation of the boundary stress tensor. Part of it is due to the pressure and thus directly to the value of the computed field while the viscous contribution involves gradients of the velocity field. Yet, gradients are known to converge slowlier than the primary field and even slowlier near boundaries. Hence, computing the boundary stress tensor by simply differentiating the numerical solution yields a rather poor result compared to the actual precision of the solution. To circumvent this defect, different "physical" approaches have been proposed, taking advantage of the conservation laws [Gariépy 2013].

In a finite volume solver, the residuals of each variables are given by the integral over the cells of the fluxes. In a stationary case, as the time derivatives are null, the residual of the density, velocity and energy express respectively the mass, momentum and energy conservation laws. In particular, the velocity residual expresses the acceleration of the fluid in the cell ($\mathbf{u}\cdot\nabla\mathbf{u}$), with respect of the sum of the forces applied on the cell (pressure, viscous, ...).

In the case of a no-slip boundary condition (as any Dirichlet), the nodal residuals of the imposed variables are discarded, in our case the velocity residual. As shown in Figure 1.15, this velocity residual is actually a part of the momentum balance: the fluid acceleration and

the forces exerted by the surrounding fluid on the cell have been integrated. Still, physically, the momentum balance in the boundary cell has to be verified. Hence, we can deduce the tangential and normal forces imposed by the body on the fluid in the cell to guaranty the momentum conservation. This is consequently the force exerted by the fluid on the body that we are interested in.

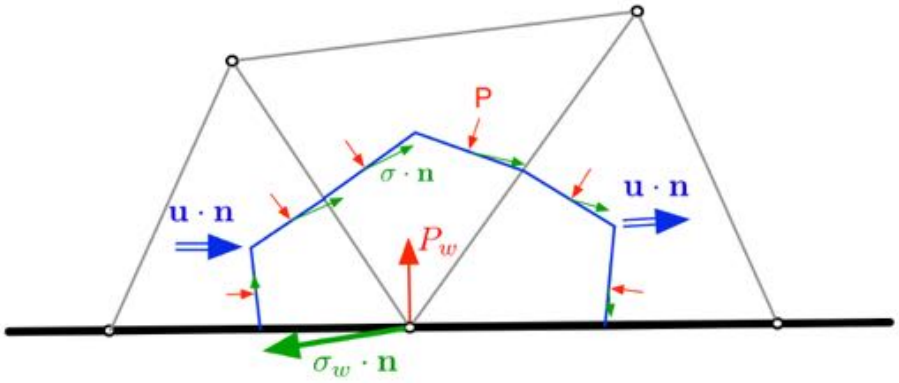


Figure 1.15: Momentum balance in a boundary cell.

1.1.4 Rotating Flow

In order to solve problems in a rotating frame, we have to take into account Coriolis and centrifugal forces and to modify boundary conditions.

Source Term

Coriolis and centrifugal forces take into account the non gallilean character of the current frame and read as

$$F_{Coriolis} = \mathbf{\Omega} \times \mathbf{u},$$

and

$$F_{centrifugal} = \mathbf{\Omega} \times \mathbf{\Omega} \times \mathbf{r} = r\Omega^2 \mathbf{e}_r,$$

where $\mathbf{\Omega}$ is the rotation vector, $\mathbf{r}$ the position vector, r the distance to the rotation axis and $\mathbf{e}_r$ the radial unit vector. To have a $\mathbb{P}^1$-exact integration of these forces, it is sufficient to consider that these forces are constant over the cell leading to the following vertex-wise integration of the source terms:

$$S_{Coriolis} = \int_{\mathcal{C}_i} \mathbf{\Omega} \times \mathbf{u} = C_i \mathbf{\Omega} \times \mathbf{u}_i,$$

and

$$S_{centrifugal} = \int_{\mathcal{C}_i} \mathbf{\Omega} \times \mathbf{\Omega} \times \mathbf{r} = C_i \mathbf{\Omega} \times \mathbf{\Omega} \times \mathbf{r}_i.$$

As the Coriolis force is orthogonal to the velocity, only centrifugal force works and produces an energy source term:

$$S^{energy}_{centrifugal} = \int_{\mathcal{C}_i} \mathbf{\Omega} \times \mathbf{\Omega} \times \mathbf{r} \cdot \mathbf{v} = C_i \mathbf{\Omega} \times \mathbf{\Omega} \times \mathbf{r}_i \cdot \mathbf{v}_i.$$

Boundary conditions corrections

For rotating flows, some changes in boundary conditions must be taken into account.

Slip Walls. In order to have a fixed geometry in the rotating frame, boundaries in the fixed frame have to be cylindrical (they can't depend on the angle around the rotation axis). In that case, no modification is needed for Slip Walls as they do not depend on the tangential velocity.

No-Slip Walls. Solid boundaries rotating at the same velocity as the frame are fixed in the rotating frame and can be treated as usual No-Slip Walls. Fixed solid boundaries in the reference frame are moving in the rotating but can be treated as No-Slip Wall with an imposed velocity $\mathbf{u} = \mathbf{\Omega} \times \mathbf{r}$. They must imperatively be cylindrical with same axis as the rotation (the radius can vary in the axis direction)!

Inflow/Outflow. Pressure inflow/outflow boundary conditions are defined in the reference frame, boundary velocity must thus be transformed into the reference frame in order to compute the new boundary velocity which is then transformed back into the rotating frame.

1.1.5 Time integration

Once the equations have been discretized in space, a set of ordinary differential equations in time is obtained. There are two ways for integrating this set of ODE in time: either using an explicit or an implicit method. All the simulations presented in this thesis were run using an implicit time integration.

Implicit time integration

For an implicit time integration, the semi-discretized RANS system becomes:

$$\frac{|C_i|}{\delta t_i^n}\delta W_i \;=\; -\mathbf{F}_i^{n+1} + \mathbf{S}_i^{n+1} + \mathbf{Q}_i^{n+1} + \mathbf{\Gamma}_i^{n+1}\,, \tag{1.16}$$

where $\delta W_i = W_i^{n+1} - W_i^n$, which, after linearization of the RHS, becomes:

$$\begin{aligned}&\left(\frac{|C_i|}{\delta t_i^n} I_d + \frac{\partial \mathbf{F}_i^n}{\partial W_i} - \frac{\partial \mathbf{S}_i^n}{\partial W_i} - \frac{\partial \mathbf{Q}_i^n}{\partial W_i} - \frac{\partial \mathbf{\Gamma}_i^n}{\partial W_i}\right)\delta W_i \\ &+ \sum_{P_j \in \mathcal{V}^1(i)} \left(\frac{\partial \mathbf{F}_i^n}{\partial W_j} - \frac{\partial \mathbf{S}_i^n}{\partial W_j} - \frac{\partial \mathbf{Q}_i^n}{\partial W_j} - \frac{\partial \mathbf{\Gamma}_i^n}{\partial W_j}\right)\delta W_j \;= -\mathbf{F}_i^n + \mathbf{S}_i^n + \mathbf{Q}_i^n + \mathbf{\Gamma}_i^n\,,\end{aligned}$$

where $P_j \in \mathcal{V}(i)$ is the set of vertices connected to vertex P_i by an edge. The first term of the LHS contributes to the diagonal of the matrix and the second term of the LHS (*i.e.,* the sum) contributes to extra-diagonal terms on line i of the matrix. We now describe each term of the matrix.

The matrix is expressed with the differentiation of first order fluxes (*i.e.* the limiters and MUSCL extrapolation are not taken into account), which enhances stability and is cheaper to compute. This acts as an approximated Jacobian but it is independent of the residual and thus does not affect the spatial and/or time order of the scheme.

Inviscid flux Jacobian. We recall that $\mathbf{F}_i^{n+1} = \sum_{P_j \in \mathcal{V}(i)} \Phi_{ij}^{HLLC}(W_{ij}^{n+1}, W_{ji}^{n+1}, \mathbf{n}_{ij})$, where W_{ij}^{n+1} and W_{ji}^{n+1} are obtained, for the second-order scheme, through MUSCL extrapolation and gradients limitations from the stencil of neighbors. This involves strong non-linear dependencies to these vertices. In order to improve the robustness of the solver, the linearization of the second-order inviscid fluxes is approximated by the linearization of the first-order inviscid fluxes $\hat{\mathbf{F}}_i^{n+1} = \sum_{P_j \in \mathcal{V}(i)} \Phi_{ij}^{HLLC}(W_i^{n+1}, W_j^{n+1}, \mathbf{n}_{ij})$. The linearization of the convective flux term thus reads for first- and second-order schemes:

$$\begin{aligned}
\Phi_{ij}^{HLLC}(W_i^{n+1}, W_j^{n+1}, \mathbf{n}_{ij}) &= \Phi_{ij}^{HLLC}(W_i^{n}, W_j^{n}, \mathbf{n}_{ij}) \\
&+ \left.\frac{\partial \Phi_{ij}^{HLLC}(W_i^{n}, W_j^{n}, \mathbf{n}_{ij})}{\partial W_i}\delta W_i\right\} \text{(A)} \\
&+ \left.\frac{\partial \Phi_{ij}^{HLLC}(W_i^{n}, W_j^{n}, \mathbf{n}_{ij})}{\partial W_j}\delta W_j\right\} \text{(B)}\,.
\end{aligned}$$

Term (A) contributes to matrix diagonal $D(i,i)$ and Term (B) contributes to matrix upper part $U(i,j)$ (here we assume that $i < j$). As $\Phi_{ji}^{HLLC} = -\Phi_{ij}^{HLLC}$, minus Term (B) contributes to matrix diagonal $D(j,j)$ and minus Term (A) contributes to matrix lower part $L(j,i)$.

Viscous flux Jacobian. Let $K = (P_i, P_j, P_k, P_l)$, we now linearize the viscous flux terms $\mathbf{S}_i^n$:

$$\mathbf{S}_i^n = -\sum_{K \ni P_i} \Phi_{i,K}^{visc}(W_i^{n+1}, W_j^{n+1}, W_k^{n+1}, W_l^{n+1}).$$

It reads:

$$\begin{aligned}
\Phi_{i,K}^{visc}(W_i^{n+1}, W_j^{n+1}, W_k^{n+1}, W_l^{n+1}) &= \Phi_{i,K}^{visc}(W_i^{n}, W_j^{n}, W_k^{n}, W_l^{n}) \\
&+ \left.\frac{\partial \Phi_{i,K}^{visc}}{\partial W_i}(W_i^{n}, W_j^{n}, W_k^{n}, W_l^{n})\delta W_i\right\} \text{(A)} \\
&+ \left.\frac{\partial \Phi_{i,K}^{visc}}{\partial W_j}(W_i^{n}, W_j^{n}, W_k^{n}, W_l^{n})\delta W_j\right\} \text{(B)} \\
&+ \left.\frac{\partial \Phi_{i,K}^{visc}}{\partial W_k}(W_i^{n}, W_j^{n}, W_k^{n}, W_l^{n})\delta W_k\right\} \text{(C)} \\
&+ \left.\frac{\partial \Phi_{i,K}^{visc}}{\partial W_l}(W_i^{n}, W_j^{n}, W_k^{n}, W_l^{n})\delta W_l\right\} \text{(D)}\,.
\end{aligned}$$

Term (A) contributes to the matrix diagonal, while Terms (B), (C), (D) contribute to the matrix extradiagonal $L(i,j)$ (resp. $L(i,k)$, $L(i,l)$) if $j < i$ or $U(i,j)$ (resp. $U(i,k)$, $U(i,l)$) if $i < j$.

Boundary conditions Jacobian. In `Wolf` the boundary conditions are fully differentiated. The linearization of the boundary conditions term reads:

$$\Phi_{Fac}^{bc}(W_i^{n+1}, \mathbf{n}_{Fac}) = \Phi_{Fac}^{bc}(W_i^{n}, \mathbf{n}_{Fac}) + \frac{\Phi_{Fac}^{bc}}{\partial W_i}(W_i^{n}, \mathbf{n}_{Fac})\delta W_i\,,$$

which contributes to the matrix diagonal. Boundary conditions terms are given in Section 1.1.3.

For no-slip boundary conditions nothing is done as the flux is constant, while for slip boundary conditions we use the differentiation of the approximate Roe solver. For subsonic/supersonic inflow/outflow boundary conditions, all cases are considered separately to provide the proper differentiation.

Dirichlet (enforced variables) boundary conditions, such as the velocity in no-slip boundary conditions are enforced in the matrix. To avoid renumbering and matrix reshaping, we do not separate imposed nodes from degrees of freedom. Instead, the nodal value of the variable is enforced, the corresponding line in the Jacobian is set to identity and the residual of the variable is set to zero so that the linear system looks like

$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ \dots & \dots & 0 & 1 & 0 \\ * & * & * & * & * \end{bmatrix}}_{\frac{\partial R}{\partial W}} \cdot \underbrace{\begin{bmatrix} * \\ * \\ * \\ \delta W_i^{\mathbf{u}} = 0 \\ * \end{bmatrix}}_{\delta W} = \underbrace{\begin{bmatrix} * \\ * \\ * \\ 0 \\ * \end{bmatrix}}_{R} .$$

Source terms Jacobians. The source terms are the sum of production ($\mathcal{P}$), destruction ($\mathcal{D}$) and diffusion terms ($\mathcal{V}$) in the Spalart Allmaras model, which only contribute to the diagonal:

$$\frac{\partial \mathbf{Q}_i^n}{\partial \tilde{\nu}_i} = \frac{\partial \mathcal{P}_i^n}{\partial \tilde{\nu}_i} + \frac{\partial \mathcal{D}_i^n}{\partial \tilde{\nu}_i} + \frac{\partial \mathcal{V}_j^n}{\partial \tilde{\nu}_i} .$$

We chose a full linearization of these terms. The detail of these terms are given in Section 2.2.1.

1.2 Linear System Resolution

The linearized system obtained in the previous Section is written in vector form:

$$\mathbf{A}^n \, \delta W^n = \mathbf{R}^n \tag{1.17}$$

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

with $\frac{\partial \hat{\mathbf{R}}^n}{\partial \mathbf{W}}$ the approximate Jacobian of the residual vector. This linear system is solved at each flow solver iteration. In practice, we ask the user to provide a maximal number k_{max} of iterations and a targetted order of magnitude by which the residual of the system must be decreased. The iteration is stopped when this targetted residual is reached.

1.2.1 Standard linear solvers

To solve the linear system, we follow the SGS approach based on Lower-Upper Symmetric Gauss-Seidel (LU-SGS) implicit solver initially introduced by Jameson [Jameson 1987] and fully developed by Sharov et al. and Luo et al. [Luo 1998, Luo 2001, Sharov 1997, Sharov 2000].

The SGS is very attractive because it uses an edge-based data structure which can be efficiently parallelized with p-threads [Alauzet 2009b, Sharov 2000]. From our experience, we have made the following - crucial - choices to solve the compressible Navier-Stokes equations.

Converging the linear system is important for the global convergence of the Navier-Stokes non-linear problem. Hence, an iterative method, such as SGS, is required. Usually, the SGS method iterates until the residual of the linear system is reduced by one or two orders of magnitude (i.e. 0.1 or 0.01).

The choice of the renumbering also impacts strongly the convergence of the linear system. While Hilbert-type (space filling curve) renumbering is very efficient for cache misses and memory contention [Alauzet 2009b, Sharov 2000], Breadth-First Search (BFS) renumbering proves to be more effective for the convergence of the implicit method and the overall efficiency.

Luo et al. [Luo 1998, Luo 2001, Sharov 2000] proposed using a simplified flux function - a Rusanov approximate Riemann solver for the convective terms and the operator spectral radius for the viscous terms - to compute Jacobians while keeping the complex flux function for the right-hand side term. However, we observed that this modification slows down the convergence of the whole process. We found it very advantageous to fully differentiate the HLLC approximate Riemann solver [Batten 1997], the FEM viscous terms and the Spalart-Allmaras source terms as presented in the previous section.

To achieve high efficiency, automation, and robustness in the resolution of the linear system of algebraic equations to steady-state, it is mandatory to have a clever strategy to specify the time step. This is done by coupling a local under-relaxation coefficient, and local CFL.

Block matrix representation

The system is written as the sum of the upper triangular part, the lower triangular part and the diagonal part:

$$(\mathbf{L} + \mathbf{D} + \mathbf{U})\,\delta\mathbf{W}^n = \mathbf{R}^n.$$

In order to optimize the convergence of the system we take advantage of the block structure of the matrices. Namely, the residual of a given variable at a node depends on the variables of the node considered and on the variables of its neighbours. These terms form blocks of known size and position, it is thus advantageous to compute matrix operations (product, inversion, ...) with block matrix operations. In particular, in many approaches, the inversion of the matrix involves the inversion of its diagonal. Using the inverse of the block diagonal instead of the simple scalar diagonal adds a lot of information to the system. In the following we will thus represent the Jacobian matrix as:

$$\begin{bmatrix} \dots & \dots & \dots & \dots & \dots \\ \dots & D_{ii} & \mathbf{0} & U_{ij} & \dots \\ \dots & \mathbf{0} & \ddots & \mathbf{0} & \dots \\ \dots & L_{ji} & \mathbf{0} & D_{jj} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix},$$

where in 2D

$$D_{ii} = \begin{bmatrix} \frac{\partial \hat{R}_i^\rho}{\partial \rho_i} & \frac{\partial \hat{R}_i^\rho}{\partial \rho u_i} & \frac{\partial \hat{R}_i^\rho}{\partial \rho v_j} & \frac{\partial \hat{R}_i^\rho}{\partial \rho e_i} \\ \frac{\partial \hat{R}_i^{\rho u}}{\partial \rho_i} & \frac{\partial \hat{R}_i^{\rho u}}{\partial \rho u_i} & \frac{\partial \hat{R}_i^{\rho u}}{\partial \rho v_j} & \frac{\partial \hat{R}_i^{\rho u}}{\partial \rho e_i} \\ \frac{\partial \hat{R}_i^{\rho v}}{\partial \rho_i} & \frac{\partial \hat{R}_i^{\rho v}}{\partial \rho u_i} & \frac{\partial \hat{R}_i^{\rho v}}{\partial \rho v_j} & \frac{\partial \hat{R}_i^{\rho v}}{\partial \rho e_i} \\ \frac{\partial \hat{R}_i^{\rho e}}{\partial \rho_i} & \frac{\partial \hat{R}_i^{\rho e}}{\partial \rho u_i} & \frac{\partial \hat{R}_i^{\rho e}}{\partial \rho v_j} & \frac{\partial \hat{R}_i^{\rho e}}{\partial \rho e_i} \end{bmatrix},$$

$$L_{ji} = \begin{bmatrix} \frac{\partial \hat{R}_j^\rho}{\partial \rho_i} & \frac{\partial \hat{R}_j^\rho}{\partial \rho u_i} & \frac{\partial \hat{R}_j^\rho}{\partial \rho v_i} & \frac{\partial \hat{R}_j^\rho}{\partial \rho e_i} \\ \frac{\partial \hat{R}_j^{\rho u}}{\partial \rho_i} & \frac{\partial \hat{R}_j^{\rho u}}{\partial \rho u_i} & \frac{\partial \hat{R}_j^{\rho u}}{\partial \rho v_i} & \frac{\partial \hat{R}_j^{\rho u}}{\partial \rho e_i} \\ \frac{\partial \hat{R}_j^{\rho v}}{\partial \rho_i} & \frac{\partial \hat{R}_j^{\rho v}}{\partial \rho u_i} & \frac{\partial \hat{R}_j^{\rho v}}{\partial \rho v_i} & \frac{\partial \hat{R}_j^{\rho v}}{\partial \rho e_i} \\ \frac{\partial \hat{R}_j^{\rho e}}{\partial \rho_i} & \frac{\partial \hat{R}_j^{\rho e}}{\partial \rho u_i} & \frac{\partial \hat{R}_j^{\rho e}}{\partial \rho v_i} & \frac{\partial \hat{R}_j^{\rho e}}{\partial \rho e_i} \end{bmatrix} \text{ and } U_{ij} = \begin{bmatrix} \frac{\partial \hat{R}_i^\rho}{\partial \rho_j} & \frac{\partial \hat{R}_i^\rho}{\partial \rho u_j} & \frac{\partial \hat{R}_i^\rho}{\partial \rho v_j} & \frac{\partial \hat{R}_i^\rho}{\partial \rho e_j} \\ \frac{\partial \hat{R}_i^{\rho u}}{\partial \rho_j} & \frac{\partial \hat{R}_i^{\rho u}}{\partial \rho u_j} & \frac{\partial \hat{R}_i^{\rho u}}{\partial \rho v_j} & \frac{\partial \hat{R}_i^{\rho u}}{\partial \rho e_j} \\ \frac{\partial \hat{R}_i^{\rho v}}{\partial \rho_j} & \frac{\partial \hat{R}_i^{\rho v}}{\partial \rho u_j} & \frac{\partial \hat{R}_i^{\rho v}}{\partial \rho v_j} & \frac{\partial \hat{R}_i^{\rho v}}{\partial \rho e_j} \\ \frac{\partial \hat{R}_i^{\rho e}}{\partial \rho_j} & \frac{\partial \hat{R}_i^{\rho e}}{\partial \rho u_j} & \frac{\partial \hat{R}_i^{\rho e}}{\partial \rho v_j} & \frac{\partial \hat{R}_i^{\rho e}}{\partial \rho e_j} \end{bmatrix}$$

LU-SGS Resolution

The linear system can be re-written:

$$(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^n = \mathbf{R}^n + (\mathbf{L}\mathbf{D}^{-1}\mathbf{U})\,\delta\mathbf{W}^n$$

The following approximate system is used:

$$(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^n = \mathbf{R}^n\,.$$

As $\mathbf{D}+\mathbf{L}$ and $\mathbf{D}+\mathbf{U}$ are lower (resp. upper) triangular matrix, $(\mathbf{D}+\mathbf{L})\mathbf{X} = \mathbf{Y}$ and $(\mathbf{D}+\mathbf{U})\mathbf{X} = \mathbf{Y}$ can be solved exactly with a forward (resp. backward) substitution sweep. Matrix $(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{U})$ can thus inverted in two sweeps which correspond to the LU-SGS approximate factorization:

$$\begin{aligned} \text{Forward sweep:} \quad (\mathbf{D}+\mathbf{L})\,\delta\mathbf{W}^* &= \mathbf{R} \\ \text{Backward sweep:} \quad (\mathbf{D}+\mathbf{U})\,\delta\mathbf{W} &= \mathbf{D}\,\delta\mathbf{W}^*\,. \end{aligned}$$

These sweeps are written point-wise with block matrix operations:

$$\begin{aligned} \delta W_i^* &= D_{ii}^{-1}\Big(R_i - \sum_{j<i} L_{ij}\delta W_j^*\Big) \\ \delta W_i &= \delta W_i^* - D_{ii}^{-1}\sum_{j>i} U_{ij}\delta W_j^*\,. \end{aligned}$$

where $\mathcal{L}(i)$ (resp. $\mathcal{U}(i)$) is the set of vertices with an index lower (resp. upper) than i. These operations being written block-wise, $\delta W_i = (\delta\rho_i, \delta\rho u_i, \delta\rho v_i, \delta\rho e_i, \delta\rho\tilde{\nu}_i)$ is actually the vector of the full solution variation at node i. The lower and upper parts can be stored or not (i.e., matrix-free) as a choice between efficiency or memory requirements. If matrix free option is chosen, the blocks U_{ij} and L_{ij} are recomputed every time they are needed. Though, the produce $L_{ij}\delta W_j^*$ comes from the differentiation of the fluxes accros the edge e_{ij}, it can thus also be replaced by $\Phi(W_i^n, W_j^n + \delta W_j^*)$.

Finally, we can see that the remaining error is directly related to $\mathbf{L}\mathbf{D}^{-1}\mathbf{U}$ which depends on the matrix conditioning. This points illustrates the dependency of the method to the renumbering.

Symetric Gauss Siedel Resolution (SGS)

In the SGS relaxation, we use a fixed point iterative approach, alternatively treating the matrix as lower or upper triangular to invert the system. Namely, we build a list of $\delta\mathbf{W}^{n,k}$ so that $A^n\delta\mathbf{W}^{n,k} \to R^n$. As the system $(\mathbf{L}+\mathbf{D}+\mathbf{U})\delta\mathbf{W}^{n,k} = R^n$ cannot be solved directly, the idea is to fix a part of the system so that we are left with an upper (resp. lower) triangular system that can be solved with a backward (resp. forward) substitution sweep:

$$(\mathbf{D}+\mathbf{L})\,\delta\mathbf{W}^{n,k+1} + \underbrace{\mathbf{U}\,\delta\mathbf{W}^{n,k}}_{\text{fixed part}} = \mathbf{R}^n. \tag{1.18}$$

It is clear that if this series converges, it is toward the solution of $A^n\delta\mathbf{W} = R^n$. The convergence of the series can be stated as

$$\begin{aligned}
(\mathbf{D}+\mathbf{L})\,\delta\mathbf{W}^{n,k+1} + \mathbf{U}\,\delta\mathbf{W}^{n,k} &= (\mathbf{D}+\mathbf{L}+\mathbf{U})\delta\mathbf{W}, \\
(\mathbf{D}+\mathbf{L})\,\Big(\delta\mathbf{W}^{n,k+1} - \delta\mathbf{W}\Big) &= -(\mathbf{U})\,\Big(\delta\mathbf{W}^{n,k} - \delta\mathbf{W}\Big), \\
\Big(\delta\mathbf{W}^{n,k+1} - \delta\mathbf{W}\Big) &= -(\mathbf{D}+\mathbf{L})^{-1}(\mathbf{U})\,\Big(\delta\mathbf{W}^{n,k} - \delta\mathbf{W}\Big).
\end{aligned}$$

The convergence of Equation (1.18) is thus directly related to the largest eigenvalue of $(\mathbf{D}+\mathbf{L})^{-1}(\mathbf{U})$. Intuitively, we would say that the more we remove from $\mathbf{U}$ and add in $(\mathbf{D}+\mathbf{L})$, the smaller $(\mathbf{D}+\mathbf{L})^{-1}(\mathbf{U})$ will be. Hence the improvement obtained by treating the diagonal as blocks. The SGS is obtained with a two steps fixed point, alternatively fixing $\mathbf{U}\,\delta\mathbf{W}^{n,k}$ as in Equation (1.18) and then $\mathbf{L}\,\delta\mathbf{W}^{n,k}$ the same way.

We first zero the unknown: $\delta W^0 = 0$. Then, k_{max} sub-iterations are made using forward and backward sweeps:

$$\begin{aligned}
(\mathbf{D}+\mathbf{L})\,\delta\mathbf{W}^{n,k+1/2} &= \mathbf{R}^n - \mathbf{U}\,\delta\mathbf{W}^{n,k} \\
(\mathbf{D}+\mathbf{U})\,\delta\mathbf{W}^{n,k+1} &= \mathbf{R}^n - \mathbf{L}\,\delta\mathbf{W}^{n,k+1/2}.
\end{aligned}$$

or rewritten point-wise with block matrix operations:

$$\begin{aligned}
\delta W_i^{n,k+1/2} &= D_{ii}^{-1}\Big(R_i - \sum_{j\in\mathcal{L}(i)} L_{ij}\delta W_j^{n,k+1/2} - \sum_{j\in\mathcal{U}(i)} U_{ij}\delta W_j^{n,k}\Big) \\
\delta W_i^{n,k+1} &= D_{ii}^{-1}\Big(R_i - \sum_{j\in\mathcal{U}(i)} U_{ij}\delta W_j^{n,k+1} - \sum_{j\in\mathcal{L}(i)} L_{ij}\delta W_j^{n,k+1/2}\Big).
\end{aligned}$$

For one sub-iteration, the SGS method is equivalent to the LU-SGS method.

In some cases, iterating the SGS method does not help to decrease the residual of the linear system. This can be due to the stiffness of the problem or simply because the linear system has already converged by its maximum order of magnitude, in which case the targeted residual was not set properly. In order to avoid costly SGS iterations which will not impact the final solution, we stop iterating when a stagnation of the residual of the linear system is detected.

Let Res_i be the residual of the linear system at the current SGS iteration i and Res_{i-1} the residual at the previous one. We consider that the residual is stagnating from iteration $i-1$ to i, if

$$|Res_i - Res_{i-1}| \quad < \quad \epsilon Res_{i-1}\,,$$

where we use $\epsilon = 10^{-3}$. We choose to stop iterating the method once we detect three stagnations.

1.2.2 SGS - Line solver

In some applications, the information is known to propagate in a privileged direction, for instance in the normal direction in a boundary layer. Additionally, the mesh may also exhibit a privileged structure in these directions, a structured boundary layer in our case. The aim of line solvers is to take advantage of this structure in order to improve the inverse of the system.

When using the SGS method, at each sweep, the new value of the solution on each vertex depends on the updated values of the preceding vertices, so that the information is propagated alongside the list of vertices. However, the information fades during the advancing process. This is why the efficiency of the method strongly depends on the mesh renumbering: two vertices are strongly correlated in the SGS if their ID number is close, while they are physically strongly correlated if they are geometrically close in the mesh. For this reason, a BFS renumbering is more efficient than a Hilbert renumbering as it tends to cluster closely vertices that are geometrically close in the mesh. However, in 3D, it is clear that a line can hardly group points together.

The idea of line solver is to renumber the mesh so that geometrical lines appear. If these lines do not close on themselves (if each node is connected uniquely to its preceding and following nodes), a tridiagonal structure appears in the matrix. This tridiagonal can be inverted exactly, instantaneously propagating the information accros each line, further improving the inversion of the system. This approach is expected to be particularly efficient in pseudo 1D problems like boundary layers where the 3D linear system is solved as the 1D linear system which is tridiagonal.

System inversion:

The standard line solver approach would be to split the matrix $\mathbf{A}$ in a tridiagonal matrix $\mathbf{T}$ obtained by concatenating the tridiagonal block of each line and in an upper and low matrix $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{L}}$. The iterative system is then solved in a block Jacobi way [Lee 2011, Soto 2003, Diosady 2009, Mavriplis 1998]:

$$\mathbf{T}\delta W^{n,k+1} = R^n - (\tilde{\mathbf{L}} + \tilde{\mathbf{U}})\,\delta W^{n,k}.$$

However, this requires to have splited the whole domain in lines. This is fairly easy to do with structured grids but not in our case as we use pseudo-structured or unstructured grids. In practice, we will generate lines only in the structured parts of the mesh, namely the boundary layers and treat the rest of the domain as unstructured.

In order to be compliant with standard SGS approach we propose to treat the tridiagonal matrices obtained from the generation of lines as blocks in the SGS sweeps. To do so, we need

to rewrite the SGS sweep linewise. In the forward sweep, the value of $\delta W^{k+1/2}$ at each node is computed successively starting with node 1 as

$$\delta W_i^{k+1/2} = D_{ii}^{-1}\Big(R_i - \underbrace{\sum_{j<i} L_{ij}\delta W_j^{k+1/2}}_{\text{left part of the line}} - \underbrace{\sum_{j>i} U_{ij}\delta W_j^{k}}_{\text{right part of the line}}\Big).$$

This expression is already written in a block line way, $\mathbf{D}_{ii}$ is a square matrix of the size of the number of variables, but it can be written with an arbitrarily large matrix as long as we are able to compute $\mathbf{D}_{ii}^{-1}$. The idea of SGS-line solver is to consider the block line around a tridiagonal that is invertible.

Figure 1.16 shows the representation of the Jacobian matrix computed on a mesh composed of two structured lines of five nodes and an unstructured part in a problem with four variables. Each node contributes through block matrices of size 4×4, represented here in brackets, each red dot being a scalar. $\mathcal{T}_1$ and $\mathcal{T}_2$ shows the tridiagonals formed by each line, each vertex is connected to the next and previous one. This tridiagonal structure is only due to the renumbering, in our case, the two lines are next to each other so that the nodes of line 1 are connected to the nodes of line 2 forming the "upper" and "lower" parts $\mathcal{U}_1$ and $\mathcal{L}_2$. The unstructured part of the matrix is splitted in a block diagonal and upper and lower rectangular blocks.

For this matrix, the first SGS-sweep is performed as:

1. We first treat the first block line of $\mathcal{T}_1$, $\mathcal{U}_1$.
   - Starting from δW^k we compute the block product $\mathcal{U}_1 \cdot \delta W^k_{j=6..20}$, where $\delta W^k_{j=6..20}$ is the vector formed of the 60 last scalar components of δW^k.
   - We compute and update $\delta W^{k+1/2}_{j=1..5} = \mathcal{T}_1^{-1}(R^n_{j=1..5} - \mathcal{U}_1 \cdot \delta W^k_{j=6..20})$. The inversion of $\mathcal{T}_1^{-1}$ is done with the Thomas algorithm.
2. We treat the second block line.
   - We compute $\mathcal{L}_2 \cdot \delta W^{k+1/2}_{j=1..5}$ from the **updated values** computed at the previous step.
   - We compute $\mathcal{U}_2 \cdot \delta W^k_{j=11..20}$
   - We compute and update $\delta W^{k+1/2}_{j=6..10} = \mathcal{T}_1^{-1}(R^n_{j=6..10} - \mathcal{U}_2 \cdot \delta W^k_{j=11..20} - \mathcal{L}_2 \cdot \delta W^{k+1/2}_{j=1..5})$
3. We then keep going with the unstructured part with the standard block SGS.
   - We compute $\mathcal{L}_3 \cdot \delta W^{k+1/2}_{j=1..10}$ from the **updated values** computed at the previous steps.
   - We compute $\mathcal{U}_3 \cdot \delta W^k_{j=12..20}$
   - We compute and update $\delta W^{k+1/2}_{j=11} = \mathcal{D}_3^{-1}(R^n_{j=11} - \mathcal{U}_3 \cdot \delta W^k_{j=12..20} - \mathcal{L}_3 \cdot \delta W^{k+1/2}_{j=1..10})$
4. ...

The second backward sweep is done the same way, starting with the unstructured part.

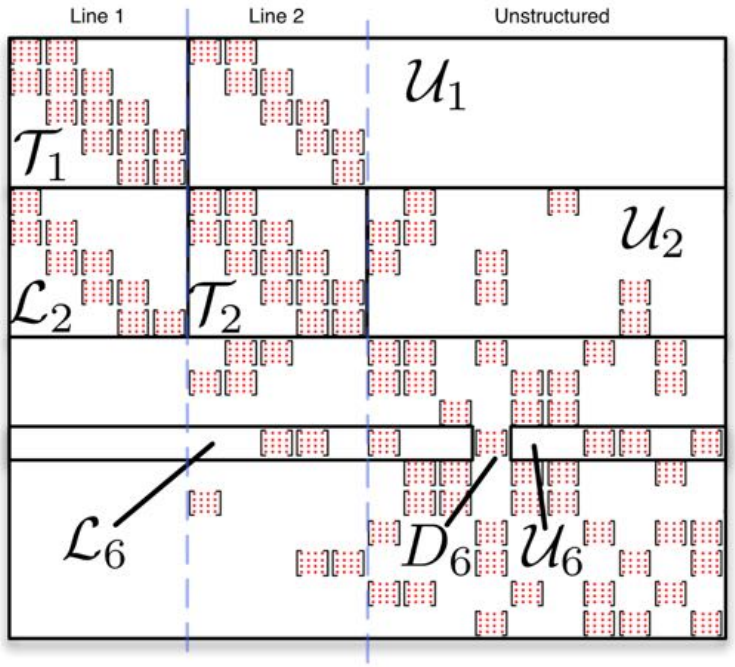


Figure 1.16: Sparse matrix obtained with a line renumbering.

Thomas Algorithm:

The Thomas Algorithm is a LU decomposition of a tridiagonal matrix. Due to the particular form of the matrix, the system can be factorized in a forward sweep. In the LU form, it can be then inverted with a forward and backward substitution. We assume the tridiagonal system to be of the form

$$\begin{bmatrix} A_1 & B_1 & & \\ C_2 & A_2 & B_2 & \\ & \ddots & \ddots & \ddots \\ & & C_n & A_n \end{bmatrix} = \begin{bmatrix} Id & & & \\ L_2 & Id & & \\ & \ddots & \ddots & \\ & & L_n & Id \end{bmatrix} \cdot \begin{bmatrix} D_1 & B_1 & & \\ & D_2 & B_2 & \\ & & \ddots & \ddots \\ & & & D_n \end{bmatrix},$$

where $A_i, B_i, C_i, D_i, L_i, U_i$ are square matrices. The system can be factorized with Algorithm 1. The system is then solved with a forward backward substitution noting that

$$y = Mx = LUx = L(\underbrace{Ux}_{x'})$$

with Algorithm 2.

SGS-line solver: lines generation

In the previous section we covered the resolution of a matrix with a proper renumbering of the vertices in order to exhibit a line structure. We present here different approaches in order to create this renumbering.

Algorithm 1 LU factorisation

1: $D_1 = A_1$
2: **for** $i = 1..n$ **do**
3: $\quad L_i = C_i D_{i-1}^{-1}$
4: $\quad D_i = A_i - L_i B_{i-1}$
5: **end for**

Algorithm 2 Forward backward substitution

1: $x'_1 = y_1$
2: **for** $i = 1..n$ **do**
3: $\quad x'_i = y_i - L_i x'_{i-1}$
4: **end for**
5: $x_n = D_n^{-1} x'_n$
6: **for** $i = n..1$ **do**
7: $\quad x_n = D_n^{-1}(x'_n - B_i x_{i+1})$
8: **end for**

Frontal approach: As we want in particular to resolve structured boundary layer meshes, the straight forwardest way to generate lines is to use a frontal approach, starting from the walls. We start by flagging the vertices on the wall frontiers and generate a list of lines, one for each flagged vertex. We can then grow all the lines together by adding the closest "free" vertex of the last vertex of the line to each line successively. This automatically grows normal lines in structured boundary layers, but will take a wrong direction when the mesh becomes isotropic (in particular close to the trailing edge) as the normal direction is no longer the smallest one. Though, this can be prevented by constraining the angular variation of the line.

Additionally, this approach fixes the number of line to the number of elements on the body or considered boundary. As consequence, this may ignore a lot of phenomenon in the volume that we may want to resolve with lines as well, such as the wake of the body or shocks. A solution to this problem could be to specify sources of lines by hand, which is incompatible with automatic mesh adaptation. Another option would be cycle through all the vertices and grow a line for each unpaired vertex encountered. However as lines are grown individually, the first lines are subject to very few constraint and yield strong constraints on the next ones. This also lets the first ones occupy more space than they should, and on the contrary shortens the last ones. Moreover, the result depends on the order in which lines are grown and thus on the initial ordering of the mesh.

Heap list approach: The major limitation of frontal approaches is the difficulty of generating lines that resolve features in the volume like wake and shocks, both of which are generally finely discretized, either by hand or with automatic mesh adaptation. Hence the idea of using the edge length to create lines, starting by the smallest ones in order to automatically create lines in the refined regions. The edges are first sorted according to their length using a heap list. These edges are then popped one after another and treated as follows.

- If none of the vertices joined by this edge is part of a line, then a new line is created,

composed of these two vertices.

- If any of the vertices of the edge is already in a line but not at one end of a line, the edge is rejected.
- If one of the vertices is at an end of a line then we check if the second vertex is a neighbour any other element of the line. If not, the edge is added to the line.
- If both vertices are at an end of two lines, we check if these two lines are touching each other by any pair of vertices. If not, we merge the two lines, connected by the considered edge.

This method automatically generates lines in the boundary layers and wakes as can be seen in Figure 1.17. However, it still tends to turn in the isotropic region. Additional constraints can be added, to limit the size of the edges considered so that lines are generated only in refined regions or on the maximal angle allowed between two edges as shown in Figure 1.17-(left).

Finally, we used here a geometric criterion (the length of the edges) to generate the lines, but we can use physical criterions, such as the dot product of the edge and the gradient of the velocity. This prioritize small edged aligned with velocity gradients. We can also use a numerical approach and sort the edges with respect to the contribution they yield in the matrix.

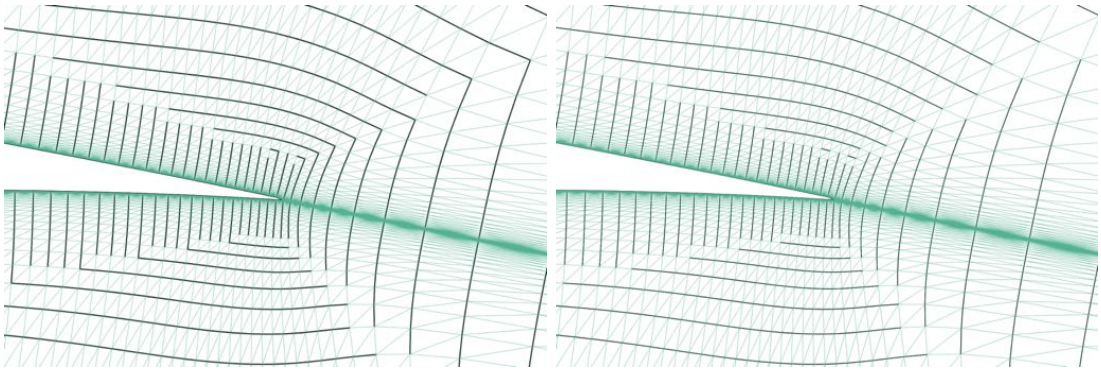


Figure 1.17: Lines generated by a heap list, zoom on the trailing edge of a RAE2822 structured mesh. Without (left) and with (right) angular limitation.

Line BFS: Additionally, in order to further improve the resolution of the system, the lines are reorganized with a Bread First Search (BFS) method. The same approach as for vertices is applied except that two lines are considered to be neighbours if any vertex of the first is connected to any vertex of second. Finally, the overall ordering is as follows.

- The vertices of the lines come first.
- Vertices of a line succeed to each other in order.
- The lines are ordered with a BFS.
- The vertices not included in any line come then, ordered with a BFS.

Flow solver iterations	CFL	SGS		Line-SGS	
		Iterations	CPU	Iterations	CPU
100	0.72	1	9.56	1	9.31
200	5.24	4	20.6	2	18.4
300	38.0	12	37.8	3	29.5
400	275	16	71.2	9	44.2
500	500	49	129	12	65.9
600	500	35	195	10	91.4
700	500	34	250	13	117

Table 1.2: Turbulent transonic RAE2822: Number of SGS/line-SGS iterations required to inverse the linear system with SGS and Line-SGS solvers at some flow solver iterations.

Speedup on structured grid:

We consider here the same turbulent transonic flow around the RAE2822 geometry as in Section 1.1.2 ($M = 0.725$, $\alpha = 2.92°$, $Re = 6.5 \times 10^6$). We use a second order MUSCL extrapolation with fourth order numerical dissipation, Piperno limiter and containment cells. Limiters are frozen at iteration 1 000. We start from a uniform solution, with $CFL = 0.1$. The CFL grows with a geometric factor of 1.02 and is limited to $CFL = 500$. We used a BFS renumbering. The linear system is converged by one order (0.1) with either a pure SGS solver or with a Line-SGS solver.

Figure 1.18 shows the convergence of the density residual during the simulation with respect to the number of iterations and CPU time, using standard SGS and Line-SGS inversion of the linear system. The convergence in term of iterations shows a very similar behaviour, the remaining error in the linear system has very little effect on the solution. This is due to the non-linearity of the problem and the use of an approximate matrix. Converging the linear system by an order of magnitude is thus sufficient. The convergence of the residual with respect to the CPU time shown in Figure 1.18(right) indicates that each iteration is more expensive with a SGS solver. In the facts, the expense of factorizing the tridiagonal system is compensated by the reduction in the number of iterations required to inverse the linear system. Table 1.2 summarizes the number of SGS/Line-SGS iterations required at each step to inverse the linear system and the cumulated CPU time for some steps. The Line-SGS solver requires about 3 times less iterations to inverse the linear system at each step.

Speedup on unstructured adapted mesh:

We then generate a pseudo-structured adapted mesh with metric aligned method (feature based adaptation on the Mach field) on the same RAE2822 case. Figure 1.19 shows the corresponding adapted mesh in which the shock, the boundary layer and the wake have been refined and the lines generated. We can see the aligned mesh generated in the boundary layer (Figure 1.19-(bottom left)) and the shock (Figure 1.19-(bottom right)). The lines generated successfully

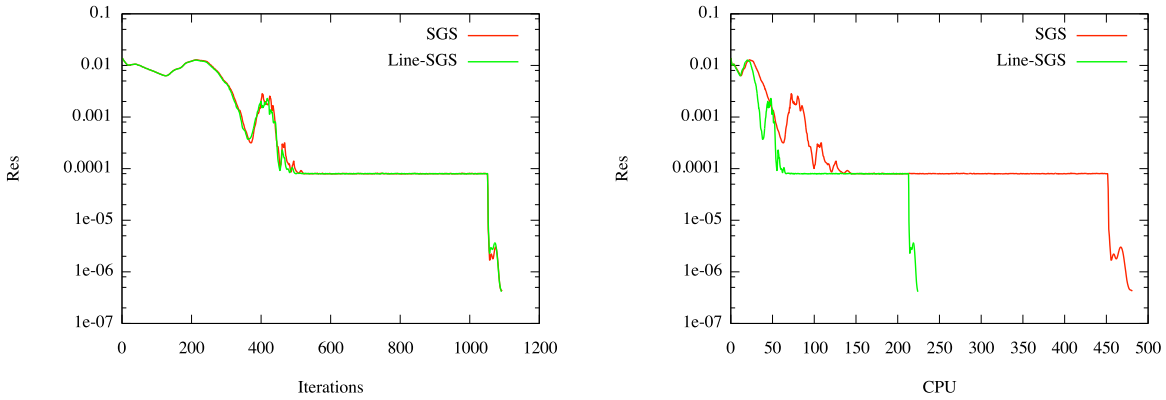


Figure 1.18: Turbulent transonic RAE2822. Convergence of the density residual with respect to the number of flow solver iterations (left) and CPU time (right) using SGS linear solver (red) and Line-SGS linear solver (green) on structured grid.

follow the shape of the mesh in these regions but due to mesh size variations or imperfections, they do not extend throughout the boundary layer as with the structured mesh. As consequence, the use of a Line-SGS solver is less efficient than on structured grid but still fasten computations as can be seen in Figure 1.20.

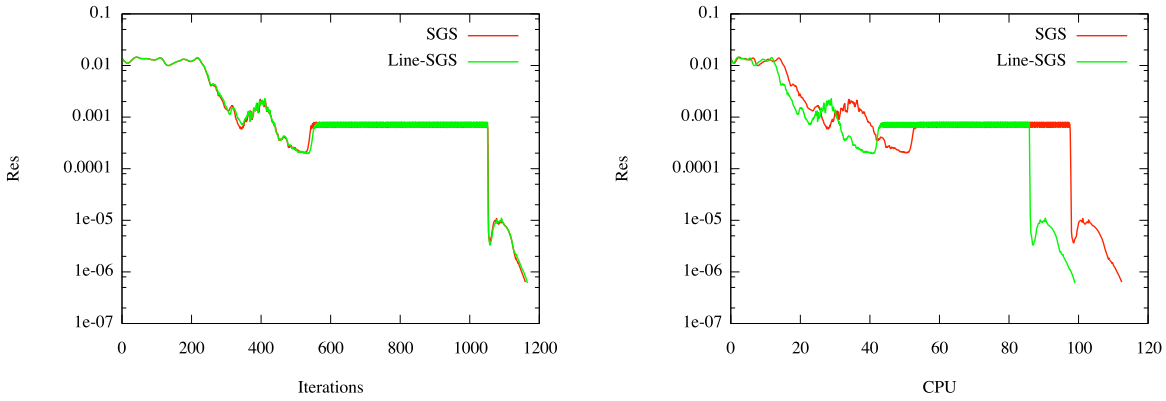


Figure 1.20: Turbulent transonic RAE2822. Convergence of the density residual with respect to the number of flow solver iterations (left) and CPU time (right) using SGS linear solver (red) and Line-SGS linear solver (green) on pseudo-structured adapted mesh.

1.2.3 Multigrid solver

Multigrid solvers are another family of pre-conditioners for the resolution of the linear system. The theory of multigrid is based on linear cases for which solving the linear system is equivalent to solve the global system. Both are thus usually mixed but in our case, it only affects the inversion of the Jacobian for the implicit time integration. Hence, if the linear system is converged down to a low residual, it has no effect on the physical solution, it can only fasten

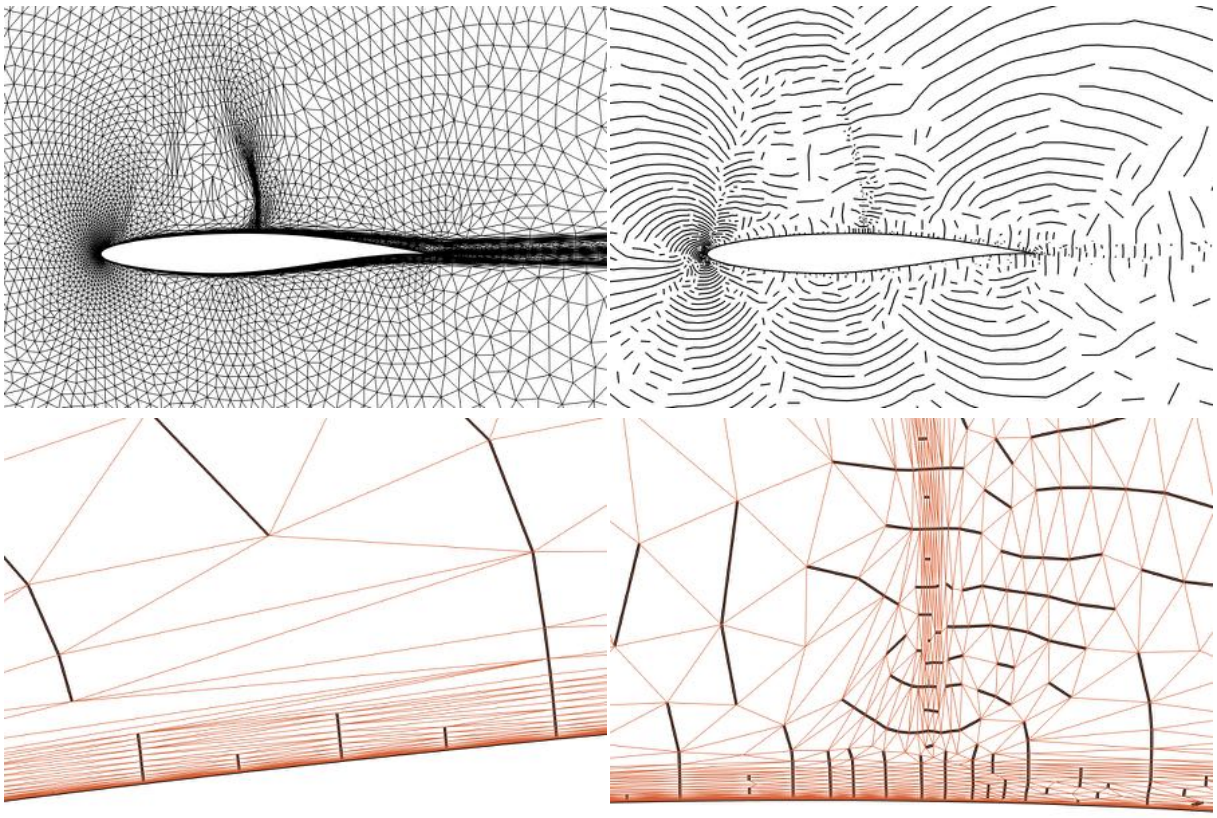


Figure 1.19: Turbulent transonic RAE2822: pseudo-structured adapted mesh (top left) and the lines generated with a geometric approach (top right). Zoom in the boundary layer (bottom left) and the shock-boundary layer interaction (bottom right).

the resolution of the linear system. However, when the linear system is not converged to a high precision, it may affect the physical resolution.

Numerical context: Iterative linear system resolution methods such as SOR, SSOR, Krylov subspaces, SGS, ... tends to propagate information geometrically accros the mesh. Let us illustrate this phenomenon with a simple 1D case:

$$\frac{\partial^2 u}{\partial x^2} = 0 \text{ on } [0,1], \quad u(0) = 1,\, u(1) = 0,$$

solved with a finite difference method. The Jacobian matrix of this problem on a uniform mesh with a node spacing of h is

$$A = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 \end{bmatrix}. \tag{1.19}$$

If we start with $u_h = 0$ in the domain, the residual is null except for the first node (boundary nodes are excluded) so that we need to solve

$$\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 \end{bmatrix} du = -\frac{1}{h^2}\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{1.20}$$

If we use a simple GMRES to solve this linear system, we will successively look for the best solution in the Krylov spaces $K^k = (r, Ar, A^2r, ..., A^kr)$. But we can see that

$$r = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad Ar = \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad A^2r = \begin{bmatrix} -5 \\ 4 \\ -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad ... \tag{1.21}$$

so that K^k is actually the subspace of $\mathbb{R}^N$ composed of the vectors whose k first components are non-null. Hence, as du is sought in K^k, only its k first component are non-null, and the solution propagates of one vertex at each iteration.

SGS is more efficient to solve this problem as it propagates the information along the whole line at each iteration. But let us compute the first iteration, as $du^0 = [0, 0, ..., 0]$ we can deduce iteratively

$$du^{1/2} = [\frac{1}{2}, \frac{1}{4}, ..., \frac{1}{2^k}, ...].$$

We can see that the information is propagated along the whole domain but fades exponentially. Figure 1.21 shows this propagation on the 10 first SGS iterations on a problem with $N = 50$. We can clearly see the exponentially decreasing du^1 in Figure 1.21-(left).

Figure 1.22-(left) shows the evolution of the nodal residual over these 10 first iterations. We can see the same propagation process, and note that the residual is progressively smoothed by the SGS-solver. Moreover, the initial discontinuous residual decreases very fast (one order of magnitude in one iteration) and then converges is slower as it get smoother. Mathematically it is traditionally shown that if we perform a Fourier analysis of the problem, high frequencies are damped faster than low frequencies. This is illustrated by Figure 1.22-(right), where we plotted the residual normalized by its maximal value every 50 iterations. While the initial discontinuous residual actually contains all frequencies, we can see that the residual is progressively restricted to first and second eigen modes. This is why iterative solvers such as SGS, SOR, SSOR, ... are called "smoothers" in the multigrid context. Still they remain approximate resolution of the matrix.

As the information is propagated through connected neighbours nodes, the idea of multigrids is to use coarser meshes to propagate faster the information and "smooth" the low frequencies

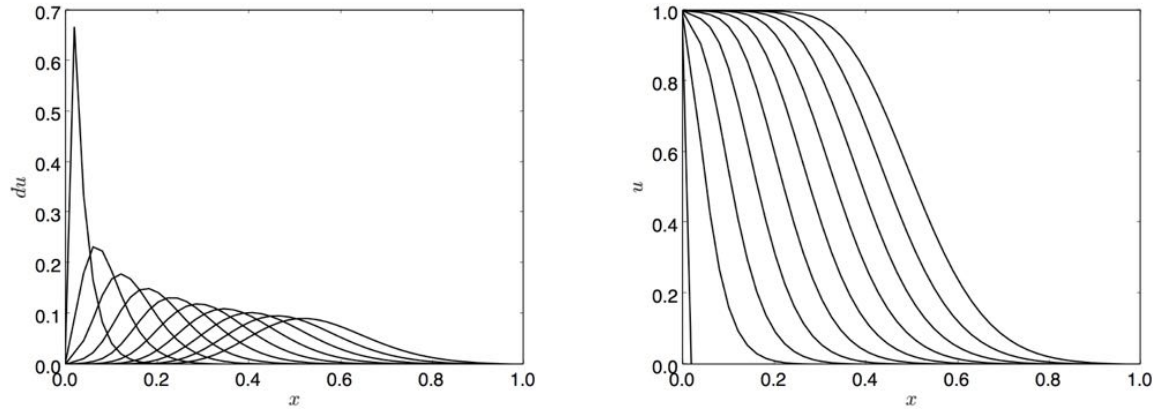


Figure 1.21: 1D laplacian, $N = 50$: propagation of the information in the domain in (left) the solution variation du and (right) the solution u^k with 10 SGS-solver iterations.

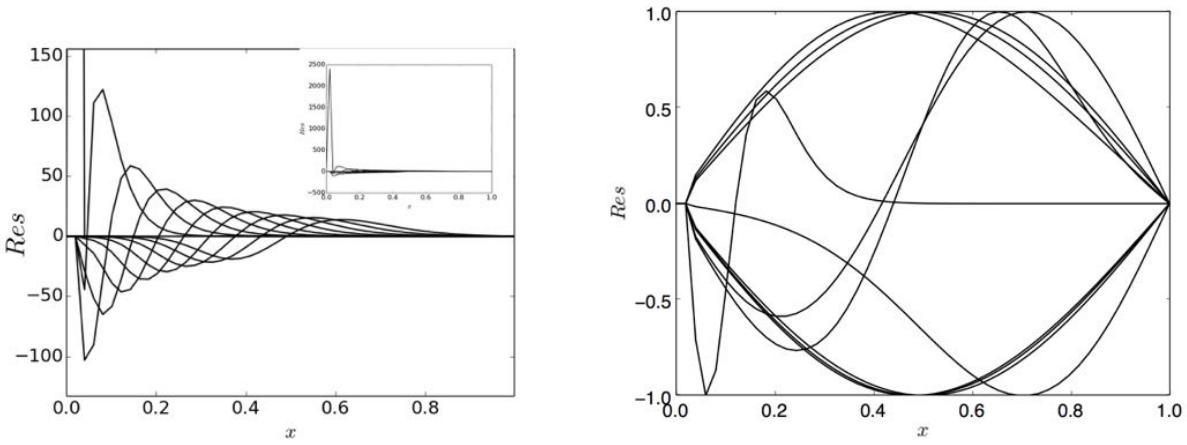


Figure 1.22: 1D laplacian, $N = 50$: (left) evolution of the residual with a SGS-solver on the 10 first iterations and (right) evolution of the residual normalized by its maximal value, with a SGS-solver every 50 iterations.

in the residual. However, this may seem deceptively simple as we do not directly deal with the residual itself but with the solution and the residual through the matrix. Hence, this has to be done consistently with the linear system involved. It rather consist in computing an approximate solution to the linear system with a smaller matrix than actually smoothing the residual.

To do so, we need to define a restricting operator to consistently transfer the residual from the fine grid to the coarse grid, compute the correction on the coarse grid and transfer it back to the fine grid. While projecting the coarse correction on the fine grid can be done straightforwardly with a linear interpolation $I_{H\to h}$, defining the restricting operator $\mathcal{R}_{h\to H}$ and the matrix on the coarse grid A^H, is a crucial and non-trivial issue. Linear problems enjoy rather good properties, and the Jacobian of the same problem on the coarse grid can be used as approximate matrix,

but as we will see, it may not be the case for non-linear problems.

General procedure:

Let us consider the linear system derived from the linearization of our numerical problem

$$A\delta W = R, \tag{1.22}$$

solved with an iterative procedure. We introduce the linear residual at the k-th iteration

$$r_k = R - A\delta W^k. \tag{1.23}$$

Note that, the resolution of Equation (1.22) can be seen at each step as a new linear problem:

$$\begin{aligned} A\delta W &= A(\delta W^k + \widetilde{\delta W}) = R, \\ A\widetilde{\delta W} &= \underbrace{R - A\delta W^k}_{=r_k}. \end{aligned}$$

Hence, each SGS iteration can be seen as a correction to the previous step, and similarly, any method can be used to compute a correction $\widetilde{\delta W}$.

To compute $\widetilde{\delta W}$ with a multigrid step, we first restrict r_k to the coarser grid. The restriction operator that we use is somehow the opposite of the linear interpolation: each vertex P_i of the fine mesh $\mathcal{H}_h$ is located in an element K_H of the coarse mesh $\mathcal{H}_H$ with its barycentric coordinates $\alpha(P_i, K_H, j)$ so that

$$P_i = \sum_{P_j \in K_H} \alpha(P_i, K_H, j) P_j.$$

In the interpolation process, the value W_i at node P_i is affected as

$$W_i = \sum_{P_j \in K_H} \alpha(P_i, K_H, j) W_j,$$

and on the opposite, in the restriction process, the residual $r_{k,i}$ at node P_i is added to the residual of each node P_j as

$$r_j^H = \mathcal{R}_{h\to H}(r_k)(P_j) = \sum_{K_H \ni P_j} \sum_{P_i \in K_H} \alpha(P_i, K_H, j) r_{k,i}.$$

The correction $\widetilde{\delta W}^H$ is computed on the coarse mesh from the linear problem

$$A^H \widetilde{\delta W}^H = r^H.$$

We will detail after the computation of A^H. $\widetilde{\delta W}^H$ can be computed with any traditional method, SGS, GMRES, or even another level of multigrid resolution. Hence, it does not necessarily satisfies the coarse linear problem, but it is a guess of the coarse solution and in turn of the fine solution. This correction is then interpolated back on the fine mesh and added to δW^k,

$$\delta W^{k+1} = \delta W^k + I_{H\to h}\left(\widetilde{\delta W}^H\right),$$

which formes the new state of the linear system:

$$A\delta W = A(\delta W^k + I_{H\to h}(\widetilde{\delta W}^H) + \widetilde{\widetilde{\delta W}}) = R.$$

Computation of A^H***:*** The standard multigrid analysis in finite elements involves linear problems and grid subdivisions, so that the coarse vector space of the test functions on the coarse mesh V_H is a subspace of the fine vector space V_h. The (constant) Jacobian of the problem on the coarse mesh $A\left(I_{h\to H}(W_H)\right)$ is a good choice for A^H. In that respect, this has been extended as well to the general case (non-linear, non-matching meshes,...).

However, for non-linear cases, some special care has to be taken. As the Jacobian is no more constant but depends on the solution, it may be strongly impacted by the coarser discretization. In the case of inviscid compressible flows, the variations in the solution induced by the discretization of shocks from a mesh to another may lead to a poor correction and potentially lead to divergence in the linear system.

In all cases, a special care has to be be taken to deal with the time dependency. The matrix on the coarse mesh must describe the same behaviour as on the fine mesh. In particular, as we use a pseudo unstationary solver, the time step has to be the same. Yet, the time step is defined with respect to the cell-size through the CFL. Hence we have to carefully interpolate the time step and not the nodal CFL from the fine grid to the coarse grid in order to compute a consistent mass matrix.

Another option is to use an approach closer to algebraic multigrid methods. The idea is to keep the geometric contributions of two meshes, but algebraically deduce the matrix on the coarse mesh from the restriction and interpolation operators. We replace the physical residual on the coarse mesh by the accumulation of the physical residual on the fine mesh:

$$\tilde{R}_H(W_H) = \mathcal{R}_{h\to H}\left(R_h\left(I_{H\to h}(W_H)\right)\right).$$

Hence, we deduce

$$\tilde{A}_H(W_H) = \frac{\partial \tilde{R}_{H,i}}{\partial W_{H,j}} = \frac{\partial [\mathcal{R}_{h\to H}]_i}{\partial W_{h,k}} \underbrace{\frac{\partial R_{h,k}}{\partial W_{h,l}}}_{A_h} \frac{\partial [I_{H\to h}]_l}{\partial W_{H,j}}.$$

The Jacobian matrix on the coarse mesh can thus be accumulated from the Jacobian on the fine mesh. To do so we go through the edges of the fine mesh that gives us the different coefficients of the fine Jacobian. We then locate both vertices of the fine edge in the coarse mesh as shown in Figure 1.23. The red edge is considered to be an edge of the fine mesh while triangles (ABD) and (BCD) are parts of the coarse mesh. We denote $\alpha(X,Y)$ the barycentric coordinate of vertex X in its triangle with respect to node Y, so that the interpolated value of the correction in F will be

$$I_{H\to h}(\widetilde{\delta W}_H)_F = \alpha(F,A)\widetilde{\delta W}_{H,A} + \alpha(F,B)\widetilde{\delta W}_{H,B} + \alpha(F,D)\widetilde{\delta W}_{H,D},$$

and similarly for E. Conversely, the restriction of the residual in B is of the form

$$r_B^H = \alpha(E,B)r_E^h + \alpha(F,B)r_F^h + \dots.$$

In order to accumulate the terms of the coarse matrix, we consider successively each vertex of the triangles. Let us first consider vertex A. The correction $\widetilde{\delta W}_{H,A}$ in A will be interpolated in F and contributes to the correction in F as

$$\frac{\partial \widetilde{\delta W}_{h,F}}{\partial \widetilde{\delta W}_{H,A}} = \alpha(F,A).$$

This small correction in F will produce a correction in the physical residual in both F and E determined by the fine Jacobian terms $A_{F,F}$ and $A_{F,E}$. These variations of the physical residual will be then accumulated back on the coarse mesh. Vertex F will contribute to the residual of A, B and D, while E will contribute to the residual of B, C and D, so that

$$\begin{aligned}
\frac{\partial \tilde{R}_A^H}{\partial \delta W_{H,A}} &= \alpha(F,A)\frac{\partial R_F}{\partial W_F}\alpha(F,A) \\
\frac{\partial \tilde{R}_B^H}{\partial \delta W_{H,A}} &= \alpha(F,A)\frac{\partial R_F}{\partial W_F}\alpha(F,B) + \alpha(F,A)\frac{\partial R_E}{\partial W_F}\alpha(E,B) \\
\frac{\partial \tilde{R}_D^H}{\partial \delta W_{H,A}} &= \alpha(F,A)\frac{\partial R_F}{\partial W_F}\alpha(F,D) + \alpha(F,A)\frac{\partial R_E}{\partial W_F}\alpha(E,D) \\
\frac{\partial \tilde{R}_C^H}{\partial \delta W_{H,A}} &= \alpha(F,A)\frac{\partial R_E}{\partial W_F}\alpha(E,C).
\end{aligned}$$

However, as the edge AC is not present in the coarse mesh, we simply discard the contribution $\frac{\partial \tilde{R}_C^H}{\partial \delta W_{H,A}}$ in order to keep the same matricial structure. Additionally, the terms of the form $\alpha(F,A)\frac{\partial R_F}{\partial W_F}\alpha(F,A)$ appear for every edge containing F but should not be counted multiple times. We thus compute them separately with a cycle on the vertices.

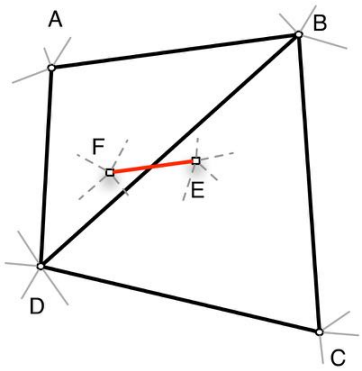


Figure 1.23: Matrix accumulation.

1.2.4 Multigrid in `Wolf`:

In `Wolf` we use the Jacobian of the coarse problem to compute the correction on the coarse mesh. We interpolate physical variable to guarantee the positivity of the pressure on coarser grids. The linear system matrix is re-build from these interpolated fields and using the cells of the coarser grid. But, to have a consistent matrix which progress of the same (local) time step, we have to interpolate linearly the local time step on the coarser grid. It should not be recomputed. This effect is clearly visible on supersonic case which fail without this modification.

In the case of RANS boundary layers, the discretization has a strong effect on the solution, particularly in the first layers of elements, near a wall. In order to minimize these effects we do not solve the turbulence model on coarser grid because it will "laminarize" the flow. We interpolate linearly μ_t on the coarser grid and we compute corrections only for the mean flow. Still, the correction may exhibit some defects. In order to counter them, we decided to limit and relax the correction to a fraction of the solution as for the standard resolution.

Algorithm 3 Multigrid solver

1: Perform NbrItePre iterations of SGS solver on the fine gird as "pre-smoothing" to compute $\delta W^{k,pre}$.
2: Interpolate the physical variables, time step and turbulent viscosity on the second grid: $W_{H_1} = I_{h\to H_1}(W_h)$.
3: Restrict the linear residual from the fine grid: $\mathcal{R}_{h\to H_1}(r_k)$.
4: Compute the Jacobian on the second grid: $\tilde{A}_{H_1}(I_{h\to H_1}(W_h))$.
5: Perform NbrItePre iterations of SGS solver on the second grid as "pre-smoothing" and compute $\widetilde{\delta W}^{pre}_{H_1}$.
6: Interpolate the physical variables, time step and turbulent viscosity on the third grid. Restrict the linear residual from the second grid and compute the Jacobian on the third grid: $W_{H_2} = I_{H_1\to H_2}(W_{H_1})$.
7: ...
8: Interpolate the physical variables, time step and turbulent viscosity on the coarsest grid: $W_{H_n} = I_{H_{n-1}\to H_n}(W_{H_{n-1}})$. Restrict the linear residual from the previous grid and compute the Jacobian on the coarsest grid. $\mathcal{R}_{H_{n-1}\to H_n}(r^{H_{n-1}})$.
9: Compute NbrItePre+NbrItePos iterations of SGS solver on the coarsest grid as "smoothing" to compute $\widetilde{\delta W}_{H_n}$.
10: Interpolate the correction from the coarsest grid to the previous one: $I_{H_n\to H_{n-1}}(\widetilde{\delta W}_{H_n})$. Relax by a factor of RlxCof and limit this correction and add it to the correction on the previous grid: $\widetilde{\delta W}_{H_{n-1}} = \widetilde{\delta W}^{pre}_{H_{n-1}} + \text{RlxCof}\cdot I_{H_n\to H_{n-1}}(\widetilde{\delta W}_{H_n})$.
11: Perform NbrItePos iterations of SGS solver on the previous grid as "post-smoothing": $\widetilde{\delta W}^{post}_{H_{n-1}}$.
12: ...
13: Perform NbrItePos iterations of SGS solver on the second grid as "post-smoothing": $\widetilde{\delta W}^{post}_{H_1}$.
14: Interpolate the correction to the computational (fine) grid. Relax and limit this correction and add it to the δW field: $\delta W_h = \delta W^{pre}_h + \text{RlxCof}\cdot I_{H_1\to h}(\widetilde{\delta W}^{post}_{H_1})$
15: Perform NbrItePos iterations of SGS solver on the fine grid as "post-smoothing": $\delta W^{k,post}$

Parameters: The overall process thus states as Algorithm 3. This forms a single iteration of multigrid solver. It is iterated up to NbrMgd times to converge the linear system at each physical (in time) iteration. At each iteration, the CFL is multiplied by a factor CFLCof, and limited to the maximal value CFLMax. We thus have to choose the following parameters

- NbrMgd : number of multi-grid cycles
- NbrItePre : number of pre-smoothing iterations
- NbrItePos : number of post-smoothing iterations
- RlxCof : relaxation coefficient for the accumulate correction
- CFLCof : CFL geometric coefficient at each iteration
- CFLMax : maximal CFL

Parameter study:

To validate our implementation in different configurations and asses the effect of the different parameters we performed a parameter study with the following values:

- NbrMgd $= \{3, 5, 10\}$
- NbrItePre $= \{0, 1, 2, 4\}$ (0 means we do only LUSGS)
- NbrItePos $= \{0, 1, 2, 4\}$
- RlxCof $= \{0.1, 0.5\}$
- CFLCof $= \{1.1, 1.2, 1.5, 2\}$
- CFLMax $= 10^6$ for Euler and Laminar and CFLMax $= 10^5$ for Turbulent

This represents 360 computations for each case (as the case NbrItePre $= 0$ and NbrItePos $= 0$ is not done). For each of the following flow regimes, we perform an inviscid, a laminar and a turbulent simulation. The cases are the following flow around a NACA0012:

- **Low mach number:** $M = 0.1$, $\alpha = 5.$, and $Re = 1\,000$ for laminar and $Re = 5 \times 10^6$ for turbulent.
- **Subsonic:** $M = 0.63$, $\alpha = 2.$, and $Re = 500$ for laminar and $Re = 5 \times 10^6$ for turbulent.
- **Transonic:** $M = 0.85$, $\alpha = 3.$, and $Re = 1\,000$ for laminar and $Re = 5 \times 10^6$ for turbulent.
- **Supersonic:** $M = 2.$, $\alpha = 3.$, and $Re = 2\,000$ for laminar and $Re = 5 \times 10^6$ for turbulent.

This represents a total of 4320 simulations. For each case we detail below the behaviour of `Wolf` depending on the multigrid parameters.

Inviscid flows

Low mach number. All cases are converging. But, NbrItePre $= 0$ is a bad choice, we have large CPU variation depending on the parameters. All the other cases are homogeneous. The optimal choice is NbrItePre $= 2$ and NbrItePos $= 1$ and CFLCof $= \{1.2, 1.5\}$. Couple (NbrItePre, NbrItePos) $= (2, 4)$ is also performing good.

Subsonic. All cases are converging. But, NbrItePre $= 0$ is a bad choice, we have large CPU variation depending on the parameters. All the other cases are homogeneous. The optimal choice is NbrItePre $= 1$ and NbrItePos $= \{0, 1, 2\}$ and CFLCof $= \{1.2, 1.5\}$. Couple (NbrItePre, NbrItePos) $= (2, 4)$ is also performing good.

Transonic. All cases are converging. But, NbrItePre $= 0$ is a bad choice, we have large CPU variation depending on the parameters. All the other cases are homogeneous. The optimal choice is NbrItePre $= 2$ and NbrItePos $= \{0, 1, 2, 4\}$.

Supersonic. Here, some cases are failing and others are are not converging. This occurs for NbrItePre = 0 and CFLCof = $\{1.5, 2\}$. These parameters should not be used at least for supersonic flows. The optimal choice is NbrItePre = 2 and NbrItePos = $\{1, 2\}$ and CFLCof = 1.1.

Conclusion. For Euler flows, the best parameters seem to be:

$$\text{NbrMgd} = 5, \quad \text{NbrItePre} = 2, \quad \text{NbrItePos} = 1, \quad \text{RlxCof} = 0.1, \quad \text{CFLCof} = 1.2$$

Laminar flows

Low mach number. All cases are converging. But, NbrItePre = 0 is a bad choice, we have large CPU variation depending on the parameters. All the other cases are homogeneous. The optimal choice is NbrItePre = 2 and NbrItePos = 4 and CFLCof = $\{1.2, 1.5\}$. Couple $(\text{NbrItePre}, \text{NbrItePos}) = (2, 4)$ is also performing good.

Subsonic. All cases are converging. But, NbrItePre = 0 is a bad choice, we have large CPU variation depending on the parameters. All the other cases are homogeneous. The optimal choice is NbrItePre = $\{1, 2\}$ and CFLCof = $\{1.2, 1.5, 2\}$. Couple $(\text{NbrItePre}, \text{NbrItePos}) = (2, 4)$ is also performing good.

Transonic. All cases are converging. But, NbrItePre = 0 is a bad choice, we have large CPU variation depending on the parameters. All the other cases are homogeneous. The optimal choice is NbrItePre = 2 and NbrItePos = $\{1, 2\}$. Couple $(\text{NbrItePre}, \text{NbrItePos}) = (2, 4)$ is a little bit slower but good too.

Supersonic. Here, some cases are failing and others are are not converging. This occurs for RlxCof = 0.5. NbrItePre = 0 is a again bad choice, we have large CPU variation depending on the parameters. The optimal choice is NbrItePre = 2 and NbrItePos = $\{0, 1\}$ and CFLCof = 1.1.

Conclusion. For Laminar flows, the best parameters seem to be:

$$\text{NbrMgd} = 5, \quad \text{NbrItePre} = 2, \quad \text{NbrItePos} = \{1, 2\}, \quad \text{RlxCof} = 0.1, \quad \text{CFLCof} = 1.2$$

Turbulent flows

Low mach number. Here, some cases are failing and others are are not converging. Cases are failing for NbrItePre = 0 and CFLCof = $\{1.5, 2\}$ (if not enough pre- and post-smoothing). Cases are not converging for combination $(\text{NbrItePre}, \text{NbrItePos}) = (1, 0),\ (1, 1)\text{and}(2, 0)$. There the sum of pre- and post-smoothing iterations should be of at least 3. The remaining cases are homogenous. Using $(\text{NbrItePre}, \text{NbrItePos}) = (2, 2)$ seems the best compromise with 3 or 5 cycles.

Subsonic. Here, some cases are failing and others are are not converging. Cases are failing or not converging for NbrItePre = $\{0, 1\}$, NbrItePos = $\{0, 1\}$. Working combination are $(\text{NbrItePre}, \text{NbrItePos}) = (2, 4),\ (4, 2)\text{and}(4, 4)$. We have to iterate to converge the cycle. The fastest is $(2, 4)$ with 3 cycle. Note that, in that context, one case do not converge for RlxCof = 0.5

Transonic. Here, some cases are failing and others are are not converging. Cases are failing or not converging for NbrItePre = 0, and couple (NbrItePre, NbrItePos) = $(1,0)$, $(1,1)$, $(1,2)$ and $(2,0)$. We have to iterate to converge the cycle. The other case are homogeneous. The optimal choice is NbrItePre = 2 and NbrItePos = 2 and NbrMgd = 3. Couple (NbrItePre, NbrItePos) = $(2,4)$ is a little bit slower but good too.

Supersonic. Here, some cases are failing and others are are not converging. Cases are failing or not converging for NbrItePre = 0, NbrItePos = 0. For RlxCof = 0.5, most of the cases are failing if a too high CFL coefficient is prescribed. The other cases are good and homogenous. The cases with NbrItePre = 4 or NbrItePos = 4 are more time consuming. The best couple are (NbrItePre, NbrItePos) = $(1,1)$, $(1,2)$ with NbrMgd = 3.

Conclusion. For turbulent flows, the best parameters seem to be:

$$\text{NbrMgd} = 3, \quad \text{NbrItePre} = 2, \quad \text{NbrItePos} = 4, \quad \text{RlxCof} = 0.1, \quad \text{CFLCof} = \{1.1, 1.2\}$$

to pass all the cases. NbrItePos can be lowered to 2 for low mach number, transonic and supersonic cases.

Multigrid best (good) practice from the parameters study

From the parameters study, these choices are mandatory for robustness (even if they are not optimal in CPU time):

- NbrMgd = 3
- NbrItePre $\in [2,4]$
- NbrItePos $\in [1,4]$
- RlxCof = 0.1
- CFLCof ≤ 1.2
- CFLMax = 10^6 for Euler and Laminar and CFLMax = 10^5 for Turbulent

Convergence analysis

The global numerical process involved in the flow solver aims at canceling the nonlinear residual vector $R(W)$. To do so we use a Newton's iterative method that relies on the following linearization of R:

$$R(W + dW) = R(W) + A|_W \cdot dW. \tag{1.24}$$

If the problem is linear, we are looking for $W + dW$ to be equal to the solution of the problem $R(W + dW) = R(W_0) = 0$ so that dW can be expressed from Equation (1.24) as

$$dW = -A|_W^{-1} \cdot R(W) \tag{1.25}$$

However in practice, inverting the system (1.25) is way to expensive so that we replace A by an approximate inverse matrix

$$G = A + E, \tag{1.26}$$

and the Equation (1.25) is solved up to a given residual so that we have

$$G \cdot dW = -R(W) + S. \tag{1.27}$$

The linear residual of the system is then expressed as $r = \|G \cdot dW + R(W)\| = \|S\|$. We propose to evaluate here the effect of these choices on the overall convergence.

To do so we will assume that R is a quadratic function (valid in the asymptotic convergence of the Newton's method) so that

$$R(W) = R(W_0) + A|_{W_0} \cdot (W - W_0) + \frac{1}{2}(W - W_0) \cdot H|_{W_0} \cdot (W - W_0) \tag{1.28}$$

where $H|_{W_0}$ is the third order tensor of the second derivatives of R in W_0, the solution of the problem. We will also approach the approximate inverse of G assuming that the error it introduces is small with regard of A, $\|E\| << \|A\|$, so that the following taylor expansion holds:

$$G^{-1} \approx (Id - A^{-1}E)A^{-1} \approx A^{-1}(Id - EA^{-1}). \tag{1.29}$$

Without approximations on A and it's inversion, the Newton's method converges quadratically if A^{-1} is bounded (as long as A is invertible). From Equation (1.25) we have

$$W^{n+1} - W_0 = W^n - W_0 - A|_{W^n}^{-1} \cdot R(W^n). \tag{1.30}$$

Performing a Taylor expansion from the solution W^n at the n^{th} iteration we have

$$0 = R(W_0) = R(W^n) + A|_{W^n} \cdot (W_0 - W^n) + \frac{1}{2}(W_0 - W^n)H|_{W^n}(W_0 - W^n). \tag{1.31}$$

from which we deduce

$$W^n - W_0 = A|_{W^n}^{-1} \cdot R(W^n) + \frac{1}{2}A|_{W^n}^{-1} \cdot \left[(W_0 - W^n)H|_{W^n}(W_0 - W^n)\right]. \tag{1.32}$$

From Equations (1.30) and (1.32) we obtain

$$W^{n+1} - W_0 = \frac{1}{2}A|_{W^n}^{-1} \cdot \left[(W^n - W_0)H|_{W^n}(W^n - W_0)\right]. \tag{1.33}$$

If $A|_{W^n}^{-1}$ and $H|_{W^n}$ are bounded in the vicinity of W_0 we obtain the quadratic convergence under the appropriates norms

$$\|W^{n+1} - W_0\| \leq \frac{1}{2}\|A^{-1}\|\|H\|\|(W^n - W_0)\|^2. \tag{1.34}$$

Let's now assume that A is exactly computed but approximately inverted so that Equation (1.25) becomes

$$dW^n = -A|_{W^n}^{-1} \cdot (R(W^n) + S). \tag{1.35}$$

The approximate inversion of A introduces an error on the next solution W^{n+1} equal to $A|_{W^n}^{-1}S$. If we perform a Taylor expansion of the residual in W^n to obtain the residual of W^{n+1} we get

$$\begin{aligned} R(W^{n+1}) \quad = \quad & \underbrace{\frac{1}{2}\left(A|_{W^n}^{-1} \cdot R(W^n)\right) H|_{W^n} \left(A|_{W^n}^{-1} \cdot R(W^n)\right)}_{\text{Residual without error}} \\ & \underbrace{+S + \left(A|_{W^n}^{-1} \cdot R(W^n)\right) H|_{W^n} \left(A|_{W^n}^{-1} \cdot S\right) + \frac{1}{2}\left(A|_{W^n}^{-1} \cdot S\right) H|_{W^n} \left(A|_{W^n}^{-1} \cdot S\right)}_{\text{Error due to S}}. \end{aligned}$$

Still assuming that A and H are bounded, we recover the quadratic convergence of the residual without S. Choosing the convergence of the residual of the linear system involves $\|S\| \leq \epsilon\|R(W^n)\|$ so that we can bound R^{n+1} as

$$\|R(W^{n+1})\| \leq \underbrace{\frac{\|A^{-1}\|^2\|H\|}{2}\|R(W^n)\|^2}_{\text{Quadratic}} + \underbrace{\epsilon R(W^n)}_{\text{Linear}} + \underbrace{(\epsilon + \frac{\epsilon^2}{2})\|A^{-1}\|^2\|H\|\|R(W^n)\|^2}_{\text{Quadratic}}. \quad (1.36)$$

We can see here that the approximate inversion of A introduces at each iteration in the next residual an error of the order of $\epsilon R(W^n)$. This in turn limits the quadratic convergence of the Newton's method asymptotically. However, it is worth noting here that in the case of strong nonlinearities, the quadratic terms may dominate due to the norm of $\|A^{-1}\|^2\|H\|$ (in particular on the initial iterations) so that the approximate inversion of A involves a relative error of the magnitude of ϵ. Namely, choosing $\epsilon = 0.1$ involves 10% error in the residual, $\epsilon = 0.01$ 1%, ...

Let's now assume that we approach A with $G = A + E$. The same developments hold and we have

$$\begin{aligned} R(W^{n+1}) &= (Id - EA|_{W^n}^{-1}) \cdot S + EA|_{W^n}^{-1} \cdot R(W^n) \\ &+ \frac{1}{2}\left(G^{-1}(S - R(W^n)) \cdot H|_{W^n}\left(G^{-1}(S - R(W^n)\right)\right) \end{aligned}$$

and

$$\begin{aligned} W^{n+1} - W_0 &= \frac{1}{2}A|_{W^n}^{-1} \cdot [(W^n - W_0)H|_{W^n}(W^n - W_0)] \\ &+ A|_{W^n}^{-1}EA|_{W^n}^{-1} \cdot R(W^n) + A|_{W^n}^{-1}(Id - EA|_{W^n}^{-1}) \cdot S. \end{aligned}$$

Here again we recover the quadratic convergence of the solution when no error is added with the term $\frac{1}{2}A|_{W^n}^{-1} \cdot [(W^n - W_0)H|_{W^n}(W^n - W_0)]$. To this, we add an error related to the use of an approximate Jacobian $A|_{W^n}^{-1}EA|_{W^n}^{-1} \cdot R(W^n)$ and an error related to the approximate inversion of G, $A|_{W^n}^{-1}(Id - EA|_{W^n}^{-1}) \cdot S$. As $\|R(W^n)\| \approx \|A\|\|W^n - W_0\|$ and $\|S\| \approx \epsilon\|R(W^n)\|$, both of these terms involve a linear convergence. Here again, we can see that it is needless to over resolve the linear system as the error due to the approximate Jacobian may dominate.

Efficiency: This analysis shows that both the use of an approximate Jacobian and the approximate resolution of the linear system introduce errors in the physical iterative process. We control, part of the approximation of the Jacobian through the CFL and the error in the linear system through the parameter α. In practice, the smaller α is, the more iteration and thus CPU time are required to solve the linear system but also the less time steps we need to achieve convergence. Similarly, the smaller the CFL is, the less stiff the matrix system is and thus the less iterations are required to solve the linear system, but the more error is introduced in the system and thus more physical iterations are required to achieve convergence of the residual. Hence, a tradeoff has to be found in order to minimize the CPU time.

Alongside these two controllable sources of errors, the non-linearity of the problem and the approximation of the Jacobian of the inviscid fluxes by the first order Jacobian introduce unavoidable errors. The non-linearity of the equations proves to be a large source of error in the initial steps of the computations, it is thus very efficient to start with a low CFL that will

increase during the computation. The initial iterations are cheaper and the error introduced by the approximation on the Jacobian is absorbed by the errors due to the nonlinearity.

On the contrary, in the asymptotic regime when the CFL is high, the error due to the approximation of the Jacobian dominates. Hence, it is useless to increase to much the CFL as the error due to the mass matrix will be dominated by the error due to the first order Jacobian. Similarly, it is useless to over converge the linear system. In practice we noted that for RANS applications, it is reasonable to increase the CFL up to 1000 and converge the linear system by one or two orders of magnitude. Though, these parameters strongly depend on the choices made to compute the Jacobian. As we will show in Section 2.2.1, an improper differentiation of the residual will introduce an error sufficient to make the process unstable and limit the achievable CFL to 5 and require to converge the linear system by at least three orders of magnitude.

1.2.5 Residual computation and limiter freezing

Statistical residual

As we seek a stationary solution, the time derivatives of the problem should be null and thus, the sum of the fluxes entering or going out of each finite volume cell should be null. This sum defines the nodal residual R_i. The standard global residual is obtained by summing the absolute values or the squared values of all the nodal residuals,

$$Res = \sum_i |R_i|.$$

But there is a wide spread in the magnitudes of order of the different nodal residuals and thus, as they converge to zero, their sum is actually representative of the highest nodal residual in the domain. It ignores the overall distribution of the residual field that can provide a pertinent information. We thus decided to compute more data on the residual. Given the maximal nodal value of the residual Res_{max}, we count the number of vertices verifying $Res_{max} > Res > 0.1Res_{max}$, $0.1Res_{max} > Res > 0.01Res_{max}$, ..., so that we obtain a distribution of the logarithm of the residual. From this we obtain among other informations

- The proportion of vertices below the desired threshold.
- The evolution of the distribution.
- A logarithmic residual.

These data can be used to assess the convergence of the solution more precisely. In particular, if all nodal residuals are converged down to low values except some pathological points, we can track this problem, while it is simply diluted in a high mean value in the standard residual evaluation.

Logarithmic residual: The logarithmic residual is computed as

$$Res_{log} = \exp\left(\frac{1}{N}\sum^{N}\log(R_i)\right).$$

To determine the convergence of a computation we usually don't look at the actual value of the residual but rather at its order of magnitude. Hence, it makes sens to look at the mean of the magnitude (log scale) of the nodal residuals instead of their direct mean value.

Limiter freezing

The convergence of stationary numerical simulation can be prevented by limite cycles, due either to instabilities that lead to unsteady solutions or the use of limiters. The unsteady solutions can be damped simply using high CFLs, hence the necessity of a robust solver. On the contrary, limiter induced instabilities tends to be stronger with high CLFs. They are due to different effects.

In regions with little variations in the solution (in the far field), the gradient of the solution ∇W is close to 0 so that it can oscillate around 0 with positive and negative values with no particular meaning. But in the MUSCL extrapolation, gradients with opposite signs are interpreted as spurious oscillations so that no extrapolation is done. Extrapolation is thus alternatively activated and deactivated due to small variations in the gradient, leading to other small variations in the solution. In conjunction with large cells in the far field, this may lead to large variations in the residual with no particular meaning.

Additionally, both limiters and HLLC solver exhibits discontinuous Jacobians due to transitions between different regimes: limiters switch between different regimes depending on the ratio of upwind and centered gradients, for example when both gradients have the same or opposite sign. HLLC solver switches between subsonic and supersonic regimes depending on the contact and acoustic wave velocities. This also may prevent convergence and produce cycles.

These processes produce small cycle variations in the solution that fixes the residual to high values in some points. As the global residual of the system is taken to be the sum of the nodal residuals, it is dominated in this case by a few nodes whose residual does not converge, while the solution reached its final state and most of the nodes are converged.

Limiter freezing

In order to tackle these oscillations, an option is to remove a part of the oscillatory process, freezing the values of the gradients in the MUSCL extrapolation. This prevents the limiter to further activate/deactivate. However, as it modifies the MUSCL extrapolation, it must not be activated while the solution is not converged. It thus requires to be able to know when it must be activated. To do so we rely on different options to trigger, either directly the limiter freezing or a stall detection.

Iteration trigger: We generally limit the total number of iterations allowed for the simulation after which the computation is simply stopped. Additionally, we can directly trigger the limiter freezing a few iteration before shutting down the simulation in order to force the convergence if steady state has already been achieved. But we can also activate the limit cycle detection (see below) much earlier after a given number of iteration, which allows earlier freezing and prevents premature false detection in the early stages of the computation.

Residual trigger: The residual of the simulation generally stalls above the target residual but below the initial residual. The problem is that we cannot raise the target residual as it may lead to unconverged solutions, but we can use it as an indicator of the state of the solution to trigger either directly the limiter freezing or the stall detection. Hence, we generally activate the limit cycle detection as soon as the residual dropped by two orders of magnitude for computations in a mesh adaptation context. Indeed, depending on the initial solution chosen (interpolated, uniform field, artificial field, ...) the initial and maximal residual may vary largely.

Statistical Residual trigger: Statistical residual offers more options to assess the convergence and cycles of the solution. As the standard residual is dominated by a few vertices with high values, the logarithm residual is far less sensitive to the limit cycles. It converges down to smaller values and can be used to trigger, the limit cycle detection or the limiter freezing as soon as it drops below a given threshold, typically 1×10^{-4}. Additionally, we can also trigger limiter freezing as soon as 95% or 99% of the vertices are converged.

Aero-coefficient trigger: As limiter induced limit cycles produce very small variations in the solution, they also lead to negligible variations in the outputs. Hence, a good indication of the state of convergence of the solution is to look at the variation in the functional outputs such as aero-coefficients. Comparing the relative variation of the outputs between two successive solutions, we can trigger separately the limit cycle detection, when the relative variation falls below 1×10^{-3} and directly the limiter freezing when the relative variation falls below 1×10^{-5}. Here again, in order to prevent false detections in the initial steps of the computation, these sensors are activated after a minimal number of iterations.

Limit cycle detection: Once activated, the limite cycle detection tracks every 20 iterations the evolution of the upper and lower bound of the residual. If none of them changes form more than 5%, the limiters are frozen.

1.3 Adjoint

Many functional output optimization problems, such as goal oriented mesh adaptation (see Section 6.1.1), require the computation of the numerical adjoint. Numerically this consists in computing the sensitivity of a given functional $j(W_h)$ to the local nodal residual $R_i(W_h)$, through the linearization of the numerical problem:

$$\frac{\partial j}{\partial R} = \frac{\partial j}{\partial W_h} \cdot \left(\frac{\partial R}{\partial W_h} \right)^{-1} = \left(\frac{\partial R}{\partial W_h} \right)^{-T} \cdot \frac{\partial j}{\partial W_h}$$

This is done in three steps:

1. Compute the sensitivity of the functional to the solution $J = \frac{\partial j}{\partial W_h}$
2. Compute the transpose of the Jacobian $A^T = \left(\frac{\partial R}{\partial W_h} \right)^T$
3. Solve the linear system $W^* = A^{-T} J$.

We detail here the numerical implementation of these steps.

1.3.1 Functional sensitivity to the solution

Usual functional outputs are generally defined as integrals over parts of the domain. The differentiation of $j(W_h)$ is directly deduced from the discretization of j.

Surface functionals

Gradient based lift and drag: The forces applied on a body are deduced from the integral of the surface forces. For RANS applications it decomposes as a pressure and viscous contribution. The nodal contribution to the pressure component reads

$$\mathrm{d}F_i^{\text{pressure}} = p_i \mathbf{n}_i \mathrm{d}S = (\gamma - 1)\left((\rho E)_i - \frac{1}{2}\frac{(\rho u)_i^2 + (\rho v)_i^2 + (\rho w)_i^2}{\rho_i}\right)\mathbf{n}_i \mathrm{d}S.$$

Thus we obtain the nodal differentiation of the pressure contribution to the forces on the body

$$\frac{\partial(\mathrm{d}F_i^{\text{pressure}})}{\partial W} = (\gamma - 1)\left(\frac{\|\mathbf{u}_i\|^2}{2}, -u_i, -v_i, -w_i, 1\right)\mathbf{n}_i \mathrm{d}S.$$

The viscous contribution to the body forces expresses similarly except that it depends on gradients, so that it involves the considered node and its neighbours. It is computed vertex-wise, the nodal normal is take to be the weighted averaged normal of the surrounding surface triangle (edges in 2D)

$$\mathbf{n}_i = \frac{1}{\sum_{T \ni i} |T|} \sum_{T \ni i} |T| \, \mathbf{n}_T,$$

and the nodal gradient is taken to be the weighted averaged gradient in the surrounding tetrahedra (triangles in 2D)

$$\nabla W_i = \frac{1}{\sum_{K \ni i} |K|} \sum_{K \ni i} |K| \, \nabla W|_K = \frac{1}{\sum_{K \ni i} |K|} \sum_{K \ni i} |K| \sum_{j \in K} W_j \nabla \phi_j,$$

with ϕ_j the P_1 linear test functions. The viscous contribution is thus computed as

$$\mathrm{d}F_i^{\text{viscous}} = \mathcal{T} \cdot \mathbf{n}_i \mathrm{d}S,$$

where $\mathcal{T}$ is given in Section 1.1.1. It can be differentiated as in the main flow solver, we write here only the contribution of the $\mathcal{T}_{xx}$ component to the force along the x direction:

$$\begin{aligned}(\mathrm{d}F_i^{\text{viscous}})_x &= [\mathcal{T}_{xx}(\mathbf{n}_i)_x + ...] \, \mathrm{d}S \\ &= \left[(\mu + \mu_t)\frac{2}{3}(2u_x - v_y - w_z)(\mathbf{n}_i)_x + ...\right] \mathrm{d}S.\end{aligned}$$

As we are on a wall, $\mu_t = 0$. The differentiation of μ through the Sutherland Law yields nodal terms that we do not develop here. The development of the gradients in each element containing node P_i leads to

$$(\mathrm{d}F_i^{\text{viscous}})_x = \frac{2}{3}\mu \left[\left(\sum_{K \ni i}\sum_{j \in K} 2u_j \nabla_x \Phi_j - v_j \nabla_y \Phi_j - w_j \nabla_z \Phi_j\right)(\mathbf{n}_i)_x + ...\right] \mathrm{d}S,$$

so that the contribution of node P_i to the viscous part of the force (and hence the values of the right-hand side J) reads

$$\frac{\partial(\mathrm{d}F_i^{\text{viscous}})}{\partial W_i} = \frac{2}{3}\mu \left(0, 2\sum_{K\ni i}\nabla_x\Phi_i, -\sum_{K\ni i}\nabla_y\Phi_i, -\sum_{K\ni i}\nabla_z\Phi_i, 0\right)(\mathbf{n}_i)_x \mathrm{d}S + \dots,$$

and its neighbours P_j contribute as

$$\frac{\partial(\mathrm{d}F_i^{\text{viscous}})}{\partial W_j} = \frac{2}{3}\mu \left(0, 2\sum_{K\ni (i,j)}\nabla_x\Phi_j, -\sum_{K\ni (i,j)}\nabla_y\Phi_j, -\sum_{K\ni (i,j)}\nabla_z\Phi_j, 0\right)(\mathbf{n}_i)_x \mathrm{d}S + \dots.$$

Residual based lift and drag: In Section 1.1.3, we recalled how the local contributions to the aero-coefficients can be obtained with higher accuracy, from the discarded velocity residual of the boundary nodes on a no-slip surface. In particular, for example, the drag is defined as the projection of the aero-forces along the reference fluid velocity direction $\mathbf{e}_\infty$, so that the local contribution to the drag reads

$$dF_i^{\text{res}} = \mathbf{e}_{\infty,x} R_i^u + \mathbf{e}_{\infty,y} R_i^v \,.$$

Hence, the contribution of each neighboring node to the drag can be deduced from its contribution to the local residual,

$$\frac{\partial F_i^{\text{res}}}{\partial W_j} = \mathbf{e}_{\infty,x}\frac{\partial R_i^u}{\partial W_j} + \mathbf{e}_{\infty,y}\frac{\partial R_i^v}{\partial W_j} \,.$$

These are the same terms as for the Jacobian used in the implicit resolution and can be thus assembled the same way. In particular, the residual R may be replace by an approximate residual $\hat{R}$.

Debit and other plane control: We can also seek to control any output computed on an immersed surface in the volume. We take here the example of the debit crossing a given plane

$$j(W) = \int_S \mathbf{u}\cdot\mathbf{n}\mathrm{d}S = \int_S \frac{\rho\mathbf{u}}{\rho}\cdot\mathbf{n}\mathrm{d}S.$$

We assume that the considered surface is geometrically present in the mesh, so that integral of $j(W)$ can be computed on a list of elements, rewritten vertex-wise

$$j(W) = \sum_{P_i}\sum_{T\ni P_i}\frac{1}{3}|T|\frac{(\rho\mathbf{u})_i}{\rho_i}\cdot\mathbf{n}_i.$$

Hence, the right-hand side term J writes

$$J_i = \sum_{T\ni P_i}\frac{1}{3}|T|\left(-\frac{\mathbf{u}_i\cdot\mathbf{n}_i}{\rho_i}, \frac{(n_i)_x}{\rho_i}, \frac{(n_i)_y}{\rho_i}, \frac{(n_i)_z}{\rho_i}, 0\right).$$

Volume functionals

Any volume functional can be expressed and controlled similarly. These functional outputs on the whole field can be in mesh adaptation a trick to control the implicit error of the whole flow field. As adjoint-based adaptation aims at minimizing $j(W) - j(W_h)$, choosing $j(W)$ as an integral over the whole field will tend to control the variations $W - W_h$ overall. We give here a few examples, discretized vertex-wise.

Mean density: We can consider the mean density of the flow as

$$j(W) = \int_\Omega \rho \, \mathrm{d}\Omega = \sum_{P_i} |C_i| \rho_i,$$

so that the right hand side writes

$$J_i = |C_i| \, (1, 0, 0, 0, 0) \, .$$

Mean Mach number: We can consider the mean Mach number of the flow to control indirectly both the density, the velocity and the energy. This should be done complementary to a standard feature-based adaptation on the Mach field. The functional thus reads

$$j(W) = \int_\Omega \frac{|\mathbf{u}|}{c} \, \mathrm{d}\Omega = \sum_{P_i} |C_i| \sqrt{\frac{(\rho u)_i^2 + (\rho v)_i^2 + (\rho w)_i^2}{\gamma p \rho}},$$

so that the right-hand side writes

$$J_i = |C_i| \frac{c}{2\gamma |\mathbf{u}| p^2} \left(|\mathbf{u}|^2 [p + (\gamma - 1)\rho |\mathbf{u}|^2 / 2], (2p + (\gamma - 1)\rho |\mathbf{u}|^2)\mathbf{u}, (\gamma - 1)|\mathbf{u}|^2 \right) .$$

Norm-Oriented functionals

In the standard adjoint-based mesh adaptation approach, we aim at controlling the error on a given functional j in L^1 norm: $|j(W) - j(W_h)|$. This functional j is considered to be a standard function depending on W. The idea of norm-oriented mesh adaptation is to consider j to be actually a function of the error $W - W_h$ itself. More generally, instead of considering j to be the integral of a given field

$$j(W_h) = \int_\Omega g(W_h) \, \mathrm{d}\Omega,$$

we consider j to be the integral of the quadratic error of a given field

$$j(W_h) = \int_\Omega (g(W_h) - g(W))^2 \, \mathrm{d}\Omega.$$

Note that any p-norm can be considered, but $p = 1$ degenerates to the standard adjoint-based mesh adaptation. Any other function of $W - W_h$ can also be considered. However, it should not be mistaken for

$$j(W_h) = \left(\int_\Omega g(W_h) - g(W) \, \mathrm{d}\Omega \right)^p ,$$

as this expression is also equivalent to standard adjoint-based adaptation.

The differentiation of j leads to

$$J = \frac{\partial j}{\partial W_h} = \int_\Omega 2(g(W_h) - g(W))\frac{\partial g}{\partial W}\,\mathrm{d}\Omega,$$

where $g(W_h) - g(W)$ is the field of the implicit error. It can be estimated with the non-linear correctors proposed in Section 4.1. Hence the right-hand side vector J reads

$$J_i = 2(g(W) - g(W_h))_i \left(\frac{\partial g}{\partial \rho}, \frac{\partial g}{\partial \rho u}, \frac{\partial g}{\partial \rho v}, \frac{\partial g}{\partial \rho w}, \frac{\partial g}{\partial \rho e}\right).$$

1.3.2 Jacobian computation and system resolution

The accuracy of the adjoint directly depends on the precision of the computed Jacobian and the inversion of the linear system. The Jacobian has thus to be computed as precisely as possible. In particular, it must not include the mass matrix. Additionally, in order to improve the differentiation of inviscid fluxes, and contrary to the main solver where the first order Jacobian is considered, we perform MUSCL extrapolation on each edge before computing contributions to the Jacobian. Though, we do not differentiate limiters as it may lead to too non-linear erratic effects.

The absence of mass matrix and the use of a more precise differentiation lead to a stiffer system that SGS solver cannot handle. As we need the linear system to be converged by at least eight orders of magnitude (10^{-8}) to obtain an accurate adjoint, we need a stronger linear solver. Thus we use a Krylov GMRES solver. It is still preconditioned with a LU-SGS solver, *i.e.* the vectors added to the Krylov subspace are not computed as $A^k J$ but as the solution obtained with a single iteration of LU-SGS from the current residual.

1.4 Verification and validation

1.4.1 Verification

Verification aims at ensuring that the mathematical problem is appropriately solved. This means that the equations and boundary conditions are appropriately implemented and bug free. In that case, the numerical solution should converge to the analytical solution of the mathematical problem with a given order. However, as there is no non trivial analytical solution to the RANS equations, this procedure relies either on manufactured solution or reference codes. This is why Roache [Roache 1998] proposed for verifications to add a well chosen source term to the equations so that a chosen field W is the exact solution. W can then be used to check the proper convergence of the method.

By injecting the chosen analytical solution W in the equations we obtain a defect residual which can be used as a source term in the discrete equation

$$\Psi_h(W_h) = \Pi_h(\Psi(W)).$$

In practice, for finite volume methods, we recall that the nodal residual before discretization is supposed to be the integral over a cell of the continuous residual

$$R_i(W) = \int_{C_i} \Psi(W)\,\mathrm{d}\Omega.$$

Hence, given a chosen field W, the numerical solution of the discretized problem

$$R_i(W_h) = \underbrace{\int_{C_i} \Psi_h(W_h)\,\mathrm{d}\Omega}_{\text{Standard residual}} - \underbrace{\int_{C_i} \Psi(W)\,\mathrm{d}\Omega}_{\text{Souce term}},$$

should converge toward W. Note that for standard computations, we have $\Psi(W) = 0$ so that we solve in some sens the exact same problem.

Here $\Psi(W)$ is a known analytical expressions but its integral $\int_{C_i} \Psi(W)\mathrm{d}\Omega$ is non trivial. Still, it has to converge faster than the numerical discretization of the equations so that we mesure the convergence of the numerical method and not the convergence of the source term. In our case, we use a linear integration which is second order accurate. In finite volume, this has the advantage to be done by cell constant integrations. Indeed, we obtain a linear integration on each element (we consider here the 2D example as it is the same in 3D), with a single Gauss quadrature point at the center of gravity of the triangle so that

$$\int_{K_{P_1}} f\,\mathrm{d}K = |K|\frac{1}{3}\sum_{i\in K} f_i.$$

But, as the triangle is split into three elementary cells, we have

$$\int_{K_{P_1}} f\,\mathrm{d}K = \sum_{i\in K} |K \cap C_i| f_i.$$

Hence we have

$$\int_{\Omega} f\,\mathrm{d}\Omega = \sum_{K} |K|\frac{1}{3}\sum_{i\in K} f_i = \sum_{K,C_i} |K \cap C_i| f_i = \sum_{C_i} |C_i| f_i.$$

Given the analytical solution of the problem W and the numerical solution W_h computed with the source term, we split the approximation error $W - W_h$ in the interpolation error and the implicit error as

$$W - W_h = \underbrace{W - \Pi_h W}_{\text{Interpolation Error}} + \underbrace{\Pi_h W - W_h}_{\text{Implicit Error}}.$$

In order to asses the convergence of the solver we choose to focus on the implicit error as the interpolation error behaviour is known. The implicit error can be advantageously exactly computed on each mesh as

$$\mathcal{E}^{Imp}_{Exa} = \sqrt{\int_{\Omega} \|\Pi_h W - W_h\|^2\,\mathrm{d}\Omega},$$

contrary to the interpolation error that requires either a finer mesh or quadratures.

Computation of the source term:

The computation of the manufactured solutions source term requires to differentiate the chosen analytical solution. We present three possible differentiations.

Direct differentiation: The most straight forward option is to directly differentiate and compute by hand the different terms. This approach requires no further tools and allows a full control on the method. But it may be a very tedious task, even for simple analytical expressions, to verify that the differentiation was done without error or typos. We used this approach for simple expressions and verifications.

Automatic differentiation: An easy option is to use an automatic differentiation library (Tapenade [Hascoet 2013], ...), or a symbolic computing environment (Maple,...). Still, it requires a preprocessing and external tools that may not be accessible or desirable.

Finite difference differentiation: Another simple option is to compute the derivatives with finite differences. This approach is a good compromise to minimize the bug possibilities as it is straight forward. However, it requires a careful management of the finite difference step in order to control numerical errors. In particular it has to be smaller than the mesh-size (by at least one order of magnitude). But, it must not go below a given threshold that depends on the floating point representation in order to avoid computational numerical errors. It has the disadvantage to limit the minimal mesh size achievable. Still we retain this option for its flexibility.

Test case

We verified the laminar Navier-Stokes component of `Wolf` in 2D with manufactured solutions, with and without mesh adaptation. To do so, we chose the following analytical solution

$$\begin{aligned}
\rho &= 0.5\,(1+\exp(-y)(\cos(10y)+\sin(10x))) \\
u &= 1+\exp\left(-1000\frac{y^2}{x+2}\right) \\
v &= y\exp\left(-1000\frac{y^2}{x+2}\right) \\
p &= 1-0.1\exp\left(-1000\frac{y^2}{x+2}\right).
\end{aligned}$$

The appropriate source term is computed and injected in the equations and the resulting solution on a 94.478 vertices mesh is shown in Figure 1.24. The mesh is adapted with a feature-based adaptation, using the mach field as sensor. Five meshes are generated for each complexity, this complexity being multiplied by two at each step. The resulting mesh is shown in Figure 1.25 for the third step, with a complexity of 94.478 vertices. Finally, Figure 1.26 shows the convergence of the implicit error. We can see here that the adaptation process successfully converges at second order, validating both the solver implementation, adaptation tools, and sensors.

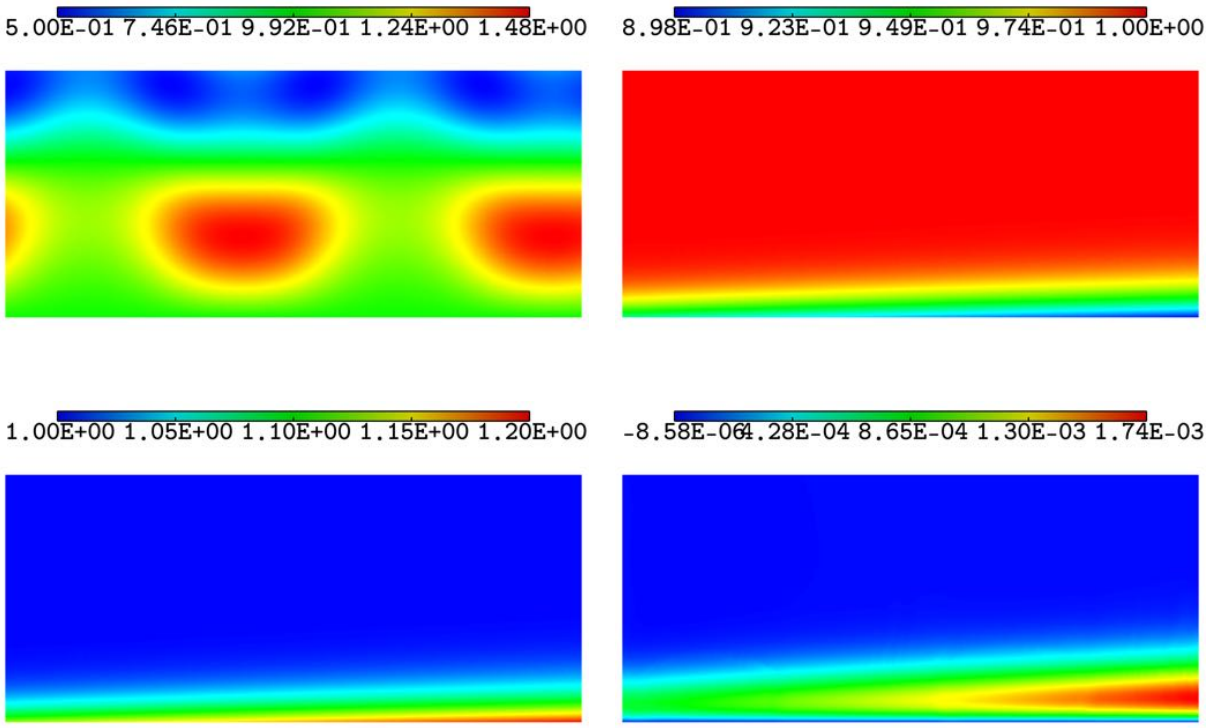


Figure 1.24: Manufactured solution: Solution field on a 94.478 vertices adapted mesh, density (top left), pressure (top right), x-component of the velocity (bottom left) and y-component of the velocity (bottom right) velocity.

1.4.2 Validation

Validation consists in assessing the performances of the physical model, here the RANS equations. We compare `Wolf` results to a reference CFD flow solvers, which are assumed to be valid and thus converge to the exact solution of the problem. The abilities of the Spallart-Almaras turbulence to represent different types of flows are well known and documented. However, even if the solver has been verified and converges to the exact solution, the initial error (the multiplying constant of the error) can depend on the implementation. It is thus an enlightening approach to run our own validation which confirms the proper coding of the equations. We compare here `Wolf` results to experiments and other CFD flow solvers.

The full verification and validation process can be found in [Menier 2014] and is summed up in Figure 1.27. From top to bottom, from left to right we can see:

- A 2D bump test case, pressure contours and velocity profiles comparison between `Wolf` and an other CFD flow solver (verification)
- A 3D irregular bump test case, surface pressure contours (verification)
- A 2D RAE test case, velocity contours and Reynolds stress profiles compared to an other

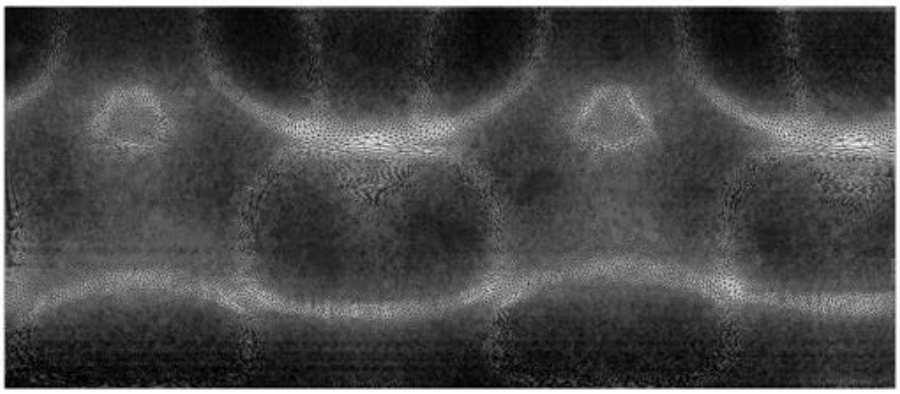


Figure 1.25: Manufactured solution: Adapted mesh generated with feature based adaptation on the Mach field.

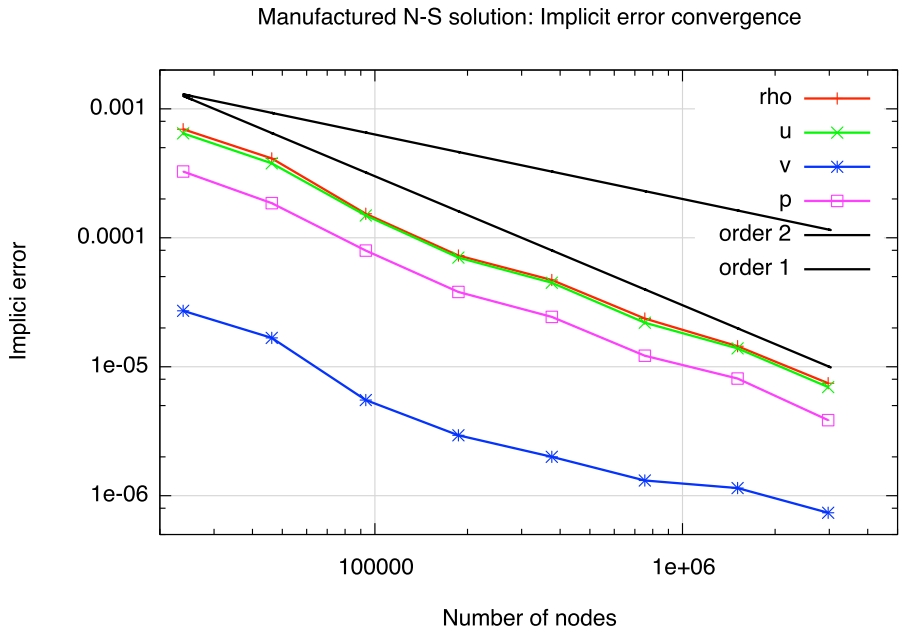


Figure 1.26: Manufactured solution: Convergence of the implicit error of the physical variables on adapted mesh generated with feature-based adaptation based on the Mach field.

CFD flow solver and experiment (Verification and Validation)

- Drag Prediction Wrokshop II configuration with nacelle, surface pressure contours and C_p/C_l

polar compared to experiment (Validation)

- Transonic Falcon, density contours accross different planes and convergence history (Validation)
- ONERA M6-Wing, surface density contours and pressure coefficients extraction compared to experiment (Verification and Validation)
- A 2D backward facing step test case, velocity contours and Reynolds stress tensor profiles compared to another CFD flow solver (Verification and Validation)
- High-lift workshop configuration, surface pressure contours and pressure coefficient extractions compared to another CFD flow solver and experiment (Validation).

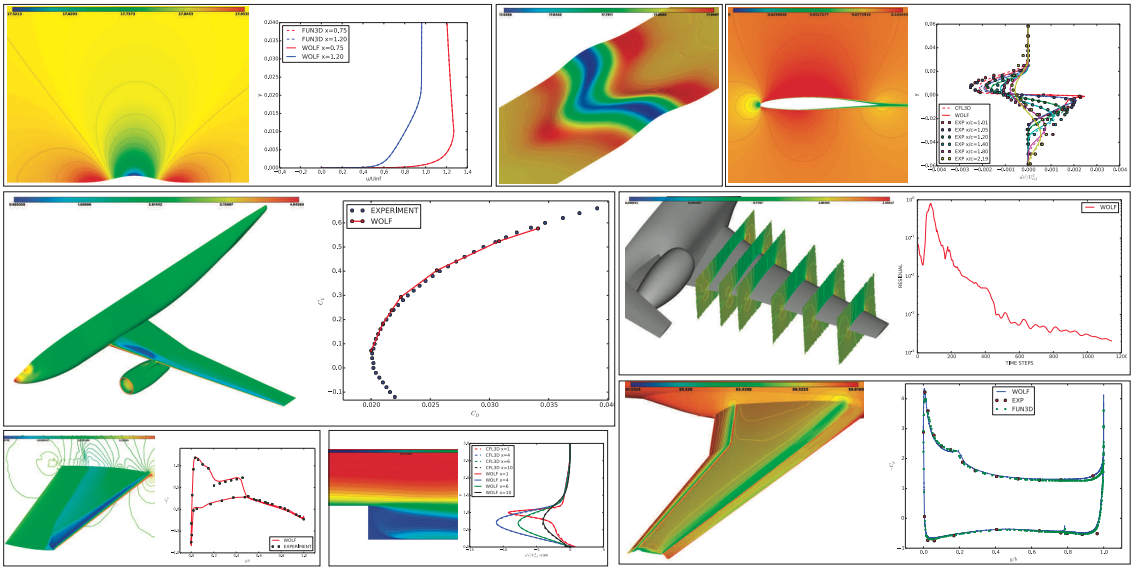


Figure 1.27: Verification and Validation test cases summed up.

Chapter 2

RANS solver description

We developed in previous chapter the resolution of the main flow in `Wolf`. We detail in this chapter the implementation of different versions of the Spalart-Allmaras equation. We emphasize on the modifications that have been made during this thesis to reach the same level of robustness in the turbulence resolution as in the main flow solver. This is what makes it possible to flawlessly solve the RANS equations on fully unstructured anisotropic adapted meshes. Additionally we introduce the implementations that have been made in the adjoint solver to compute the turbulent adjoint in the goal-oriented mesh adaptation context.

2.1 Turbulence modeling

According to the standard approach to turbulence modeling based upon the Boussinesq hypothesis, the turbulence is modeled with an eddy viscosity μ_t, which is added to the laminar (or dynamic) viscosity μ. The dynamic viscosity is usually taken to be a function of the temperature, whereas μ_t is obtained using a turbulence model. Here we choose the Spalart-Allmaras (SA) one equation turbulence model [Spalart 1992] given by the following equation:

$$\frac{\partial \tilde{\nu}}{\partial t}+\mathbf{u}\cdot\nabla\tilde{\nu} = c_{b1}[1-f_{t2}]\tilde{S}\tilde{\nu}-\left[c_{w1}f_w-\frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2+\frac{1}{\sigma}\left[\nabla\cdot((\nu+\tilde{\nu})\nabla\tilde{\nu})+c_{b2}\|\nabla\tilde{\nu}\|^2\right]+f_{t1}\Delta\mathbf{u}^2\,, \tag{2.1}$$

where $\tilde{\nu}$ is the turbulent kinematic viscosity and all the constants are defined below. In the standard model the trip term is being left out, *i.e.,* $f_{t1}=0$. Moreover, some implementations also ignore the f_{t2} term as it is argued that if the trip is not included, then f_{t2} is not necessary [Eca 2007]. In `Wolf`, this simplified version has been considered and we prefer to write it under the following form, which is more appropriate for its discretization with the finite element/finite volume method. Indeed, Equation (2.1) can be decomposed into the following terms:

$$\frac{\partial \rho\tilde{\nu}}{\partial t}+\underbrace{\mathbf{u}\cdot\nabla\rho\tilde{\nu}}_{convection} = \underbrace{c_{b1}\tilde{S}\rho\tilde{\nu}}_{production}\underbrace{-c_{w1}f_w\rho\left(\frac{\tilde{\nu}}{d}\right)^2}_{destruction}+\underbrace{\frac{\rho}{\sigma}\nabla\cdot((\nu+\tilde{\nu})\nabla\tilde{\nu})}_{dissipation}+\underbrace{\frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}\|^2}_{diffusion}\,.$$

Notice that this is not a conservative model. If a conservative form of the Spalart-Allmaras is foreseen, we have to consider the variation proposed by Catris and Aupoix [Catris 2000]. The turbulent eddy viscosity is computed from:

$$\mu_t=\rho\tilde{\nu}f_{v1}\,,$$

where

$$f_{v1}=\frac{\chi^3}{\chi^3+c_{v1}^3} \quad \text{and} \quad \chi=\frac{\tilde{\nu}}{\nu} \quad \text{with} \quad \nu=\frac{\mu}{\rho}\,.$$

Additional definitions are given by the following equations:

$$f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}} \quad \text{and} \quad \tilde{S} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2} \quad \text{where} \quad \Omega = \|\nabla \times \mathbf{u}\| .$$

d is the distance to nearest wall which is computed for each vertex at the beginning of the simulation. The set of closure constants for the model is given by

$$\sigma = \frac{2}{3}, \qquad c_{b1} = 0.1355, \quad c_{b2} = 0.622, \quad \kappa = 0.41,$$
$$c_{w1} = \frac{c_{b1}}{\kappa} + \frac{1+c_{b2}}{\sigma}, \quad c_{w2} = 0.3, \qquad c_{w3} = 2, \qquad c_{v1} = 7.1 .$$

Finally, the function f_w is computed as:

$$f_w = g\left(\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right)^{1/6} \quad \text{with} \quad g = r + c_{w2}\left(r^6 - r\right) \quad \text{and} \quad r = \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, 10\right) .$$

2.1.1 Spalart-Allmaras Negative Model

In the standard version of the SA turbulence model, the turbulent variable has to be limited in order to prevent it from becoming negative. This may slow the convergence of the overall process in some cases. In the Negative Spalart-Allmaras turbulence model, $\tilde{\nu}$ is allowed to become negative transitory. In that case, the following equation is solved instead:

$$\frac{\partial \rho\tilde{\nu}}{\partial t} + \underbrace{\mathbf{u}\cdot\nabla\rho\tilde{\nu}}_{convection} = \underbrace{c_{b1}\Omega\rho\tilde{\nu}}_{production} + \underbrace{c_{w1}\rho\left(\frac{\tilde{\nu}}{d}\right)^2}_{destruction} + \underbrace{\frac{\rho}{\sigma}\nabla\cdot((\nu+\tilde{\nu}f_n)\nabla\tilde{\nu})}_{dissipation} + \underbrace{\frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}\|^2}_{diffusion} .$$

with

$$f_n = \frac{c_{n1}+\chi^3}{c_{n1}-\chi^3}, \quad c_{n1} = 16.$$

Additionally, the turbulent eddy viscosity μ_t is set to zero when $\tilde{\nu}$ is negative.

Note that in the standard Spalart-Allmaras negative model the production term is written $c_{w1}(1-c_{t3})\rho\left(\frac{\tilde{\nu}}{d}\right)^2$, as the limit of the production term of the Spalart-Allmaras model, as $\lim_{\tilde{\nu}\to 0} f_{t2} = c_{t3}$. The c_{t3} term is thus ignored here to be consistent with the model when $\tilde{\nu} > 0$. Similarly, f_w in the destruction term is replaced by its limit in 0: $\lim_{\tilde{\nu}\to 0} f_w = 1$. Note that the destruction term is written with a "+" so that it becomes a production term and the f_n term is negative so that the term $\tilde{\nu}f_n$ remains positive.

2.1.2 Spalart-Allmaras Quadratic Constitutive Relation (SA-QCR)

Each turbulence model is known to fail at modelling some flows. In particular, the Boussinesq hypothesis, *i.e.* the assumed proportionality of the turbulent stress tensor and the strain tensor, is known to be false in some cases. For example, impinging jets or recirculating flows are classical counter example to this hypothesis. It is thus expected that turbulence models that relie on the Boussinesq hypothesis poorly predict such flows. In order to improve the prediction

of recirculating flows that appear at the junction of the body and the wing of an aircraft, [Spalart 2000] proposed the following Quadratic Constitutive Relation.

The turbulent equations are the same as for the standard SA turbulence model and the Boussinesq hypothesis is amended so that the turbulent stress tensor depends on the shear stress and rotation tensor through a quadratic relation:

$$\mathcal{T}_{QCR} = \mathcal{T}_{SA} - C_{cr1}\left(O \cdot \mathcal{T}_{SA}^T + \mathcal{T}_{SA} \cdot O^T\right)$$

where $\mathcal{T}_{SA}$ is the turbulent stress tensor of the standard SA turbulence model with the Boussinesq hypothesis, O is the normalized rotation tensor

$$O = 2\frac{\mathcal{W}}{\sqrt{\nabla\mathbf{u}^T : \nabla\mathbf{u}}},$$

and

$$\mathcal{W} = \frac{1}{2}\left(\nabla\mathbf{u} - \nabla\mathbf{u}^T\right).$$

The constant C_{cr1} is fixed to $C_{cr1} = 0.3$. Note that the matrix norm $\sqrt{\nabla\mathbf{u}^T : \nabla\mathbf{u}}$ expresses as

$$\sqrt{\nabla\mathbf{u}^T : \nabla\mathbf{u}} = \sqrt{\frac{\partial u_i}{\partial x_j}^2} = \sqrt{u_x^2 + u_y^2 + u_z^2 + v_x^2 + v_y^2 + v_z^2 + w_x^2 + w_y^2 + w_z^2}.$$

2.2 Spatial discretization

Following the same notations and definitions as in Section 1.1.2, the aforementioned models are discretized using the same mixed finite elements/finite volumes approach as follows.

Linear convection: The turbulent variable $\tilde{\nu}$ is linearly convected:

$$\Phi_{ij}^{\rho\tilde{\nu}}(W_i, W_j, \mathbf{n}_{ij}) = \begin{cases} \eta\ \rho\tilde{\nu}_i & \text{if} \quad \eta > 0 \\ \eta\ \rho\tilde{\nu}_j & \text{otherwise} \end{cases} \qquad \text{where} \quad \eta = \frac{1}{2}\left(\mathbf{u}_i \cdot \mathbf{n}_{ij} + \mathbf{u}_j \cdot \mathbf{n}_{ij}\right).$$

Discretization of the Spalart-Allmaras dissipation term: The Spalart-Allmaras dissipation term is discretized element-wise with the FEM as the main flow viscous terms:

$$\Phi_{visc,K}^{SA}(W_i, W_j, W_k, W_l) = |K|\ \frac{1}{\sigma}\rho_i\Big(\left(\nu|_K + \tilde{\nu}|_K\right)\nabla\tilde{\nu}|_K \cdot \nabla\phi_i|_K\Big),$$

where $\nu|_K$ and $\tilde{\nu}|_K$ are the mean values of the viscosity and turbulent eddy viscosity on element K and $\nabla\tilde{\nu}|_K$ the constant P_1 gradient of $\tilde{\nu}|_K$ on the element.

Discretization of the diffusion term: The diffusion term $\frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}\|^2$ can be discretized as a source term as

$$\Phi_{diffus,i}^{SA}(W_i) = \int_{C_i}\frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}\|^2 \approx |C_i|\frac{c_{b2}\rho}{\sigma}\|\nabla\tilde{\nu}_i\|^2,$$

where $\nabla\tilde{\nu}_i$ is the nodal gradient, obtained here with a L2 projection, in 2D:

$$\nabla\tilde{\nu}_i = \frac{1}{|C_i|}\sum_{K\ni i}\frac{\nabla\tilde{\nu}|_K}{3}$$

and in 3D:

$$\nabla\tilde{\nu}_i = \frac{1}{|C_i|}\sum_{K\ni i}\frac{\nabla\tilde{\nu}|_K}{4}.$$

Discretization of the source terms: Finally, the Spalart-Allmaras source terms (diffusion, production and destruction) are discretized by simple integration on each vertex cell:

$$\mathbf{Q}_i = |C_i| Q(W_i)\,,$$

where $|C_i|$ is the volume of the vertex cell.

2.2.1 Differentiation

These terms are then differentiated as follows to compute the turbulent part of the Jacobian.

Linear convection differentiation: The advection being linear we have:

$$\frac{\partial \Phi_{ij}^{\rho\tilde{\nu}}}{\partial \tilde{\nu}_i}(W_i, W_j, \mathbf{n}_{ij}) = \begin{cases} \eta\ \rho & \text{if} \quad \eta > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{where} \quad \eta = \frac{1}{2}\left(\mathbf{u}_i \cdot \mathbf{n}_{ij} + \mathbf{u}_j \cdot \mathbf{n}_{ij}\right)\,.$$

Source terms differentiation: As the source terms only depend on the considered nodal values, they are directly fully differentiated as

$$\frac{\partial \mathbf{Q}_i}{\partial \tilde{\nu}_i} = |C_i| \frac{\partial Q(W_i)}{\partial \tilde{\nu}_i}\,.$$

Approximate viscous terms differentiation: The easiest way to approximate the Jacobian of the viscous fluxes is by its spectral radius. The viscous flux accros each face (edge) is estimated as:

$$S_{ij} = \frac{\rho}{\sigma}(\nu + \tilde{\nu})\nabla\tilde{\nu} \cdot \mathbf{n}_{ij} \approx \frac{\rho}{\sigma}(\nu + \tilde{\nu})\frac{\tilde{\nu}_i - \tilde{\nu}_j}{x_i - x_j},$$

so that the diagonal and extra-diagonal contributions of the edge ij are taken to be

$$\frac{\partial S_i}{\partial \tilde{\nu}_i} \approx \frac{\rho}{\sigma}\frac{(\nu + \tilde{\nu})}{x_i - x_j} \quad \text{and} \quad \frac{\partial S_i}{\partial \tilde{\nu}_j} \approx -\frac{\rho}{\sigma}\frac{(\nu + \tilde{\nu})}{x_i - x_j}.$$

Exact viscous terms differentiation: An exact differentiation as for main flow viscous terms is also possible. We express the mean values and gradients of $\tilde{\nu}$ as the sum of the nodal values $\tilde{\nu}_m$ on the element:

$$\Phi_{visc,K}^{SA}(W_i, W_j, W_k, W_l) = |K|\ \frac{1}{\sigma}\rho_i \left(\left(\nu|_K + \frac{1}{d+1}\sum_{m\in K} \tilde{\nu}_m \right) \sum_{m\in K} \tilde{\nu}_m \nabla\phi_m|_K \cdot \nabla\phi_i|_K \right)$$

where d holds for the dimension, so that

$$\frac{\partial \Phi_{visc,K}^{SA}(W_i, W_j, W_k, W_l)}{\partial \tilde{\nu}_m} = \frac{|K|}{\sigma}\rho_i \left(((\nu|_K + \tilde{\nu}|_K)\ \nabla\phi_m|_K \cdot \nabla\phi_i|_K) + \left(\frac{1}{d+1}\ \nabla\tilde{\nu}|_K \cdot \nabla\phi_i|_K \right) \right)\,.$$

Although often used, the approximate differentiation of the viscous terms in the turbulent equations has a tremendous impact on the stability and the overall convergence of the system. Figure 2.1 shows the convergence history of the turbulent residual during the computation of a turbulent LS89 case at $CFL = 200$. The red curve shows the absence of convergence of the turbulent residual with an approximate differentiation of the viscous terms while the green curve shows that the turbulent system converges flawlessly with an exact differentiation. In that case, in order to achieve a convergence with an approximate differentiation we have to limit the CFL to 5 which significantly slows the convergence.

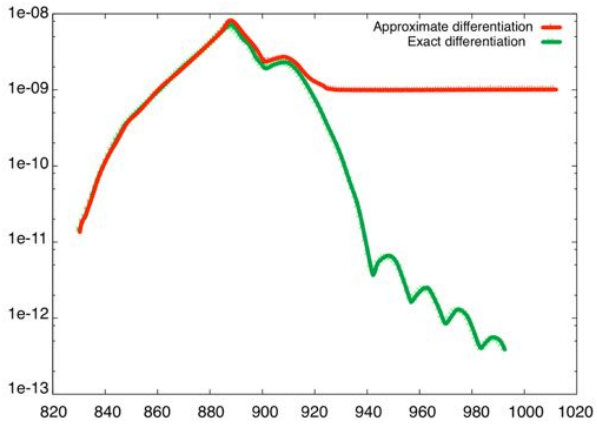


Figure 2.1: Convergence history of turbulent residual on a turbulent case at $CFL = 200$.

Differentiation of dissipation term: As for the viscous terms, the Jacobian of the dissipation terms can be approximated or even ignored, but this reduces the robustness of the process and the global CFL that can be used. It is thus interesting to properly differentiate this term. As the diffusion term involves gradients, it has to be differentiated element wise but we have to take into account how the nodal gradient is computed:

$$\Phi^{SA}_{diffus,i} = |C_i| \frac{c_{b2}\rho}{\sigma} \left(\frac{1}{|C_i|} \sum_{K \ni i} \frac{\nabla \tilde{\nu}|_K}{3} \right) \cdot \left(\frac{1}{|C_i|} \sum_{K \ni i} \frac{\nabla \tilde{\nu}|_K}{3} \right),$$

where we recall

$$\nabla \tilde{\nu}|_K = \sum_{j \in K} \tilde{\nu}_j \nabla \phi_j|_K.$$

We can see that this expression only involves the neighbours of the considered node so that the differentiation of $\Phi^{SA}_{diffus,i}$ with respect of any of these neighbouring nodes can be expressed as

$$\frac{\partial \Phi^{SA}_{diffus,i}}{\partial \tilde{\nu}_j} = 2 \frac{c_{b2}\rho}{\sigma} \nabla \tilde{\nu}_i \cdot \frac{1}{|C_i|} \sum_{K \ni (i,j)} \frac{\nabla \phi_j|_K}{3},$$

which can be computed element-wise

$$\frac{\partial \Phi^{SA}_{diffus,i,K}(W_i, W_j, W_k, W_l)}{\partial \tilde{\nu}_m} = \frac{2 c_{b2}\rho}{3\sigma |C_i|} \nabla \tilde{\nu}_i \cdot \nabla \phi_m|_K,$$

2.3 Wall Functions

Standard Wall Functions: Wall functions were initially developed for high-Reynolds number turbulence models which become invalid in the laminar part of the turbulent boundary layer. Standard logarithmic wall functions were derived from the turbulent Couette flow. It can be shown that in absence of pressure gradients the non-dimensional tangential velocity and distance to the wall expressed as:

$$u^+ = \frac{u}{u_\tau}, \quad \text{and} \quad y^+ = \frac{y u_\tau}{\nu}.$$

ν is the cinematic viscosity, $u_\tau = \sqrt{\tau_w/\rho}$ with τ_w the wall shear stress. u^+ and y^+ are related by a given function $u^+ = F^+(y^+)$ which is independent of the flow. It can be shown that this function behaves asymptotically near $y^+ = 0$ as

$$u^+ = F^+(y^+) \underset{y^+ \to 0^+}{\sim} y^+,$$

and to greater values

$$u^+ = F^+(y^+) \underset{y^+ \to +\infty}{\sim} \frac{1}{\kappa} ln(Ey^+),$$

where κ is the Von Karman constant and E a roughness parameter (for a smooth wall we have $\kappa = 0.42$ and $E = 9.0$). These relations are theoretically valid for a high-Reynolds Couette flow with no pressure gradient. However for many flows, these relations still hold and the first relation in 0^+ can be extended up to $y^+ = 5$ which is called the viscous sublayer (as turbulent viscosity is negligible). The second relation is called logarithmic-layer and considered to be valid from $y^+ \approx 20 - 30$ up to a distance y^+ which depends on the intensity of the pressure gradient and the Reynolds number. It is commonly admitted that a reasonable upper bound for good result should be $y^+ \approx 50 - 100$.

As standard high-Reynolds turbulence models successfully model the logarithmic layer, they are usually matched with wall functions at a given distance d from the wall, ideally chosen so that the corresponding dimensionless wall-distance $d^+ \approx 30$. The wall function is then supposed to model the inner part of the boundary layer (below $y^+ = 30$) and provide a coherent boundary condition to the flow. This is achieved with a Robin type boundary condition. Given the distance d and u the tangential velocity of the flow on the boundary, the nonlinear equation

$$\frac{u}{u_\tau} = \frac{1}{\kappa} ln\left(E\frac{du_\tau}{\nu}\right),$$

is solved for u_τ (with Newton iterations or dichotomy typically). The normal derivative of the flow field is imposed through the tangential stress tensor,

$$(\tau \cdot \mathbf{n}) \cdot \mathbf{t} = \tau_w = \rho u_\tau^2,$$

and turbulent quantities are deduced from u_τ and similar appropriate relations depending on the considered model.

These wall functions can similarly be used with low-Reynolds turbulence models. However, as usual flows always present pressure gradients, the logarithmic-law is never exactly verified so that standard wall functions provide unconsistent boundary conditions, leading to numerical results that actually depends on the chosen distance d. Moreover, although low-Reynolds turbulence models are valid at any distance of the wall, logarithmic wall functions cannot be used below $y^+ = 20$ as the logarithmic relation does not hold.

Universal and Model Specific wall functions: To overcome this problem, universal wall functions have been developed, providing an analytic profile valid throughout the boundary layer. However, as these analytic profiles do not match the exact profile of the turbulence model, they still provide inconsistent boundary conditions leading to wall distance dependent results (even with no pressure gradients). This motivation lead to the development of model specific wall

functions [Knopp 2006, Kalitzin 2005, Frazza 2015]. The main idea of this approach is that the dependency of the solutions to the distance used in the wall function is due to the inconsistency of the boundary condition imposed by the wall function. It was shown [Kalitzin 2005] that using tabulated values of an actual low-Reynolds turbulence model as wall function allowed to obtain the same results as wall integration, with no dependency to the wall distance chosen. In our case, we will use the analytical solution of the Spalart-Allmaras turbulence model [Allmaras 2012]:

$$\begin{aligned} u^+ &= 5.0333908790505579 \\ &+ 2.5496773539754747 \log((y^+ + 8.148221580024245)^2 + 7.4600876082527945^2) \\ &- 1.3301651588535228 \log((y^+ - 6.9287093849022945)^2 + 7.468145790401841^2) \\ &- 3.599459109332379 \operatorname{Arctan}(y^+ + 8.148221580024245, 7.4600876082527945) \\ &- 3.6397531868684494 \operatorname{Arctan}(y^+ - 6.9287093849022945, 7.468145790401841) \end{aligned}$$

where $Arctan(x, y)$ is the arctangent of y/x taking into account the quadrant in which the point (x, y) is located to yield the proper signed angle.

Both of these wall functions can be used at any wall distance below $y^+ = 50$. Both are also supposed to tend to wall integration as $d \to 0$.

Wall functions implementation: Numerically, the aim of wall functions is to modify the nodal residual near the boundaries in order to mimic the presence of the desired boundary layer. This is usually done either with an off wall implementation or a volume implementation as shown in Figure 2.2. In the off wall representation (left part of Figure 2.2) the computational domain is shifted at a distance d of the physical wall and the boundary layer region in red is computed by the wall function. To do so a consistent shear stress is applied to the fluid in the volume through the numerical boundary edge in blue. This method is relatively simple to implement and the chosen distance d is independent of the mesh. This guaranties the ability to yield converging computations toward an unique solution with high-Reynolds turbulence models when the mesh size decreases [Lacasse 2004]. Though, this off-wall distance d implicitly modifies the shape of the body as an additional flux virtually flows through this space, which can be undesired.

In the volume representation, the first layer of elements touching the boundary layer are flagged and an additional viscous flux is computed in the cells to take into account the turbulent boundary layer. In other words, the velocity field is assumed not to be linear in these element (for a $P1$ discretization) but instead to match the logarithmic profile of the turbulent boundary layer. The boundary edge in blue is thus coincident with the physical boundary and is a standard no-slip boundary condition. The major advantage of this approach is that it does not modifies the geometry of the body. However it yields severe restrictions on the mesh as the wall distance is fixed by the height of the first elements. For standard wall function this limits the minimal mesh size that can be used as the dimension less wall distance d^+ should not go below $d^+ = 30$. Moreover, this approach is relatively simple to implement for structured grids but it becomes cumbersome for unstructured grids (not to mention 3D cases).

In this work we use an off-wall prescription of wall functions, so that the wall distance d can be fixed independently of the mesh. However, we will either fix the distance d to a given

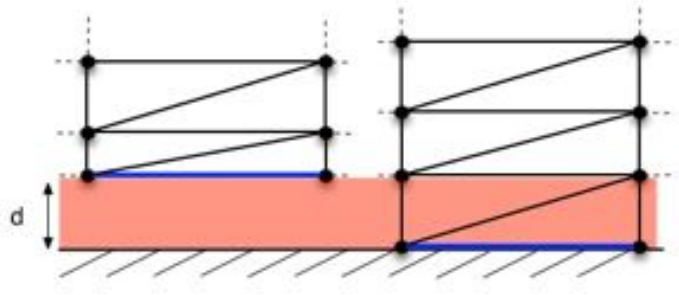


Figure 2.2: Off-wall implementation (left) and volume implementation (right) of wall functions

chosen value as in the traditional approach or fix it depending on the mesh size. In that case we compute the height of the elements connected to the surface and fix the distance d everywere to the mean value of the height of these elements (see Figure 2.3 right) or on each point to the minimal height of the elements to which it belongs (see Figure 2.3 left). Although this choice would be a constraint for standard wall functions, it enables here universal wall functions to degenerate to a wall-resolved computation as the mesh size decreases. Moreover this provides a relatively consistent treatment of the boundary condition depending on the mesh size: if the first elements size is about $y^+ = 1$ it is reasonable to use a standard wall-resolved model. On the contrary, if the first elements are too coarse it is better to use wall-functions.

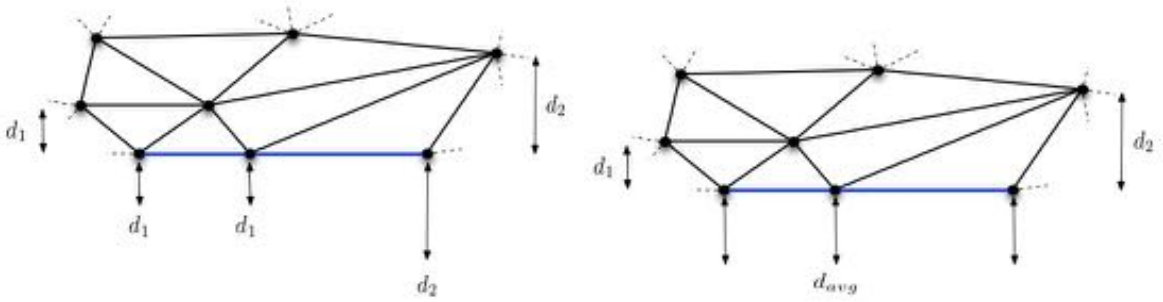


Figure 2.3: Local minimal off-wall distance computation (left) average off-wall distance computation(right)

We use here a two velocities scale [Lacasse 2004] wall function: instead of solving $u^+ = F^+(y^+)$ for u_τ, we determine a first velocity scale from the boundary value of ν_{SA} as

$$u_1 = \frac{\nu_{SA}}{\kappa d},$$

from which y^+ is determined as

$$y^+ = \frac{du_1}{\nu}.$$

The second velocity scale is deduced from

$$u_2 = \frac{u_t}{F^+(y^+)},$$

with $u_t = \|\mathbf{u} - (\mathbf{u}\cdot\mathbf{n})\mathbf{n}\|$ the tangential velocity, from which we can compute the friction stress tensor as

$$\tau_w = \rho u_1 u_2.$$

This implementation was initially proposed for heat transfer cases as it improves predictions near stagnation points by preventing y^+ from going to 0 as $u_t \to 0$. It is also straight forward to implement and does not require to solve a non-linear system.

From these quantities we deduce the boundary fluxes through the wall function boundary condition,

$$\begin{aligned}
\Phi_\rho &= 0 \\
\Phi_{\mathbf{u}} &= \tau_w \left(-\frac{\mathbf{u} - (\mathbf{u}\cdot\mathbf{n})\mathbf{n}}{\|\mathbf{u} - (\mathbf{u}\cdot\mathbf{n})\mathbf{n}\|} \right) \\
\Phi_e &= \tau_w \|\mathbf{u} - (\mathbf{u}\cdot\mathbf{n})\mathbf{n}\| \\
\Phi_{\nu_{SA}} &= \frac{2}{3}(\nu + \nu_{SA})\kappa u_1.
\end{aligned}$$

2.3.1 RANS solution initialization

The computation of RANS problems is significantly slower than inviscid flows. This is partly due to the boundary layer that is particularly slow to converge. When starting a computation from nothing, the choice of the initial field has thus a strong influence on the convergence of the problem. In order to improve this convergence, we use different initial fields. The idea is to compute at a very low expense an initial analytical field that will significantly improve the shape of the boundary layer.

Uniform field

The solver needs at least an initial physical field that has a positive density and a positive pressure. The basic initialization consist in setting the whole domain to a reference constant flow. This obviously produce a very steep (inexistent) boundary layer that will require initially a low CFL and many iterations to propagate.

Linear field

In order to ease this and be able to start with a higher CFL, we need to smooth the boundary layer. To do so we use a linear blending from the wall to the external flow. In order to further improve this initialization, we need to carefully chose the blending distance. Too small it will yield a too steep boundary layer. Too big it will produce the opposite problem, a boundary layer that takes a lot of time to get thinner.

To evaluate this optimal distance, we relie on boundary layer theory. The idea is that for bluff bodies, we will have a decent description of the growth of the boundary layer with a flat plate. This follows the same approach as for boundary layer mesh generation. We first identify a theoretical stagnation point, namely the leading point of the body. We can then use the empirical law giving the boundary layer thickness along the body

$$\delta(x) \approx \frac{0.16x}{(Re_x)^{1/7}},$$

where the Reynolds number Re_x is given by

$$Re_x = \frac{\rho U_\infty x}{\mu},$$

x being the distance of the considered point to the stagnation point.

Once the boundary layer thickness is determined, we specify a linear layer as

$$\mathbf{u} = \mathbf{U}_\infty \frac{d}{\delta(x)} \text{ for } d \leq \delta(x), \quad \mathbf{u} = \mathbf{U}_\infty \text{ for } d > \delta(x),$$

where d holds for the distance to the nearest wall, and similarly for the turbulent variable. Obviously, in presence of curved boundaries, this approach does not provide a flow aligned with the wall, but at least the profile is smooth.

This approach works well for bluff bodies in 2D with attached flow. Obviously it is false in presence of detachment and recirculation but it still provides a consistent initialization. It suffers yet from a major defect in 3D. On a 3D wing, the origin of the boundary layer ("stagnation point") should be taken on the leading edge of the wing, upstream of each point on the wing. Thus, contrary to 2D case, there is not only one stagnation point but a line of stagnation points to be identified. Moreover, these are not actual stagnation points as the fluid still flows sideways on the leading edge of the wing. It makes them even more difficult to identify.

If the leading point of the wing is chosen as origin, we will end up with a too thick boundary layer on the other side of the wing for a swept wing. It would be possible to consider a line or lines as origin, depending on the position of the considered point on the wing but then we would have to deal with the body, and any other element on the body beside. One can easily imagine the ridiculous amount of work it would require to compute an initial field for a high lift configuration with flaps and slats.

Logarithmic field

Following the same idea, and knowing that we deal with a turbulent boundary layer, we can impose the turbulent logarithmic profile. To do so we rely again on an empiric law, the Schlichting law, giving the friction coefficient with respect to the distance to the leading edge

$$C_f \approx [2\log_{10}(Re_x) - 0.65]^{-2.3}.$$

From this we deduce the wall shear stress

$$\tau_w = \frac{C_f \rho U_\infty^2}{2},$$

and the turbulent reference velocity

$$u_\tau = \sqrt{\frac{u_\tau}{\rho}}.$$

We can then compute the dimensionless wall distance

$$y^+ = \frac{d u_\tau}{\nu},$$

and the dimensionless profile

$$u^+ = y^+ \text{ for } y^+ < 10.9338028, \quad u^+ = \frac{\ln(y^+)}{0.41} + 5.1 \text{ for } y^+ \geq 10.9338028.$$

The dimensioned velocity profile is deduces as

$$u = \min(u_\tau u^+, U_\infty).$$

This approach provides a better boundary layer profile but suffers the same defects as the linear profile and thus cannot be applied to 3D cases.

Wall-Laws initialization

As we mentioned, the inner parts of the boundary layers are a limiting factor of the convergence of RANS cases, while the external flow converges very fast, as an inviscid flow. Meanwhile, the *ad-hoc* approaches presented above provide a rather good boundary layer profile but struggle to properly evaluate the friction coefficient on complex geometries. The idea here is to use wall functions to remove the inner parts of the boundary layers and precompute an initial external flow. This can be done either with a coarser mesh or by ignoring the elements below a given distance. As it is an external flow, this computation converges very fast.

The resulting solution can the be used to compute the local friction coefficient and compute the same boundary layer profile as for the logarithmic initialization. The advantage of this method is that it can deal with any geometry and detachment or recirculations. However, we must ensure it is not stuck by limiters or instabilities. Anyway, this computation has to be fast and does not requires to be well converged.

2.3.2 RANS adjoint

The extension of adjoint mesh adaptation to RANS applications requires the development of new error estimates exposed in Part 6.1.2. But these tools also requires the computation of the turbulent adjoint. We develop here the modifications that have to be made to the adjoint solver in order to compute this term.

Implementation

The adjoint system is a linearization of the non-linear primal equations and requires the inversion of the Jacobian of the problem. Hence, the precision of the result depends on the linearization. It is thus essential to use a Jacobian as accurate as possible. In particular, no mass matrix is added, so that the matrix is much stiffer, which requires the use of a Krylov-GMRES to inverse it.

`Wolf` was initially an inviscid flow solver, the turbulent part has been added later next to the main flow, so that the linear systems in the primal solver are segregated. This means that the turbulent system and mean flow are weakly coupled by the non-linear time steps. If such a structure is used in the adjoint system, we would entirely loose the coupling between both system of equations. In other words, the RANS adjoint would ignore the presence of RANS equations.

More precisely, if we consider the adjoint on the drag of a body, as $\mu_t = 0$ is null on the wall, the viscous contribution to the drag reads

$$\text{dDrad} \approx \mu \frac{\partial(\mathbf{u}\cdot\mathbf{t})}{\partial \mathbf{n}}\bigg|_0,$$

so that there is no contribution to the RANS part, and the right-hand side contribution is of the form (in 3D, 3 velocity components $*$)

$$Rhs_i = (0, *, *, *, 0, 0).$$

In the decoupled case the linear system has thus the form

$$W^* = \begin{bmatrix} \frac{\partial R_{\rho,u,v,w,p}}{\partial(\rho,u,v,w,p)} & 0 \\ 0 & \frac{\partial R_{\hat{\nu}}}{\partial \hat{\nu}} \end{bmatrix}^{-T} \cdot \begin{bmatrix} * \\ 0 \end{bmatrix}, \tag{2.2}$$

so that the turbulent components of the adjoint are null. It is thus essential to take into account the coupling between the turbulent system and the main flow.

As the matrix in the adjoint system is stored block-wise, we include the turbulent terms in the same block as the main flow and compute the coupling terms. Basically, the turbulent system influences the main flow through the turbulent viscosity, while the main flow influences the turbulent system with the advection and production terms. Note that the most important term to compute the adjoint is the turbulent viscosity as we want to know how the error in the turbulent equations influences the computation of the output.

Differentiation of turbulent viscosity: Directly we have

$$\mu_t = \rho\hat{\nu} f_{v1} = \rho\hat{\nu}\frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\hat{\nu}}{\nu}$$

so that

$$\frac{\partial \chi}{\partial \hat{\nu}} = \frac{1}{\nu}, \qquad \frac{\partial f_{v1}}{\partial \hat{\nu}} = \frac{3c_{v1}^3\chi^2}{\nu(\chi^3 + c_{v1}^3)^2},$$

and

$$\frac{\partial \mu_t}{\partial \hat{\nu}} = \rho\left(f_{v1} + \frac{3c_{v1}^3\chi^3}{(\chi^3 + c_{v1}^3)^2}\right). \tag{2.3}$$

We recall that the viscous fluxes are expressed as

$$\Phi_{i,K}^{visc}(W_i, W_j, W_k, W_l) = \int_{\partial C_i \cap K} S|_K \cdot \mathbf{n}\, \mathrm{d}\gamma,$$

with S expressed from

$$\mathcal{T} = (\mu + \mu_t)\left[(\nabla\otimes\mathbf{u} + {}^t\nabla\otimes\mathbf{u}) - \frac{2}{3}\nabla.\,\mathbf{u}\,\mathbb{I}\right]$$

so that we can compute

$$\frac{\partial \Phi_{i,K}^{visc}}{\partial \hat{\nu}}(W_i, W_j, W_k, W_l) = \int_{\partial C_i \cap K} \frac{\partial \mu_t}{\partial \hat{\nu}}\left[(\nabla\otimes\mathbf{u} + {}^t\nabla\otimes\mathbf{u}) - \frac{2}{3}\nabla.\,\mathbf{u}\,\mathbb{I}\right]\cdot\mathbf{n}\,\mathrm{d}\gamma. \tag{2.4}$$

We expressed above the momentum equation part of the viscous fluxes, the energy equation part expresses as well.

Differentiation of the production term: The production term is integrated vertex-wise and reads

$$\rho\hat{\nu}c_{b1}\left(\Omega+\frac{\hat{\nu}}{\kappa^2\tilde{d}^2}f_{v2}\right).$$

As

$$\Omega=\sqrt{\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right)^2+\left(\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x}\right)^2+\left(\frac{\partial v}{\partial z}-\frac{\partial w}{\partial y}\right)^2}$$

derives from gradients, we will assume it to be computed element-wise and derive it so. We recall that on each elements the solution is discretized with P_1 basis functions so that

$$W|_K=\sum_{i\in K}W_i\Phi_i|_K,\qquad \nabla W|_K=\sum_{i\in K}W_i\nabla\Phi_i|_K,\qquad \frac{\nabla W|_K}{\partial W_j}=\nabla\Phi_j|_K.$$

Hence, we have

$$\frac{\partial(\rho c_{b1}\hat{\nu}\hat{S})_i}{\partial\rho_i} = c_{b1}\hat{\nu}\hat{S}, \tag{2.5}$$

$$\frac{\partial(\rho c_{b1}\hat{\nu}\hat{S})_i}{\partial u_j} = \rho c_{b1}\hat{\nu}\frac{\frac{\partial\phi_j}{\partial y}\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right)+\frac{\partial\phi_j}{\partial z}\left(\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x}\right)}{\Omega}\frac{|K|}{|C_i|}, \tag{2.6}$$

$$\frac{\partial(\rho c_{b1}\hat{\nu}\hat{S})_i}{\partial v_j} = \rho c_{b1}\hat{\nu}\frac{-\frac{\partial\phi_j}{\partial x}\left(\frac{\partial u}{\partial y}-\frac{\partial v}{\partial x}\right)+\frac{\partial\phi_j}{\partial z}\left(\frac{\partial v}{\partial z}-\frac{\partial w}{\partial y}\right)}{\Omega}\frac{|K|}{|C_i|}, \tag{2.7}$$

$$\frac{\partial(\rho c_{b1}\hat{\nu}\hat{S})_i}{\partial w_j} = \rho c_{b1}\hat{\nu}\frac{-\frac{\partial\phi_j}{\partial x}\left(\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x}\right)-\frac{\partial\phi_j}{\partial y}\left(\frac{\partial v}{\partial z}-\frac{\partial w}{\partial y}\right)}{\Omega}\frac{|K|}{|C_i|}, \tag{2.8}$$

where the first term can be computed vertex-wise, and the three other have to be computed element-wise.

Differentiation of the advection term: We recall that the turbulent variable is advected linearly edge-wise as

$$\Phi_{ij}^{\rho\tilde{\nu}}(W_i,W_j,\mathbf{n}_{ij})=\begin{cases}\eta\ \rho\tilde{\nu}_i & \text{if}\quad \eta>0\\ \eta\ \rho\tilde{\nu}_j & \text{otherwise}\end{cases}\qquad\text{where}\quad \eta=\frac{1}{2}\left(\mathbf{u}_i\cdot\mathbf{n}_{ij}+\mathbf{u}_j\cdot\mathbf{n}_{ij}\right),$$

so that we immediately have

$$\frac{\partial\Phi_{ij}^{\rho\tilde{\nu}}}{\partial u_k}(W_i,W_j,\mathbf{n}_{ij})=\begin{cases}\frac{1}{2}\mathbf{n}_x\ \rho\tilde{\nu}_i & \text{if}\quad \eta>0\\ \frac{1}{2}\mathbf{n}_x\ \rho\tilde{\nu}_j & \text{otherwise}\end{cases}\qquad\text{where}\quad \eta=\frac{1}{2}\left(\mathbf{u}_i\cdot\mathbf{n}_{ij}+\mathbf{u}_j\cdot\mathbf{n}_{ij}\right),$$

and similarly for v and w.

Validation

We can think of different options to verify the validity of the implementation of the adjoint. As the discrete adjoint should converge toward the continuous adjoint, we can use manufactured

solutions for the adjoint problem and perform a grid convergence study as for the primal problem. Similarly, the adjoint should provide a linear correction of the functional output through the estimation of the residual truncation error. Using a manufactured solution to prescribe an analytical solution of the primal problem, we can perform a grid convergence study and verify that the correction is accurate. However, these approaches require extensive developments that we have performed in 1D only.

A basic way to verify if the adjoint has been implemented without errors is to test its dimension independency. The results predicted by a flow solver have to be the same, no matter the consistent system of unity used to solve them (International system, Imperial system, ...). In particular, dimensionless quantities such as drag or lift coefficient and mach number have to be left unchanged, while the numerical values associated to the velocity (for example) varies from three orders of magnitude, whether they are expressed in $m \cdot s^{-1}$ or $mm \cdot s^{-1}$.

The adjoint is defined by the relation

$$dJ = W^* \cdot dR,$$

where dJ is the variation of an output functional, W^* is the adjoint and dR a residual variation. Hence, if the output functional considered is dimensionless, dJ should be the same, no matter the system chosen. Meanwhile, as the residual is dimensioned dR will depend on the system chosen so that W^* will have to scale oppositely. Furthermore, as the numerical result should be independent of the system and the variables are scaled by a fixed factor, the nodal product should verify

$$[\,(W^*_\rho)_i \,\frac{\rho_i}{t}\,]^{\text{System 1}} = [\,(W^*_\rho)_i \,\frac{\rho_i}{t}\,]^{\text{System 2}}.$$

All variables should also verify the same relation.

Chapter 3

Periodic computations for turbomachinery applications

In this chapter we specifically detail the various modifications that have been made in `Wolf` to compute periodic rotating flows and simulate turbomachinery applications.

3.1 Implementation

In the case of periodic flows, we assume that the solution flow has a given periodicity (*i.e.* it is invariant by a given geometrical transformation $u(T(x)) = u(x)$) due to periodic physical properties. We will consider translation and rotation periodicities. We can then restrain computation to (at least) a single period and emulate the rest of the domain with periodic boundary conditions. The domain is thus delimited by an arbitrary surface Γ_{per} and its periodic mapping $\Gamma'_{per} = T(\Gamma_{per})$. In the periodic mesh, surfaces Γ_{per} and Γ'_{per} are meshed identically, thus there is a one-to-one mapping between their vertices and faces. As the periodicity is enforced in the discretization so that each vertex on Γ_{per} has a linked vertex on Γ'_{per}, there is no solution interpolation on the boundaries.

To enforce strongly the periodicity, we use ghost or virtual entities (vertices and elements). Indeed, vertices on periodic boundaries have initially only a part of their geometric stencil (*i.e.,* a part of their geometric entities ball) represented inside the mesh of the domain. The other part of the geometric stencil is on the other side of the domain around the linked vertices. As these vertices have to be treated like inner vertices, their geometric stencil are completed with ghost/virtual entities (vertices and elements) using the geometric stencil of their linked periodic vertices. Two choices in the implementation are possible:

- virtual entities are linked on existing entities on the other side of the domain using the linked vertex. This is memory efficient because no extra-entities are created, just a memory link is used. However, such an implementation impacts all the source code. Thus, each new implemented functionality should take care of the periodicity

- ghost entities are created inside the mesh. Then, there is a little memory overhead but in that case periodic vertices are computed similarly to inner vertices. This clearly facilitates the implementation and the impact of the periodicity in the code is localized in a few number of functions.

In `Wolf`, we choose the second method. Therefore, the initial mesh (Figure 3.1 left) is completed at the beginning of the simulation with ghost entities as can be seen in Figure 3.1 right.

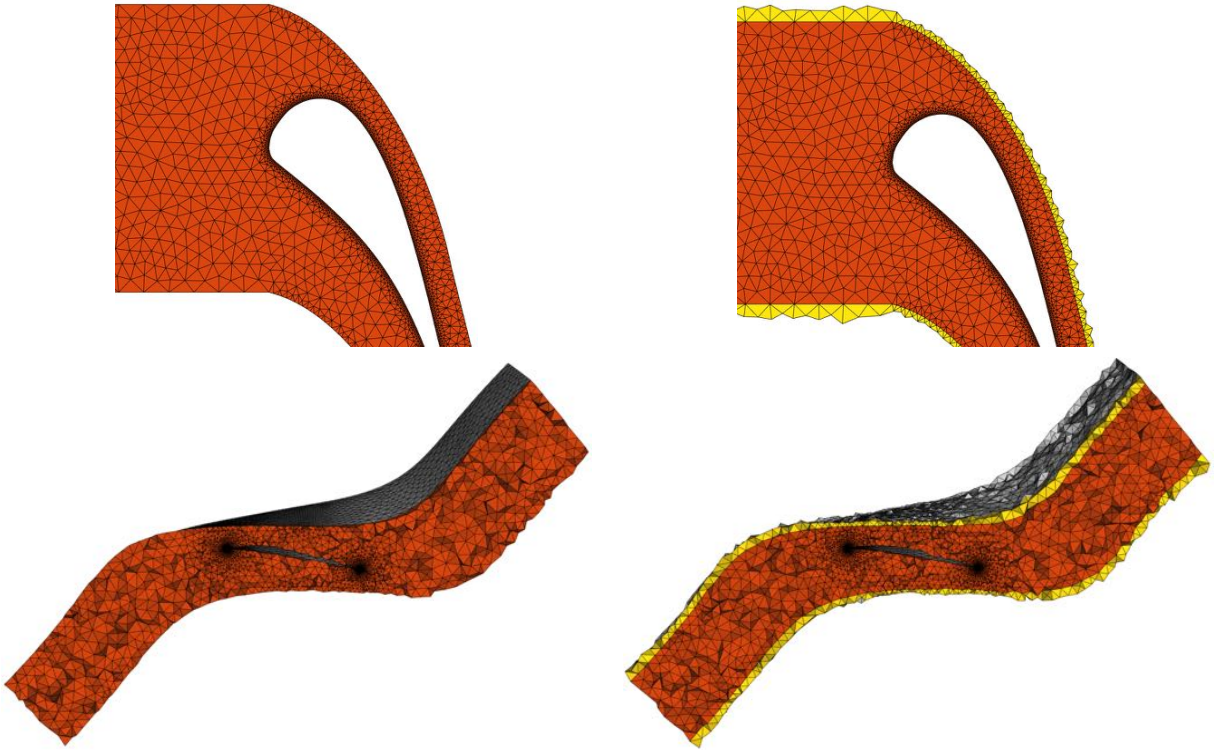


Figure 3.1: Periodic domains without (left) and with (right) ghost entities. Inner element in orange, ghost elements in yellow. Example on the 2D LS89 case (top) with vertical translation periodicity and the 3D RO37 case (bottom) with axial rotation periodicity.

Time explicit resolution

For explicit time integration only inner nodes fluxes are required. The solution at ghost nodes is copied at the beginning of each iteration from their inner counterparts before computing gradients and fluxes.

Time implicit resolution

Theoretically, with an implicit resolution, inner nodes fluxes should depend directly on inner nodes values and ghost nodes should not play any role. Thus, the matrix should be corrected and ghost nodes not included. As we assume (for translation periodicity)

$$u_i^{ghost} = u_i$$

we know that

$$\frac{\partial R_j}{\partial u_i} = \frac{\partial R_j}{\partial u_i^{ghost}}$$

which allows to appropriately correct the matrix by simply copying the ghost coefficients in their paired nodes coefficients.

For sake of simplicity, we chose not to take into account directly the periodicity in the linear system. Periodic vertices are treated as Dirichlet points with an imposed variation of the solution. As for standard Dirichlet points their contribution to the Jacobian is fixed to identity (see Section 1.1.5), but instead of being fixed to 0, the right hand side is equal to the variation of the solution on the periodic vertex. The solution variation δW is thus copied, at each SGS iteration, from the inner vertices to the ghost vertices through the right hand side. This weakens the coupling through the periodic frontier but still takes it into account.

The correction in the matrix is here taken into account iteratively, but it is strictly equivalent. Indeed, at convergence, we have

$$\frac{\partial R_i}{\partial u_j}\delta u_j = -R_i,$$

so that for an inner point P_i connected to a ghost point P_j^{ghost} we have

$$\frac{\partial R_i}{\partial u_j^{ghost}}\delta u_j^{ghost} = -R_i.$$

As $\delta u_j^{ghost} = \delta u_j$ has been copied and converged, we have

$$\frac{\partial R_i}{\partial u_j^{ghost}}\delta u_j = -R_i,$$

and as by construction $\frac{\partial R_i}{\partial u_j^{ghost}} = \frac{\partial R_i}{\partial u_j}$, we actually verify the periodicity in the linear system.

3.2 Periodic Adjoint

We also have to take into account the periodicity in the adjoint resolution. In the main solver we were able to simplify the matrix contribution by the use of ghost vertices. However, this rely on a direct modification of the linear system solver in `Wolf`. As we use another linear system solver in the adjoint resolution, using the same approach would require to apply the same modifications in the external library. Hence, we choose to properly take into account the periodicity in the matrix instead for adjoint resolution.

Figure 3.2 shows the periodic computational domain of a LS89. Vertex C is the periodic of A and ghost vertex D is the periodic of B. The periodicity hypothesis implies nonetheless that D has the same value as B but also that C has the same value as A. Hence, even though only D is generated as ghost vertex, C should be considered as well. Additionally, as the nodal values of D are copied from B, the residual of A actually depends on B.

Matrix construction: Contrary to the main solver, the full periodic matrix is assembled in the adjoint system. Periodic vertices are first identified (D is periodic of B, C is periodic of A, ...) and the proper connectivity is reconstructed:

- Any periodic vertex is considered to be connected to no one: the line associated to C and D in the matrix are fixed to identity and the entries linking A to D and B to C are set tot 0 (not allowed).

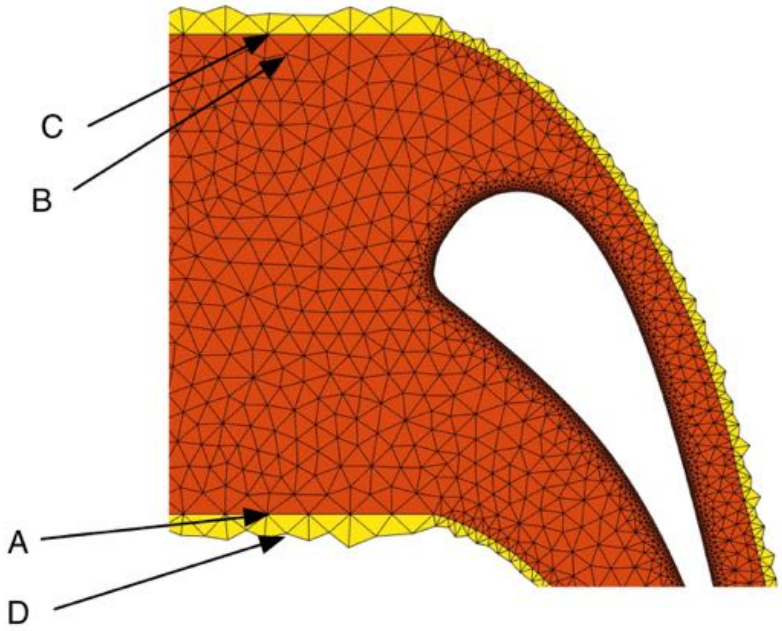


Figure 3.2: Ghost vertices and elements in a periodic domain.

- If an inner vertex is connected to a periodic vertex (like A and D), then it is reconnected to the proper inner periodic vertex (A is reconnected to B).

As the matrix is stored in a sparse CSR structure in the adjoint resolution, it simply consist in changing indices. The same treatment is applied when the matrix is assembled.

Periodic terms computation: In order to properly compute the periodic terms in the matrix we use the ghost structure. Namely, we are able to compute $\frac{\partial R_A}{\partial W_D}$ and we want to deduce $\frac{\partial R_A}{\partial W_B}$. We have to distinguish two cases here. In the translation case, the nodal values (density, velocity and pressure) are copied from node B to node D so that

$$\frac{\partial W_D}{\partial W_B} = Id,$$

and thus we can deduce the composition

$$\frac{\partial R_A}{\partial W_B} = \frac{\partial R_A}{\partial W_D}\frac{\partial W_D}{\partial W_B} = \frac{\partial R_A}{\partial W_D}.$$

The term $\frac{\partial R_A}{\partial W_D}$ being simply computed on the ghost part of the mesh, the block matrix can then be directly copied. In the rotation case, the density and the pressure are still copied but the velocity vector has to be rotated as $\mathbf{u}_D = \mathcal{R}\mathbf{u}_B$, so that we have

$$\frac{\partial W_D}{\partial W_B} = \begin{bmatrix} 1 & & \\ & \mathcal{R} & \\ & & 1 \end{bmatrix}.$$

The term $\frac{\partial R_A}{\partial W_D}$ can still be computed on the ghost part of the mesh, but we then have to apply a rotation to the block matrix before assembling it in the periodic contribution.

In practice, we can thus use the same routines as for the standard solver, except that when we assemble the matrix, we check the periodicity of the vertices. We then reconnect properly the inner vertices and rotate the matrix contributions if needed.

3.3 Applications

3.3.1 LS89 case

Case description

The LS89 Von Karman Institut's geometry is considered here in two regimes described in Table 3.1. The first one corresponds to the MUR35 case in the original publication, while the other one is a slightly different variant. The gaz is described with the perfect gaz law with $r = 287.04$, $\gamma = 1.4$, and the viscosity follows the Sutherland law with $\mu_0 = 1.716 \times 10^{-5}$, $T_0 = 273.15$ K and $S = 110.4$.

The initial solution is a constant flow in x direction (same as inflow). The first computation is initiated with the first order scheme and pursued with the second order one. Three second order schemes have been successfully tested, with third-, fourth- and sixth-order viscous dissipation, respectively with VanAlbada, Piperno and Koren limiters.

The first case remains subsonic while the second one exhibits a slightly supersonic zone with a shock as can be seen on the Mach isocontours in Figure 3.3.

	Case 1	Case 2
Inflow Total Pressure	$1.828 \times 10^5\ Pa$	$1.828 \times 10^5\ Pa$
Inflow Total Temperature	$413.3\ K$	$413.3\ K$
Outflow Pressure	$0.65\ P_t^{in} = 1.1882 \times 10^5\ Pa$	$0.57\ P_t^{in} = 1.04196 \times 10^5\ Pa$

Table 3.1: LS89 states description for each test case.

Geometry issues

The LS89 geometry provided by the Von Karman Institute is defined by a collection of points measured on the blade's surface. It contains uncertainties and numerical errors. When a spline or a curve is fitted through these points, even with a smoothing, some oscillations appear on the surface geometry. These oscillations of the geometry are invisible but produce spurious oscillations in the solution as can be seen on the pressure and friction coefficients extractions in Figure 3.4. In an adaptative procedure, these oscillations of the solution are detected by the error estimates which focuses on them by refining these regions.

In order to verify this issue, we "smoothed" the geometry by retaining one point every 150 points. This leads to a slightly different geometry shown in Figure 3.6 which produces smoother result as exhibited in Figure 3.5. Although it leads to a different result, we retained

the smoothed geometry for the present mesh adaptation study in order to have analyzable results, without oscillations and multiple shocks.

Note that in all this thesis, the algebraic friction coefficients are plotted in 2D. The boundary of each body being a closed curve, it is oriented clockwise and the local contribution to the aero-forces is projected on the local oriented tangential direction to obtain the signed friction coefficient. Hence, on the LS89 case, the top side of the blade is oriented in the same direction as the flow resulting in a positive friction coefficient, while the bottom side of the blade is oriented in the opposite direction, resulting in a negative friction coefficient. This convention has been kept to ease the reading of the "Cf" plots.

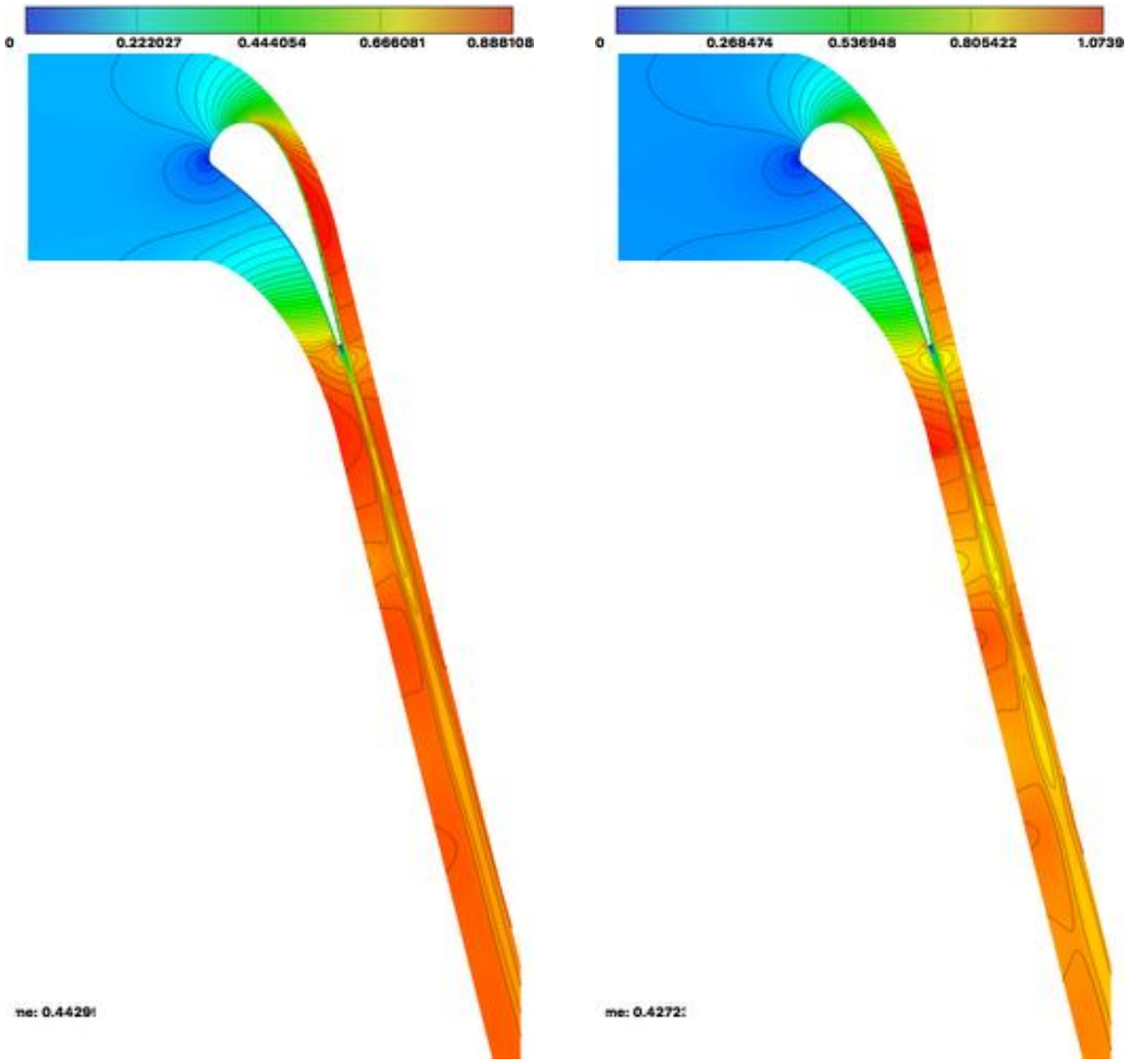


Figure 3.3: LS89 case 1 (left) and case 2 (right) Mach isosontours.

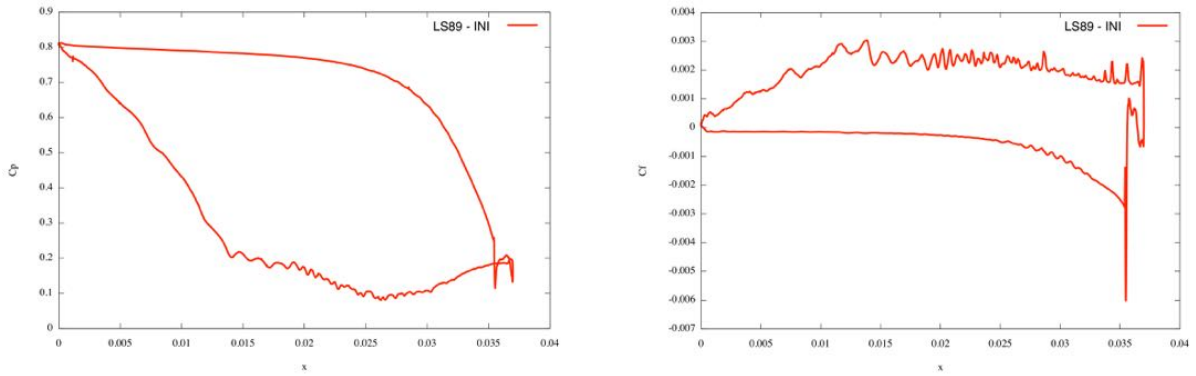


Figure 3.4: LS89 case 1. Skin pressure (left) and skin friction (right) coefficients extractions on the initial geometry.

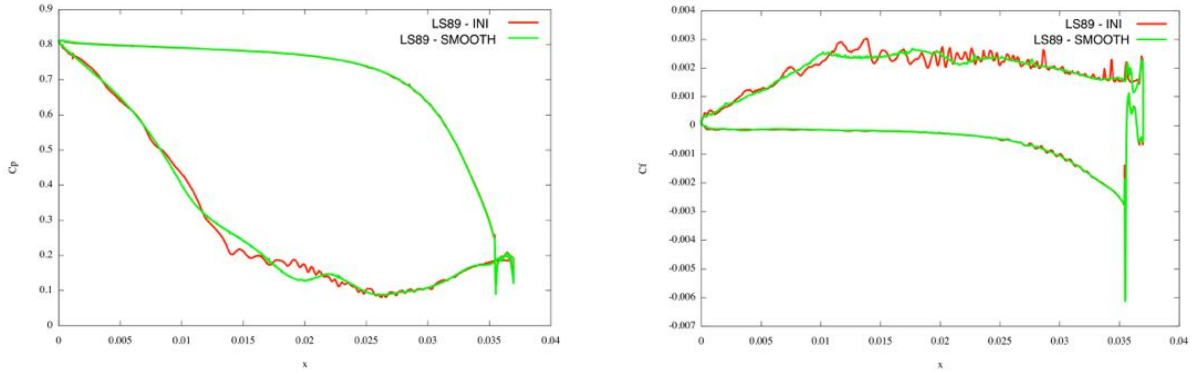


Figure 3.5: LS89 case 1. Comparison of the skin pressure (left) and skin friction (right) coefficients extractions on the initial (red) and the smoothed geometry (green).

Mesh Convergence

In order to study the convergence of the flow solver, we generated a serie of meshes for both geometries. These meshes were automatically generated using the metric-aligned method [Marcum 2014], according to the metric of the initially hand-made mesh. As the geometry of this case is pretty simple, these meshes are of good quality, the boundary layer and wake are well discretized. The main difference with adapted meshes presented in the next section lies in the shock region which is not known a priori, thus, not refined in the initial hand-made mesh.

Convergence of the error in L^2 norm for both cases and geometries are shown in Figure 3.7. The second order scheme with a fourth-order numerical dissipation and the Piperno limiter have been used for this convergence study. The first computation on the coarsest mesh is initiated with the first order scheme and pursued at second order. The initial solution is a constant flow in x direction (same as inflow). Then, initial solution, on finer meshes are linearly interpolated from the previous mesh final result which enables to start immediately with the second-order scheme.

We can see that all variables converge at second order. The error is defined here as the L^2

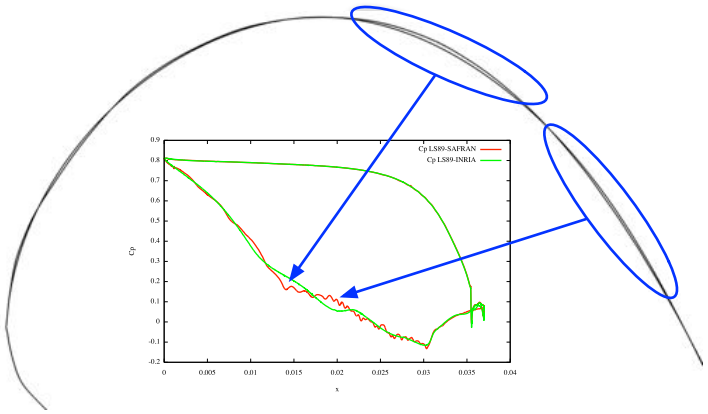


Figure 3.6: LS89 case 1. Difference between the initial and the smoothed geometry and its impact on the solution.

norm of the difference between the solution on the current mesh and the solution on the finest mesh.

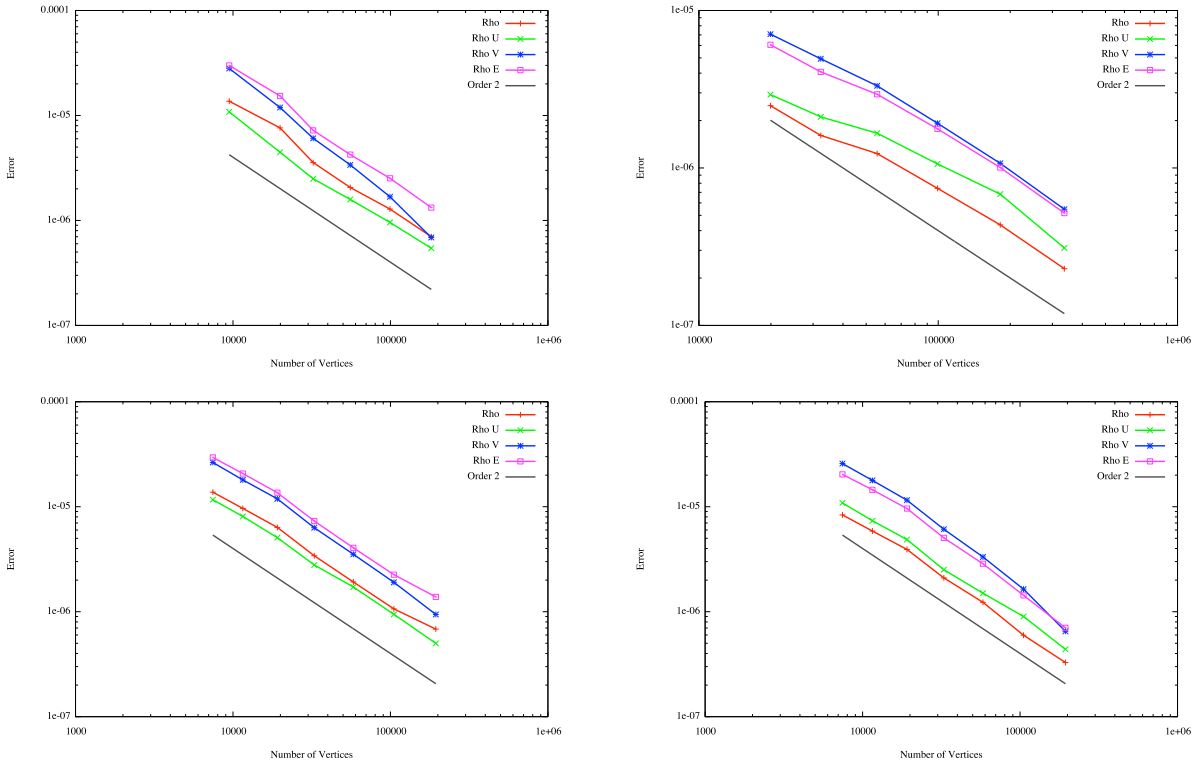


Figure 3.7: LS89 cases. L^2 norm convergence on non-adapted meshes. Initial geometry (top) and smoothed geometry (bottom) for case 1 (left) and case 2 (right).

Mesh adaptation applied to LS89

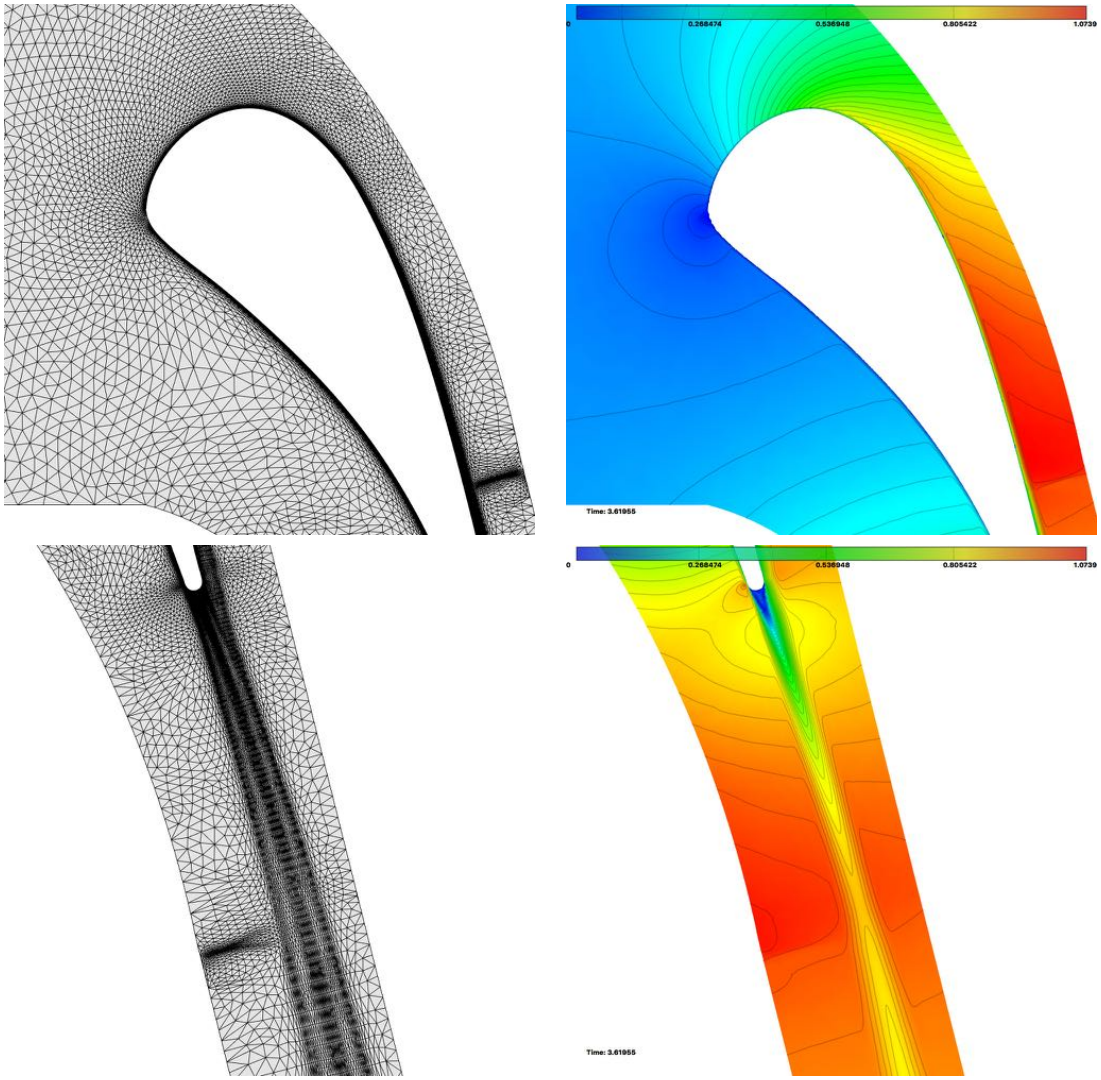


Figure 3.8: LS89 case 2. Details of the adapted meshes (left) and Mach contours (right) for the second test case on LS89 geometry around the blade (top) and in the wake (bottom).

The LS89 geometry is relatively simple, it is thus easy to accurately discretised by hand the boundary layer and the wake in order to have a fairly good mesh. This is why we retained here the second case, with a shock. This shock significantly influences the debit but has no *a-priori* known position. It is thus difficult to define an appropriate mesh without knowing the solution.

We choose here the Mach field as sensor. The mesh is iteratively adapted, starting from an initial uniform mesh composed of 5 000 points, the complexity is multiplied by two at each step. Three sub-iteration for each complexity are performed to converge the mesh and solution at this given complexity. Vertices are reprojected on the blade geometry, while periodic frontiers are left unchanged. The second-order scheme with fourth-order numerical dissipation and the

Piperno limiter have been used here with Barth cells.

Figure 3.8 shows an overview of the resulting mesh with 31.782 vertices and the associated Mach solution. The boundary layer, the wake and the shock have been appropriately and automatically refined. The shock is well discretized and sharp, it successfully goes through the periodic boundary and interacts with the wake.

A zoom on the trailing edge (see Figure 3.9) reveals more details on how the mesh is adapted to the solution, and a zoom on the shock (see Figure 3.10) shows, how the metric-aligned method generates automatically, structured mesh in the boundary layer and the shock regions. It is worth noting the abrupt size transition (leading to poor quality elements) in the shock region, near the periodic boundary, due to the fact that periodic frontiers are left unchanged. As the initial discretization of the periodic boundary is too coarse, with respect to the size required to discretize the shock, the shock is diffused accros the periodic frontier. This is visible on the Mach contours in Figure 3.10.

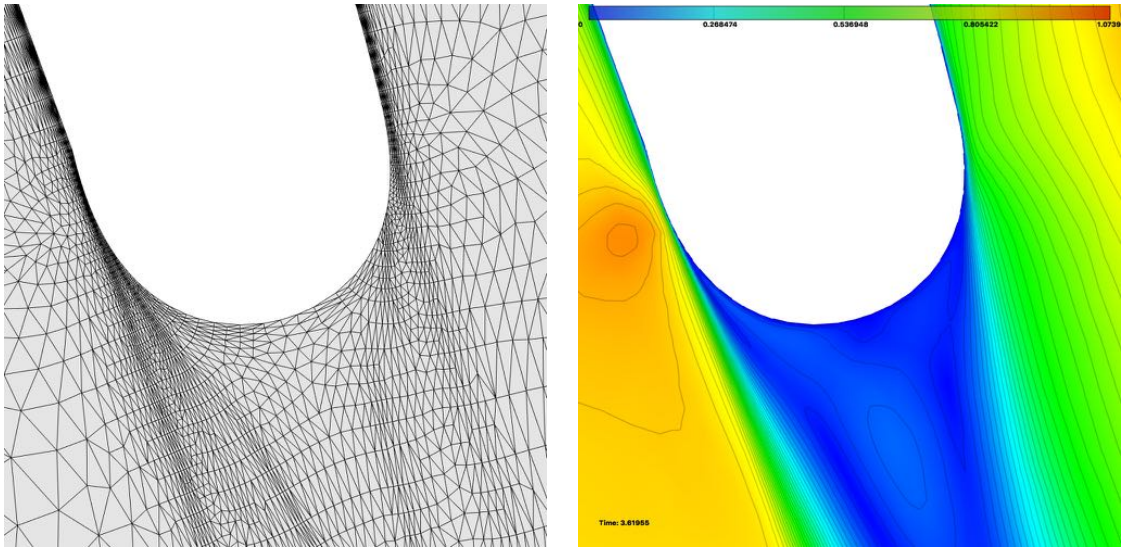


Figure 3.9: LS89 case 2. Zoom on the mesh (left) and Mach contours (right) for second test case one LS89 geometry in the blade trailing edge .

LS89 adjoint mesh adaptation.

We now generate adapted meshes with the goal-oriented strategy developed in Section 6.1, with the implementation described in Section 3.2 for the second LS89 test case. As the LS89 represents a turbine guide vane, the thermal fluxes on the blade are often a quantity of interest. We used here adiabatic wall, so we have no thermal boundary fluxes but we focused on friction coefficients which also require a fine discretization near the walls. To do so, we choose the lift of the blade as functional output. Contrary to the previous study, we were able to adapt the periodic boundaries with the approach presented in Section 7.1, we thus used here a more traditional domain. The correct implementation of periodic boundary condition in both primal and adjoint solver are thus even more important to properly deal with the wake of the blade.

Figure 3.11 shows the adjoint fields computed. They are presented on the domain with ghost elements. The nodal values of the ghost vertices are simply copied from their periodic counterpart, showing that the adjoint is correctly implemented and periodic. Though, we can note the discontinuity of the adjoint variables on the inflow and outflow boundary conditions. In the facts, the discrete adjoint is known to exhibit discontinuities on the boundaries in various cases [Venditti 2002a] and according to [Huang 2018] this may be due here to the absence of viscous contributions in the inflow/outflow boundary conditions. Indeed, we assumed the flow to be inviscid and independent of the normal direction in Section 1.1.3. This inconsistency has very little impact on the primal solution but may be responsible for these discontinuities in the adjoint near the inflow/outflow.

Figures 3.12 and 3.13 show respectively a general view and a zoom on the trailing edge of the blade. These figures show the adapted meshes and Mach fields obtained with feature-based adaptation using the Mach field as sensor and goal-oriented mesh adaptation based on the skin friction of the blade. We can see that while feature based adaptation refines all features of the flow (boundary layer, wake and shock), the adjoint based adaptation immediately focuses on the boundary layer and the blade in general. Hence, as can be seen in Figure 3.12, the wake of the blade is less discretized and thus diffused, but it has little effect on the lift of the blade. Note here that as the adjoint is periodic, the effect of the wake of the blade on its own lift is taken into account but also on the other blades through the periodic boundary. Put the other way, we take into account the contribution of all blades on the lift of the considered blade. As computations are done at the same complexity, elements that are saved in the wake are spend to discretize finely the boundary layer in the adjoint case. This leads to an early capture of the skin friction as shown in Figure 3.14.

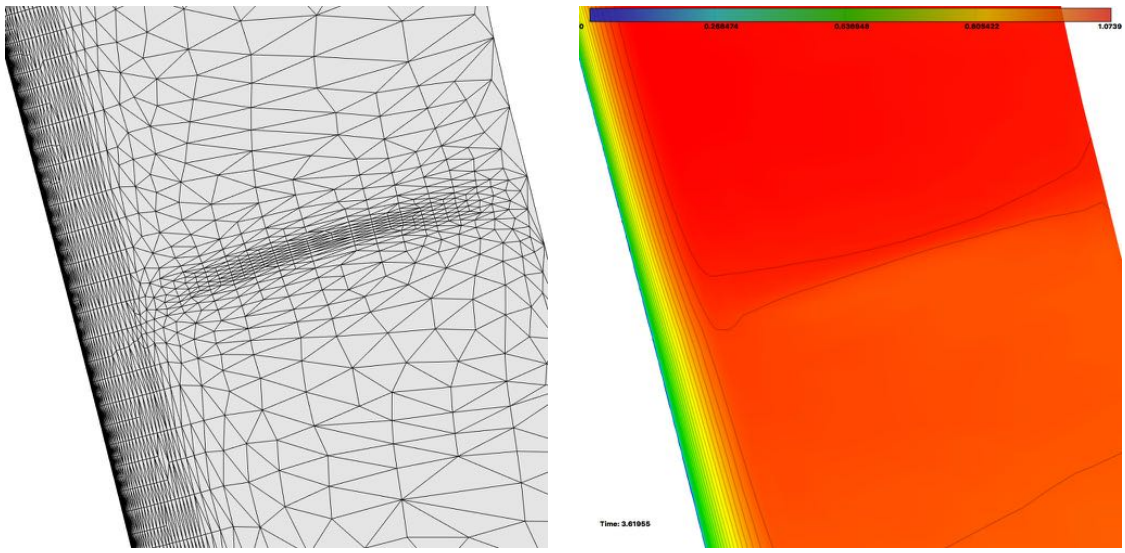


Figure 3.10: LS89 case 2. Zoom on the mesh (left) and Mach contours (right) for second test case one LS89 geometry in the shock region.

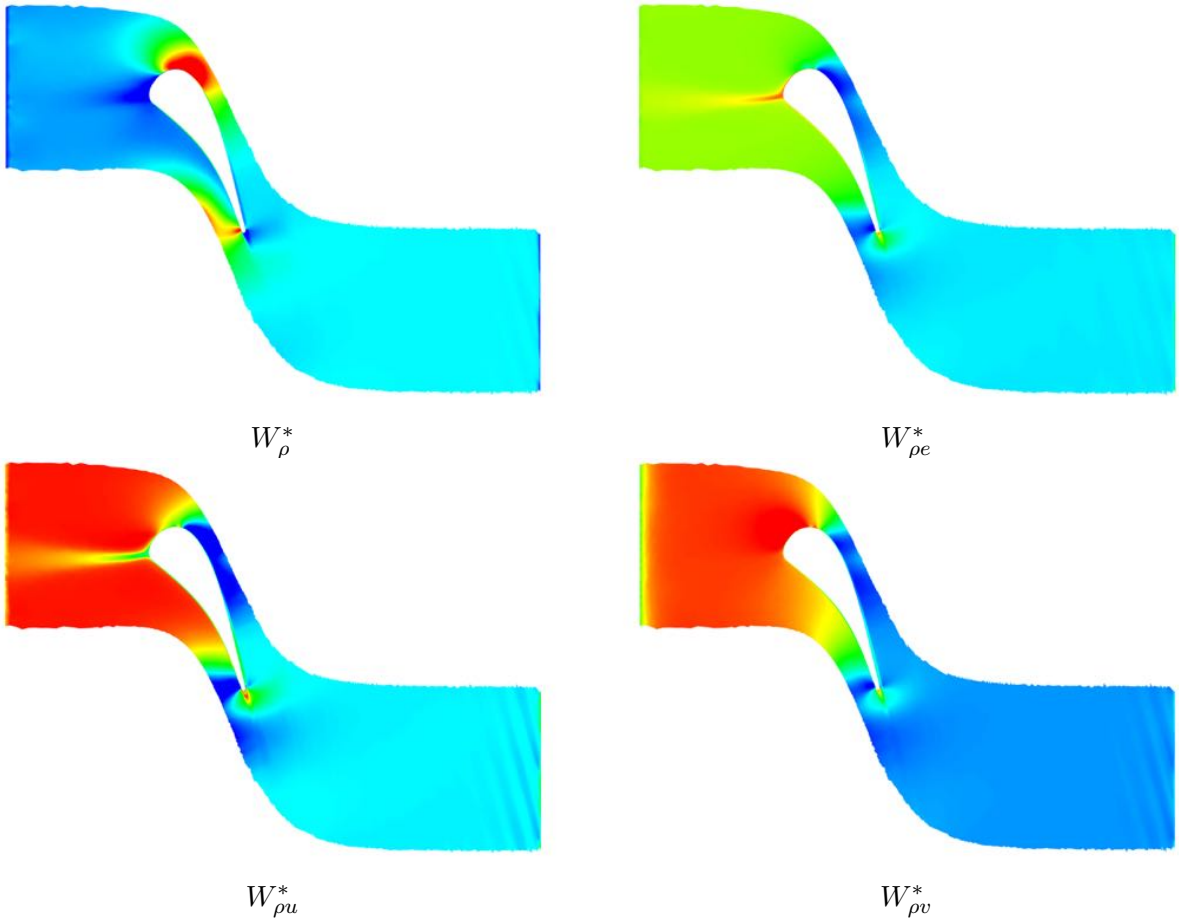


Figure 3.11: LS89 case 2. Periodic adjoint variables for the lift functional.

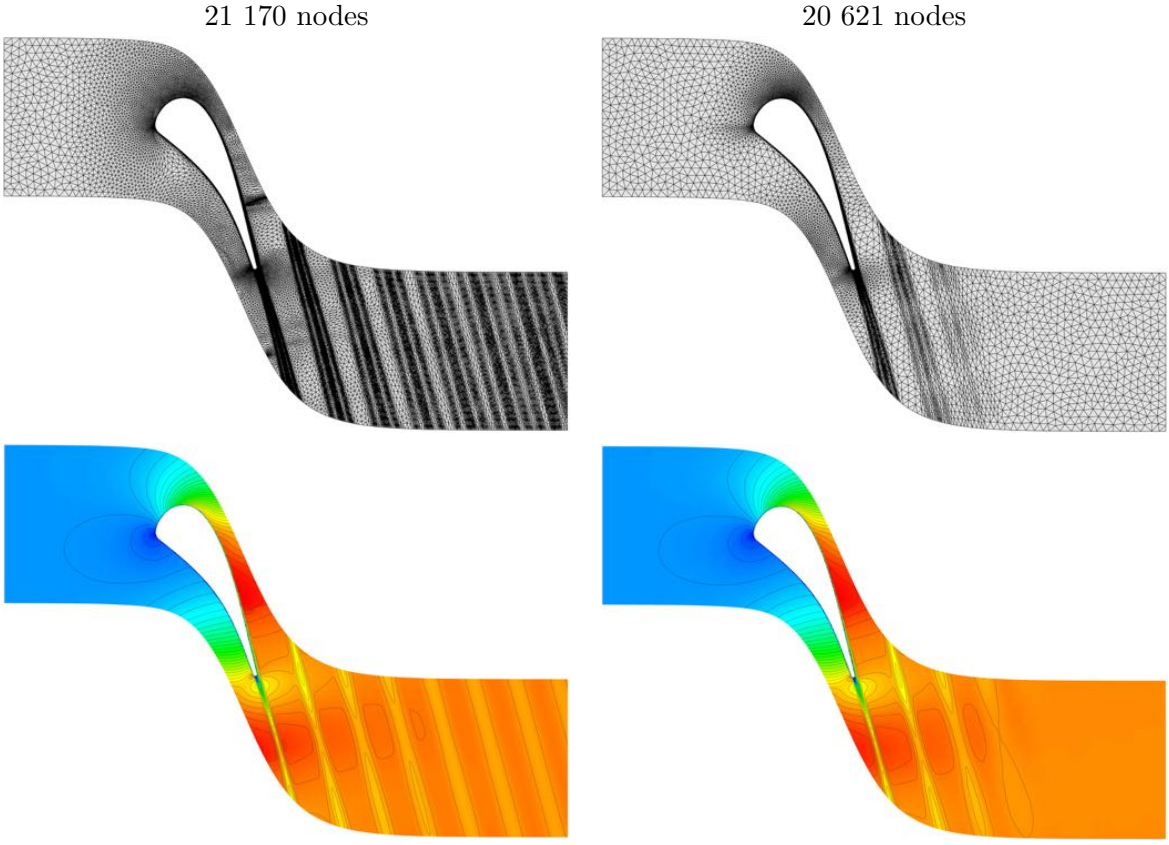


Figure 3.12: LS89 case 2. Adapted meshes (top) and corresponding Mach fields (bottom), obtained with feature-based adaptation using the Mach field as sensor (left) and goal-oriented mesh adaptation based on the friction coefficients (right).

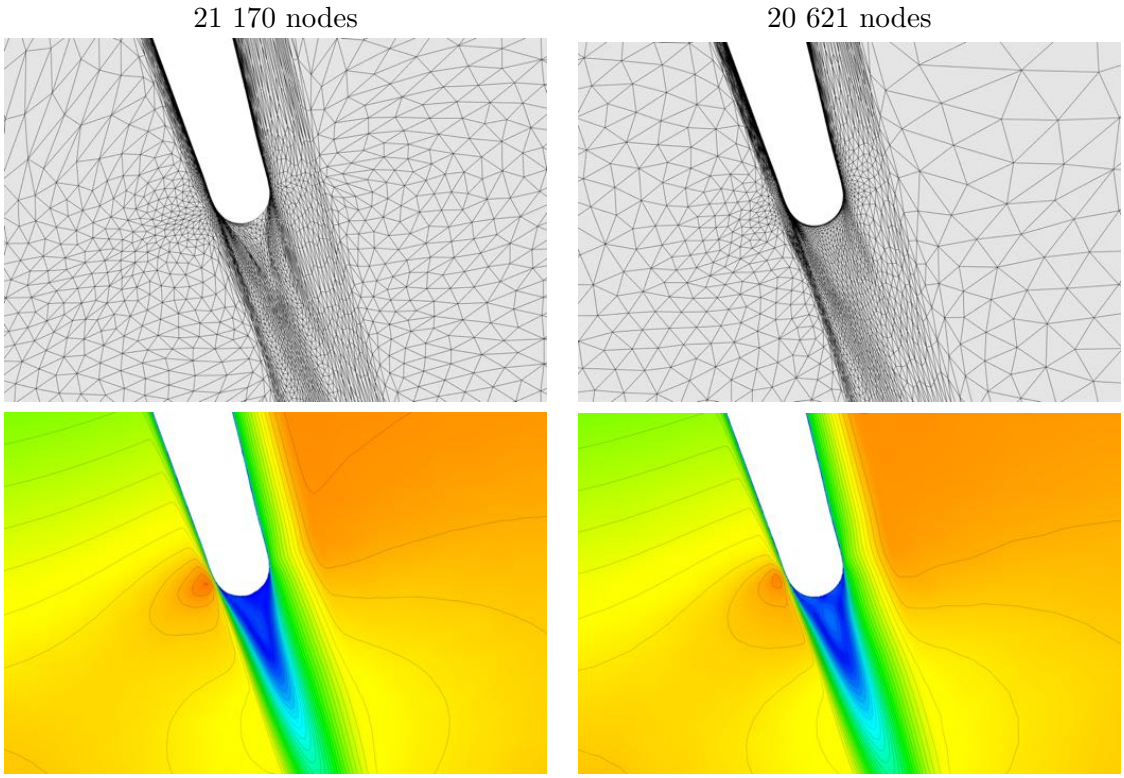


Figure 3.13: LS89 case 2. Zoom on the trailing edge. Adapted meshes (top) and corresponding Mach fields (bottom), generated with feature-based adaptation using the Mach filed as sensor (left) and goal-oriented mesh adaptation based on the friction coefficients (right).

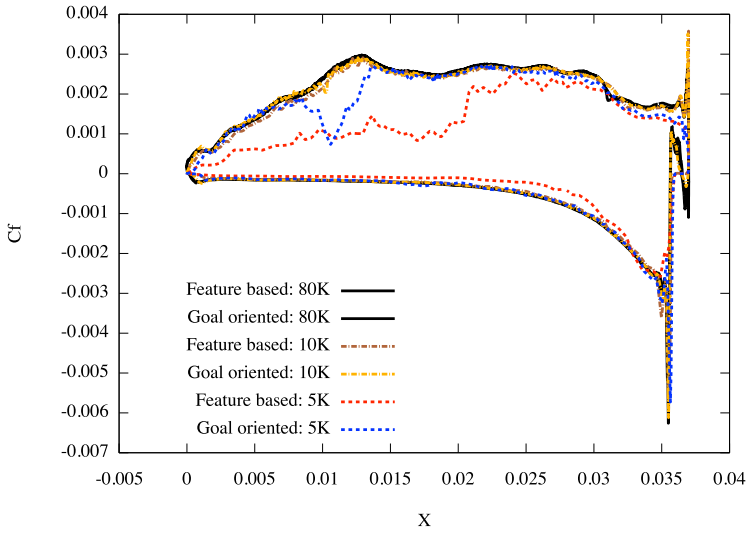


Figure 3.14: LS89 case 2. Friction coefficient on LS89 blade with feature-based adaptation using the Mach field (red) with 5 361vertices, and goal-oriented mesh adaptation (blue) with 4 813 vertices, compared to a reference fine solution (black) with 81 224 vertices.

3.3.2 RO37 case

In this first study, performed with Safran Tech, we have validated the periodic flow solver in 3D and performed a preliminary mesh adaptation study.

Case description

The NASA Rotor 37 geometry is considered here, its regime is described in Table 3.2. The gaz is described with the perfect gaz law with $r = 287.04$, $\gamma = 1.4$, and the viscosity follows the Sutherland law with $\mu_0 = 1.716 \times 10^{-5}$, $T_0 = 273.15$ K and $S = 110.4$. As the geometry is periodic, the simulation is restrained to a single sector of 10 degrees in the rotating frame. The geometry is thus fixed. The computation is initialized with the first-order scheme and a constant solution in the reference frame, *i.e.* rotating in the rotating frame. This choice was led by the fact that inflow/outflow boundary conditions were less stable than solid boundary conditions. The computation is then pursued with the second-order scheme, fourth-order dissipation with the Piperno limiter.

Inflow Total Pressure	$1.013 \times 10^5\ Pa$
Inflow Total Temperature	$300.0\ K$
Outflow Pressure	$1.013 \times 10^5\ Pa$
Rotating Speed	$1\ 800\ rad.s^{-1}$

Table 3.2: RO37 test case state description.

Mesh adaptation

In this study, we performed a mesh adaptation with fixed boundary layer and fixed periodic boundaries. We identify the structured boundary layer in the initial mesh and leave it unchanged in the mesh adaptation process. The local Mach field is used as sensor for the adaptation, and the interpolation error is controlled in L^2 norm.

The initial mesh, hand-made and provided by Safran contains, 2 64 712 vertices, 504 136 triangles and 16 417 280 tetrahedra. The final mesh shown here contains 6 348 990 vertices and 37 498 453 tetrahedra.

Results are shown in Figures 3.15 and 3.16, we can see how the main features of the flow (shocks, wake, shock-boundary layer interaction, ...) are automatically refined which significantly improves the quality of the solution. Details of the interaction between the shock and the boundary layer are shown in Figures 3.21 and 3.22. Such features are impossible to predict and to appropriately discretize without having an accurate solution of the case, while mesh adaptation automatically captures it and increases the accuracy of the mesh in that region.

Alongside to the shocks, smoother details are also well discretized thanks to the L^2 norm adaptation. This is illustrated by the tip vortex generated by the clearance between the blade and the outer wall of the compressor. This vortex, as can be seen in Figure 3.17, is created near the leading edge of the blade and goes away from it, sustained by the shear layer due to the clearance. A closer look on it in Figures 3.18 and 3.20, shows that the vortex is well refined

and accurately captured despite the presence of more stronger phenomena like the shear layer

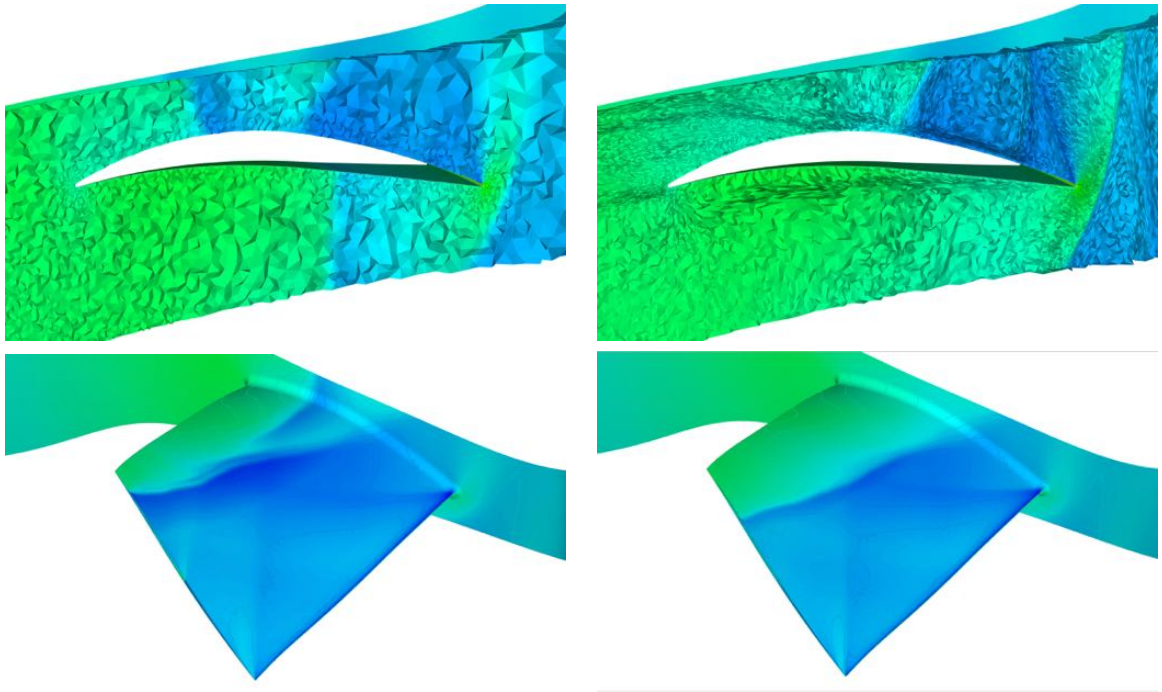


Figure 3.15: RO37 test case. Comparison between the initial (left) and the adapted (right) RO37 meshes. Pressure in a cut through the volume (top) and on the surface (bottom).

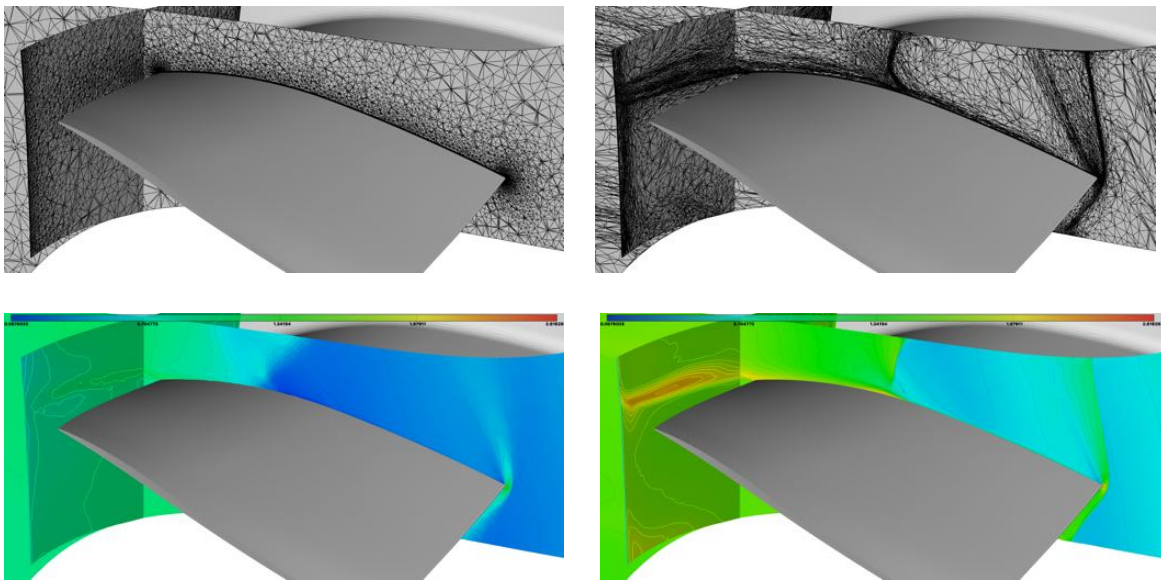


Figure 3.16: RO37 test case. Cuts through the volume over the blade and in the wake. Initial (left) and adapted (right) configuration, meshes (top) and velocity contours (bottom).

and shocks displayed in Figure 3.19.

In Figures 3.22 and 3.17, the structured boundary layer, left unchanged during mesh adaptation process is visible in the vicinity of the blade. In this first early study, we were not able to adapt the boundary layers and we still relied on the traditional approach, *i.e.* generating a structured boundary layer on walls. However, Figures 3.22 and 3.17 illustrate the difficulties of this approach as the structured boundary layer must not impact negatively the solution in the domain. While this structured boundary layer seems to be sufficiently fine in Figure 3.22, which is not even true as we will see, it size is clearly questionable in Figure 3.17 with respect to the vortex and inner mesh size. Figure 3.17 also illustrates the difficulty of generating boundary layers in geometries like the tip gap. We will see in Section 6.1 how to tackle this issue.

Figures 3.22 also shows on the top side of the domain, that elements left unchanged on the periodic frontier are at least one order bigger than the requested size. This leads to a strong diffusion of the shock and the generation of ill conditioned elements. This will in turn lead to unphysical results and the break down of the solver. This issue will be addressed in Section 7.1.

Additionally, in order to illustrate the versatility of mesh adaptation in turbomachinery applications, we compared three different regimes at the following rotating speed: 1625 rad/s, 1800 rad/s and 1975 rad/s. Resulting meshes and solutions are shown in Figure 3.24, exhibiting both relative local Mach number solution field and adapted meshes on a plane at about 50 percent of the spawn. These three cases mimics a performance study at different rotation velocities in order to determine functioning points such as stall or optima. Such studies require numerous computations. However, as can be seen in Figure 3.24, the shock-boundary layer system strongly depends on the considered regime. The shock that forms on the leading edge is initially clearly detached from the blade and hits the second blade (below the first one) through the periodicity aproximatively at mid-chord. This shock is initally relatively weak and induces a small separation of the boundary layer. As the rotation speed increases, the shock angle gets smaller and the shock itself gets closer to the blade. It hits the second blade closer to the trailing edge forming a lambda shock and inducing a larger separation zone. Both lambda shock and

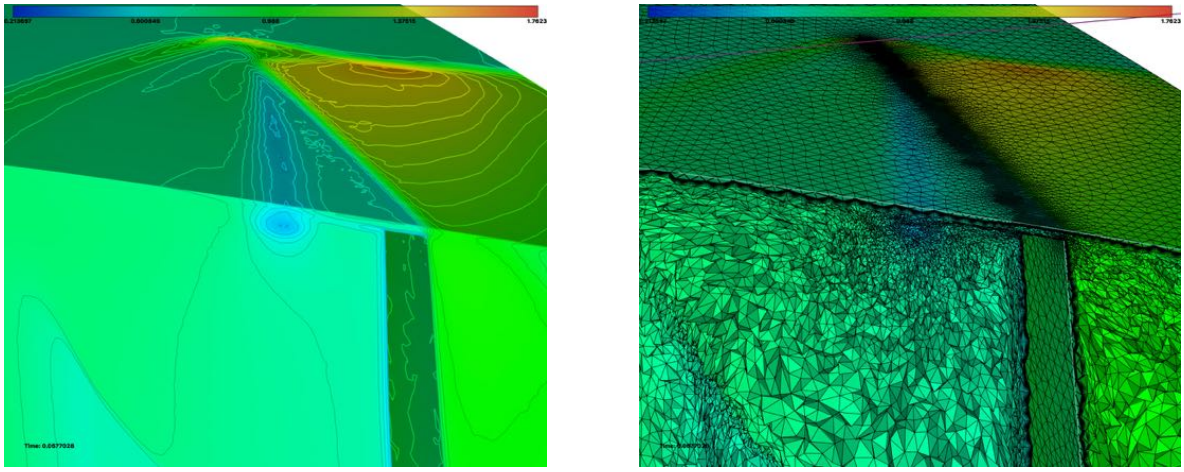


Figure 3.17: RO37 test case: Cut through the volume over the blade displaying the clearance vortex. Density contours (left) and adapted mesh (right).

separation enlarge and move backward as the rotation velocity keeps increasing.

All these physical features require an appropriate discretization to be correctly computed and play an important role in the performance of the compressor[Gou 2018, Denton 1997]. Losses and choke flow are determined by the shock system which induces the boundary layer separation which is responsible of the remaining losses. In the third case, the boundary layer separation produces a blockage which induces a flow reacceleration leading to a secondary passage shock. Meanwhile, the part of the boundary layer that does not reattach produces an important radial transport impacting the apparent efficiency. Finally, the tip gap flow is known to have an important influence on compressor efficiency and requires a proper fine discretization.

It is obvious here that it is impossible to have a "good" mesh for all three configurations

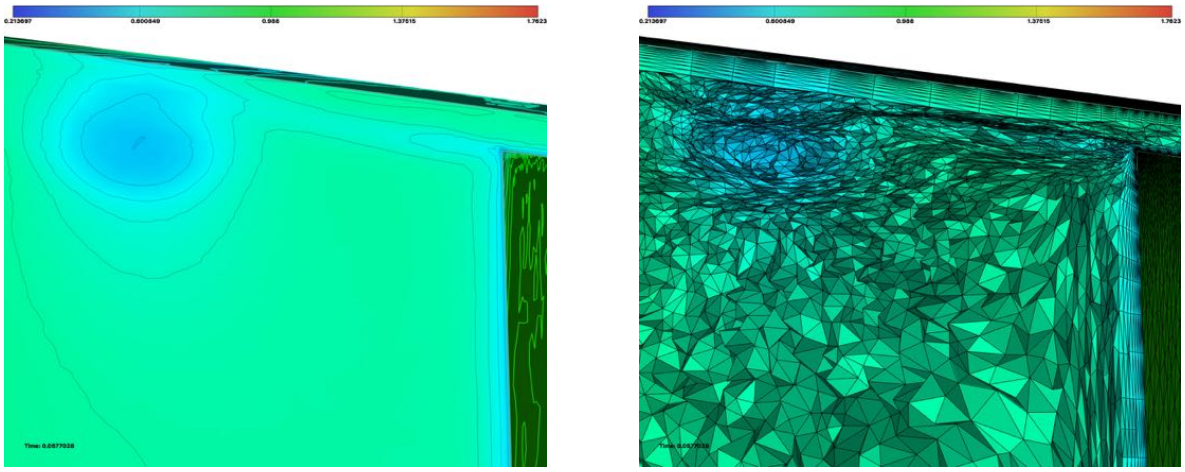


Figure 3.18: RO37 test case: Cut through the volume over the blade, zoom on the clearance and the vortex facing upwind. Density contours (left) and adapted mesh (right).

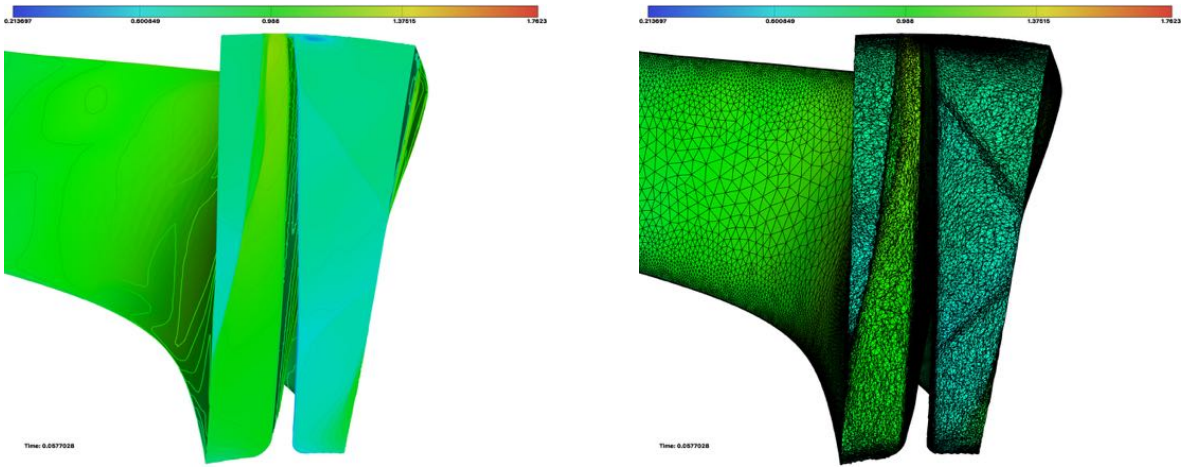


Figure 3.19: RO37 test case: Cut through the volume over the blade, facing downwind. Density contours (left) and adapted mesh (right).

unless it is fine everywhere and thus particularly expensive to use. Figure 3.23 shows for comparison the same cut as in Figure 3.24 for the second case on the initial non-adapted mesh. The lambda shock is replaced by a single shock while a second passage shock should form although it is strongly diffused. The interaction of both shocks with the boundary layer is thus totally different, changing the overall performance. Automatic anisotropic mesh adaptation performs here well, providing an efficient mesh for each computation.

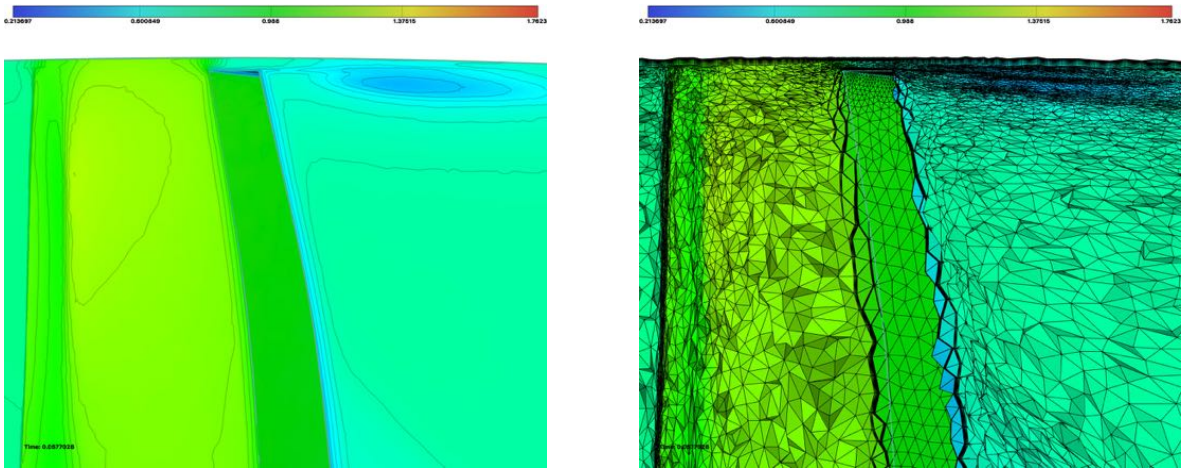


Figure 3.20: RO37 test case: Cut through the volume over the blade, zoom on the clearance and the vortex facing downwind. Density contours (left) and adapted mesh (right).

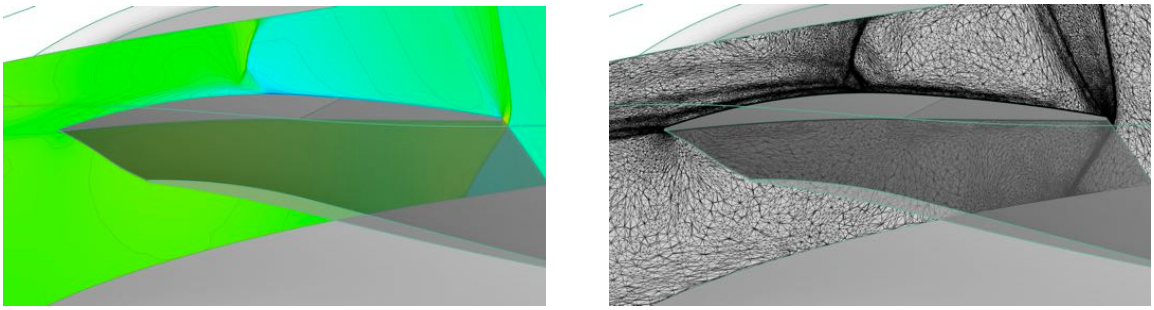


Figure 3.21: RO37 test case: Zoom on the cut through the volume over the blade where the shock boundary layer interaction occurs. Pressure contours (left) and adapted mesh (right).

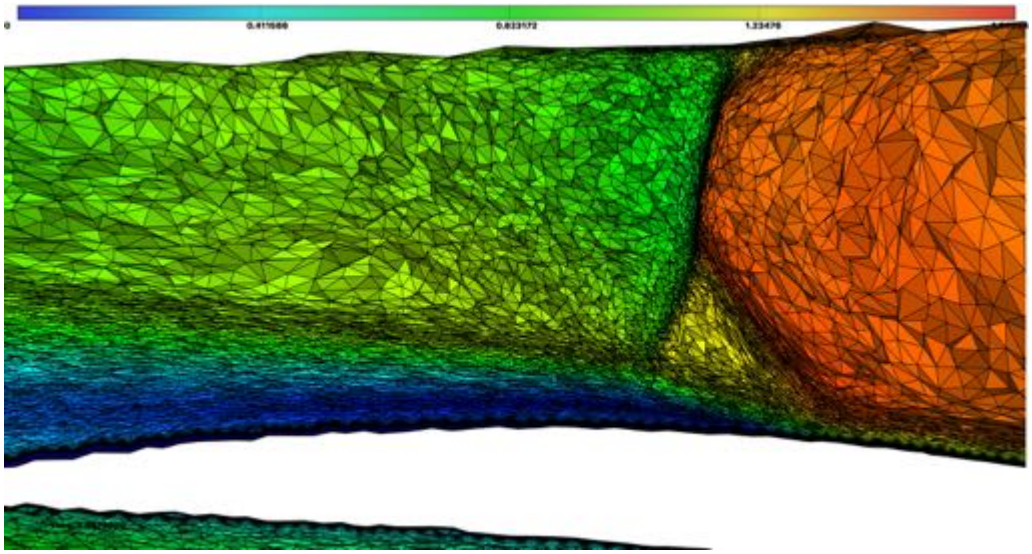


Figure 3.22: RO37 test case: Zoom on the lambda shock due to the shock-boundary layer interaction. Solution represents the local relative Mach number.

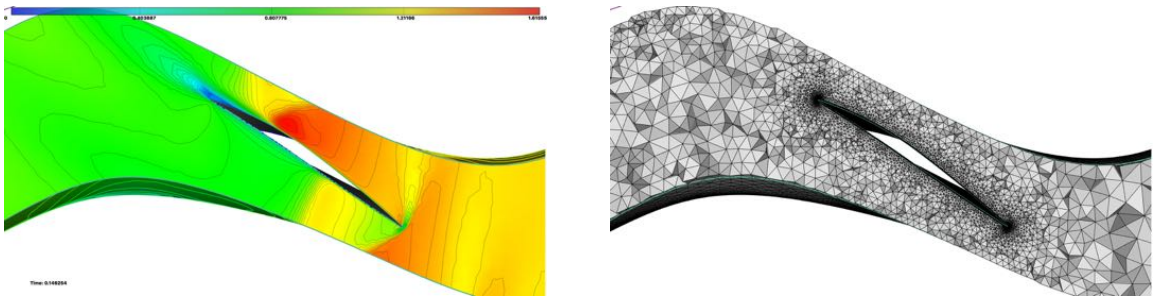


Figure 3.23: RO37 test case: Relative Mach number solution field and initial coarse non-adapted mesh for the second case, at a rotation speed of 1 800 rad/s.

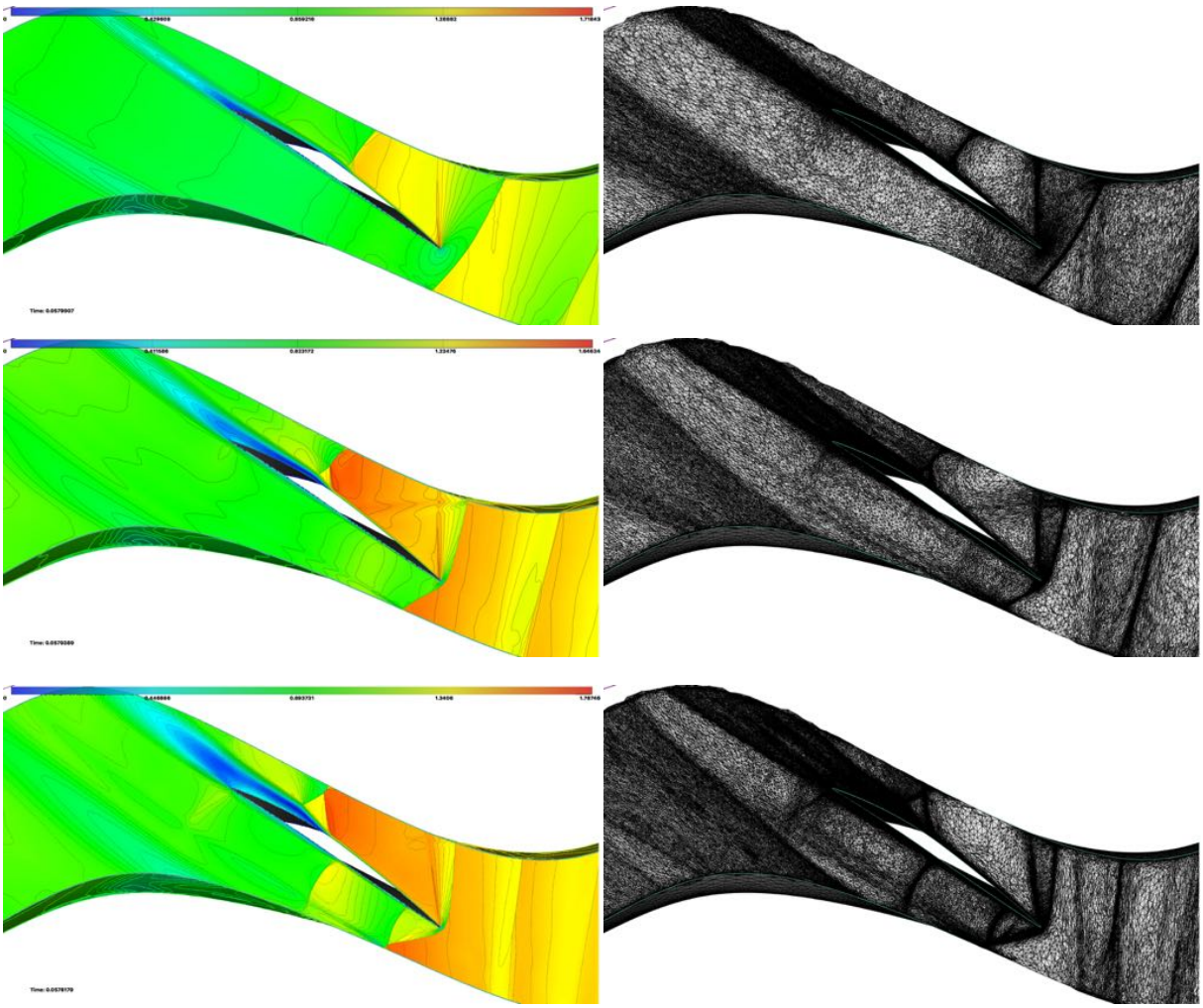


Figure 3.24: RO37 test case: Relative Mach number solution fields and adapted meshes for three rotation velocities: 1 625 rad/s, 1 800 rad/s and 1 975 rad/s.

Chapter 4

Solution correction for error estimation

4.1 Non-linear Corrector computation

The numerical simulation of engineering problems involves a discrete solution which is alleged to converge toward the continuous solution of the problem as the size of the elements of the mesh decreases. As the exact analytical solution of the problem is sought, the difference between the numerical and the analytical solution can be seen as a noise. In an engineering context, this noise can have disastrous consequences on the numerical prediction which can, in turn, lead to actual accidents or misconceptions.

It is thus essential to reduce and estimate this error. This is usually done by leading a grid convergence study, where the error is estimated with the difference between two solutions computed on two successives grids. However, this approach is valid only if the asymptotic regime has already been reached and requires an additional finer grid (thus expensive) to appropriately estimate the error of the current solution. Moreover, the prescription and generation of a sequence of nested grids is time consuming and requires the expertise of an engineer.

Mesh convergence study is mandatory to have a reasonable confidence into the discrete outputs, but it will eventually fail. Problems in mesh convergence are often related with insufficient resolution of local small details of the flow. Mesh adaptation techniques aim at adressing these two difficulties by automatically generating multiple successive grids following a sensor which estimates the error related to the local mesh size [Loseille 2007a, Alauzet 2010a]. It does not require any mesh prescription, but it needs a relatively accurate estimation of the numerical error to be efficient and to stop when the mesh and the solution are converged.

These estimations are based on the fact that discrete solution does not satisfy the continuous problem and the projection of the exact solution on the discrete space does not satisfy the discrete problem either. This can be stated in an *a priori* and an *a posteriori* ways :

$$\Psi_h(\Pi_h W) = \varepsilon_h \neq 0, \tag{4.1}$$

$$\Psi(W_h) = \varepsilon \neq 0, \tag{4.2}$$

where $\Pi_h W$ is the discrete projection of the continuous field W and W_h is the discrete solution. Ψ holds for the differential operator defining the equation we try to solve and Ψ_h is the discretization of Ψ. Given the solution at the vertices of the mesh P_i, we have $(\Pi_h W)(P_i) = W(P_i)$ and $(\Pi_h W)(P) = \sum_i W(P_i)\varphi_i(P)$, with $\varphi_i(P)$ the usual shape functions. The idea behind correction is to use these defect equations in order to estimate an improved solution. Note that

this will improve the quality of the solution in a discrete sense, we want the corrected numerical solution $\widetilde{W_h}$ to be as close as possible to the projection on the mesh of the exact solution $\Pi_h W$.

Linear Correction. This has been addressed by different means in the past with the DWR (Dual Weighted Residual) method[Becker 1996, Fidkowski 2011, Venditti 2000, Venditti 2002b, Jones 2006, Giles 2002, Hartman 2010] method, goal-oriented [Loseille 2010] or ETE (Error Transport Equations) [Derlaga 2017, Hay 2006, Layton 2002, Pierce 2004]. Although the first type of method focuses on the correction of a functional it relies on the same principles. A smoother solution is recovered on a finer mesh, and its non null residual (on the finer mesh) is weighted by the interpolation on the finer mesh of the adjoint computed on the current mesh, to compute a correction of the given functional. Formally, assuming the numerical solution W_h is close to the analytical solution W, the linearization of the functional reads:

$$j_h(\Pi_h W) \approx j_h(W_h) + \frac{\partial j_h}{\partial W} \cdot (\Pi_h W - W_h), \tag{4.3}$$

As the projection $\Pi_h W$ is unknown, it is traditionally formally replaced by $\Pi_h W_{h/2}$ to be estimated. This requires to compute $W_{h/2}$ which is expensive, and if we have $W_{h/2}$, there is no point trying to correct W_h, we can immediately use the better value of $W_{h/2}$. Instead, the difference between W_h and $W_{h/2}$ can be expressed on the finer mesh with the linearization of the numerical problem

$$\Pi_{h/2} W_h - W_{h/2} \approx \left(A_{h/2}\right)^{-1} \cdot \Psi_{h/2}(\Pi_{h/2} W_h), \tag{4.4}$$

where $A_{h/2} = \dfrac{\partial \Psi_{h/2}}{\partial W_{h/2}}$ is the jacobian of $\Psi_{h/2}$. From Relations (4.3) and (4.4), we obtain

$$j_{h/2}(W_{h/2}) \approx j_{h/2}(\Pi_{h/2} W_h) + W^*_{h/2} \cdot \Psi_{h/2}(\Pi_{h/2} W_h),$$

where $W^*_{h/2} = J_{h/2} \cdot A^{-1}_{h/2} = A^{-T}_{h/2} \cdot J_{h/2}$ is the adjoint of the problem.

In practice, in order to avoid the computation and the inversion of the large matrix $A_{h/2}$, the functional j and the numerical system are assumed to have a similar behavior on the finer mesh so that $W^*_{h/2}$ is replaced by the interpolation $\Pi_{h/2} W^*_h$. This is thus formally equivalent to the ETE [Derlaga 2017, Hay 2006, Layton 2002, Pierce 2004] which explicitly computes the corrected solution from the linearization of the problem as

$$\widetilde{W_h} \approx W_h - (A_h)^{-1} \cdot \Psi^*_h(W_h),$$

where Ψ^*_h is a defect residual obtained by different reconstruction means to estimate the error introduced by the discretization. The linearization of the problem is thus a determining factor for the quality of the corrected solution [Gary 2015]. Most importantly, both approaches rely on the linearization of the numerical problem $\dfrac{\partial \Psi_h}{\partial W_h}$ and the precision of the result depends on the calculus of this matrix. In particular, approximate Jacobian should not be used as it will lead to a poor correction. Finally, would the computation of the matrix $\dfrac{\partial \Psi_h}{\partial W_h}$ even be exact, these two approaches are limited by the fact that the residual is linearized.

Non-Linear Correction. In order to overcome these limitations, we propose to compute and add a source term to the problem and to converge it again. We recall that we are interested in the correction of the nodal values of the solution. It is worth noting that if we are able to compute the vector $\Psi_h(\Pi_h W)$, we can use it as a source term in the discrete problem solving:

$$\Psi_h(\widetilde{W}_h) = \Psi_h(\Pi_h W),$$

which solution is obviously the projection of the exact solution. In that sense, this would provide a perfect corrector. However, as we do not have access to W or even $\Pi_h W$, we propose to estimate the corrected residuals based on finer meshes, but without computing the solution on these finer meshes as it would be too expensive.

Note that the resolution of $\Psi_h(\widetilde{W}_h) = \Psi_h(\Pi_h W)$ would require – depending on the non-linearity of the problem – a number of iterations significantly higher than one. Approaching the solution in a single step

$$\widetilde{W}_h \approx W_h + \left(\frac{\partial \Psi_h}{\partial W_h}\right)^{-1} \cdot \Psi_h(\Pi_h W),$$

would lead to a significant loss of precision for nonlinear problems, any accurate the Jacobian can be. Meanwhile, iterating with the same solver has several advantages:

- its implementation is extremely simple as it consists in adding a source term,
- it does not require a more precise linearization than the one of the flow solver,
- it automatically takes into account all numerical features and nonlinearities of the solver (limiters, flux reconstruction, ...).

4.1.1 1D RANS Example

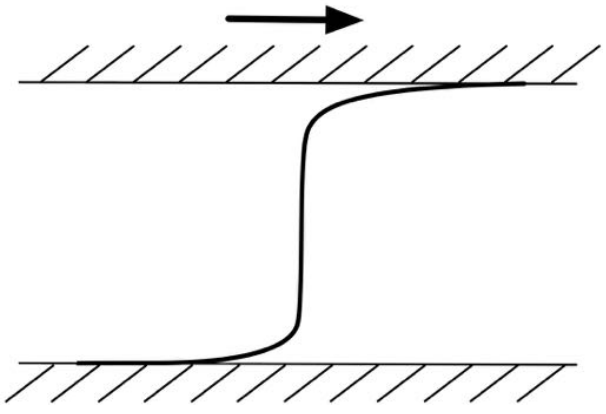


Figure 4.1: Couette flow profile.

We consider here an incompressible RANS Couette flow: the considered domain is delimited on the bottom side by an infinite wall and on the top by an infinite plate sliding along the x axis at a constant velocity (see Figure 4.1). A pressure gradient can eventually be added in

the sliding direction. As the domain is independent with regard to the x-direction, the flow is independent of the x-direction. From the mass conservation

$$\partial_x u + \partial_y v = 0,$$

we deduce that the y-component v of the velocity is constant and null. The RANS equations degenerates to

$$\partial_y \left((\mu + \mu_t)\partial_y u\right) = \partial_x p, \tag{4.5}$$

$$\partial_y \left((\nu + \hat{\nu})\partial_y \hat{\nu}\right) = -c_{b2}\partial_y \hat{\nu}\partial_y \hat{\nu} - \sigma c_{b1}\hat{S}\hat{\nu} + \sigma c_{w1} f_w \left(\frac{\hat{\nu}}{d}\right)^2. \tag{4.6}$$

The pressure gradient $\partial_x p$ is not a variable but a parameter that we will fix to zero for now. Focusing on the mean flow equation, we obtain the following finite-element problem and discretization:

$$\int_0^1 (\mu + \mu_t)\partial_y u \partial_y \phi = 0, \quad \forall \phi, \tag{4.7}$$

$$\int_0^1 (\mu + \mu_{t,h})\partial_y u_h \partial_y \phi_h = 0, \quad \forall \phi_h, \tag{4.8}$$

where we use here linear shape functions ϕ_h.

Discretization of μ_t***:*** We recall that μ_t is computed as

$$\mu_t = \rho f_{v1}\hat{\nu}, \quad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}}, \quad \chi = \frac{\hat{\nu}}{\nu}. \tag{4.9}$$

It can be discretized either nodally as

$$\mu_{t,h} = \sum_i \mu_{t,i}\phi_{h,i},$$

where $\mu_{t,i}$ is computed nodally with Equation (4.9), or element wise as

$$\mu_{t,h} = \rho f_{v1,h}\hat{\nu}_h, \quad f_{v1,h} = \frac{\chi_h^3}{\chi_h^3 + c_{v1}^3}, \quad \chi_h = \frac{\hat{\nu}_h}{\nu}.$$

As integrals are computed with quadratures, the second expression is computed for each quadrature point and is thus formally equivalent to a higher polynomial reconstruction. It may lead to negative values of the turbulent viscosity depending on the quadrature points chosen.

It is interesting to note [Kalitzin 2005] that according to the turbulent boundary layer theory, both the dimensionless eddy viscosity $\nu_t^+ = \frac{\nu_t}{\nu}$ and the dimensionless Spalart-Allmaras variable $\hat{\nu}^+ = \frac{\hat{\nu}}{\nu}$ are supposed to be linear in the log-layer, $\nu_t = \hat{\nu} = \nu\kappa y^+$. In the viscous sublayer, the dimensionless eddy viscosity is not linear $\nu_t^+ = f_{v1}\hat{\nu}^+ = (y^+)^4$ but f_{v1} is defined so that $\hat{\nu}$ is supposed to be linear throughout the boundary layer, so that its discretization with linear shape functions is theoretically exact no matter the boundary mesh size. Figure 4.2 shows the behaviour of f_{v1} and the dimensionless turbulent viscosity ν_t^+ in the near wall region with

respect to the dimensionless distance y^+. We can see that the variations of the turbulent viscosity introduced by f_{v1} is localized in the viscous layer and transition layer (below $y^+ = 20$). If the first off-wall node lies over $y^+ = 20$, the nodal discretization of μ_t will totally ignore the presence of the viscous sublayer even if the discretization of $\hat{\nu}$ is exact. On the contrary, the behaviour of μ_t in the viscous sublayer can be recovered with a higher precision if an element-wise disretization of μ_t is used. However, as we will see, this effect is negligible compared to the discretization of u.

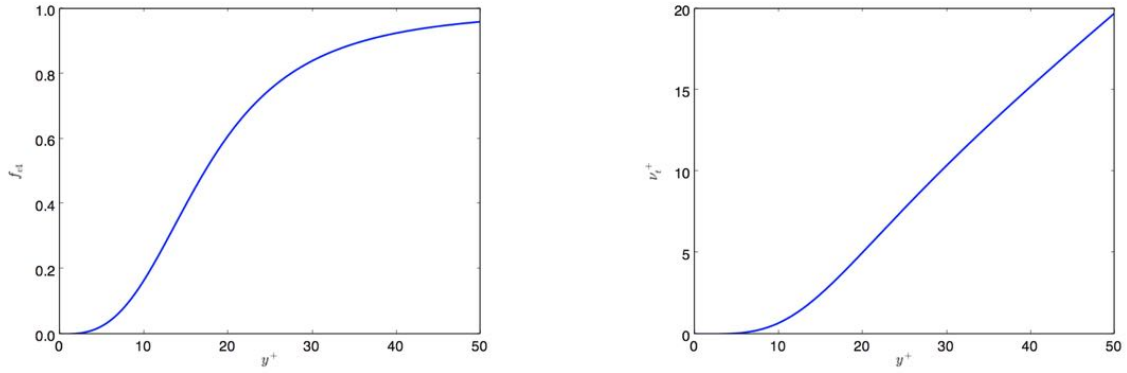


Figure 4.2: f_{v1} (left) and ν_t^+ in the near wall region.

Solution correction: In the particular RANS 1D case, the finite element corrector can be simply stated. Replacing $u = \Pi_h u + (u - \Pi_h u)$ and $\mu_t = \Pi_h \mu_t + (\mu_t - \Pi_h \mu_t)$ in Equation (4.7), we obtain,

$$\int_0^1 \left(\mu + \left(\Pi_h \mu_t + (\mu_t - \Pi_h \mu_t)\right)\right) \partial_y \left(\Pi_h u + (u - \Pi_h u)\right) \partial_y \phi = 0, \quad \forall \phi.$$

Developing this equation yields the equation verified by $\Pi_h u$, for all ϕ

$$\begin{aligned} \int_0^1 (\mu + \Pi_h \mu_t) \partial_y \Pi_h u \partial_y \phi = \int_0^1 & - (\mu_t - \Pi_h \mu_t) \partial_y \Pi_h u \partial_y \phi && \text{Error on } \mu_t \\ & - (\mu + \Pi_h \mu_t) \partial_y (u - \Pi_h u) \partial_y \phi && \text{Error on } u \\ & - (\mu_t - \Pi_h \mu_t) \partial_y (u - \Pi_h u) \partial_y \phi && \text{Non-linear term} \end{aligned}$$

This equation relates $\Pi_h u$ and thus the implicit error $u_h - \Pi_h u$ to the interpolation errors $u - \Pi_h u$ and $\mu_t - \Pi_h \mu_t$. We can see that given our actual discrete solution u_h on the mesh, we only need to estimate $u - \Pi_h u$ and $\mu_t - \Pi_h \mu_t$ to obtain a corrected solution.

We will now compare the different contributions of these errors. To this end, we first compute a reference solution on a fine mesh. This solution will then be used to provide the $u - \Pi_h u$ and $\mu_t - \Pi_h \mu_t$ values to compute the right-hand side source term of the equation. Note here that we decoupled the mean flow equation from the Spalart equation by using directly $\Pi_h \mu_t$ instead of $\mu_{t,h}(\Pi_h \hat{\nu})$. The flow characteristics are summarized in Table 4.1, additionally, the vertices are distributed as

$$y_i = 0.5 \frac{\tanh\left(\lambda(\frac{i}{N} - 0.5)\right)}{\tanh(0.5\lambda)} + 0.5,$$

so that the boundaries are refined, with $N = 200$ and $\lambda = 7.3$ for the fine solution and $N = 40$, $\lambda = 5$ for the coarse solution. This results in first wall spacings of about $y_1^+ = 0.5$ for the fine solution and $y_1^+ = 34$ for the coarse solution.

Reynolds number: $Re = 1 \times 10^6$	Velocity: $U = 1$
Length: $L = 1$	Density: $\rho = 1$
Pressure: $P = 1$	

Table 4.1: Couette Flow characteristics

We first verify that the non-linear term is actually a second order error term and is negligible by computing $\tilde{u}$ the solution of the following equation on the coarse mesh

$$\int_0^1 (\mu + \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi = \int_0^1 -(\mu_t - \Pi_h \mu_t) \partial_y (u - \Pi_h u) \partial_y \phi.$$

We can see in Figure 4.3 (top left) that the corrected solution $\tilde{u}$ (red) is almost identical to the initial solution (black), confirming that the non linear term can be neglected.

Surprisingly, the error on μ_t seems to have very little effect on the solution alone as can be seen in Figure 4.3 (top right). We solved here,

$$\int_0^1 (\mu + \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi + (\mu_t - \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi = 0.$$

This is probably due to the fact that μ_t is almost linear so that its interpolation error is actually very small.

Figure 4.3 (bottom left) shows the large correction produced by the error of u, solving

$$\int_0^1 (\mu + \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi = \int_0^1 -(\mu + \Pi_h \mu_t) \partial_y (u - \Pi_h u) \partial_y \phi.$$

Although the turbulent boundary layer is governed by the turbulence model, this leads us to the counter intuitive conclusion that the largest part of the error on the mean flow actually comes from the discretization of the velocity itself. It is thus consistent to prioritize the discretization of the mean flow for turbulent applications. However, as we will see later, this requires a particular attention.

Finally, Figure 4.3 (bottom right) shows the correction of the solution taking into account both the error on u and μ_t, *i.e.* solving

$$\int_0^1 (\mu + \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi + (\mu_t - \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi = \int_0^1 -(\mu + \Pi_h \mu_t) \partial_y (u - \Pi_h u) \partial_y \phi. \tag{4.10}$$

This corrected solution is very close to the actual projected solution $\Pi_h u$. This confirms that it is consistent to focus on the first order terms of the error for further analysis. It may seem surprising to see that this correction is larger than the sum of each individual correction. Indeed, as we decoupled the mean flow from the turbulent state using $\Pi_h \mu_t$, the equation $\int_0^1 (\mu + \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi$ is now linear, and so is Equation (4.18). However, the correction of μ_t

brought by the term $(\mu_t - \Pi_h\mu_t)\partial_y\tilde{u}\partial_y\phi$ actually introduces a non-linearity which can be stated as follows. Consider u_0 the solution of the equation with no correction, $u_1 = u_0 + \delta u_1$ the corrected solution taking into account the error on u and $u_2 = u_0 + \delta u_2$ the corrected solution taking into account the error on μ_t. We have

$$\int_0^1 (\mu + \Pi_h\mu_t)\partial_y u_0 \partial_y\phi = 0, \tag{4.11}$$

$$\int_0^1 (\mu + \Pi_h\mu_t)\partial_y \delta u_1 \partial_y\phi = \int_0^1 -(\mu + \Pi_h\mu_t)\partial_y(u - \Pi_h u)\partial_y\phi, \tag{4.12}$$

$$\int_0^1 (\mu + \Pi_h\mu_t + (\mu_t - \Pi_h\mu_t))\partial_y \delta u_2 \partial_y\phi = \int_0^1 -(\mu_t - \Pi_h\mu_t)\partial_y u_0 \partial_y\phi. \tag{4.13}$$

Using these relations and injecting $\tilde{u} = u_0 + \delta u_1 + \delta u_2$ in the equation with both corrections leaves us with the following residual

$$\begin{aligned}\int_0^1 \underbrace{(\mu + \Pi_h\mu_t)\partial_y u_0 \partial_y\phi}_{\text{Initial solution:=0}} &+ \underbrace{(\mu + \Pi_h\mu_t)\partial_y \delta u_1 \partial_y\phi + (\mu + \Pi_h\mu_t)\partial_y(u - \Pi_h u)\partial_y\phi}_{\text{Error on u:=0}} \\ &+ \underbrace{(\mu + \Pi_h\mu_t + (\mu_t - \Pi_h\mu_t))\partial_y \delta u_2 \partial_y\phi + (\mu_t - \Pi_h\mu_t)\partial_y u_0}_{\text{Error on } \mu_t\text{:=0}} \\ &+ \underbrace{(\mu_t - \Pi_h\mu_t)\partial_y \delta u_1 \partial\phi}_{\text{Remaining residual}}.\end{aligned}$$

This remaining residual show how the correction on μ_t enhances the correction on u. Although individually the error on u is predominant, the error on μ_t contributes to a larger correction.

This analysis shows the influence of the different error terms in a 1D boundary layer. However, these expressions cannot be used in practice as they involve $\Pi_h\mu_t$ which is unknown and the interpolation of both variables that are unknown too. Let's first assume that we are able to consistently estimate the interpolation errors and deal with $\mu_{t,h}$ instead of $\Pi_h\mu_t$. We now solve the standard discrete equation

$$\int_0^1 (\mu + \mu_{t,h})\partial_y u_0 \partial_y\phi = 0, \tag{4.14}$$

instead of Equation (4.11) to solve our initial solution. Figure 4.4-(top left) shows the difference between the solutions obtained with Equations (4.14) and (4.11), the second being considered as a correction of the first one. It is clearly negligible.

We now perform the same comparison as previously replacing $\Pi_h\mu_t$ by $\mu_{t,h}$ and keeping the interpolation errors. Figure 4.4-(top right) show the correction taking into account the error on μ_t and Figure 4.4-(bottom left) the correction taking into account the error on u. As previously, the error on μ_t produces a smaller correction than the error on u and in turn employing both produces an even larger correction, see Figure 4.4-(botto right).

It is worth noting on Figures 4.3-(top right) and 4.4-(top right) that, here again, although changing $\Pi_h\mu$ to $\mu_{t,h}$ in Equation (4.14) has a little effect on the solution, it enhances the correction/error due to interpolation errors. Still, the implicit error due to the interpolation

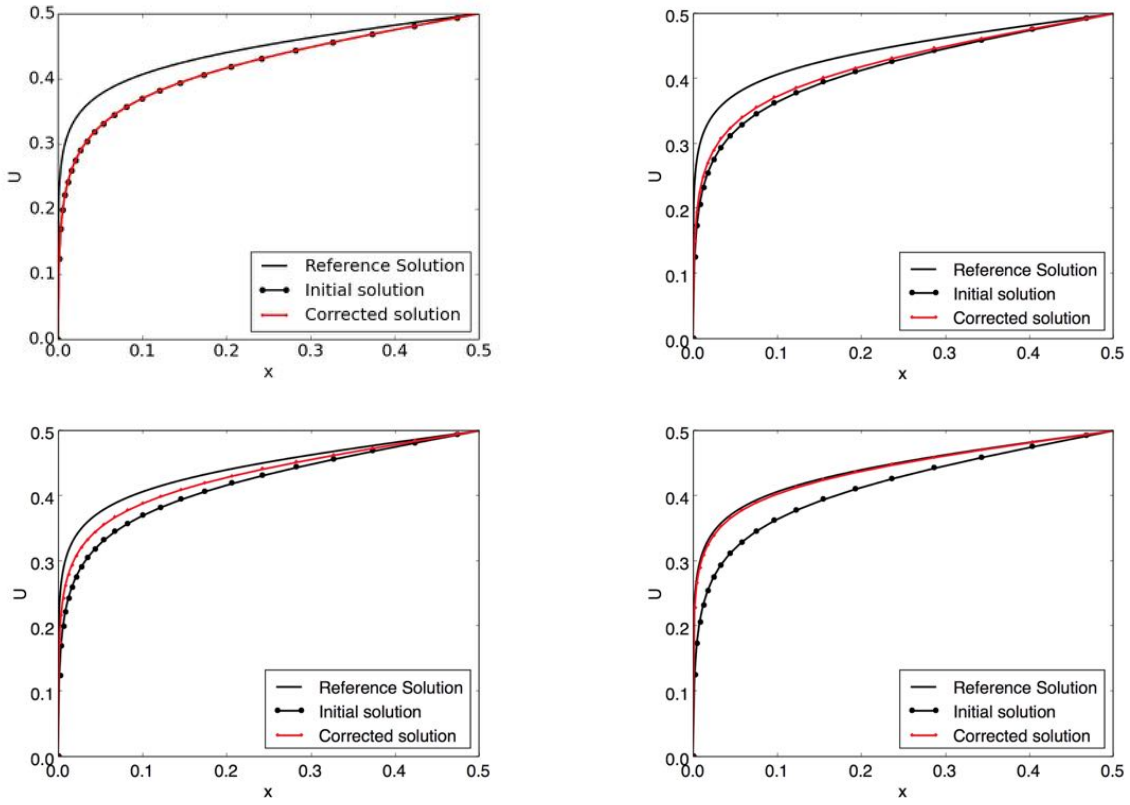


Figure 4.3: Couette flow: corrected velocity profiles with exact error residual and exact interpolation error using: second order error terms (top left), error on μ_t (top right), error on u (bottom left), error on u and μ_t (bottom right).

error of μ_t and u is large and although the value of $\Pi_h \mu_t$ cannot be easily computed, the discretization errors of μ_t and u can be estimated by traditional means:

$$\int_0^1 (\mu + \mu_{t,h}) \partial_y \tilde{u} \partial_y \phi + (\mu_t - \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi = 0, \tag{4.15}$$

$$\int_0^1 (\mu + \mu_{t,h}) \partial_y \tilde{u} \partial_y \phi = \int_0^1 -(\mu + \mu_{t,h}) \partial_y (u - \Pi_h u) \partial_y \phi, \tag{4.16}$$

$$\int_0^1 (\mu + \mu_{t,h}) \partial_y \tilde{u} \partial_y \phi + (\mu_t - \Pi_h \mu_t) \partial_y \tilde{u} \partial_y \phi = \int_0^1 -(\mu + \mu_{t,h}) \partial_y (u - \Pi_h u) \partial_y \phi. \tag{4.17}$$

Estimation of $u - \Pi_h u$***:*** In order to estimate the interpolation errors we rely on a standard quadratic reconstruction. The solution u is assumed to be quadratic over each elements

$$u = \underbrace{\frac{u(y_i)(y_{i+1} - y) + u(y_{i+1})(y - y_i)}{y_{i+1} - y_i}}_{\text{Linear interpolation}} + \underbrace{\frac{H(u)}{2}(y - y_i)(y - y_{i+1})}_{\text{Quadratic part}}$$

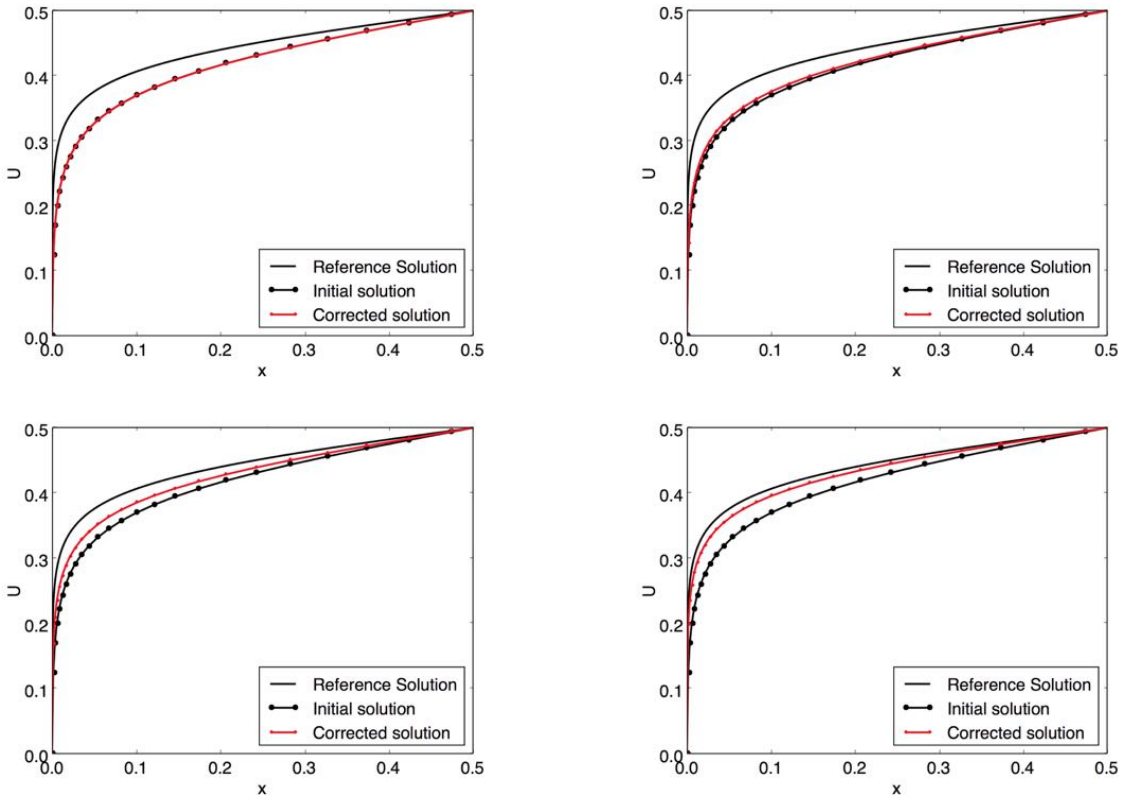


Figure 4.4: Couette flow: corrected velocity profiles with exact error residual, approximate $\mu_{t,h}$ and exact interpolation error using: second order error terms (top left), error on μ_t (top right), error on u (bottom left), error on u and μ_t (bottom right).

where $H(u)$ is the hessian of u. As we are working with a linear representation of the solution, the interpolation error $u - \Pi_h u$ can be estimated with the quadratic term $\frac{H(u)}{2}(y - y_i)(y - y_{i+1})$. This hessian has to be reconstructed from the discrete values of the field u_h. As this is a 1D case, it can be directly computed with standard finite differences:

$$H(u) = 2\frac{(y_{i+1} - y_i)u_{i-1} - (y_{i+1} - y_{i-1})u_i + (y_i - y_{i-1})u_{i+1}}{(y_{i+1} - y_i)(y_{i+1} - y_{i-1})(y_i - y_{i-1})}.$$

For the boundary nodes, as u_{i-1} or u_{i+1} are not defined, we copy the value of the neighbour node in the domain (which is equivalent to an upstream/downstream finite difference evaluation). From this hessian, we can reconstruct on each element the error terms as

$$(u - \Pi_h u)|_{[y_i, y_{i+1}]} = \frac{H(u)}{2}(y - y_i)(y - y_{i+1}).$$

Additionally, we can note that the error on u induces a constant term $-(\mu + \mu_{t,h})\partial_y(u - \Pi_h u)$ that does not depend on $\tilde{u}$. This term can thus be treated as a source term in the resolution and does not require any further modification of the initial solver. We thus propose to extend this approach to all terms so that the correction only involves the computation of a constant

source term. To do so, we remove the dependency to $\tilde{u}$ in the error on μ_t by replacing it by u_h. Finally, we replace the constant discrete value of the turbulent viscosity $\mu_{t,h}$ by the discrete corrected value $\widetilde{\mu_{t,h}}$, so that the equation solved is exactly the same as the primal solution with an additional constant source term:

$$\int_0^1 (\mu + \widetilde{\mu_{t,h}})\partial_y \tilde{u} \partial_y \phi = -\int_0^1 \left[(\mu + \mu_{t,h})\partial_y(u - \Pi_h u) + (\mu_t - \Pi_h \mu_t)\partial_y u_h\right] \partial_y \phi. \tag{4.18}$$

Figure 4.5 shows the resulting correction compared to the previous correction obtained with the exact error. As it could be expected the correction with the approximate hessian is less accurate, but surprisingly it is tremendously poorer!

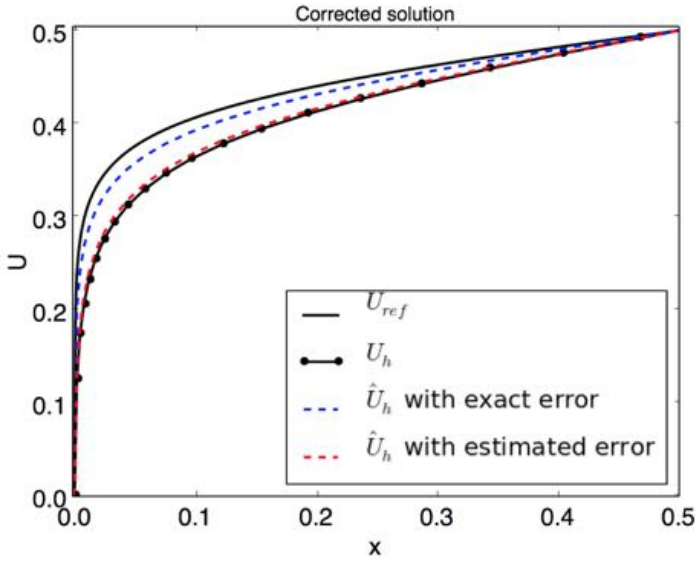


Figure 4.5: Couette flow: corrected solutions with constant source term, using exact interpolation error (dashed blue) and reconstructed hessian estimation of interpolation error (dashed red).

Correction of the numerical Hessian: We have introduced three differences with the exact correction:

- Replace $\mu_{t,h}$ by $\widetilde{\mu_{t,h}}$ in the residual,
- Fix $\tilde{u}$ to u_h in the error on μ_t,
- Replace $u - \Pi_h u$ and $\mu_t - \Pi_h \mu_t$ by their hessian based estimations.

The first modification is very unlikely to be responsible for this large deviation as replacing $\Pi_h \mu_t$ by $\mu_{t,h}$ had very little effect. Further more, as $\widetilde{\mu_{t,h}}$ would be expected to be corrected and thus more accurate that $\mu_{t,h}$ we expect this modification to actually improve the correction. Similarly, we do not expect the second modification to yield a large deviation as it only modifies

the nodal values of the left hand side of $(\mu_t - \Pi_h\mu_t)\partial_y\tilde{u}$. This is why focus on the third modification.

A carefull investigation of the exact source term $(\mu + \mu_{t,h})\partial_y(u - \Pi_h u)\partial_y\phi$ shows that it is almost null every where except in the first elements of the boundary layer. Figure 4.6-(top) shows in black the values of $(u - \Pi_h u)$, we can clearly see the large spike in the first element dominating the overall interpolation error field. The discretization error in the first element is more that ten times larger that in the second element. We recover and quantify a well known fact, when dealing with a wall-resolved turbulence model, the numerical error is concentrated in the first layer of element unless it is fine enough (here $y^+ = 34 >> 1$). However, we can see here an interesting fact, the hessian reconstruction of the error vastly under-estimates the error in the first element.

The red curve in Figure 4.6-(top) is the reconstruction of the interpolation error based on the hessian computed from the discrete values of the fine solution. We can see in the zoom in Figure 4.6-(bottom right) that this process reasonably recovers the hessian and the interpolation error in the volume. But in Figure 4.6-(bottom left) it is clear that it fails in the two first layers, underestimating the error in the first element and overestimating it in the second. This is due to the insufficient discretization with regard of the steep variations of the solution and the slower convergence of the hessian on the boundary.

This effect is worsened by the use of the discrete numerical solution u_h instead of the exact projection $\Pi_h u$. As can be seen in Figure 4.6-(top), the variations of u_h are smoother than those of $\Pi_h u$, which leads to a further under estimation of the error depicted by the blue curve in Figure 4.6-(top). Here again, this approach provides a reasonable estimation of the error in the volume as shown by Figure 4.6-(bottom right) but fails at the boundary, see Figure 4.6-(bottom left).

In order to evaluate the real influence of this miscomputation, we recompute the corrected solution obtained with Equation (4.17) except that we now cancel $u - \Pi_h u$ in the first element. The result is shown in Figure 4.7, the corrected solution obtained in red is compared to the corrected solution obtained with exactly the same terms but taking into account the interpolation error in the first element. This element is clearly responsible for the vast majority of the error in the solution, and the solution obtained is consistent with the one obtained with the numerical estimation based on the hessian in Figure 4.5.

The standard way of dealing with this error is to get rid of it by imposing a fine proper discretization with mesh boundary layers. However, for the present correction and in an error estimation context it is critical to be able to consistently take into account this error. This is why we propose here a correction of the boundary hessian based on the boundary layer theory. As with wall functions, we can add in the process the knowledge of the actual profile of u and precompute the interpolation error $\int_0^h (u - \Pi_h u)$ in the first element. To do so, we use the

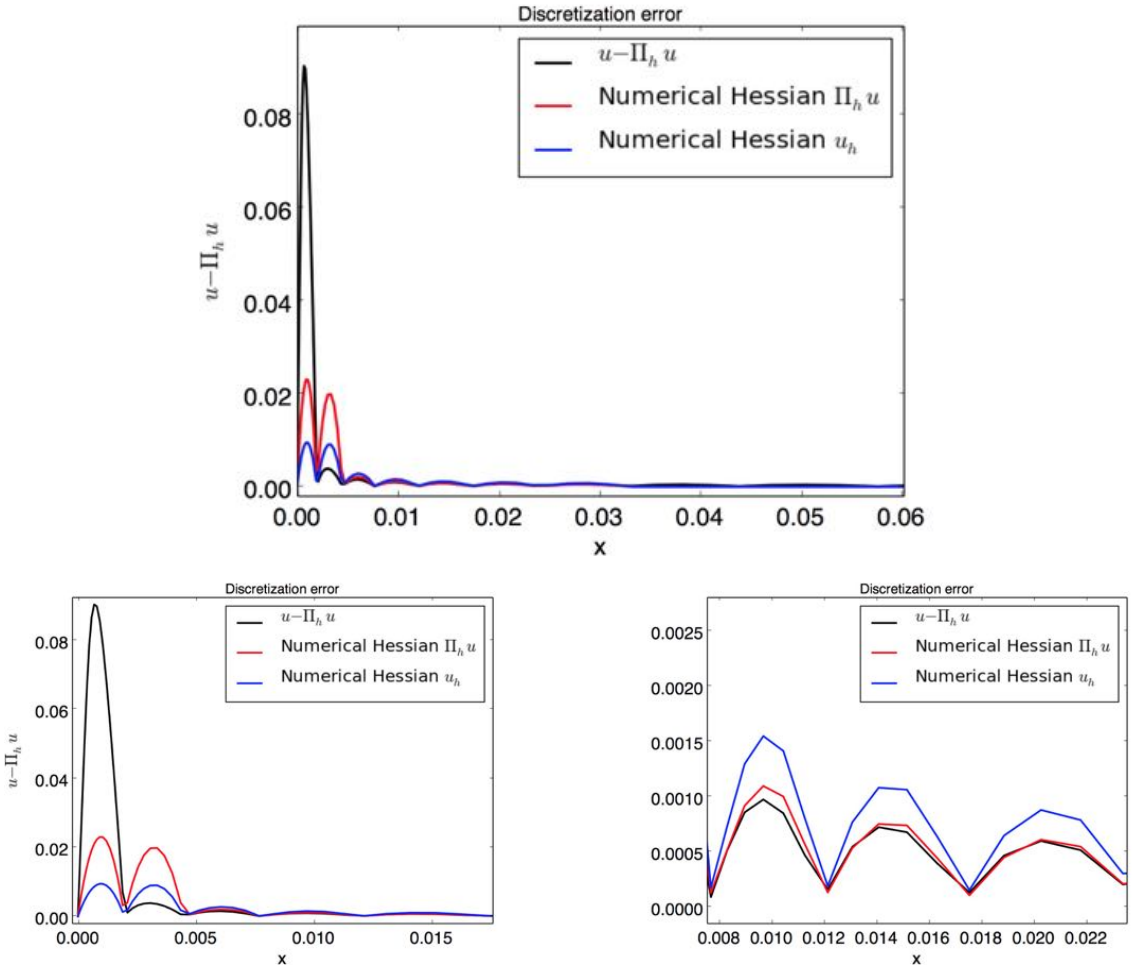


Figure 4.6: Turbulent Couette flow: Exact $u - \Pi_h u$ field in the boundary layer (black) and its estimation with hessian reconstruction from the discrete projection $\Pi_h u$ (red) and the discrete numerical solution u_h (blue) (top). Zoom on the first elements (bottom left) and inside the domain (bottom right).

dimensionless Spalart-Allmaras turbulence profile:

$$\begin{aligned} u^+ &= 5.0333908790505579 \\ &+ 2.5496773539754747 \log((y^+ + 8.148221580024245)^2 + 7.4600876082527945^2) \\ &+ 1.3301651588535228 \log((y^+ - 6.9287093849022945)^2 + 7.468145790401841^2) \\ &- 3.599459109332379 \operatorname{Arctan}(y^+ + 8.148221580024245, 7.4600876082527945) \\ &- 3.6397531868684494 \operatorname{Arctan}(y^+ - 6.9287093849022945, 7.468145790401841), \end{aligned}$$

and compute a tabulation of $\int_0^{h^+} u^+ - \Pi_h u^+$ for h^+ values varying from 0.1 to 200.

We recall that in the error estimation context we approach the real field u by a quadratic reconstruction. As u is proportional to u^+ the interpolation error of u will also be proportional

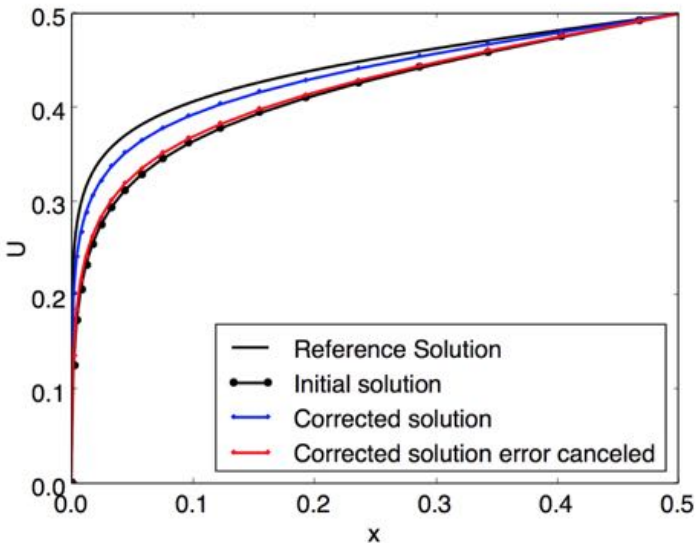


Figure 4.7: Turbulent Couette flow: Influence of the first element. Comparison of corrected velocity profiles with constant source term using exact interpolation error. Reference case (blue) and test case with the interpolation error canceled in the first element (red).

to u^+:

$$\int u - \Pi_h u = u_\tau \int u^+ - \Pi_h u^+,$$

where the interpolation error on u^+ is approached as

$$u^+ - \Pi_h u^+ \approx \frac{1}{2} H(u^+)(y)(h^+ - y).$$

This gives us an error model on each element that states

$$\int_0^{h^+} u^+ - \Pi_h u^+ \mathrm{d}y = \int_0^{h^+} \frac{1}{2} H(u^+)(y)(h^+ - y)\mathrm{d}y = \frac{1}{2} H(u^+) \frac{(h^+)^3}{6}.$$

In our particular case we know that $H(u^+)$ is poorly computed in the first element, but we also know the exact value of the interpolation error $\int_0^{h^+} u^+ - \Pi_h u^+$ in this element thanks to the aforementioned analytical expression. The idea is thus to replace the value of the hessian $H(u^+)$ in the first element so that the error model matches the analytical error that we computed

$$\left(\int_0^{h^+} u^+ - \Pi_h u^+ \right)^{\text{analytical}} = \frac{1}{2} \big(H_{eq}(u^+)\big)(h^+) \frac{(h^+)^3}{6}.$$

Hence by definition

$$\big(H_{eq}(u^+)\big)(h^+) = 12 \frac{(\int_0^{h^+} u^+ - \Pi_h u^+)^{\text{analytical}}}{(h^+)^3}$$

Figure 4.8 shows the ratio of the interpolation error and y^3, and its analytical fitting

$$
\begin{aligned}
6\frac{\int_0^{h^+} u^+ - \Pi_h u^+ \mathrm{d}y}{(h^+)^3} &\approx (a + b(h^+) + c(h^+)^2 + d(h^+)^3)\exp(-(h^+)/50) \\
&+ (e + f(h^+) + g(h^+)^2 + h(h^+)^3 + i(h^+)^4)\exp(-(h^+)/j)
\end{aligned}
$$

with

$$
\begin{array}{llll}
a = 5.70820064 \times 10^{-2} & b = -9.89281611 \times 10^{-4} \\
c = 8.49161598 \times 10^{-6} & d = -1.96360130 \times 10^{-8} \\
e = -5.74864536 \times 10^{-2} & f = -2.80888016 \times 10^{-2} \\
g = -1.20054969 \times 10^{-2} & h = 3.43013035 \times 10^{-4} \\
i = -5.15563915 \times 10^{-4} & j = 1.73570795.
\end{array}
$$

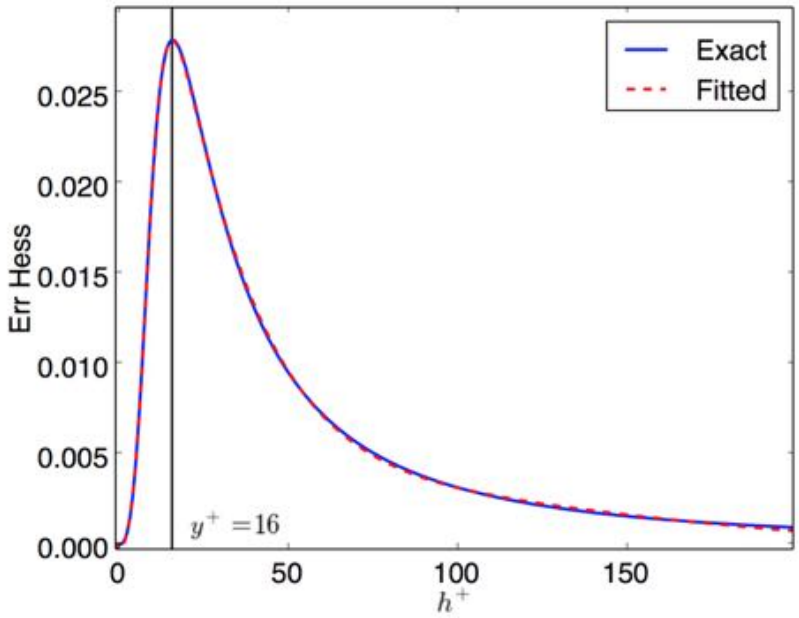


Figure 4.8: Fitting of the dimensionless turbulent equivalent hessian of u^+.

This ratio is (half of) the second derivative of the equivalent quadratic function that would yield the same interpolation error as u^+ on the first element $[0, h^+]$. We will call it the equivalent hessian. It is worth noting here that as u^+ is linear in the viscous layer, the interpolation error remains null in the viscous layer, so that $H_{eq}(u^+)$ is null near 0. Similarly, as u^+ has a logarithmic behaviour for $y^+ > 30$, its interpolation error is of the form $h\log(h)$ for greater values so that $H_{eq}(u^+)$ tends to 0. Note that the choice of exponentials in the fitting is motivated by numerical reasons and does not provides a correct asymptotic equivalent of $H_{eq}(u^+)$ for larger values. Nevertheless, it would be hazardous to use such developments with an element larger than $h^+ = 200$. Finally, we can note that the transition zone is responsible for a strong increment of the interpolation error.

Once this dimensionless profile is computed we can deduce the equivalent hessian of any solution through u_τ and ν like with wall functions:

$$u = u_\tau u^+\left(\frac{u_\tau y}{\nu}\right),$$

so that

$$H_{eq}(u)\big|_y = \frac{u_\tau^3}{\nu^2} H_{eq}(u^+)\big|_{\frac{u_\tau y}{\nu}}. \tag{4.19}$$

Validation: We now apply this process to the previous correction process stated as:

Algorithm 4 RANS corrector with boundary correction

1. Compute the primal solution u_h
2. Compute the approximate hessian $H(u_h)$
3. **Compute** $u_\tau = \sqrt{\mu \partial_y\big|_{y=0} u_h}$
4. **Compute** $H(u)\big|_{y=y_1} \approx \frac{u_\tau^3}{\nu^2} H_{eq}(u^+)\big|_{y^+=\frac{u_\tau x_1}{\nu}}$
5. **Replace the boundary values of the hessian by the corrected value**
6. Compute the source term
7. Compute the corrected solution with the source term.

In order to verify the validity of the dimensionless analysis, we also compute a solution on the fine grid. From this solution we deduce u_τ and compute the estimated interpolation error from the tabulated $H_{eq}(u^+)$. We compare in Figure 4.9 the estimated interpolation error to the actual computation of the interpolation error. The dimensioned distance to the wall corresponding to $y^+ = 1$ is here equal to $y = 5.72668080221 \times 10^5$. We can thus see that we have a reasonable estimation of the interpolation error up to $y^+ \approx 150$.

We then compare in Figure 4.10 the estimation of $u - \Pi_h u$ profiles in the first element to the exact one in solid black line. Our current quadratic reconstruction (blue dashed line) provides a good estimation of $u - \Pi_h u$. Both does not overlap as the second is not a quadratic fonction but their integral is the same. This provides a much better estimation of the interpolation error than the simple hessian reconstruction computed on the coarse mesh (red dash-doted line). But this estimation is based on the value of $u_\tau = 0.0174$ computed on the fine grid. In practice, we only know the value of the current $u_\tau = 0.0296$ computed on the coarse mesh. This larger value of u_τ tends to increase the computed value of $H(u)$ as it is proportional to the cube of u_τ but also increase the value of y^+ from 34.3 to 58.2 which slightly diminishes $H(u^+)$ here. It results in an overestimation of the interpolation error represented by the red dashed line. However in practice, the hessian on one element is taken to be the mean value of the hessian computed on each point of the element. As we only correct the hessian of the boundary node,

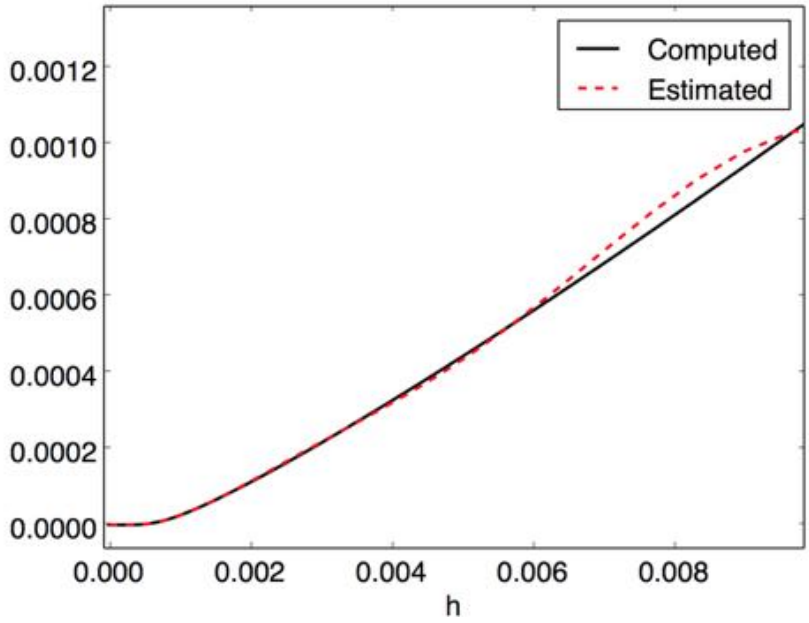


Figure 4.9: Turbulent Couette flow: Validation of the interpolation error estimation in the first element using fitted equivalent hessian.

this overestimation is (by luck) mitigated by the underestimation of the numerical hessian, so that the estimation of $u - \Pi_h u$ is close to the exact one, as shown by the red line with circles.

Used in the solution correction process, this in turn provide a consistent correction as shown in Figure 4.11.

4.1.2 Formal discretization

Finite Element discretization

In the stanrdard Galerkin (eventually stabilized) approach, the strong form of the equations over a given domain Ω:

$$\Psi(W) = 0$$

is reshaped in a weak form

$$\int_\Omega \Psi(W)\varphi \mathrm{d}\Omega = 0, \quad \forall \varphi \in V(\Omega),$$

with additional boundary conditions and where $V(\Omega) = H_0^1(\Omega) = \left\{\varphi \in L_0^2(\Omega) \mid \nabla\varphi \in L^2(\Omega)\right\}$ is the functional space in which the solution is sought and $\Psi(W)$ is a differential operator depending on the problem. The problem is discretized by restricting the functional space to a subset $V_h(\Omega)$ of $V(\Omega)$. In the standard Finite Element approach, the domain Ω is split in a finite number of discrete elements K_i (triangles, tetrahedra, hexahedra, prisms...) on which the restriction $\varphi_{h|K_i}$ is polynomial for each function $\varphi_h \in V_h(\Omega)$ and so that $\cup_i K_i = \Omega$. The value

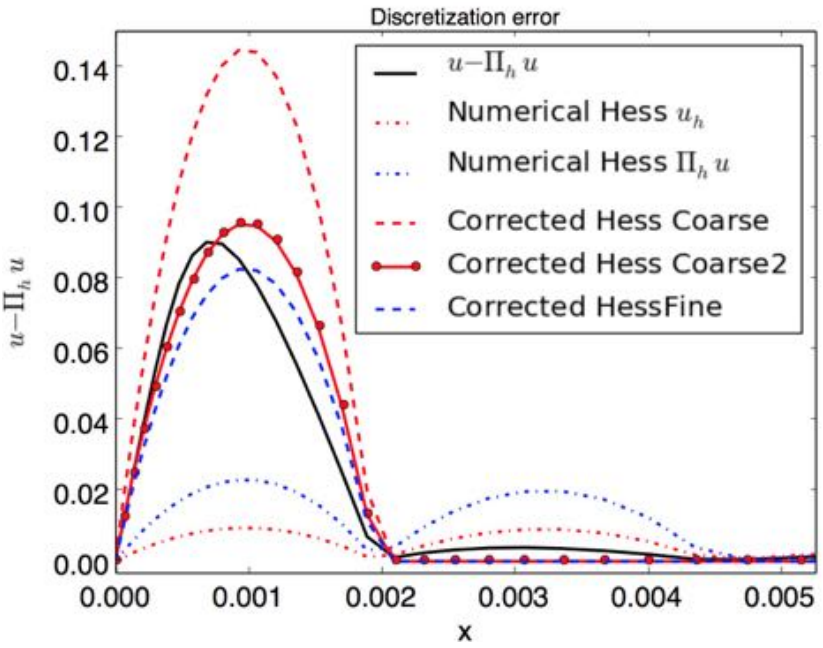


Figure 4.10: Turbulent Couette flow: Comparison of reconstructions of $u - \Pi_h u$ in the first elements using standard hessian reconstruction and equivalent hessian boundary corrections.

of $\varphi_{h|K_i}$ on each element is determined by a set of given nodal values and the functions φ_i – defined by setting the i^{th} node to 1 and the other to 0 – form a basis of $V_h(\Omega)$. The residual vector of the discrete problem is defined by

$$\Psi_h^i(W_h) = \int_\Omega \Psi(W_h)\varphi_i \mathrm{d}\Omega.$$

Finite Volume discretization

In the standard Finite Volume approach, the domain Ω is split in a finite number of cells $\cup_i C_i = \Omega$ and the residual is defined as

$$\Psi_h^i(W) = \int_{C_i} \Psi(W)\mathrm{d}\Omega,$$

for any continuous field W, where C^i is the cell associated with vertex P_i. The problem is discretized defining the vector $W_i = \int_{C_i} W\mathrm{d}\Omega$, supposed to be the mean value of the field on each cell and a reconstruction operator $\mathcal{U}$ which recovers a field W on each cell from the discrete values W_i. This approach is equivalent to Discontinuous Galerkin approach [Barth 2002, Dervieux 1992, Selmin 1996].

For flow problems, the advection terms can be written as a divergence, thus using the Green formulae, the integral of advection term is the integral of the advection fluxes over the boundaries of the cell :

$$\int_{C^i} \nabla \cdot \Psi \mathrm{d}\Omega = \int_{\partial C^i} \Psi \cdot \mathbf{n} \mathrm{d}\gamma,$$

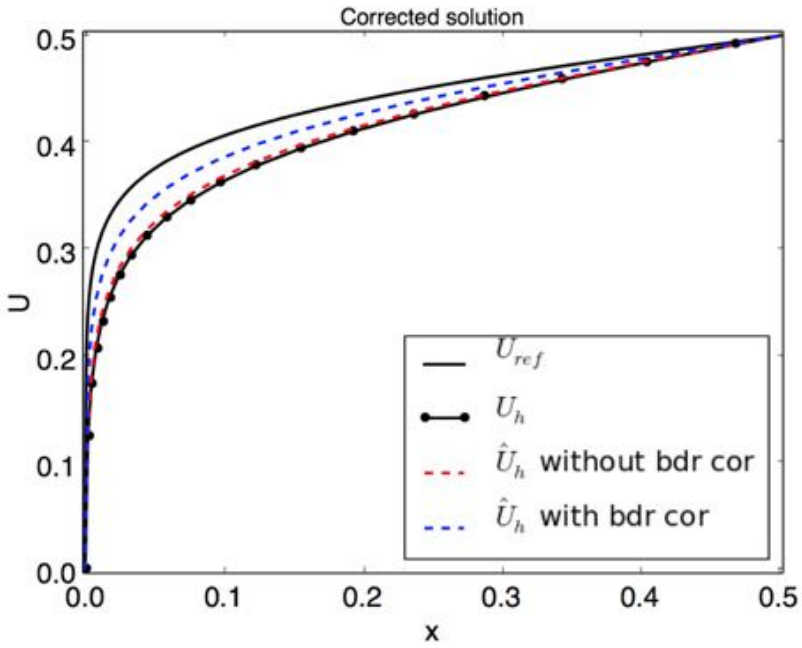


Figure 4.11: Turbulent Couette flow: Comparison of corrected velocity profiles with constant source term using approximate hessian reconstruction of interpolation error with and without equivalent hessian boundary correction.

where $\mathbf{n}$ is the inward normal. In our case, we will consider a vertex-centered formulation described here after.

4.1.3 Finite Volume Corrector

Influence stencil of advection: As seen in Section 1.1.2, the convective part of fluxes for cell C_i writes

$$\mathbf{F}_i = \sum_{e \in \mathcal{E}_i} \Phi^{inv}(e) = \sum_{P_j \ \in \mathcal{V}^1(P_i)} \Phi_{ij}^{HLLC}(W_{ij}, W_{ji}, \mathbf{n}_{ij}). \tag{4.20}$$

For a given neighbour P_j of P_i, W_{ij} and W_{ji} are computed with a MUSCL extrapolation as stated previously. In particular, as can be seen in Figure 1.14, W_{ji} depends on the centered gradient and thus on the nodal values of P_i and P_j but also on the upwind gradient. As the upwind gradient is computed with the nodal values of the upwind tetrahedron/triangle K_{ji}, W_{ji} and in turn the HLLC flux associated with the edge P_iP_j depend on the nodal values of the neighbours of P_j. The inviscid fluxes of the cell C_i and thus the residual of node P_i depend on the neighbours of P_i and the neighbours of its neighbours which forms the second order ball: $\mathcal{V}^2(P_i)$.

Influence Stencil of viscous terms: As each element produces a viscous flux for each of its nodes we can deduce that the total viscous fluxes for a given cell/node i is:

$$\mathbf{S}_i = \sum_{K \ni P_i} \Phi^{visc}_{i,K}(W_i, W_j, W_k, W_l), \tag{4.21}$$

so that the viscous fluxes of a node only depend on the first order ball of P_i: $\mathcal{V}^1(P_i)$.

Influence Stencil of source terms: As the source terms of the Spalart-Allmaras turbulence model involve the gradients of the mean flow which are computed with a L^2-projection, their influence stencil is the same as viscous terms *e.g.* the first order ball: $\mathcal{V}^1(P_i)$.

Corrector's source term computation: In the Finite Volume approach, the nodal residual is defined as the integral of the continuous residual of a reconstructed solution over a cell or equivalently for inviscid flow as the integral of the fluxes through the boundary of the cell. In that sense, for a given mesh an infinite family of continuous fields $\widehat{W}$ verifies $\Psi_h(\widehat{W}) = 0$, the reconstructed continuous field based on the discrete solution is one of them.

However, if we consider finer meshes, the only field verifying $\Psi_{h/n}(\widehat{W}) = 0$ for any sub-mesh (*i.e.* $\forall n$) is the solution of the continuous problem: $\widehat{W} = W$. Thus, the residual of the discrete solution interpolated on a finer mesh $\Psi_{h/2}(\Pi_{h/2}W_h) = \varepsilon_{h/2}$ gives us a useful indication of the quality of the solution. We propose the following approach, inspired by multigrid methods, to compute a corrected solution $\widetilde{W}_h$ from the residual $\Psi_{h/2}(\Pi_{h/2}W_h)$:

Algorithm 5 Finite Volume corrector computation

1. Compute the solution of $\Psi_h(W_h) = 0$ on the initial mesh: W_h
2. Linearly interpolate the solution on the $h/2$-mesh: $\Pi_{h/2}W_h$
3. Compute the $h/2$-mesh residual: $\Psi_{h/2}(\Pi_{h/2}W_h) = \varepsilon_{h/2}$
4. Accumulate the $h/2$-mesh residual on the h-mesh: $S_h = \mathcal{A}_h^{h/2}(\varepsilon_{h/2})$
5. Perform a few iterations of the flow solver with S_h as source term to compute the corrected solution: $\Psi_h(\widetilde{W}_h) = S_h$

In the general case where the size is divided by n, to accumulate on the h-mesh the residual computed on the h/n-mesh, we split the accumulation process depending on the possible configurations. We consider a h-mesh vertex P_i, the different cases are:

- If the h/n-mesh vertex Q_k coincides with the h-mesh vertex P_i, then the full residual is accumulated: $\varepsilon_{h/n}(P_i)$
- If the h/n-mesh vertex Q_k belongs to a h-mesh edge e_j containing P_i then we compute the barycentric coordinates $\beta_i(e_j)$ of Q_k w.r.t P_i on edge e_j. The accumulated residual is $\beta_i(e_j)\varepsilon_{h/n}(Q_k)$

- If the h/n-mesh vertex Q_k belongs to a h-mesh face f_j containing P_i then we compute the barycentric coordinates $\beta_i(f_j)$ of Q_k w.r.t P_i on face f_j. The accumulated residual is $\beta_i(f_j)\varepsilon_{h/n}(Q_k)$
- If the h/n-mesh vertex Q_k is inside a h-mesh element K_j containing P_i then we compute the barycentric coordinates $\beta_i(K_j)$ of Q_k w.r.t P_i in element K_j. The accumulated residual is $\beta_i(K_j)\varepsilon_{h/n}(Q_k)$.

For the h-mesh vertex P_i, the accumulation process is expressed as

$$\begin{aligned}[\mathcal{A}_h^{h/n}(\varepsilon_{h/n})][P_i] = \; & \varepsilon_{h/n}(P_i) + \textstyle\sum_{Q_k \in e_j | e_j \ni P_i} \beta_i(e_j)\varepsilon_{h/n}(Q_k) \\ & + \textstyle\sum_{Q_k \in f_j | f_j \ni P_i} \beta_i(f_j)\varepsilon_{h/n}(Q_k) + \sum_{Q_k \in K_j | K_j \ni P_i} \beta_i(K_j)\varepsilon_{h/n}(Q_k)\,.\end{aligned} \tag{4.22}$$

A 2D exemple of this accumulation process at the element level is illustrated in Figure 4.12 for a $h/2$-mesh.

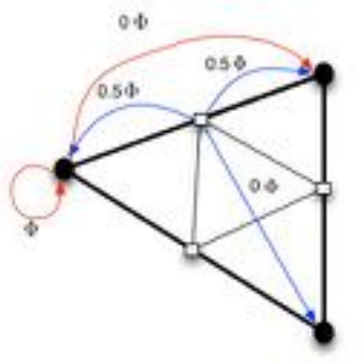


Figure 4.12: Linear accumulation process.

Using the same solver to compute the initial solution, the residual on the $h/2$-mesh and the corrected solution ensures that the corrector encompasses all the features of the solver (Extrapolation, Riemann solver, limiters ...). The number of iterations required to compute the corrected solution depends on the problem considered (essentially its nonlinearity) and the solver used. As the initial and corrected solution are relatively close to each other, the number of iterations required to compute the corrected solution is typically about 1/10 to 1/100 of the number of iterations required to compute the initial solution. Finer subdivision ($h/4$, $h/8$, ...) of the initial mesh can also be used to improve the predictions. It is very important to note that linearly interpolating the discrete solution W_h on the $h/2$-mesh (*i.e.* using $\Pi_{h/2}(W_h)$) is nonetheless convenient but also essential to the precision of the correction. Replacing $\Pi_{h/2}(W_h)$ by a better polynomial reconstruction tends to reduce the efficiency of the correction. Indeed, as we want to quantify the error of W_h with respect to the $h/2$-mesh, it is counter productive to try to improve $\Pi_{h/2}(W_h)$.

Numerical Implementation. Dividing the mesh size by two multiplies the number of elements by 4 in 2D and 8 in 3D, it would be thus particularly expensive to fully generate the sub-meshes for large practical cases (hundred of million of elements, see Part 4.3.4). As the residual

computation and accumulation are local operations, they can be done at a local level, by virtually generating the sub-meshes of the vicinity of each vertex. That way no additional memory cost is implied and the process can be straightforwardly done in parallel. Similarly, multiple computations of the fluxes for the same element can be avoided with a careful ordering detailed here after.

We recall that from Relations (1.5), (4.20), (4.21) and (1.15) we can express the nodal residual as the sum of different terms that can be computed separately:

$$\begin{aligned}\Psi_h^i = - \sum_{P_j \in \mathcal{V}^1(P_i)} \Phi_{ij}^{HLLC}(W_{ij}, W_{ji}, \mathbf{n}_{ij}) \quad &+ \quad \sum_{K \ni P_i} \Phi_{i,K}^{visc}(W_i, W_j, W_k, W_l) \\ &+ \quad \mathbf{Q}_i + \sum_{F^{bdy} \ni P_i} \Phi_{i,F^{bdy}}^{bdy}(W_i, W_j, W_k).\end{aligned} \tag{4.23}$$

One can see – as detailed in Section 1.1.2 – that the computation of the flux of a given vertex requires the information of the vertices of the first order ball (neighbours of the vertex) and some vertices of the second order ball (neighbours of the neighbours).

Figure 4.13 shows an example of a 2D mesh (black lines) and its subdivision by two (grey lines). In the following, we analyse different possible contributions of vertices of the $h/2$-mesh to the computation of the residual of the vertices belonging to the h-mesh. Let A and B be two vertices of the h-mesh connected by an edge as shown in Figure 4.13. We will provide which vertices of the $h/2$-mesh are involved in computing the source term of vertex B. Starting with the solution W_h on the h-mesh (black circles and black lines), it is projected linearly on the $h/2$-mesh (grey lines and white squares). The residual is computed at each vertices of the $h/2$-mesh using Relation (4.23). Then, the residuals are accumulated back on the h-mesh using the operator defined by Relation (4.22).

We first analyze the contribution associated with each kind of vertices in the $h/2$-mesh. We can see in Figure 4.13 that the residual of vertex C, which belongs to the $h/2$-mesh, contributes as shown in Figure 4.12 to the source term of vertices A and B of the h-mesh with a weighting factor 1/2. The residual of vertex D contributes to the source term of vertex A but not of vertex B (barycentric coordinate 0), however, its solution is required to compute the residual of vertex C and thus contributes indirectly to the source term of vertex B. Finally, the residual of vertex E only contributes to the source term of A, but, as it is contained in an upwind triangle of the edge DC (see Section 1.1.2), it contributes through the MUSCL extrapolation to the residual of vertex C and thus indirectly to the source term of vertex B.

In turn, the source term of vertex B depends on the nodal values of vertices in its third order ball in the $h/2$-mesh (neighbours of the neighbours of its neighbours) which is contained in the second order ball of the h-mesh. Therefore, if we want to compute the source term of vertex B, we do not need the whole $h/2$-mesh but only the vicinity of vertex B, which can be generated on the fly. The subdivision of the second order ball of vertex B in the h-mesh is *a-priori* sufficient, but with a more careful analysis it can be reduced to the first order ball by an appropriate treatment.

To this end, let us consider the first order ball of B in the h-mesh. After its subdivision, we analyze more precisely the contributions of the vertices of the $h/2$-mesh to the source term of B. The residual of the vertices located on the internal edges of the ball of B (like vertex

C) need to be computed. As seen above, the vertices of the h-mesh in the ball of B do not contribute directly to the source term of B. And similarly, the residual of the vertices located on the external edges of the ball of B (like vertex D) do not need to be computed. These vertices only contributes through gradients and extrapolations to the residual computed in the first order ball of B. It is interesting to note that the flux of the convective term of the residual of vertex C writes

$$\sum_{e \in \mathcal{E}(C)} \Phi(e) = \Phi(AC) + \Phi(CD) + \Phi(CB) + ...,$$

so that the contribution from vertex C to the source term of vertex A and B will be both of the form

$$S_{A/B} = ... + 0.5\,(\Phi(AC) + \Phi(CD) + \Phi(CB) + ...) + ...$$

Thus, the contribution of each edge can be computed independently only once and added to both source terms of A and B. Note here that this is possible thanks to the linearity of the accumulation process.

In order to compute these terms only once, we generate the subdivided local sub-mesh around vertex B as shown in Figure 4.14. The grey part of the $h/2$-mesh is not needed and not generated at all. The black part of the $h/2$-mesh is generated for the interpolation of the nodal values and used to compute gradients and extrapolations. The convective fluxes are only computed for the red edges. That way, only half of the residual of vertex C is computed and accumulated to vertices A and B. The other half of the residual of C is computed using the subdivided local sub-mesh around vertex A. Note that the upwind and downwind elements (considered for the MUSCL extrapolation) of the red edges always exist in the subdivided local sub-mesh. Computing separately the convective flux contributions to the source term enables to only consider the first order ball of each vertex for the subdivided local sub-mesh.

Similarly, the viscous fluxes are computed only on the sub-triangles encompassed by the red edges to avoid the multiple computations of the viscous contributions. The central sub-triangle contribution is computed by the vertex with the smallest ID in the triangle of h-mesh and added to the source term of the three vertices of the triangle.

As regards nodal source terms, each vertex of the h-mesh computes its own contribution on the $h/2$-mesh. And, the contribution of each vertex of the $h/2$-mesh located on the edges of the h-mesh is computed by the edges's vertex with the smallest ID. For instance, in Figure 4.14, if A has a smaller ID than B then the nodal source term flux of C is computed by A.

Finally, boundary terms are simply computed by the closest h-mesh vertex. In our case, if edge AB was a boundary edge, the contribution of AC would be computed by A and BC by B.

In conclusion, using this strategy, only the first order ball of each h-mesh vertex is considered and subdivided, and each contribution is computed only once which minimizes computational costs.

The approach is similar in 3D. However, when we split the edges of a tetrahedron of the h-mesh (let us assume that this tetrahedron is regular, with all its edges of length h), we are left with (see Figure 4.15):

- four tetrahedra of size $h/2$, one issued from each vertex of the initial tetrahedron,
- and an octahedron in the middle which can be divided in four tetrahedra.

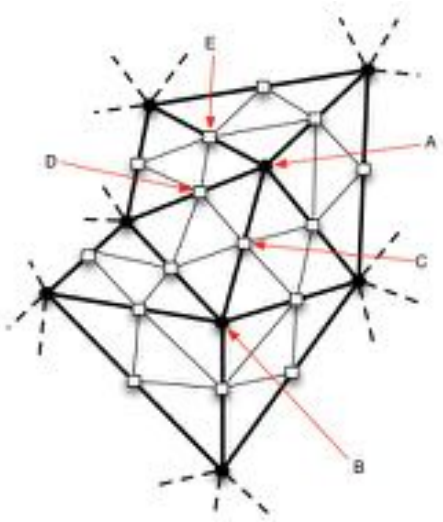


Figure 4.13: Mesh Subdivision.

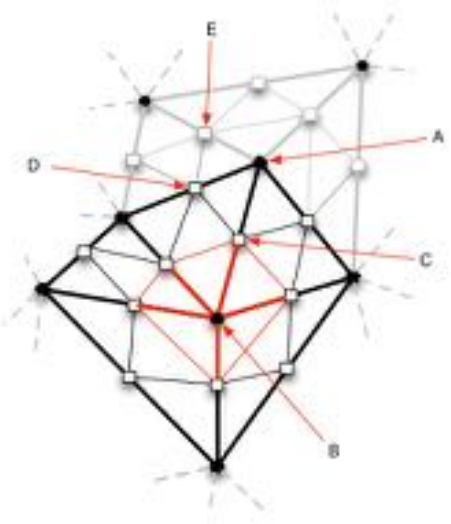


Figure 4.14: Sub-mesh used for the computation of the source term for vertex B

There are three possible choices (diagonal edges) to split the octahedron into tetrahedra and this choice may favour a particular direction (see red dashed edges in Figure 4.15). Moreover, we need to guarantee that the chosen diagonal edge has an upwind and a downwind tetrahedra in the sub-mesh of at least one vertex of the h-tetrahedron. It is usual to end up in a situation where this is impossible. Even in a regular case, for a given tetrahedron, we can see that whatever vertex we choose, the three diagonals have either an upwind or downwind tetrahedron which is out of the ball of the vertex.

For that reason, we decide to add a point at the center of the octahedron and to split it into eight tetrahedra as shown in Figure 4.15. These tetrahedra have three edges of length $h/2$ and three edges of length $\frac{h}{2\sqrt{2}}$. Thus, each h-mesh tetrahedron is split into 12 $h/2$-mesh tetrahedra. This subdivision guarantees that all internal edges have always simultaneously an upwind and a downwind tetrahedra in the subdivision of at least one first order ball.

Similarly to the 2D case, in order to compute the fluxes only once and to guarantee that each edge has an upwind and a downwind tetrahedra in the local subdivided sub-mesh when we compute the HLLC flux, we treat them as follows:

- the $h/2$-edges issued from the h-edges (bold black lines in Figure 4.15) are treated as in 2D by the closest h-mesh vertex (three for each vertex). For instance, vertex 1 will compute the $h/2$-edges 1-8, 1-9 and 1-10.

- the $h/2$-edges on the faces (black fine lines in Figure 4.15) are computed by the closest h-mesh vertex (three for each vertex). For instance, vertex 1 will compute the $h/2$-edges 8-9, 8-10 and 9-10.

- the $h/2$-edges occurring from the subdivision of the octahedron and connecting the central vertex to a mid-edge vertex (red lines in Figure 4.15) are treated by the h-mesh vertex

of the h-edge containing the mid-edge vertex, with the smallest global ID (as indicated in Figure 4.15).

As regard to the viscous residuals, for each h-mesh tetrahedron, we choose that each vertex of the h-mesh computes the viscous terms of three tetrahedra of the $h/2$-mesh:

- the tetrahedron containing this vertex
- the two tetrahedra issued from the subdivision of the octahedron that are aligned with this vertex and the middle of the octahedron.

In Figure 4.16, the three sub-tetrahedra computed by vertex 1 are represented in red. Note that this is an arbitrary choice, and any other choice can be made as viscous terms do not involve extrapolation.

Finally, the boundary faces are treated like viscous terms in 2D: each boundary triangle is subdivided in four triangles, each sub-triangle containing a h-mesh vertex is treated by this vertex and the central triangle is treated by the vertex of the face with the smallest ID.

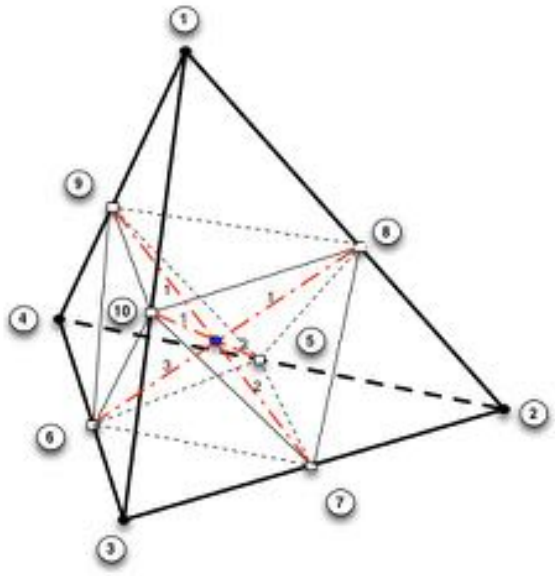


Figure 4.15: 3D Mesh Subdivision and sub-edges treatment repartition.

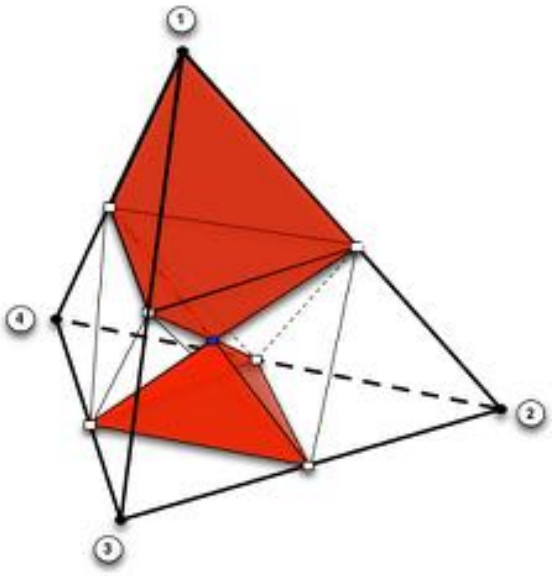


Figure 4.16: Three sub-tetrahedra treated by vertex 1 to compute the viscous fluxes.

4.1.4 Finite Element Corrector

The Finite Element corrector is based on the idea that a perfect source term would be $\Psi_h(\Pi_h W)$. As we do not have access to $\Pi_h W$ we cannot compute it but we can estimate its value. This corrector follows the idea proposed in the usual approach to estimate functional error: in PDEs the most important part of the error does not come from the local value of the solution field W but from its variations ∇W and ΔW. Thus locally reconstructing a consistent smooth solution can provide useful information.

In Finite Element approach, the solution is defined through a weak formulation (before integrations by parts) :

$$\int_\Omega \Psi(W)\varphi \,\mathrm{d}\Omega = 0, \quad \forall \varphi \in V,$$

$$\int_\Omega \Psi(W_h)\varphi_h \,\mathrm{d}\Omega = 0, \quad \forall \varphi_h \in V_h \subset V.$$

In particular, the exact solution verifies

$$\int_\Omega \Psi(W)\varphi_h \,\mathrm{d}\Omega = 0, \quad \forall \varphi_h \in V_h,$$

but generally

$$S_h = \int_\Omega \Psi(\Pi_h W)\varphi_h \,\mathrm{d}\Omega \neq 0, \quad \forall \varphi_h \in V_h.$$

We have,

$$\int_\Omega \Psi(W)\varphi_h \,\mathrm{d}\Omega = \int_\Omega \Psi(\Pi_h W + (W - \Pi_h W))\varphi_h \,\mathrm{d}\Omega = 0, \quad \forall \varphi_h \in V_h,$$

and by linearizing the fluxes in $\Pi_h W$ and W_h we have,

$$\Psi(\Pi_h W + (W - \Pi_h W)) \approx \Psi(\Pi_h W) + \left.\frac{\partial \Psi}{\partial W}\right|_{\Pi_h W} \cdot (W - \Pi_h W),$$

and

$$\Psi\left(W_h + (W - \Pi_h W)\right) \approx \Psi(W_h) + \left.\frac{\partial \Psi}{\partial W}\right|_{W_h} \cdot (W - \Pi_h W) \approx \Psi(W_h) + \left.\frac{\partial \Psi}{\partial W}\right|_{\Pi_h W} \cdot (W - \Pi_h W).$$

From these two relations we deduce for all $\varphi_h \in V_h$:

$$\begin{aligned}
\int_\Omega \Psi(\Pi_h W)\varphi_h \,\mathrm{d}\Omega \;\approx\; & \int_\Omega \Psi(\Pi_h W + (W - \Pi_h W))\varphi_h \,\mathrm{d}\Omega && (4.24)\\
& - \int_\Omega \left.\frac{\partial \Psi}{\partial W}\right|_{\Pi_h W} \cdot (W - \Pi_h W)\varphi_h \,\mathrm{d}\Omega \\
\approx\; & \int_\Omega \Psi(W_h)\varphi_h \,\mathrm{d}\Omega - \int_\Omega \Psi(W_h + W - \Pi_h W)\varphi_h \,\mathrm{d}\Omega && (4.25)\\
\approx\; & -\int_\Omega \Psi(W_h + W - \Pi_h W)\varphi_h \,\mathrm{d}\Omega. && (4.26)
\end{aligned}$$

This expression relies on the fact that W_h is relatively close to $\Pi_h W$ so that the derivatives $\left.\frac{\partial \Psi}{\partial W}\right|_{\Pi_h W} \approx \left.\frac{\partial \Psi}{\partial W}\right|_{W_h}$ are (reasonably) approximately the same. It has two major advantages. First, it does not require to compute the derivative $\left.\frac{\partial \Psi}{\partial W}\right|_{W_h}$ which saves a lot of troubles: as we mentioned previously, in linear approaches, the quality of the correction depends on the choice and the precision of these derivatives. Second, it automatically takes into account fairly well the non-linearities of the residual. As long as the higher derivatives $\left.\frac{\partial^k \Psi}{\sum_k \partial_k W}\right|_{\Pi_h W} \approx \left.\frac{\partial^k \Psi}{\sum_k \partial_k W}\right|_{W_h}$

are relatively close, which is a reasonable assumption as the correction $W - \Pi_h W$ is a purely local correction, the Taylor developments mentioned previously can be expanded.

In Relation (4.26), we still need to estimate $W - \Pi_h W$ which represents a loss of smoothness due to the discretization of the solution, it does not depend on the absolute value of the solution but on its variations. The term $W_h + W - \Pi_h W$ can be thus estimated by reconstructing a smooth solution $\mathcal{U}_h^{k+1}(W_h)$ [Clément 1975, Zienkiewicz 1992a, Bank 1993] from the actual numerical solution W_h as shown in Figure 4.17. Finally the source term can be expressed as

$$\int_\Omega \Psi(\mathcal{U}_h^{k+1}(W_h))\varphi_h \, \mathrm{d}\Omega = S_h(\varphi_h), \quad \forall \varphi_h \in V_h, \tag{4.27}$$

where $\mathcal{U}^{k+1}$ holds for the $(k+1)^{th}$ order reconstruction if the solution is of order k. We propose the following procedure to compute the corrector:

Algorithm 6 Finite Elements corrector computation

1. Compute the solution on the initial mesh: W_h
2. Reconstruct the smooth solution on each element: $\mathcal{U}_h^{k+1}(W_h)$
3. Compute the residual: $S_h^i = R_h^i(\mathcal{U}_h^{k+1}(W_h)) = \displaystyle\int_\Omega \Psi(\mathcal{U}_h^{k+1}(W_h))\varphi_h^i \, \mathrm{d}\Omega$
4. Perform a few iterations of the flow solver with S_h as source term to compute the corrected solution: $\Psi_h(\widetilde{W_h}) = S_h$

Note that there is no need for a finer mesh in this case as the integrals are computed on the same elements, but more quadrature points are required to have a consistent computation of the source term given by Relation (4.27).

4.2 Numerical results: manufactured solutions

The main interest of manufactured solutions is to be able to set-up a problem where we know the exact solution W even if the equations are complex such as the Navier-Stokes equations. There is a very few known analytical solutions to the Navier-Stokes equations, not to mention the RANS equations, this is why Roache [Roache 1998] proposed to add a well chosen source term to the equations so that a chosen field W be the exact solution. By injecting the chosen analytical solution W in the equation we obtain a defect residual which can be used as a source term in the equation:

$$\Psi_h(W_h) = \Pi_h(\Psi(W)).$$

In the sequel, given the analytical solution of a problem W and the numerical solution W_h, we split the approximation error $W - W_h$ in the interpolation error and the implicit error as

$$W - W_h = \underbrace{W - \Pi_h W}_{\text{Interpolation Error}} + \underbrace{\Pi_h W - W_h}_{\text{Implicit Error}}.$$

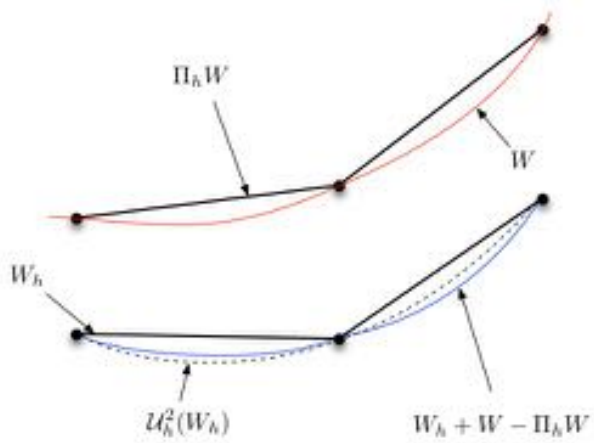


Figure 4.17: Reconstruction of a smoother solution from a discrete solution (dashed line) compared to the exact solution (red line) and the reconstructed solution with the exact defect (blue line).

The interpolation error is entirely related to the discretization of the solution and can only be improved by a refinement of the mesh. Here, we are interested in the implicit error as we want to improve the discrete value of our solution. We can thus compute analytically the exact implicit error:

$$\mathcal{E}_{Exa}^{Imp} = \sqrt{\int_\Omega \|\Pi_h W - W_h\|^2 \, \mathrm{d}\Omega},$$

and the estimated implicit error:

$$\mathcal{E}_{Est}^{Imp} = \sqrt{\int_\Omega \|\widetilde{W}_h - W_h\|^2 \, \mathrm{d}\Omega},$$

as they can be expressed element-wise as polynomial expressions. Similarly, we define a function j computed from W, and we define the exact functional error:

$$\mathcal{E}_{Exa}^{Fun} = j(W_h) - j(\Pi_h W),$$

the estimated functional error:

$$\mathcal{E}_{Est}^{Fun} = j(W_h) - j(\widetilde{W}_h),$$

and the corrected functional error:

$$\mathcal{E}_{Cor}^{Fun} = j(\widetilde{W}_h) - j(\Pi_h W).$$

Note that the interpolation error is not taken into account here as we compare the numerical functional $j(W_h)$ to $j(\Pi_h W)$ which is an approximation of the exact $j(W)$. However, this interpolation error can be taken into account by traditional error estimation [Chen 2007, Loseille 2011a].

In the sequel, we will use manufactured solution to specify the exact solution.

4.2.1 Finite elements 1D nonlinear advection diffusion problem

We present here a simple one dimensional non-linear example of the use of *a-posteriori* corrector for finite elements. The equation solved is

$$\mu \frac{\partial^2 u}{\partial x^2} + u \frac{\partial u}{\partial x} + f = 0,$$

over the segment $[0, 1]$, with $\mu = 0.01 + 0.1x^2$. The source term is chosen as

$$f(x) = -\mu \exp(-\lambda x)\lambda^2 - (\exp(-\lambda x) - 1 - x(\exp(-\lambda) - 2))(-\lambda \exp(-\lambda x) - (\exp(-\lambda) - 2)),$$

so that the exact solution of the problem is

$$u = \exp(-\lambda x) - 1 - x(\exp(-\lambda) - 2)),$$

with $\lambda = 100$. The λ value can be used to tune the problem stiffness while the value of μ determines the importance of the non-linearity. As the variations of the solution are very steep near 0, we distribute the 1D mesh vertices quadratically:

$$P_i : x_i = \left(\frac{i}{N}\right)^2,$$

for a mesh having N vertices. The problem is solved directly with 10 to 20 Newton-Raphson iterations with an exact Jacobian, and the corrected solution $\tilde{u}_h$ is computed with a few additional iterations (typically 3). For comparison, the standard linear corrector is simply the first iteration of our proposed non-linear correction. Thus, we distinguish the estimation of the implicit error and the functional error with the standard linear corrector and with our non-linear corrector as the linear estimation $\mathcal{E}^{Imp}_{LinEst}$ and the non-linear estimation $\mathcal{E}^{Imp}_{NLEst}$.

The considered functional here is

$$j(u) = \int_0^{0.5} u \, \mathrm{d}x.$$

j is a simple integral but it can be any functional output (lift, drag, noise, ...), see Section 4.3.2.

The resulting solutions are shown in Figure 4.18. We can see the influence of the non-linear advection term on the primal and corrected solutions. Both linear and non-linear correctors improves the solution and can thus be used to compute a pertinent estimation of the error as can be seen in Figures 4.19-(left) and (right). However, due to the non-linear term, the linear correction significantly under estimates the error, while the non-linear correction gives a much more precise estimation. Figures 4.20-(left) and (right) show the convergence of the resulting functional and implicit errors of the primal and corrected solutions. As expected, both corrected solutions yield lower errors but the non-linear correction yields much lower error levels compared to the linear correction. More interestingly, we can see in Figure 4.20 that the functional corrected errors converge at a faster rate than the exact functional error. This again shows that the corrector is accurate. Note that this convergence can be achieved because we are dealing with the implicit part of the functional error. The interpolation error remains of the same order and thus in turn dominates the total error.

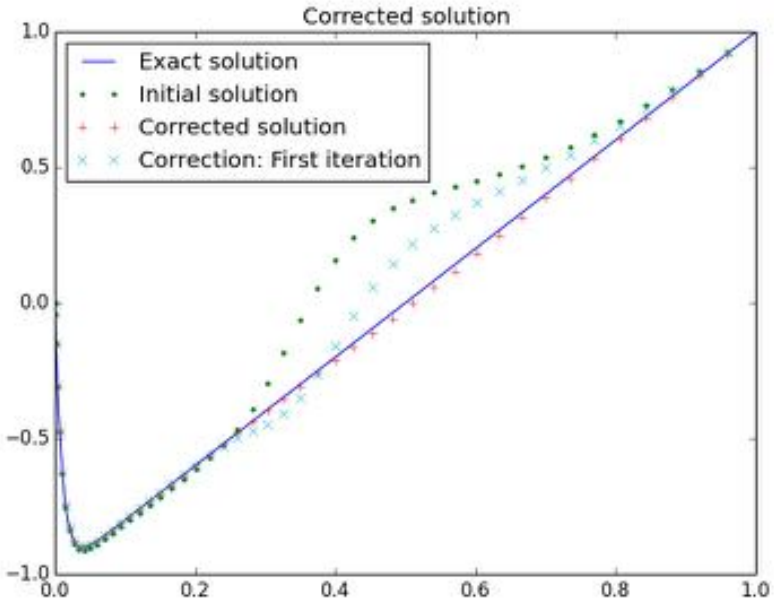


Figure 4.18: Comparison of linear and non-linear 1D corrector with a manufactured solution.

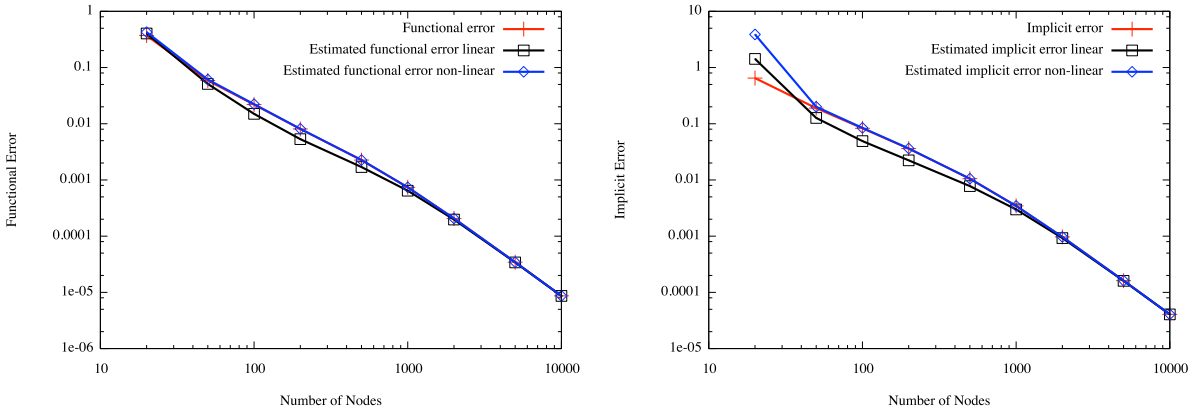


Figure 4.19: Convergence of the functional error $\mathcal{E}^{Fun}_{Exa}$ (left) and implicit error $\mathcal{E}^{Imp}_{Exa}$ (right), their linear estimation $\mathcal{E}^{Fun}_{LinEst}$ and $\mathcal{E}^{Fun}_{LinEst}$, and non-linear estimation $\mathcal{E}^{Fun}_{NLEst}$ and $\mathcal{E}^{Fun}_{NLEst}$.

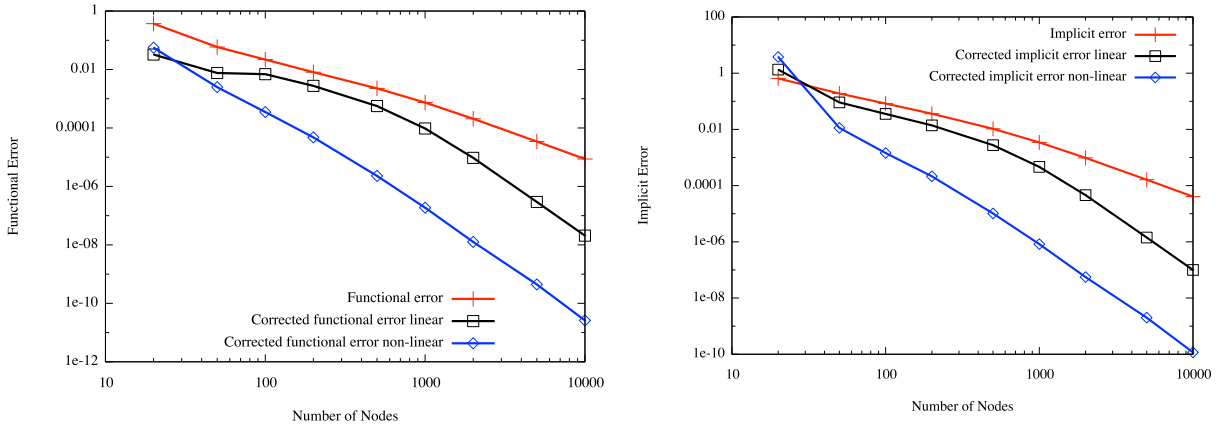


Figure 4.20: Convergence of the functional error $\mathcal{E}^{Fun}_{Exa}$ and implicit error $\mathcal{E}^{Imp}_{Exa}$ of the primal solution, their linear correction $\mathcal{E}^{Fun}_{LinCor}$ and $\mathcal{E}^{Imp}_{LinCor}$ and non-linear correction $\mathcal{E}^{Fun}_{NLCor}$ and $\mathcal{E}^{Imp}_{NLCor}$.

4.2.2 Finite volume 2D Navier-Stokes equations

We then validate the *a priori* Finite Volume non-linear correction for Navier-Stokes equations with the same approach. The analytic solution is chosen to be

$$\begin{aligned} \rho &= 1. + 0.1 * \cos(x) \\ u &= 1. + \exp(-\frac{y^2}{10}) \\ v &= 0. + y * \exp(-\frac{y^2}{10}) \\ p &= 1. + 0.1 * \cos(x * y). \end{aligned}$$

The Navier-Stokes equations are solved on the unit square domain, with our in-house flow solver `Wolf` described in Section 1.1. The source term is here integrated cell-wise. The implicit error convergence of all variables is shown in Figure 4.21. Although the corrected errors do not converge at a faster rate as in 1D, they remain smaller than the error on the primal solution. Typically, the corrected solution yields the same error level as the primal solution on a mesh having 2.5 times more vertices. Moreover, the corrector successfully improves all variables and in turn, any functional computed with those variables. It thus provides a pertinent estimation of the implicit error of the primal solution.

4.3 Numerical results: applied cases

4.3.1 2D Laminar NACA0012

We now validate the present corrector in an adaptive context on a NACA0012 geometry, at a Reynolds number of 500 with a purely laminar viscous flow. The Mach number is fixed to $M = 0.1$ and the angle of attack to $\alpha = 3°$. Adapted meshes are generated iteratively with

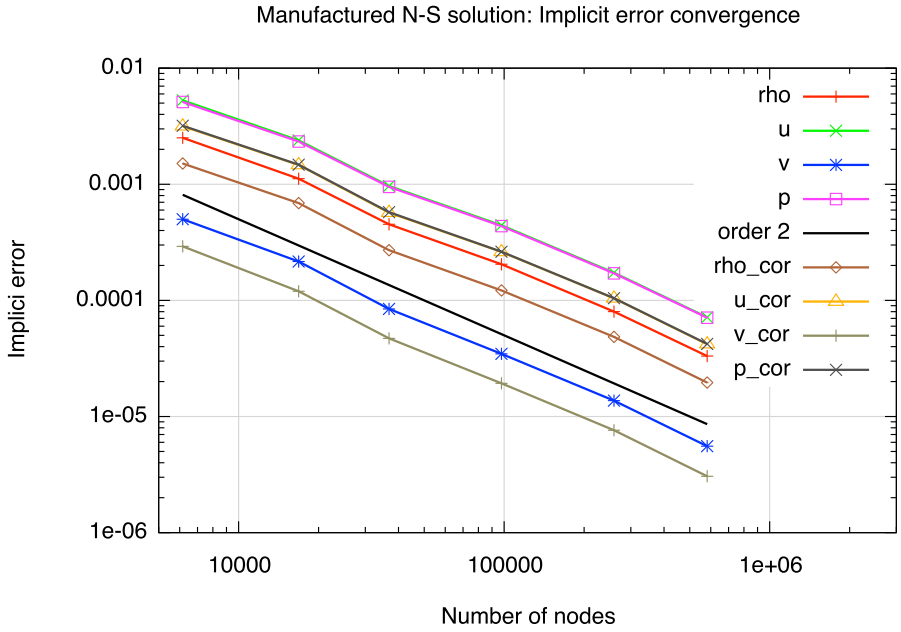


Figure 4.21: Convergence of the implicit error for primal and corrected solution of the 2D Navier-Stokes equations with a manufactured solution.

`feflo.a` [Loseille 2013, A. Loseille 2017] using a goal-oriented error estimate on the drag coefficient [Loseille 2010]. An example of the resulting meshes and solutions is shown in Figure 4.22.

Figure 4.24 shows the convergence of the drag computed for both the initial (red) and corrected (blue) solution. We can see that corrector moderately improves the solution and the computation of the drag coefficient. The corrected solution on a 5×10^3 nodes grid provides the same result as the solution on a 8×10^3 nodes grid. The moderate improvement is due to, the mesh adaptation context that introduces a side effect: an adapted mesh with four times the number of vertices of a given adapted mesh is not its subdivision by two. Indeed, it has been further improved by the adaptation and provides a better solution than a simple subdivision. We will try to quantify this effect with the next test case.

Still, the primal and corrected solutions converge to the same value, so that the difference between both of them gives and accurate estimation of the remaining numerical error. This second property is particularly important in both adaptative and fixed grids context as it informs on how confident we can be in a given solution. It also indicates if the grid convergence study should be continued or not.

4.3.2 2D Turbulent NACA0012

The corrector is then validated on a turbulent case with the same geometry, at a Reynolds number of 5×10^5 with the Sapalart-Allmaras turbulence model. The Mach number is fixed to $M = 0.7$ and the angle of attack to $\alpha = 1°$. Adapted meshes are generated iteratively with

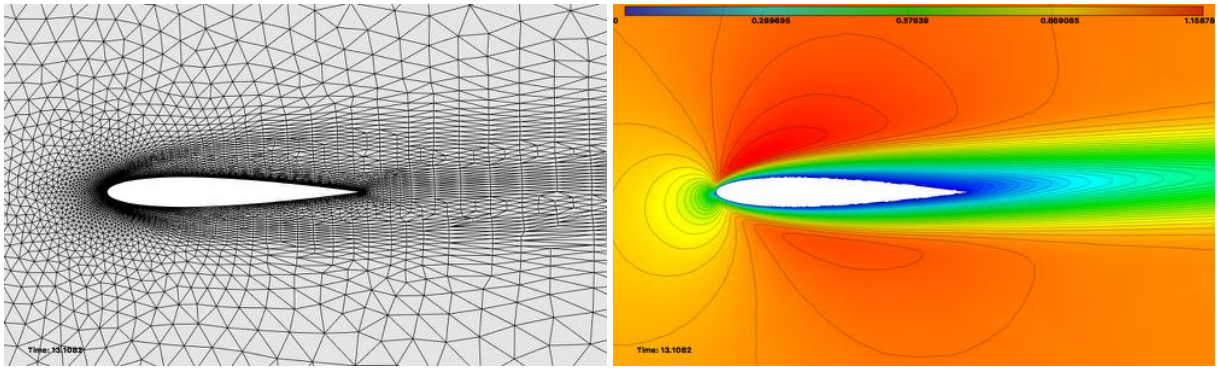


Figure 4.22: Laminar subsonic NACA0012: Velocity field contours (right) computed on an adapted mesh (left) using goal-oriented error estimate on the drag at $Re = 500$, $M = 0.1$ and $\alpha = 3°$: 8009 vertices.

`feflo.a` using a hessian-based error estimate based on the Mach number field. An example of the resulting meshes and solutions is shown in Figure 4.23. Additionally, we generate for each mesh its $h/2$ subdivision and computed the solution on it for comparison in order to analyse the influence of mesh adaptation on the performance of the corrector.

Figure 4.25 show the convergence of the drag computed with a first and second order scheme for both the initial (red) and corrected (black) solution. The blue curve represents for each mesh the drag computed on its $h/2$ mesh subdivision which actual complexity is thus four times larger. Both curves exhibits the same trend as the previous laminar example, but it is now possible to quantify how mesh adaptation improves the solution on a mesh with four times more vertices compared to the actual $h/2$-mesh subdivison. This dims the apparent efficiency of the non-linear corrector. In particular, we can see that the corrector is very efficient with a first order scheme as it yields the same solution as the $h/2$-mesh. It is slightly less efficient with a second order scheme but still satisfying.

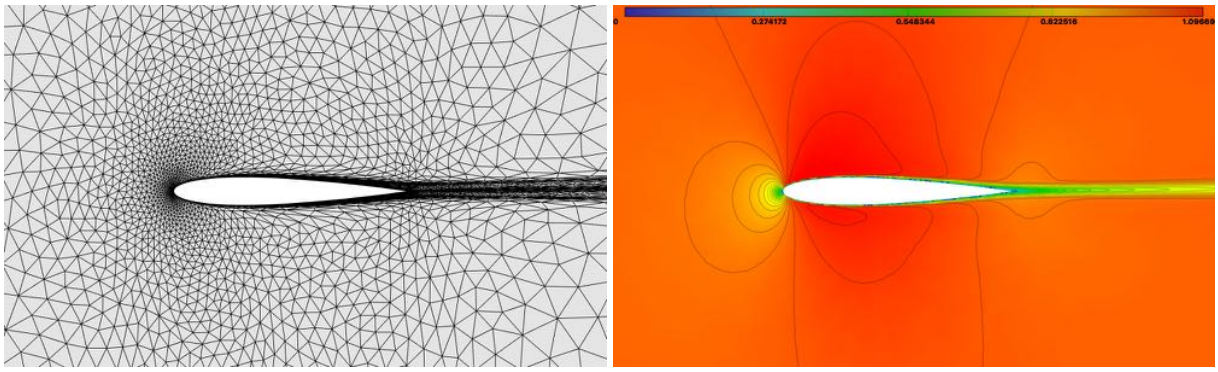


Figure 4.23: Turbulent NACA0012: Velocity fields contours (right) computed on an adapted mesh (left) generated using feature-based error estimate on the Mach number field at $Re = 5 \times 10^5$ $M = 0.7$ and $\alpha = 1°$: 5394 vertices .

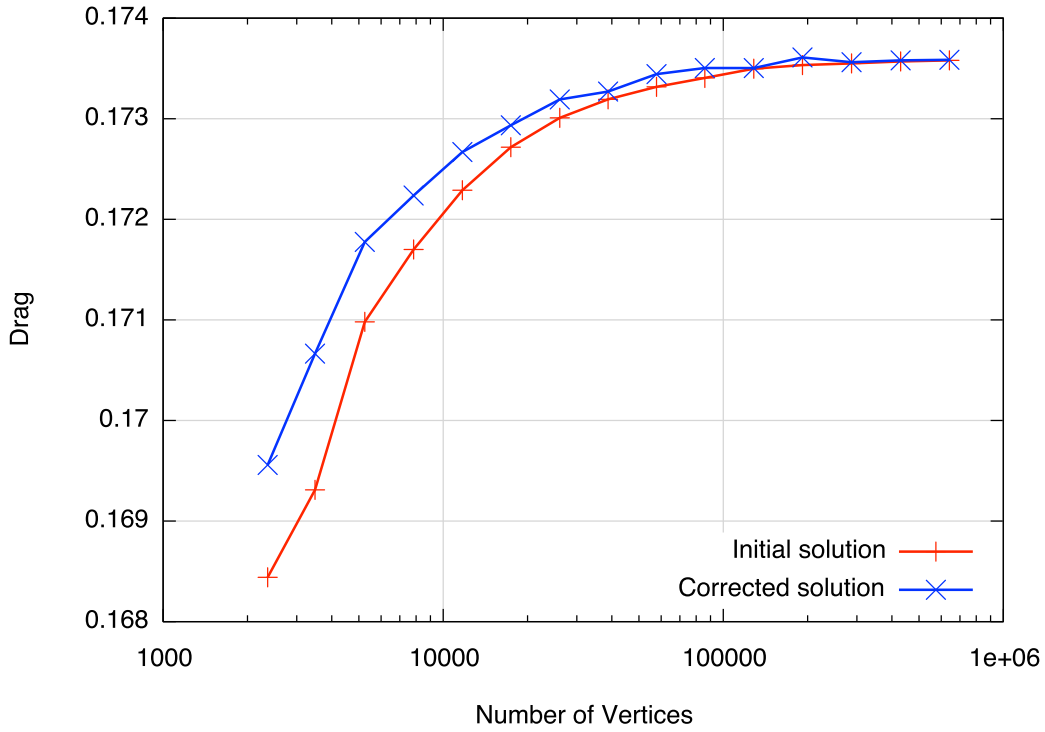


Figure 4.24: Laminar NACA0012: Convergence of the drag for initial solution (red) and corrected solution (blue).

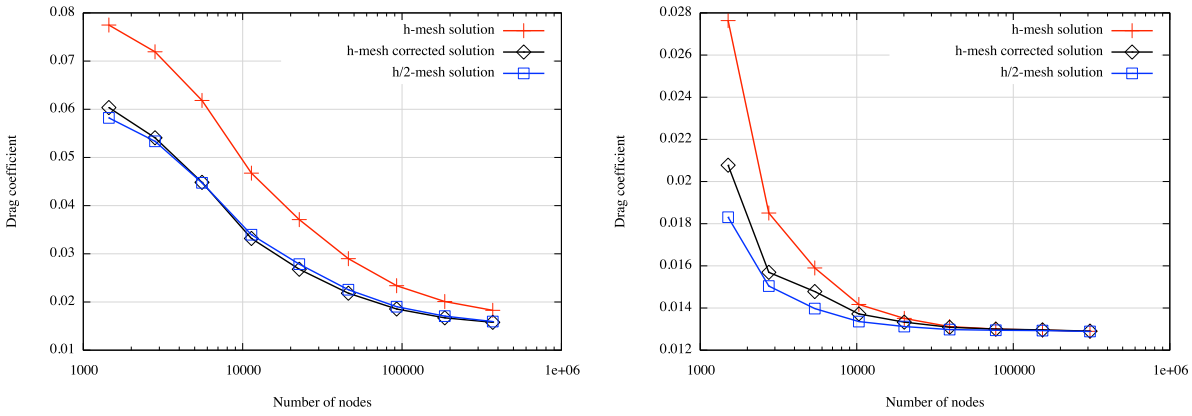


Figure 4.25: Turbulent NACA0012: Convergence of the drag with a first (left) and second (right) order scheme. Initial solution (red), the corrected solution (black) and the solution on the mesh divided by two (blue). Blue curve is plotted with a mesh complexity divided by four to be easily compared to the black curve.

4.3.3 3D ONERA M6 wing turbulent Case

We show here the improvements this method can provide on non-adapted meshes in 3D. We consider the ONERA M6 wing at Mach number $M = 0.1$, angle of attack $\alpha = 3.06°$ and Reynolds number of $Re = 11 \times 10^6$. A series of four meshes was generated using `AFLR` [Marcum 1996, Marcum 1998], with a dimensionless wall spacing based on the Reynolds number of $y^+ = 3,\ 2,\ 1,\ 0.5$. These meshes are composed of 110 009, 497 660, 2 699 130 and 19 252 616 vertices. The coarsest mesh is shown in Figure 4.26-(left).

We can see in Figure 4.26-(left) that the corrector significantly improves the drag prediction. It gives on the second mesh the same drag as the solution without correction on the third mesh. The corrector is even more efficient when the error is not optimally reduced between two grids. We also have a good estimation of the quality of our last solution, without the need of any finer mesh: despite the fine resolution of the boundary layer, we can be pretty confident on the fact that grid convergence is not achieved yet. This is mainly due to the fact that the wake of the wing is poorly discretized, as can be seen in Figure 4.26-(right). This pollutes the computation of the drag.

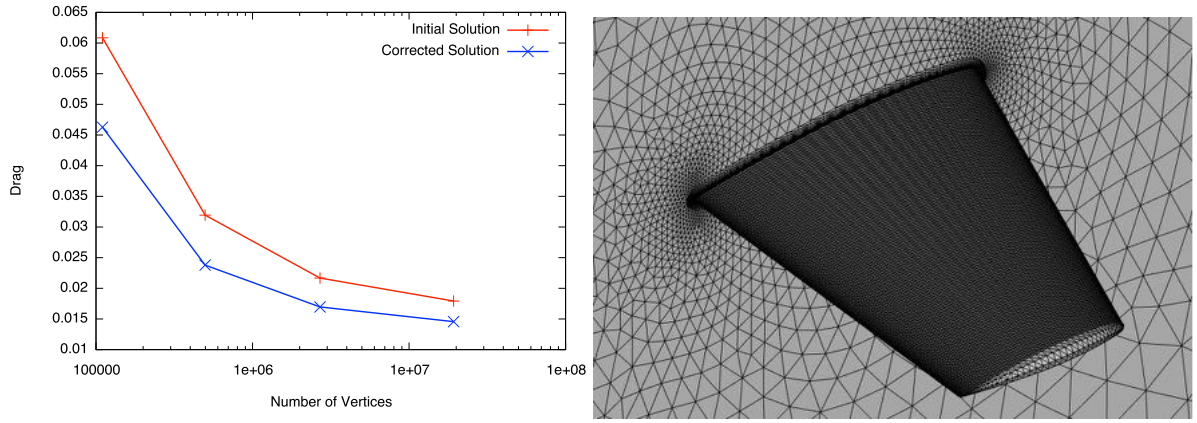


Figure 4.26: Turbulent M6 wing: Convergence of the drag (left) for the initial solution (red) and corrected solution (green) at Reynolds number of 11×10^6 and angle of attack $\alpha = 3.06°$. Coarsest mesh used (right).

4.3.4 3D inviscid supersonic case: Sonic Boom Prediction Workshop

The last example is a supersonic case on a realistic aircraft configuration. The considered geometry was proposed for the 2nd AIAA Sonic-Boom Prediction Workshop which aims at minimizing the noise produced by supersonic aircrafts on the ground. Different geometries, meshes and configurations were proposed, we focus here on the C25D flow through Euler case, see Figure 4.27. In order to properly lead the shape optimisation process, the pressure signature under the aircraft has to be accurate, this is why a grid convergence study is required to determine the numerical error. To this end, mesh adaptation has proven to be very efficient [Alauzet 2010b] and we expect the corrector to provide a second pertinent indication on the convergence of the solution.

Figure 4.28-(left) show the pressure signatures (normalized pressure difference with the free stream pressure) under the aircraft at a distance $R/L = 1$ [1], computed on tailored grids provided for the workshop. Error bars represent the estimated error of each nodal value computed with the nonlinear corrector. We can see that the corrector indicates that the error level remains relatively high, even on the 13 million nodes grid. This has been confirmed by the workshop: the grid convergence is not achieved. This is visible on the leading shock which is still relatively smooth and the shocks in the aft part of the signature ($65 < X < 80$) which are still forming. Note that the actual subdivision by two of this 13 million nodes mesh would have yielded a 104 millions nodes mesh!

Figure 4.28-(right) shows the same computations using goal-oriented mesh adaptation [Loseille 2010] which have been generated to optimally capture the pressure signature. We can see that the error bars are much smaller and tend to reduce during the adaptation process. This is especially visible on the mid-part of the signature ($50 < X < 60$). It is even more enlightening to compare the error levels predicted by the corrector on the tailored grid composed of 13 million nodes in the mid-part of the signature to the actual difference between the solutions on the tailored grid and adapted grid composed of 5.9 million nodes. The error predicted by the corrector on tailored grid matches the difference between the solution on the tailored grid (which is not converged) and the solution on the adapted grid (which is converged in this region).

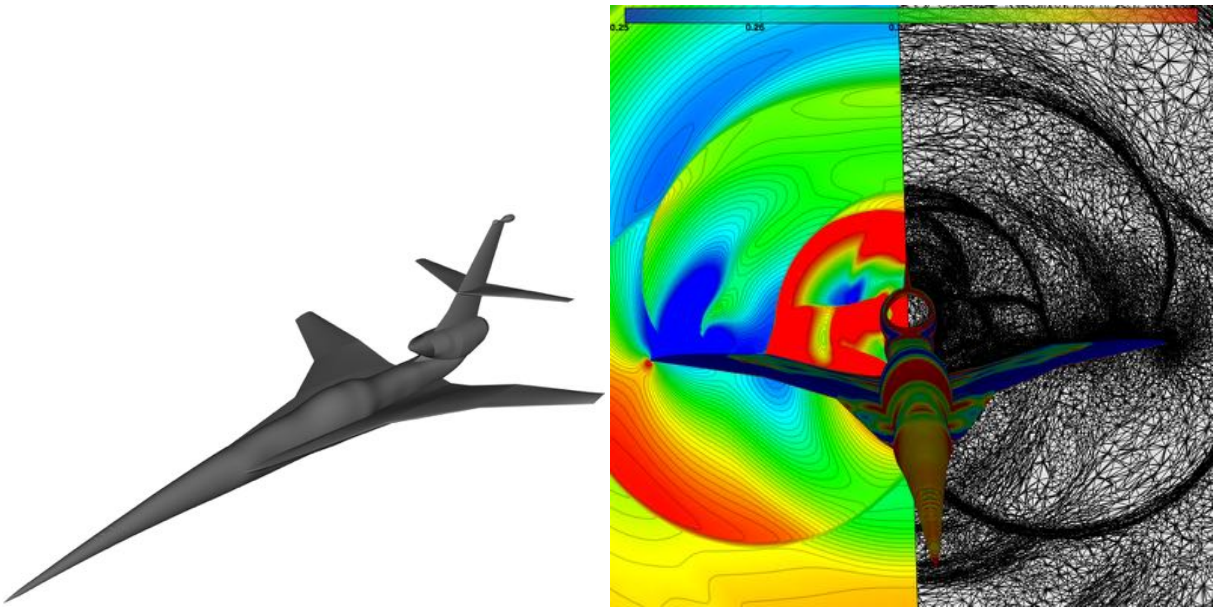


Figure 4.27: 2nd AIAA Sonic-Boom Prediction Workshop: C25D Geometry (left) and cut plane in the volume behind the aircraft (right). Mach field and adapted mesh obtained with a goal-oriented mesh adaptation on the pressure signature.

[1] R is the distance to the aircraft and L is the length of the aircraft

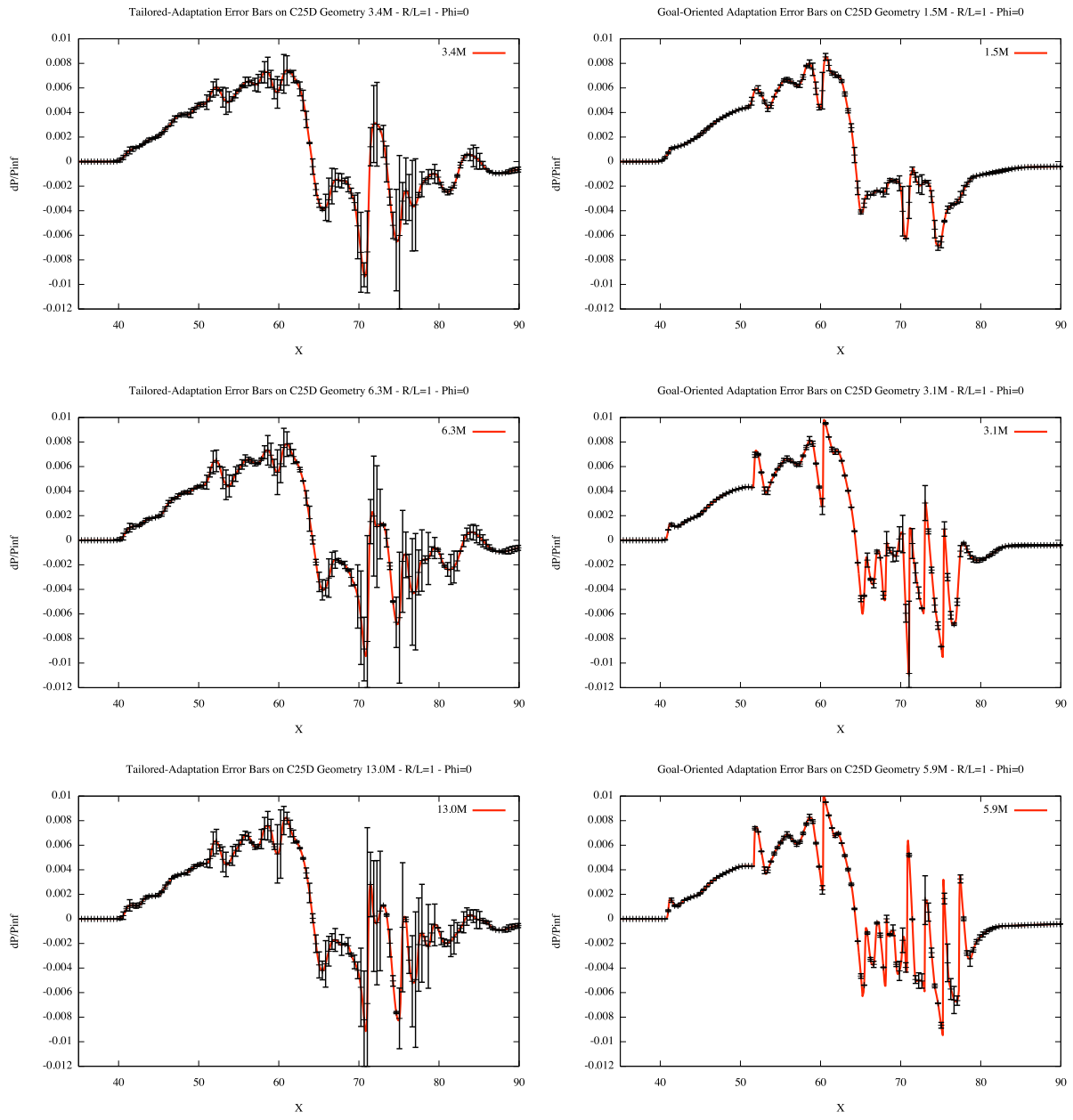


Figure 4.28: 2nd AIAA Sonic-Boom Prediction Workshop: Pressure difference under the plane (red line) for tailored meshes (left) of 3.4, 6.3 and 13 million vertices and adapted meshes (right) of 1.5, 3.1 and 5.9 million vertices and estimated error provided by nonlinear corrector (black error bars).

4.4 Conclusion and perspectives

In this Chapter we shown that the Finite Volume and Finite Element non-linear correctors provide an efficient and pertinent correction of the solution and in turn an estimation of the

numerical error. This non-linear approach has proven to bring a significant improvement for non-linear problems, removing the dependency to the choices in the linearization. It has been successfully applied to complex applied cases, both in 2D and 3D.

Furthermore, we introduced a general analytical context and analysis for numerical errors, in particular for RANS computations. It is a key step to toward RANS mesh adaptation as the analysis and minimization of this error will yield optimal meshes, as presented in the second part of this thesis. Moreover, we introduced a correction of the boundary hessian of the solutions which will prove helpful to further improve error estimation and mesh adaptation.

Beyond the assessment of the numerical errors, non-linear correction provides a coherent solution defect which is essential to Norm-Oriented sensor (see Section 1.3). It also opens the way for a non-linear adjoint. Indeed, as explained in Section 1.3, the standard adjoint analysis relies on a linearization of the problem. We thus hope to further improve the process with the following approach.

We recall that numerically, the resolution of the adjoint system consists in determining the solution variations dW due to a perturbation $\mathcal{E}$ of the residual so that

$$\Psi(W) = 0, \qquad \Psi(W + dW) = \mathcal{E}.$$

The standard adjoint theory relies on the linearization of the equations to compute the adjoint problem as

$$dW = \left(\frac{\partial \Psi}{\partial W}\right)^{-1} \mathcal{E}.$$

This is obviously limited to small variations of the solution.

A simple quadratic approximation of the residual would involve the resolution of

$$\left(\frac{\partial \Psi}{\partial W}\right) \cdot dW + \frac{1}{2} dW \cdot \left(\frac{\partial^2 \Psi}{\partial W^2}\right) \cdot dW = \mathcal{E},$$

however it is impossible to deal the quadratic terms of this expression in multi-dimension except with iterative methods. Hence, we proposed in the previous part an approach to take into account the non-linearities of the solver and compute $dW(\mathcal{E})$, for a single value of $\mathcal{E}$.

Still, in the adjoint analysis, we need a local model of $dW(\mathcal{E})$ that can be used over a whole range of values $\mathcal{E}$. A linear model is easier to use but as we saw previously, the direct linearization of the equations may lead to miscomputations and in particular we would like at least the linear model to be (almost) exact for the non-linear correction $dW(\mathcal{E})$ that we are able to compute with the non-linear corrector:

$$G \cdot \mathcal{E} = dW(\mathcal{E}).$$

This is possible in the case of a quadratic model as the quadratic term verifies

$$dW \cdot \left(\frac{\partial^2 \Psi}{\partial W^2}\right) \cdot dW = dW \cdot \left(\left.\frac{\partial \Psi}{\partial W}\right|_{W+dW} - \left.\frac{\partial \Psi}{\partial W}\right|_{W}\right).$$

Hence, for a quadratic function we have the simple expression

$$\Psi(W + dW) = \frac{1}{2}\left(\left.\frac{\partial \Psi}{\partial W}\right|_{W+dW} + \left.\frac{\partial \Psi}{\partial W}\right|_{W}\right) \cdot dW.$$

This expression is valid for only one value of dW, that we compute with our non-linear corrector.

So, we propose in the mesh adaptation context, when using the non-linear corrector to replace the standard Jacobian in the adjoint system by the mean value of the initial Jacobian and the corrected Jacobian. This approach has already shown promising results on simple 1D cases, and requires further tests on more realistic cases.

Conclusion

In this first part we detailed the implementation of our flow solver `Wolf` and the methodology to deal with turbulent RANS computations on fully unstructured grids. We developed the various implementations that have been made in `Wolf` during this thesis in order to be able to deal with various flows and methodologies, such as for periodic turbomachinery applications. The RANS computations being noticeably more expensive than the inviscid ones, we developed and tested different modifications in the linear system resolution and residual computations to fasten the simulations. This makes it possible and affordable to perform mesh adaptation on such cases, even though this study shown that standard SGS approach was already efficient for the linear system inversion. A significant effort has been dedicated to the improvement of the robustness of `Wolf` for RANS computations, reducing and carefully controlling the approximations that have been made in the solver. Hence `Wolf` can perform high Reynolds number simulations on fully unstructured meshes, as we will see in the second Part of this thesis.

Beside the main solver, we also developed several tools for error estimation and mesh adaptation by extending the inviscid adjoint solver to turbulent flows and taking into account the possible periodicity of the flow. We introduced a non-linear solution correction strategy to assess precisely the errors involved in the numerical simulation. This analysis will prove useful for mesh adaptation in the second Part of this thesis.

Despite some improvements, RANS computations are still more expensive than they should be. Limiters are still the limiting factor in terms of CPU time in most computations, preventing the solution from a fast convergence. We managed to improve the detection and treatment of limite cycles but failed to solve this issue in general. The adjoint computation is the second bottleneck in the current numerical simulations in a mesh adaptation context. As we mentioned, the adjoint resolution is a stiff problem that requires a strong linear solver. To this end we use a standard GMRES approach, which requires in some cases more memory and CPU time than the primal solver. We thus seek alternative solutions to have a memory and CPU efficient resolution of the adjoint. Moreover, we now start working on very large cases, requiring large amount of memory. `Wolf` has proven to be efficient so far with threads parallelization but, we reached the memory limits achievable on single machines and we would need a MPI version of `Wolf` to keep going on.

Part II

Anisotropic mesh adaptation for RANS applications

Introduction

RANS computations have become the norm in various industries, thanks to their relative simplicity and affordable cost. They have been extensively tested and are now trusted for a large panel of flows. Though, paradoxically, after three decades of intensive use, RANS computations are still a challenge. This is due in part to the meshes required to appropriately deal with the boundary layers and all potentially complex flow features involved [Larsson 2014]. This issue has worsen across the years, following the continuously increasing expectations with regard to the geometries complexity.

Indeed, meshes are generally generated following guidelines built over the years thanks to the experience accumulated on different cases [Chan 2017]. Still, their validity is regularly questioned. The AIAA CFD high-lift prediction workshop[2] [Rumsey 2018] and AIAA CFD drag prediction workshop[3] [Levy 2014] are excellent examples of the current trend in applied turbulent computations. These regroup numerous participants to state and compare the current abilities in term of CFD predictions. A common set of meshes is generated and provided to all participants to ensure a common basis. These meshes are generated in a traditional way, *eg.* following the intuition and experience of a mesh generation specialist. Still, during these workshops, numerous participants proposed several modifications to the meshes to improve the convergence and predictions. This shows that these meshes are far from being optimal.

We can cite as a simple example the generation of structured boundary layer meshes. Even though boundary layers are precisely located, their discretization is an acute problem. It is an active topic of research even today as indicated by the numerous publications and approaches proposed to deal with boundary layer mesh generation [Alauzet 2017, Alauzet 2015, Aubry 2009, Bottasso 2002, Garimella 2000, Ito 2002, Loseille 2011c, Marcum 1982, Woeber 2017]. Beyond the fact that the generation/insertion of the structured mesh is challenging, the prescription of its size is still an open question. Various mesh size progressions are regularly proposed, each one being different for each case, indicating the need for a more systematic approach. Indeed, boundary layer generation rules are generally built on flat boundary layers assumptions which are questionable for complex curved geometries and in presence of ridges.

Furthermore, the CAD preprocessing and mesh generation steps are generally the dominant bottle-neck in current days CFD computations, as they require user implication at many levels [Slotnick 2014]. With regard to the mesh generation, as we mentioned, several user prescriptions are required to define the mesh sizes. This is an impediment for automation in many processes such as shape optimization in design.

Meanwhile, anisotropic mesh adaptation, based on a well established mathematical theory, has proven to be very efficient for inviscid applications. Achieving the same performance for RANS applications is a demanding process, as it requires a robust and appropriate mesh generation and consistent error estimation sensors. In the first part of this thesis, we detailed the implementation of our RANS flow solver `Wolf` and we will show in this part that these careful developments make it capable of computing high Reynolds number flows on fully unstructured

[2] https://hiliftpw.larc.nasa.gov/
[3] https://aiaa-dpw.larc.nasa.gov/

meshes. This drastically eases the mesh generation process by removing the boundary layer constraint. The error sensor choice, and by this mean the mesh size prescription has also been a major obstacle to the spread of mesh adaptation to RANS applications. Over the years, many empirical (and discutable) approaches have been proposed to automatically detect regions of interest [Bibb 2006, Dwight 2008]. Many aim at refining zones with strong gradients or relie on scalar sensors to determine the mesh size, but do not provide anisotropy information [Bibb 2006]. This is why the hessian of the Mach number field was used in [Venditti 2003] to determine the anisotropy of the mesh in conjunction with scalar sensor. These approaches lack of a rigorous mathematical context which makes them insufficiently efficient to compete with user prescribed meshes [Park 2016].

In this part, we extend the inviscid goal oriented analysis to viscous flows, via a systematic analysis of the numerical errors involved. We show on different test cases that this approach successfully deals with anisotropy and provides a competitive mesh prescription throughout adaptation. Moreover, we show that this robust approach can flawlessly deal with complex geometries and lead to mesh convergent computations. This process being fully automatic, it requires no further user intervention.

Chapter 5

Continous mesh framework for mesh adaptation

We recall in this chapter the basis of the continuous mesh framework used to describe and optimize meshes in a mesh adaptation process. The continuous mesh is the base of modern unstructured mesh adaptation. It provides a continuous analytical description of the discrete representation of the mesh. This allows to use the full strength of mathematical tools developed for continuous analysis to analyse and optimize mesh related problems. This in turn yield a direct mathematical connection between the numerical error and the local mesh size and shape.

5.1 Continous mesh

5.1.1 A simple 1D example

Let us assume we seek the best linear discretization of a function f over the segment $[0,1]$ with a given number of nodes N. In other words, we seek N points $0 < x_1 < ... < x_N < 1$ on which we represent the discrete linear projection of the function f as

$$\Pi_h(f)(x_i) = f(x_i), \quad \forall x \in [x_i, x_{i+1}],\ \Pi_h(f)(x) = \frac{(x_{i+1}-x)f(x_i) + (x-x_i)f(x_{i+1})}{x_{i+1}-x_i},$$

so that the interpolation error in L^1-norm

$$\mathcal{E} = \int_0^1 |f(x) - \Pi_h(f)(x)| \mathrm{d}x$$

is minimal. This expression can be naturally split onto each sub-segment as

$$\mathcal{E} = \sum_i \int_{x_i}^{x_{i+1}} |f(x) - \Pi_h(f)(x)| \mathrm{d}x,$$

but even in simple cases, it is impossible to compute analytically these integrals.

Local error model: We will thus try to approach this expression at the leading order with a Taylor expansion of f and $\Pi_h(f)$ on each sub-segment as

$$f(x) \approx f\left(\frac{x_{i+1}+x_i}{2}\right) + \left(x - \frac{x_{i+1}+x_i}{2}\right) f'\left(\frac{x_{i+1}+x_i}{2}\right) + \frac{\left(x - \frac{x_{i+1}+x_i}{2}\right)^2}{2} f''\left(\frac{x_{i+1}+x_i}{2}\right) + ...$$

Doing so, the constant and linear terms cancels and we are left with the following approximation

$$\int_{x_i}^{x_{i+1}} |f(x) - \Pi_h(f)(x)| \mathrm{d}x \approx \frac{|x_{i+1}-x_i|^3}{12} \left| f''\left(\frac{x_{i+1}+x_i}{2}\right) \right| + ...$$

Basically, this consists in approaching f on each sub-segment with its quadratic reconstruction.

The expression of $\mathcal{E}$ is greatly simplified as

$$\mathcal{E} \approx \sum_i \frac{|x_{i+1}-x_i|^3}{12}\left|f''\left(\frac{x_{i+1}+x_i}{2}\right)\right|,$$

but it is still quite difficult to optimize. First because it is not easy to express the constraint $0 < x_1 < ... < x_N < 1$, and then because the derivatives of $\mathcal{E}$ with respect to each x_i involves multiple terms including third derivatives of f.

Reshaping the problem: To further simplify this expression we will solve it with a fixed point approach. Starting from an initial set of nodes, we assume that $f''\left(\frac{x_{i+1}+x_i}{2}\right)$ is constant, so that the derivatives of $\mathcal{E}$ only involves $(x_{i+1}-x_i)^3$. We then replace the optimization problem on the set of x_i by an optimization problem on the length $l_i = x_{i+1} - x_i$ of each segment so that the problem expresses

$$\mathcal{E} \approx \sum_i \frac{(l_i)^3}{12}\Bigg|\underbrace{f''\left(\frac{x_{i+1}+x_i}{2}\right)}_{\text{Independent of } l_i}\Bigg| \qquad \text{so that } \sum_i l_i = 1.$$

Discrete minimization: To find the minima of this expression we use the Lagrangian

$$\mathcal{L}(l_0, ..., l_N, \lambda) = \sum_i \frac{(l_i)^3}{12}\left|f''\left(\frac{x_{i+1}+x_i}{2}\right)\right| - \lambda\left(\sum_i l_i - 1\right).$$

Finding the minimum of $\mathcal{E}$ under the constraint $\sum_i l_i = 1$ is equivalent to finding the minimum of $\mathcal{L}$ (Euler theorem). We thus compute the derivative of $\mathcal{L}$ with respect to each l_i

$$\frac{\partial \mathcal{L}}{\partial l_i} = \frac{l_i^2}{4}\left|f''\left(\frac{x_{i+1}+x_i}{2}\right)\right| - \lambda = 0,$$

and deduce the values of

$$l_i = \frac{2\sqrt{\lambda}}{\sqrt{\left|f''\left(\frac{x_{i+1}+x_i}{2}\right)\right|}}.$$

Knowing that $\sum l_i = 1$, we obtain

$$1 = \sum_i l_i = 2\sqrt{\lambda}\sum_i \frac{1}{\sqrt{\left|f''\left(\frac{x_{i+1}+x_i}{2}\right)\right|}},$$

so that

$$\lambda = \frac{1}{4\left(\sum_i \left(\left|f''\left(\frac{x_{i+1}+x_i}{2}\right)\right|\right)^{-1/2}\right)^2}.$$

Finally,

$$l_i = \left(\sqrt{\left| f''\left(\frac{x_{i+1}+x_i}{2} \right) \right|} \left(\sum_j \left(\left| f''\left(\frac{x_{j+1}+x_j}{2} \right) \right| \right)^{-1/2} \right) \right)^{-1}.$$

From these expressions we then generate a new set of x_i. In this new state, the $f''\left(\frac{x_{j+1}+x_j}{2}\right)$ have been changed and we restart the process from the new configuration. This goes on until the fixed point process converges which is achieved when

$$x_{i+1} - x_i = \left(\sqrt{\left| f''\left(\frac{x_{i+1}+x_i}{2} \right) \right|} \left(\sum_j \left(\left| f''\left(\frac{x_{j+1}+x_j}{2} \right) \right| \right)^{-1/2} \right) \right)^{-1}.$$

Continuous expression: The local error model that we assumed is asymptotically valid when the element size tends to 0. Then, it is tempting to consider no more the l_i distribution as a discrete set but rather as a function of x. Namely, we can represent $l(x)$ as being constant equal to l_i on each sub-segment of length l_i (l does not need to be continuous).

As the element size tend to 0, it is natural to replace the sums by integrals as

$$\frac{(x_{i+1}-x_i)^3}{12} \left| f''\left(\frac{x_{i+1}+x_i}{2} \right) \right| \approx \int_{x_i}^{x_{i+1}} \frac{(l(x))^2}{12} |f''(x)| \mathrm{d}x,$$

so that the error becomes asymptotically

$$\mathcal{E} \approx \int_0^1 \frac{(l(x))^2}{12} |f''(x)| \mathrm{d}x.$$

The constraint is slightly more complex to express, as the same treatment leads to

$$1 = \sum_{i=1}^{N} l_i = \sum_{i=1}^{N} \int_{x_i}^{x_{i+1}} 1 \mathrm{d}x = \int_0^1 1 \mathrm{d}x,$$

which does not give any information. But we can use a similar trick to count the elements as

$$N = \sum_{i=1}^{N} 1 = \sum_{i=1}^{N} \int_{x_i}^{x_{i+1}} \frac{1}{l_i} \mathrm{d}x \approx \int_0^1 \frac{1}{l} \mathrm{d}x.$$

Hence, we deduce the continuous Lagrangian of the problem

$$\mathcal{L}(l, \lambda) = \int_0^1 \frac{l^2(x)|f''(x)|}{12} + \lambda \left(\frac{1}{l(x)} - N \right) \mathrm{d}x.$$

This expression can be differentiated with respect to l and the optimality expresses

$$\forall \delta l, \qquad \int_0^1 \left(\frac{2l(x)|f''(x)|}{12} - \frac{\lambda}{l^2(x)} \right) \delta l(x) \mathrm{d}x,$$

so that

$$\frac{2l(x)|f''(x)|}{12} - \frac{\lambda}{l^2(x)} = 0.$$

As previously, we deduce the expression of $l(x)$

$$l(x) = \frac{(6\lambda)^{1/3}}{|f''(x)|^{1/3}},$$

and its normalisation

$$l(x) = \frac{\int_0^1 |f''(s)|^{1/3} \mathrm{d}s}{N} \frac{1}{|f''(x)|^{1/3}}.$$

It is particularly confusing at this point to note that we do not recover a similar expression as with the discrete approach. More importantly, both expressions are not even of the same order with respect to the second derivative f'', the discrete approach yields a scaling proportional to $|f''|^{-1/2}$ while the continuous approach yields a scaling proportional to $|f''|^{-1/3}$! We can clear this paradox by looking at a simple case.

Let us now consider that the second derivative f'' is given by $f'' = f_1$ if $0 \leq x < 1/2$ and $f'' = f_2$ if $1/2 \leq x \leq 1$. We can directly provide de l distribution with the continuous approach as

$$l(x) = \begin{cases} \dfrac{1}{2N} \dfrac{|f_1|^{-1/3} + |f_2|^{-1/3}}{|f_1|^{-1/3}} & \text{if } 0 \leq x < 1/2 \\ \dfrac{1}{2N} \dfrac{|f_1|^{-1/3} + |f_2|^{-1/3}}{|f_2|^{-1/3}} & \text{if } 1/2 \leq x \leq 1 \end{cases}.$$

But as we mentioned, the discrete approach requires the convergence of the fixed point process.

We can solve it here with an other optimization, noting that the domain is split in two, so that we can consider that there are N_1 elements between 0 and $1/2$ and N_2 elements between $1/2$ and 1. As f'' is constant on each half domain, so is l_i, so that

$$l_1 = \frac{1}{2N_1} \qquad \text{and} \qquad l_2 = \frac{1}{2N_2}.$$

From the previous development we get

$$l_i = \left(\sqrt{|f_i|} \left(N_1 |f_1|^{-1/2} + N_2 |f_2|^{-1/2} \right) \right)^{-1},$$

where we do not know the balance between N_1 and N_2.

It is here enlightening to rewrite the whole error analysis as

$$\mathcal{E} = N_1 \frac{l_1^3}{12} |f_1| + N_2 \frac{l_2^3}{12} |f_2| = \frac{l_1^2}{24} |f_1| + \frac{l_2^2}{24} |f_2|.$$

We can see that the individual error of each element is somehow weighted by the number of elements of the considered size, so that we recover the same expression as for the continuous approach with the discrete Lagrangian

$$\mathcal{L}(l_1, l_2, \lambda) = \frac{l_1^2}{24} |f_1| + \frac{l_2^2}{24} |f_2| + \lambda \left(\frac{1}{l_1} + \frac{1}{l_2} - N \right).$$

In practice, the fixed point will progressively balance the elements in the domain, until the discrete expression of the size reaches the continuous value

$$l_i = \left(\sqrt{|f_i|} \left(N_1 |f_1|^{-1/2} + N_2 |f_2|^{-1/2} \right) \right)^{-1} = \frac{1}{2N} \frac{|f_1|^{-1/3} + |f_2|^{-1/3}}{|f_i|^{-1/3}}.$$

This shows that the continuous approach is nonetheless more flexible but also already includes a global normalization that needs to be converged in the discrete process. The continuous expression

$$l(x) = \frac{\int_0^1 |f''(s)|^{1/3} \mathrm{d}s}{N} \frac{1}{|f''(x)|^{1/3}},$$

is also valid for any number of elements as it only implies to change the value of N in the expression, scaling the length.

Metric space and continuous mesh: We assumed in the previous development that the function l was conveniently considered to be constant on each sub-segment of length l_i so that we had a clear relationship between the length of an element and the function. But in practice the expression of l is continuous and it is unclear how to generate a mesh from its values. The solution of this problem lies in the way we imposed the constraint on the number of elements in the continuous approach. We recall that we used the fact that on an element of size l_i we had $1 = \int_{x_i}^{x_{i+1}} \frac{1}{l_i} \mathrm{d}x$, which can be directly extended to the continuous case. Hence, for a continuous distribution of size $l(x)$, as we have $\int_0^1 1/l(x) \mathrm{d}x = N$, we can define unambiguously that each element verifies

$$\int_{x_i}^{x_{i+1}} 1/l(x) \mathrm{d}x = 1.$$

This actually defines a metric space where the length of each segment $[a, b]$ is measured as

$$d(a, b) = \int_a^b d(x) \mathrm{d}x,$$

where $d(x) = 1/l(x)$ is the density. Our optimal mesh generated from the distribution of $l(x)$ is unit in this metric space. As we have an equivalence in 1D between this mesh and the $d(x)$ distribution, d is called continuous mesh.

5.2 Extension to higher dimensions

In one dimension the description of a mesh is rather simple as it consists only in a collection of points. In higher dimensions beside the position of the nodes, we need to add the connectivity of the different elements in a way that guaranties that the mesh is valid (conform, elements of positive volumes, ...). In the previous section, although the continuous expression was simpler to handle, we were able to derive a discrete expression of the error and optimal mesh. Here the problem is semi-discrete, we would have to optimize simultaneously the position of the nodes and their connectivity, which would be extremely difficult and time consuming. Instead, we will relie on the same continuous mesh description.

5.2.1 Euclidian and Riemannian metric spaces

Euclidian space

We consider the vector space $\mathbb{R}^n$, typically $n = 2$ or 3 in our case, with its canonical basis. A **scalar product** is a Symmetric Positive Definite (SPD) form. This form can be represented in

the canonical basis by a SPD matrix $\mathcal{M} = (m_{ij})_{1 \leq i,j \leq n}$ so that the scalar product is written:

$$\begin{array}{rccl} (\cdot\,,\cdot)_{\mathcal{M}}: & \mathbb{R}^n \times \mathbb{R}^n & \longrightarrow & \mathbb{R}^+ \\ & (\mathbf{u},\mathbf{v}) & \longmapsto & (\mathbf{u},\,\mathbf{v})_{\mathcal{M}} \;=\; \mathbf{u}^T\,\mathcal{M}\,\mathbf{v} \;=\; \displaystyle\sum_{j=1}^{n}\sum_{i=1}^{n} m_{ij}u_iv_j\,. \end{array} \tag{5.1}$$

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

From the scalar product we define the underlying Euclidian norm

$$\begin{array}{rccl} ||\cdot||_{\mathcal{M}}: & \mathbb{R}^n & \longrightarrow & \mathbb{R}^+ \\ & \mathbf{u} & \longmapsto & ||\mathbf{u}||_{\mathcal{M}} = \sqrt{\mathbf{u}^T\,\mathcal{M}\,\mathbf{u}}\,. \end{array} \tag{5.2}$$

and lengths in the Euclidian space:

$$\begin{array}{rccl} d_{\mathcal{M}}: & \Omega \times \Omega & \longrightarrow & \mathbb{R}^+ \\ & (P,\,Q) & \longmapsto & d_{\mathcal{M}}\,(P,\,Q) = \sqrt{\mathbf{PQ}^T\,\mathcal{M}\,\mathbf{PQ}}\,, \end{array} \tag{5.3}$$

This distance induces a metric space structure on the vector space, so that $\mathcal{M}$ is called a *metric tensor* or simply *metric*.

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

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

$$\ell_{\mathcal{M}}\,(\mathbf{e}) = \sqrt{\mathbf{e}^T\,\mathcal{M}\,\mathbf{e}}\,, \tag{5.4}$$

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

$$\cos\theta = \frac{(\mathbf{v_1},\,\mathbf{v_2})_{\mathcal{M}}}{||\mathbf{v_1}||_{\mathcal{M}}\,||\mathbf{v_2}||_{\mathcal{M}}} \tag{5.5}$$

- the volume of element K is:

$$|K|_{\mathcal{M}} = \sqrt{\det\mathcal{M}}\,|K|_{\mathcal{I}}\,. \tag{5.6}$$

As $\mathcal{M}$ is SPD, it is **diagonalizable in an orthonormal basis**:

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

$$\text{where} \begin{cases} \Lambda \;= \text{diag}\,(\lambda_1,\,\ldots,\,\lambda_n) \text{ is the diagonal matrix made of the eigenvalues of } \mathcal{M}\,, \\ \mathcal{R} \;= (\,\mathbf{r_1}\,|\,\mathbf{r_2}\,|\,\ldots\,|\mathbf{r_n}\,)^T \text{ is the unitary matrix (\textit{i.e.} } \mathcal{R}^T\mathcal{R} = \mathcal{I}) \\ \quad\;\; \text{made of the eigenvectors of } \mathcal{M}\,. \end{cases} \tag{5.7}$$

Basically, an Euclidian metric, compared to the canonical metric, changes the way of measuring length depending on the direction in which the edge/vector/segment considered is oriented.

An intuitive and useful **geometric representation** of a metric tensor is thus its unit ball, namely the set vectors of unit length in this metric. Given a vector $\mathbf{u} = (X, Y, Z)$, its length in the metric space writes

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

so that the set of unit vectors verify

$$\mathcal{M}_{xx}X^2 + \mathcal{M}_{yy}Y^2 + \mathcal{M}_{zz}Z^2 + 2\mathcal{M}_{xy}XY + 2\mathcal{M}_{xz}XZ + 2\mathcal{M}_{yz}YZ - 1 = 0.$$

This defines a conic, and as $\mathcal{M}$ is SPD, it is an ellipse in 2D and an ellipsoid in 3D as shown in Figure 5.1.

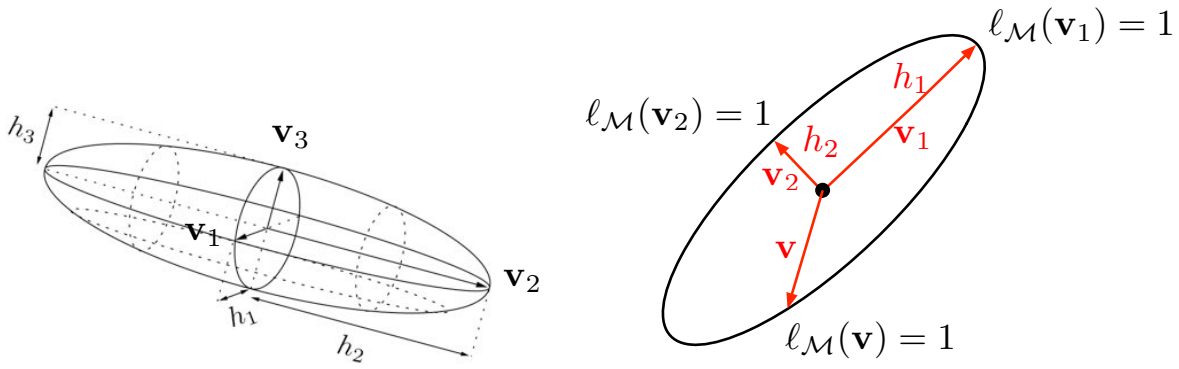


Figure 5.1: Unit balls associated with metric $\mathcal{M} = \mathcal{R}\,\Lambda\,\mathcal{R}^T$ in two and three dimensions.

A metric tensor $\mathcal{M}$ provides another useful information: the application that maps the unit ball associated with $\mathbb{I}$ to the unit ball associated with $\mathcal{M}$. The matrix of this transformation in canonical basis $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e_n})$ is:

$$\mathcal{M}^{-\frac{1}{2}} = \mathcal{R}\,\Lambda^{-\frac{1}{2}}\,\mathcal{R}^T, \quad \text{where } \Lambda^{-\frac{1}{2}} = \text{diag}\left(h_1 = \lambda_1^{-\frac{1}{2}}, \ldots, h_n = \lambda_n^{-\frac{1}{2}}\right). \tag{5.8}$$

This **natural mapping** corresponds to the change of basis from the canonical basis to the orthonormal diagonalization basis of $\mathcal{M}$ and is depicted in Figure 5.2.

Riemannian metric space

The scalar product of Euclidian spaces is the same on the whole space, which means that the distance definition is the same for each point of the space (the unit ball is the same everywhere). We now consider a set of SPD tensors $\mathbf{M} = (\mathcal{M}(P))_{P\in\Omega}$, also called metric tensor field, defined on the whole domain $\Omega \subset \mathbb{R}^n$. Locally at point P, $\mathcal{M}(P)$ induces a scalar product on $\mathbb{R}^n \times \mathbb{R}^n$. The vector space, with this new structure, is called a **Riemannian metric space**. In this thesis, we will use the same notation $\mathcal{M}$ to speak of the metric field and of the metric tensor at a given point. Notation $\mathbf{M}$ will only be used if the distinction is necessary for pedagogical purposes.

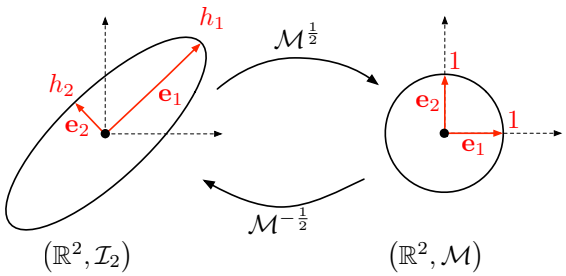


Figure 5.2: Natural mapping $\mathcal{M}^{-\frac{1}{2}}$ associated with metric $\mathcal{M}$ in two dimensions. It sends the unit ball of $\mathbb{I}_2$ onto the unit ball of $\mathcal{M}$.

Remark 1. *Unlike usual Riemannian spaces, there is no notion of manifold in Riemannian metric spaces. However, Riemannian metric spaces can be assimilated to functions representing Cartesian surfaces, and the metric tensor defined for a point of the space is a scalar product on the tangent plane for that point. Figure 5.3 gives an example of a Cartesian surface associated with a Riemannian metric space. This Riemannian metric space is pictured by drawing the unit ball of the metric at some points of the domain. A link with differential geometry is proposed in [Olivier 2011].*

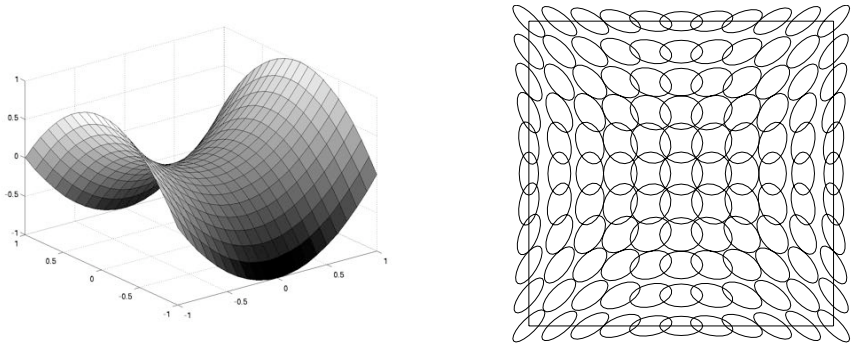


Figure 5.3: Left, example of a Cartesian surface embedded in $\mathbb{R}^3$. Right, geometric visualization of a Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in[0,1]\times[0,1]}$ associated with this surface. At some points $\mathbf{x}$ of the domain, the unit ball of $\mathcal{M}(\mathbf{x})$ is drawn.

We can extend the notions of length, angle and volume from the Euclidian space case. The following **geometrical quantities** are defined:

- the length of any path Γ parametrized by $\gamma : t \in [0,1] \longmapsto \gamma(t)$ is defined as

$$\ell_{\mathcal{M}}(\Gamma) = \int_0^1 ||\gamma'(t)||_{\mathcal{M}(\gamma(t))}\, \mathrm{d}t.$$

In the case of an edge $\mathbf{e} = \mathbf{PQ}$, the parametrization expresses $\gamma : t \in [0,1] \longmapsto P + t\,\mathbf{PQ}$

and we have:

$$\ell_{\mathcal{M}}(\mathbf{e}) = \int_0^1 \sqrt{\mathbf{PQ}^T \mathcal{M}(P + t\,\mathbf{PQ})\,\mathbf{PQ}}\,\mathrm{d}t\,, \tag{5.9}$$

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

$$\cos\theta = \frac{(\mathbf{v_1}, \mathbf{v}_2)_{\mathcal{M}(P)}}{||\mathbf{v}_1||_{\mathcal{M}(P)}\,||\mathbf{v}_2||_{\mathcal{M}(P)}}\,, \tag{5.10}$$

- assuming the metric field $\mathcal{M}$ to be continuous, the volume of an elementary volume $\mathrm{d}K$ expresses as the limit to the Euclidian case,

$$|\mathrm{d}K|_{\mathcal{M}} = |\mathrm{d}K|_{\mathbb{I}}\,\sqrt{\det \mathcal{M}(\mathbf{x})},$$

so that the measure of any volume expresses as

$$|\Omega|_{\mathcal{M}} = \int_\Omega \sqrt{\det \mathcal{M}(\mathbf{x})}\,\mathrm{d}\Omega. \tag{5.11}$$

The volume of an element K should be computed with this expression but we will approximate it at first order in the asymptotic regime:

$$|K|_{\mathcal{M}} \approx |K|_{\mathbb{I}}\,\sqrt{\det \mathcal{M}(G_K)}, \quad \text{where } G_K \text{ is the barycenter of } K. \tag{5.12}$$

5.2.2 Unit mesh

Metric fields, as defined in the previous section, enables to modify locally geometric quantities such as lengths and volumes as well as to provide a local orientation (prescribing two privileged axes). They are thus a good way, both elegant and mathematically efficient, to prescribe sizes and orientations for the elements of an anisotropic adapted mesh. The main idea of metric-based mesh adaptation, introduced for the first time in [George 1991], is to use directly a Riemannian metric space within the mesher to compute the necessary geometrical quantities, and to generate a *unit mesh* with respect to this Riemannian metric space.

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

$$\forall i = 1, ..., 6, \;\; \ell_{\mathcal{M}}(\mathbf{e_i}) = 1. \tag{5.13}$$

If K is composed only of unit length edges in an Euclidian metric space, then it can be mapped with Equation (5.8) into a unit regular tetrahedra, thus its volume $|K|_{\mathcal{M}}$ in $\mathcal{M}$ is constant equal to:

$$|K|_{\mathcal{M}} = \frac{\sqrt{2}}{12} \;\text{ and }\; |K| = \frac{\sqrt{2}}{12}(\det(\mathcal{M}))^{-\frac{1}{2}}, \tag{5.14}$$

where $|K|$ is its Euclidean volume.

Although this definition is very simple, things are more complicated when it comes to defining a **unit mesh**. *Stricto sensu*, a unit mesh is mesh whose edges are all unit with respect

to the prescribed metric field. However, the existence of a mesh composed of unit elements is not assured. Even in the simplest case of $\mathcal{M}(P) = \mathbb{I}(P)$ for each point P, *i.e.* the canonical Euclidean space, it is well known that $\mathbb{R}^3$ cannot be filled with regular tetrahedra (that are unit with respect to the identity metric). So, the constraint on the sizes of the edges has to be relaxed. We introduce the notion of **quasi unit element**. A tetrahedron is said to be quasi unit with respect to a metric field $\mathcal{M}$ if its edges are close to unit, *i.e.* $\forall i,\ \ell_{\mathcal{M}}(\mathbf{e_i}) \in [\frac{1}{\sqrt{2}}, \sqrt{2}]$.

Additionally, in order to avoid elements with a null volume (see [Loseille 2011a]), we have to add a constraint on the volume, which is achieved through a quality function:

$$Q_{\mathcal{M}}(K) = \frac{\sqrt{3}}{216} \frac{\left(\sum_{i=1}^{6} \ell^2_{\mathcal{M}}(\mathbf{e_i})\right)^{\frac{3}{2}}}{|K|_{\mathcal{M}}} \in [1, +\infty]\,. \tag{5.15}$$

For a regular tetrahedron in the metric, the quality function is equal to 1, whereas it tends to $+\infty$ for a null volume tetrahedron. Hence, an element close to a perfectly unit element has a quality close to 1. This leads to the following definition. A tetrahedron K defined by its list of edges $(\mathbf{e_i})_{i=1\ldots6}$ is said to be **quasi-unit** for Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ if

$$\forall i \in [1,6], \quad \ell_{\mathcal{M}}(\mathbf{e_i}) \in \left[\frac{1}{\sqrt{2}}, \sqrt{2}\right] \quad \text{and} \quad Q_{\mathcal{M}}(K) \in [1, \alpha] \ \text{ with } \ \alpha > 1\,, \tag{5.16}$$

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

In practice, it is this definition that is used in meshing software. For any kind of desired mesh (uniform, adapted isotropic, adapted anisotropic), the mesh generator will generate a mesh that is unit with respect to the prescribed metric space. The resulting mesh is uniform in the metric space while it is adapted in the canonical Euclidian space. This is illustrated in Figure 5.4.

5.2.3 Operations on metrics

The main advantage of using metrics is the existence of well-posed operations on metrics, that have a direct geometric interpretation with unit ellipsoids. The most useful ones are metric intersection and metric interpolation.

Metric intersection

The **metric intersection** consists in keeping the most restrictive size constraint in all directions imposed by a set of metrics. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two metric tensors given at a point. The intersection of $\mathcal{M}_1$ and $\mathcal{M}_2$ is the metric tensor $\mathcal{M}_{1\cap2}$ that prescribes 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 $\mathcal{M}_2$. From a geometric point of view, the intersection between two metrics is not the intersection between two ellipsoids as their geometric intersection is not an ellipsoid, but it is the largest ellipsoid representing $\mathcal{M}_{1\cap2}$ included in the geometric intersection of the ellipsoids associated with $\mathcal{M}_1$ and $\mathcal{M}_2$, cf. Figure 5.5-(left).

The ellipsoid (metric) verifying this property is obtained by using simultaneous reduction of the quadratic forms associated with the two metrics. Simultaneous reduction theory for two

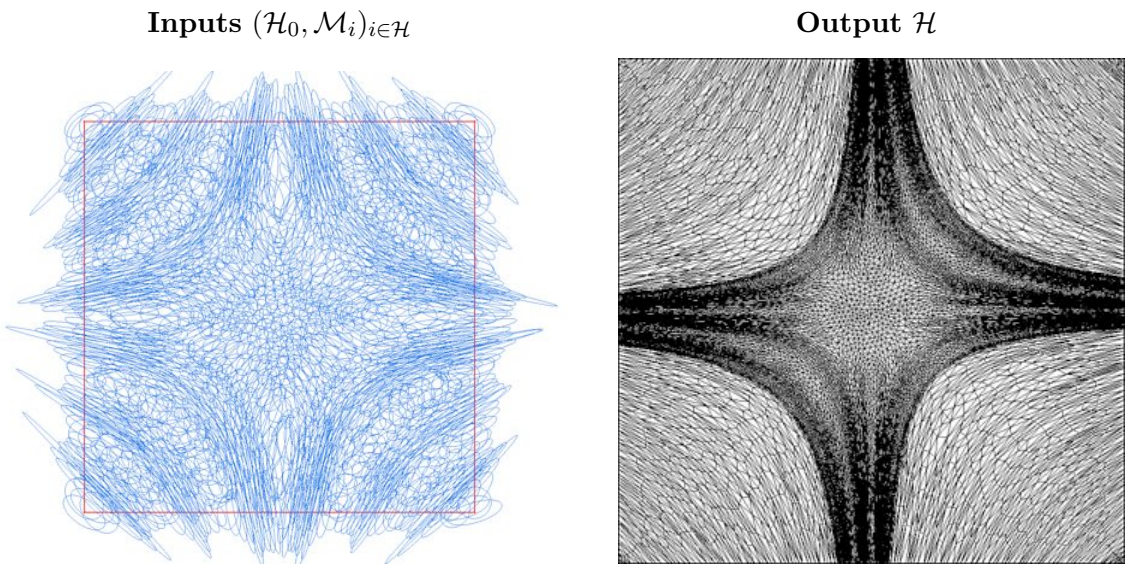


Figure 5.4: Metric-based mesh generation. Left, specified Riemannian metric space. Right, unit mesh in the prescribed Riemannian metric space which becomes adapted anisotropic in the Euclidean space.

quadratic forms, one of which being positive definite, tells that there exists a common basis that is orthonormal for the positive definite form and orthogonal for the other. We place ourselves in this basis in two steps.

First, let us consider $\mathcal{M}_1^{-1/2}$ (which exists because $\mathcal{M}_1$ is positive definite):

$$\mathcal{M}_1 \;=\; \mathcal{P}\mathrm{diag}(\lambda_1,\lambda_2,\lambda_3)\mathcal{P}^T \;\Rightarrow\; \mathcal{M}_1^{-1/2} \;=\; \mathcal{P}\mathrm{diag}(1/\sqrt{\lambda_1},1/\sqrt{\lambda_2},1/\sqrt{\lambda_3})\mathcal{P}^T.$$

This matrix represents a mapping that sends metric $\mathcal{M}_1$ onto the identity. Hence, the rows of $\mathcal{M}_1^{-1/2}$ form an orthogonal basis that is orthonormal for the scalar product induced by $\mathcal{M}_1$.

Let :

$$\begin{aligned}\overline{\mathcal{M}_1} &= {\mathcal{M}_1^{-1/2}}^T\mathcal{M}_1\mathcal{M}_1^{-1/2} \;=\; \mathbb{I},\\ \overline{\mathcal{M}_2} &= {\mathcal{M}_1^{-1/2}}^T\mathcal{M}_2\mathcal{M}_1^{-1/2}.\end{aligned}$$

As $\overline{\mathcal{M}_2}$ is a real, symmetric matrix, it is diagonalizable in an orthonormal basis. Noting $\mathcal{P}$ the orthonormal matrix whose columns are the eigenvectors:

$$\begin{aligned}\overline{\mathcal{M}_2} &= \mathcal{P}\begin{pmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{pmatrix}\mathcal{P}^T,\\ \overline{\mathcal{M}_1} &= \mathcal{P}\,\mathbb{I}\mathcal{P}^T = \mathbb{I}.\end{aligned}$$

The two metrics are then intersected in this latter basis:

$$\overline{\mathcal{M}_{1\cap 2}} = \mathcal{P}\begin{pmatrix}\max(\lambda_1,1) & 0 & 0\\ 0 & \max(\lambda_2,1) & 0\\ 0 & 0 & \max(\lambda_3,1)\end{pmatrix}\mathcal{P}^T.$$

And finally:

$$\mathcal{M}_{1\cap 2} \;=\; \mathcal{M}_1^{1/2^T}\,\overline{\mathcal{M}_{1\cap 2}}\,\mathcal{M}_1^{1/2}\,. \tag{5.17}$$

Remark 2. *$\mathcal{M}_1$ and $\mathcal{M}_2$ are in general* **not** *codiagonalizable! We did not performed here a change of basis that would express $\overline{\mathcal{M}} = \mathcal{P}\mathcal{M}\mathcal{P}^{-1}$. The difference lies in the fact that $\mathcal{M}_1^{-1/2}$ is not an orthonormal matrix, and $\left(\mathcal{M}_1^{-1/2}\right)^T \neq \left(\mathcal{M}_1^{-1/2}\right)^{-1}$. In particular, although $\mathcal{M}_1$ is scaled to the identity matrix by the first transformation, its expression in this basis is* **not** *the identity matrix:*

$$\mathcal{M}_1^{-1/2^T}\mathcal{M}_1\mathcal{M}_1^{-1/2} = \mathcal{I} \quad \textit{but} \quad \mathcal{M}_1^{1/2}\mathcal{M}_1\mathcal{M}_1^{-1/2} \neq \mathcal{I}.$$

Hence, neither $\mathcal{M}_1$ nor $\mathcal{M}_2$ is diagonal in the basis that we found.

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 can be used after solving an optimization problem [Loseille 2008].*

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 with Relation (5.11) is generally performed using quadrature points to approximate the integral. It thus requires the computation of some interpolated metrics inside the considered element. Figure 5.5 illustrates metric interpolation along a segment, for which the initial data are the endpoints metrics.

Several interpolation schemes have been proposed in [Alauzet 2003b, Alauzet 2003a] which are based on the simultaneous reduction. The main drawback of these approaches is that the interpolation operation is not commutative, and the result depends on the order in which the operations are performed when more than two metrics are involved. Moreover, some useful properties such as the maximum principle are not verified by these schemes. A consistent operational framework, the log-Euclidian framework, was introduced in [Arsigny 2006], which we use to design an interpolation scheme.

We first define the notion of metric logarithm and exponential:

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

where $\ln(\Lambda) = \mathrm{diag}(\ln(\lambda_i))$ and $\exp(\Lambda) = \mathrm{diag}(\exp(\lambda_i))$. 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 (as $\mathcal{M}_1$ and $\mathcal{M}_2$ commute, they are

diagonalizable in the same basis). The space of metric tensors, supplied with the logarithmic addition $\oplus$ and the logarithmic scalar multiplication $\odot$ is a vector space.

From the log-Euclidean framework, we define the following linear interpolation operator. 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:

$$\mathcal{M}(\mathbf{x}) = \bigoplus_{i=1}^{k} \alpha_i \odot \mathcal{M}(\mathbf{x}_i) = \exp\left(\sum_{i=1}^{k} \alpha_i \, \ln(\mathcal{M}(\mathbf{x}_i))\right). \tag{5.18}$$

This interpolation is commutative. Moreover, it has been demonstrated in [Arsigny 2006] that this interpolation preserves the maximum principle, *i.e.,* for an edge $\mathbf{pq}$ with endpoints metrics $\mathcal{M}(\mathbf{p})$ and $\mathcal{M}(\mathbf{q})$ such that $\det(\mathcal{M}(\mathbf{p})) < \det(\mathcal{M}(\mathbf{q}))$ then we have $\det(\mathcal{M}(\mathbf{p})) < \det(\mathcal{M}(\mathbf{p} + t\,\mathbf{pq})) < \det(\mathcal{M}(\mathbf{q}))$ for all $t \in [0, 1]$.

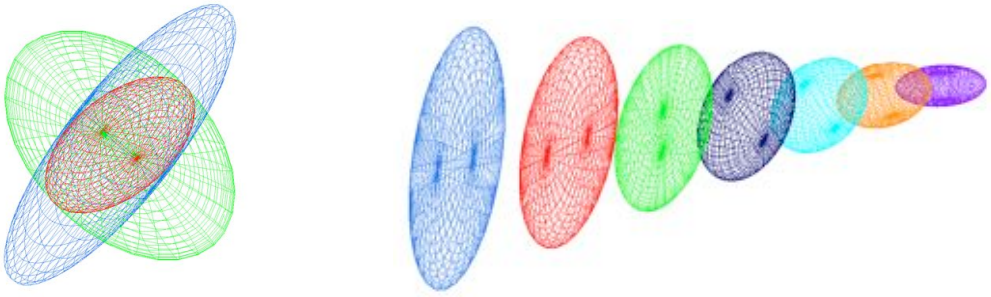


Figure 5.5: *Left, view illustrating the metric intersection procedure with the simultaneous reduction in 3D. In red, the resulting metric of the intersection of the blue and green metrics. Right, metric interpolation along a segment where the endpoints metrics are the blue and violet ones.*

5.3 The continuous mesh framework

In [Loseille 2008, Loseille 2011a, Loseille 2011b] the concept of **continuous mesh** was introduced, that has become central in our way to consider metric-based mesh adaptation. We have seen that Riemannian metric spaces are a useful tool to generate adapted meshes, but the continuous mesh framework brings them to another stage, and establishes a duality between the discrete domain and the continuous domain which is based on Riemannian metric fields. This duality enables a practical mathematical representation of adapted meshes, and allows us to use powerful tool from calculus of variations. This is particularly useful to derive appropriate error estimates, and find meshes that are optimal with respect to the error model.

The results for this section are presented in 3D, although they are easily extended to nD.

5.3.1 Duality discrete-continous: a new formalism

The key notion of unit element, defined in section 5.2.2, is once again fundamental to draw a correspondence between the discrete and the continuous domain. It is actually used to define classes of equivalence of discrete elements: let $\mathcal{M}$ be a metric tensor, there exists a non-empty infinite set of unit elements with respect to $\mathcal{M}$. Conversely, given an element K such that $|K| \neq 0$, then there is a unique metric tensor $\mathcal{M}$ for which element K is unit with respect to $\mathcal{M}$.

Consequently, a discrete element can be viewed as a discrete representative of an equivalence class formed by all the unit elements of a metric $\mathcal{M}$. Figure 5.6 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**.

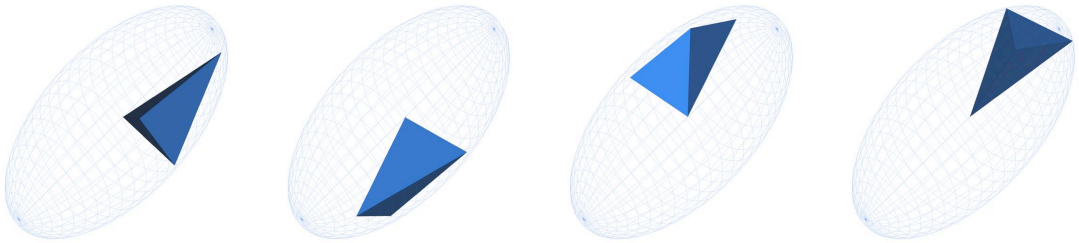


Figure 5.6: Several unit elements with respect to a metric tensor in 3D.

All the discrete representatives of a continuous element $\mathcal{M}$ share some common properties, called invariants, which justify the use of this equivalence relation. They connect the geometric properties of the discrete elements to the algebraic properties of metric $\mathcal{M}$. The two main invariants are:

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

$$\forall\,(\mathbf{e}_i, \mathbf{e}_j), \qquad \mathbf{e}_i^T\,\mathcal{M}\,\mathbf{e}_i = 1, \tag{5.19}$$

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

$$|K| = \frac{\sqrt{3}}{4}\,\det(\mathcal{M}^{-\frac{1}{2}}) \text{ in 2D} \quad \text{and} \quad |K| = \frac{\sqrt{2}}{12}\,\det(\mathcal{M}^{-\frac{1}{2}}) \text{ in 3D.} \tag{5.20}$$

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

$$\forall \mathbf{x} \in \Omega, \quad \mathcal{M}(\mathbf{x}) = d^{\frac{2}{3}}(\mathbf{x})\,\mathcal{R}(\mathbf{x}) \begin{pmatrix} r_1^{-\frac{2}{3}}(\mathbf{x}) & & \\ & r_2^{-\frac{2}{3}}(\mathbf{x}) & \\ & & r_3^{-\frac{2}{3}}(\mathbf{x}) \end{pmatrix} \mathcal{R}^T(\mathbf{x}), \tag{5.21}$$

where

- h_i the characteristic length are deduced from the eigen values $h_i = 1/\sqrt{\lambda_i}$
- density d is equal to: $d = (\lambda_1\lambda_2\lambda_3)^{\frac{1}{2}} = (h_1h_2h_3)^{-1}$, with λ_i the eigenvalues of $\mathcal{M}$
- anisotropic quotients r_i are equal to: $r_i = h_i^3\,(h_1h_2h_3)^{-1}$
- $\mathcal{R}$ is the eigenvectors matrix of $\mathcal{M}$ representing the orientation (of unit size).

The density d controls only the local level of accuracy of $\mathbf{M}$: increasing or decreasing it changes the scaling of the elements but not the anisotropic properties or the orientation, while the anisotropy is given by the anisotropic quotients and the orientation by matrix $\mathcal{R}$. By construction, we have

$$\det(\mathcal{M}^{\frac{1}{2}}) = d,$$

so that $d(\mathbf{x})$ plays the same role as in Section 5.1.1. So, we can here again count the number of elements using their volume

$$1 = \int_K \frac{1}{|K|} = \int_K \frac{12}{\sqrt{2}}\,\det(\mathcal{M}^{\frac{1}{2}}) = \frac{12}{\sqrt{2}}\int_K d(\mathbf{x})$$

and define the notion of complexity $\mathcal{C}$ of $\mathbf{M}$:

$$\mathcal{C}(\mathbf{M}) = \int_\Omega d(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_\Omega \sqrt{\det(\mathcal{M}(\mathbf{x}))}\,\mathrm{d}\mathbf{x}. \tag{5.22}$$

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

From this formulation of a Riemannian metric field arises a duality between meshes and Riemannian metric spaces. This duality is justified by the strict analogy between the following discrete and continuous notions: orientation vs. $\mathcal{R}$, stretching vs. r_i, size vs. d and number of elements vs. $\mathcal{C}(\mathbf{M})$. However, the class of discrete meshes represented by $\mathbf{M}$ is complex to describe. Indeed, we have seen that a unit mesh cannot be defined as a set of strictly unit elements, but the notion of quasi-unit elements has to be invoked once again. The following definition is adopted: in the continuous mesh framework, a **continuous mesh** of a domain Ω is defined by a set of continuous elements $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$, *i.e.,* a Riemannian metric space. It represents all the meshes that are unit for $\mathbf{M}$.

5.3.2 Continuous linear interpolation

The model that we have just defined is used to derive error estimates. As in Section 5.1.1, we will now seek a continuous local error model. The goal was to avoid to use upper bounds of the interpolation error to derive adapted meshes, as it is usually done in studies on the interpolation error, and to be able to compute the error for any function on any continuous mesh.

Let $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ be a continuous mesh of a domain Ω and let u be a non linear function which is assumed to be only twice continuously differentiable. We seek a well-posed definition of the continuous linear interpolation error $\|u - \pi_{\mathcal{M}}u\|_{L^1(\Omega)}$ 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 $\|u - \Pi_h u\|_{L^1(\Omega_h)}$ where Π_h is defined by a mesh $\mathcal{H}$ of a discretized domain Ω_h which is a unit mesh with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$. The interpolation error is first derived locally (on an element) for a quadratic function, then is extended to a global definition (for any point of the domain).

Local continuous interpolate

As in Section 5.1.1, we will first express the leading order of the interpolation error on a single element, depending on its shape and size. As, in 1D, it can be shown that the quadratic term in the Taylor expansion of the considered function is the main source of error. This integral thus depends on the hessian of the function and the shape of the element.

Let us consider a quadratic function u defined on a domain $\Omega \subset \mathbb{R}^3$, and a continuous element $\mathcal{M}$. For all unit elements K with respect to $\mathcal{M}$, the interpolation error of u in L^1 norm can be reshaped on a reference element $\hat{K}$, unit in the euclidian metric, with the transformation $\mathbf{x} \to \hat{\mathbf{x}} = \mathcal{M}^{\frac{1}{2}}\mathbf{x}$, so that

$$\int_K |u(\mathbf{x}) - \Pi_h u(\mathbf{x})| \mathrm{d}\mathbf{x} = \int_{\hat{K}} \left| u\left(\mathcal{M}^{-\frac{1}{2}}\hat{\mathbf{x}}\right) - \Pi_h u\left(\mathcal{M}^{-\frac{1}{2}}\hat{\mathbf{x}}\right)\right| \det(\mathcal{M}^{-\frac{1}{2}}) \mathrm{d}\hat{\mathbf{x}}. \tag{5.23}$$

As $u(\mathbf{x})$ is quadratic, so is $\hat{u} : \hat{\mathbf{x}} \to u(\mathcal{M}^{-\frac{1}{2}}\hat{\mathbf{x}})$ and its hessian verifies

$$H_{\hat{u}} = {\mathcal{M}^{-\frac{1}{2}}}^T H_u \mathcal{M}^{-\frac{1}{2}}.$$

The transformation $\mathbf{x} \to \hat{\mathbf{x}} = \mathcal{M}^{\frac{1}{2}}\mathbf{x}$ maps the element K on a regular element of an arbitrary orientation. In order to ease the computation of the integral, we further rotate it on a single reference element with a rotation $\mathcal{R}$ so that $H_{\hat{u}} = \mathcal{R}^T {\mathcal{M}^{-\frac{1}{2}}}^T H_u \mathcal{M}^{-\frac{1}{2}}\mathcal{R}$. At this point we decompose $\hat{u} - \Pi_h\hat{u}$ on the quadratic basis function depending on its hessian and directly compute the integral and show

$$\begin{aligned}
\int_{\hat{K}} |\hat{u}(\hat{\mathbf{x}}) - \Pi_h\hat{u}(\hat{\mathbf{x}})| \, \mathrm{d}\hat{\mathbf{x}} &= C \ \mathrm{trace}\left(|H_{\hat{u}}|\right) \\
&= C \ \mathrm{trace}\left(\mathcal{R}^T {\mathcal{M}^{-\frac{1}{2}}}^T |H_u| \mathcal{M}^{-\frac{1}{2}}\mathcal{R}\right) \\
&= C \ \mathrm{trace}\left({\mathcal{M}^{-\frac{1}{2}}}^T |H_u| \mathcal{M}^{-\frac{1}{2}}\right),
\end{aligned}$$

with $C = \frac{\sqrt{3}}{64}$ in 2D, and $C = \frac{\sqrt{2}}{240}$ in 3D. This expression shows that the interpolation error does not depend on the orientation of the element but is only a function of the Hessian H_u of u and of the continuous element $\mathcal{M}$. Using this expression in Equation (5.23), we obtain

- In 3D:
$$\|u - \Pi_h u\|_{L^1(K)} = \frac{\sqrt{2}}{240} \det(\mathcal{M}^{-\frac{1}{2}}) \, \mathrm{trace}(\mathcal{M}^{-\frac{1}{2}} \, H_u \, \mathcal{M}^{-\frac{1}{2}}). \tag{5.24}$$
- In 2D:
$$\|u - \Pi_h u\|_{L^1(K)} = \frac{\sqrt{3}}{64} \det(\mathcal{M}^{-\frac{1}{2}}) \, \mathrm{trace}(\mathcal{M}^{-\frac{1}{2}} \, H_u \, \mathcal{M}^{-\frac{1}{2}}).$$

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

Global continuous interpolate

To define a global continuous linear interpolate, the problem is once again how to move from an expression valid for *one continuous element* to an expression for the case in which *the metric varies point-wise*. Let us now suppose now that the continuous mesh $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is varying and that the function u is no more quadratic but only twice continuously differentiable. Equality (5.24) does not hold anymore, but all the terms of the right-hand-side $\mathcal{M}$ and H are still well defined continuously.

In the vicinity of $\mathbf{a}$, u_Q is the quadratic approximation of smooth function u and $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ reduces to $\mathcal{M}(\mathbf{a})$ in the tangent space. There exists a unique function $\pi_{\mathcal{M}}$ such that:

$$\forall \mathbf{a}\in\Omega\,,\quad |u-\pi_{\mathcal{M}}u|(\mathbf{a}) \;=\; \frac{\|u_Q-\Pi_h u_Q\|_{L^1(K)}}{|K|} \;=\; \frac{1}{20}\,\mathrm{trace}\big(\mathcal{M}(\mathbf{a})^{-\frac{1}{2}}\,|H(\mathbf{a})|\,\mathcal{M}(\mathbf{a})^{-\frac{1}{2}}\big)\,, \tag{5.25}$$

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

This result underlines another discrete-continuous duality by pointing out a continuous counterpart of the interpolation error. For this reason, the following formalism was adopted: $\pi_{\mathcal{M}}$ is called *continuous linear interpolate* and $|u-\pi_{\mathcal{M}}u|$ represents the continuous dual of the interpolation error. From a practical point of view, we deduce the following analogy. 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:

$$\|u-\Pi_h u\|_{L^1(\Omega_h)} = \sum_{K\in\mathcal{H}} \|u-\Pi_h u\|_{L^1(K)}. \tag{5.26}$$

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

$$\|u-\pi_{\mathcal{M}}u\|_{L^1(\Omega)} = \int_\Omega |u-\pi_{\mathcal{M}}u|(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{5.27}$$

There is no global guarantee on the reliability of the continuous interpolation error given by Relation (5.27), and in particular, there is no *a priori* relationship between (5.26) and (5.27). The only guarantee is the local equivalence given by Equation (5.25). However, the local guarantee becomes global when the mesh is unit with respect to a constant metric tensor and when the function is quadratic. In the latter case, by neglecting error due to the boundary discretization, we have the equality:

$$\|u-\Pi_h u\|_{L^1(\Omega_h)} = \|u-\pi_{\mathcal{M}}u\|_{L^1(\Omega)}\,, \tag{5.28}$$

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

Several examples are given in [Loseille 2011b], both analytic and numerical, that confirm the validity of this analogy. They show that the model is accurate and the equivalence (5.26)≈(5.27) is observed even for non quadratic functions and non-constant continuous meshes, and that the error due to the fact that the mesh generator generates edges with length not strictly equal to one is negligible. In particular, the range for the lengths of the edges given in Section 5.2.2 ensures reliable numerical results.

5.3.3 Summary

We have presented a framework that draws a correspondence between the discrete domain and the continuous domain. This framework is summarized in Table 5.1.

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

Table 5.1: The continuous mesh model draws a correspondence between the discrete domain and the continuous domain.

5.4 Multiscale mesh adaptation

In the previous section, we have presented the continuous mesh model, on which is based a powerful framework to handle mathematically adapted meshes. It reinforces the use of metrics that was made previously, since it establishes a duality between metric fields and adapted meshes, and it enables to work with metrics instead of discrete meshes.

In the following section, we explain how this framework is used for mesh adaptation.

5.4.1 The non-linear adaptation loop

So far, we have seen fundamental mathematical concepts used in metric-based mesh adaptation. Let us now see how they are involved in mesh adaptation.

Steady algorithm

The basic principle of metric-based mesh adaptation is: from an initial solution on an initial mesh, an error estimate is computed (based on an error estimate model), from which a metric field is computed, which is used to generate a mesh adapted to the solution. However, the mesh will be adapted to the numerical solution on the initial mesh, but not necessarily to the physical solution: if a new numerical solution is computed on the adapted mesh, new physical features may appear, to which the mesh is not adapted. This why the mesh adaptation problem is intrinsically a **non-linear** problem.

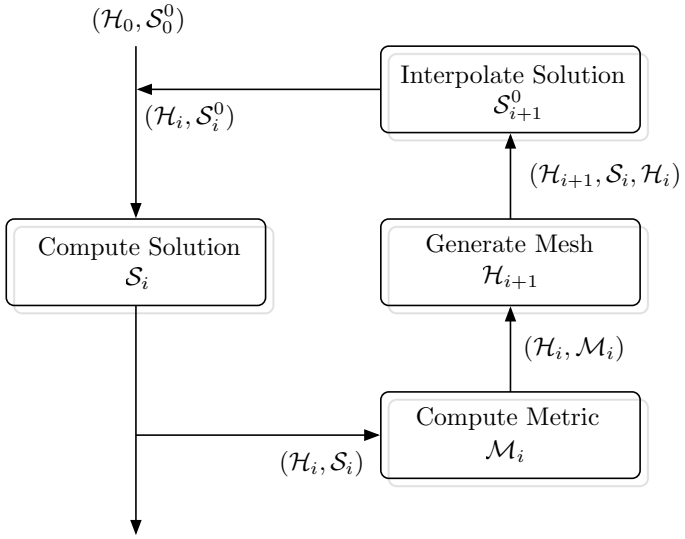


Figure 5.7: The non-linear steady adaptation loop. i is the adaptive loop iteration index, $\mathcal{H}^i$, $\mathcal{S}^i$, $\mathcal{S}_0^i$ and $\mathbf{M}_i$ denote the mesh, the solution, the initial solution and the metric field at iteration i, respectively.

Therefore, an **iterative process** is necessary to converge the mesh/solution couple, that generates a sequence of consecutively adapted meshes. In the steady state case, there is only one mesh and one solution. The algorithm is illustrated in Figure 5.7. As input is given a pair of initial mesh and solution: $(\mathcal{H}^0, \mathcal{S}^0)$. (Note that the first solution can be the uniform solution or any partly converged solution). A first solution $\mathcal{S}^1$ is computed, from which a metric field $\mathcal{M}^1$ is deduced as will be explained further, and a new mesh $\mathcal{H}^1$ is generated using this metric field. Solution $\mathcal{S}^1$ is then transferred to $\mathcal{H}^1$. The new pair $(\mathcal{H}^1, \mathcal{S}^1)$ is then used as input for a new iteration of the algorithm.

The process ends when the solution stops evolving, for a given number of vertices, or formally speaking when an error threshold is reached. It is because the number of vertices is fixed that

the process eventually converges: at some point, the vertices are optimally distributed, and only increasing the number of vertices could reduce the error. In practice, the number of vertices is given approximately through the notion of mesh complexity (see Section 5.3), and the number of iterations of the algorithm is set in advance to reach a good level of convergence (typically five iterations are performed).

5.4.2 Optimal control of the interpolation error and optimal meshes

Mesh adaptation consists in finding the mesh that minimizes a certain error on a domain, for a certain sensor function. As already mentioned previously, the error we consider is the interpolation error, that we control in L^p norm - depending on the choice of p, different aspects of the solution are captured, as will be seen later. The problem is stated *a priori*:

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

(P) is too complex to be solved directly, since the unknown, *i.e.* the mesh, is made up of vertices and their topology, which is far too many degrees of freedom. Besides, several optimal meshes can be found for one sensor function (think about merely swapping an element), so the problem is ill-posed. On the other hand, it is possible to show that the problem moved to the continuous domain is well posed, and can be solved using calculus of variations. Moreover, the continuous formulation of the adaptation problem uses a global point of view, while usual methods focus on a local analysis of the error. We have shown in Section 5.1.1 that the continuous approach provides a much faster convergence. The reformulated problem is:

$$\text{Find } \mathbf{M}_{opt} \text{ having a complexity of } N \text{ such that } \mathbf{E}_{L^p}(\mathbf{M}_{opt}) = \min_{\mathbf{M}} \|u - \pi_{\mathcal{M}} u\|_{L^p(\Omega)}. \qquad (5.29)$$

Using the definition of the linear continuous interpolate $\pi_{\mathcal{M}}$ given by Equation (5.25), the well-posed global optimization problem of finding the optimal continuous mesh minimizing the continuous interpolation error in L^p norm can be established:

$$\begin{aligned} \text{Find } \mathbf{M}_{L^p} = \min_{\mathbf{M}} \mathbf{E}_{L^p}(\mathbf{M}) &= \left(\int_\Omega \big(u(\mathbf{x}) - \pi_{\mathcal{M}} u(\mathbf{x})\big)^p \, \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}} \\ &= \left(\int_\Omega \mathrm{trace}\Big(\mathcal{M}(\mathbf{x})^{-\frac{1}{2}} |H_u(\mathbf{x})| \mathcal{M}(\mathbf{x})^{-\frac{1}{2}} \Big)^p \, \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}}, \end{aligned} \qquad (5.30)$$

under the constraint $\mathcal{C}(\mathbf{M}) = \displaystyle\int_\Omega d(\mathbf{x}) \, \mathrm{d}\mathbf{x} = N$. The constraint on the complexity notably avoids the trivial solution where all $(h_i)_{i=1,3}$ are zero which provides a null error.

Contrary to a discrete analysis, this problem can be solved globally by using a calculus of variations that is well-defined on the space of continuous meshes. In [Loseille 2011b], it is proven that Problem (5.30) admits a unique solution. We will give here a brief summary of the proof.

Let u be a twice continuously differentiable function defined on $\Omega \subset \mathbb{R}^3$, H_u its Hessian. As in Section 5.1.1, the constrained optimization problem is reshaped in a simple minimization

problem with the lagrangian

$$\mathcal{L}(\mathbf{M},\lambda)=\int_\Omega \mathrm{trace}\Big(\mathcal{M}(\mathbf{x})^{-\frac{1}{2}}|H_u(\mathbf{x})|\mathcal{M}(\mathbf{x})^{-\frac{1}{2}}\Big)^p\,\mathrm{d}\mathbf{x}-\lambda\left(\int_\Omega d(\mathbf{x})\,\mathrm{d}\mathbf{x}-N\right). \tag{5.31}$$

We used here the fact that $x\to x^{\frac{1}{p}}$ increases with x so that finding the minimum of $x^{\frac{1}{p}}$ is equivalent to finding the minimum of x. Using the decomposition in Equation (5.21), we rewrite Equation (5.31) as a minimization problem on the characteristics of the metric, its density d, its anisotropic ratios r_1 and r_2 and its eigen directions $\mathbf{v}_1$ and $\mathbf{v}_2$, in 3D. Note that the third anisotropic ratio is related to the two others by the relation $r_1r_2r_3=1$ and the third eigen direction is orthogonal to the first ones. Hence, we obtain the minimization problem

$$\mathcal{L}(d,\overline{\mathcal{M}},\lambda)=\left(\int_\Omega d^{\frac{-2p}{3}}(\mathbf{x})\,\mathrm{trace}\Big(\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}|H_u(\mathbf{x})|\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}\Big)^p\,\mathrm{d}\mathbf{x}\right)-\lambda\left(\int_\Omega d(\mathbf{x})-N\right), \tag{5.32}$$

with $\overline{\mathcal{M}}$ the anisotropic metric of unit determinant

$$\overline{\mathcal{M}}=\mathcal{R}(\mathbf{x})\begin{pmatrix} r_1^{-\frac{2}{3}}(\mathbf{x}) & & \\ & r_2^{-\frac{2}{3}}(\mathbf{x}) & \\ & & r_3^{-\frac{2}{3}}(\mathbf{x})\end{pmatrix}\mathcal{R}^T(\mathbf{x}).$$

In Equation (5.32), the anisotropic properties of the metric field have been decoupled from the isotropic scaling on which the constraint is imposed. This greatly simplifies the resolution of the problem as we now have

$$\frac{\partial\mathcal{L}}{\partial d} = \int_\Omega\left(-\frac{2p}{3}d^{-\frac{2p+3}{3}}(\mathbf{x})\,\mathrm{trace}\Big(\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}|H_u(\mathbf{x})|\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}\Big)^p-\lambda\right)\delta d(\mathbf{x}),\quad \forall\delta d \tag{5.33}$$

$$\frac{\partial\mathcal{L}}{\partial\overline{\mathcal{M}}} = \int_\Omega d^{\frac{-2p}{3}}(\mathbf{x})\frac{\partial}{\partial\overline{\mathcal{M}}}\left(\mathrm{trace}\Big(\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}|H_u(\mathbf{x})|\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}\Big)^p\right)\delta\overline{\mathcal{M}}(\mathbf{x}),\quad \forall\delta\overline{\mathcal{M}}, \tag{5.34}$$

so that

$$-\frac{2p}{3}d^{-\frac{2p+3}{3}}(\mathbf{x})\,\mathrm{trace}\Big(\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}|H_u(\mathbf{x})|\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}\Big)^p = \lambda \tag{5.35}$$

$$\frac{\partial}{\partial\overline{\mathcal{M}}}\left(\mathrm{trace}\Big(\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}|H_u(\mathbf{x})|\overline{\mathcal{M}}(\mathbf{x})^{-\frac{1}{2}}\Big)^p\right) = 0. \tag{5.36}$$

Note here that Equation (5.36) is independent of the density d and can be solved separately. Namely, the anisotropic properties of the optimal metric field is a purely local property that only depends on the hessian of u. The detailed resolution of Equation (5.36) can be found in [Loseille 2011b]. Basically, it consists in verifying that the error is minimal when the eigen direction of the metric are aligned with those of the hessian and then deduce the optimal anisotropic ratio.

Once the optimal anisotropy field $\overline{\mathcal{M}}(\mathbf{x})$ has been determined, we replace it in Equation (5.35), which gives an expression of d with respect to λ and $\overline{\mathcal{M}}$. Finally, as in Section 5.1.1, λ is eliminated using the constraint $\mathcal{C}(\mathbf{M})=\displaystyle\int_\Omega d(\mathbf{x})\,\mathrm{d}\mathbf{x}=N$.

Finally, the optimal continuous mesh $\mathbf{M}_{L^p}(u)$ minimizing Problem (5.30) reads locally:

$$\mathcal{M}_{L^p}(\mathbf{x}) = N^{\frac{2}{3}} \left(\int_\Omega \det(|H_u(\bar{\mathbf{x}})|)^{\frac{p}{2p+3}} \, \mathrm{d}\bar{\mathbf{x}} \right)^{-\frac{2}{3}} \det(|H_u(\mathbf{x})|)^{\frac{-1}{2p+3}} \, |H_u(\mathbf{x})| \, . \tag{5.37}$$

It verifies the following properties:

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

$$\|u - \pi_{\mathcal{M}_{L^p}} u\|_{L^p(\Omega)} = 3 \, N^{-\frac{2}{3}} \left(\int_\Omega \det \left(|H_u| \right)^{\frac{p}{2p+3}} \right)^{\frac{2p+3}{3p}} \, . \tag{5.38}$$

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

$$\|u - \pi_{\mathcal{M}^N_{L^p}} u\|_{L^p(\Omega)} \leq \frac{Cst}{N^{2/3}}. \tag{5.39}$$

  Relation (5.39) points out a global second order of mesh convergence.

5.4.3 General error control

In the previous sections we developed an analysis and minimization of the interpolation error in L^p norm. To do so, we computed a continuous local error model, valid in the asymptotic regime. The key point to determine the minimizing metric field was the fact that we were able to separate the isotropic scaling from the anisotropy. This is somehow natural as the error should be auto-similar as the element size tends to zero.

We thus extend our analysis to an arbitrary local error model $\varepsilon(\mathcal{M})$. We mention here only the dependency to the metric field but ε may depend on many other given fields (solution, adjoint, ...). This error model can be expressed as a function of the density d and the unit metric $\overline{\mathcal{M}}$ as previously, $\varepsilon(d, \overline{\mathcal{M}})$. We then seek an asymptotic expansion of ε with respect to the density as the mesh size tends to 0 ($h \to 0 \iff d \to \infty$):

$$\varepsilon(d, \overline{\mathcal{M}}) \sim_{d\to\infty} \epsilon(\overline{\mathcal{M}}) d^\beta. \tag{5.40}$$

The same error analysis is performed as previously with the lagrangian

$$\mathcal{L}(d, \overline{\mathcal{M}}, \lambda) = \int_\Omega \epsilon \left(\overline{\mathcal{M}}(\mathbf{x}), \mathbf{x} \right) d^\beta(\mathbf{x}) \mathrm{d}\mathbf{x} - \lambda \left(\int_\Omega d(\mathbf{x}) \mathrm{d}\mathbf{x} - N \right) \tag{5.41}$$

so that

$$\beta d^{\beta-1} \epsilon \left(\overline{\mathcal{M}}(\mathbf{x}), \mathbf{x} \right) = \lambda \tag{5.42}$$

$$\frac{\partial}{\partial \overline{\mathcal{M}}} \epsilon(\overline{\mathcal{M}}(\mathbf{x})) = 0. \tag{5.43}$$

Equation (5.43) can be solved separately depending on its expression to compute $\overline{\mathcal{M}}(\mathbf{x})$. In the L^p case, we were able to derive an analytical expression but the same resolution can be done iteratively. Though, it is important to note that contrary to the discrete case that we analysed, it is here a purely local operation.

Knowing $\overline{\mathcal{M}}(\mathbf{x})$, we can then solve Equation (5.42) as

$$d(x) = \left(\frac{\lambda}{\beta \epsilon \left(\overline{\mathcal{M}}(\mathbf{x}) \right)} \right)^{\frac{1}{\beta - 1}} \tag{5.44}$$

and normalize

$$\left(\frac{\lambda}{\beta} \right)^{\frac{1}{\beta-1}} \int_\Omega \epsilon \left(\overline{\mathcal{M}}(\mathbf{x}) \right)^{-\frac{1}{\beta-1}} \mathrm{d}\mathbf{x} = N \tag{5.45}$$

so that

$$d(\mathbf{x}) = \frac{N}{\displaystyle\int_\Omega \epsilon \left(\overline{\mathcal{M}}(\mathbf{x}) \right)^{-\frac{1}{\beta-1}} \mathrm{d}\mathbf{x}} \left(\epsilon \left(\overline{\mathcal{M}}(\mathbf{x}) \right) \right)^{-\frac{1}{\beta-1}} \tag{5.46}$$

and finally

$$\mathcal{M}(\mathbf{x}) = N^{\frac{2}{3}} \left(\int_\Omega \epsilon \left(\overline{\mathcal{M}}(\mathbf{x}) \right)^{-\frac{1}{\beta-1}} \mathrm{d}\mathbf{x} \right)^{-\frac{2}{3}} \left(\epsilon \left(\overline{\mathcal{M}}(\mathbf{x}) \right) \right)^{-\frac{1}{\beta-1}} \overline{\mathcal{M}}(\mathbf{x}). \tag{5.47}$$

Thus, the global balance can be done, no matter how the local error $\epsilon \left(\overline{\mathcal{M}}(\mathbf{x}) \right)$ is computed. As we saw in Section 5.1.1, it does not require an iterative process to converge contrary to a discrete approach.

In the case of the interpolation error in L^p-norm analysed above, we have

$$\epsilon(\overline{\mathcal{M}}) = \mathrm{trace} \left(\overline{\mathcal{M}}_{L^p}^{-\frac{1}{2}} \left| \mathcal{H}_u \right| \overline{\mathcal{M}}_{L^p}^{-\frac{1}{2}} \right)^p,$$

and

$$\beta = -\frac{2p}{3}.$$

5.4.4 Error equidistribution

We now analyse in the general case the error per element given by in the optimal metric. Let us consider a mesh Ω_h unit for the optimal metric $\mathbf{M}_{L^p}$ and K a unit element of Ω_h. The local error on K writes

$$e(K) = \int_K \epsilon(\overline{\mathcal{M}}(\mathbf{x})) d^\beta(\mathbf{x}) \mathrm{d}\mathbf{x},$$

with, thanks to Equation 5.47

$$\epsilon(\overline{\mathcal{M}}(\mathbf{x})) = C_1 d(\mathbf{x})^{1-\beta},$$

where

$$C_1 = \left(\frac{N}{\displaystyle\int_\Omega \epsilon \left(\overline{\mathcal{M}}(\mathbf{x}) \right)^{-\frac{1}{\beta-1}} \mathrm{d}\mathbf{x}} \right)^{1-\beta}.$$

Hence, we have

$$e(K) = \int_K C_1 d(\mathbf{x})^{1-\beta} d^\beta(\mathbf{x}) \mathrm{d}\mathbf{x} = \int_K C_1 d(\mathbf{x}) \mathrm{d}\mathbf{x}.$$

By construction, we recall that as K is unit in the metric, $\int_K d(\mathbf{x})\mathrm{d}\mathbf{x} = V_{unit}$ so that the error per element is constant

$$e(K) = C_1 V_{unit}.$$

In other words, the error is equidistributed in the domain by the normalization process.

In the case of the interpolation error in L^p norm, we recall that

$$\epsilon(\overline{\mathcal{M}}(\mathbf{x})) = \mathrm{trace}\left(\overline{\mathcal{M}}_{L^p}^{-\frac{1}{2}}(\mathbf{x}) \left|\mathcal{H}_u(\mathbf{x})\right| \overline{\mathcal{M}}_{L^p}^{-\frac{1}{2}}(\mathbf{x}) \right)^p,$$

and

$$\beta = -\frac{2p}{3}.$$

Thus the element error writes,

$$e_{L^p}^p(K) = \int_K \mathrm{trace}\left(\mathcal{M}_{L^p}^{-\frac{1}{2}}(\mathbf{x}) \left|\mathcal{H}_u(\mathbf{x})\right| \mathcal{M}_{L^p}^{-\frac{1}{2}}(\mathbf{x}) \right)^p dK\,,$$

where

$$\mathcal{M}_{L^p}^{-\frac{1}{2}}(\mathbf{x}) = D_{L^p}^{-\frac{1}{2}} \det\left(\left|\mathcal{H}_u(\mathbf{x})\right|\right)^{\frac{1}{2(2p+3)}} \left|\mathcal{H}_u(\mathbf{x})\right|^{-\frac{1}{2}}$$

with

$$D_{L^p} = N^{\frac{2}{3}} \left(\int_\Omega \det\left(\left|\mathcal{H}_u(\tilde{\mathbf{x}})\right|\right)^{\frac{p}{2p+3}} d\Omega \right)^{-\frac{2}{3}}.$$

So,

$$\mathcal{M}_{L^p}^{-\frac{1}{2}}(\mathbf{x}) \left|\mathcal{H}_u(\mathbf{x})\right| \mathcal{M}_{L^p}^{-\frac{1}{2}}(\mathbf{x}) = D_{L^p}^{-1} \det\left(\left|\mathcal{H}_u(\mathbf{x})\right|\right)^{\frac{1}{2p+3}} \mathcal{I}_3$$

and we obtain the error per element

$$e_{L^p}^p(K) = \int_K 3^p D_{L^p}^{-p} \det\left(\left|\mathcal{H}_u(\mathbf{x})\right|\right)^{\frac{p}{2p+3}} dK = 3^p D_{L^p}^{-p} \det\left(\left|\mathcal{H}_K\right|\right)^{\frac{p}{2p+3}} |K|\,,$$

the hessian $\mathcal{H}_u$ being assumed constant per element. Finally, as K is unit in the metric, we have

$$\begin{aligned} |K| &= V_{unit} \left(\det \mathcal{M}_K\right)^{-\frac{1}{2}} \\ &= V_{unit} \left(D_{L^p}^3 \det\left(\left|\mathcal{H}_K\right|\right)^{-\frac{3}{2p+3}} \det\left(\left|\mathcal{H}_K\right|\right) \right)^{-\frac{1}{2}} \\ &= V_{unit} D_{L^p}^{-\frac{3}{2}} \det\left(\left|\mathcal{H}_K\right|\right)^{-\frac{p}{2p+3}}\,, \end{aligned}$$

and thus

$$e_{L^p}^p(K) = 3^p V_{unit} D_{L^p}^{-\frac{2p+3}{2}} = cste\,.$$

5.4.5 Application to numerical solutions

In practice, we do not know the whole solution, but only the solution at the vertices of the mesh. Let us now describe how the interpolation theory is applied when only u_h, a piecewise linear approximation of the solution, is known. Indeed, in this particular case, the interpolation error estimate is neither applied directly to u nor to u_h.

Let $\bar{V}_h^k$ be the space of piecewise polynomials of degree k (possibly discontinuous) 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. We denote by R_h a reconstruction operator applied to numerical approximation u_h. This reconstruction operator can be either a recovery process [Zienkiewicz 1992a], a hierarchical basis [Bank 1993], or an operator connected to an a posteriori estimate [Huang 2010]. We assume that the reconstruction $R_h u_h$ is more accurate than u_h for a given norm $\|.\|$ in the sense that:

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

From the triangle inequality, we deduce:

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

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

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

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

$$\|u - u_h\|_{L^p(\Omega_h)} \leq \frac{3\,N^{-\frac{2}{3}}}{1-\alpha}\left(\int_\Omega \det\left(|H_{R_h u_h}(\mathbf{x})|\right)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}\right)^{\frac{2p+3}{3p}}\,.$$

This majoration is true for any $k \geq 1$ but is too large for $k > 1$. Though, the same analysis can be performed in general for derivatives of order $k+1$ and similar relations can be found in [Coulaud 2016, Mirebeau 2010].

Remark 4. *It is important to note that $\mathcal{M}_{L^p}$ defined by Relation (5.37) applied to $R_h u_h$ does not enables to generate an optimal adapted mesh to control the approximation error $\|u - u_h\|$. If all assumptions are verified, this analysis states that such generated adapted meshes control the approximation error (we have only an upper bound).*

In the context of numerical simulations, u_h lies in V_h^1 and its derivatives ∇u_h in $\bar{V}_h^0$. We propose a reconstruction operator from V_h^1 into V_h^2 based on $\mathbb{P}_2$ Lagrange finite element test functions. As approximate solution u_h is only known at mesh vertices, we need to reconstruct mid-edge values. To this end, we consider the L^2-projection operator $\mathcal{P} : \bar{V}_h^0 \to V_h^1$ defined by [Clément 1975]:

$$\nabla_R u_h = \mathcal{P}(\nabla u_h) = \sum_{\mathbf{p}_i \in \mathcal{H}} \nabla_R\, u_h(\mathbf{p}_i)\phi_i \quad \text{where} \quad \nabla_R\, u_h(\mathbf{p}_i) = \frac{\sum_{K_j \in S_i} |K_j| \nabla(u_{h|K_j})}{\sum_{K_j \in S_i} |K_j|}\,,$$

where $\mathbf{p}_i$ denotes the i^{th} vertex of mesh $\mathcal{H}$, S_i is the stencil of $\mathbf{p}_i$, ϕ the basis function of V_h^1 and $|K_j|$ denotes the volume of element K_j. These nodal recovered gradients are used to evaluate mid-edge values. For edge $\mathbf{e} = \mathbf{pq}$, the mid-edge value $u_h(\mathbf{e})$ is given by:

$$u_h(\mathbf{e}) = \frac{u_h(\mathbf{p}) + u_h(\mathbf{q})}{2} + \frac{\nabla_R u_h(\mathbf{p}) - \nabla_R u_h(\mathbf{q})}{8} \cdot \mathbf{pq}\,,$$

which corresponds to a cubic reconstruction. The reconstructed function $R_h u_h$ of V_h^2 writes:

$$R_h u_h = \sum_{\mathbf{p}_i} u_h(\mathbf{p}_i)\psi_{\mathbf{p}_i} + \sum_{\mathbf{e}_j} u_h(\mathbf{e}_j)\psi_{\mathbf{e}_j}\,,$$

where $\psi_{\mathbf{p}} = \phi_{\mathbf{p}}\,(2\,\phi_{\mathbf{p}} - 1)$ and $\psi_{\mathbf{e}} = 4\,\phi_{\mathbf{p}}\,\phi_{\mathbf{q}}$ are the $\mathbb{P}_2$ Lagrange test functions. This reconstructed function can be rewritten $R_h u_h = u_h + z_h$ and by definition verifies:

$$\Pi_h R_h u_h = u_h \quad \text{thus} \quad \Pi_h z_h = 0\,.$$

Therefore, we deduce:

$$\|R_h u_h - \Pi_h R_h u_h\| = \|u_h + z_h - u_h\| = \|z_h - \Pi_h z_h\|\,.$$

Finally, the approximation error can be estimated by evaluating the interpolation error of z_h:

$$\|u - u_h\| \;\leq\; \frac{1}{1-\alpha}\|z_h - \Pi_h z_h\| \;\leq\; \frac{3\,N^{-\frac{2}{3}}}{1-\alpha}\left(\int_\Omega \det\left(|H_{z_h}(\mathbf{x})|\right)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x}\right)^{\frac{2p+3}{3p}}\,.$$

Note that the Hessian of z_h lies in $\bar{V}_h^0$. If nodal values are needed to build $\mathcal{M}_{L^p}$, then the L^2-projection operator can be applied to these Hessians, as explained in Section 6.1.5. This recovery procedure is somehow similar to the ones of [Zienkiewicz 1992a, Zienkiewicz 1992b]. Other reconstruction operators can be applied such as the double L^2-projection, the least square method or eventually the Green formula based approach [Frey 2005].

5.4.6 Control of the error in L^p norm

The classic analysis for mesh adaptation [Alauzet 2003a, Castro-Díaz 1997] was performed in L^∞ norm. However this resulted in a loss of anisotropy of the adapted meshes in some cases [Loseille 2007b]. Indeed, let us consider the Heaviside function with a $\delta > 0$ step: $u(x) = \delta$ if $x > 0$ and $u(x) = 0$ if $x \leq 0$, on the segment $[-1, 1]$. Given a uniform mesh of this domain with size parameter h, then we can demonstrate that the interpolation error in L^p norm converges at order $O(h^{\frac{1}{p}})$ and thus the spatial convergence order is $O(h^{\frac{1}{p}})$. When p tends to infinity, the expected convergence order is $O(1)$. In other words, the L^∞ norm will never converge in presence of discontinuities. In practice, thanks to the prescription of a minimal size, the algorithm does not diverge. An L^∞ error estimate is thus not suitable for solutions with discontinuities. Consequently, L^p strategies become of main interest in the case of discontinuous solutions.

Another problem with the use of L^∞ error estimates is that a control of the interpolation error in L^∞ does not capture the small-scale features of the solution. Several modifications

of the L^∞ interpolation error estimate have been considered [Castro-Díaz 1997, Löhner 1990, Frey 2005] to overcome this issue. However, such local normalizations are only slightly more sensitive and the control of the interpolation error remains in L^∞ norm. Another interest of the optimal metric in 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 2011b]. 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 5.8 (top left) where a cut through the line $y = 0$ is depicted.

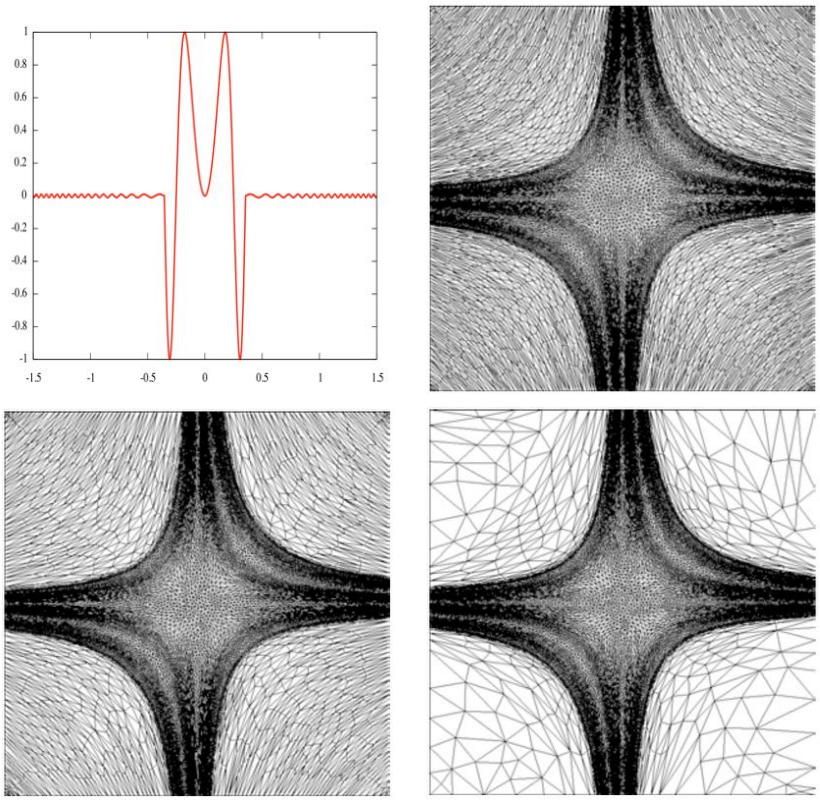


Figure 5.8: Top left, representation of function f_1 along the cut line $y = 0$. Optimal adapted meshes for norms L^1 (top right), L^2 (bottom left) and L^4 (bottom right). Each mesh is composed of about 7 000 vertices.

The mesh adaptation process based on the control of the interpolation error is analyzed for

the L^1, L^2 and L^4 norms. Figure 5.8 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 L^p norm with a lower p (L^1 is the best) whereas the L^4 norm ignores small amplitudes regions and clearly refines more large amplitudes areas. This behavior is due to the term $\det |H_u|^{\frac{-1}{2p+n}}$ in Relation (5.37) which gives more sensitivity to lower p norm. It illustrates that controlling the interpolation error in L^p norm with a small value of p enables to capture all the scales of the solution.

5.4.7 Conclusion

In this Chapter we introduced the continuous mesh context and the various tools to relate interpolation error to the continuous mesh size and optimize it. This is the base of RANS error estimation and mesh adaptation sensors presented in the following chapter.

Chapter 6

RANS mesh adaptation

6.1 RANS goal-oriented mesh adaptation

In this section we will relate the adjoint-based error estimation for RANS equations to the local mesh size and deduce the optimal metric.

6.1.1 General non-linear error estimation

Standard error estimation tools rely on linear problems. The first step to deal with RANS equations is thus to linearize them. In Section 4.1, we shown that given a continuous residual Ψ, solving

$$\int_\Omega \Psi(\tilde{W}_h)\varphi_h \,\mathrm{d}\Omega = -\int_\Omega \Psi(W_h + W - \Pi_h W)\varphi_h \,\mathrm{d}\Omega,$$

could provide a pertinent correction and thus a pertinent error estimation. We traduce here this result in the standard Finite Element error estimation context.

We consider the linear discretization of the given set of equations

$$\Psi(W) = 0,$$

over the domain Ω. We denote Ω_h the discretization of Ω with simplicial elements (triangles, tetrahedra) by the mesh $\mathcal{H}$. We introduce the continuous and discrete functional spaces in which solutions are sought, $V = [H^1(\Omega)]^n$ and

$$V_h = \left\{\varphi_h \in V \cap \mathcal{C}^0 \quad | \quad \varphi_h|_K \text{ is affine } \forall K \text{ element of } \mathcal{H}\right\},$$

n being the number of variables, 5 in 2D and 6 in 3D for RANS equations with one turbulent variable. The continuous and discrete problem thus write

$$a(u,\varphi) = (f,\varphi), \qquad \forall \varphi \in V \tag{6.1}$$

$$a(u_h,\varphi_h) = (f_h,\varphi_h), \qquad \forall \varphi_h \in V_h \tag{6.2}$$

where a is continuous, non-linear with respect of its first argument and linear with respect of its second argument. We note $\bar{a}|_u$ the linearization of a in u. This is thus a billinear form. We introduce the functional $j(u)$ whose error will be minimized.

The mesh $\mathcal{H}$ being represented by the continuous metric $\mathcal{M}$, we seek for the solution of the minimization problem

$$\mathcal{M}_{Opt} = \min_{\mathcal{C}(\mathcal{M})=N} |j(u_h) - j(u)|.$$

In most cases, j is non-linear so that we write

$$j(u_h) - j(u) \approx \underbrace{J|_{u_h}(u_h - \Pi_h u)}_{\text{Implicit error}} + \underbrace{J|_{u_h}(\Pi_h u - u)}_{\text{Interpolation error}}, \tag{6.3}$$

where J is the differentiation of j in u_h. The interpolation error can be treated by usual means, we thus only keep here the implicit error. As J is linear, we obtain from the Ritz theorem that there exist $g \in V$ and $g_h \in V_h$ so that

$$(g, v) = J(v), \qquad \forall v \in V \tag{6.4}$$
$$(g_h, v_h) = J(v_h), \qquad \forall v_h \in V_h. \tag{6.5}$$

From a discrete point of view, J can be represented by a vector G so that given a variation δW of the solution,

$$J(\delta W) = G \cdot \delta W.$$

G is the linearization at each cell of the contribution to the functional. a being a billinear form, the Ritz theorem also implies that there exit a continuous adjoint w and discrete adjoint w_h so that

$$\bar{a}|_u(v, w) = J(v), \qquad \forall v \in V \tag{6.6}$$
$$\bar{a}|_{u_h}(v_h, w_h) = J(v_h), \qquad \forall v_h \in V_h. \tag{6.7}$$

From a discrete point of view, $\bar{a}|_{u_h}$ is the Jacobian of the residual, represented by the matrix $A_{ij} = \bar{a}|_{u_h}(\varphi_i, \varphi_j)$. The discrete residual being $R_i = a(u_h, \varphi_i) - (f_h, \varphi_i)$, any perturbation of the residual dR (due to mesh modifications), will lead to a modification of the solution expressed as

$$dW \approx A^{-1} dR$$

which will lead to a correction of the functional j,

$$dJ \approx G \cdot dW \approx G A^{-1} dR = dR \cdot W^*,$$

with

$$W^* = A^{-T} G.$$

Equation (6.7) thus writes

$$dJ = \underbrace{dW A}_{=dR} W^*.$$

From Equations (6.5) and (6.7) we obtain

$$J(u_h - \Pi_h u) = \bar{a}|_{u_h}(u_h - \Pi_h u, w_h) \tag{6.8}$$
$$\approx a(u_h, w_h) - a(\Pi_h u, w_h) \tag{6.9}$$
$$= \underbrace{a(u_h, w_h) - (f_h, w_h)}_{=0 \text{ from (6.2)}} - (a(\Pi_h u, w_h) - (f_h, w_h)) \tag{6.10}$$
$$= \underbrace{a(u, w_h) - (f, w_h)}_{=0 \text{ from (6.1)}} - (a(\Pi_h u, w_h) - (f_h, w_h)) \tag{6.11}$$
$$= \bar{a}|_u(u - \Pi_h u, w_h) + (f_h - f, w_h). \tag{6.12}$$

With Equation (6.12) we relate the implicit error $J(u_h - \Pi_h u)$ to interpolation errors. However, while $(f_h - f, w_h)$ is a simple weighted interpolation error, $\bar{a}|_u(u - \Pi_h u, w_h)$ actually involves gradients and second derivatives of interpolation errors that must be specifically treated.

In practice, $a(u, \varphi)$ will be expressed with a non-linear function Ψ as

$$a(u, \varphi) = \int_\Omega \Psi(u)\varphi \, \mathrm{d}\Omega,$$

so that $\bar{a}|_u(u - \Pi_h u, w_h)$ can be considered in two different ways

$$\begin{aligned} \bar{a}|_u(u - \Pi_h u, w_h) &= \int_\Omega \frac{\partial \Psi}{\partial u}(u - \Pi_h u) w_h \, \mathrm{d}\Omega, & (6.13) \\ &\approx \int_\Omega (\Psi(u) - \Psi(\Pi_h u)) w_h \, \mathrm{d}\Omega. & (6.14) \end{aligned}$$

These two expressions lead to two different error estimations.

6.1.2 RANS goal-oriented error estimation: Linearization

In the following sections, in order to ease the analysis, we refer to the velocity and coordinates components either with a natural notation

$$\mathbf{u} = (u, v, w) \quad \text{and} \quad \mathbf{x} = (x, y, z),$$

or with an indicial notation

$$\mathbf{u} = (u_i) \quad \text{and} \quad \mathbf{x} = (x_i),$$

depending on the context in which it is used. We also write the partial derivatives

$$\frac{\partial u_i}{\partial x_j} = (u_i)_{x_j},$$

for brevity.

Let us now apply results of Section 6.1.1 to RANS equations formulated in Section 1.1.1. Using notations of Equations (1.2),(1.3) and (1.4), Equation (6.1) simply writes

$$\begin{aligned} 0 &= a(W, W^*) \\ &= \int_\Omega ((F_1(W)_x + F_2(W)_y + F_3(W)_z) - (S_1(W)_x + S_2(W)_y + S_3(W)_z) - Q(W)) \cdot W^* \, \mathrm{d}\Omega. \end{aligned} \tag{6.15}$$

where the convective (inviscid) fluxes write

$$\begin{aligned} F_1(W) &= \left(\rho u,\ \rho u^2 + p,\ \rho uv,\ \rho uw,\ u(\rho E + p), \rho u\tilde{\nu}\right)^T, \\ F_2(W) &= \left(\rho v,\ \rho uv,\ \rho v^2 + p,\ \rho vw,\ v(\rho E + p), \rho v\tilde{\nu}\right)^T, \\ F_3(W) &= \left(\rho w,\ \rho uw,\ \rho vw,\ \rho w^2 + p,\ w(\rho E + p), \rho w\tilde{\nu}\right)^T. \end{aligned} \tag{6.16}$$

and the viscous fluxes

$$\begin{aligned} S_1(W) &= \Big(0,\ \tau_{xx},\ \tau_{xy},\ \tau_{xz},\ u\tau_{xx}+v\tau_{xy}+w\tau_{xz}+(\lambda+\lambda_t)\mathcal{T}_x, \frac{\rho}{\sigma}(\nu+\tilde{\nu})\tilde{\nu}_x\Big)^T, \\ S_2(W) &= \Big(0,\ \tau_{xy},\ \tau_{yy},\ \tau_{yz},\ u\tau_{xy}+v\tau_{yy}+w\tau_{yz}+(\lambda+\lambda_t)\mathcal{T}_y, \frac{\rho}{\sigma}(\nu+\tilde{\nu})\tilde{\nu}_y\Big)^T, \\ S_3(W) &= \Big(0,\ \tau_{xz},\ \tau_{yz},\ \tau_{zz},\ u\tau_{xz}+v\tau_{yz}+w\tau_{zz}+(\lambda+\lambda_t)\mathcal{T}_z, \frac{\rho}{\sigma}(\nu+\tilde{\nu})\tilde{\nu}_z\Big)^T, \end{aligned} \tag{6.17}$$

where τ_{ij} are the components of laminar stress tensor defined by:

$$\tau_{ij} = (\mu+\mu_t)\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}(\mu+\mu_t)\frac{\partial u_k}{\partial x_k}\delta_{ij}\,.$$

where δ_{ij} is the Kroneker symbol. Source terms Q are treated in Section 6.1.3.

As mentioned in Section 6.1.1, a key step of the process is the necessary linearization of the equations. This can be done using either Equation (6.13) or Equation (6.14) and there are thus different options presented below.

Linearization of the convective fluxes

Linearization by flux difference: The analysis of inviscid fluxes has been performed in [Loseille 2010] using Relation (6.14). Namely, we have

$$\begin{aligned} \delta J^{inv} &\approx \int_\Omega \Big(\Big(F_1(W)_x + F_2(W)_y + F_3(W)_z\Big) - \\ &\quad \Big(F_1(\Pi_h W)_x + F_2(\Pi_h W)_y + F_3(\Pi_h W)_z\Big)\Big)\cdot W^*\,\mathrm{d}\Omega, \\ &\approx \int_\Omega \Big(\Big(F_1(W)_x - (\Pi_h F_1(W))_x\Big) + \Big(F_2(W)_y - (\Pi_h F_2(W))_y\Big) \\ &\quad + \Big(F_3(W)_z - (\Pi_h F_3(W))_z\Big)\Big)\cdot W^*\,\mathrm{d}\Omega. \end{aligned}$$

Integrating by parts, we obtain

$$\begin{aligned} |\delta J^{inv}| &\approx \Big|\int_\Omega \Big(\Big(F_1(W)-\Pi_h F_1(W)\Big)\cdot W_x^* + \Big(F_2(W)-\Pi_h F_2(W)\Big)\cdot W_y^* \\ &\quad + \Big(F_3(W)-\Pi_h F_3(W)\Big)\cdot W_z^*\Big)\,\mathrm{d}\Omega\Big|, \\ &\leq \int_\Omega |W_x^*|\cdot|F_1(W)-\Pi_h F_1(W)| + |W_y^*|\cdot|F_2(W)-\Pi_h F_2(W)| \\ &\quad + |W_z^*|\cdot|F_3(W)-\Pi_h F_3(W)|\,\mathrm{d}\Omega, \end{aligned}$$

so that the inviscid part of the error δJ^{inv} can be treated as a weighted L^1 interpolation error of the convective fluxes. This estimate is composed of 8 terms in 2D and 15 terms in 3D.

Linearization with direct differentiation: Inviscid fluxes can also be linearized using Relation (6.13) and the direct differentiation of $(F_i)_{i=1,2,3}$ with respect to any set of variables W:

$$F_i(W) - F_i(\Pi_h W) \approx A_i \cdot (W - \Pi_h W).$$

We use here conservative variables which are natural variables for the FV-FE solver described in Section 1.1, but entropic variables can be used as well. We then obtain

$$\begin{aligned} \delta J^{inv} &\approx \int_\Omega \Big(\Big(F_1(W)_x + F_2(W)_y + F_3(W)_z \Big) - \\ &\quad \Big(F_1(\Pi_h W)_x + F_2(\Pi_h W)_y + F_3(\Pi_h W)_z \Big) \Big) \cdot W^* \, \mathrm{d}\Omega, \\ &\approx \int_\Omega \Big(\sum_i (A_i \cdot (W - \Pi_h W))_{x_i} \Big) \cdot W^* \, \mathrm{d}\Omega, \end{aligned} \tag{6.18}$$

and integrating by parts, we finally get

$$\begin{aligned} |\delta J^{inv}| &\approx \Big| \int_\Omega \Big(\sum_i A_i^T \cdot (W^*)_{x_i} \Big) \cdot (W - \Pi_h W) \, \mathrm{d}\Omega \Big|, \\ &\leq \int_\Omega \Big| \sum_i A_i^T \cdot (W^*)_{x_i} \Big| \cdot |W - \Pi_h W| \, \mathrm{d}\Omega. \end{aligned} \tag{6.19}$$

so that δJ^{inv} can be now treated as a weighted interpolation error on the considered set of variables. This estimate is composed of 4 terms in 2D an 5 terms in 3D. Note that contrary to previous estimate, the sum lies in the absolute value, so that this estimate may be sharper.

We recall that in 2D:

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ q^2 - (\rho u)^2 & -(\gamma - 3)(\rho u) & -(\gamma - 1)(\rho v) & \gamma - 1 \\ -(\rho u)(\rho v) & \rho v & \rho u & 0 \\ (2q^2 - \gamma E)(\rho u) & \gamma E - (\gamma - 1)(\rho u)^2 - q^2 & -(\gamma - 1)(\rho u)(\rho v) & \gamma(\rho u) \end{bmatrix} \tag{6.20}$$

and

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -(\rho u)(\rho v) & (\rho v) & (\rho u) & 0 \\ q^2 - (\rho v)^2 & -(\gamma - 1)(\rho u) & -(\gamma - 3)(\rho v) & \gamma - 1 \\ (2q^2 - \gamma E)(\rho v) & -(\gamma - 1)(\rho u)(\rho v) & \gamma E - (\gamma - 1)(\rho v)^2 - q^2 & \gamma(\rho v) \end{bmatrix} \tag{6.21}$$

with

$$q^2 = \frac{1}{2}(\gamma - 1)((\rho u)^2 + (\rho v)^2).$$

In 3D

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ q^2-(\rho u)^2 & -(\gamma-3)(\rho u) & -(\gamma-1)(\rho v) & -(\gamma-1)(\rho w) & (\gamma-1) \\ -(\rho u)(\rho v) & (\rho v) & (\rho u) & 0 & 0 \\ -(\rho u)(\rho w) & (\rho w) & 0 & (\rho u) & 0 \\ (2q^2-\gamma E)(\rho u) & \gamma E-(\gamma-1)(\rho u)^2-q^2 & -(\gamma-1)(\rho u)(\rho v) & -(\gamma-1)(\rho u)(\rho w) & \gamma(\rho u) \end{bmatrix} \tag{6.22}$$

$$A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ -(\rho u)(\rho v) & (\rho v) & (\rho u) & 0 & 0 \\ q^2-(\rho v)^2 & -(\gamma-1)(\rho u) & -(\gamma-3)(\rho v) & -(\gamma-1)(\rho w) & (\gamma-1) \\ -(\rho v)(\rho w) & 0 & (\rho w) & (\rho v) & 0 \\ (2q^2-\gamma E)(\rho v) & -(\gamma-1)(\rho u)(\rho v) & \gamma E-(\gamma-1)(\rho v)^2-q^2 & -(\gamma-1)(\rho v)(\rho w) & \gamma(\rho v) \end{bmatrix} \tag{6.23}$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -(\rho u)(\rho w) & (\rho w) & 0 & (\rho u) & 0 \\ -(\rho v)(\rho w) & 0 & (\rho w) & (\rho v) & 0 \\ q^2-(\rho w)^2 & -(\gamma-1)(\rho u) & -(\gamma-1)(\rho v) & -(\gamma-3)(\rho w) & (\gamma-1) \\ (2q^2-\gamma E)(\rho w) & -(\gamma-1)(\rho u)(\rho w) & -(\gamma-1)(\rho v)(\rho w) & \gamma E-(\gamma-1)(\rho w)^2-q^2 & \gamma(\rho w) \end{bmatrix} \tag{6.24}$$

with

$$q^2 = \frac{1}{2}(\gamma-1)((\rho u)^2+(\rho v)^2+(\rho w)^2).$$

Linearization of viscous terms

Viscous terms involve fluxes S_i that are expressed in terms of gradients of the velocity and temperature fields. We have to estimate terms of the form

$$\int_\Omega \frac{\partial(u_i - \Pi_h u_i)}{\partial x_j}\frac{\partial W_h^*}{\partial x_k}\,\mathrm{d}\Omega.$$

Spectral radius estimate: As the discrete adjoint W_h^*, that is computed and used in practice, is represented by a piece-wise linear function, some concerns have been raised in [Belme 2018] about the evaluation of these terms. Namely, [Belme 2018] argues that W_h^* should be considered as a piece-wise linear function for the error estimation and develops a specific error estimate (see Section 6.1.4) that states

$$\left|\int_\Omega \frac{\partial(u_i - \Pi_h u_i)}{\partial x_j}\frac{\partial \Pi_h W^*}{\partial x_k}\,\mathrm{d}\Omega\right| \le K_d \int_\Omega |r(H(W^*))|\,|u_i - \Pi_h u_i|\,\mathrm{d}\Omega,$$

where $r(H(W^*))$ is the spectral radius of the hessian of W^*, and K_d is a constant: $K_d = 3$ in 2D and $K_d = 6$ in 3D. In practice, we do not know W^* and $H(W^*)$ is reconstructed (estimated) from the discrete values of W_h^*.

Direct hessian estimate: In practice, as in the spectral radius analysis, the hessian $H(W^*)$ is still needed and reconstructed from discrete values. Instead of performing the analysis presented in Section 6.1.4, we can directly use the values of the reconstructed hessian to state

$$\begin{aligned}\left|\int_\Omega \frac{\partial(u_i-\Pi_h u_i)}{\partial x_j}\frac{\partial \Pi_h W^*}{\partial x_k}\,\mathrm{d}\Omega\right| &= \left|\int_\Omega (u_i-\Pi_h u_i)\frac{\partial^2 W^*}{\partial x_k \partial x_j}\,\mathrm{d}\Omega\right|,\\ &\leq \int_\Omega |H_{kj}(W^*)|\,|u_i-\Pi_h u_i|\,\mathrm{d}\Omega.\end{aligned}$$

Analysis in entropic variables (u,v,T) ***with spectral radius majoration:*** In practice, in 2D, we have:

$$\delta J^{vis} = \delta J^{vis}_{\rho u} + \delta J^{vis}_{\rho v} + \delta J^{vis}_{\rho E},$$

where $\delta J^{vis}_{\rho u}$, $\delta J^{vis}_{\rho v}$, $\delta J^{vis}_{\rho E}$ represent the contributions from:

- The x-component velocity equation:

$$\begin{aligned}\delta J^{vis}_{\rho u} \approx \int_\Omega \Big(&\Big(2(\mu+\mu_t)u_x - \frac{2}{3}(\mu+\mu_t)(u_x+v_y)\Big)\\ &-\Big(2(\mu+\mu_t)(\Pi_h u)_x - \frac{2}{3}(\mu+\mu_t)\Big((\Pi_h u)_x+(\Pi_h v)_y\Big)\Big)\Big)(\rho u)^*_x\\ &+\Big((\mu+\mu_t)(u_y+v_x) - (\mu+\mu_t)\Big((\Pi_h u)_y+(\Pi_h v)_x\Big)\Big)(\rho u)^*_y\,\mathrm{d}\Omega,\end{aligned} \tag{6.25}$$

- The y-component velocity equation:

$$\begin{aligned}\delta J^{vis}_{\rho v} \approx \int_\Omega \Big(&(\mu+\mu_t)(v_x+u_y) - (\mu+\mu_t)\Big((\Pi_h v)_x+(\Pi_h u)_y\Big)\Big)(\rho v)^*_x\\ &+\Big(\Big(2(\mu+\mu_t)v_y - \frac{2}{3}(\mu+\mu_t)(u_x+v_y)\Big)\\ &-\Big(2(\mu+\mu_t)(\Pi_h v)_y - \frac{2}{3}(\mu+\mu_t)\Big((\Pi_h u)_x+(\Pi_h v)_y\Big)\Big)\Big)(\rho v)^*_y\,\mathrm{d}\Omega,\end{aligned} \tag{6.26}$$

- The energy equation:

$$\begin{aligned}\delta J^{vis}_{\rho E} \approx &\int_\Omega \Big(\Big(u\tau_{xx}+v\tau_{xy}+(\lambda+\lambda_t)\mathcal{T}_x\Big) - \Big((\Pi_h u)\overline{\tau_{xx}}+(\Pi_h v)\overline{\tau_{xy}}+(\lambda+\lambda_t)(\Pi_h\mathcal{T})_x\Big)\Big)(\rho E)^*_x\\ +&\Big(\Big(u\tau_{xy}+v\tau_{yy}+(\lambda+\lambda_t)\mathcal{T}_y\Big) - \Big((\Pi_h u)\overline{\tau_{xy}}+(\Pi_h v)\overline{\tau_{yy}}+(\lambda+\lambda_t)(\Pi_h\mathcal{T})_y\Big)\Big)(\rho E)^*_y\,\mathrm{d}\Omega,\end{aligned} \tag{6.27}$$

  where we have noted for readability

$$\overline{\tau_{ij}} = (\mu+\mu_t)\left(\frac{\partial \Pi_h u_i}{\partial x_j}+\frac{\partial \Pi_h u_j}{\partial x_i}\right) - \frac{2}{3}(\mu+\mu_t)\frac{\partial \Pi_h u_k}{\partial x_k}\delta_{ij}.$$

Both $\delta J^{vis}_{\rho u}$ and $\delta J^{vis}_{\rho v}$ are linear with respect to the velocity gradients, so that we obtain directly with the spectral radius majoration,

$$\begin{aligned} |\delta J^{vis}_{\rho u}| &\approx (\mu+\mu_t)\Big|\int_\Omega \Big(2(u-\Pi_h u)_x - \frac{2}{3}\Big((u-\Pi_h u)_x + (v-\Pi_h v)_y\Big)\Big)(\rho u)^*_x \\ &\quad + \Big(\Big((u-\Pi_h u)_y + (v-\Pi_h v)_x\Big)\Big)(\rho u)^*_y \,\mathrm{d}\Omega\Big|, \\ &\leq \int_\Omega (\mu+\mu_t)\Big(7|u-\Pi_h u| + |v-\Pi_h v|\Big)\Big|r\Big(H\big((\rho u)^*\big)\Big)\Big|. \end{aligned} \tag{6.28}$$

and

$$|\delta J^{vis}_{\rho v}| \leq \int_\Omega (\mu+\mu_t)\Big(7|v-\Pi_h v| + |u-\Pi_h u|\Big)\Big|r\Big(H\big((\rho v)^*\big)\Big)\Big|. \tag{6.29}$$

The energy equation involves non-linear terms that need to be linearized. By writing $u_i = \Pi_h u_i + \delta_{u_i}$ with $\delta_{u_i} = u_i - \Pi_h u_i << \Pi_h u_i$, we can linearize the terms as

$$u_i\tau_{ji} - \Pi_h u_i \overline{\tau_{ji}} \approx (u_i - \Pi_h u_i)\tau_{ji} + u_i(\tau_{ji} - \overline{\tau_{ji}}),$$

so that we have

$$\begin{aligned} \delta J^{vis}_{\rho E} &\approx \int_\Omega \Big((u-\Pi_h u)\tau_{xx} + u(\tau_{xx} - \overline{\tau_{xx}}) + (v-\Pi_h v)\tau_{xy} \\ &\quad + v(\tau_{xy} - \overline{\tau_{xy}}) + (\lambda+\lambda_t)\Big(\mathcal{T}_x - (\Pi_h \mathcal{T})_x\Big)\Big)(\rho E)^*_x \\ &\quad + \Big((u-\Pi_h u)\tau_{xy} + u(\tau_{xy} - \overline{\tau_{xy}}) \\ &\quad + (v-\Pi_h v)\tau_{yy} + v(\tau_{yy} - \overline{\tau_{yy}}) + (\lambda+\lambda_t)\Big(\mathcal{T}_y - (\Pi_h \mathcal{T})_y\Big)\Big)(\rho E)^*_y \,\mathrm{d}\Omega. \end{aligned} \tag{6.30}$$

The terms of the form $(u_i - \Pi_h u_i)\tau_{ji}$ do not involve interpolation error on gradients anymore and are directly treated as

$$\Big|\int_\Omega \Big((u-\Pi_h u)\tau_{xx}\Big)(\rho E)^*_x \,\mathrm{d}\Omega\Big| \leq \int_\Omega \big|(\rho E)^*_x \tau_{xx}\big|\,|u-\Pi_h u|\,\mathrm{d}\Omega.$$

The other terms are treated with the spectral radius majoration and putting all errors together we obtain the following estimations.

In two dimensions ($K_d = 3$), we have the following error bound for h sufficiently small:

$$|\delta J^{vis}| \leq \int_\Omega G_u\,|u-\Pi_h u|\,d\Omega + \int_\Omega G_v\,|v-\Pi_h v|\,d\Omega + \int_\Omega G_T\,|T-\Pi_h T|\,d\Omega\,,$$

with the coefficients:

$$\begin{aligned} G_u &= (\mu+\mu_t)\left(7\,|\,r(H(W^*_{\rho u}))| + |\,r(H(W^*_{\rho v}))| + \Big(7|u|+|v|\Big)\,|\,r(H(W^*_{\rho E}))| + \frac{5}{3}|\omega_v|\right) \\ G_v &= (\mu+\mu_t)\left(|\,r(H(W^*_{\rho u}))| + 7\,|\,r(H(W^*_{\rho v}))| + \Big(|u|+7|v|\Big)\,|\,r(H(W^*_{\rho E}))| + \frac{5}{3}|\omega_u|\right) \\ G_T &= 3\,(\lambda+\lambda_t)\,|\,r(H(W^*_{\rho E}))| \end{aligned}$$

and the $\boldsymbol{\omega}$ vector defined by:

$$\nabla u_i \times \nabla W^*_{\rho E} = \omega_{u_i}.$$

In three dimensions ($K_d = 6$), we have the following error bound for h sufficiently small:

$$\begin{aligned} |\delta J^{vis}| \;&\leq\; \int_\Omega G_u \,|u - \Pi_h u|\;\mathrm{d}\Omega + \int_\Omega G_v \,|v - \Pi_h v|\;\mathrm{d}\Omega \\ &+\; \int_\Omega G_w \,|w - \Pi_h w|\;\mathrm{d}\Omega + \int_\Omega G_T \,|T - \Pi_h T|\;\mathrm{d}\Omega\,, \end{aligned}$$

with the coefficients:

$$\begin{aligned} G_u &= (\mu+\mu_t)\,\left(20\,|\,r(H(W^*_{\rho u}))| + 2\,|\,r(H(W^*_{\rho v}))| + 2\,|\,r(H(W^*_{\rho w}))|\right. \\ &+ \left.\Big(20|u| + 2|v| + 2|w|\Big)\,|\,r(H(W^*_{\rho E}))| + \frac{5}{3}|\omega_{w,y} - \omega_{v,z}|\right) \\ G_v &= (\mu+\mu_t)\,\left(2\,|\,r(H(W^*_{\rho u}))| + 20\,|\,r(H(W^*_{\rho v}))| + 2\,|\,r(H(W^*_{\rho w}))|\right. \\ &+ \left.\Big(2|u| + 20|v| + 2|w|\Big)\,|\,r(H(W^*_{\rho E}))| + \frac{5}{3}|\omega_{u,z} - \omega_{w,x}|\right) \\ G_w &= (\mu+\mu_t)\,\left(2\,|\,r(H(W^*_{\rho u}))| + 2\,|\,r(H(W^*_{\rho v}))| + 20\,|\,r(H(W^*_{\rho w}))|\right. \\ &+ \left.\Big(2|u| + 2|v| + 20|w|\Big)\,|\,r(H(W^*_{\rho E}))| + \frac{5}{3}|\omega_{v,x} - \omega_{u,y}|\right) \\ G_T &= 6\,(\lambda+\lambda_t)\,|\,r(H(W^*_{\rho E}))| \end{aligned}$$

and the $\boldsymbol{\omega}$ vector defined by:

$$\nabla u_i \times \nabla W^*_{\rho E} = \boldsymbol{\omega}_{u_i} = \left(\omega_{u_i,x},\, \omega_{u_i,y},\, \omega_{u_i,z}\right)^T\,.$$

Again, we observe that the error model is a sum of interpolation error in L^1-norm weighted by algebraic functions. In 2D, we have now 12 terms and in 3D 19 terms.

Analysis in entropic variables (u, v, T) ***with direct hessian majoration:*** If we perform the same analysis but use the direct hessian majoration instead of the spectral radius majoration we obtain the following estimate in 2D:

$$\begin{aligned} |\delta J^{vis}| \;\leq \int_\Omega \Bigg[\;& |u - \Pi_h u|\,(\mu+\mu_t)\left|\tfrac{4}{3}(\rho u)^*_{xx} + (\rho u)^*_{yy} + \tfrac{1}{3}(\rho v)^*_{xy} + \tfrac{4}{3}u(\rho E)^*_{xx} + \tfrac{1}{3}v(\rho E)^*_{xy} + u(\rho E)^*_{yy}\right| \\ + \;& |v - \Pi_h v|\,(\mu+\mu_t)\left|(\rho v)^*_{xx} + \tfrac{4}{3}(\rho v)^*_{yy} + \tfrac{1}{3}(\rho u)^*_{xy} + \tfrac{4}{3}v(\rho E)^*_{xx} + \tfrac{1}{3}u(\rho E)^*_{xy} + v(\rho E)^*_{yy}\right| \\ + \;& |T - \Pi_h T|\,(\lambda+\lambda_t)\left|(\rho E)^*_{xx} + (\rho E)^*_{yy}\right| \Bigg]\; d\Omega \end{aligned}$$

and in 3D:

$$
\begin{aligned}
|\delta J^{vis}| \quad \leq \int_\Omega \Bigg[\quad & |u-\Pi_h u|\,(\mu+\mu_t)\left|\tfrac{4}{3}(\rho u)^*_{xx}+(\rho u)^*_{yy}+(\rho u)^*_{zz}+\tfrac{1}{3}(\rho v)^*_{xy}+\tfrac{1}{3}(\rho w)^*_{xz}\right. \\
& \left. +\tfrac{4}{3}u(\rho E)^*_{xx}+u(\rho E)^*_{yy}+u(\rho E)^*_{zz}+\tfrac{1}{3}v(\rho E)^*_{xy}+\tfrac{1}{3}w(\rho E)^*_{xz}\right| \\
+ \quad & |v-\Pi_h v|\,(\mu+\mu_t)\left|\tfrac{1}{3}(\rho u)^*_{xy}+(\rho v)^*_{xx}+\tfrac{4}{3}(\rho v)^*_{yy}+(\rho v)^*_{zz}+\tfrac{1}{3}(\rho w)^*_{yz}\right. \\
& \left. +\tfrac{1}{3}u(\rho E)^*_{xy}+v(\rho E)^*_{xx}+\tfrac{4}{3}v(\rho E)^*_{yy}+v(\rho E)^*_{zz}+\tfrac{1}{3}w(\rho E)^*_{yz}\right| \\
+ \quad & |w-\Pi_h w|\,(\mu+\mu_t)\left|\tfrac{1}{3}(\rho u)^*_{xz}+\tfrac{1}{3}(\rho v)^*_{yz}+(\rho w)^*_{xx}+(\rho w)^*_{yy}+\tfrac{4}{3}(\rho w)^*_{zz}\right. \\
& \left. +\tfrac{1}{3}u(\rho E)^*_{xz}+\tfrac{1}{3}v(\rho E)^*_{yz}+w(\rho E)^*_{xx}+w(\rho E)^*_{yy}+\tfrac{4}{3}w(\rho E)^*_{zz}\right| \\
+ \quad & |T-\Pi_h T|\,(\lambda+\lambda_t)\left|(\rho E)^*_{xx}+(\rho E)^*_{yy}+(\rho E)^*_{zz}\right| \Bigg]\, d\Omega
\end{aligned}
$$

Here again, the major difference with the spectral radius analysis is that all weighting terms can be encompassed in the same absolute value. The majoration is thus more accurate which may lead to a sharper error estimate.

Analysis in conservative variables $(\rho, \rho u, \rho v, \rho E)$***:*** Using Relation (6.13), we can perform the same analysis as for convective fluxes, with the major difference that the viscous contributions $S_i(W)$ do not depend directly on the set of variable W but on its gradients. W being a vector, ∇W is a matrix, the differentiation of $S_i(W)$ with respect to ∇W is thus a third order tensor that we will avoid for readability. Instead, we will introduce the differentiation of S_i with respect to each derivative of W:

$$K_{i,j} = \frac{\partial S_i}{\partial W_{x_j}}.$$

The gradient operator being linear, we have

$$S_1(W) - S_1(\Pi_h W) \approx K_{1,1}\cdot(W-\Pi_h W)_x + K_{1,2}\cdot(W-\Pi_h W)_y + K_{1,3}\cdot(W-\Pi_h W)_z.$$

Thus, we obtain from Equation (6.15):

$$
\begin{aligned}
\delta J^{vis} \quad &\approx \quad \int_\Omega \Big(\big(S_1(\Pi_h W)_x + S_2(\Pi_h W)_y + S_3(\Pi_h W)_z \big) - \big(S_1(W)_x + S_2(W)_y + S_3(W)_z \big) \Big) \cdot W^*\, \mathrm{d}\Omega, \\
&\approx \quad \int_\Omega \Big(\sum_{i=1}^{3} \Big(\sum_{j=1}^{3} K_{i,j}\cdot(W-\Pi_h W)_{x_j} \Big)_{x_i} \Big) \cdot W^*\, \mathrm{d}\Omega,
\end{aligned}
\tag{6.31}
$$

and integrating by parts we obtain

$$\delta J^{vis} \approx \int_\Omega \sum_{i=1}^{3} \Big(\sum_{j=1}^{3} K_{i,j}\cdot(W-\Pi_h W)_{x_j} \Big) \cdot (W^*)_{x_i}\, \mathrm{d}\Omega. \tag{6.32}$$

Assuming $K_{i,j}$ to be constant, we can use either the spectral or direct hessian majoration. In practice, we do not express the $K_{i,j}$ matrices (that are sparse) but rather differentiate directly the equations as previously, with the difference that we seek an expression in conservatives variables. To do so, we use the fact that

$$(u_i)_{x_j} = \frac{1}{\rho}(\rho u_i)_{x_j} - \frac{(\rho u_i)}{\rho^2}\rho_{x_j},$$

and

$$T = \frac{\rho E}{\rho} - \frac{(\rho u)^2 + (\rho v)^2 + (\rho w)^2}{2\rho^2}.$$

Additionally, we use the following partial linearization

$$(\Pi_h u_i)_{x_j} = \frac{1}{\rho}(\Pi_h(\rho u_i))_{x_j} - \frac{(\rho u_i)}{\rho^2}(\Pi_h \rho)_{x_j}.$$

Hence, in 2D we obtain for the x-component velocity equation

$$\begin{aligned}
\delta J^{vis}_{\rho u} \approx & \int_\Omega (\mu+\mu_t)\left[2\left(\frac{1}{\rho}(\rho u)_x - \frac{\rho u}{\rho^2}\rho_x\right) - \frac{2}{3}\left(\frac{1}{\rho}(\rho u)_x - \frac{\rho u}{\rho^2}\rho_x + \frac{1}{\rho}(\rho v)_y - \frac{\rho v}{\rho^2}\rho_y\right)\right. \\
& -\left(2\left(\frac{1}{\rho}(\Pi_h(\rho u))_x - \frac{\rho u}{\rho^2}(\Pi_h\rho)_x\right)\right. \\
& \left.\left. -\frac{2}{3}\left(\frac{1}{\rho}(\Pi_h(\rho u))_x - \frac{\rho u}{\rho^2}(\Pi_h\rho)_x + \frac{1}{\rho}(\Pi_h(\rho v))_y - \frac{\rho v}{\rho^2}(\Pi_h\rho)_y\right)\right)\right](\rho u)^*_x \\
& +(\mu+\mu_t)\left[\left(\frac{1}{\rho}(\rho u)_y - \frac{\rho u}{\rho^2}\rho_y + \frac{1}{\rho}(\rho v)_x - \frac{\rho v}{\rho^2}\rho_x\right)\right. \\
& \left. -\left(\frac{1}{\rho}(\Pi_h(\rho u))_y - \frac{\rho u}{\rho^2}(\Pi_h\rho)_y + \frac{1}{\rho}(\Pi_h(\rho v))_x - \frac{\rho v}{\rho^2}(\Pi_h\rho)_x\right)\right](\rho u)^*_y \,\mathrm{d}\Omega, \\
= & \int_\Omega (\mu+\mu_t)\left[\frac{4}{3}\frac{1}{\rho}(\rho u - \Pi_h(\rho u))_x - \frac{4}{3}\frac{\rho u}{\rho^2}(\rho - \Pi_h\rho)_x\right. \\
& \left. -\frac{2}{3}\frac{1}{\rho}(\rho v - \Pi_h(\rho v))_y + \frac{2}{3}\frac{\rho v}{\rho^2}(\rho - \Pi_h\rho)_y\right](\rho u)^*_x \\
& +(\mu+\mu_t)\left[\left(\frac{1}{\rho}(\rho u - \Pi_h(\rho u))_y - \frac{\rho u}{\rho^2}(\rho - \Pi_h\rho)_y\right.\right. \\
& \left.\left. +\frac{1}{\rho}(\rho v - \Pi_h(\rho v))_x - \frac{\rho v}{\rho^2}(\rho - \Pi_h\rho)_x\right)\right](\rho u)^*_y \,\mathrm{d}\Omega,
\end{aligned}$$

and similarly for the other equations. Putting all errors together, we integrate by parts to use a direct hessian majoration. To do so, we assume all coefficient in front of any derivative to be constant to ease the math, although it would be possible to take them into account too. We obtain in 2D:

$$\begin{aligned}
|\delta J^{vis}| \approx & \int_\Omega |\rho - \Pi_h\rho|\,|G_\rho| + |\rho u - \Pi_h(\rho u)|\,|G_{\rho u}| \\
+ & \ |\rho v - \Pi_h(\rho v)|\,|G_{\rho v}| + |\rho E - \Pi_h(\rho E)|\,|G_{\rho E}|\,\mathrm{d}\Omega,
\end{aligned} \tag{6.33}$$

with

$$\begin{aligned}
G_{\rho u} = & \ \tfrac{\mu+\mu_t}{3\rho}\left(4(\rho u)^*_{xx} + 3(\rho u)^*_{yy} + (\rho v)^*_{xy} + 4u(\rho E)^*_{xx} + 3u(\rho E)^*_{yy} + v(\rho E)^*_{xy}\right) \\
& -u\tfrac{\lambda+\lambda_t}{\rho}\left((\rho E)^*_{xx} + (\rho E)^*_{yy}\right) \\
G_{\rho v} = & \ \tfrac{\mu+\mu_t}{3\rho}\left((\rho u)^*_{xy} + 3(\rho v)^*_{xx} + 4(\rho E)^*_{yy} + u(\rho E)^*_{xy} + 3v(\rho E)^*_{xx} + 4v(\rho E)^*_{yy}\right) \\
& -v\tfrac{\lambda+\lambda_t}{\rho}\left((\rho E)^*_{xx} + (\rho E)^*_{yy}\right) \\
G_{\rho E} = & \ \tfrac{\lambda+\lambda_t}{\rho}\left((\rho E)^*_{xx} + (\rho E)^*_{yy}\right) \\
G_\rho = & \ -uG_{\rho u} - vG_{\rho v} - EG_{\rho E}
\end{aligned}$$

and in 3D

$$
\begin{aligned}
|\delta J^{vis}| \approx & \int_\Omega |\rho - \Pi_h\rho|\,|G_\rho| + |\rho u - \Pi_h(\rho u)|\,|G_{\rho u}| + |\rho v - \Pi_h(\rho v)|\,|G_{\rho v}| \\
+ & |\rho w - \Pi_h(\rho w)|\,|G_{\rho w}| + |\rho E - \Pi_h(\rho E)|\,|G_{\rho E}|\,\mathrm{d}\Omega,
\end{aligned}
\tag{6.34}
$$

with

$$
\begin{aligned}
G_{\rho u} = & \tfrac{\mu+\mu_t}{3\rho}\Big(4(\rho u)^*_{xx} + 3(\rho u)^*_{yy} + 3(\rho u)^*_{zz} + (\rho v)^*_{xy} + (\rho w)^*_{xz} + 4u(\rho E)^*_{xx} + 3u(\rho E)^*_{yy} \\
& + 3u(\rho E)^*_{zz} + v(\rho E)^*_{xy} + w(\rho E)^*_{xz}\Big) - u\tfrac{\lambda+\lambda_t}{\rho}\Big((\rho E)^*_{xx} + (\rho E)^*_{yy} + (\rho E)^*_{zz}\Big) \\
G_{\rho v} = & \tfrac{\mu+\mu_t}{3\rho}\Big((\rho u)^*_{xy} + 3(\rho v)^*_{xx} + 4(\rho v)^*_{yy} + 3(\rho v)^*_{zz} + (\rho w)^*_{yz} + u(\rho E)^*_{xy} + 3v(\rho E)^*_{xx} \\
& + 4v(\rho E)^*_{yy} + 3v(\rho E)^*_{zz} + w(\rho E)^*_{yz}\Big) - v\tfrac{\lambda+\lambda_t}{\rho}\Big((\rho E)^*_{xx} + (\rho E)^*_{yy} + (\rho E)^*_{zz}\Big) \\
G_{\rho w} = & \tfrac{\mu+\mu_t}{3\rho}\Big((\rho u)^*_{xz} + (\rho v)^*_{yz} + 3(\rho w)^*_{xx} + 3(\rho w)^*_{yy} + 4(\rho w)^*_{zz} + u(\rho E)^*_{xz} + v(\rho E)^*_{yz} \\
& + 3w(\rho E)^*_{xx} + 3w(\rho E)^*_{yy} + 4w(\rho E)^*_{zz}\Big) - w\tfrac{\lambda+\lambda_t}{\rho}\Big((\rho E)^*_{xx} + (\rho E)^*_{yy} + (\rho E)^*_{zz}\Big) \\
G_{\rho E} = & \tfrac{\lambda+\lambda_t}{\rho}\Big((\rho E)^*_{xx} + (\rho E)^*_{yy} + (\rho E)^*_{zz}\Big) \\
G_\rho = & -uG_{\rho u} - vG_{\rho v} - wG_{\rho w} - EG_{\rho E}
\end{aligned}
$$

6.1.3 RANS goal-oriented error estimation: Case of Spalart-Allmaras equations

Following the same approach, we extend the adjoint-based error estimation to the RANS equations and detail the contributions of the turbulence to the error model.

Mean flow: The expressions developed in the previous section already include the contribution of the turbulent viscosity to the main flow. But as we shown in our analysis in Section 4.1.1, the turbulent viscosity, contrary to the laminar viscosity, varies from several orders of magnitude in the domain. Its discretization has thus an impact on the precision of the solution and should be taken into account in the error analysis. This rises error terms on the turbulent viscosity and thermal conductivity that can be treated as standard weighted interpolation errors

$$
\begin{aligned}
&\left|\int_\Omega -\frac{\partial}{\partial x_i}\left((\mu_t - \Pi_h\mu_t)\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)\right)W^*\right| \\
&\qquad \le \int_\Omega |\mu_t - \Pi_h\mu_t|\left|\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)\frac{\partial W^*}{\partial x_i}\right|.
\end{aligned}
$$

Turbulent equation: We recall the turbulent equation

$$
\frac{\partial\rho\tilde\nu}{\partial t} + \underbrace{\mathbf{u}\cdot\nabla\rho\tilde\nu}_{convection} = \underbrace{c_{b1}\tilde S\rho\tilde\nu}_{production} - \underbrace{c_{w1}f_w\rho\left(\frac{\tilde\nu}{d}\right)^2}_{destruction} + \underbrace{\frac{\rho}{\sigma}\nabla\cdot((\nu+\tilde\nu)\nabla\tilde\nu)}_{dissipation} + \underbrace{\frac{c_{b2}\rho}{\sigma}\|\nabla\tilde\nu\|^2}_{diffusion}.
$$

The different terms can be treated either separately, or with a full differentiation. Theoretically, we should take into account the dependency of terms f_w, f_1, ... with respect to the variables. However, we present here a simplified version of the linearization where we ignore these terms. A broader analysis of the influence of these terms should be conducted to determine the optimal

approach. We present here separately the analysis of the advection, dissipation, diffusion and production/destruction terms of the Spalart-Allmaras equation in primitive variables (ρ, u, v, p) and turbulent variable $\hat{\nu}$, but the same analysis can be done with any set of variables. As these different terms involve interpolation error on the considered variables they can be summed together. Note that this equation is weighted by the turbulent adjoint.

The convection term can be treated as previously as

$$\left| \int_\Omega (\mathbf{u} \cdot \nabla \rho\tilde{\nu} - (\Pi_h \mathbf{u}) \cdot \nabla(\Pi_h \rho\tilde{\nu})) W^*_{\rho\tilde{\nu}} \mathrm{d}\Omega \right| \leq \int_\Omega |\mathbf{u} - \Pi_h \mathbf{u}| \cdot |(\nabla \rho\tilde{\nu}) W^*_{\rho\tilde{\nu}}| \mathrm{d}\Omega + \int_\Omega \rho |\tilde{\nu} - \Pi_h(\tilde{\nu})| |\nabla \cdot (W^*_{\rho\tilde{\nu}} \mathbf{u})| \mathrm{d}\Omega.$$

The dissipation term can be treated with either one of the two analysis proposed earlier. For simplicity we use here a partial differentiation, assuming ρ and $(\nu + \tilde{\nu})$ to be constant. Nevertheless, these could be differentiated as well, adding interpolation and gradient terms. We obtain

$$\left| \int_\Omega \Big[\frac{\rho}{\sigma} \nabla \cdot ((\nu + \tilde{\nu}) \nabla \tilde{\nu}) - \frac{\rho}{\sigma} \nabla \cdot ((\nu + \tilde{\nu}) \nabla (\Pi_h \tilde{\nu})) \Big] W^*_{\rho\tilde{\nu}} \mathrm{d}\Omega \right| \leq K_d \int_\Omega \left| r(H(W^*_{\rho\tilde{\nu}})) (\frac{\rho}{\sigma}(\nu + \tilde{\nu})) \right| |\tilde{\nu} - \Pi_h \tilde{\nu}| \mathrm{d}\Omega.$$

or with the direct hessian majoration

$$\left| \int_\Omega \Big[\frac{\rho}{\sigma} \nabla \cdot ((\nu + \tilde{\nu}) \nabla \tilde{\nu}) - \frac{\rho}{\sigma} \nabla \cdot ((\nu + \tilde{\nu}) \nabla (\Pi_h \tilde{\nu})) \Big] W^*_{\rho\tilde{\nu}} \mathrm{d}\Omega \right| \leq \int_\Omega \frac{\rho}{\sigma} (\nu + \tilde{\nu}) \left| \nabla \cdot \nabla W^*_{\rho\tilde{\nu}} \right| |\tilde{\nu} - \Pi_h \tilde{\nu}| \mathrm{d}\Omega.$$

The diffusion term can be directly differentiated (here again we neglect the variations of ρ and different other linearizations can be considered) and yield the following gradient term

$$\left| \int_\Omega \Big(\frac{c_{b2}\rho}{\sigma} \|\nabla \tilde{\nu}\|^2 - \frac{c_{b2}\rho}{\sigma} \|\nabla (\Pi_h \tilde{\nu})\|^2 \Big) W^*_{\rho\tilde{\nu}} \mathrm{d}\Omega \right| \leq \int_\Omega 2 \frac{c_{b2}\rho}{\sigma} |\nabla W^*_{\rho\tilde{\nu}} \cdot \nabla \tilde{\nu}| \, |\tilde{\nu} - \Pi_h \tilde{\nu}| \mathrm{d}\Omega.$$

Finally, to treat the production and dissipation terms we consider S, ρ and f_w to be constant. This is motivated by the fact that these terms are strongly non-linear and their direct differentiation could lead to spikes in the metric and in turn in the generated mesh. However, a more extensive study of these terms would be profitable. So far, we treat them as

$$\left| \int_\Omega \left(c_{b1} \tilde{S} \rho \tilde{\nu} - c_{w1} f_w \rho \left(\frac{\tilde{\nu}}{d} \right)^2 - c_{b1} \tilde{S} \rho (\Pi_h \tilde{\nu}) + c_{w1} f_w \rho \left(\frac{\Pi_h \tilde{\nu}}{d} \right)^2 \right) W^*_{\rho\tilde{\nu}} \mathrm{d}\Omega \right| \leq \int_\Omega \left| \Big(c_{b1} \tilde{S} \rho - 2 c_{w1} f_w \rho \frac{\tilde{\nu}}{d^2} \Big) W^*_{\rho\tilde{\nu}} \right| |\tilde{\nu} - \Pi_h \tilde{\nu}| \mathrm{d}\Omega$$

6.1.4 Second derivatives error estimation

We detail here the mathematical analysis of the errors induced by second order derivative terms to obtain a majoration in the continuous mesh context. Let us first recall the general adjoint analysis in a continuous mesh context. We derived an expression for the functional error depending on the discrete adjoint and the interpolation error. Although it is written in a continuous form that can be generalized as,

$$\mathcal{E}_J = \int_\Omega \varepsilon(W - \Pi_h W, W_h^*)\,\mathrm{d}\Omega,$$

this expression is implicitly written on a mesh as it involves the interpolation error $W - \Pi_h W$. Hence, it can be rewritten

$$\mathcal{E}_J = \sum_K \int_K \varepsilon(W - \Pi_h W, W_h^*)\,\mathrm{d}K.$$

In this expression, the discrete adjoint W_h^* depends on the considered mesh, however, as it converges to the solution of the continuous adjoint problem W^*, we assume W_h^* to be the projection of the continuous adjoint: $W_h^* = \Pi_h(W^*)$.

The goal is then to minimise $\mathcal{E}_J$ with respect to the mesh. In the continuous mesh context, elements are generated as unit in the metric space, the semi discrete mesh optimization problem can thus be replaced by a minimization problem on the continuous metric field. However, to do so, we have to express $\mathcal{E}_J$ with respect to the metric $\mathcal{M}$. The idea is to express the integral of the error over each element

$$\int_{\mathcal{M}(K)=1} \varepsilon(W - \Pi_h W, W_h^*)\,\mathrm{d}K = \epsilon(W, W^*, \mathcal{M})|_K,$$

and re-express it as the integral of a constant over each element

$$\int_{\mathcal{M}(K)=1} \varepsilon(W - \Pi_h W, W_h^*)\,\mathrm{d}K = V_{unit} \int_{\mathcal{K}=1} \epsilon(W, W^*, \mathcal{M})|_K \,\det(\mathcal{M})^{\frac{1}{2}}\,\mathrm{d}K,$$

where we used the fact that the physical volume of the element K unit in the metric $\mathcal{M}$ expresses as

$$|K| = V_{unit}\det(\mathcal{M})^{-\frac{1}{2}}.$$

We thus rewrite $\mathcal{E}_J$ as

$$\mathcal{E}_J = V_{unit} \sum_K \int_K \epsilon(W, W^*, \mathcal{M})|_K \,\det(\mathcal{M})^{\frac{1}{2}}\,\mathrm{d}K,$$

and as the expression of $\epsilon(W, W^*, \mathcal{M})|_K$ is valid for any element size, we approach its integral by the integral of the continuous expression of ϵ, so that

$$\mathcal{E}_J \approx V_{unit} \sum_K \int_K \epsilon(W, W^*, \mathcal{M}) \,\det(\mathcal{M})^{\frac{1}{2}}\,\mathrm{d}K = V_{unit} \int_\Omega \epsilon(W, W^*, \mathcal{M}) \,\det(\mathcal{M})^{\frac{1}{2}}\,\mathrm{d}\Omega.$$

The Navier-Stokes/RANS equations involve error terms with second derivatives. If we choose not to directly use the reconstructed hessian of the discrete adjoint, we need a local error model expressing $\epsilon(W, W^*, \mathcal{M})$ with respect to the metric $\mathcal{M}$. We analyze here the contribution of the terms occurring from second derivatives, of the form:

$$\varepsilon = \int_\Omega \nabla(W - \Pi_h W)\nabla W_h^* \, \mathrm{d}\Omega,$$

on triangles. In particular, as the error ε decomposes as a sum over the elements

$$\varepsilon = \sum_K \int_K \nabla(W - \Pi_h W)\nabla W_h^* \, \mathrm{d}K,$$

we seek an expression of $\int_K \nabla(W - \Pi_h W)\nabla W_h^* \, \mathrm{d}K = |K|\epsilon(W, W^*, \mathcal{M})$ where K is unit in the metric $\mathcal{M}$. To this end we assume:

- The solution W can be locally represented by a quadratic field.
- The discrete adjoint W_h^* converges toward the continuous adjoint W^* so that W_h^* on each element K can be assumed to be the projection of W^*, W^* being represented by a quadratic field.
- The local variations of the metric field are negligible and $\mathcal{M}$ is assumed to be locally constant.

We study now in more details the behaviour in 2D of the local error

$$\int_K \nabla(W - \Pi_h W)\nabla W_h^* \, \mathrm{d}K.$$

To do so, we approximate $W - \Pi_h W$ with a quadratic function and $W^* = W_0^* + \nabla W_0^* \cdot X + \frac{1}{2} X \cdot H_{W^*} \cdot X$ by its second order Taylor expansion. We recall that the element K considered is unit in the metric $\mathcal{M}$ so that the transformation $T^{-1} = \mathcal{M}^{\frac{1}{2}}$ maps the triangle K onto a unit triangle

$$\hat{K}(\theta) = \frac{2}{3}\Big[(\cos(\theta), \sin(\theta)), (\cos(\theta + \frac{\pi}{3}), \sin(\theta + \frac{\pi}{3})), (\cos(\theta + \frac{2\pi}{3}), \sin(\theta + \frac{2\pi}{3}))\Big].$$

For the purpose of this study, we will assume that we are given a transformation $T = \mathcal{M}^{-\frac{1}{2}}$ and work in the unit triangles $\hat{K}(\theta)$ with the transformation

$$\int_K \nabla(W - \Pi_h W)\nabla W_h^* \, \mathrm{d}K = \int_{\hat{K}} T^{-1}\hat{\nabla}(W - \Pi_h W)T^{-1}\hat{\nabla} W_h^* \det(T) \, \mathrm{d}\hat{K}.$$

We recall that in the unit space, the gradients and hessians become

$$\widehat{\nabla} W = T\nabla W, \tag{6.35}$$

$$\widehat{H}_W = T^T H(W) T. \tag{6.36}$$

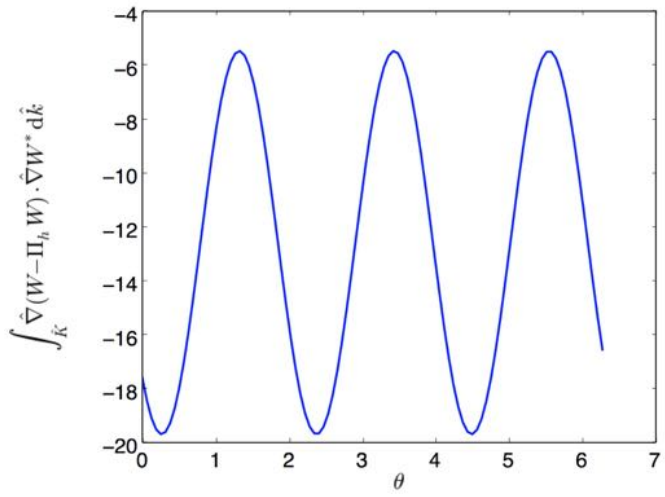


Figure 6.1: Dependency of the laplacian based error with respect to the orientation of the triangle.

Given $\hat{H}_W$, $\hat{H}_{W^*}$ and $\hat{\nabla}W^*$, we can compute analytically $\int_{\hat{K}} T^{-1}\hat{\nabla}(W - \Pi_h W)T^{-1}\hat{\nabla}W_h^* \det(T)\,\mathrm{d}\hat{K}$. To do so, we use a second mapping onto the standard finite element triangle $[(0,0),(1,0),(0,1)]$ where $W - \Pi_h W$ decomposes on the three basis functions

$$W - \Pi_h W = \frac{W_{xx}}{2}x(x-1) + \frac{W_{yy}}{2}y(y-1) + W_{xy}xy.$$

We obtain the following dependency of the error with respect to θ as shown in Figure 6.1.

Contrary to the traditional interpolation error, this one depends on the orientation of the element considered. A further analysis of this error shows that the mean value of the error depends only on the hessian of W^*, H_{W^*}. Meanwhile the amplitude of the oscillations around this mean value depends only on the constant gradient of W^* in its Taylor expansion ∇W_0^*, and this amplitude is null if $\nabla W_0^* = 0$. Moreover, one can note that the error is $\frac{2\pi}{3}$ periodic and seems to verify $\epsilon(\theta + \frac{\pi}{2}) - \bar{\epsilon} = -(\epsilon(\theta) - \bar{\epsilon})$ (symmetric with respect to its mean value).

Under the assumption that $\mathcal{M}$ is locally constant we can consider the element K and one of its neighbours that should be unit in the metric too. This neighbour element will be subject to the same hessians and gradients values of W and W^* so that it will have the same error function ϵ, but rotated by $\frac{\pi}{2}$ or π so that the sum of their error would be exactly the mean value. This suggest that the error on triangles considered by pairs should not depend on the orientation and only on the metric field. Following this idea, we introduce two different estimations of ϵ.

Spectral radius majoration

An estimation for isotropic elements was proposed in [Belme 2018]. The idea is to express the error edge-wise, splitting the integral over the domain as a sum of integrals over each triangle

and to integrate by parts to obtain

$$\begin{aligned}\int_\Omega \nabla(W - \Pi_h W)\nabla W_h^* \mathrm{d}\Omega &= \sum_K \int_K \nabla(W - \Pi_h W)\nabla W_h^* \mathrm{d}K \\ &= \sum_K \int_{\partial K}(W - \Pi_h W)\nabla W_h^* \cdot \mathbf{n}\mathrm{d}l \\ &= \sum_e \int_e (W - \Pi_h W)[\nabla W_h^*] \cdot \mathbf{n}\mathrm{d}l.\end{aligned} \tag{6.37}$$

where $[\nabla W_h^*]$ is the difference (jump) of the gradients between the two elements sharing the edge.

As W_h^* is assumed to represent a continuous W^*, these gradients are equal to the gradients of W^* at the center of gravity of each triangle, so that

$$[\nabla W_h^*] \cdot \mathbf{n} \approx (\mathbf{n} \cdot H_{W^*} \cdot \mathbf{n})h,$$

with h the height of the triangle. We insist on the fact that this approximation is only valid on isosceles triangles!! This expression is then bounded as

$$|(\mathbf{n} \cdot H_{W^*} \cdot \mathbf{n})h| \leq h\ r(H_{W^*}),$$

with $r(H_{W^*})$ the spectral radius of H_{W^*}.

With the aforementioned bound, we can bound the sum edge-wise

$$\sum_e \int_e (W - \Pi_h W)[\nabla W_h^*] \cdot \mathbf{n}\mathrm{d}l \leq \sum_e h\, r(H_{W^*}) \int_e (W - \Pi_h W)\mathrm{d}l, \tag{6.38}$$

but it still requires to be reshaped in a continuous form. To do so, in [Belme 2018], each edge e is associated to a "diamond" domain D_e composed of the two small triangles joining the vertices of the edge and the center of gravity of each triangle sharing edge e (see Figure 6.2). The interpolation error over the edge e and the domain D_e are alleged to be related as

$$\frac{1}{|e|}\int_e (W - \Pi_h W)\mathrm{d}l \approx \frac{1}{|D_e|}\int_{D_e} (W - \Pi_h W)\mathrm{d}\Omega.$$

As $|D_e| = h|e|$, Equation (6.38) can be rewritten as

$$\sum_e h\ r(H_{W^*}) \int_e (W - \Pi_h W)\mathrm{d}l \approx \sum_e r(H_{W^*}) \int_{D_e} (W - \Pi_h W)\mathrm{d}\Omega, \tag{6.39}$$

and noticing that the union $\cup_e D_e$ covers the whole domain, we obtain the weighted interpolation error

$$\int_\Omega \nabla(W - \Pi_h W)\nabla W_h^* \mathrm{d}\Omega \leq \sum_K \int_K r(H_{W^*})|(W - \Pi_h W)|\mathrm{d}K. \tag{6.40}$$

Analytical analysis

Assuming the metric to be locally constant, as the mesh generated is unit in the metric space, there exist a linear transformation $T = \mathcal{M}^{-\frac{1}{2}}$ so that the considered triangle and its neighbours are the image of unit triangles by T, *i.e.* $K = T(\hat{K})$. As $\mathcal{M}^{-1} = T^T T$, T is defined up to a rotation and a translation (that conserve the length).

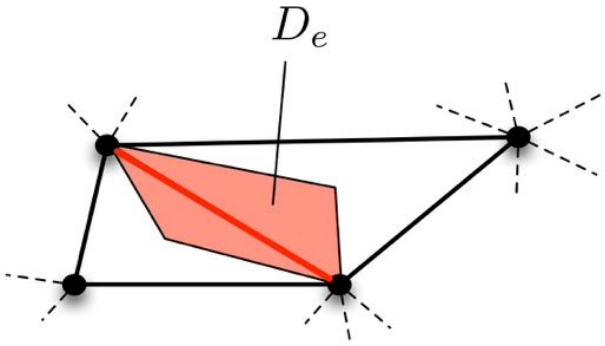


Figure 6.2: Diamond domain D_e associated with an edge.

We thus rewrite the integral over each triangle as an integral over a unit triangle as

$$\varepsilon = \sum_K \int_{T^{-1}(K)} T^{-1}\hat{\nabla}(W - \Pi_h W) \cdot T^{-1}\hat{\nabla}W_h^* \det(T) \mathrm{d}\hat{K}.$$

We then reshape the interpolation error integrating by part as a sum of interpolation errors on edges in the unit space:

$$\varepsilon = \sum_K \det(T) \int_{\partial T^{-1}(K)} T^{-T}T^{-1}([\hat{\nabla}W_h^*] \cdot \mathbf{n})(W - \Pi_h W)\mathrm{d}\hat{l},$$

with $[\hat{\nabla}W]$ the difference (jump) of gradients on both side of the edges. We thus consider the pairs of triangles on both sides of edges $\hat{e}$.

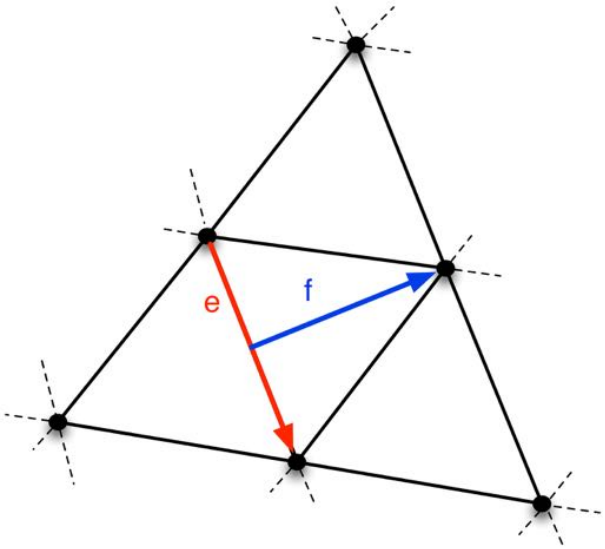


Figure 6.3: Orthogonal basis in the unit reference triangle $\hat{K}$.

We choose the vector basis $(\mathbf{e}, \mathbf{f})$ with $\mathbf{e}$ along the edge $\hat{e}$ and $\mathbf{f}$ orthogonal to $\mathbf{e}$, along the height of the triangle, as shown in Figure 6.3. As we assume the discrete adjoint to be

the linear projection of a continuous adjoint represented by a quadratic field $W_h^* = \Pi_h W^*$, the gradient of W_h^*, constant on each element, is equal to the gradient of W^* at the center of gravity $\hat{\nabla}(\Pi_h W^*)|_K = \hat{\nabla} W^*(G_K)$. We deduce

$$[\hat{\nabla} W_h^*] = \hat{\nabla}(\Pi_h W^*)|_{\hat{K}_1} - \hat{\nabla}(\Pi_h W^*)|_{\hat{K}_2} = \hat{\nabla} W^*(G_{\hat{e}} + \mathbf{f}/3) - \hat{\nabla} W^*(G_{\hat{e}} - \mathbf{f}/3) = \frac{2}{3}\widehat{H}_{W^*}\mathbf{f},$$

with $G_{\hat{e}}$ the midle of the edge $\hat{e}$. The gradient variation being constant on the edge, we can rewrite the interpolation error on each edge as

$$\begin{aligned}\int_{\hat{e}} T^{-T}T^{-1}([\hat{\nabla} W_h^*]\cdot\mathbf{n})(W-\Pi_h W)\mathrm{d}\hat{l} =& \mathcal{M}[\hat{\nabla} W_h^*]\cdot\mathbf{n}\int_{\hat{e}}(W-\Pi_h W)\mathrm{d}\hat{l}\\ =& \frac{4}{3\sqrt{3}}(\mathbf{f}\mathcal{M}\hat{H}_{W^*}\mathbf{f})\,\frac{1}{6}(\mathbf{e}\hat{H}_W\mathbf{e}),\end{aligned}$$

as $\mathbf{n} = \frac{2}{\sqrt{3}}\mathbf{f}$. This expression still depends on the orientation of the edge. We need to add in the process the information that edges form triangles, we thus rewrite this expression for the three edges of each triangle. The triangle being unit, we know the three couples (edge,normal) as

$$(\mathbf{e},\mathbf{f}),\quad (\frac{\mathbf{e}}{2}+\mathbf{f},\frac{3\mathbf{e}}{4}-\frac{\mathbf{f}}{2}),\quad (\mathbf{f}-\frac{\mathbf{e}}{2},\frac{3\mathbf{e}}{4}+\frac{\mathbf{f}}{2})$$

so that

$$\begin{aligned}\tfrac{2}{9\sqrt{3}}\textstyle\int_{\hat{K}}\hat{\nabla}(W-\Pi_h W)\cdot\mathcal{M}\hat{\nabla} W_h^*\mathrm{d}\hat{K} =& \ (\mathbf{e}\hat{H}_W\mathbf{e})(\mathbf{f}\mathcal{M}\hat{H}_{W^*}\mathbf{f})\\ +& \ ((\tfrac{\mathbf{e}}{2}+\mathbf{f})\hat{H}_W(\tfrac{\mathbf{e}}{2}+\mathbf{f}))((\tfrac{3\mathbf{e}}{4}-\tfrac{\mathbf{f}}{2})\mathcal{M}\hat{H}_{W^*}(\tfrac{3\mathbf{e}}{4}-\tfrac{\mathbf{f}}{2}))\\ +& \ ((\mathbf{f}-\tfrac{\mathbf{e}}{2})\hat{H}_W(\mathbf{f}-\tfrac{\mathbf{e}}{2}))((\tfrac{\mathbf{f}}{2}+\tfrac{3\mathbf{e}}{4})\mathcal{M}\hat{H}_{W^*}(\tfrac{\mathbf{f}}{2}+\tfrac{3\mathbf{e}}{4})).\end{aligned} \tag{6.41}$$

Reorganizing the terms we obtain

$$\begin{aligned}\tfrac{2}{9\sqrt{3}}\textstyle\int_{\hat{K}}\hat{\nabla}(W-\Pi_h W)\cdot\mathcal{M}\hat{\nabla} W_h^*\mathrm{d}\hat{K} =& \ \tfrac{9}{24}(\mathbf{e}\hat{H}_W\mathbf{e})(\mathbf{e}\mathcal{M}\hat{H}_{W^*}\mathbf{e}) + \tfrac{9}{8}(\mathbf{e}\hat{H}_W\mathbf{e})(\mathbf{f}\mathcal{M}\hat{H}_{W^*}\mathbf{f})\\ -& \ \tfrac{3}{2}(\mathbf{e}\hat{H}_W\mathbf{f})(\mathbf{e}\mathcal{M}\hat{H}_{W^*}\mathbf{f})\\ +& \ \tfrac{9}{8}(\mathbf{f}\hat{H}_W\mathbf{f})(\mathbf{e}\mathcal{M}\hat{H}_{W^*}\mathbf{e}) + \tfrac{1}{2}(\mathbf{f}\hat{H}_W\mathbf{f})(\mathbf{f}\mathcal{M}\hat{H}_{W^*}\mathbf{f}),\end{aligned} \tag{6.42}$$

which *a priori* still depends on the orientation of $\mathbf{e}$. In order to further investigate this expression, we note that the terms can be written up to a rotation as

$$((R\mathbf{e})\hat{H}_W(R\mathbf{e}))((R\mathbf{f})\mathcal{M}\hat{H}_{W^*}(R\mathbf{f})) = (\mathbf{e}[R^T\hat{H}_W R]\mathbf{e})(\mathbf{f}[R^T\mathcal{M}\hat{H}_{W^*}R]\mathbf{f}),$$

which only changes the hessians, so that we write it for

$$(\mathbf{e},\mathbf{f}) = \left(\begin{pmatrix}1\\0\end{pmatrix}, \frac{\sqrt{3}}{2}\begin{pmatrix}0\\1\end{pmatrix}\right).$$

Equation (6.42) thus becomes

$$\begin{aligned}\tfrac{2}{9\sqrt{3}}\textstyle\int_{\hat{K}}\hat{\nabla}(W-\Pi_h W)\cdot\mathcal{M}\hat{\nabla} W_h^*\mathrm{d}\hat{K} =& \ \tfrac{9}{24}(\hat{H}_W)_{xx}(\mathcal{M}\hat{H}_{W^*})_{xx} + \tfrac{3}{4}\tfrac{9}{8}(\hat{H}_W)_{xx}(\mathcal{M}\hat{H}_{W^*})_{yy}\\ -& \ \tfrac{3}{2}\tfrac{3}{4}(\hat{H}_W)_{xy}(\mathcal{M}\hat{H}_{W^*})_{xy}\\ +& \ \tfrac{9}{8}\tfrac{3}{4}(\hat{H}_W)_{yy}(\mathcal{M}\hat{H}_{W^*})_{xx} + \tfrac{1}{2}\tfrac{9}{16}(\hat{H}_W)_{yy}(\mathcal{M}\hat{H}_{W^*})_{yy},\end{aligned} \tag{6.43}$$

or

$$\begin{aligned}\int_{\hat{K}} \hat{\nabla}(W-\Pi_h W)\cdot\mathcal{M}\hat{\nabla}W_h^*\mathrm{d}\hat{K} = \;& \tfrac{27\sqrt{3}}{16}\Big[((\hat{H}_W)_{xx}(\mathcal{M}\hat{H}_{W^*})_{xx}+(\hat{H}_W)_{yy}(\mathcal{M}\hat{H}_{W^*})_{yy})\\ +\;& 3((\hat{H}_W)_{xx}(\mathcal{M}\hat{H}_{W^*})_{yy}+(\hat{H}_W)_{yy}(\mathcal{M}\hat{H}_{W^*})_{xx})\\ -\;& 4(\hat{H}_W)_{xy}(\mathcal{M}\hat{H}_{W^*})_{xy}\Big].\end{aligned} \tag{6.44}$$

Knowing that traditional interpolation error involves the trace of $\hat{H}_W$, we look for a similar expression and note that

$$\int_{\hat{K}} \hat{\nabla}(W-\Pi_h W)\cdot\mathcal{M}\hat{\nabla}W_h^*\mathrm{d}\hat{K} = \tfrac{27\sqrt{3}}{16}\left[3\,\mathrm{trace}(\hat{H}_W)\,\mathrm{trace}(\mathcal{M}\hat{H}_{W^*})-2\,\mathrm{trace}(\hat{H}_W\mathcal{M}\hat{H}_{W^*})\right]. \tag{6.45}$$

Eventhough this expression has been obtained for a specific value of $(\mathbf{e},\mathbf{f})$, it does not depend on the orientation due to the traces. Basically, we have,

$$\mathrm{trace}(R^T\hat{H}_W R R^T\mathcal{M}\hat{H}_{W^*}R)=\mathrm{trace}(\hat{H}_W\mathcal{M}\hat{H}_{W^*}RR^T)=\mathrm{trace}(\hat{H}_W\mathcal{M}\hat{H}_{W^*}).$$

This proves that $\int_{\hat{K}}\mathcal{M}\hat{\nabla}(W-\Pi_h W)\cdot\hat{\nabla}W_h^*\mathrm{d}\hat{K}$ only depends on the metric as

$$\begin{aligned}\int_K\;& \nabla(W-\Pi_h W)\cdot\nabla W_h^*\mathrm{d}K =\\ &\det(\mathcal{M}^{-\frac{1}{2}})\frac{27\sqrt{3}}{16}\left[3\,\mathrm{trace}(\mathcal{M}^{-\frac{1}{2}}H_W\mathcal{M}^{-\frac{1}{2}})\,\mathrm{trace}(H_{W^*})-2\,\mathrm{trace}(\mathcal{M}^{-\frac{1}{2}}H_W H_{W^*}\mathcal{M}^{-\frac{1}{2}})\right],\end{aligned} \tag{6.46}$$

and so

$$\begin{aligned}\int_\Omega\;& \nabla(W-\Pi_h W)\cdot\nabla W_h^*\,\mathrm{d}\Omega =\\ &\int_\Omega\frac{27\sqrt{3}}{16}\left[3\,\mathrm{trace}(\mathcal{M}^{-\frac{1}{2}}H_W\mathcal{M}^{-\frac{1}{2}})\,\mathrm{trace}(H_{W^*})-2\,\mathrm{trace}(\mathcal{M}^{-\frac{1}{2}}H_W H_{W^*}\mathcal{M}^{-\frac{1}{2}})\right]\mathrm{d}\Omega.\end{aligned} \tag{6.47}$$

6.1.5 Hessian computation

L^2-Projection

The L^2-projection hessian recovery consists in computing gradients of gradients where the gradients are computed as a cell averaged of the P_1 element gradients. Given a node P_i and a field u_h, we can compute its gradient for each element $\nabla u_h|_K$ containing P_i. The nodal gradient of u is defined as

$$\nabla u_h|_{P_i}=\frac{1}{\sum_{K_j\ni P_i}|K_j|}\sum_{K_j\ni P_i}|K_j|\,\nabla u_h|_{K_j}.$$

This defines a new vector field, and we can compute its gradients $\nabla(\nabla u_h)_x$ and $\nabla(\nabla u_h)_y$. Note that for a continuous field u, the cross derivatives $\frac{\partial^2 u}{\partial x\partial y}$ and $\frac{\partial^2 u}{\partial y\partial x}$ should be equal. But, this is not the case for the numerical reconstruction obtained. For RANS mesh adaptation it is crutial to post process the hessian obtained and enforce

$$H_{xy}(u_h)=H_{yx}(u_h)=\frac{1}{2}((\nabla(\nabla u_h)_x)_y+(\nabla(\nabla u_h)_y)_x),$$

as with highly anisotropic elements, any perturbation in these terms will lead to catastrophic modifications in the anisotropy direction in the boundary layer.

Least-Square Reconstructions

The Least-Square fitting consists in minimizing the discrete nodal L^2 error between an analytical reconstruction and the discrete field. Namely, given a node P_i, a set of neighbours P_j and the solution field u_j on these nodes, we seek the analytical quadratic field w minimizing

$$\varepsilon = \sum (w(P_j) - u_j)^2.$$

The field w can be expressed (exactly) with its Taylor expansion centered in P_i:

$$w(P) = w_0 + G \cdot (P - P_i) + \frac{1}{2}(P - P_i)H(P - P_i),$$

where, G and H are the gradient and the hessian of w in P_i. Hence, ε is a differentiable expression of G and H, and as w minimizes ε, we have

$$\frac{\partial \varepsilon}{\partial G_x} = ... = \frac{\partial \varepsilon}{\partial H_{xx}} = ... = 0.$$

Note that here $P_j - P_i$ is not a variable but a data, so that the nodal values $w(P_j)$ can be expressed as a linear combinations of the $(P_j - P_i)$, $(P_j - P_i)^2$, ..., as

$$w(P_i) = (w_0, G_x, ..., H_{xx}, H_{xy}, ...) \cdot (1, (P_j - P_i)_x, .., (P_j - P_i)_x^2, (P_j - P_i)_x(P_j - P_i)_y, ...).$$

The derivatives of ε expresses thus simply as

$$\begin{array}{rcl} \frac{\partial \varepsilon}{\partial w_0} & = & 2\sum (w(P_j) - u_j) \\ \frac{\partial \varepsilon}{\partial G_x} & = & 2\sum (w(P_j) - u_j)(P_j - P_i)_x \\ & ... & \\ \frac{\partial \varepsilon}{\partial H_{xx}} & = & 2\sum (w(P_j) - u_j)(P_j - P_i)_x^2 \end{array} \tag{6.48}$$

which forms the linear system

$$\begin{bmatrix} \sum 1 & \sum (P_j - P_i)_x & ... & \sum (P_j - P_i)_x^2 & \sum (P_j - P_i)_x (P_j - P_i)_y & ... \\ \sum (P_j - P_i)_x & \sum (P_j - P_i)_x^2 & ... & \sum (P_j - P_i)_x^3 & \sum (P_j - P_i)_x^2 (P_j - P_i)_y & ... \\ \vdots & & & & & \\ \sum (P_j - P_i)_x^2 & \sum (P_j - P_i)_x^3 & ... & \sum (P_j - P_i)_x^4 & \sum (P_j - P_i)_x^3 (P_j - P_i)_y & ... \\ \vdots & & & & & \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ G_x \\ \vdots \\ H_{xx} \\ \vdots \end{bmatrix}$$
$$= \begin{bmatrix} \sum u_j \\ \sum u_j (P_j - P_i)_x \\ \vdots \\ \sum u_j (P_j - P_i)_x^2 \\ \vdots \end{bmatrix}. \tag{6.49}$$

The solution of the minimization problem can thus be obtained by solving the linear system of Equation (6.49). Different variations of this method can be considered. The nodal value of

w in P_i can either be fixed to be equal to u_i or part of the minimization problem. If $w_0 = u_i$ is fixed, the first row and column of Equation (6.49) are removed. In the second case (that we described over), w_0 is free to change so that the nodal difference $w_0 - u_i$ has to be minimized too and P_i has to be considered as a neighbour of itself and taken into the sum. Higher order versions of w (cubic, quartic, ...) can be considered as well, adding variables to the linear system.

The hessian and gradients of u_h are straightforwardly deduced from w. The Least-Square fitting requires at least a non-degenerated stencil for the linear system to be invertible. Namely, it requires to have at least as many vertices as degrees of freedom. In practice, for quadratic fitting the first order ball in a mesh is sufficient in 3D.

Some prefer a weighted version of the least-square reconstruction. It consists in weighting the nodal errors in sum ε according to the precision that each node is alleged to have. In practice, we use a second order ball and the interpolation error of the direct neighbours of P_i are weighted by a factor 1 while the interpolation error of the neighbours of the neighbours of P_i are weighted by 0.1. That way, the closer nodes to P_i have a stronger contribution. Additionally, if w_0 is not fixed, the contribution of P_i is weighted by 10.

6.2 Mesh adaptation cycle for applied cases

We described in Section 5.4.1 the standard adaptation loop for a given complexity. For applied cases, beyond an optimal adapted mesh and computation at a given complexity, it is also important to study the convergence of the solution with increasing mesh complexity to estimate the numerical errors. This is why we use the general mesh adaptation described in Algorithm 7. The idea is to optimize the adaptation by starting the computations on coarse meshes that are cheap to use. The solutions being interpolated for a mesh to another, computations on finer grids are then faster to perform as we start from a good initial solution.

The list of the final sub-iteration at each complexity form the global convergence. Both the global convergence and the convergence of the sub-iterations are good indicators of the state of the simulation. The number of sub-iterations is either fixed to a given number or determined through a sensor. Basically, we choose a function of interest such as the drag or the lift and check its convergence by comparing its successive values on successives grids.

Meshes are iteratively generated with **feflo.a** [Loseille 2013, A. Loseille 2017]. The metric fields are computed either with:

- feature-based mesh adaptation using the Mach fiels as sensor, *i.e.* minimizing the interpolation error in L^2-norm of the Mach field (see Section 5.4)
- goal-oriented mesh adaptation based on the lift or the drag, *i.e.* minimizing the implicit error committed on the considered output (see Section 6.1).

6.3 Applied cases: NASA Rotor 37

One of the strategies initially foreseen to deal with RANS mesh adaptation was to regenerate at each iteration a structured boundary layer and to adapt with a traditional unstructured

Algorithm 7 General mesh adaptation loop

Initial mesh-solution couple: $(\mathcal{H}_0, S_0)$
Initial complexity: $\mathcal{C}_0$
while $\mathcal{C}_{Ite} < \mathcal{C}_{max}$ **do**
    for $SubIte < SubIteMax$ **do**
        Compute the optimal adaptation metric: $\mathcal{M}_{Ite,SubIte}$
        Generate the new adapted mesh at complexity $\mathcal{C}_{Ite}$: $\mathcal{H}_{Ite,SubIte}$
        Interpolate previous solution on new mesh:

$$(\mathcal{H}_{Ite,SubIte-1}, S_{Ite,SubIte-1}) \rightarrow (\mathcal{H}_{Ite,SubIte}, S^{Ini}_{Ite,SubIte})$$

        Compute new solution on $\mathcal{H}_{Ite,SubIte}$: $S_{Ite,SubIte}$
        if $\frac{|\text{sensor}_{SubIte-1} - \text{sensor}_{SubIte}|}{|\text{sensor}_{SubIte}|} \leq$ tolerance **then**
            Break
        end if
    end for
    $\mathcal{C}_{Ite+1} = \text{coef} \cdot \mathcal{C}_{Ite}$
end while

approach the rest of the domain. To do so, we would have to separately adapt the surface mesh, before regenerating a boundary layer. Beside the facts that it raises question on how to deal with the generation of the boundary layer and how to handle the volume metric in the process, it requires to wisely choose how to project the metric on the surface. Additionally, we need to know when to stop the growth of the boundary layer. We will see here that this is an extremely difficult question to solve and that traditional approaches based on a fixed number of layers or anisotropy criterions yield poor results.

We analyse here in more details the mesh generated in Section 3.3.2 and compare it to a full unstructured mesh adaptation. We recall that in Section 3.3.2 we generated adapted meshes for the RO37 (Rotor 37) case with feature-based adaptation using the Mach field with L^2 projection hessian reconstruction. To do so, we started from a mesh with a prescribed boundary layer that we pre-identified and keep unchanged throughout the process. Hence, the same boundary layer is used for all meshes. In that sense, it will be a good illustration of the consequences of a bad choice in the surface metric or boundary layer thickness.

For the present study, we generated adapted meshes with feature-based adaptation on the Mach field with L^2 projection hessian reconstruction. But this time the whole domain, including boundary layers, is adapted with a full unstructured mesh. We generated five meshes for each of the following complexities: 200 000, 400 000, 800 000, 1 600 000 and 3 200 000 vertices.

We recall that the RO37 is simulated in the rotating frame attached to a blade, rotating at 1 800 rad.s^{-1}. Assuming the flow to be periodic, we restrain the domain to a 10° sector around a single blade. Hence, the blade and the rotor are modelled with no-slip walls, while the casing is simulated with a circular sliding wall (see Section 1.1.4 for details). Inflow and outflow are imposed using specific boundary conditions developed in Section 1.1.3. The main characteristics of the flow are listed in Table 6.1 and the parameters used in the solver in Table 6.2

Reference State:		Boundary conditions	
Density	1.275349 $kg.m^{-3}$	Inflow total pressure	1.01300 P_{ref}
Velocity	(400,0,0) $m.s^{-1}$	Inflow total temperature	1.098297639 T_{ref}
Pressure	10^5 Pa	Outflow static pressure	1.01300 P_{ref}
Dynamic viscosity	1.716×10^{-5} $Pa.s^{-1}$		

Table 6.1: NASA RO37: Flow description.

Turbulence model	SA Negative	Riemann Solver	HLLC
MUSCL extrapolation	Order 2 - V4	Limiter	Piperno
Cells	Median	Gradients reconstruction	L^2-projection
Renumbering	None	Initial CFL	10
CFL progression	Linear: 1.15	CFL_{max}	100
Freeze limiter iteration	1500	Max Solver iterations	2000
Targeted physical residual	10^{-8}	Targeted residual reduction	10^{-4}
Linear solver	SGS	Max linear iterations	50
Targeted linear residual	0.1		
Aero reference surface	0.04	Aero reference length	0.2
Center of pressure	$(0.55, 0.35, -0.1)$		

Table 6.2: NASA RO37: Solver parameters.

6.3.1 Geometry description

The correct geometric representation of the different walls is crucial here, beyond the respect of the CAD. Indeed, the RO37 case involves rotating boundary layers, so that if the casing is not circular up to a sufficient precision, the boundary velocity imposed will not be correctly aligned with the wall. This miss-alignement will create pressure and density spikes on the rotating wall. We recall that computations are performed in the rotating frame so that the rotating blade is fix in the computational frame, while fixed elements in the reference frame are rotating in the computational frame. Similarly, imperfections on the rotating disk will induce shock waves and oscillations in the density and pressure fields on the wall boundary condition. Finally, variations of the geometry in the process will induce miss-alignements in the metric field with the boundary that will generate spikes in the mesh during the adaptation process.

In order to re-project vertices on the geometry at each mesh adaptation step, we do not rely on the full CAD as it would be too expensive. Instead, we generate a cubic reconstruction of the geometry with a clean tesselation of the CAD as shown in Figure 6.4.

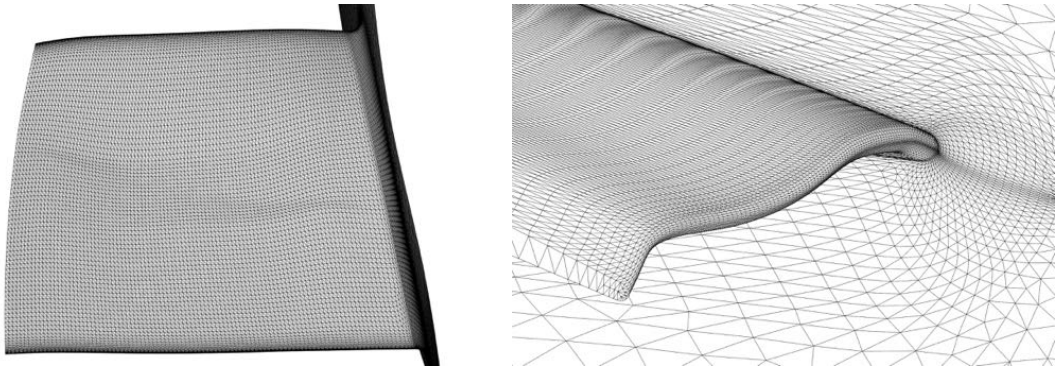


Figure 6.4: NASA RO37: Tesselation of the blade used as reference geometry supplied with a cubic reconstruction.

6.3.2 Mesh comparison

The tesselation of the CAD is also used as initial mesh to start the adapted computation. It is composed of 80 774 vertices and 392 956 tetrahedra and is generated as a mesh for inviscid computation: it contains no boundary layer mesh. We compare here an adapted mesh generated with a fixed structured boundary layer coming from our first study with an adapted mesh obtained with a full unstructured mesh adaptation process. The first mesh contains 6 607 814 vertices, 519 894 triangles and 38 835 389 tetrahedra, while the second is composed of 3 516 488 vertices 484 406 triangles and 20 294 221 tetrahedra.

We can already note that although the first mesh contains almost twice more vertices and tetrahedra than the second, they contain roughly the same number of triangles, falsely suggesting that they have a similar discretization of the surfaces. However, as can be seen in Figure 6.5, the anisotropic discretization of the blade saves a lot of elements with a much more accurate representation of the geometry. Figure 6.6, shows a global view of the surface mesh and density field on the blade. We can see how the surface mesh has been automatically adapted to deal with the shock boundary layer interaction and the recirculation zone. This provides a much cleaner and sharper solution. The imprint of the shock is sharper (despite the boundary layer) and surprisingly, the structured boundary layer even produces noise, part of which is due to the transition with the unstructured mesh.

A cut in the shock-boundary layer interation zone points out in Figure 6.7 how the shock cuts through the boundary layer down to the skin, leading to the detachment of the boundary layer. We can observe that the adaptation of the surface mesh improves the representation of the density field and pressure coefficient on the skin. But, the non-adaptation of the structured boundary layer also seems to lead to the diffusion of a part of the lambda shock. So far, as we noticed previously, the structured boundary layer seems to play a very negligible role as it appears to be smaller than the physical boundary layer itself, which is adapted. However, a closer look at its interaction with other physical phenomena in Figure 6.8 shows that, it actually prevents the bow shock from going down to the skin. We also note that the mesh size prescribe by feature-based adaptation is already finer close to the skin than the initial boundary layer, despite the fact that the mesh with structured boundary layer contains more elements.

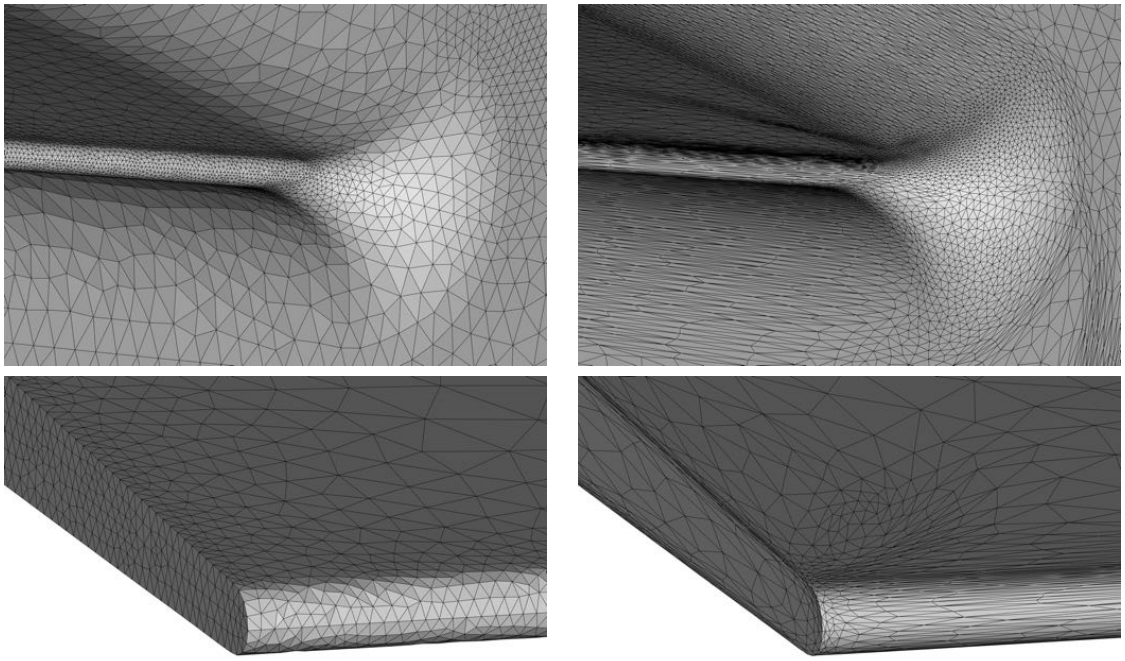


Figure 6.5: NASA RO37: Discretization of the leading edge leading edge of the blade without (left) and with (right) mesh adaptation.

Figure 6.8 perfectly illustrates one of the major problem to solve when generating adapted structured boundary layer. As we mentioned, the main strategy would have been at each iteration to adapt the skin of the blade and then generate a structured boundary layer mesh. However, it is here clear that the choice of the height and the growth of the structured boundary layer is a crucial point. It has to take into account both the boundary layer itself but also external features of the flow such as the shocks in order to compute accurately interactions.

Similarly, the adaptation of the blade is a non-trivial task. In the full unstructured case, the metric used to adapt the surface mesh is the projection on the surface of the local nodal volume metric. Hence, the surface mesh automatically fits with the volume mesh and is continuously adapted throughout the boundary layer mesh, up to the main flow field. But, if we have to adapt separately the surface and then grow/insert a structured boundary mesh, we would have to choose a surface metric. Then, this metric would be propagated vertically (normally to the surface) throughout the boundary layer by the structured geometry. But, in Figure 6.8, we can see that the shock hits diagonally the surface. Hence, if we use the metric computed somewhere above the boundary layer, the foot of the shock will be diffused, while if we use the metric computed on the skin, the shock will not be sufficiently discretized above and will never make it through.

An option would be compute the mean or the intersection of the metrics (in a sense to be defined) throughout the boundary layer and use it to adapt the surface. In our case it would spread the shock on the surface spilling ressources. Another solution would be to subdivide the

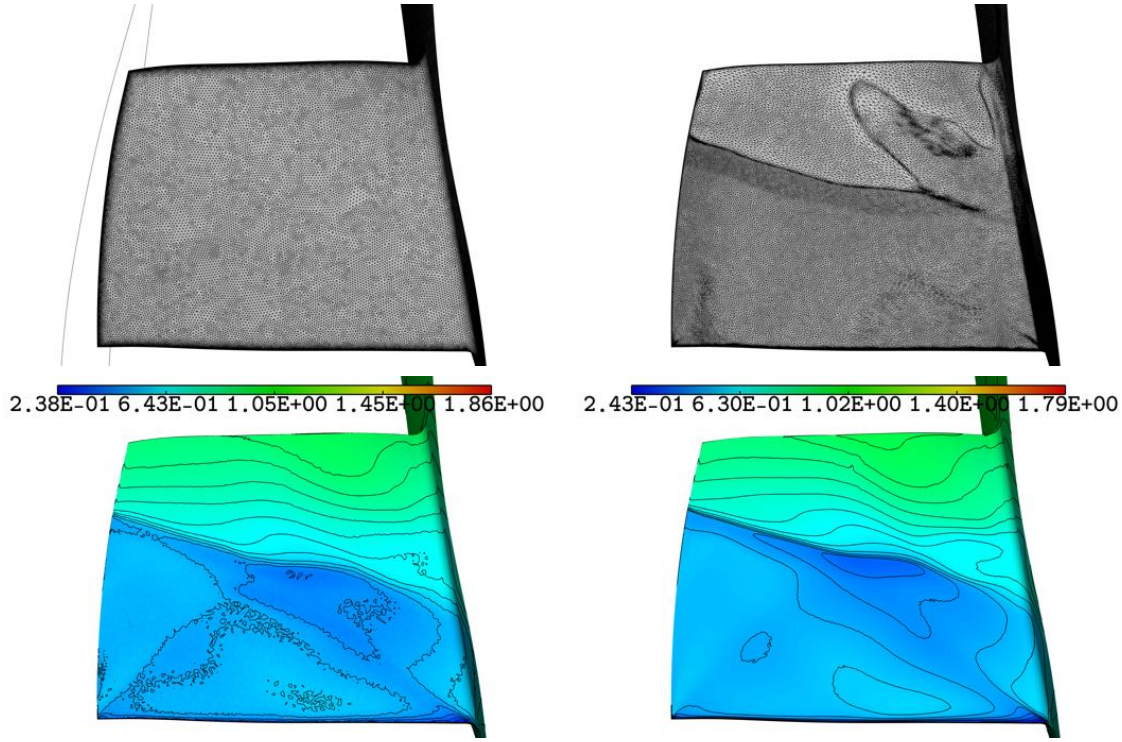


Figure 6.6: NASA RO37: Surface mesh (top) and density field (bottom) on the blade with frozen structured boundary layer mesh (left) and with full unstructured boundary layer mesh adaptation (right).

structured boundary layer mesh following the metric once it is generated. This would affect the structure of the boundary layer that would not be cartesian any more.

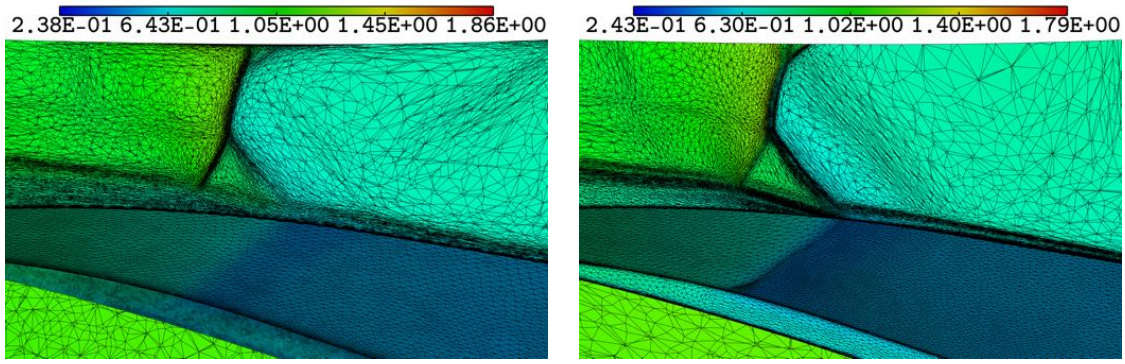


Figure 6.7: NASA RO37: Cut in the region of the shock boundary layer interaction. Mesh and density field with frozen structured boundary layer mesh (left) and with full unstructured boundary layer mesh adaptation (right).

We recover in Figure 6.9 the same behaviour near the leading edge of the blade. This time the structured boundary layer mesh that is used is thicker than the physical boundary layer

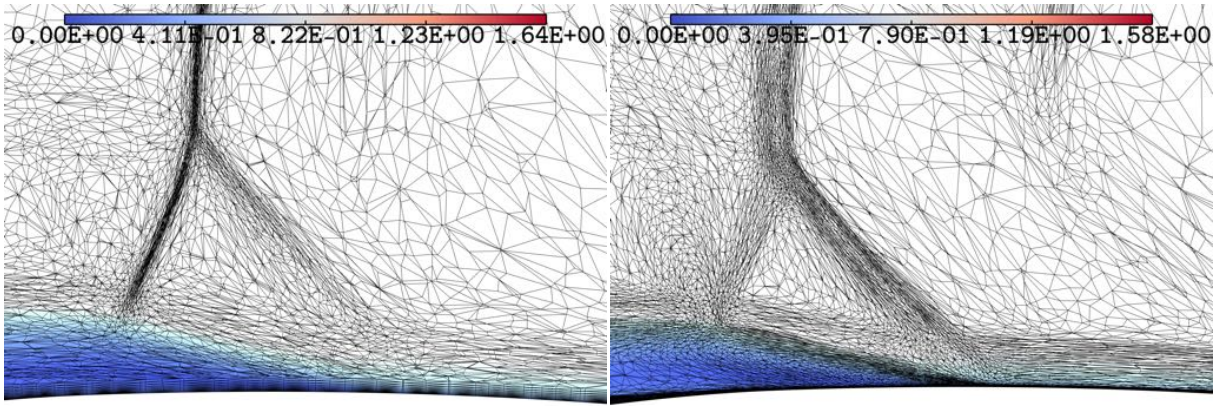


Figure 6.8: NASA RO37: Cut in the shock boundary layer interation, mesh and velocity field with frozen structured boundary layer mesh (left) and with full unstructured boundary layer mesh adaptation (right).

and directly affects the solution. In particular, we can see that it prevents the shock from going down to the skin. Meanwhile, the full unstructured approach automatically adapts to the solution and prevents the artificial diffusion of the shock.

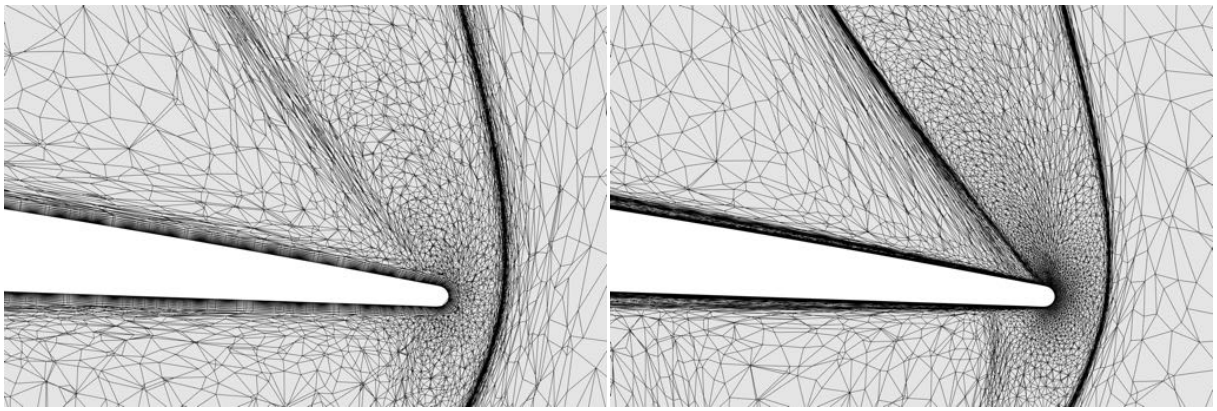


Figure 6.9: NASA RO37: Cut in the mesh on the leading edge of the blade with frozen structured boundary layer mesh (left) and with full unstructured boundary layer mesh adaptation (right).

Another interesting point is the discretization of the tip gap flow and vortex. We can see in Figure 6.10 the structure of the vortex visualized with the density field and the meshes generated, at its formation (bottom) and when it strays from the blade (top). The zoom on the formation of the vortex shows the variety of the structures that takes place and have been automatically captured both on the top of the blade and in the tip gap. These structures are absorbed in the structured boundary layer mesh initially generated and are thus not captured. We can note here again the oscillations in the density field produced in presence of the structured boundary layer. We insist on the difficulty that represents the generation of a structured boundary layer in a such enclosed space with sharp edges as the tip gap. This is automatically handled with

unstructured mesh adaptation.

Further downstream in Figure 6.10, we can see in the full unstructured adapted case both the structures of the vortex and the jet that forms in the tip gap. Part of these structures takes place near the wall, and are thus absorbed in the structured boundary layer mesh. But they are not aligned with the wall and are thus probably not discretized as well as they should be.

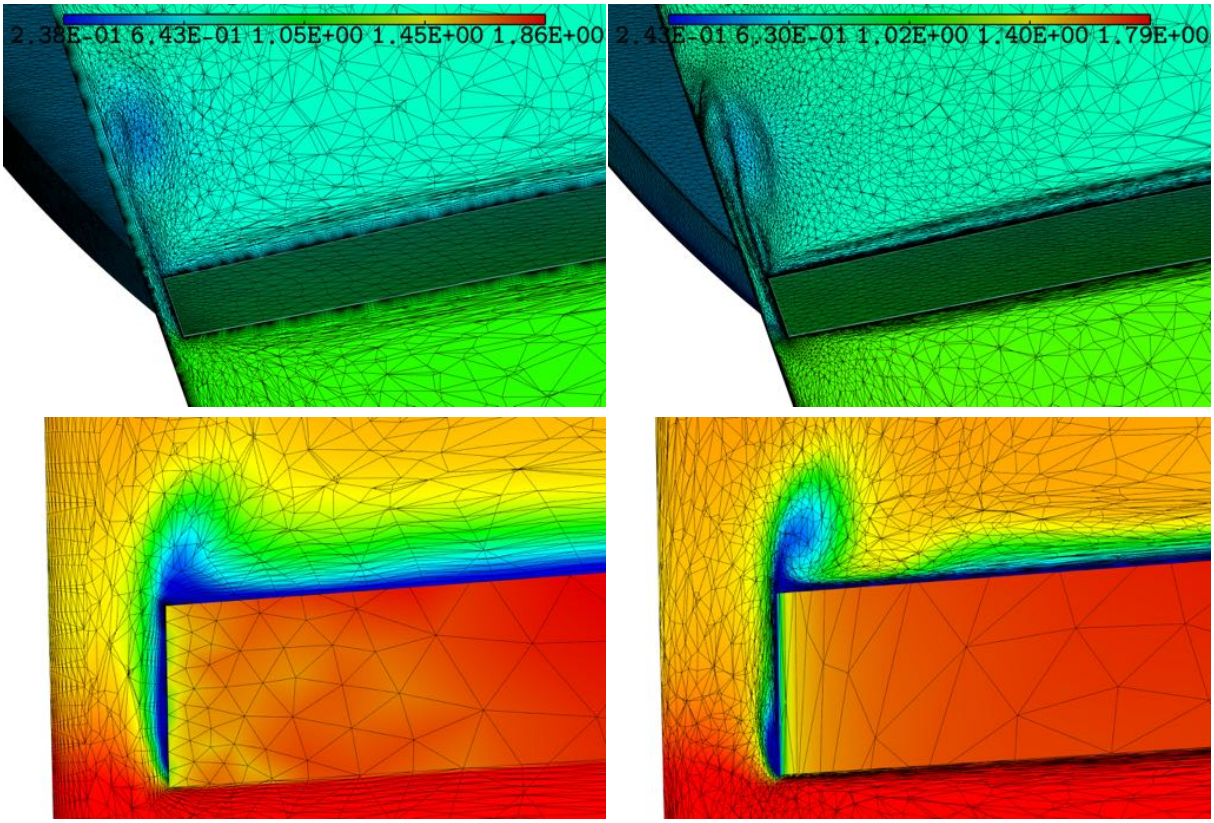


Figure 6.10: NASA RO37: Cut in the mesh on the leading edge of the blade with frozen structured boundary layer mesh (left) and with full unstructured boundary layer mesh adaptation (right).

Finally, Figure 6.11 depicts the convergence of the torque and thrust exerted on the blade throughout the adaptation process. The red curve shows every iterations of the adaptation, while the black curve shows the final mesh at each complexity. We can see the importance of iterating on the coarse meshes as most of the convergence of both outputs is performed on these. Still, as they are coarse meshes, these are very cheap solutions to compute. The sub iterations also give a sharp estimate of the convergence of the solution both on the current complexity but also with respect to the overall computation. Hence, we can see here that we have wide variations on the coarse meshes but very little on finer meshes. Yet, performing a fixed number of iterations at each complexity is clearly suboptimal as we would prefer to perform more iterations on the coarse meshes where the process is clearly not converged, while we would like to skip some of the expensive computations on finer meshes.

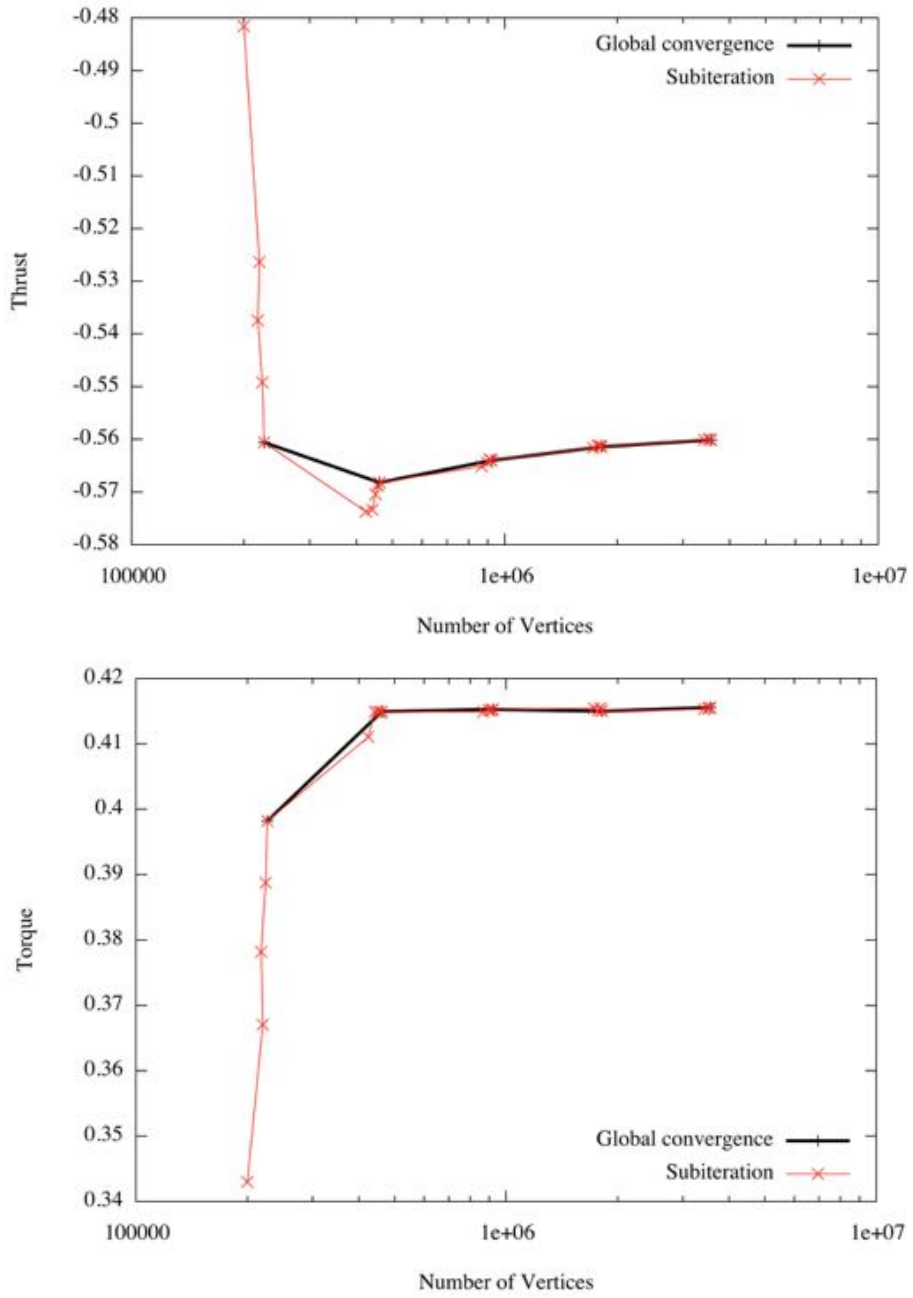


Figure 6.11: NASA RO37: Convergence of the torque (bottom) and thrust (top) with feature-based mesh adaptation on the Mach field.

6.4 Applied cases: M6 wing

The Onera M6 wing is a classical validation test case for external flow solvers. It has been extensively studied in wind tunnel and with numerical simulations thanks to its simple geometry. Still its transonic regime involves a large deal of interesting physical phenomenon. We show

here how different mesh adaptation strategies perform with this case.

6.4.1 Case description

The M6 wing is a swept, semi-spawn wing with an aspect ratio of 3.8. Its leading edge has a sweep angle of 30.0° and its trailing edge a sweep angle of 15.8°, see Figure 6.12-(left). It has been studied under different flight conditions but we will retain here only the parameters listed in Table 6.3.

Mach Number	0.84	Angle of attack	3.06°
Reynolds Number	14.6×10^6	Reference Reynolds Length	1
Reference Temperature	300		

Table 6.3: Onera M6 wing: Flight conditions.

The geometry is scaled so that the root chord of the wing (on which the Reynolds number is based) is equal to 1. For all computations, the wing is immersed in a semi-spherical domain of radius 100 in order to minimize the effects of the far field modelization. The other half of the domain is modelled through a symmetry plane. We use adiabatic solid walls on the wing and Steger-Warming Riemannian boundary conditions to impose the far field boundary conditions. For all computations we use the Spalart-Allmaras negative model presented in Section 2.1.1. The geometry of the wing is defined with a cubic reconstruction of a fine surface mesh. The extensive list of parameters used in the solver to perform these computations is liste in Table 6.4, see Chapter 1 for details. All computations are started on a non-adapted "Euler mesh" with no boundary layer, composed of 18 798 vertices and 96 661 tetrahedra, shown in Figure 6.12-(right).

Turbulence model	SA Negative	Riemann Solver	HLLC
MUSCL extrapolation	Order 2 - V4	Limiter	Piperno
Cells	Median	Gradients reconstruction	L^2-projection
Renumbering	BFS	Initial CFL	10
CFL progression	Linear: 5	CFL_{max}	100
Freeze limiter iteration	1000	Max Solver iterations	2000
Targeted physical residual	10^{-8}	Targeted residual reduction	10^{-3}
Linear solver	SGS	Max linear iterations	30
Targeted linear residual	0.1		
Aero reference surface	1.1531508	Mean Aerodynamic Chord	0.801672
Center of pressure	$(0, 0, 0)$		

Table 6.4: Onera M6 wing: Solver parameters.

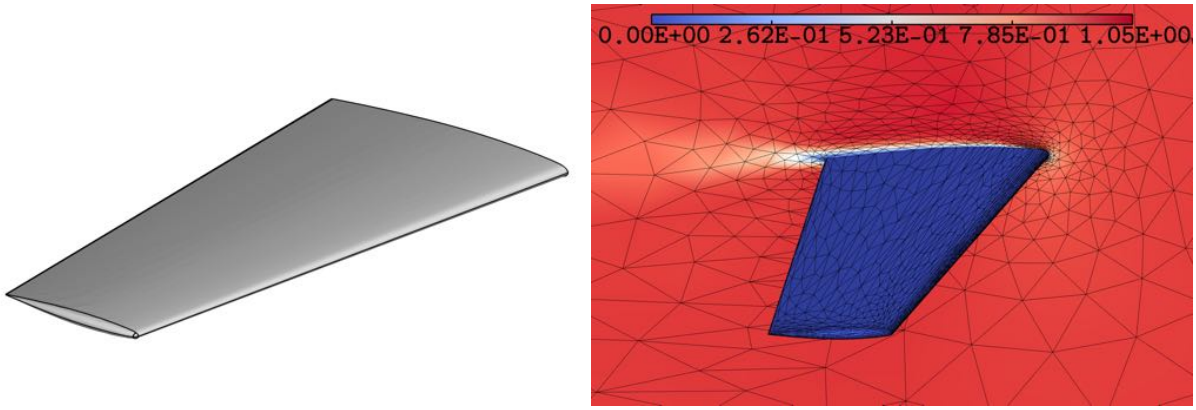


Figure 6.12: Onera M6 wing geometry (left) and velocity field computed on the initial mesh (right).

6.4.2 Feature-based mesh adaptation

We first lead a standard feature-based mesh adaptation by minimizing the L^2-norm of the interpolation error of the Mach field. The whole domain is adapted with an unstructured mesh, including the boundary layer region. We start with an isotropic Euler-kind mesh, with an element size of about 0.1 near the wing (see Figure 6.12-(right)). We perform up to 20 sub-iterations to converge the mesh-solution couple at each of the following complexities: [20 000, 40 000, 80 000, 160 000, 320 000, 640 000, 1 280 000, 2 560 000, 5 120 000]. The convergence of the solution is assessed by comparing the current drag to the two previously computed drags. If they vary from less than 0.5% the solution is considered to be converged and we multiply the complexity by two.

This process is clearly visible in Figure 6.13 where we plotted the sub-iterations alongside the global convergence for the total drag (top), the pressure drag (bottom left) and the viscous drag (bottom right). We have large variations of the drag between two grids on coarse adapted meshes, which requires several sub-iterations to converge: 8 for the coarsest level, 14 for the second and 6 for the third. Still, these computations are very cheap as they are done with very few degrees of freedom. On the opposite, on fine meshes, the drag variations are very small so that only 3 sub-iterations are required (the minimum imposed). Although these meshes are much bigger (12M nodes), only the first of the three computations is expensive. Indeed, as the solution is interpolated between two grids and does not vary much, it requires very few iterations for the solver to converge. These grid to grid variations of the drag are a good indication of the convergence of the solution.

The main concern with RANS feature-based adaptation was the ability of the process to successfully deal with the turbulent boundary layer while using a simple sensor such as the Mach field. Figure 6.13-(bottom left) seems to indicate that the viscous component successfully converges. Additionally, Figure 6.14 shows the convergence of the surface meshes generated throughout the adaptation process and the corresponding dimensionless wall spacing. We can see that the wall spacing is initially rather high (> 10) but successfully decreases down to

about $y^+ = 1$ in front of the shock. Note that the dimensionless wall spacing is computed here against the reference spacing $d_{y^+=1}$ (see Section 2.3.1), it is thus not exactly the local y^+. In Figures 6.15 and 6.16 we can see, as described in 2D in Section 6.7, that the physical boundary layer thickness is decreasing during the adaptation process until it reaches the proper size (especially visible on the trailing edge). This also indicates the progressive adaptation of the boundary layer, similar to what we observe in shocks. Although we would like a finer refinement of the boundary layer, this is consistent with the error the we choose to minimize. The choice of the L^2-norm of the interpolation error of the Mach field lead to a balance in the resolution of the boundary layer, the wake and the shock structure.

Generally, Figures 6.15 and 6.16 show how the mesh is progressively adapted to the different features of the flow in an automated manner, shocks, boundary layer and wake throughout the process. The different shock structures, especially on the aft part of the wing would be particularly difficult to predict and discretize manually, still they are automatically captured. We can note in Figure 6.14 the anisotropy of the mesh that is generated on the wing to deal with the turbulent boundary layer. This is performed under the constraint of the thin boundary layer mesh and the respect of the geometry of the wing. We can note in Figures 6.15 and 6.16 the absence of spurious oscillations and shock waves which indicates that the geometry is smoothly discretized. This validates the cubic reconstruction of the geometry approach to deal with the geometry.

6.4.3 Comparison of hessian reconstructions

We introduced in Section 6.1.5 different method for the hessian reconstruction of the sensor field, that are mandatory for the computation of the optimal metric field. We compare here the different meshes and solutions obtained with these methods. To do so, we performed the same convergence study as in the previous section, with L^2-projection (L2P), Least-Square (LSQ) and Quadratic Least-Square (QLSQ) reconstructions of the hessian. Additionally, we also compared the effect of metric gradation [Borouchaki 1998, Alauzet 2010c] on this case with the L^2-projection hessian reconstruction by generating a series of meshes with and without gradation.

The resulting surface meshes and dimensionless wall spacing for meshes of about 200 K vertices and 1.6 M vertices are shown in Figure 6.17 and 6.18 respectively. Coarse meshes obtained with QLSQ hessian reconstruction seems to be slightly coarser on the skin than meshes obtained with LSQ reconstruction which also seems coarser than the ones obtained with L2P reconstruction. This trend seems to inverse on finer meshes. None of this is visible on the surface meshes and metric gradation does not have a visible effect either. It is worth noting here that surprisingly, meshes generated without gradation are not particularly noisy.

This trend is confirmed by Figure 6.19 which shows the global convergence of the pressure and viscous drag components. The viscous drag computed on meshes obtained with L2P hessians reconstruction converge clearly faster than with LSQ and QLSQ. However, this is less visible on the pressure drag. Consistently, gradation show very little effect on these outputs. Still, it is important to note here that all methods converge to the same value.

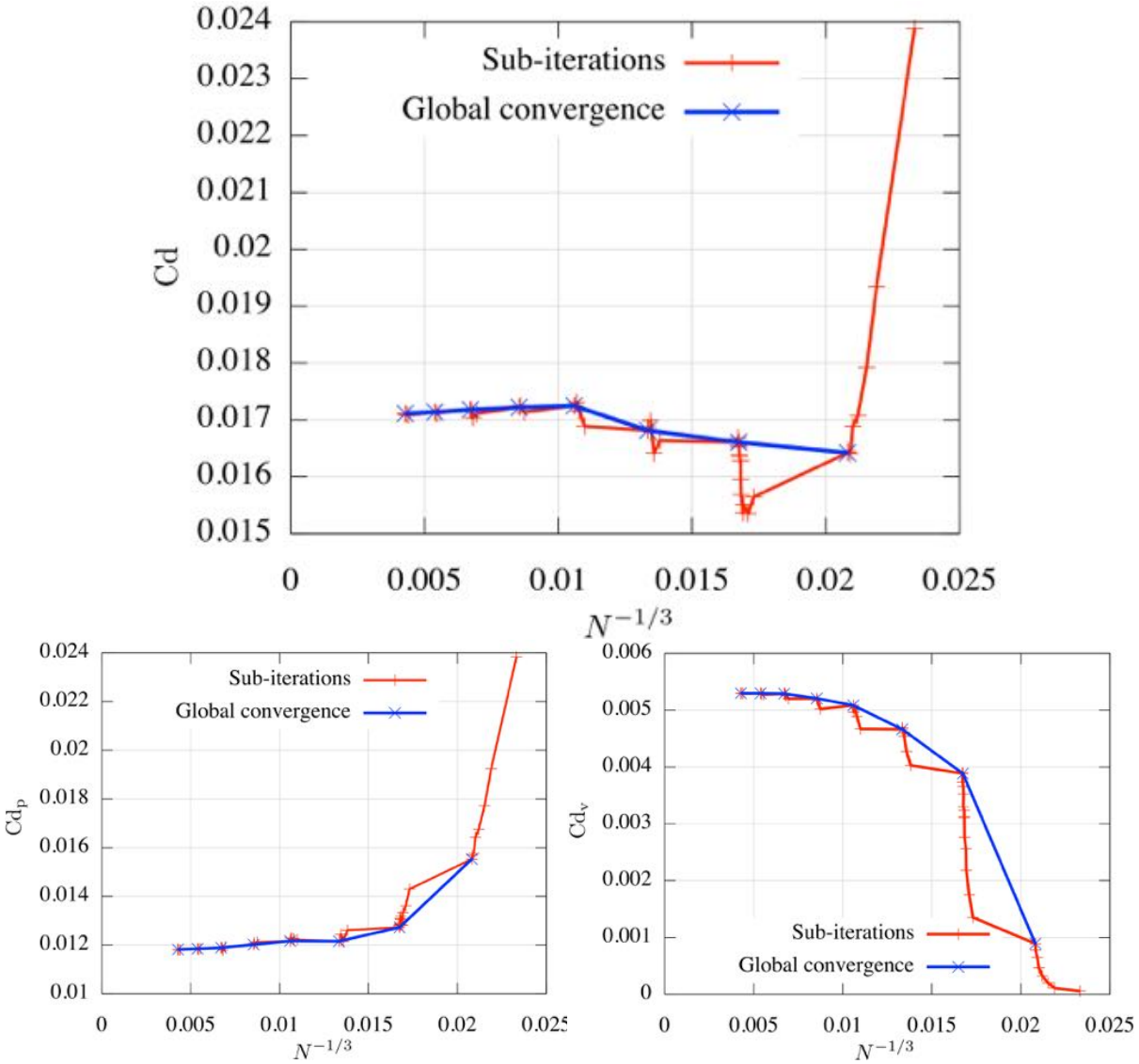


Figure 6.13: Adapted RANS M6 wing: Convergence of the total drag (top), pressure drag (bottom left) and viscous drag (bottom right) contributions, with feature-based adaptation on the Mach field.

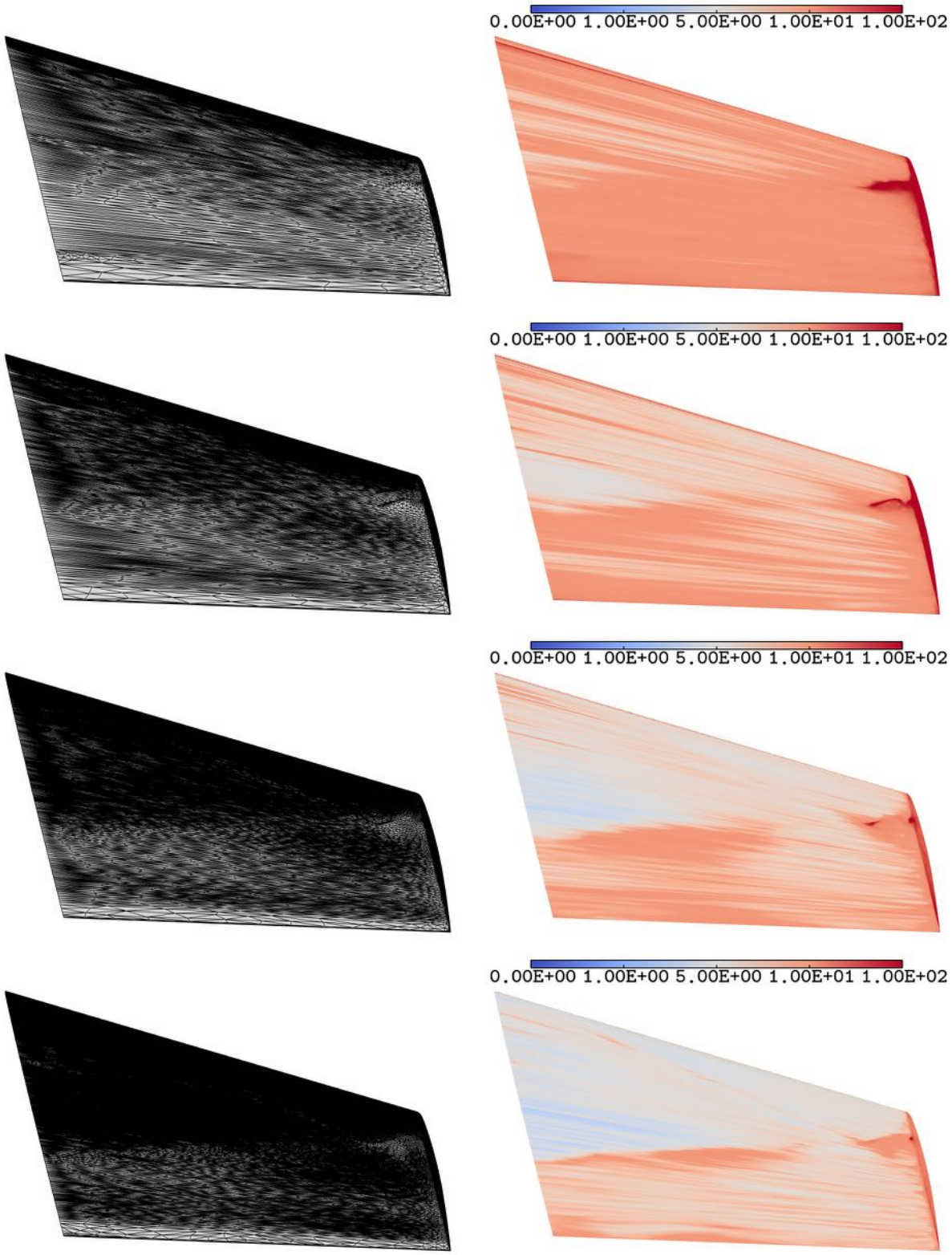


Figure 6.14: Adapted RANS M6 wing: Surface adapted meshes (left) and dimensionless wall-spacing y^+ (right) for the first layer of elements of the meshes generated with feature-based adaptation based on the Mach field. From top to bottom: adapted meshes composed of 200K, 400K, 800K and 1.6M nodes.

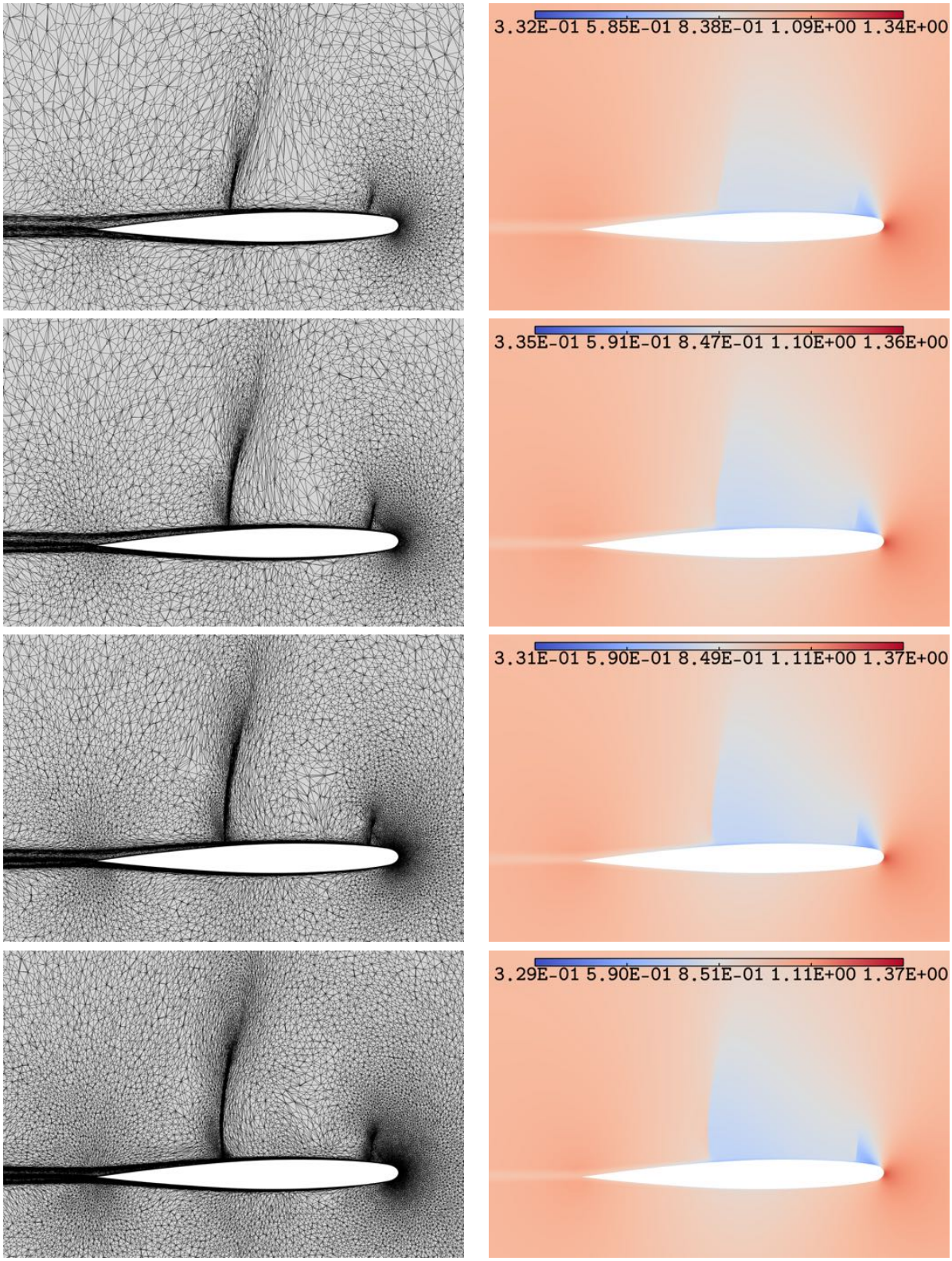


Figure 6.15: Adapted RANS M6 wing: Cut in the volume and density field at $y = 0.5$ for the meshes generated with feature-based adaptation based on the Mach field. From top to bottom: adapted meshes composed of 200K, 400K, 800K and 1.6M nodes.

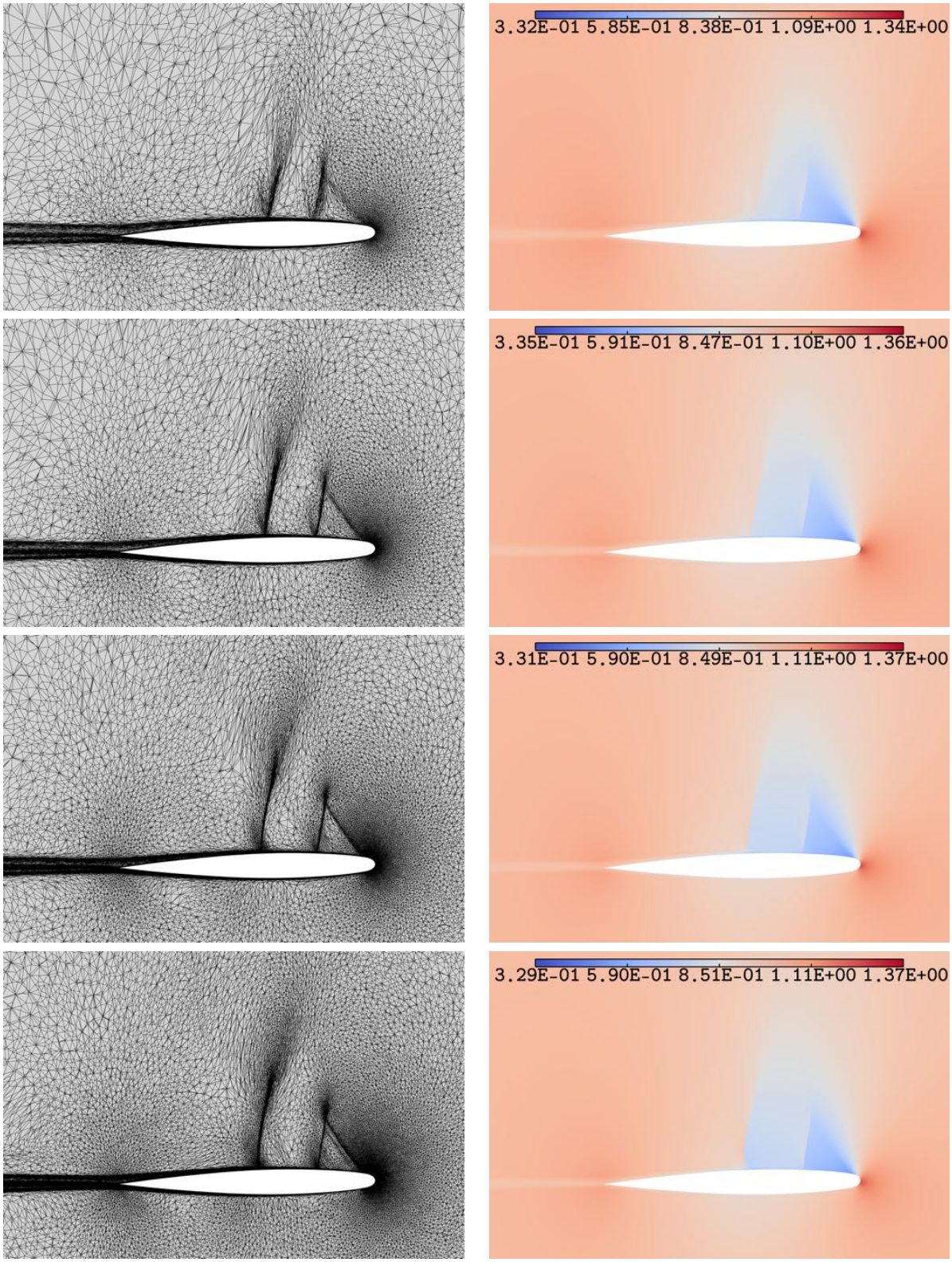


Figure 6.16: Adapted RANS M6 wing: Cut in the volume and density field at $y = 1.0$ for the meshes generated with feature-based adaptation based on the Mach field. From top to bottom: adapted meshes composed of 200K, 400K, 800K and 1.6M nodes.

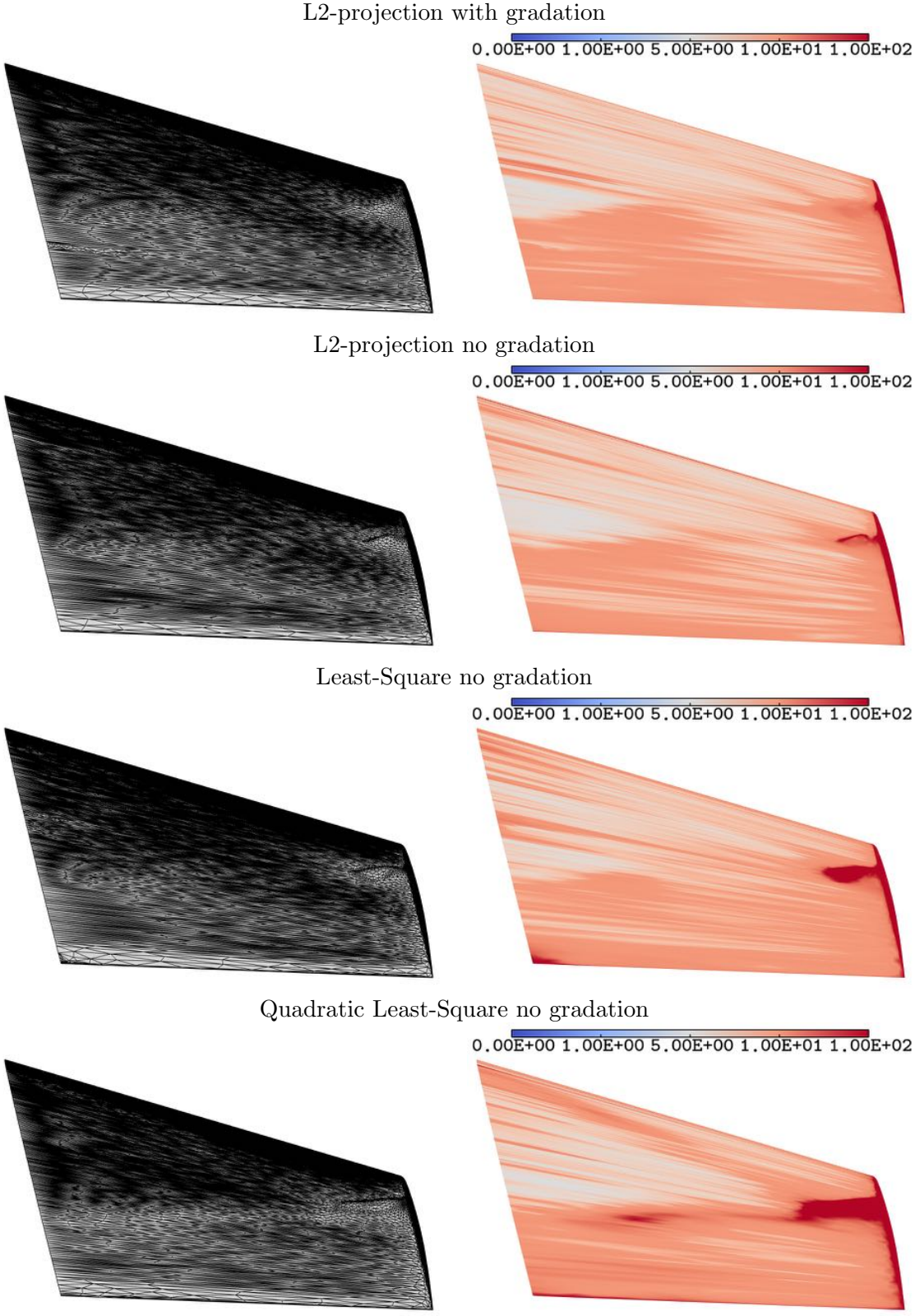


Figure 6.17: Adapted RANS M6 wing: Surface meshes and dimensionless wall-spacing y^+ for the first layer of elements of the adapted meshes generated with feature-based adaptation based on the Mach field obtained with different hessian reconstructions and gradations. Meshes are composed of about 400K vertices.

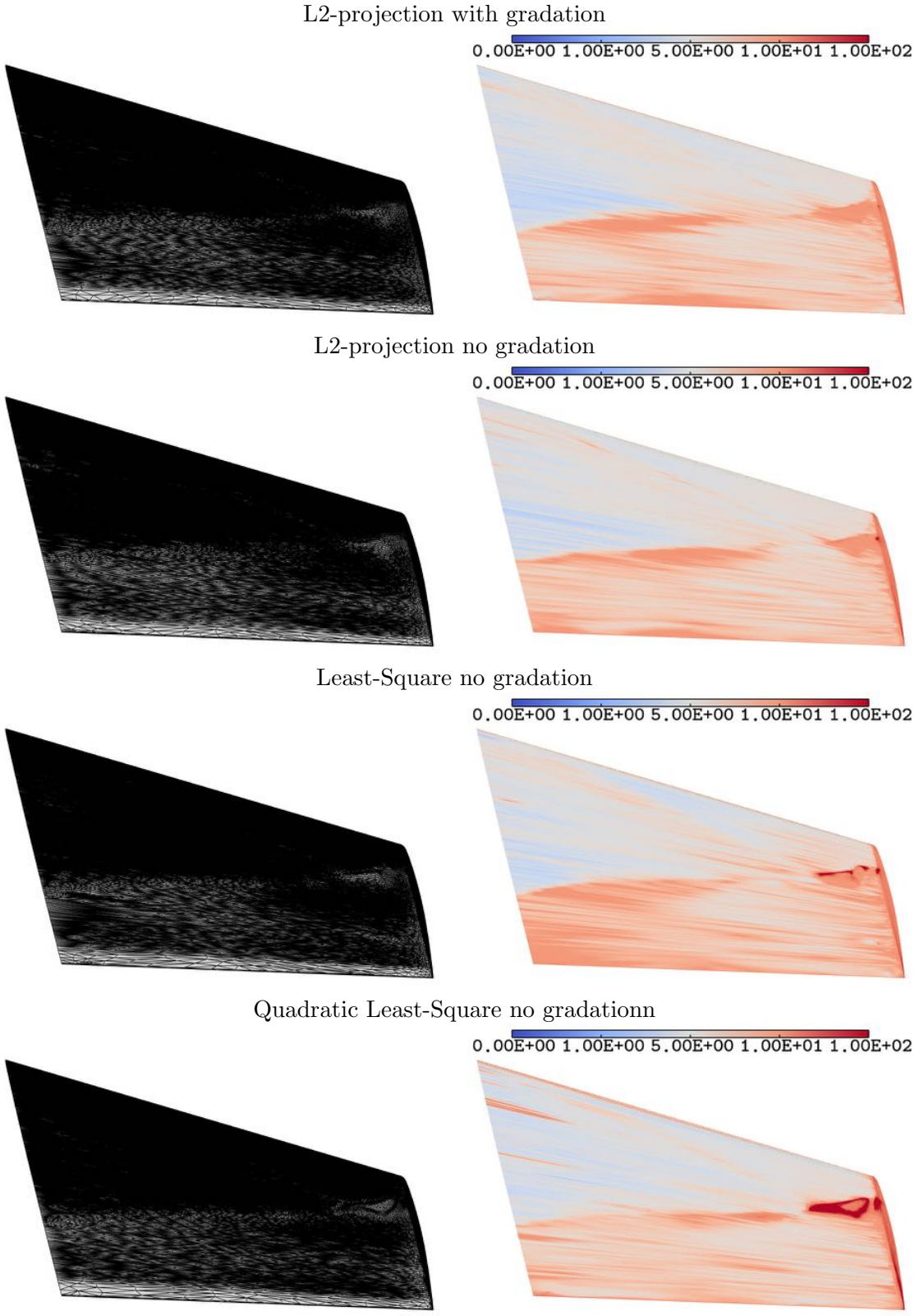


Figure 6.18: Adapted RANS M6 wing: Surface meshes and dimensionless wall-spacing y^+ for the first layer of elements of the adapted meshes generated with feature-based adaptation based on the Mach field obtained with different hessian reconstructions and gradations. Meshes are composed of about 1.6M vertices.

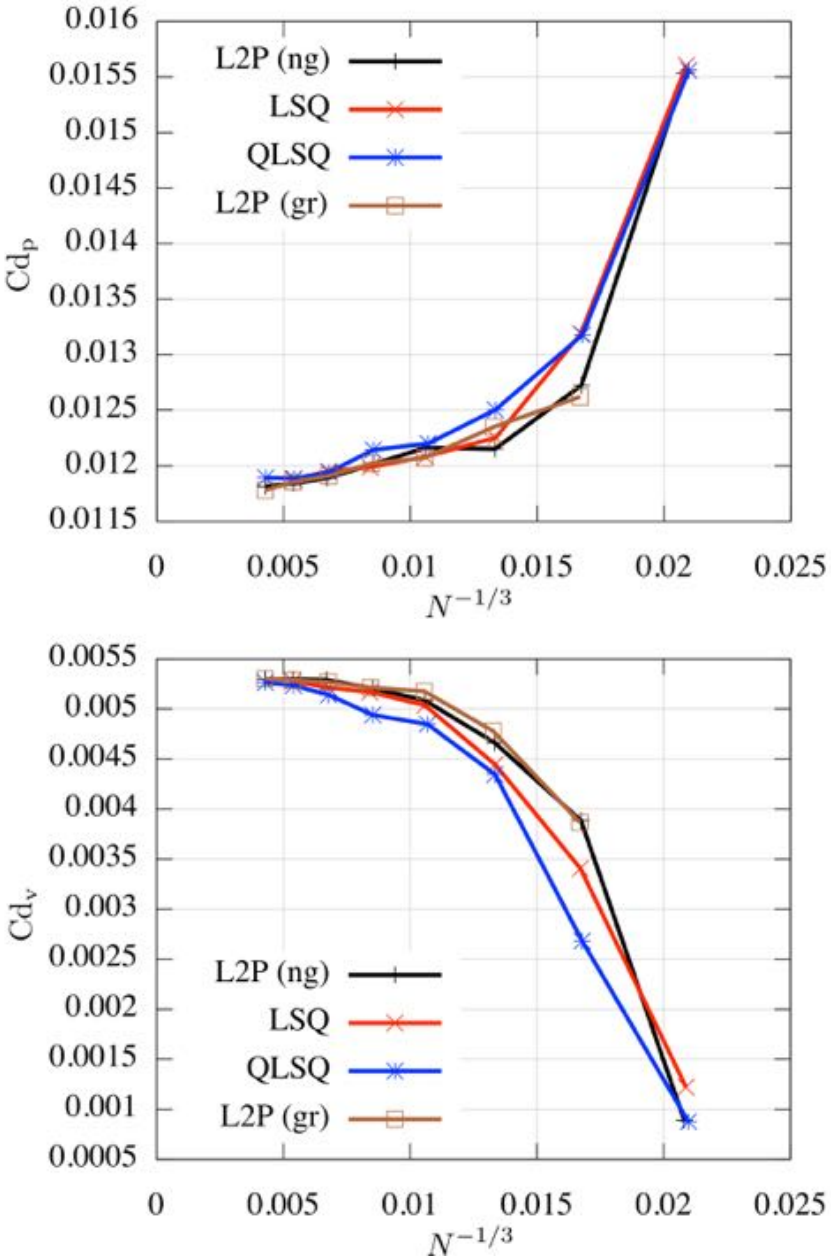


Figure 6.19: Adapted RANS M6 wing: Convergence of the pressure and viscous drag components for the feature-based adaptation based on the Mach field with different hessian reconstructions and gradations.

6.4.4 Goal-oriented mesh adaptation

We developed in Section 6.1 the mathematical context for RANS goal-oriented mesh adaptation. Here, we validate this analysis by comparing the adjoint-based adaptation with and without viscous terms to feature-based adaptation. One validation is that adapted meshes obtained with the viscous goal-oriented approach are more optimal (better) to compute the targeted output, *i.e.* predict a better value with the same number of degrees of freedom. We retained the drag as functional output.

Historically, our flow solver `Wolf` was an Euler flow solver and inviscid adjoint based mesh adaptation has been developed and extensively validated. In the present work, we take into account additionally the contribution of viscous terms in the error analysis. We proposed, in Section 6.1, two main approaches to deal with second order derivatives in the main flow, namely the spectral radius approach and the full hessian approach. We use here the later one with a direct differentiation of the equations. We do not take into account the turbulent equation in the present analysis but we consider the turbulent viscosity in the viscous terms in the main flow. Though, our analysis in Section 4.1.1 tends to indicate that they may be negligible or at least of small effect.

From our previous study in Section 6.4.3, we chose the meshes and solutions obtained with L^2-projection hessian reconstruction without gradation as reference for this case. Then, we compare the solutions obtained with inviscid adjoint mesh adaptation and viscous adjoint mesh adaptation in order to assess both the effects of adjoint mesh adaptation in general and the contribution of viscous terms in particular.

Figure 6.20 shows the convergence of the pressure and viscous components of the drag obtained with both adjoints compared to standard feature-based adaptation (blue). It is here clear that the viscous adjoint mesh adaptation (black line) tremendously improves the computation of functional of interest (here the drag). It is particularly visible on the viscous component of the drag. As shown in Figure 6.21, this is due to an early capturing of the boundary layer. Comparing Figure 6.21 to Figure 6.14 we can see that the 150K vertices adapted mesh produced by adjoint-based mesh adaptation is already as refined on the skin as the 1.6M vertices mesh produced with the feature-based mesh adaptation. Similarly, Figure 6.22 shows the comparison between the surface meshes obtained with adjoint based and feature-based mesh adaptation. We can see how finer is the surface mesh produces with adjoint based mesh adaptation.

Similarly, Figure 6.23 shows a cut in the volume meshes obtained with the feature-based mesh adaptation based on the Mach field and the viscous adjoint-based mesh adaptation on the drag. Comparing to the finer meshes (bottom), we can see that, in the coarse mesh (150K) obtained with adjoint based mesh adaptation, the shock structure is already at the right place and we can already see the leading part of the shock structure discretized. In the mesh produced by the feature-based adaptation the improper discretisation drives the second shock forwards. This may be due to its interaction with the boundary layer which is insufficiently discretized and thus thicker as can be seen especially on the trailing edge. This boundary layer has already the proper thickness on the adjoint based mesh.

Surprisingly, inviscid adjoint based mesh adaptation produces very bad results, even compared to feature-based mesh adaptation. As it only consists in a weighting of the interpolation

error of inviscid fluxes, we expected it to perform at least as good as the simple interpolation error control. But it seems that the improper weighting drastically disturbs the process. Figure 6.24 shows the comparison between the surface meshes and dimensionless wall spacing of meshes produced by viscous and inviscid adjoint-based mesh adaptation. We can see that inviscid terms tends to concentrate elements on the leading edge of the wing, neglecting the rest of it. This leads to very fine elements on the leading edge ($y^+ < 1$) but very large element ($y^+ > 10^3$!!) after, in particular where the shocks interacts with the boundary layer.

This illustrates the importance of the proper treatment of viscous terms and validates the numerical developments of Section 6.1 and their implementation.

6.4.5 Comparison with other codes

Finally, we compare our results to those predicted by other solvers, gathered in the NASA Turbulence Modeling Ressource. All these computations were obtained with the Spalar-Allmaras negative model on the same geometry with structured and unstructured grids. Figure 6.25 shows the grid convergence of the lift (top left), the drag (top right) coefficients and the pressure (bottom left) and viscous (bottom right) components of the drag with respect to the grid index. The solutions computed by `Wolf` are in agreement with the predictions of other solvers which consolidates the validation of the turbulent part of the solver.

`Wolf` with adjoint based adapted meshes also shows an early convergence of the lift, drag and pressure component of the drag compared to other methods. Structured meshes generated for this study contain a prescribed structured boundary layer. We thus expect them to have a good control of the prediction of the viscous component of the drag. Still, `Wolf` performs similarly on fully unstructured grids generated with adjoint-based mesh adaptation. The general consensus was that a fully unstructured grid was not suitable for high-resolution RANS simulations and in particular to compute gradients. But, this study proves that this is possible and it validates the proposed fully unstructured approach to deal with the boundary layer mesh.

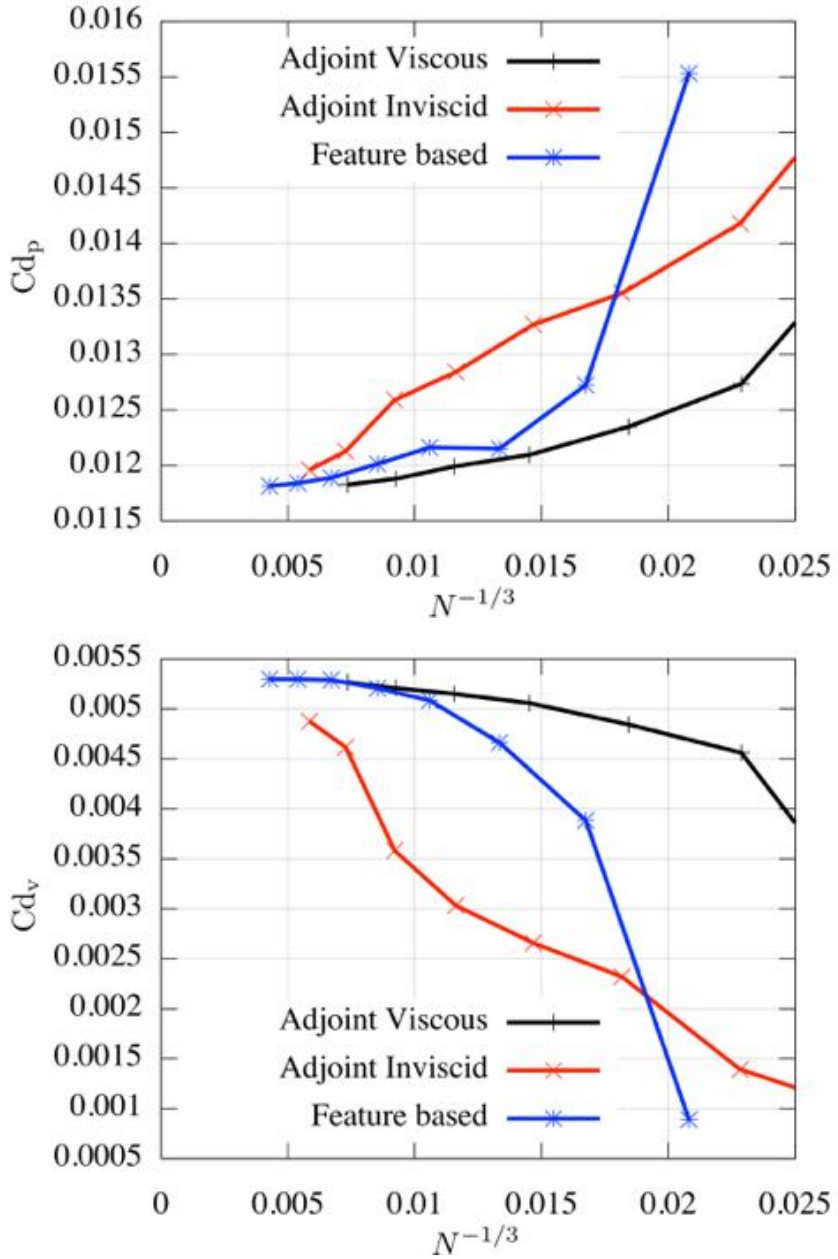


Figure 6.20: Adapted RANS M6-Wing: Comparison of the convergence of the pressure and viscous components of the drag with the feature-based (blue), inviscid (red) and viscous (black) adjoint-based mesh adaptation.

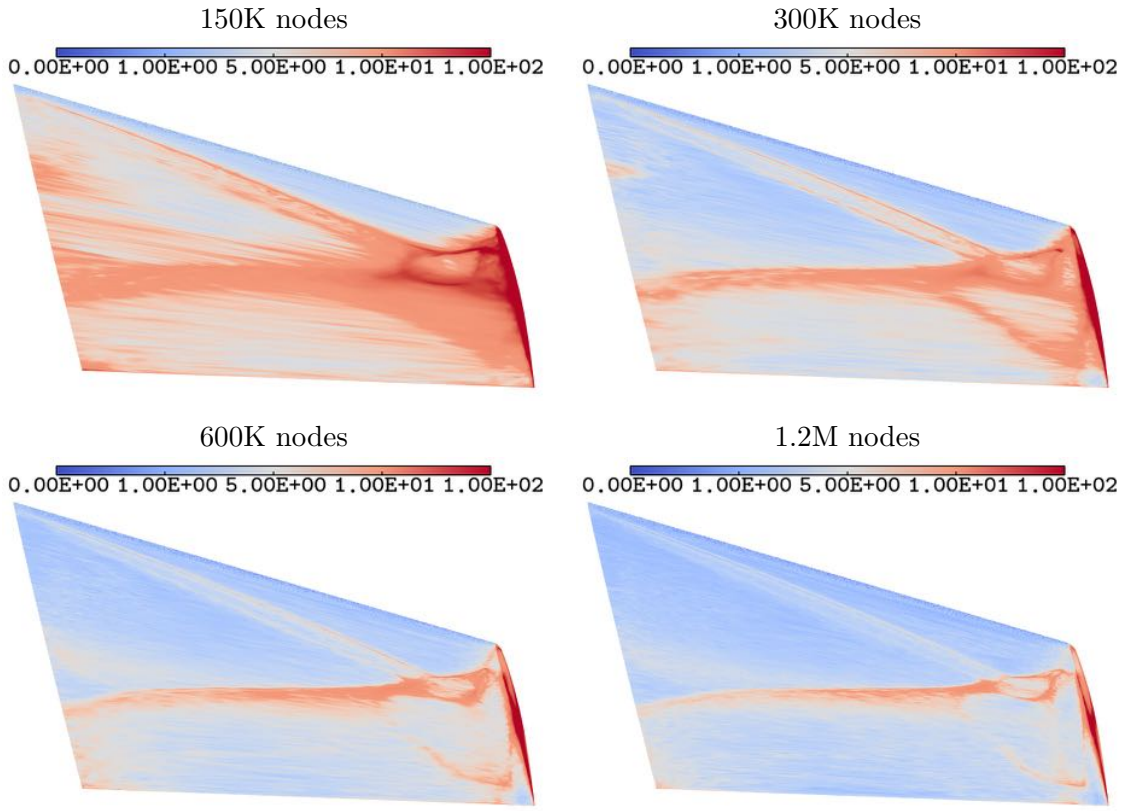


Figure 6.21: Adapted RANS M6 wing: Dimensionless wall-spacing y^+ of the first layer of elements of the meshes generated with adjoint based mesh adaptation on the drag.

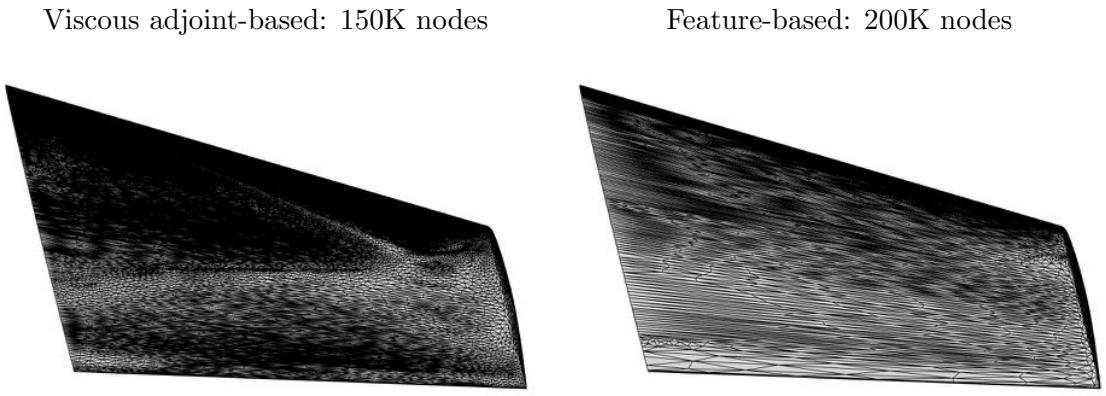


Figure 6.22: Adapted RANS M6 wing: Surface adapted meshes obtained with viscous adjoint-based (left) and feature-based (right) mesh adaptation.

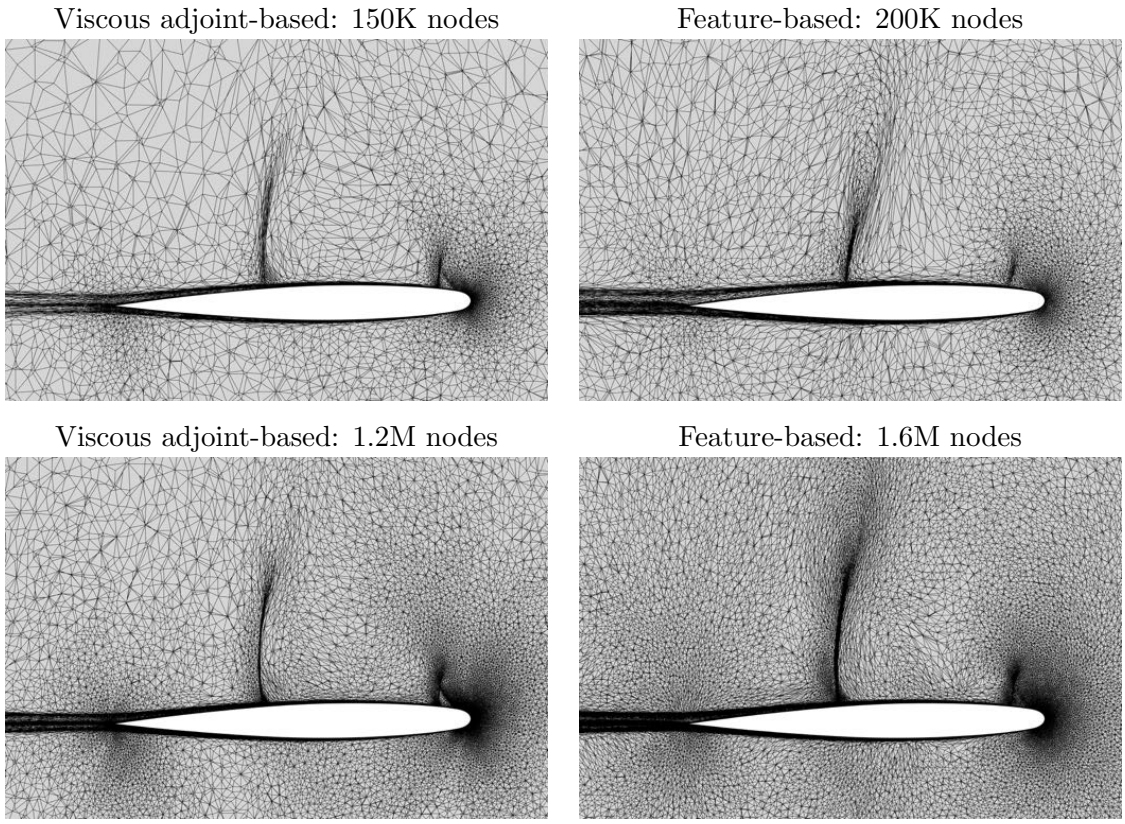


Figure 6.23: Adapted RANS M6 wing: Cut at $y = 0.5$ in the volume of coarse meshes obtained with viscous adjoint-based (left) and feature-based (right) mesh adaptation. Coarse meshes (top) compared to finer meshes (bottom).

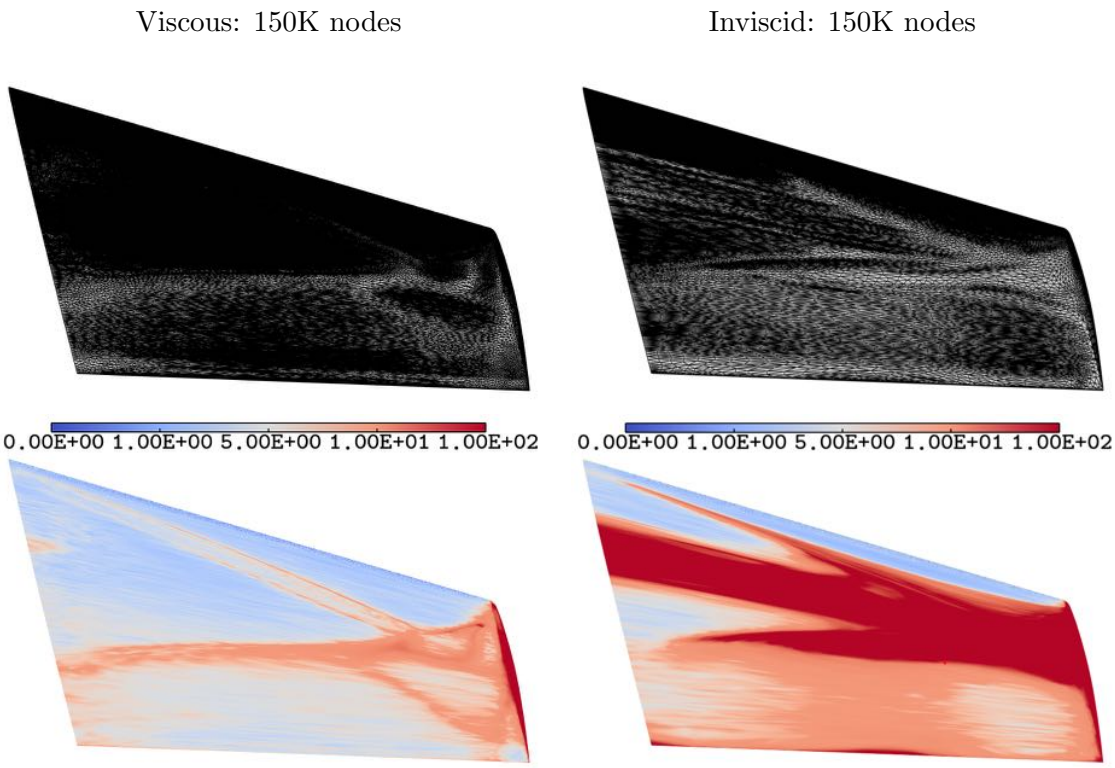


Figure 6.24: Adapted RANS M6 wing: Surface adapted meshes (top) and dimensionless wall spacing (bottom) obtained with viscous (left) and inviscid (right) adjoint based mesh adaptation.

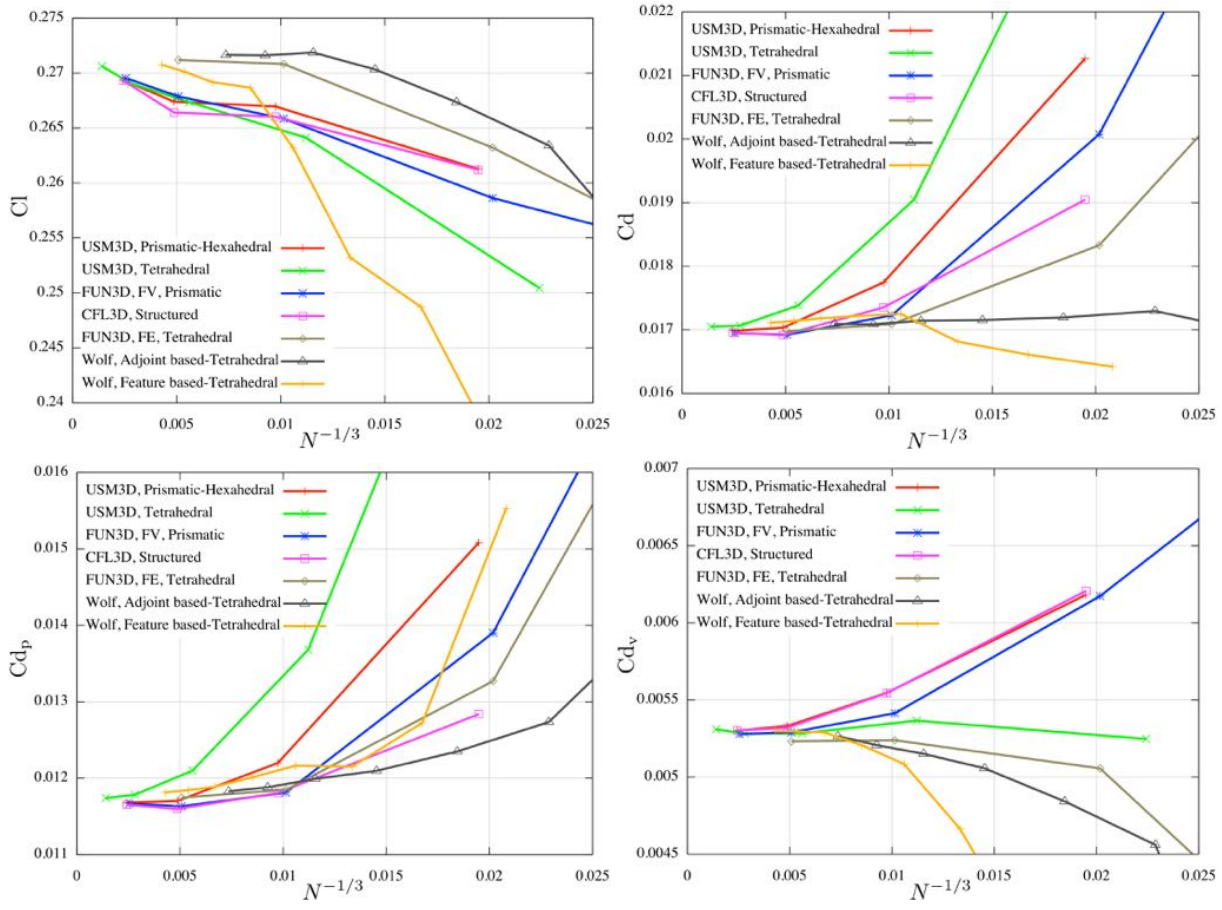


Figure 6.25: Adapted RANS M6 wing: Comparison of the convergence of the lift (top left), drag (top right) and pressure (bottom left) and viscous (bottom right) components of the drag predicted by `Wolf` with feature-based and adjoint based mesh adaptation to other solvers (USM3D,FUN3D-FV,FUN3D-FE,CFL3D).

6.5 Applied cases: Trap wing

The aircraft high-lift configurations computations are interesting and challenging cases for mesh adaptation strategies. The purpose of these simulations is to accurately predict the lift of an aircraft in high-lift configuration at low speed, *i.e.* upon take-off or landing. More importantly, prediction of the maximum angle of attack and the associated lift before stall is critical for the design of the aircraft as it determines the maximal take-off weight. Depending on the angle of attack they involve several physical features difficult to predict *a-priori* such as shear layers, wakes and detachments, that have large effects on the solution.

6.5.1 Case description

Figure 6.26: NASA Trap wing geometry.

The trap wing geometry is a simple three elements high-lift swept wing mounted on a simple body as shown in Figure 6.26. It was studied during the first AIAA high-lift prediction workshop [Rumsey 2011], both with numerical simulations and wind tunnel measurements. The objectives of this workshop was to assess the prediction capabilities of numerical simulations, deduce guidelines for computations and further area of research. The results of this workshop are public on the NASA website[1] and thus constitute an excellent case to validate and compare the capabilities of RANS mesh adaptation to standard solvers and mesh generation tools.

We focus here on the first case with slats deflected at 30° and flaps at 25°. The initial geometry was provided in inches so we scaled it up to meters. The wing is considered in the following conditions:

[1] https://hiliftpw.larc.nasa.gov/index-workshop1.html

Mach number	0.2
Reynolds number	4.3×10^6
Reference Length	1.0067036
Reference temperature	288.9
Angles of attack	3, 6, 13, 28, 33°

Turbulence model	SA Negative	Riemann Solver	HLLC
MUSCL extrapolation	Order 2 - V4	Limiter	Piperno
Cells	Median	Gradients reconstruction	L^2-projection
Renumbering	BFS	Initial CFL	1
CFL progression	Geometric: 1.05	CFL_{max}	1000
Freeze limiter iteration	1000	Max Solver iterations	1300
Targeted physical residual	10^{-8}	Targeted residual reduction	10^{-4}
Linear solver	SGS	Max linear iterations	50
Targeted linear residual	0.05		
Aero reference surface	2.0464682	Mean Aerodynamic Chord	1.0067036
Center of pressure	$\begin{pmatrix} 0.8722868 \\ 0.02413 \\ 0.0 \end{pmatrix}$		

Table 6.5: Trap wing: Solver parameters.

The solver parameters used to solve these cases are summed up in Table 6.5. The different mesh adaptations are performed either with the standard feature-based mesh adaptation based on the Mach field or with the viscous goal-oriented strategy on the lift coefficient. Viscous terms are directly weighted by the hessian of the adjoint, through a full linearization of the equations in primitive variables (see Section 6.1.2). As for the Onera M6 wing, we use the global strategy described in Section 6.2, the complexity is increased at each step and up to twenty sub-iterations are performed at fixed complexities: 200 000, 400 000, 800 000, 1 600 000 and 3 200 000 vertices. The next step is computed as soon as the lift varies from less than 0.5% between the three last sub-iterations. As for the Onera M6 wing, the initial mesh is coarse and contains no boundary layer mesh as depicted in Figure 6.27.

6.5.2 Results

Figure 6.28 shows the surface mesh of the 1.6 M nodes adapted mesh generated with the goal-oriented strategy on the lift coefficient. Figures 6.29 and 6.31 show cuts of the mesh and solution fields at 28% of the spawn of the wing. In Figure 6.31, we can see how the mesh has been adapted to the flow, discretizing as in 2D the shear layer and the boundary layer (see Section 6.7.2).

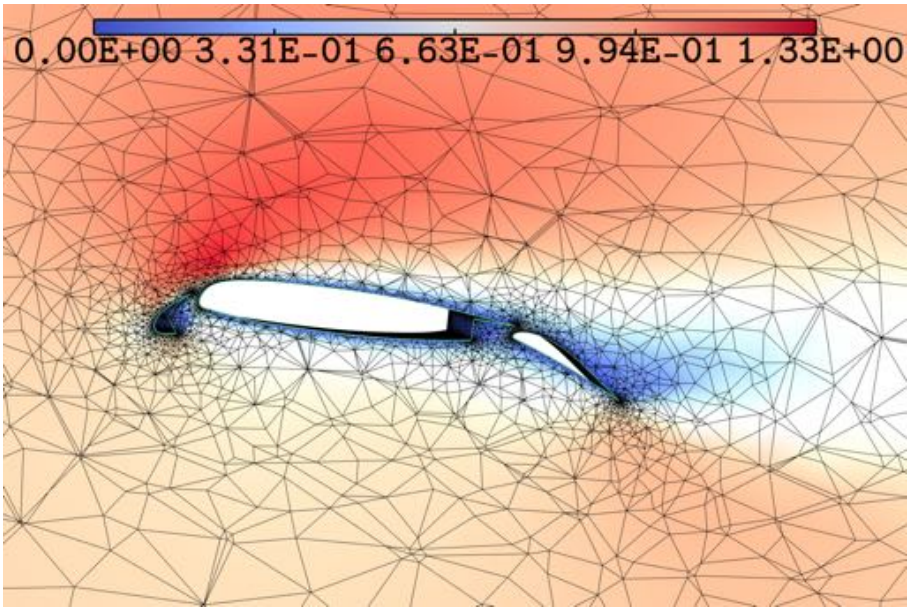


Figure 6.27: NASA Trap wing: Initial mesh and velocity field.

We will see in Section 6.7.2 that the correct computation of the shear layer was essential to the proper prediction of the flow on the flaps. Figure 6.31 also shows the tremendous advantage of using full unstructured mesh adaptation, including the boundary layer mesh as the junction of structured boundary layer mesh in such enclosed geometries is generally poor. Here, we can see that the wake is seamlessly discretized from the slat to the main wing.

Wind tunnel measurements were also lead for the first high-lift prediction workshop. In Figure 6.32 we compare datas, the pressure coefficients predicted by wolf on the wing at 28 and 85 % to these experimental. We observe a fairly good agreement of the numerical prediction with the experimental results. Similarly, Figure 6.30 shows, for different angles of attack, the lift coefficient predicted by `Wolf` on adapted meshes generated with the viscous goal-oriented strategy on the lift. Each of these meshes has been obtained with a separate adaptation process and is composed of about 6M nodes. The other entries in Figure 6.30 were provided by the different participants of the high-lift workshop and have been computed on meshes of about 20M nodes. Figure 6.30 shows again a fairly good agreement of `Wolf` with the experimental measurements and other participants, although it seems to slightly underestimate the lift coefficient.

As can be seen in Figure 6.33, this is probably due to the unsufficient mesh convergence. Indeed, these computations were lead with multi thread parallelization on machines with 28 CPUs and 63,5 Go RAM memory. Hence, we were not able to reach the same mesh sizes. However, Figure 6.33 shows that `Wolf` on adapted meshes performs rather like the best participants and already predicts on a 6M nodes adapted mesh the same lift as on hand made best practice meshes with 20M nodes. Anyway, as for the Onera M6 wing, Figure 6.34 depicts the tremendous improvement brought by viscous goal-oriented mesh adaptation when compared to feature-based mesh adaptation, leading to an early convergence of the solution.

These results may seem disappointing as adapted meshes do not seem to bring a particular

improvement compared to hand made best practice meshes, even though they are adapted for each angle of attack. Maybe it is because, the trap wing geometry is extremely simple and it is thus possible to generate a good mesh with standard procedure. This will not be true on more complex geometries like the HLCRM (see Section 6.6). However, one should keep in mind that these adapted meshes are generated automatically in the adaptive process, without any human time spent to prescribe, generate and verify the different meshes. The overall process to generate best practice meshes generally takes several weeks.

Moreover the dispersion of results at 28° (and generally at high angle of attack) highlights another strength of mesh adaptation strategy: its robustness. Several participants to the workshop reported difficulties to start computations at high angles of attack and the necessity to initialize them with previous computations at lower angle.

Figure 6.28: Trap wing: Top view of the 1.6 M nodes adapted surface mesh generated by the viscous adjoint-based adaptation on the lift.

Finally, Figure 6.28 presents a comparison of the mesh convergence of the lift obtained with the feature-based adaptation using the Mach field and the goal-oriented mesh adaptation on the lift. Here again this highlights the tremendous improvement brought by goal-oriented mesh adaptation strategy. Figure 6.35 gives an interesting insight of the phenomena taking place here. We can see on meshes generated with feature-based adaptation that much ressources are spend in the wake which is not the case for meshes generated with goal-oriented mesh adaptation. Hence, we can see that the boundary layer that is initially detached on the whole wing, is progressively reattached on the main element and then on the flap, with the consequence to

progressively increase the lift.

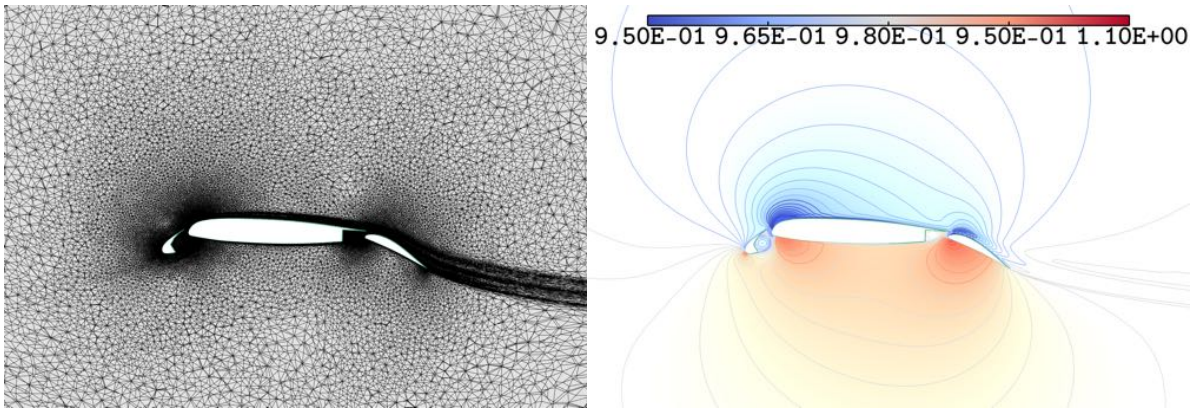


Figure 6.29: Trap wing: Cut in the volume at 28% of the spawn of the wing. General view of the mesh (top) and density field (bottom).

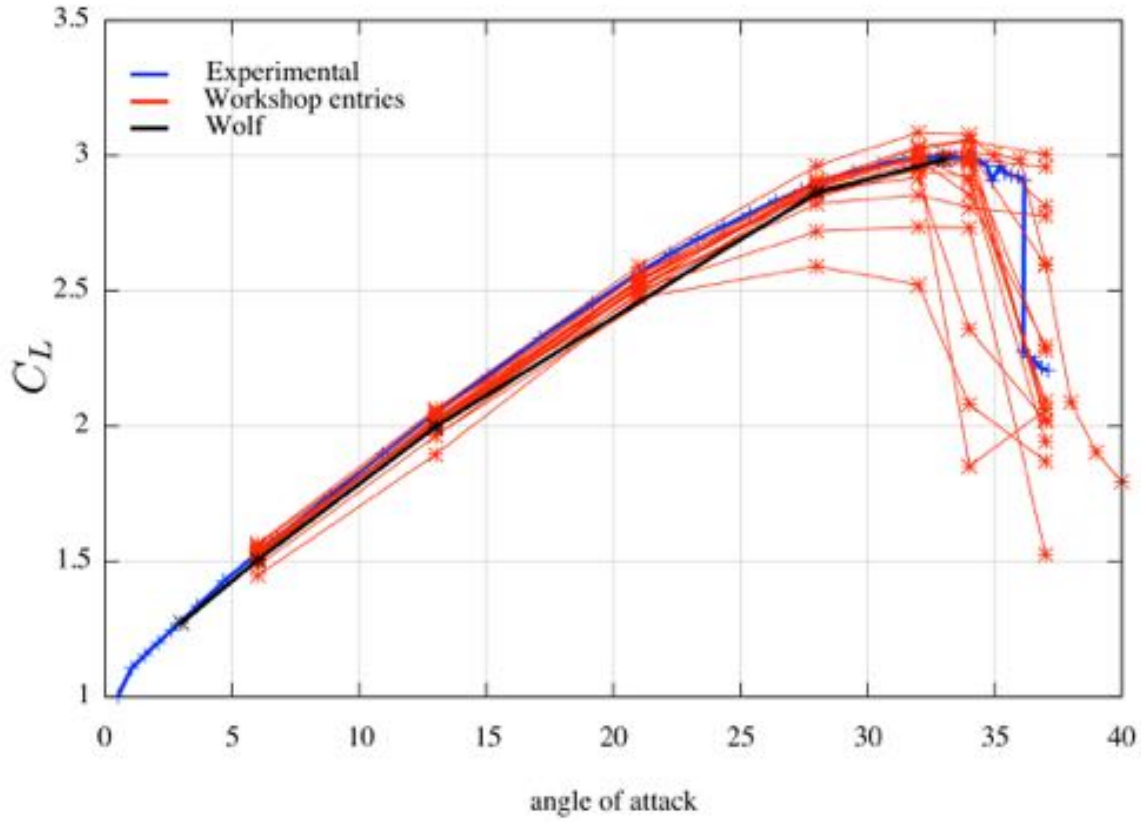


Figure 6.30: Trap wing: Lift coefficient at different angles of attack predicted by `Wolf` and other codes, compared to experimental results.

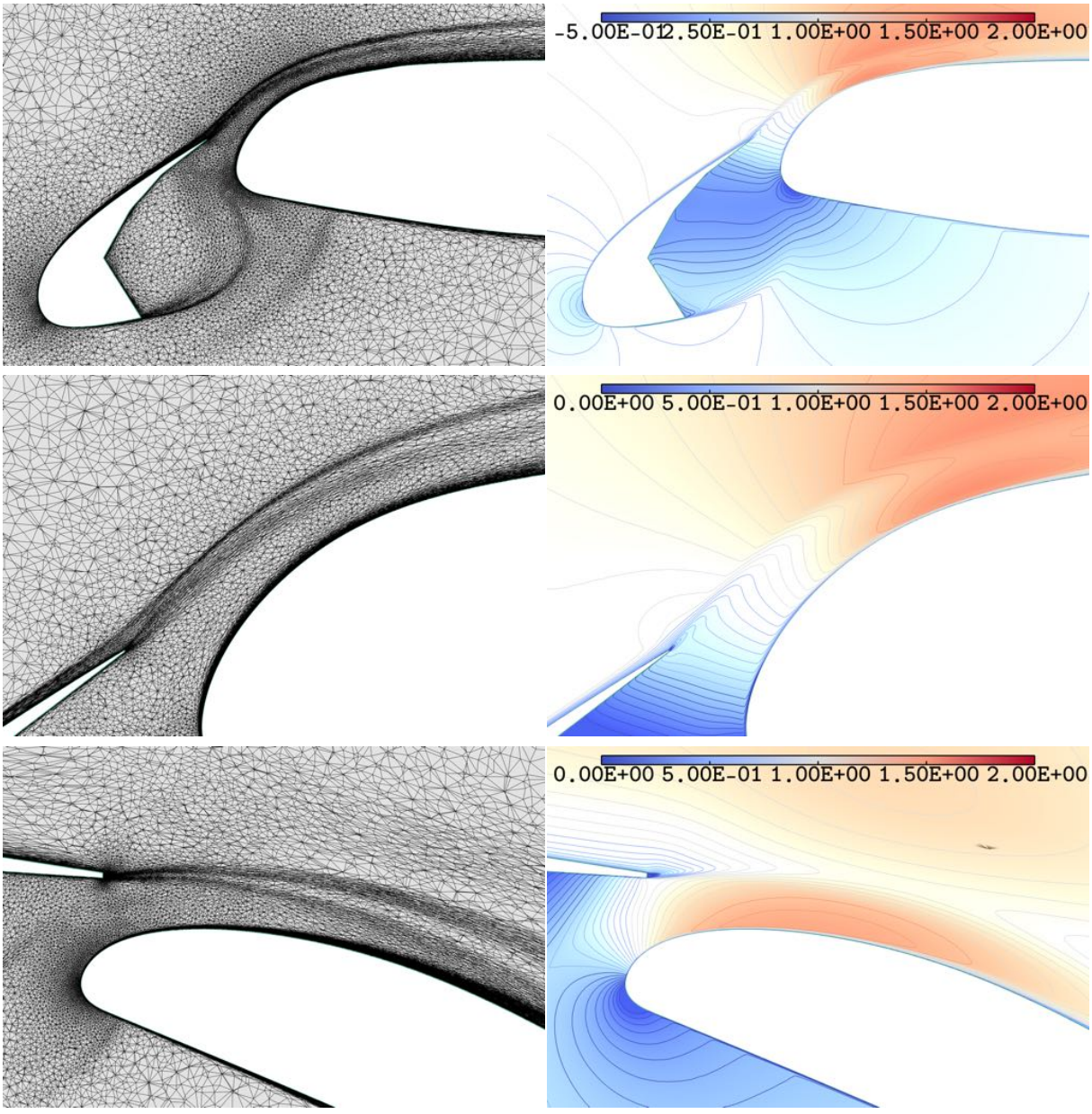


Figure 6.31: Trap wing: Cuts in the volume at 28% of the spawn of the wing. Meshes (left) and X-component of the velocity field (right) in close-up views on the slat region (top), shear layer (middle) and flap region (bottom).

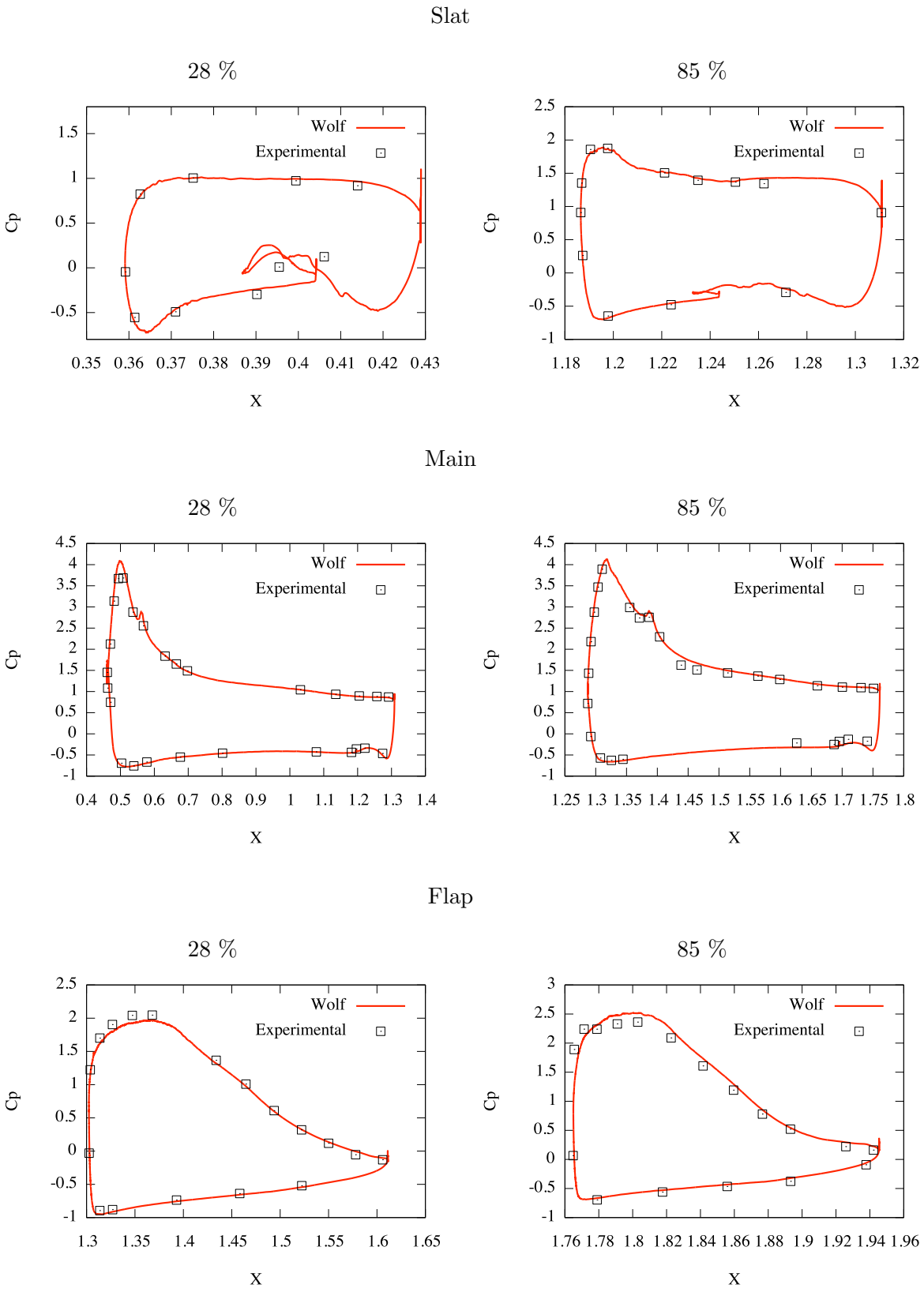


Figure 6.32: Trap wing: Pressure coefficient predicted by `Wolf`, compared to experimental measurements at 28 % and 85 % of the wing.

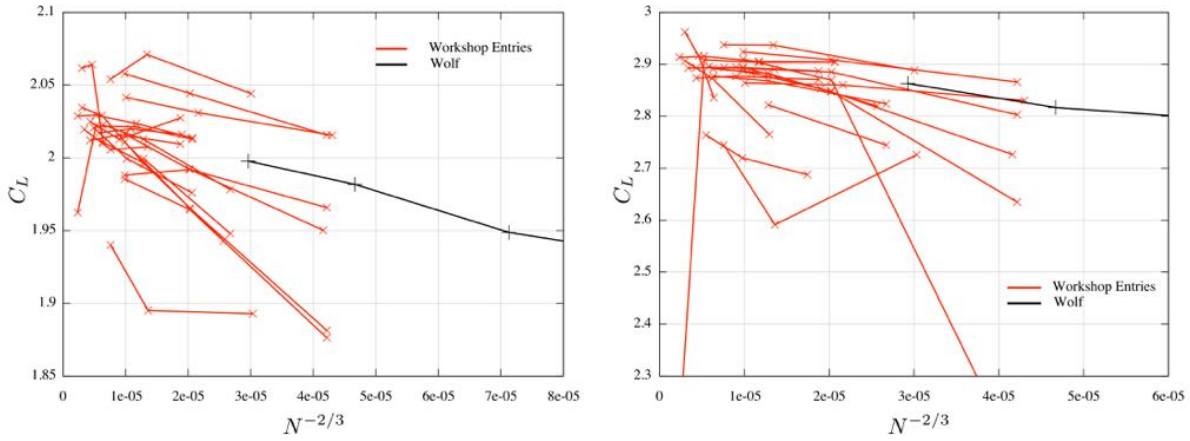


Figure 6.33: Trap wing: Mesh convergence study of the lift predicted by `Wolf` at 13° (left) and 28° (right) compared to other participants.

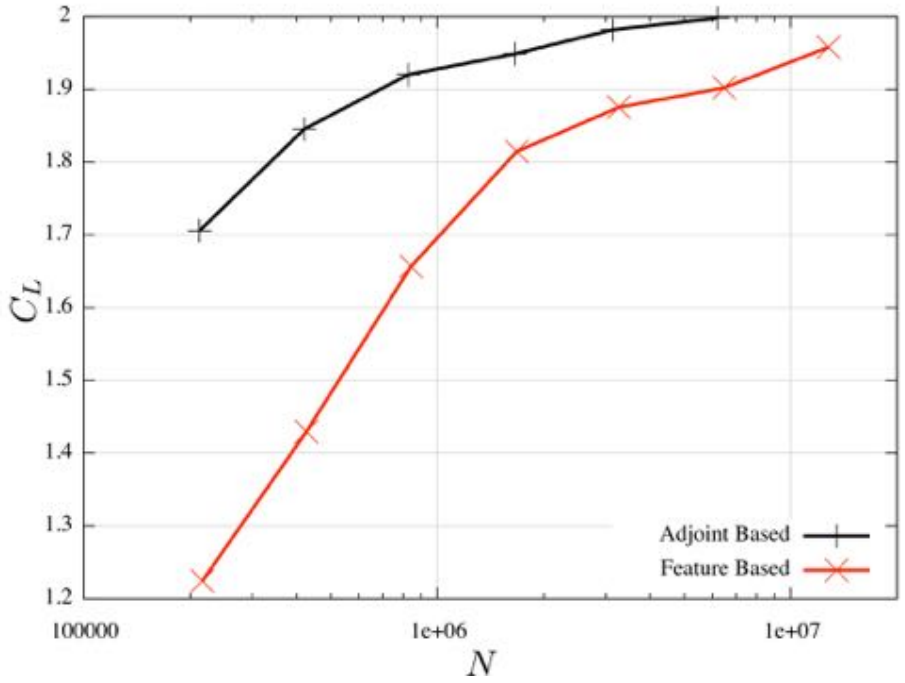
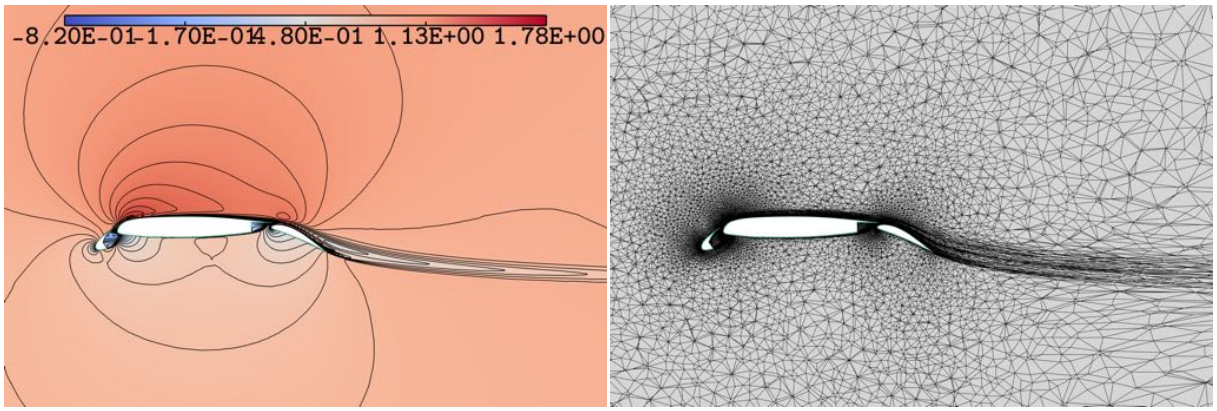
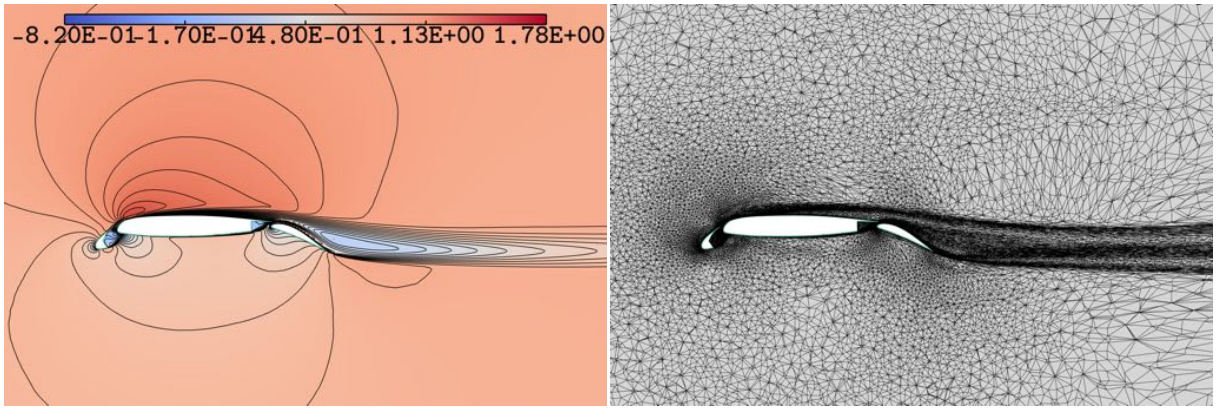


Figure 6.34: Trap wing: Convergence of the lift with feature-based and goal-oriented mesh adaptation.

Adjoint based: 800K nodes



Feature-based: 800K nodes



Feature-based: 1600K nodes

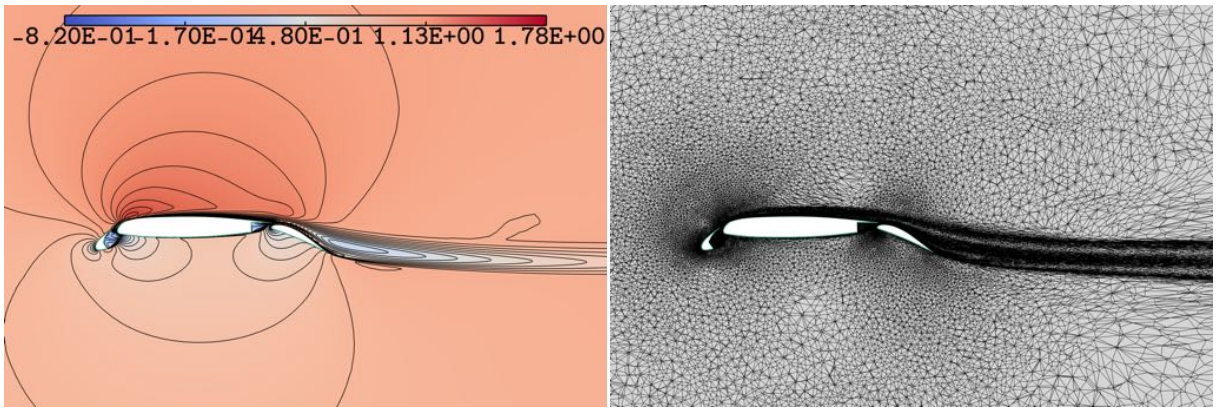


Figure 6.35: Trap wing: Cut in the volume at 28% of the wing. Density field (left) on meshes (right) obtained with goal-oriented adaptation on the lift (top) and feature-based adaptation using the Mach field (middle and bottom).

6.6 Applied cases: High-Lift Common Research Model

The geometry described in Section 6.5 is a simplified high-lift wing. The slats and flaps extend over the whole spawn so that the geometry is almost the extrusion of a 2D profile. It is thus relatively easy to generate a reasonably good mesh to properly discretize the flow. Still, the first high-lift workshop exhibited a large spread in the results for high angles of attack.

We will study in this part the High-Lift Common Research Model (HLCRM) geometry that was used for the third AIAA high-lift workshop [Rumsey 2018]. This geometry, shown in Figure 6.36, is more realistic, slats and flaps does not extend over the whole spawn and the flaps are split in two parts with a gap in-between (see Figure 6.37). It is thus more complex to generate a proper mesh in all the regions of interest for this geometry every where as we will see.

The initial geometry was provided in inches so we scaled it into meters. We consider the following reference physical quantities to define our flight conditions:

Density	$1.22599\ kg.m^{-3}$
Velocity	$(67.3785, 0.000, 9.469437) == 68,04\ m.s^{-1}$
Pressure	$101352.99\ Pa$
Dynamic viscosity	$1.792510 \times 10^{-4}\ Pa.s^{-1}$
Temperature	$288,15K$
Length	$7.00532\ m$
Surface	$191,84477\ m^2$
Sound velocity	$340,2\ m.s^{-1}$
Angle of attack	$8°$
Mach number	0.2
Reynolds	3.26×10^6

The solver parameters used to solve these cases are summed up in Table 6.6. The different mesh adaptations are performed with the viscous goal-oriented strategy on the lift coefficient. Viscous terms are directly weighted by the hessian of the adjoint, through a full linearization of the equations in primitive variables (see Section 6.1.2). As for the trap wing, we use the global strategy described in Section 6.2, the complexity is increased at each step and up to twenty sub-iterations are performed at fixed complexities: 200 000, 800 000, 3 200 000, 6 400 000 and 12 800 000 vertices. The next step is computed as soon as the lift varies from less than 0.5% between the three last sub-iterations.

The main conclusion of the third high-lift workshop was that computations were not mesh converged even with the 236M nodes grid. As can be seen in Figure 6.43, there is a huge dispersion of the predicted lift, even at 8° angle of attack. Many contributors reported a high mesh dependency of the solutions, making it difficult to discuss the influence of turbulence models. Several authors proposed different modifications to the meshes generated in order to improve the predictions, among which:

Turbulence model	SA Negative	Riemann Solver	HLLC
MUSCL extrapolation	Order 2 - V4	Limiter	Piperno
Cells	Median	Gradients reconstruction	L^2-projection
Renumbering	BFS	Initial CFL	0.1
CFL progression	Geometric: 1.05	CFL_{max}	200
Freeze limiter iteration	1000	Max Solver iterations	1300
Targeted physical residual	10^{-8}	Targeted residual reduction	10^{-3}
Linear solver	SGS	Max linear iterations	30
Targeted linear residual	0.1		
Aero reference surface	191.84477	Aero reference length	7.00532
Center of pressure	$\begin{pmatrix} 33.655 \\ 11.8872 \\ 4.4958 \end{pmatrix}$		

Table 6.6: HLCRM: Solver parameters.

- increase the resolution of the wake
- increase the resolution of the upper surface of the slat and generally of the surface of the wing
- increase the resolution of the leading edge
- have a coarser discretization of the body
- smooth the junction of the structured boundary layers over the slat, main wing and flaps
- change the progression in the boundary layer mesh generation.

It is interesting to note that this is exactly what goal-oriented mesh adaptation tends to do automatically. It means on one hand that viscous goal-oriented mesh prescriptions are coherent with what is observed in practice and in the other hand, viscous goal-oriented mesh prescriptions could be used to improve best practices mesh generation guidelines. As can be seen in Figure 6.40-(top), the different structures in the wake are automatically refined rather precisely, the discretization of the slat and the main wing (Figure 6.40-(middle and bottom)) is very fine compared to the body. Figure 6.38 shows a transversal cut of the junction of the mesh between the slat, the main wing and the flap, in the hand-made mesh and adapted mesh. We can clearly see the contrast between the fine discretization of the shear layers in the adapted mesh and their absence in the hand-made mesh. Knowing the importance of the capture of these phenomena, we can easily understand the improvement obtained when smoothing the boundary layer meshe junction.

Similarly, Figure 6.39 shows a cut in the mesh over the wing near the flap gap and over the tip of the wing. We observe the much finer discretization of the flap in the adapted case. The discretization of the shear layer is also clearly visible along side the vortical structures generated over the tip of the wing. Another ambition of the workshop was to study the physical phenomenon taking place in high-lift configuration flows. However, as noticed by some contributors, although these structures are successfully captured initially, they are rapidly dissipated by the not enough refined mesh over the wing (see Figure 6.39-(bottom)). Mesh adaptation successfully discretizes these features and guaranty their proper computation.

Another relevant detail is the jet that forms in the gap between the two flaps and lead to a large detached zone. As can be seen in Figure 6.41, the discretization of this jet has a strong influence on the size of the detached zone, and in turn on the resulting lift. Looking at the difference between the mesh generated by goal-oriented mesh adaptation strategy and the one generated by a standard approach, we can easily foresee the strong mesh dependency of the flow in this region. Moreover, the generation of structured boundary layer in such enclosed gaps with sharp squared edges is a challenge on itself. Still, it is successfully automatically handled by mesh adaptation. Here again, a full gap and a partially sealed gap geometry were provided in order to study the influence of this gap flow. But, we can easily understand that nothing can be inferred as long as the variations of the solution due to the mesh dependency is larger than the modifications due to the geometry change.

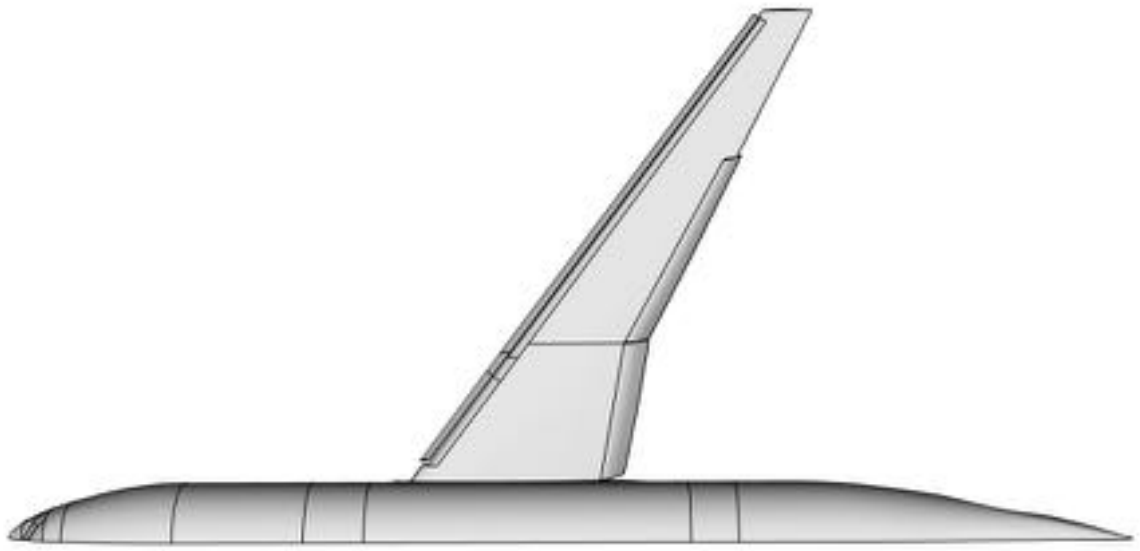


Figure 6.36: HLCRM: Top view of the geometry of the plane.

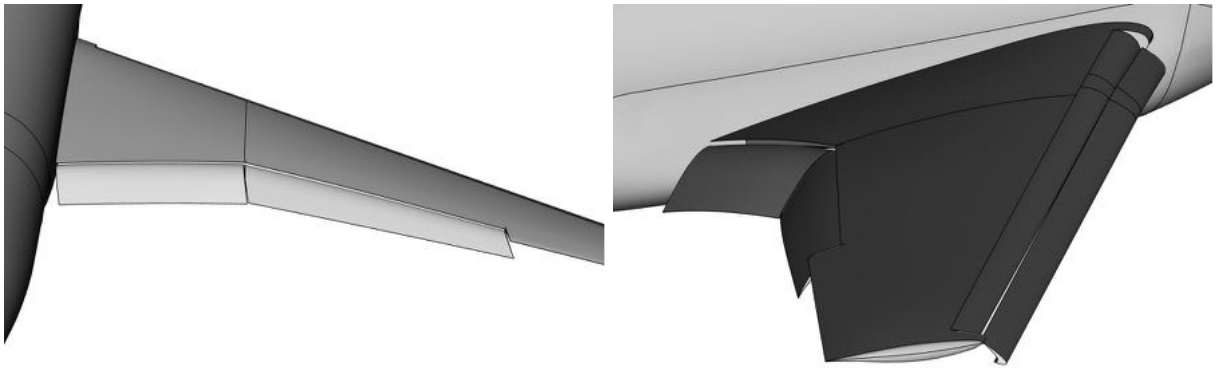


Figure 6.37: HLCRM: Back and lateral view of the wing geometry.

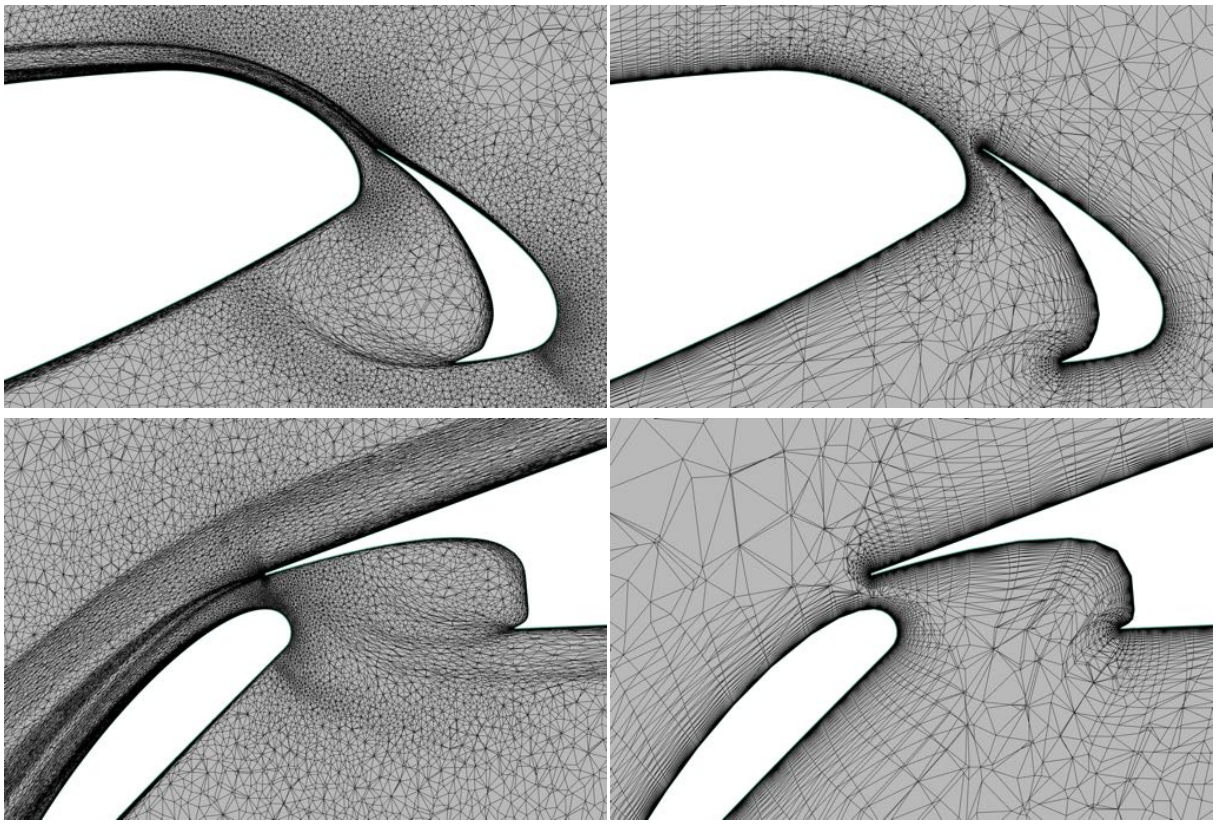


Figure 6.38: HLCRM: Cut in the volume in the mesh generated with goal-oriented mesh adaptation (30M nodes, left) and the standard mesh (20M nodes, right) at the leading and trailing edge of the main wing.

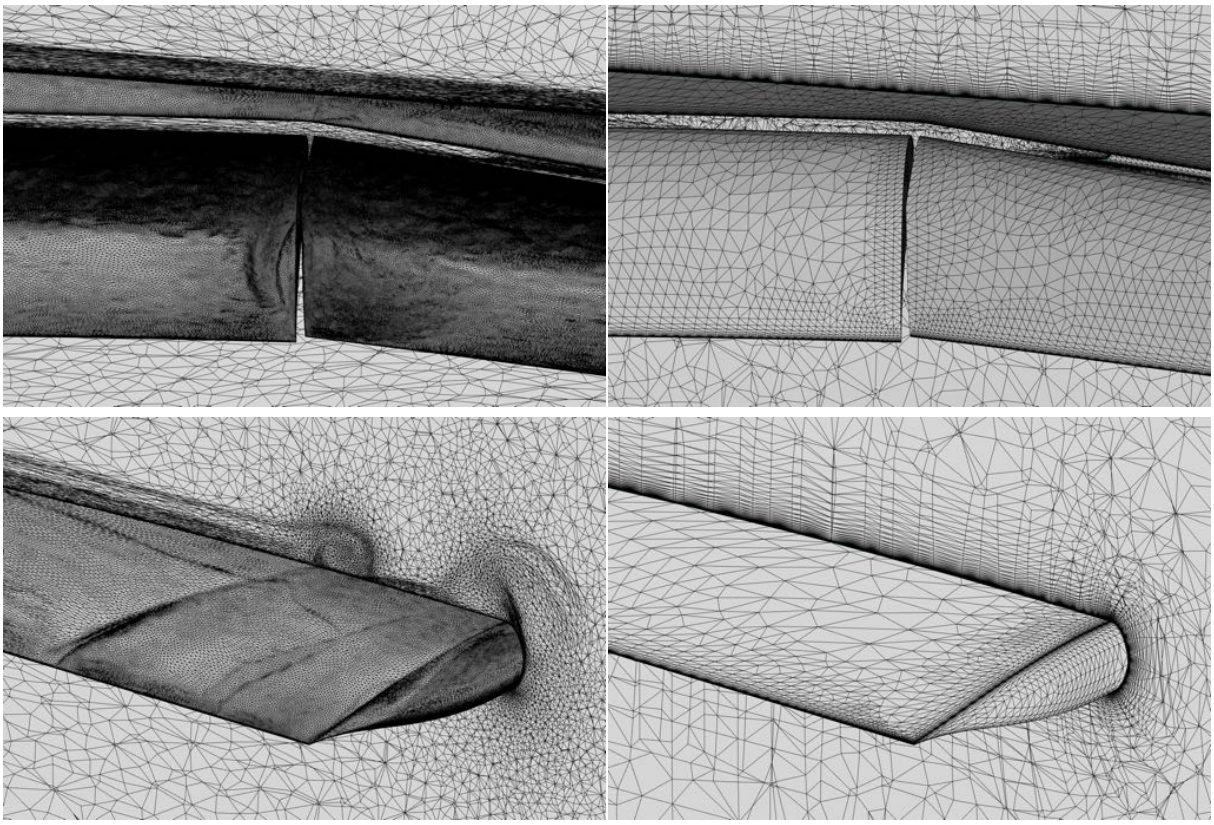


Figure 6.39: HLCRM: Rear view of the surface mesh and vertical cut of the mesh over the flaps gap and tip wing.

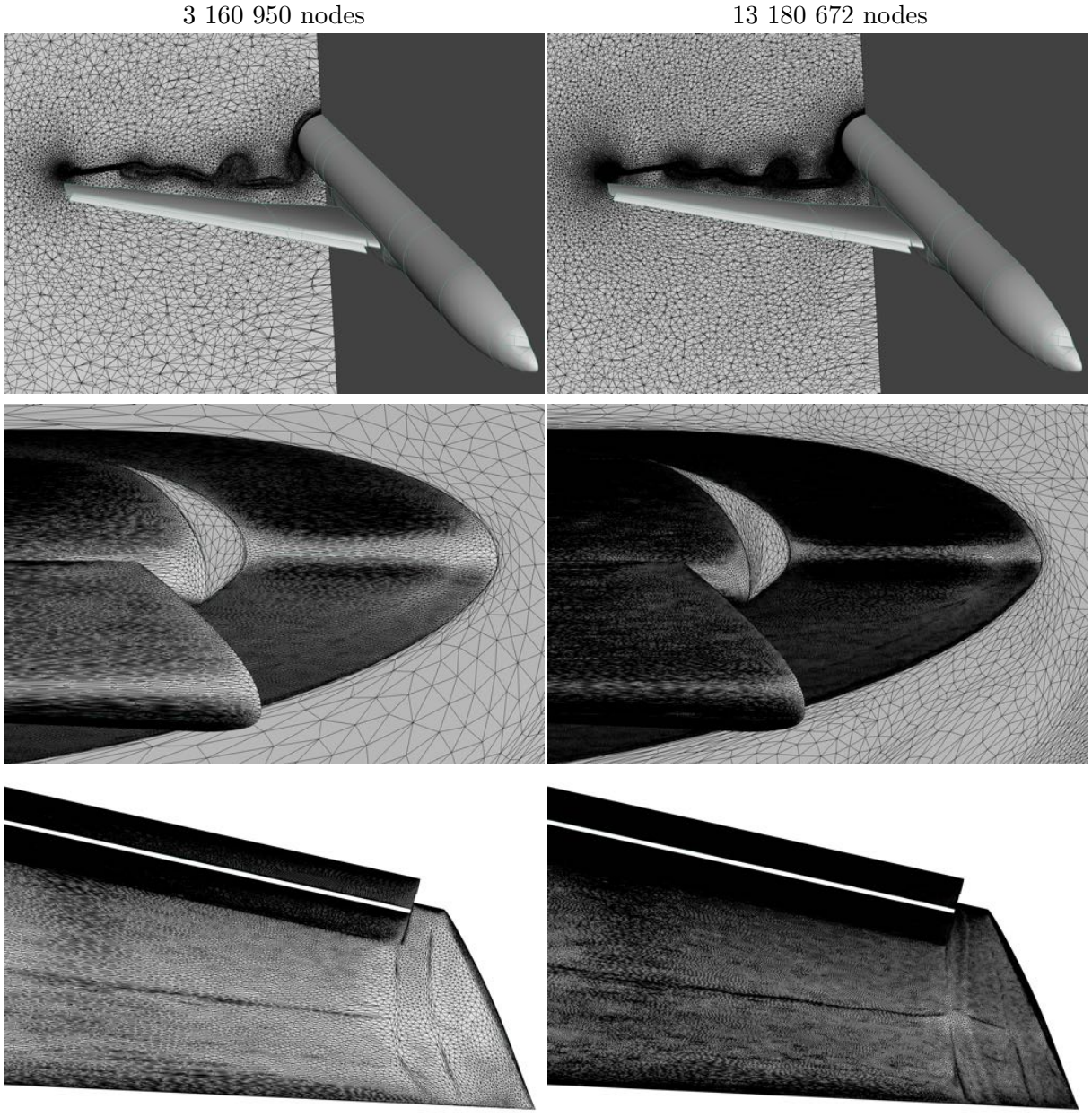


Figure 6.40: HLCRM: Cut in the volume mesh in the wake (top), surface mesh on the slat (middle) and on the tip of the wing (bottom).

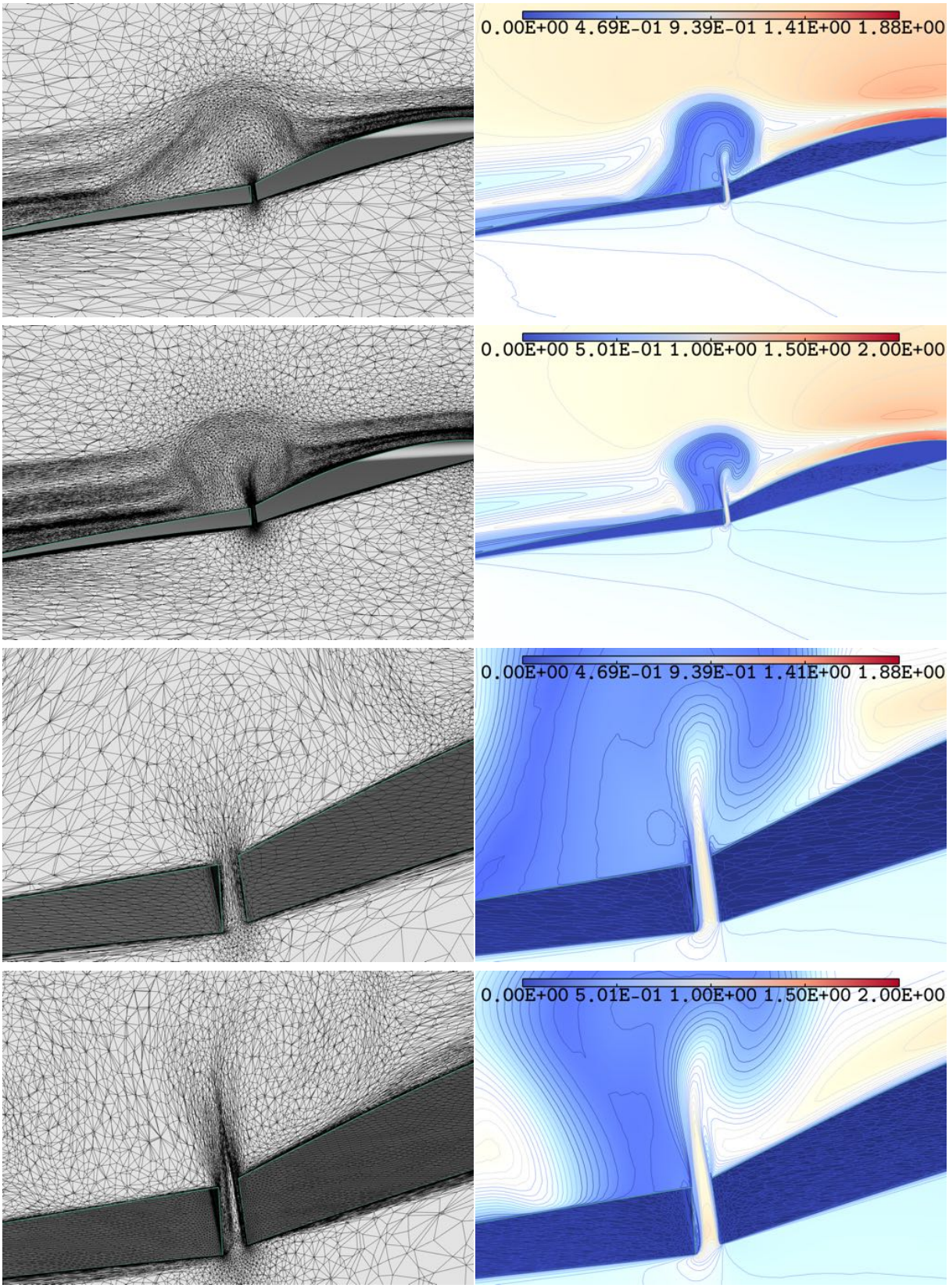


Figure 6.41: HLCRM: Convergence of the mesh (left) and z-component of the velocity field (right) in a cut over the flaps gap. From top to bottom, 13M nodes mesh, 30M nodes mesh, zoom in the 13M nodes mesh, zoom in the 30M nodes mesh.

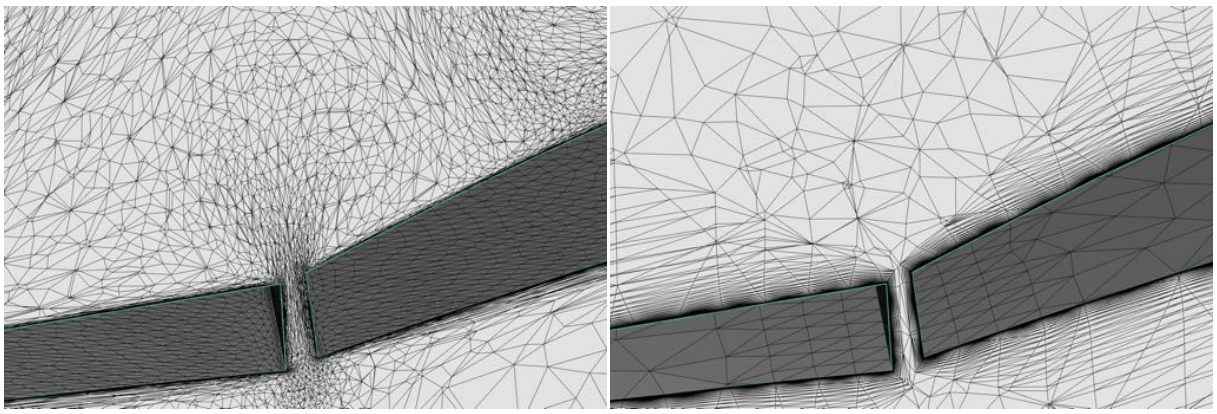


Figure 6.42: HLCRM: Comparison of the goal-oriented adapted 13M nodes mesh (left) and traditional hand-made 20M nodes mesh (right) over the flaps gap.

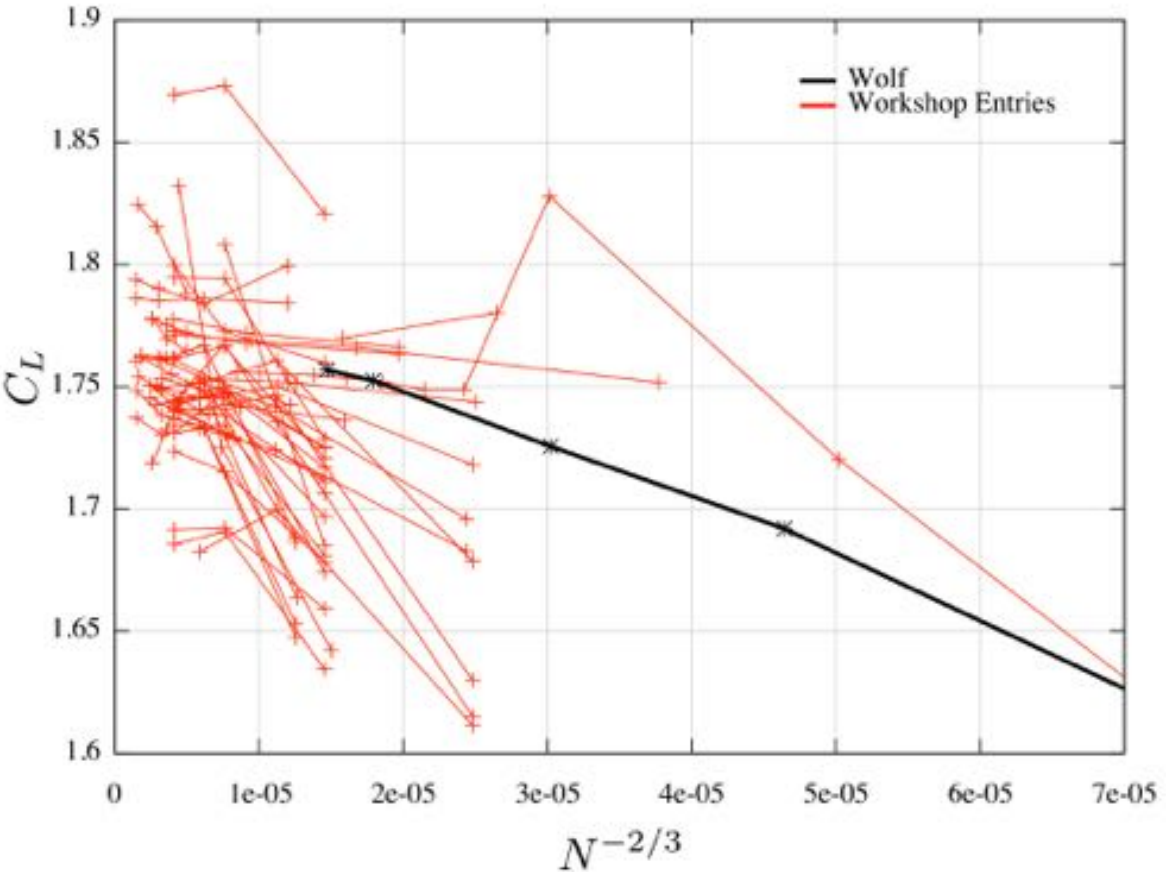


Figure 6.43: HLCRM: Convergence of the lift predicted by `Wolf` on adapted meshes generated with goal-oriented mesh adaptation strategy on the lift coefficient, compared to other participants of the workshop.

6.7 Improvement of RANS mesh adaptation

In the previous sections, we saw that we were able to successfully generate adapted meshes for turbulent RANS simulations using feature-based and adjoint-based approaches. Still, the convergence is underwhelming compared to what we could expect. This may be due to the presence of boundary layers that are slowly detected and adapted by the error sensors. We propose here an analysis of this phenomenon and some solutions to alleviate it using wall functions introduced in Section 2.3.

6.7.1 Adaptation with Wall-Laws: Subsonic NACA0012 validation

We first investigate the mesh adaptation computation of the flow around a NACA0012 using wall functions and standard wall resolution. The flow considered has the following characteristics:

Mach number	0.5	Angle of attack	1°
Reynolds number	5×10^5	Reference temperature	$300K$

We use the general mesh adaptation loop described in Section 6.2 with feature-based mesh adaptation strategy, using the Mach field as sensor. For the present study we started with an initial mesh complexity of 2500 vertices and multiplied the complexity by two at each step. For each step we perform five sub-steps to converge the mesh/solution couple for the given complexity.

We compare the solutions obtained with the low-Reynolds Spalart-Allmaras turbulence model with:

- a standard wall resolution,
- wall functions with a fixed distance of $d = 1 \times 10^{-4}$,
- wall functions with a fixed distance of $d = 1 \times 10^{-5}$,
- wall functions with a fixed distance of $d = 1 \times 10^{-6}$,
- adaptive wall functions.

The meshes and solutions obtained with feature-based mesh adaptation strategy using the standard wall resolution are shown in Figure 6.44. We clearly see the negative feedback between mesh adaptation and the low-Reynolds turbulence model. The insufficient discretization near the wall leads to a too viscous boundary layer that grows too rapidly compared to the reference, properly discretized solution (Figure 6.44 bottom). In turn, mesh adaptation tends to produce coarser grids to match the thicker boundary layer, leading to an even poorer discretization. As can be seen in Figure 6.44, the mesh adaptation process initially captures the too coarse boundary layer which progressively gets thiner. The process still converges toward an appropriate discretization but this negative feedback leads to an artificially slow convergence of the solution.

This negative feedback is alleviated using adaptive wall functions. The same grid/solution convergence is shown in Figure 6.45. Adaptive wall functions treat coarse grids with a large wall distance so that the inner, stiff, part of the boundary layer is not resolved. The mesh resolution is thus sufficient to discretize the boundary layer, which in turn is not too viscous. This leads to a thiner boundary layer on which the mesh adaptation concentrates the nodes leading to an even better discretization of the boundary layer. As the elements size near the wall decreases, the wall distance computed by wall function decreases leading to a stiffer boundary layer and a thiner mesh size requirement. In other words, using a smaller wall distance matches the boundary layer profile closer to the wall where the solution varies more, it requires thus a finer mesh to capture these variations. But, if the discretization is insufficient for the variation of the solution, the boundary layer gets too viscous and thickens, generating larger elements, leading to a larger wall distance, *etc.* ... An equilibrium is found when, smaller elements would lead to a smaller wall distance and thus a much stiffer boundary layer profile for which their resolution would be insufficient.

Figure 6.46 shows the velocity profiles predicted by wall integration and wall functions with fixed distances close to the trailing edge. The wall distances have been fixed to $d = 10^{-4}$, $d = 10^{-5}$ and $d = 10^{-6}$, corresponding to dimensionless wall distances of about respectively $d^+ \approx 50$, $d^+ \approx 5$, $d^+ \approx 0.5$. As shown in Figure 6.44, we can see that wall integration yields a thicker boundary layer on coarse meshes. Moreover, the velocity profile is smooth, preventing the mesh adaptation from efficiently refining the mesh in this region. Similarly, wall function with $d^+ \approx 0.5$ behaves like a wall integration, leading to a thick boundary layer. As expected, wall functions with $d^+ \approx 5$ and $d^+ \approx 50$ provide better results, the second converging faster than the first one.

This is a rather favorable case for wall functions as the flow is attached with mild pressure gradients, it is not surprising to have good results with larger d^+ especially on coarse grids. However, in presence of larger pressure gradients it is critical to minimize d^+ in order to minimize their effect. Figure 6.47 shows how adaptive wall functions achieve both. In the left figure we can see how the velocity profile converges rapidly toward wall integrated solution from coarse grids. On coarse grids they behave like wall-functions with about $d^+ = 100$, immediately providing good predictions. On finer grids they degenerate toward wall integration, as can be seen in the right figure.

Figure 6.47 suggests we can further optimize the choice of the wall distance with respect to the mesh size. On the 613 064 nodes mesh, we can prescribe a smaller wall distance to be closer to wall integration as the discretization is sufficient. In facts it is also possible to switch to a standard wall integration on the boundary edges that reached a given minimal wall distance.

This approach gives satisfying results on both fine and coarse meshes. In particular, we saw that there was a negative feedback between the turbulence model and mesh adaptation process using wall integration. This is due to the fact that the mesh adaptation sensor is "unaware" of the existence of the boundary layer leading to its improper discretization. This is alleviated by adding some information about the boundary layer through the use of wall functions. From this observation, we developed a hessian boundary correction of the sensor for the wall integration cases in order to add the same positive feedback as with wall functions.

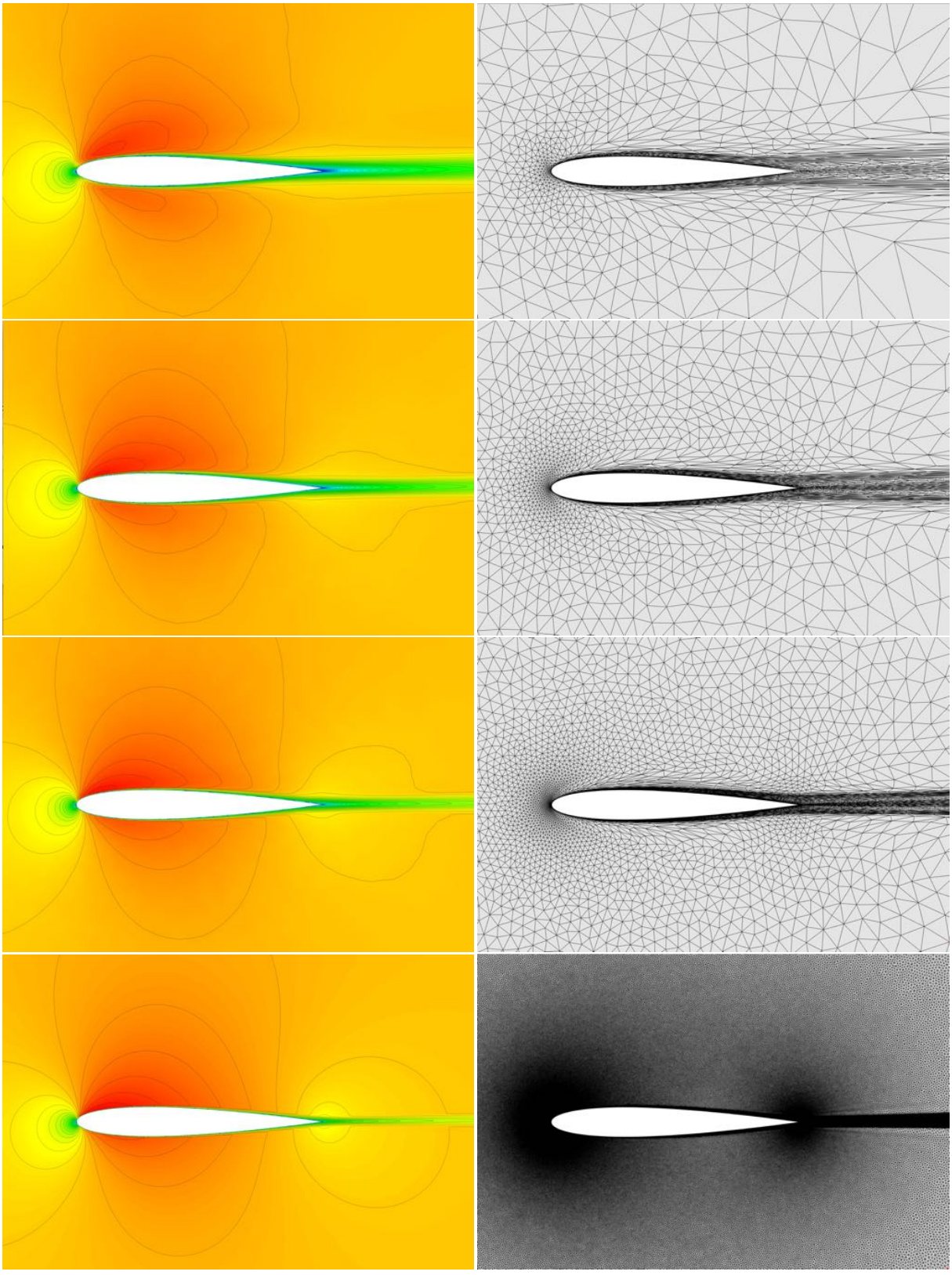


Figure 6.44: Subsonic NACA0012: Solutions (left) and adapted meshes (right) convergence with standard wall resolution. Meshes composed of: 2453, 4907, 9667 and 602715 vertices.

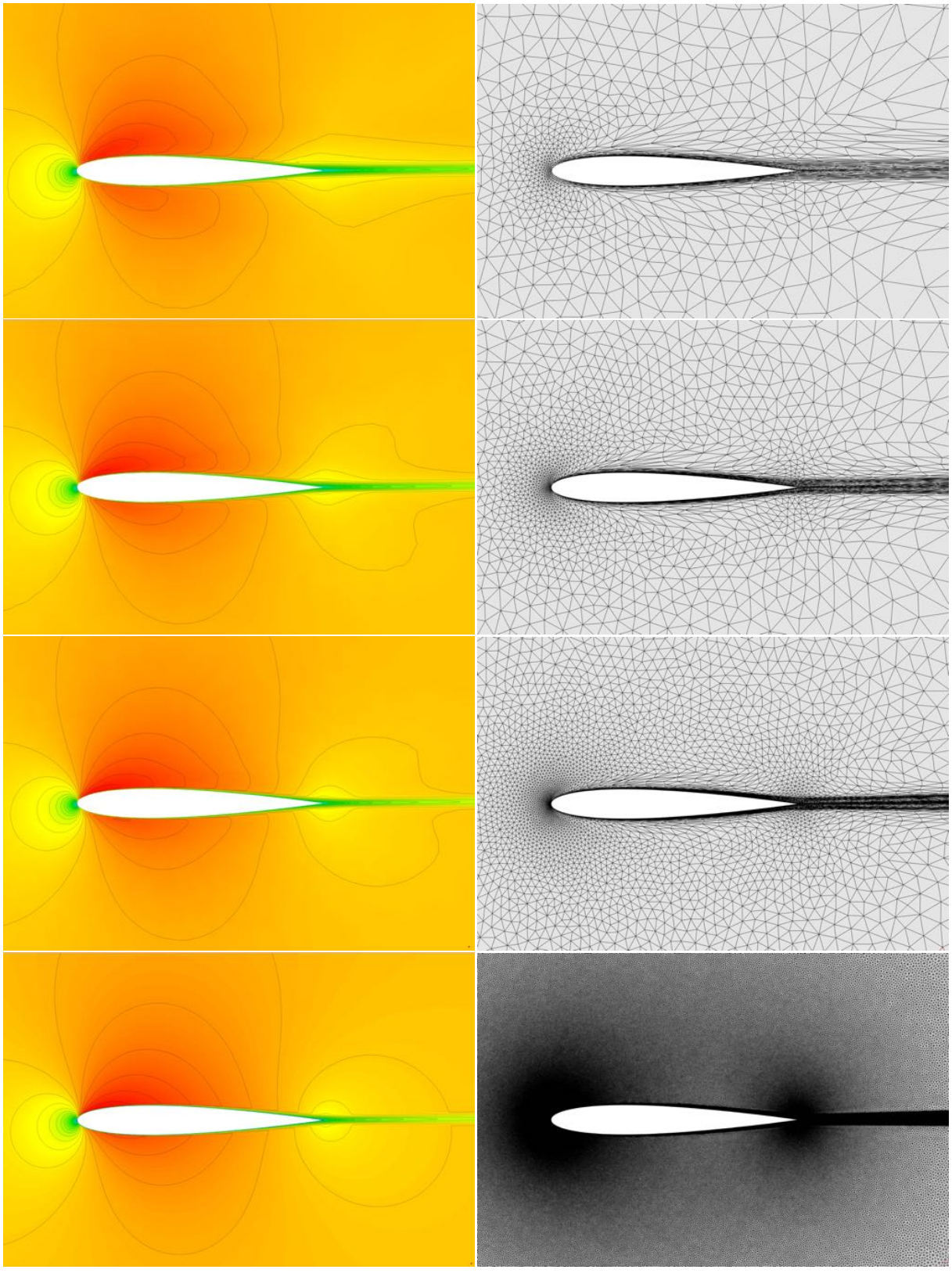


Figure 6.45: Subsonic NACA0012: Solutions (left) and adapted meshes (right) convergence with adaptive wall functions. Meshes composed of: 2522, 4813, 9536 and 613064 nodes.

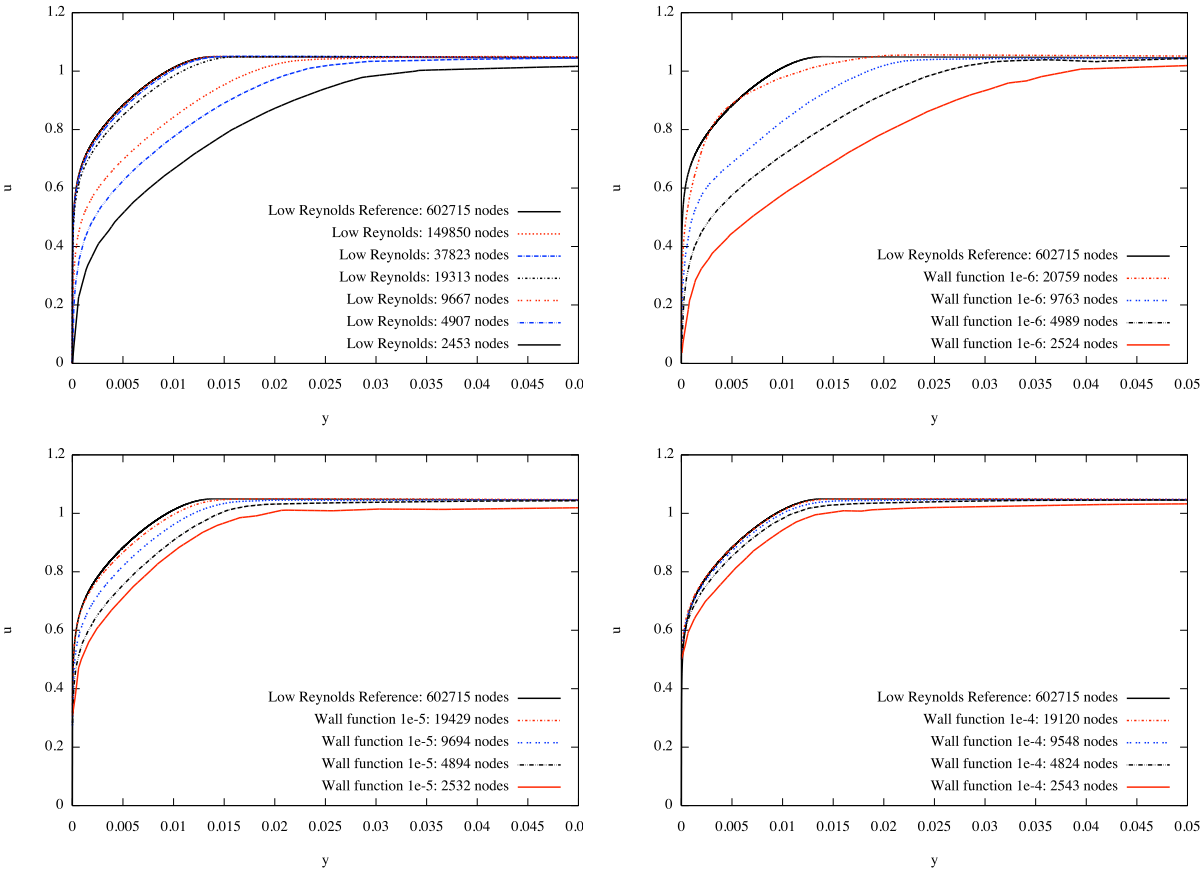


Figure 6.46: Subsonic NACA0012: Convergence of velocity profiles above the NACA0012 trailing edge using wall integration (top left) and wall functions with fixed wall distances of $d = 10^{-6}$ (top right), $d = 10^{-5}$ (bottom left) and $d = 10^{-4}$ (bottom right).

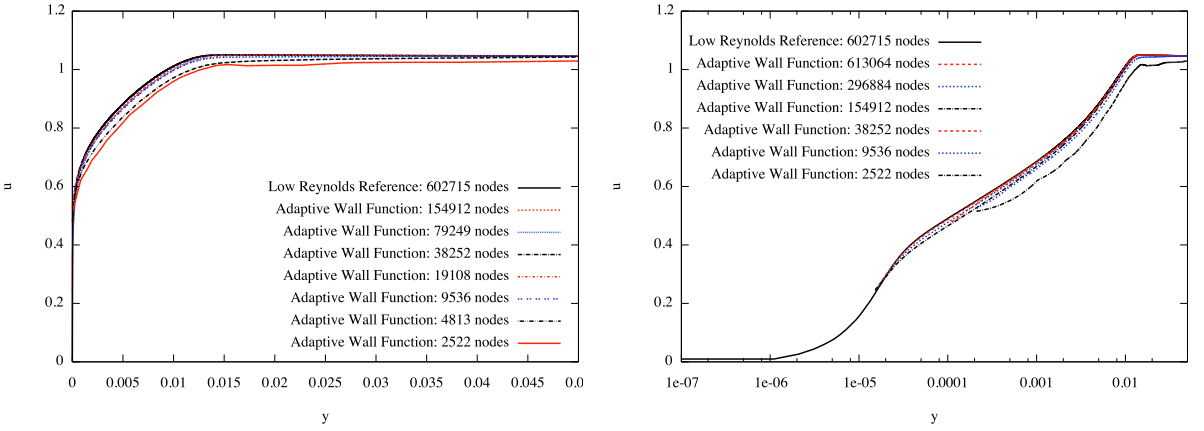


Figure 6.47: Subsonic NACA0012: Convergence of velocity profiles above the NACA0012 trailing edge using adaptive wall functions. Linear (left) and log (right) scales.

6.7.2 Hessian boundary correction: impact on a 2D High-Lift configuration

We have shown in the previous section that the under estimation of the error in the boundary layer tends to slow the convergence of RANS mesh adaptation strategy. This has already been highlighted and studied in Section 4.1.1. In this section, we derived a correction of the boundary hessian in order to improve our global RANS solution correction process. Feature- and adjoint-based mesh adaptation strategies also relie on a hessian reconstruction to estimate the interpolation error and compute the optimal mesh to minimize the numerical error. This is why, the underestimation of the hessian near the boundary leads to an underestimation of the error in these elements. This in turn drives the minimization process to less refine the near-wall region as it is considered to produce fewer error than it actually does.

Hence, we propose to correct the boundary hessian before the global minimization in the adaptation process. This can be done both for the feature and adjoint based approaches. In Section 4.1.1, we deduced from the boundary gradient $\partial_{\mathbf{n}}\mathbf{u}$ and the size of the first element an equivalent hessian that gives the proper interpolation error in the normal direction. We can thus consider this corrected hessian as the actual hessian of the tangential velocity in the normal direction

$$\tilde{h}(\mathbf{u}\cdot\mathbf{n}) = \mathbf{n}\cdot\tilde{H}(\mathbf{u}\cdot\mathbf{n})\cdot\mathbf{n}\,.$$

In the case of feature-based adaptation on the Mach field, this corrected hessian is used to compute the normal hessian of the Mach field: $\tilde{h}(M) = \tilde{h}(\mathbf{u}\cdot\mathbf{n})/c$. In the case of adjoint-based adaptation, each variable must be corrected separately, with a proper decomposition in the tangential space.

The normal hessian is then expressed in the cartesian coordinates as

$$\tilde{H}(W) = \tilde{h}(\mathbf{u}\cdot\mathbf{n})[\mathbf{n}\otimes\mathbf{n}],$$

and added to the hessian computed by the standard process. Finally, the overall process states

Algorithm 8 RANS hessian boundary correction

1. Compute the primal solution W_h
2. Compute the approximate hessian $H(W_h)$
3. Compute $u_\tau = \sqrt{\mu(\partial_{\mathbf{n}}u)_{\text{wall}}}$
4. Compute $\tilde{h}(\mathbf{u}\cdot\mathbf{n})|_0 \approx \dfrac{u_\tau^3}{\nu^2}H_{eq}(u^+)\Big|_{y^+=\frac{u_\tau x_1}{\nu}}$
5. Add the corrected boundary values of the hessian $\tilde{H}(W_h) = H(W_h) + \tilde{h}(\mathbf{u}\cdot\mathbf{n})[\mathbf{n}\otimes\mathbf{n}]$ to the approximate values.

This analysis holds for flat boundary layers, but in the case of curved surfaces, additional terms have to be taken into account.

Application to High-Lift 2D configuration: This case is the classical three elements airfoil high-lift configuration considered here at Mach 0.175, with a Reynolds number of 15.1×10^6 and an angle of attack of 16.21°. This is a very interesting case for mesh adaptation as the shear layer that develops behind the first element, above the boundary layer of the second element is difficult to predict. It is thus difficult to generate *a-priori* a proper structured mesh. Figure 6.49 shows the influence of the discretization of the shear layer by hand-made meshes shown in Figure 6.50 on the overall solution. We can see in Figures 6.49-(top) that if the shear layer is not properly discretized as shown in Figure 6.50-(top), it does not appears in the solution. This difference is relatively small on the second foil element (Figure 6.49-(top left and bottom left)) but it induces a large difference on the last element as can be seen in Figure 6.49-(left). However, we can easily realize how difficult it is to prescribe *a-priori* the mesh shown in Figure 6.50-(bottom), with the only knowledge of the solution shown in Figure 6.49-(top).

Mesh adaptation strategy automatically refines this region as can be seen in Figure 6.51. However, this mesh has been obtained with an iterative feature-based mesh adaptation procedure that, as we saw, tends to refine slowly the boundary layers. This mesh has been generated following one of the initial approaches foreseen to deal with RANS mesh adaptation: A fixed structured boundary layer of 5 elements has been inserted in the mesh while the volume is adapted with metric aligned method [Marcum 2014] to generate pseudo structured meshes. This treatment is inconsistent with mesh adaptation strategy which balances the distribution of degrees of freedom in order to get an optimal mesh. Namely, there are to many elements in the boundary layer initially and to few in the last computations. It also makes it extremely difficult or impossible to adapt the boundary surface (especially in 3D) as the boundary layer mesh has either to be frozen or adapted as a whole.

This is also a very interesting case for the present study as the treatment of the boundary layer is crucial as it has a tremendous effect on the overall result. We can see in Figure 6.52 that if the boundary layer resolution is not sufficient, the resulting flow is detached, contrary to the expected flow. Feature-based adaptation detects slowly the boundary layers and inserting a fixed structured boundary layer mesh replaces the adjoint by identifying the near wall region as a region of interest.

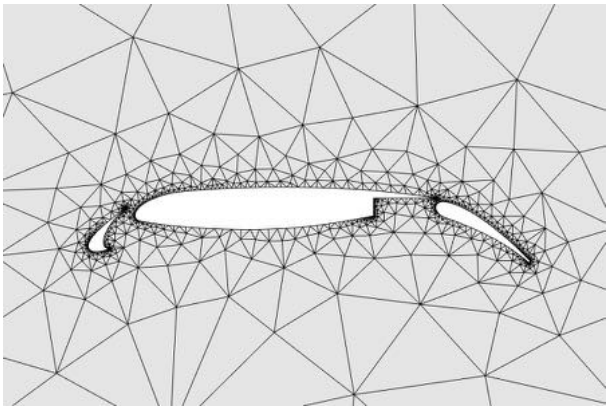


Figure 6.48: Three elements airfoil: Initial mesh used to start the mesh adaptation process.

As previously, we performed a mesh adaptation with standard wall functions at a fixed

distance, an adaptation with adaptive wall function, an adaptation with a wall integrated resolution and additionally an adaptation with wall integration using our boundary correction of the hessian in the error estimation. Meshes are iteratively generated following the global mesh adaptation loop described in Section 6.2 with a feature-based adaptation on the Mach number field. Ten meshes are generated for each of the following complexities: 5 000, 10 000, 20 000, 40 000, 80 000,160 000, 320 000, 640 000, 1 280 000 and 2 560 000 vertices. The initial mesh is shown in Figure 6.48

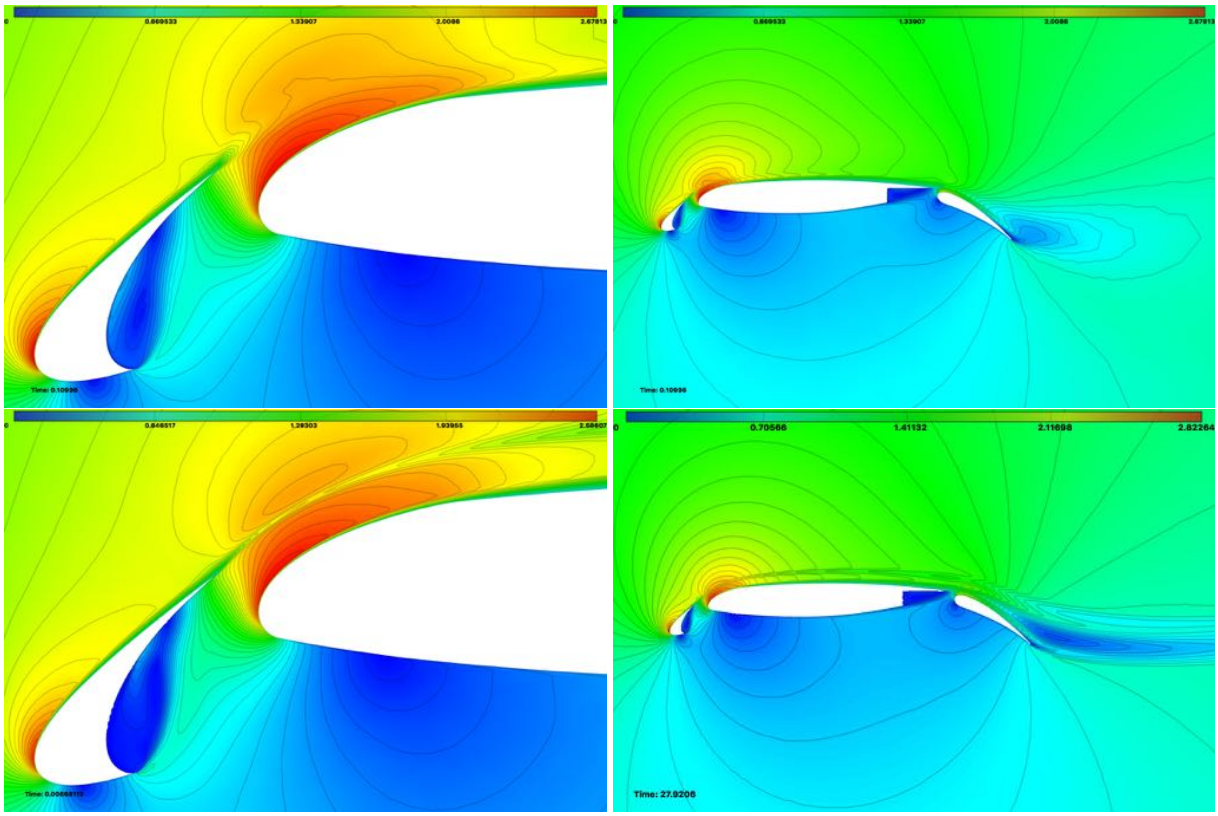


Figure 6.49: Three elements airfoil: Velocity contours around the three elements high-lift configuration computed on hand-made meshes without shear layer adaptation (top) and with shear layer adaptation (bottom). Close up view of the leading edge (left) and general view (right).

The resulting computed lift coefficient is shown in Figure 6.53. We can see that the detachment of the flow due to the under-resolution of the boundary layer leads to an under prediction of the lift when using wall integration. The boundary layer resolution becomes sufficient around 10^5 nodes to re-attach the flow and converge to the expected solution. The initial iterations are thus useless with regard to the final solution.

Meanwhile, we can see that the standard wall functions also lead to an initial detached flow but converge earlier. Contrary to the NACA0012 case, the adaptive wall laws converge later than universal wall functions with fixed distance but still earlier than wall integration. This highlights again the need for a more clever choice of the wall distance. However, adaptive wall functions still behave like wall integration asymptotically.

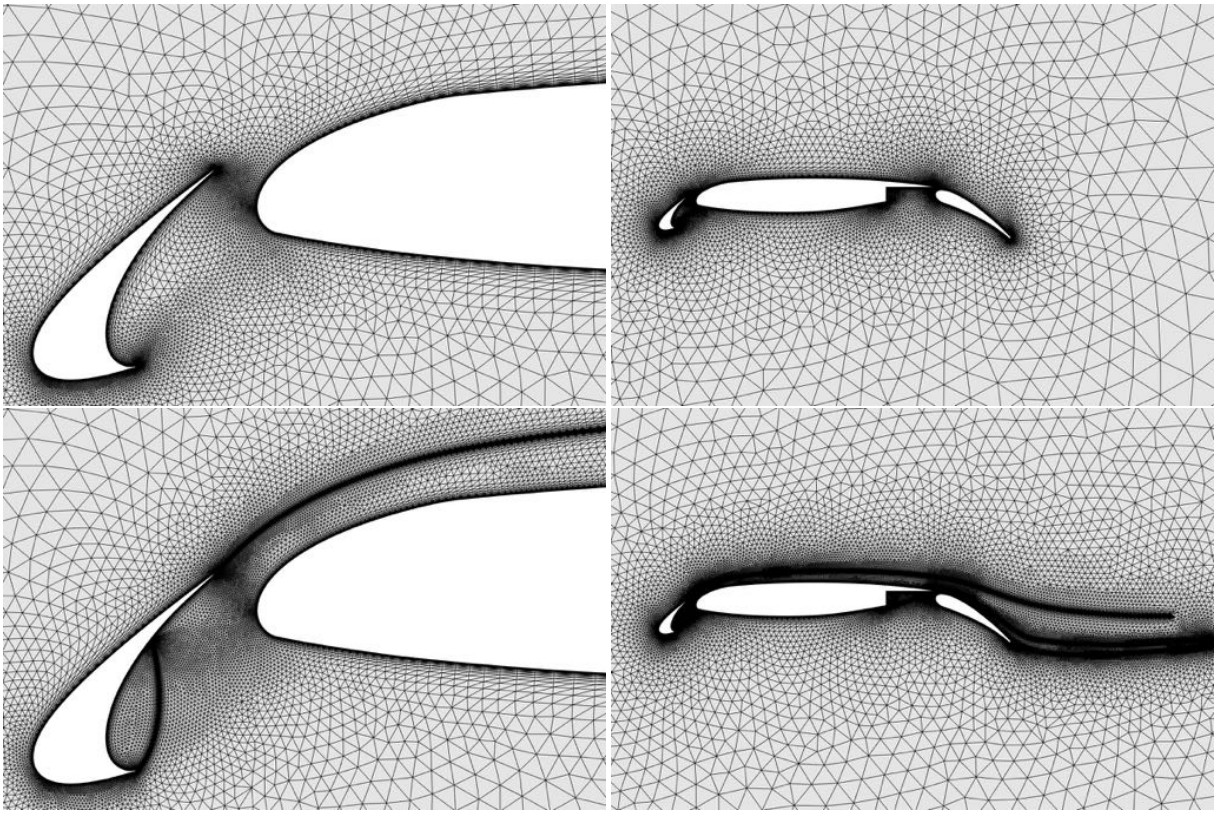


Figure 6.50: Three elements airfoil: Hand made meshes without shear layer adaptation (top) and with shear layer adaptation (bottom). Close up view of the leading edge (left) and general view (right).

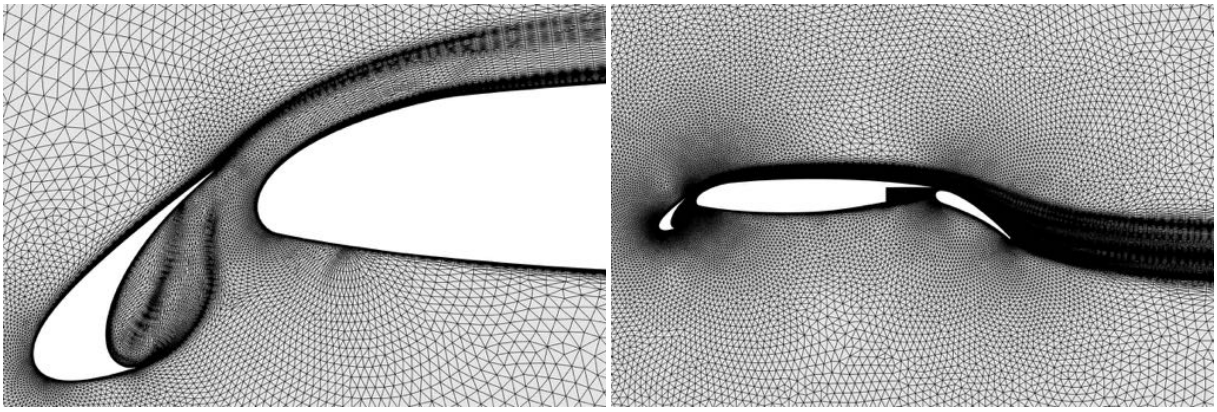


Figure 6.51: Three elements airfoil: Adapted mesh generated by feature-based adaptation using the Mach field as sensor.

Finally, the hessian boundary correction provides a much faster convergence and earlier reattachment. This is due to an early capture of the boundary layer depicted in Figure 6.54. We notice that the mesh prescribed by the corrected sensor is a lot more refined in the near wall

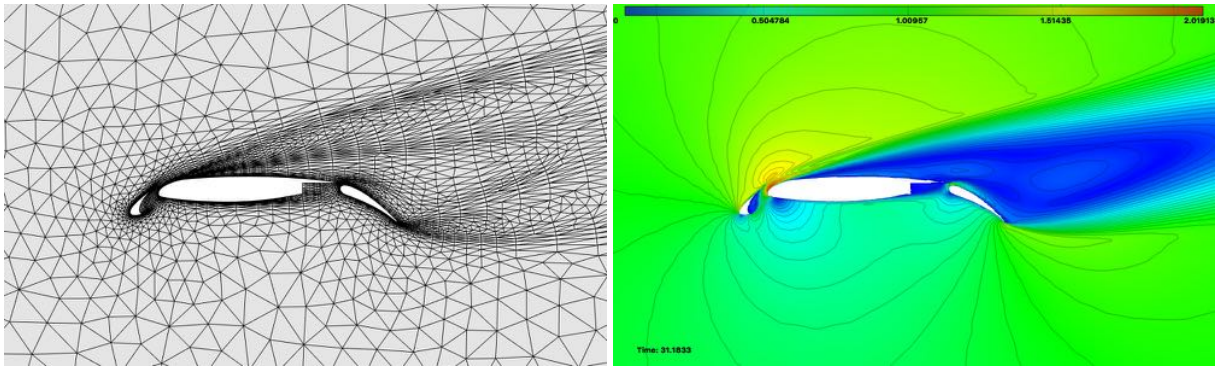


Figure 6.52: Three elements airfoil: Velocity field (right) and adapted mesh generated with a feature-based mesh adaptation based on the mach field without boundary layer insertion.

region as compared to the standard sensor. It is worth mentioning that contrary to a standard boundary layer insertion, this approach is naturally compatible with mesh adaptation. Indeed it does not affect the adaptation of the surfaces and the mesh complexity in the boundary layer is balanced with the main flow.

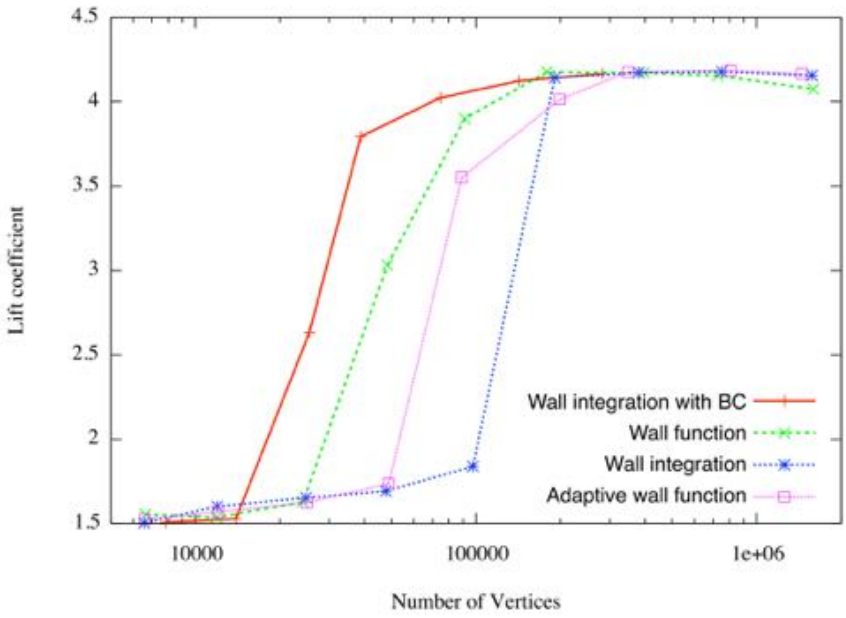


Figure 6.53: Lift convergence on the three element air foil geometry using wall functions and wall integration.

6.7.3 Conclusion

Although naive, this first implementation of adaptive wall functions already shows satisfying results for mesh adaptation strategies. The use of universal wall functions enables to prescribe any wall distance even below $y^+ = 30$ providing much flexibility. The use of an adaptive wall

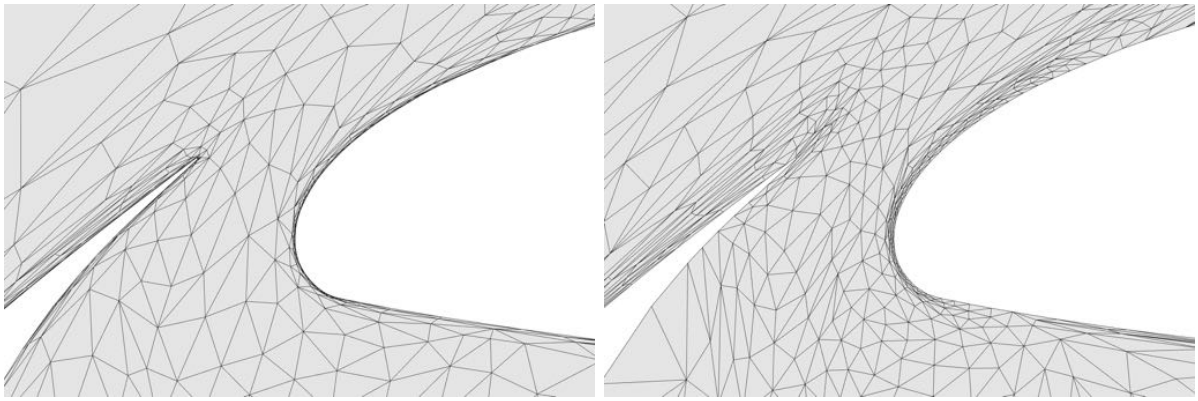


Figure 6.54: Three elements airfoil: Adapted meshes generated by feature-based mesh adaptation based on the Mach field with hessian boundary correction (left, 7 854 vertices) and without correction (right, 11 957 vertices).

distance ensures a proper description of the boundary layer depending on the current mesh size. Ultimately, it degenerates to a standard wall integration as the mesh size decreases.

However, the naive linear correlation between the wall distance and mesh size is insufficient as it does not takes fully account of the flow state. Moreover, we used here for simplicity a hessian based mesh adaptation which is highly inefficient as it adapts the whole wake, wasting ressources. For those reasons an adjoint based approach taking into account boundary contributions would be more appropriate to perform the mesh adaptation and prescribe the wall distance.

The hessian boundary correction of the adaptation sensor also showed very promising results. Similarly, it should be extended to adjoint based sensors and 3D cases.

6.8 Conclusion

In this chapter we have introduced and developed the underlying mathematical analysis involved in the adjoint-based goal-oriented mesh adaptation strategy. We presented different sensors and implementations to take into account the viscous and turbulent contributions in the mean flow and turbulent equation. These viscous sensors are then tested on different applied cases.

We first demonstrated the viability of unstructured RANS mesh adaptation on the NASA Rotor 37 case, using a simple feature-based adaptation based on the Mach field to minimize its interpolation error in L^2-norm. This highlights the capacity of our finite volume flow solver `Wolf` to perform RANS computations on fully unstructured meshes. This is then confirmed in our study of the Onera M6 wing, during which we validate the implementation of viscous goal-oriented sensors, demonstrating their superiority over standard feature-based mesh adaptation. In the same time, we compared our predictions with `Wolf` on adapted meshes to other research and commercial solvers on standard grids. By this mean we demonstrate the ability of modern numerical scheme to accurately and reliably compute turbulent flows with RANS models on fully unstructured meshes. This observation is then confirmed by computations on the trap wing and High-Lift Common Research Model cases on which `Wolf` exhibits on adapted meshes an early convergence of the lift.

These simulations highlight the automatism and robustness of the proposed mesh adaptation process. Still, we could not reach the desired mesh complexity on complex cases as we are currently limited by the memory consumption of the adjoint resolution in the first place and the mean flow resolution. This indicated the acute need for memory optimization of the process and the necessary implementation of a MPI version of `Wolf`. Despite these limitations, we shown that adapted meshes with an adjoint-based strategy can yield, for a given precision, a reduction of the number of vertices required by a factor of five to ten.

Finally, we proposed a strategy based on turbulent boundary layer theory to further improve RANS mesh adaptation. This approach shown encouraging results on a simple 2D case and the analysis should be continued in 3D and improved for curved boundaries.

Chapter 7

Periodic mesh adaptation

7.1 Periodic mesh adaptation

As we choose to use a strong geometric periodicity in the solver, it has to be enforced in the mesh at each adaptation. Theoretically it simply requires to copy the periodic vicinity of any element near the periodic boundary to perform any operation. However, it would require several structural modifications in the mesh adaptation tool which is not desired.

In a first attempt, we developed an approach inspired by [Dobrzynski 2011]. The idea is that the periodic frontier has been chosen arbitrarily and is not different from any other separation in the domain and could be adapted in the volume. To do so, we only need to immerse the frontier in the periodic domain.

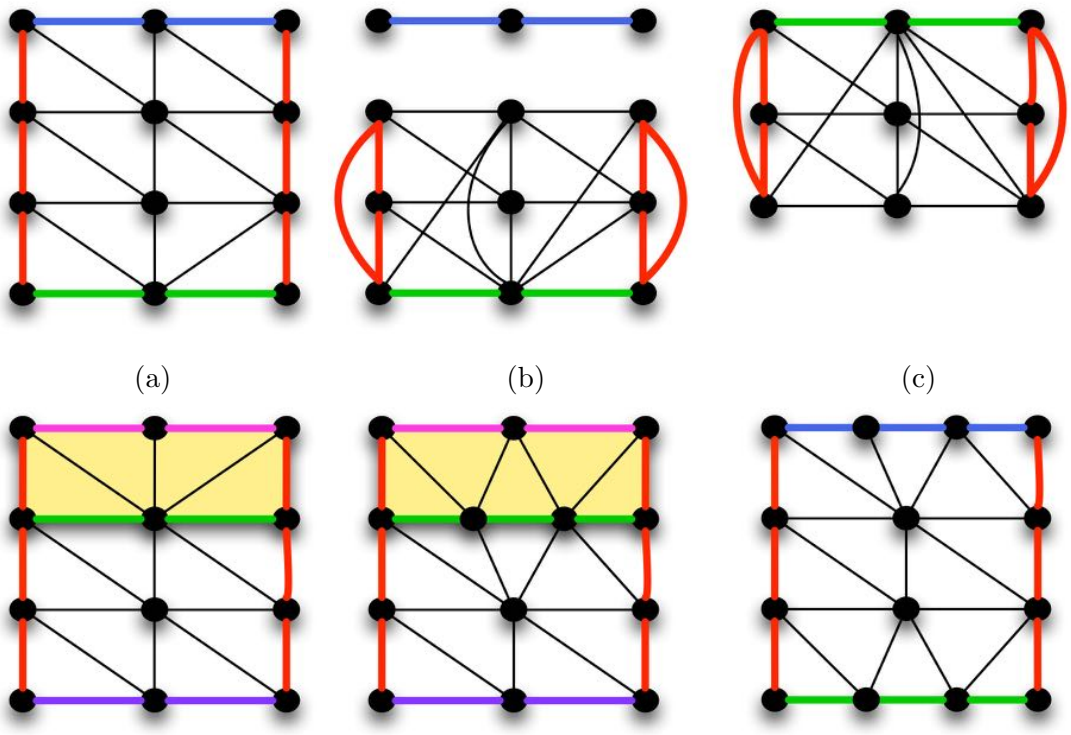


Figure 7.1: Periodic mesh adaptation process.

We provide in Figure 7.1 the schematic process of periodic mesh adaptation. We start from a periodic mesh in the vertical direction, lines in blue and green representing both sides of the

periodic domain. Lines in red represent any other frontier. Starting from the initial mesh, Figure 7.1 (a), we go through the elements and wrap the mesh on itself, replacing the vertices of one side of the domain by their periodic counterpart. This step affects all volume elements (thin black lines) and the non-periodic frontiers (red lines). This leads to a topologically periodic mesh with elements of negative volume, shown in Figure 7.1 (b). The vertices and periodic edges that have been disconnected from the rest of the domain, shown in blue in Figure 7.1 (b), are removed.

Vertices can be moved independently, keeping the topology of the mesh. We then simply translate a few layers (any number) of vertices to their periodic location without changing the connectivity. In Figure 7.1 we move only one layer of vertices, those on the other side of the periodic domain, in green and obtain Figure 7.1 (c). The elements that have been displaced in this process are flagged in order to be identified later, in yellow in Figure 7.1 (d).

The vertices on the edges of the periodic domain are identified and duplicated in order to recreate their periodic counterpart. Elements are then reconnected to the proper vertices in order to recover a mesh with elements of positive volume, as shown in Figure 7.1 (d). The new periodic frontiers are identified and added to the final mesh, in pink and purple in Figure 7.1 (d).

The mesh can then be adapted, the old periodic line, in green, is treated as non-manifold and adapted while keeping the geometry of the two domains. The new periodic lines are left unchanged, guaranteeing the periodicity of the mesh. We thus obtain Figure 7.1 (e), where the periodic line in green has been adapted.

The new mesh could be used for the next computation step, however, as the domain has been changed (translated), it would make it difficult to interpolate solutions between to successives meshes. To circumvent this, we apply the same periodic transformation in a backward direction in order to recover the initial domain. We start by removing one periodic frontier and wrap the mesh on itself, translate all vertices in a flagged element to their periodic position and reconstruct the topology (boundary edges, duplicate vertices, ...). We finally obtain mesh (f) in Figure 7.1, where the periodic frontiers have been successfully adapted.

Practical cases are shown in Figure 7.2. Several layers of elements have been displaced in both cases, flagged in green. Displacing more than one layer of elements gives more space to the mesh adaptation tool `feflo.a` to perform mesh operations. This is helpful in presence of strong anisotropy such as in shocks or in boundary layers regions, as shown in Figure 7.3. We can see that in this case, the mesh adaptation tool is left with very little margin. Fortunately, these regions have been adapted in the volume previously, and the periodic frontier that we are interested in is immersed in the new domain.

We can see the adaptation of this frontier illustrated in Figure 7.4. The complexity being raised at this step, the size of the elements decreases globally. But as the periodic boundary is left unadapted during the initial step, we can see that we are left with too coarse elements on it in Figure 7.4-(left). Though, as the frontier is immersed in the new domain, it can be adapted as shown in Figure 7.4-(right).

7.1.1 Handling the metric

In order to adapt the mesh in the transformed configuration, we need the corresponding metric. Hence, it has to be transformed consistently with the mesh, in particular, a unit edge in the

initial configuration must remain unit in the transformed configuration. Obviously, the metric is the same in the part of the domain that is not displaced. For translation periodicity, the metric space is unchanged, so that copying the metric from the initial node to the translated node is sufficient. This is not the case for rotation periodicity. A rotation changes the directions of the metric, hence the metric has to be copied and rotated from the initial node to the translated node. Let us denote by e a unit vector in the metric $\mathcal{M}$, representing an edge:

$$e\mathcal{M}e = 1.$$

The edge will be moved across the domain by the rotation and its orientation will change. We thus assimilate here the edge to a vector determining its orientation. This vector becomes in the rotation process

$$\bar{e} = Re,$$

where R is the matrix of the rotation. As this new edge $\bar{e}$ has to remain unit in the new metric $\bar{\mathcal{M}}$, we deduce

$$1 = \bar{e}\bar{\mathcal{M}}\bar{e} = e(R^T\bar{\mathcal{M}}R)e.$$

Thus, we deduce

$$\bar{\mathcal{M}} = R\mathcal{M}R^T, \tag{7.1}$$

where we use that for a rotation $RR^T = Id$.

7.1.2 Periodicity recovery

It is easier to have the list of pairs of periodic vertices given in entry, however, it may be necessary to reconstruct this data. The idea is to find the list of pairs of vertices $(P_i, P_j = T(P_i))$ in a given set of vertices $\{P_k\}_{k=1..N}$. The naive approach described in Algorithm 9 consists in cycling on the list of vertices and compare each vertex to the transformation of the others in order to find its periodic counterpart.

Algorithm 9 Quadratic algorithm for periodicity recovery

```
L = {P_k}_{k=1..N}
while L ≠ ∅ do
    remove P_i the first element of list L
    for P_j ∈ L do
        if ( P_j = T(P_i) ) then
            remove P_j from list L
            pair (P_i, P_j)
        end if
    end for
end while
```

However, this approach is quadratic and is way too expensive in 3D even on relatively small meshes. This is why we developed the following linear frontal algorithm using the topology of the mesh for a faster periodic recovery. We start by extracting the meshes of the periodic surfaces/lines and compute the elements neighbours. We then choose an element of the periodic

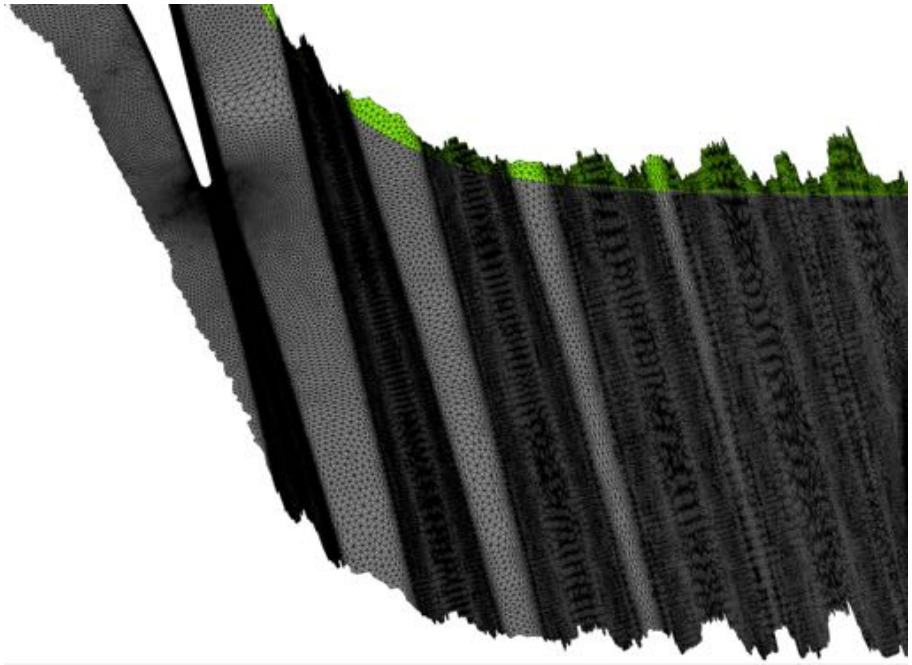
frontier (the first) and go through the list of elements in order to find its periodic counterpart (*i.e* the element which vertices are the periodic transformation of the first element).

Figure 7.5 show these two initial elements where the vertices are linked by red lines. Once we have paired two elements, we know that the element seen by one of its vertex through the opposite edge is the periodic element seen by the associated periodic vertex. These two elements are represented in green in Figure 7.5 (left). If the element has not yet been visited, it is added to the list of elements to deal with and the seeing vertex is paired (bold red line in Figure 7.5 (right)) if it has not been yet. The algorithm then pursue with all non visited neighbouring elements as shown in Figure 7.5 (right) and as long as the front is non-empty. Once there is no more elements in the front, the whole connex component of the boundary has been paired and visited, ending the algorithm.

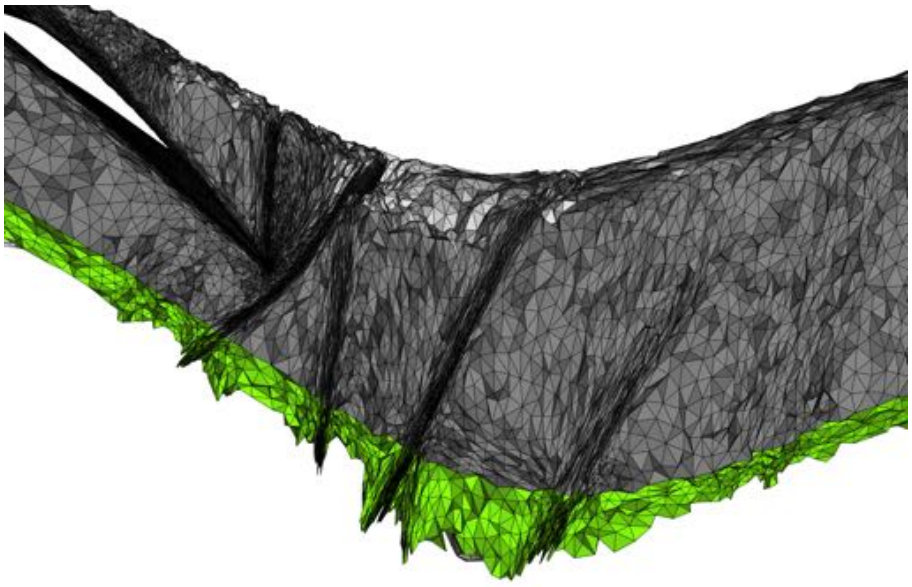
Algorithm 10 Linear algorithm for periodicity recovery

```
1: List(P_i): list of vertices
2: List(T_i): list of elements
3: for T_j ∈ List(T_k) do
4:     if ( T_j = T(T_1)) then
5:         break
6:     end if
7: end for
8: Pair vertices of T_1 and T_j
9: Flag T_1 and T_j
10: L = [(T_1, T_j)]
11: while L ≠ ∅ do
12:     remove (T_i, T_j) the first couple of list L
13:     for P_k ∈ T_i do
14:         let T_l be the element seen by P_k
15:         if T_l is not flaged then
16:             Let T_m be the element seen by Per(P_k) in T_j
17:             Flag T_m and T_l
18:             Add (T_l, T_m) to L
19:             Pair the remaining vertices of T_l and T_m
20:         end if
21:     end for
22: end while
```

This algorithm works well as long as the periodic frontier is manifold and connex. If the frontier has several connex components, the process has to be done for on each of them. If the frontier is non-manifold, the same procedure can be done using nodal connectivity. For each vertex, we compare its neighbours to the neighbours of its periodic counterpart. This approach is locally quadratic on the ball of each vertex (size of approximatively 6 on a surface) but still linear on the overall list of vertices. Note that this process enables to pair periodic vertices but it also makes possible to check the validity of the connectivity of the periodic frontiers and the correctness of the periodic transformation of each vertex.



LS89-2D



RO37-3D

Figure 7.2: Periodic mesh adaptation: domain translation and rotation on a LS89 and RO37 case. The grey elements are in the initial domain, the green elements have been displaced to their periodic location by the transformation.

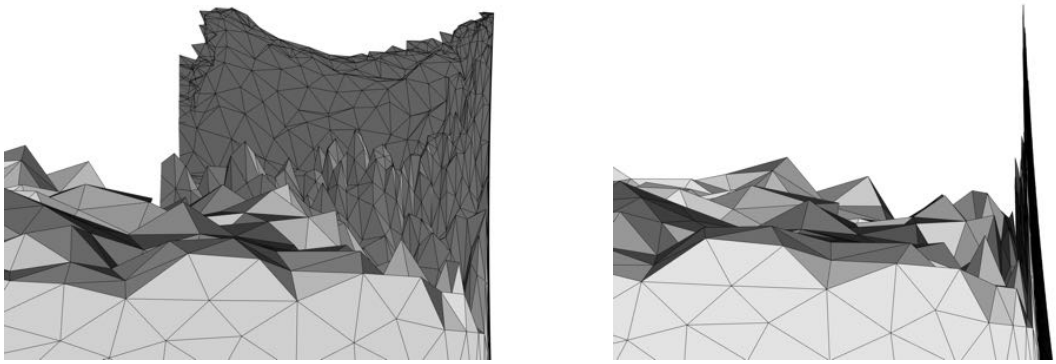


Figure 7.3: Elements displaced in presence of a boundary layer. Lateral view (left) and front view (right).

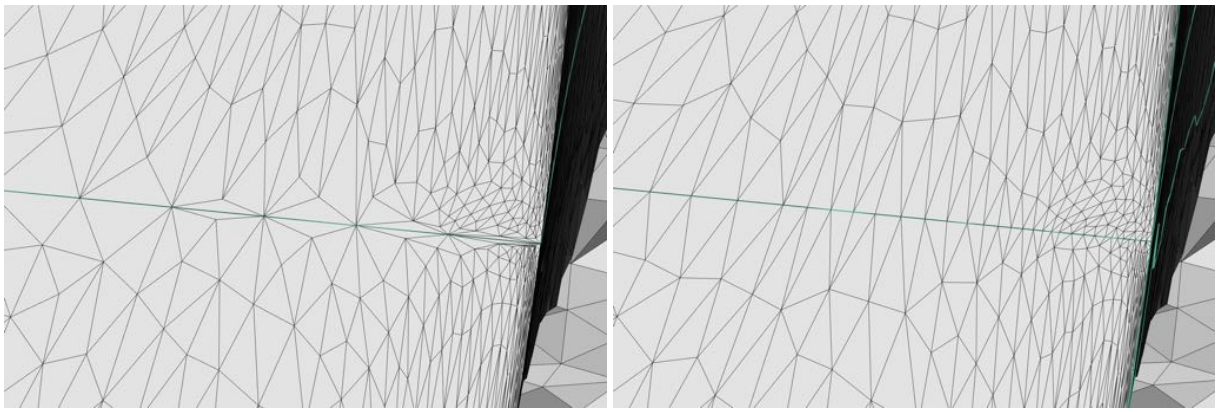


Figure 7.4: Adaptation of the immersed periodic frontier. Mesh before (left) and after (right) adaptation.

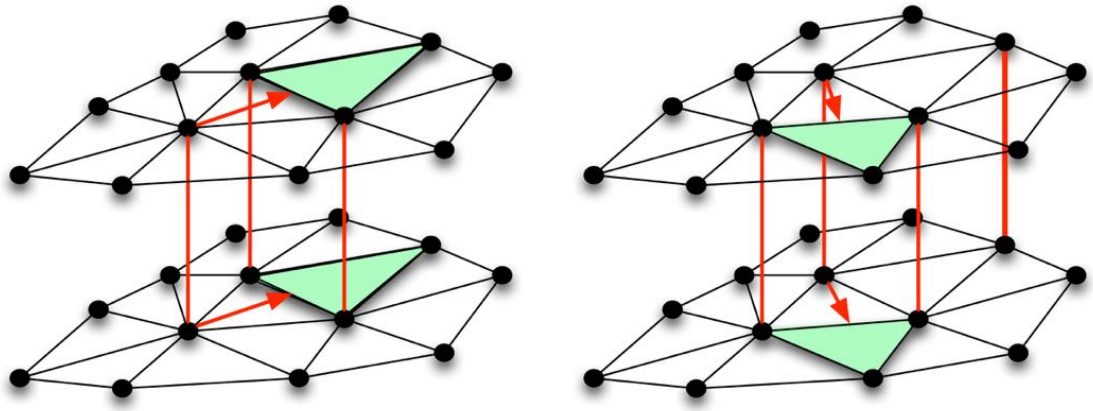


Figure 7.5: Topological periodicity recovery.

7.2 Application to turbomachinery

7.2.1 LS89 case

Description

We consider again the LS89 case, but this time a periodic adaptation is performed. In order to illustrate the versatility of mesh adaptation we choose different outflow pressures (summed up here after) so that a shock appears on the suction side in different places impacting the boundary layer and the wake (see Figure 7.6). As the periodic boundaries are now adapted, we do not need anymore the domain to be aligned with the wake of the blade. We thus use a more traditional "S" shaped domain (see Figure 7.6).

- Total inflow pressure: $1.828 \times 10^5 Pa$,
- Total inflow temperature: $413.3K$,
- Static outflow pressure: $1.08 \times 10^5 Pa$, $1.04 \times 10^5 Pa$, $1.00 \times 10^5 Pa$ and $0.90 \times 10^5 Pa$.

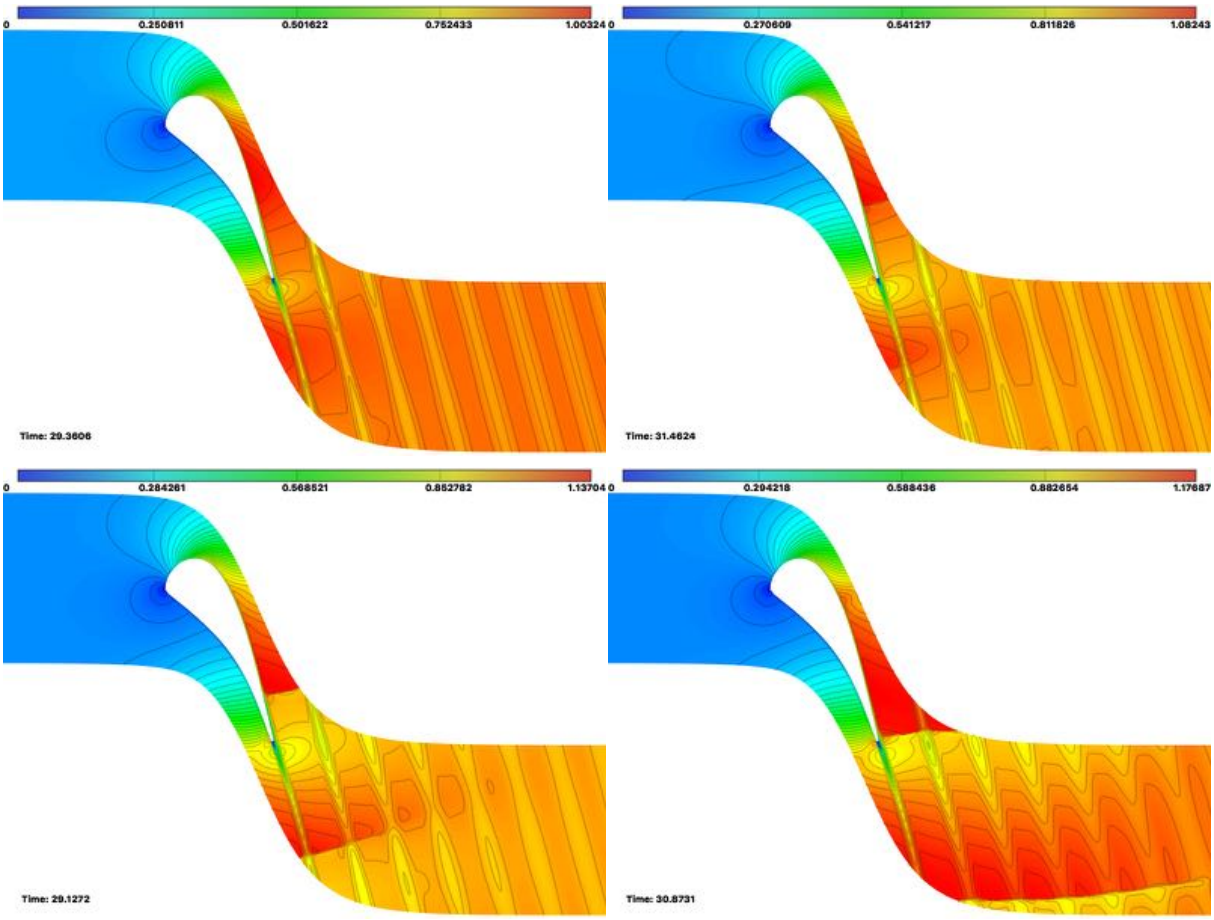


Figure 7.6: LS89 case: Mach isocontours for different pressure differences: $P_{out} = \alpha P_{in}^{tot}$. $\alpha =$ 0.5908 (top left), 0.5689 (top right), 0.5470 (bottom left) and 0.4923 (bottom right).

Reference density	1.275349 $kg.m^{-3}$
Reference velocity	(400, 0, 0) $m.s^{-1}$
Reference pressure	10^5 Pa
Dynamic viscosity	1.716×10^{-5} $Pa.s^{-1}$
Total inflow pressure	$1.828 \times 10^5 Pa$
Total inflow temperature	$413.3K$
Static outflow pressure	$1.08 \times 10^5 Pa$
	$1.04 \times 10^5 Pa$
	$1.00 \times 10^5 Pa$
	$0.90 \times 10^5 Pa$

Table 7.1: LS89 case: Flow description.

Turbulence model	SA Negative	Riemann Solver	HLLC
MUSCL extrapolation	Order 2 - V4	Limiter	Piperno
Cells	Median	Gradients reconstruction	L^2-projection
Renumbering	None	Initial CFL	50
CFL progression	Linear: 10	CFL_{max}	100
Freeze limiter iteration	1500	Max Solver iterations	2000
Targeted physical residual	10^{-12}	Targeted residual reduction	10^{-4}
Linear solver	SGS	Max linear iterations	20
Targeted linear residual	0.1		
Aero reference surface	1.	Aero reference length	1.
Center of pressure	(0,0,0)		

Table 7.2: LS89 case: Solver parameters.

The solvers parameters used to solve this case are listed in Table 7.2. The computation starts from a non-adapted mesh composed of 27503 vertices, refined around the blade, with no structured boundary layer (see Figure 7.9-(bottom left)) . In order to have a quasi-structured mesh, the boundary layer is automatically adapted with the metric aligned method [Marcum 2014]. For now, the periodic domain is adapted using the idea introduced by [Dobrzynski 2011] described in Section 7.1, five layers of elements close to one side of the periodic domain are moved to the other side of the domain. The domain is then adapted, keeping the geometric periodic line so that the physical domain is conserved ensuring valid operation between meshes. The adapted layers are then sent back. However, contrary to [Dobrzynski 2011], the geometry of the periodic line is kept so that the computational domain does not change. This prevents its degradations and guarantees the validity of the interpolation between meshes.

Physical conservative fields are used as sensor to adapt, providing an optimal metric minimizing the interpolation error of each field, the final metric being the intersection of all. Note that no particular criterion (such as shock or vortex detection) has to be specified, the resulting mesh is the mesh minimizing the interpolation error in L^p-norm of each field. This guaranties the convergence of the solution and the proper representation of any physical phenomenon of interest. Meanwhile, the use of the L^2 norm contrary to L^∞ norm ensures a balance between strong and weak phoenomenon. The wake of the blade is thus refined alongside shocks and boundary layers. This case already shows the versatility and efficiency of mesh adaptation for turbomachinery applications and in this process the importance of periodic adaptation. As the pressure difference increases, the initial subsonic flow becomes transonic and a shock appears on the succion side of the blade. This shock is initially weak and requires an appropriate fine discretization in the proper position to be computed. Then, this shock moves toward the trailing edge and becomes stronger. Similarly, the wake is subject to a strong diffusion if not discretized properly, although it is relatively easier to predict.

Results

We intentionally started with an inappropriate mesh shown in Figure 7.9, simply refined near the blade with no adaptation of the boundary layer, shock or wake. Figure 7.7 shows intermediate adapted meshes generated for each case while the corresponding solution is shown in Figure 7.6. The mesh is automatically adapted to the wake and shock depending of their position with no *a priori* knowledge of the solution. Meanwhile, the boundary layer is progressively discretized, the height of the first cell decreasing from $y^+ = 20$ in the first meshes to $y^+ = 1$ in the final mesh. Details of the adapted mesh for case with $\alpha = 0.5689$ are shown in Figure 7.8. We can clearly see how the unstructured mesh becomes quasi structured in the boundary layer, shock and wake thanks to the metric-aligned method.

In Figure 7.9, we can see the influence of mesh adaptation on the quality of the solution of the second case ($\alpha = 0.5689$)with a moderate shock. The shock appears here only if it is appropriately discretized, while the initial boundary layer is too thick. Moreover the shock interacts with the boundary layer, thickening it, which does not appear on the initial mesh, the boundary layer is evenly thicker. These features influence both heat transfer at blade's skin and turbine performances, and thus have to be appropriately computed.

Figure 7.10 demonstrates the importance of periodic mesh adaptation. When the periodic frontiers are appropriately adapted, we can see the shock crossing the frontier without being affected. This is what we expect from a numerical periodic frontier, to behave as an other domain. When the periodic frontiers are left unadapted, as the inner mesh is continuously refined, as fine as the initial mesh can be on the boundary, it inevitably ends to be too coarse compared to the required mesh size at a point of the adaptation. This leads to an inversion of the local anisotropy which deteriorates the quality of the solution and finally generates instabilities leading to unphysical results and finally the break down of the solver. Figure 7.11 shows a zoom on the density iso-contour on the other side of the periodic domain. It is clear that the shock has been diffused by the periodic boundary condition in absence of proper adaptation.

A similar constatation is done for the wake of the blade. Figure 7.12 shows coarse adapted meshes and a zoom on finer adapted meshes generated without (left) and with (right) periodic adaptation. We can see that periodic mesh adaptation yield very few differences on coarse meshes as the discretization of the periodic boundary condition is sufficient. But it is clear on fine meshes that its absence induces strong noise in the field and in turn in the process as it is captured by mesh adaptation.

The influence of these meshes on the solution is depicted in Figure 7.13 where we can see the turbulence variable and the velocity being abruptly diffused while crossing the frontier. Figure 7.14 gives a clearer view of the negative impact of non-adapting the periodic frontiers on the turbulence variable.

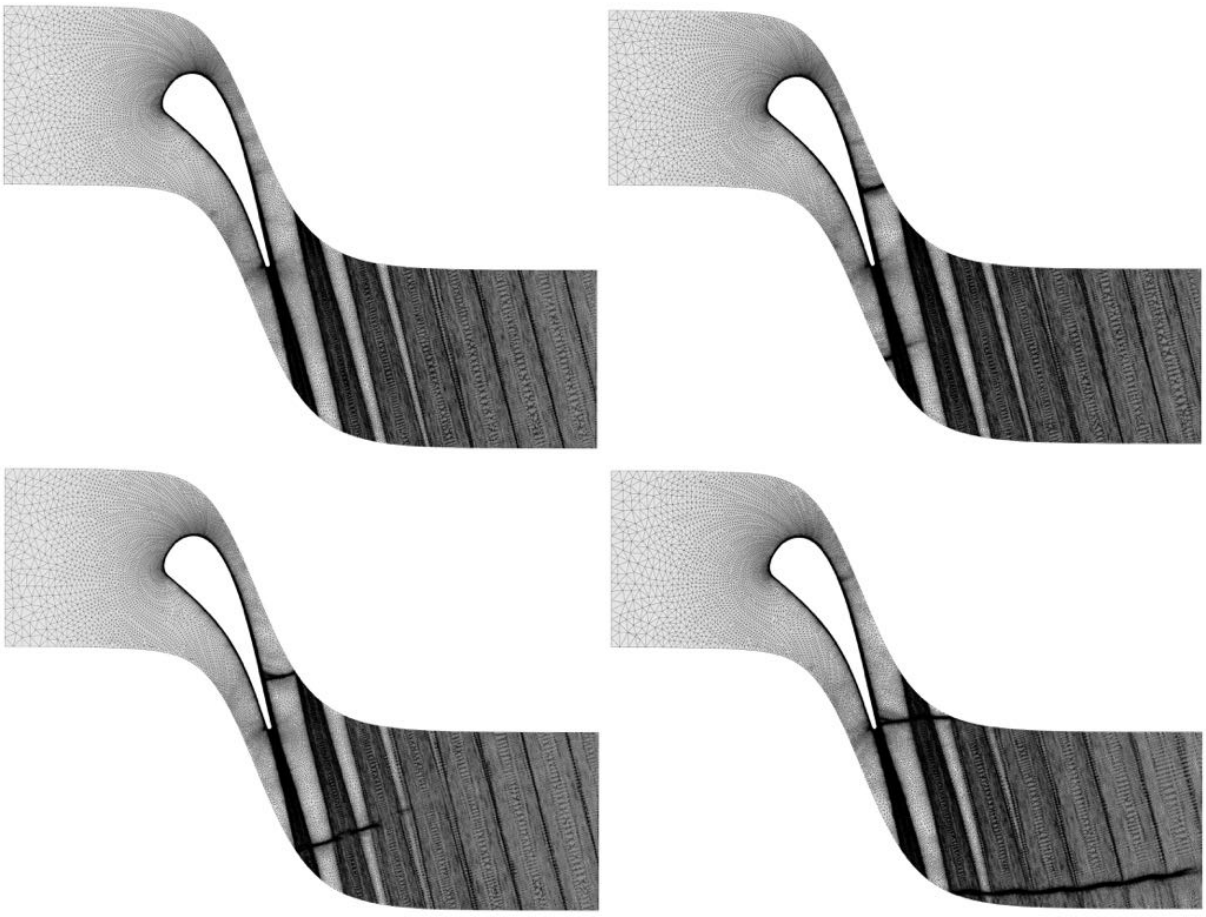


Figure 7.7: LS89 case: intermediate adapted mesh generated for different pressure differences $P_{out} = \alpha P_{in}^{tot}$. $\alpha = 0.5908$ (top left, 124610 vertices), 0.5689 (top right, 126366 vertices), 0.5470 (bottom left, 125965 vertices) and 0.4923 (bottom right, 127149 vertices).

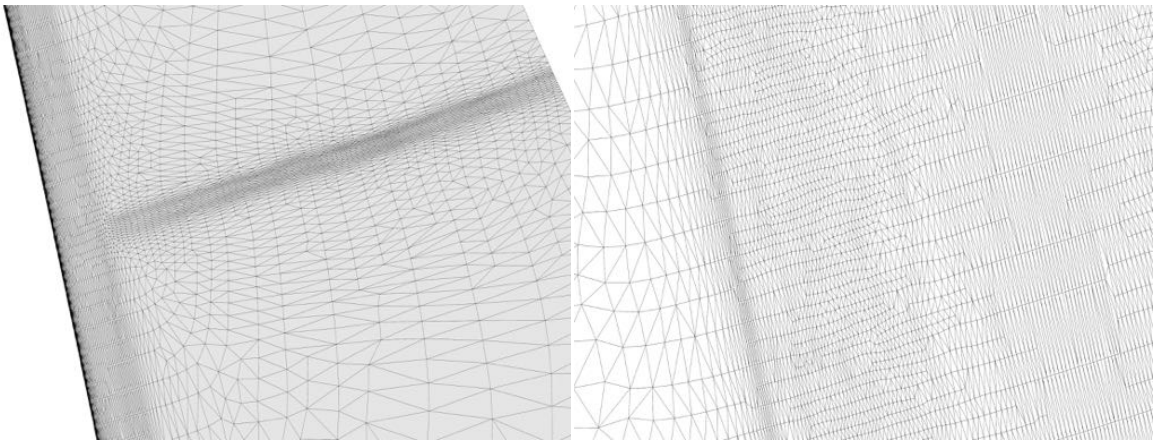


Figure 7.8: LS89 case: Close up view on the adapted mesh in the shock and the boundary layer (left) and in the wake (right).

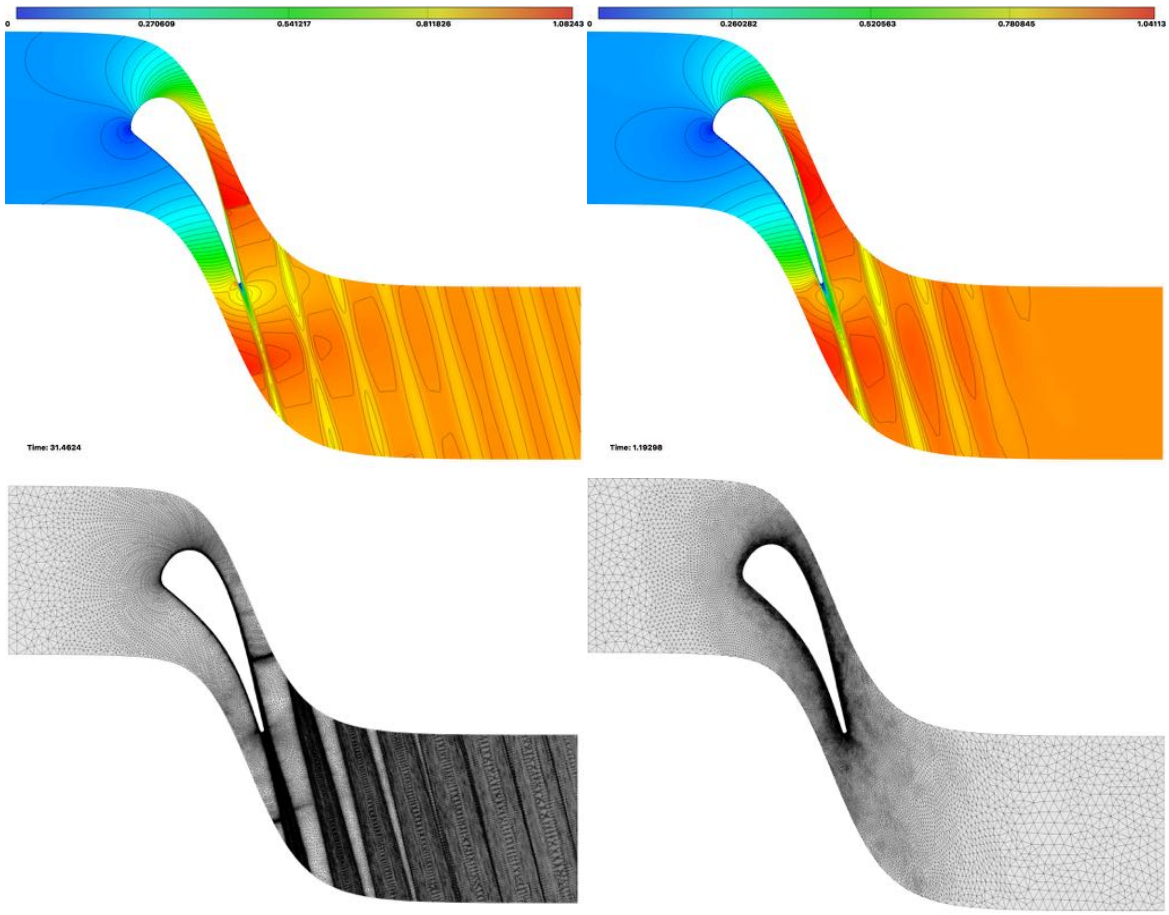


Figure 7.9: LS89 case: Influence of adaptation on the second case. Mach number isocontours (top) on the adapted adapted mesh (bottom left, 125965 vertices) and unadapted initial mesh (right, 31790 vertices).

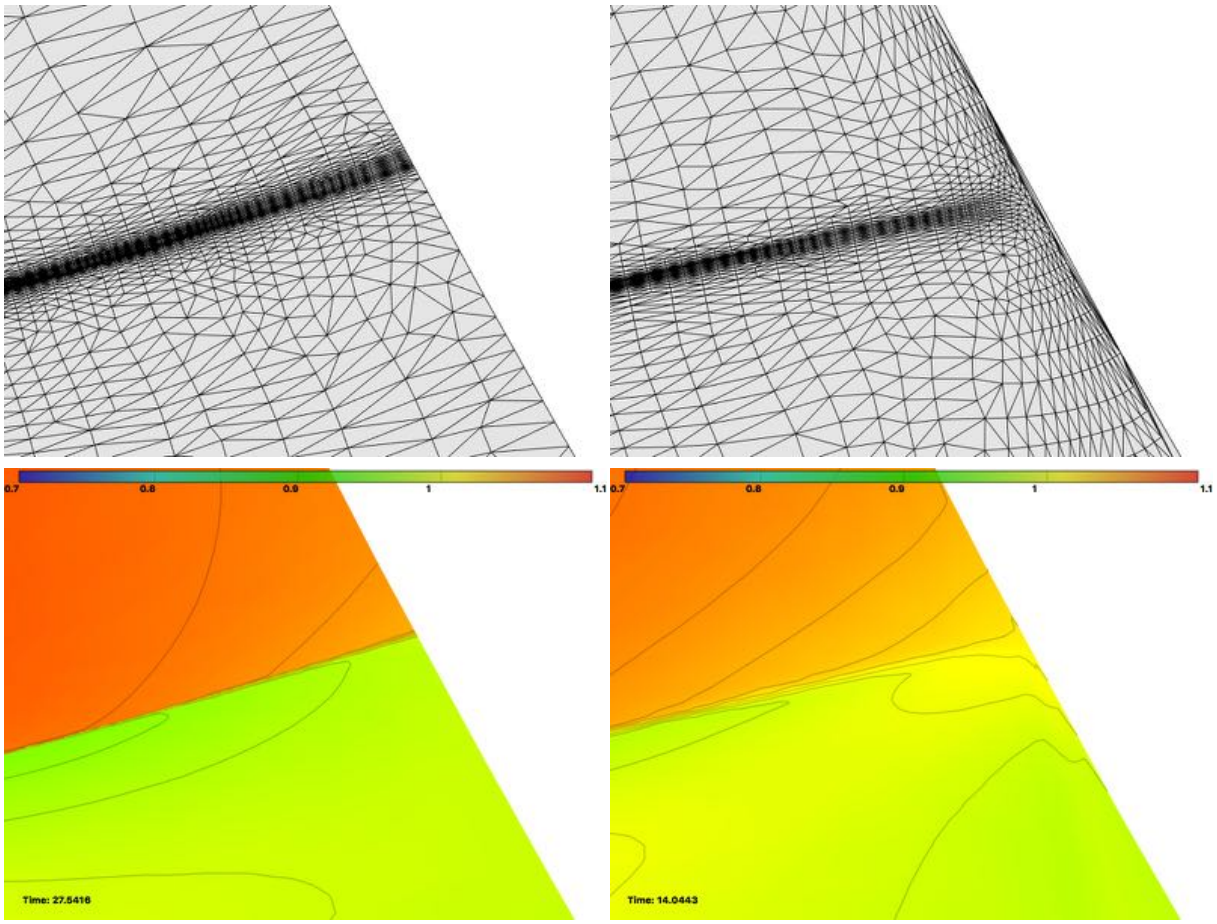


Figure 7.10: LS89 case: Influence of periodic adaptation on shock: adapted mesh (top left) and Mach number isocountours (bottom left) with periodic adaptation and adapted mesh (top right) and Mach number isocountours (bottom right) without periodic adaptation.

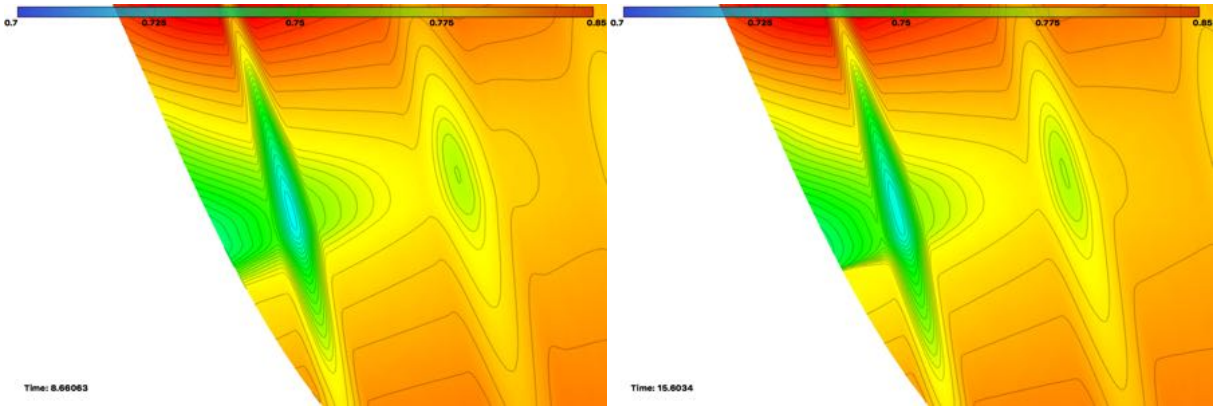


Figure 7.11: LS89 case: Influence of periodic adaptation on shock, zoom on the density isocontour on the periodic frontier: without periodic adaptation (left) and with periodic adaptation (right) .

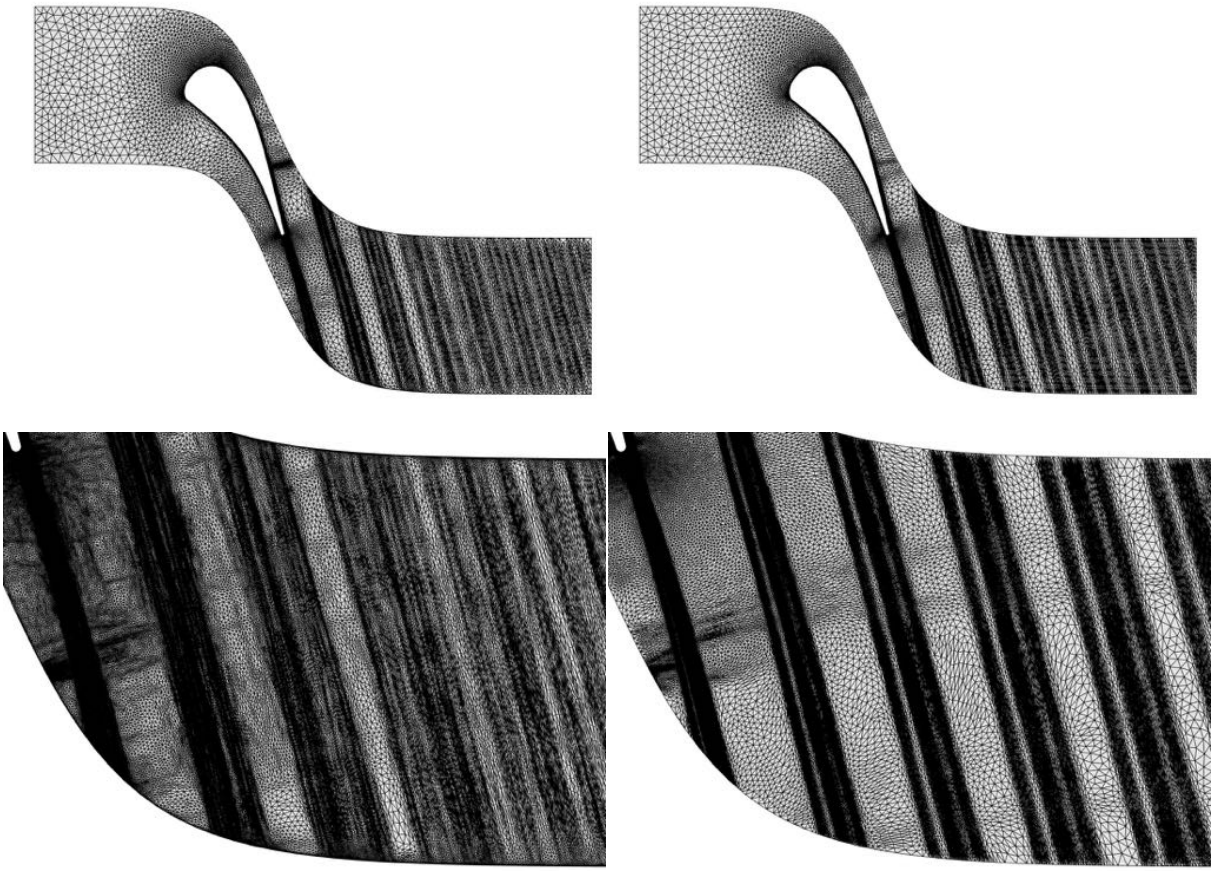


Figure 7.12: LS89 case: Coarse (top) and fine (bottom) adapted meshes generated with feature based adaptation based on the Mach field without (left) and with (right) periodic frontier adaptation.

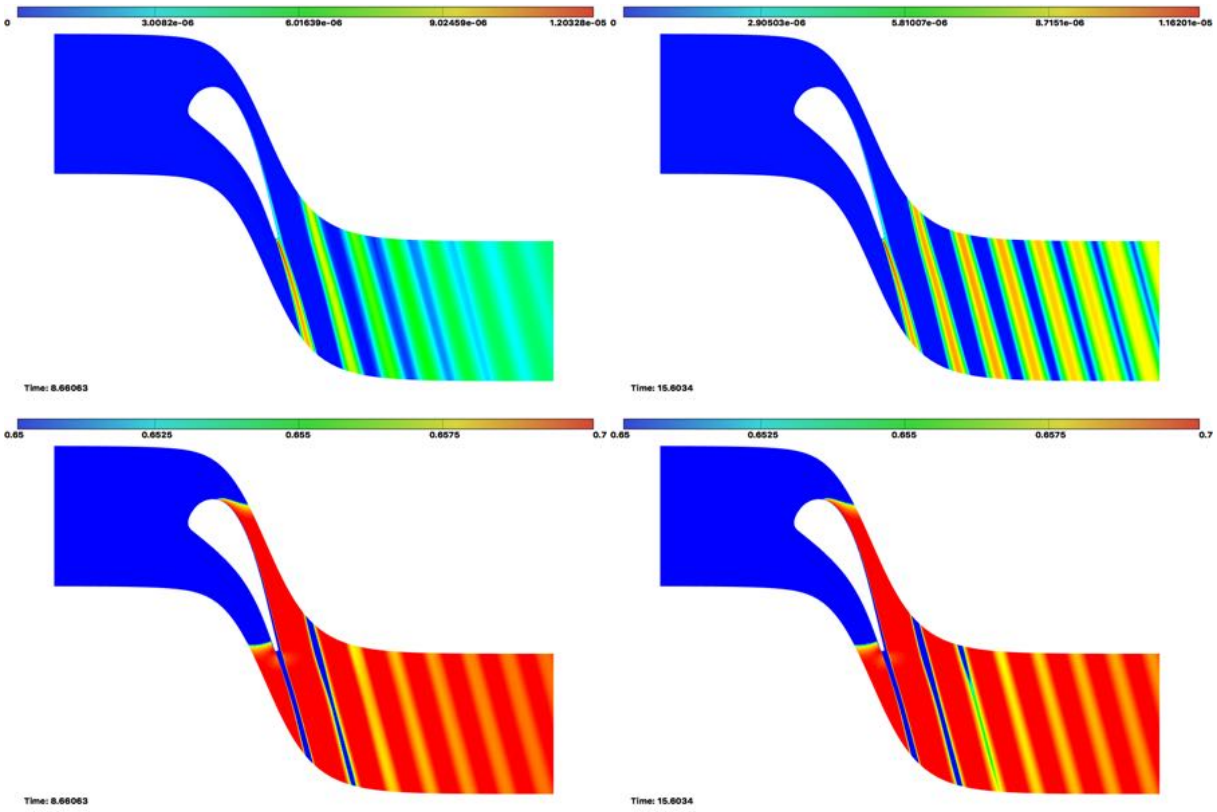


Figure 7.13: LS89 case: Influence of periodic adaptation on the turbulence variable (top) and velocity norm (bottom), without (left) and with (right) periodic mesh adaptation.

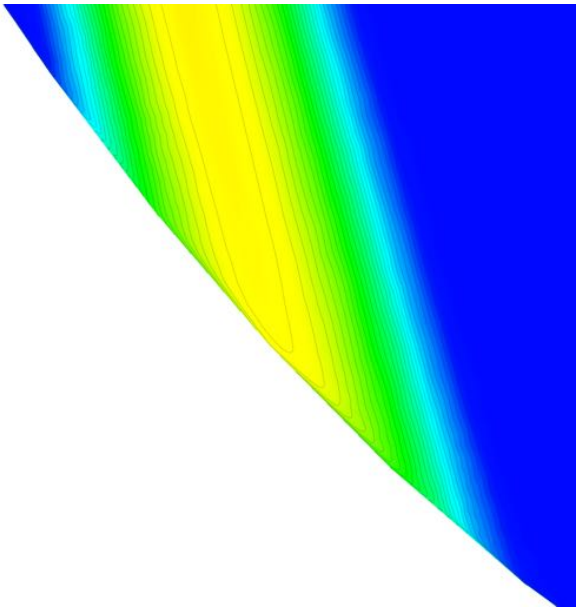


Figure 7.14: LS89 case: Close up view on the artificial diffusion induced in the turbulence variable crossing the non-adapted periodic frontier.

7.2.2 RO37 case

We now perform the same analysis on the 3D, RO37 case.

Case description

The NASA Rotor 37 geometry is considered here, its regime and the solver parameters used to solve this case have been described in Tables 6.1 and 6.2. The gaz is described with the perfect gaz law with $r = 287.04$, $\gamma = 1.4$, and the viscosity follows the Sutherland law with $\mu_0 = 1.716 \times 10^{-5}$, $T_0 = 273.15$ K and $S = 110.4$. The simulation is restrained to a single sector of 10 degres, in the rotating frame, the geometry is thus fixed. The computation is directly run with second order scheme, fourth-order dissipation with the Piperno limiter.

Mesh adaptation

We use a feature-based mesh adaptation, minimizing the interpolation error of the Mach field in L^2-norm. Hessians are reconstructed using L^2 projection. Five iterations are computed at each of the following complexities: 200 000, 400 000, 800 000, 1 600 000 and 3 200 000 vertices. The whole domain is adapted, including periodic boundaries and boundary layers with full unstructured mesh. The periodic frontiers are adapted using the method developed in Section 7.1. The geometry of the domain is prescribed with P_3 reconstruction from the initial mesh, for surface adaptation.

In order to assess the effect of periodic mesh adaptation, we compare a mesh containing 3.516.488 vertices and 20.294.221 tetrahedra to an adapted mesh generated in the previous study presented in Section 3.3.2, containing 6.607.814 vertices and 38.835.389 tetrahedra. Figure 7.15 shows a global view of the periodic frontier in both cases. In the non-adapted case, it is left unchanged and is thus the same as the initial mesh. Meanwhile, in the adapted case we observe that all structures of the flow have been successfully adapted, in particular the shocks upstream and the wake downstream. This discretization has in turn a very strong impact on the solution. In particular, we can see in Figure 7.16 a close up view on the periodic frontier, where a shock crosses it. It is clear that in the non-adapted case the initial discretization is way too coarse to deal with such structures. Moreover, we can tell the discreptancy between the mesh size required in the volume by the adaptation and on the surface. This apparent discontinuity creates ill-conditioned elements with a strong anisotropy in the opposite direction and an abrupt transition. This will in turn lead to the breakdown of the solver.

We now investigate the impact of this improper discretization on the mesh. Although both computations have been done in a single 10° sector, we duplicated the meshes for visualisation. Figure 7.17 shows a cut in the mesh perpendicular to the axis of the blade. We can see the shocks produced by the leading edge of the blade propagating upstream. These shocks are clearly diffused by the periodic frontier when it is not adapted, they barely cross it four times, while in the adapted case, shocks are unaffected by the boundary and propagate up to the inflow. Moreover, while in the adapted case the periodic junction is seamless and barely visible, the non-adaptation of the periodic frontier yields a strong constraint in the mesh which makes it frankly visible. This produces a lot of noise in the mesh and the solution.

A closer look at the shocks near the leading edge reveals in Figure 7.18 how the two leading edge shocks are actually suddenly artificially diffused as they cross the periodic frontier. We can tell how this affects the solution as the second one seems to even disappear while the first one is actually reflected and hits the blade back. After, in front of the lambda shock we see in the adapted case a secondary shock which is absent in the non-adapted case. Here again, the junction is seamless in the adapted case, while the non-adapted case produces concentration of ill-conditioned elements next to the periodic boundary condition. An even closer look at the shock boundary layer interaction where the lambda shock hits the blade shows how the shock is diffused by the non-adapted domain. We can tell the size of the elements on the frontier by the spread of the shock that is created by this interaction. This interaction directly modifies the shape of the shock but also generated a wake in the middle of the vane which modifies the debit and velocity of the flow.

Finally, Figure 7.19 shows a global view of another cut perpendicular to the blade. This summarizes the aforementioned constatations, we can see the on the Mach field how shocks are diffused in the non-adapted case and does not propagate upwind as far as with periodic adaptation. We also observe in the Mach field the reflection of the shocks on the periodic frontier and the generation of a huge artificial wake. Finally, we distinguish a similar impact of the discretization on the wake. Overall, the solution obtained with periodic mesh adaptation is definitely cleaner and the use of mesh adaptation without periodicity definitely pollutes the solution.

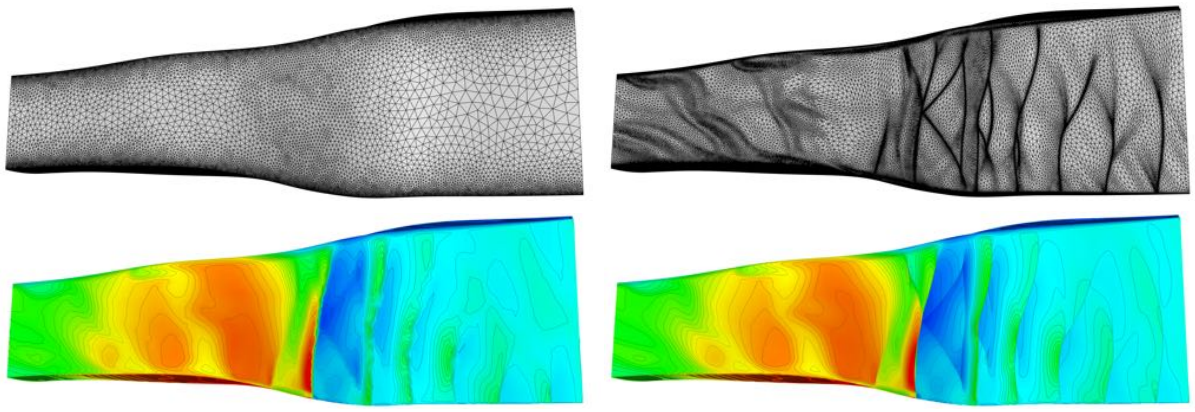


Figure 7.15: NASA RO37: Global view of the mesh (top) on the periodic frontier in the non-adapted (left) and adapted case (right), and the corresponding mach field (bottom).

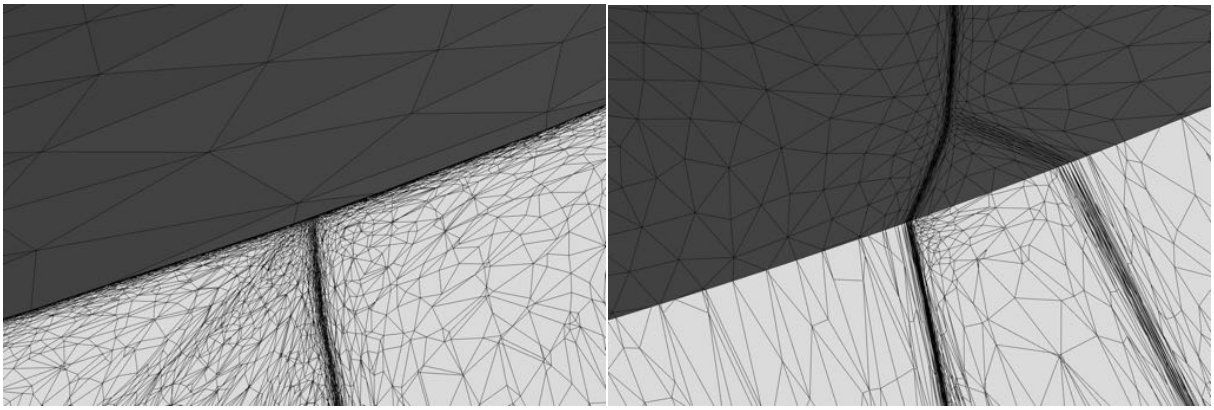


Figure 7.16: NASA RO37: Impact of the discretization of the periodic frontier on a shock.

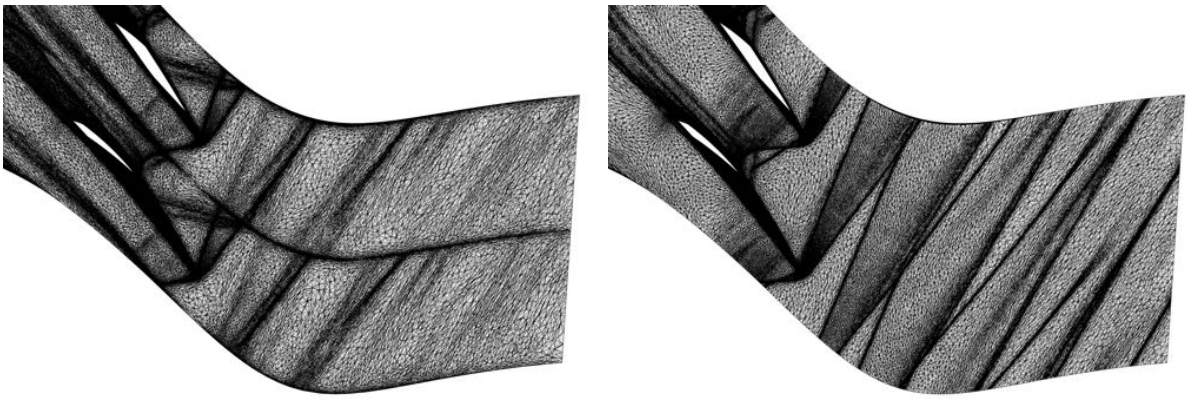


Figure 7.17: NASA RO37: Propagation of shocks upstream of the RO37 blades without (left) and with (right) periodic mesh adaptation.

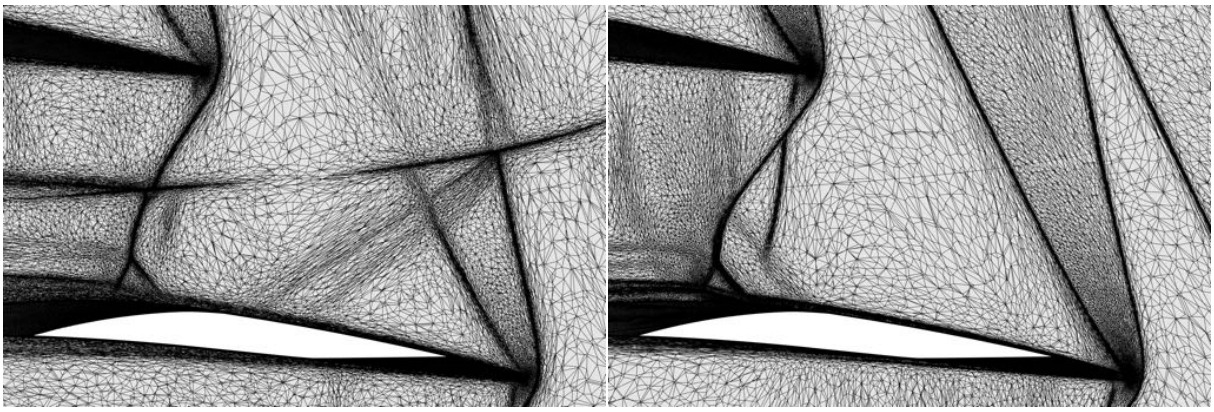


Figure 7.18: NASA RO37: Zoom on the leading edge shocks of the RO37 blades without (left) and with (right) periodic mesh adaptation.

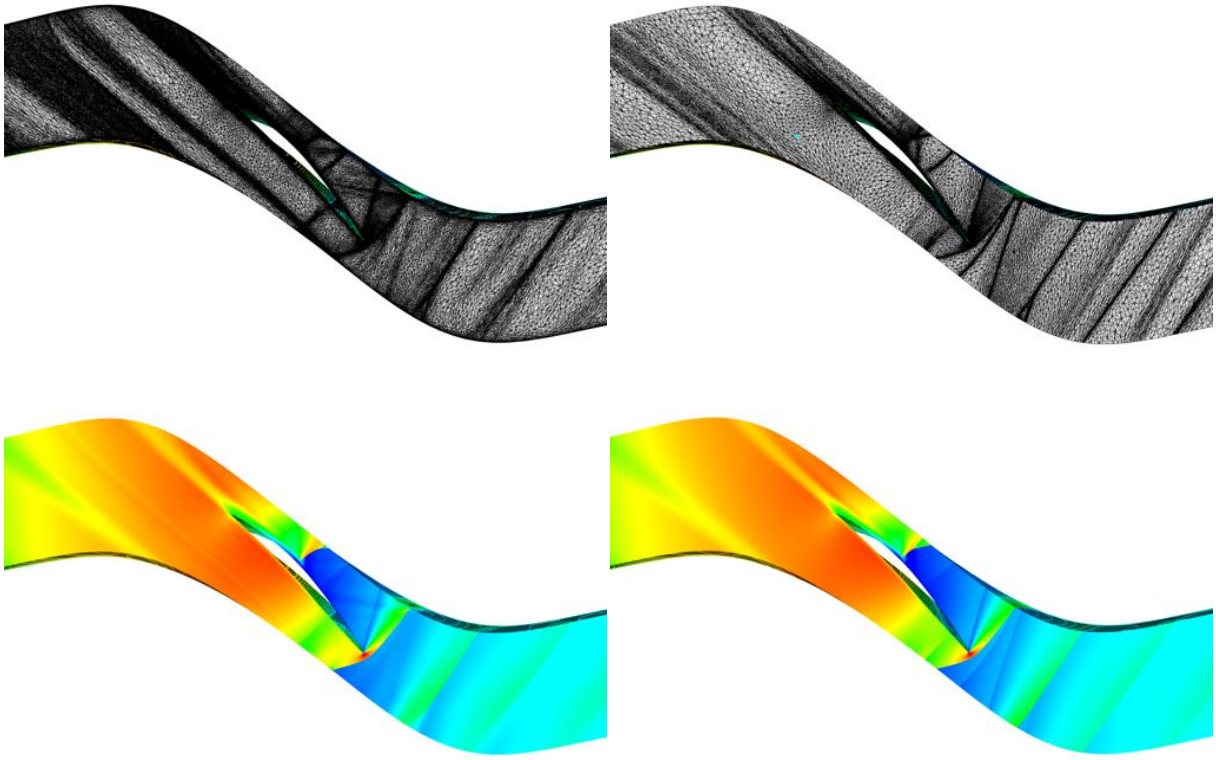


Figure 7.19: NASA RO37: Global view of the mesh (top) and the solution (bottom) in a cut in the domain perpendicular to the blade without (left) and with (right) periodic mesh adaptation.

Conclusion

In this part, we introduced the mathematical context and analysis for output based RANS mesh adaptation. We first developed the analytical analysis of the discretization errors involved in RANS simulations and the appropriate treatment to deal with their non-linearity. We then detailed the implementation of different adjoint-based error sensors derived from this analysis. The estimation of interpolation errors on second order derivatives is addressed by different means.

These sensors are then used to perform several mesh adaptations on different applied cases with increasing complexity. By this mean we simultaneously demonstrate the achievability of computations on fully unstructured grids with finite volume flow solvers such as `Wolf` and the accuracy of the error sensors developed previously. These computations clearly establish the superiority of adjoint-based mesh adaptation strategies over feature-based mesh adaptation. They also show the capacity of our adaptation process to automatically deal with complex geometries without any human intervention, saving a large amount of user involved time. In order to fasten mesh convergence and computation we introduced two approaches based on turbulent boundary layer theory. These implementations showed promising results in 2D and should be further extended to 3D. Additional developments should also be conducted to extend their use to goal-oriented mesh adaptation considering the improvement that can be expected.

Then, we introduced a specific treatment for periodic mesh adaptation. We validated this approach on 2D and 3D turbomachinery cases showing the benefits that a proper mesh adaptation can bring to these applications. Still, we only performed mesh adaptation studies with feature-based mesh adaptation strategy and a preliminary adjoint-based strategy in 2D. Considering the results provided by goal-oriented mesh adaptation on aerodynamics cases, we will validate the development proposed in the first part of this thesis to implement a periodic adjoint.

All the mesh adaptation studies performed in this part used viscous goal-oriented approach, but ignored the contribution of the turbulent equation to the numerical error. In particular, the turbulent viscosity was accounted for, as a constant viscosity in the error analysis. We proposed in this part an approach to deal with these contributions, that should be tested and validated. Moreover, we proposed different formulations of the error sensors, but we did not provided an extensive comparison of their behaviour yet. Additionally, a careful study of the effects of boundary curvature on the surface mesh should be performed. Finally, the computations in this thesis were performed using thread parallelization, which limits the meshes we can use, to about ten million vertices. A MPI implementation of `Wolf` would enable a more extensive validation of the goal-oriented mesh adaptation on the cases presented in this thesis.

Conclusion and perspectives

In numerical simulations and in particular with mesh adaptation strategy, the couple mesh-solution cannot be separated from each other. Hence, there is a global tradeoff between the minimal workload required for the development and use of both the solver and mesh generation. Some appropriate choices and a carful implementation of the flow solver can significantly improve the robustness of the flow solver and in turn lower the quality requirements on the mesh. In particular, solvers are historically built on an evolution of finite difference analysis an many still relie on a block structured discretization. Anyway, most solvers require (or are at least are believed so) a regular quasi structured discretization of the boundary layer.

In this thesis we shown with our vertex centred mixed finite-elements - finite-volume flow solver `Wolf`, that numerical simulations of turbulent flows with RANS models were possible with a fully unstructured anisotropic discretization of the boundary layer. This was made possible by a minimization of the approximations both in the residual computation, and the implicit linear solver. Modern schemes based on mixed finite-elements - finite-volume discretization and MUSCL extrapolations lead to a stable and accurate resolution of the flow on anisotropic unstructured meshes. Meanwhile, a full differentiation of the equations enables a fast and robust convergence with an implicit solver.

Finally, as part of research contracts with Boeing Commercial Aircrafts, we implemented and validated different additional versions of the Spalart-Allmaras turbulence model. Similarly, as part of research contracts with Safran Tech, we made the necessary developments in the flow solver and mesh adaptation process to deal with periodic flows in a rotating frame in order to perform computations and mesh adaptations on turbomachinery applications.

In the global numerical simulation workflow for engineering, an accurate estimation of the numerical errors is mandatory. To this end, we developed a linear and non-linear analysis of the error in RANS computations for finite volume and finite element discretization. We proposed a non-linear correction of the solutions, extending the traditional linear approach. This correction proved to provide an accurate estimation of the numerical error on theoretical and applied cases such as for the Sonic Boom Prediction Workshop aircraft configuration.

From this analysis, we developed a general adjoint-based error estimation and mesh adaptation strategy for non-linear equations. This paradigm encompasses and extends previous inviscid and laminar goal-oriented analysis to RANS equations, to prescribe optimal anisotropic meshes for the computation of a specific functional output. Additionally, we proposed a coupling with wall functions to further improve theses error sensors in the boundary layers.

In the second part of this thesis, we tested and validated these error sensors, performing different mesh adaptations on several complex geometries and flows. Taking advantage of public experimental and numerical results published in the NASA validation database and AIAA workshops, we compared our predictions with `Wolf` on adapted meshes to experimental data and other predictions obtained with various standard commercial and research solvers. We shown that adjoint-based mesh adaptation strategy successfully provides optimal meshes for high-lift and drag cases. The results obtained with `Wolf` on meshes generated with goal-oriented mesh

adaptations nonetheless fitted with experimental data but also exhibited early convergence, reducing the mesh complexity required by a factor of 5 to 10. This robust and fully automatic strategy is particularly efficient when it comes to generate rapidly optimal meshes for complex geometries and flows, where traditional approaches usually require several month. Finally, we demonstrated the viability and benefits of standard multi-scale mesh adaptation strategy for turbomachinery applications.

Perspectives

The main achievement of this PhD was to show the viability and efficiency of automatic mesh adaptation for RANS equations. Still, this approach requires a strong solver, *i.e.* capable of leading computations on fully unstructured and anisotropic meshes. Many commercial solvers do not meet this requirement and the user may not be able to circumvent this issue for various reasons. The generation of quasi-structured boundary layer meshes, guided by the RANS error sensors developed in this thesis is thus a valuable challenge. Different approaches are foreseen, such as metric-align and metric-orthogonal strategies, but it sill remains an open question.

Similarly, the accurate re-projection of vertices on the CAD during mesh adaptation is an essential element for reliable industrial applications. The computations performed in this thesis relied on an automatic cubic reconstruction of the CAD based on an initial mesh. This approach proved to be surprisingly robust and accurate and should be perfected, in particular with respect to the control of the geometric approximation error. Different approaches can be considered in order to directly work with the given CAD or with a finely tuned cubic reconstruction fitting the surface up to a required precision.

As regard to the solver, despite some major improvements, the computation of turbulent flows with RANS equations in `Wolf` are still too expensive when compared to what we consider they ought to be. First of all, the use of limiters that are mandatory for robustness is today the major impediment to the convergence. It has been noticed that the apparition of limite cycles due to limiters depends on the mesh considered. The development of limiters with smoother behaviour and the understanding of their interaction with to the mesh is thus a promising way.

Similarly, the resolution of the adjoint system is known to be stiff. This has been overcome in `Wolf` by the use of a Krylov linear solver, but at the cost of CPU time and huge memory requirements. The development of a memory efficient approach is thus a high priority, alongside of a MPI implementation of `Wolf` to achieve computations on larger meshes. Despite the efficiency of adjoint-based mesh adaptation, large challenging cases as high-lift prediction on a full body aircraft still require large computational power.

This thesis focused on mesh adaptation for steady RANS computations on fixed geometries, but the general methodology that we proposed can be extended to other applications. Indeed, most realistic applications are unsteady and involve moving geometries. For example, in the NASA RO37 case, we simplified the problem assuming a periodic stationary flow. But, in practice, this compressor blade should be considered as a stage of a whole compressor, with rotating parts. The number of blades are generally chosen to be prime between each rotors and stators in order to prevent resonances, so that the flow is actually never periodic. Moreover, simulations of critical interest, such as the propagation of an asymmetric stall of the compressor

are, by nature, unstationary and not periodic. This require to model the whole circular domain and additional stages.

These requirements have been difficult to meet so far as the simulation of moving geometries with traditional mesh generation is difficult. Beside the large amount of computational ressources needed, it demands specific complex treatments to appropriately match the different moving parts of the meshes. The ALE formulation and unsteady mesh adaptation approach proposed in [Barral 2015] used in conjunction with the unsteady adjoint proposed in [Gauci 2018] would provide an interesting solution to automatically and consistently deal with unsteady applications. To this end, it will necessitate to extend the sensors presented in this thesis to unsteady flows. This will also require a careful management of the boundary layer around moving parts.

The current trend is also to develop multi-physic simulations. Keeping the NASA RO37 case as example, the aerodynamic load deforms the blade and should be taken into account in a design process. Here again, traditional processes lack of automatism and robustness. Coupled mesh adaptation can provide an efficient approach for both the structure and fluid computations. In particular, the balance between the computational resources to be spend in the structure and the fluid simulation is unknown and would be prescribed automatically with a coupled adjoint.

Additionally, RANS and U-RANS models are known to poorly predict detached flows that are generally crucial for applications such as stall prediction. As we mentioned, a significant effort is dedicated today to the development of LES and hybrid high order methods to deal with such problems. But, these approaches suffer from the same limitations as traditional RANS computations: the mesh is generated *a-priori* while many complex features of the flow are difficult to predict. Mesh adaptation strategies could drastically reduce the mesh size required for LES computations without deteriorating the numerical precision. To this end, the general error analysis developed in this thesis can be extended to interpolation errors of higher orders in conjunction with the aforementioned unsteady mesh adaptation strategies. Though, the major limitation here is related to the chaotic behaviour of the Navier-Stokes equations. Indeed, close initial solution diverge from each-other exponentially, so that standard adjoint computations diverge in these cases and progressive refinement do not lead to mesh convergence. Instead, new structures that were filtered by the mesh appear in the flow as the mesh size decreases. A specific filtering of the flow, should thus be accounted for in the error analysis in order to make these simulations reliable.

Acknowledgments

Bibliography

[A. Loseille 2017] F. Alauzet A. Loseille and V. Menier. *Unique cavity-based operator and hierarchical domain partitioning for fast parallel generation of anisotropic meshes.* Computer-Aided Design, vol. 85, pages 53–67, 2017.

[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).

[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).

[Alauzet 2009a] F. Alauzet and A. Loseille. *High Order Sonic Boom Modeling by Adaptive Methods.* RR-6845, INRIA, February 2009.

[Alauzet 2009b] F. Alauzet and A. Loseille. *On the use of space filling curves for parallel anisotropic mesh adaptation.* In Proceedings of the 18th International Meshing Roundtable, pages 337–357. Springer, 2009.

[Alauzet 2010a] F. Alauzet. *Size gradation control of anisotropic meshes.* Finite Elem. Anal. Des., vol. 46, pages 181–202, 2010.

[Alauzet 2010b] F. Alauzet and A. Loseille. *High Order Sonic Boom Modeling by Adaptive Methods.* J. Comp. Phys., vol. 229, pages 561–593, 2010.

[Alauzet 2010c] F. Alauzet and M. Mehrenberger. *P1-conservative solution interpolation on unstructured triangular meshes.* International Journal for Numerical Methods in Engineering, vol. 84, no. 13, pages 1552–1588, 2010.

[Alauzet 2015] F. Alauzet and D. Marcum. *A closed advancing-layer method with connectivity optimization based mesh movement for viscous mesh generation.* Eng. w. Comp., vol. 31, pages 545–560, 2015.

[Alauzet 2017] F. Alauzet, A. Loseille and D. Marcum. *On a robust boundary layer mesh generation process.* In 55th AIAA Aerospace Sciences Meeting, AIAA Paper 2017-0585, Grapevine, TX, USA, Jan 2017.

[Allmaras 2012] S.R. Allmaras, F.T. Johnson and P.R. Spalart. *Modifications and clarifications for the implementation of the Spalart-Allamas turbulence model.* In 7th International Conference on Computational Fluid Dynamics, Big Island, HI, USA, Jul 2012.

[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.

[Aubry 2009] R. Aubry and R. Löhner. *Generation of viscous grids at ridges and corners.* Int. J. Numer. Meth. Engng, vol. 77, pages 1247–1289, 2009.

[Bank 1993] R.E. Bank and R.K. Smith. *A posteriori error estimate based on hierarchical bases.* SIAM J. Numer. Anal., vol. 30, pages 921–935, 1993.

[Barral 2015] N. Barral. *Time-accurate anisotropic mesh adaptation for three-dimensional moving mesh problems.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2015.

[Barth 1994] T.J. Barth. *Aspects of unstructured grids and finite-volume solver for the Euler and Navier-Stokes equations.* In Von Karman Institute Lecture Series, 1994.

[Barth 2002] Timothy J. Barth and Mats G. Larson. *A Posteriori Error Estimates for Higher Order Godunov Finite Volume Methods on Unstructured Meshes.* 2002.

[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.

[Becker 1996] R. Becker and R. Rannacher. *A feed-back approach to error control in finite element methods: basic analysis and examples.* East-West J. Numer. Math., vol. 4, pages 237–264, 1996.

[Belme 2011] A. Belme. *Unsteady aerodynamic and adjoint method.* PhD thesis, Université de Nice - Sophia Antipolis, Nice, France, 2011.

[Belme 2018] A. Belme, F. Alauzet and A. Dervieux. *An a priori anisotropic Goal-Oriented Estimate for Viscous Compressible Flow and Application to Mesh Adaptation.* Journal of Computational Physics, 2018.

[Bibb 2006] K. Bibb, P. Gnoffo, M. Park and W. Jones. *Parallel, Gradient-Based Anisotropic Mesh Adaptation for Re-Entry Vehicle Configuratons.* In 9th AIAA/ASME Joint Thermophysics and Heat Transfer Conference, page 3579, 2006.

[Bodony 2005] D.J. Bodony and S.K. Lele. *On using large-eddy simulation for the prediction of noise from cold and heated turbulent jets.* Physics of Fluids, vol. 17, no. 8, page 085103, 2005.

[Boniface 2017] J.C. Boniface. *Rescaling of the Roe scheme in low Mach-number flow regions.* Journal of Computational Physics, vol. 328, pages 177 – 199, 2017.

[Borouchaki 1998] H. Borouchaki, F. Hecht and P.J. Frey. *Mesh gradation control.* Int. J. Numer. Meth. Engng, vol. 43, no. 6, pages 1143–1165, 1998.

[Bottasso 2002] C.L. Bottasso and D. Detomi. *A procedure for tetrahedral boundary layer mesh generation.* Engineering with Computers, vol. 18, no. 1, pages 66–79, 2002.

[Bourgault 2009] Y. Bourgault, M. Picasso, F. Alauzet and A. Loseille. *On the use of anisotropic error estimators for the adaptative solution of 3-D inviscid compressible flows.* Int. J. Numer. Meth. Fluids, vol. 59, pages 47–74, 2009.

[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.

[Catris 2000] S. Catris and B. Aupoix. *Density corrections for turbulence models.* Aerospace Science and Technology, vol. 4, pages 1–11, 2000.

[Chan 2017] W.K. Chan. *Best practices on overset structured mesh generation for the high-lift crm geometry.* In 55th AIAA Aerospace Sciences Meeting, page 0362, 2017.

[Chen 2007] L. Chen, P. Sun and J. Xu. *Optimal anisotropic meshes for minimizing interpolation errors in L^p-norm.* Math. Comp., vol. 76, no. 257, pages 179–204, 2007.

[Chitale 2014] K.C. Chitale, O. Sahni, M.S. Shephard, S. Tendulkar and K.E. Jansen. *Anisotropic adaptation for transonic flows with turbulent boundary layers.* AIAA Journal, vol. 53, no. 2, pages 367–378, 2014.

[Clément 1975] P. Clément. *Approximation by finite element functions using local regularization.* Revue Française d'Automatique, Informatique et Recherche Opérationnelle, vol. R-2, pages 77–84, 1975.

[Coulaud 2016] O. Coulaud and A. Loseille. *Very High Order Anisotropic Metric-Based Mesh Adaptation in 3D.* Procedia engineering, vol. 163, pages 353–365, 2016.

[Cournède 2006] P.-H. Cournède, B. Koobus and A. Dervieux. *Positivity statements for a Mixed-Element-Volume scheme on fixed and moving grids.* European Journal of Computational Mechanics, vol. 15, no. 7-8, pages 767–798, 2006.

[Darmofal 2013] D.L. Darmofal, S.R. Allmaras, M. Yano and J. Kudo. *An adaptive, higher-order discontinuous Galerkin finite element method for aerodynamics.* In AIAA conference paper, numéro 2013-2871, 2013.

[Debiez 2000] C. Debiez and A. Dervieux. *Mixed-Element-Volume MUSCL methods with weak viscosity for steady and unsteady flow calculations.* Comput. & Fluids, vol. 29, pages 89–118, 2000.

[Denton 1997] J.D. Denton. *Lessons from rotor 37.* Journal of Thermal Science, vol. 6, no. 1, pages 1–13, 1997.

[Derlaga 2017] Joseph M. Derlaga and Michael A. Park. Application of exact error transport equations and adjoint error estimation to aiaa workshops. American Institute of Aeronautics and Astronautics, 2017/11/09 2017.

[Dervieux 1992] A. Dervieux, L. Fezoui and F. Loriot. *On high resolution extensions of Lagrange-Galerkin finite element schemes.* RR-1703, INRIA Research Report, June 1992.

[Dervieux 2003] A. Dervieux, D. Leservoisier, P.L. George and Y. Coudière. *About theoretical and practical impact of mesh adaptations on approximation of functions and of solution of PDE.* Int. J. Numer. Meth. Fluids, vol. 43, pages 507–516, 2003.

[Ding 2018] F. Ding, J. Liu, C. Shen, W. Huang, Z. Liu and S. Chen. *An overview of waverider design concept in airframe/inlet integration methodology for air-breathing hypersonic vehicles.* Acta Astronautica, 2018.

[Diosady 2009] L.T. Diosady and D.L. Darmofal. *Preconditioning methods for discontinuous Galerkin solutions of the Navier–Stokes equations.* Journal of Computational Physics, vol. 228, no. 11, pages 3917–3935, 2009.

[Dobrzynski 2010] W. Dobrzynski. *Almost 40 years of airframe noise research: what did we achieve?* Journal of aircraft, vol. 47, no. 2, pages 353–367, 2010.

[Dobrzynski 2011] C. Dobrzynski, M. Melchior, L. Delannay and J.F. Remacle. *A mesh adaptation procedure for periodic domains.* International journal for numerical methods in engineering, vol. 86, no. 12, pages 1396–1412, 2011.

[Dwight 2008] R.P. Dwight. *Heuristic a posteriori estimation of error due to dissipation in finite volume schemes and application to mesh adaptation.* Journal of Computational Physics, vol. 227, no. 5, pages 2845–2863, 2008.

[Eca 2007] L. Eca, M. Hoekstra, A. Hay and D. Pelletier. *Verification of RANS Solvers with Manufactured Solutions.* Eng. with Comput., vol. 23, no. 4, October 2007.

[Fidkowski 2011] Krzysztof J. Fidkowski and David L. Darmofal. *Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics.* AIAA Journal, vol. 49, no. 4, pages 673–694, 2017/11/06 2011.

[Frazza 2015] L. Frazza, A. Hay and D. Pelletier. *A logarithmic formulation for low-Reynolds number turbulence models with adaptive wall-functions.* In 22nd AIAA Computational Fluid Dynamics Conference. American Institute of Aeronautics and Astronautics, 2017/11/09 2015.

[Frazza 2017] Loïc Frazza, Adrien Loseille and Frédéric Alauzet. *Anisotropic mesh adaptation for turbomachinery applications.* In 23rd AIAA Computational Fluid Dynamics Conference, page 3299, 2017.

[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.

[Gariépy 2013] M. Gariépy, B. Malouin, J.Y. Trépanier and E. Laurendeau. *Far-field drag decomposition applied to the drag prediction workshop 5 cases.* Journal of Aircraft, vol. 50, no. 6, pages 1822–1831, 2013.

[Garimella 2000] R.V. Garimella and M.S. Shephard. *Boundary layer mesh generation for viscous flow simulations.* Int. J. Numer. Meth. Fluids, vol. 49, pages 193–218, 2000.

[Gary 2015] Y. Gary, O. Gooch and F. Carl. Accuracy of discretization error estimation by the error transport equation on unstructured meshes - nonlinear systems of equations. American Institute of Aeronautics and Astronautics, 2017/11/09 2015.

[Gauci 2018] E. Gauci. *Goal-oriented metric-based mesh adaptation for unsteady CFD simulations involving moving geometries*. PhD thesis, Université Côte d'Azur, Nice, France, 2018.

[George 1991] P.L. George, F. Hecht and M.G. Vallet. *Creation of internal points in Voronoi's type method. Control and adaptation*. Adv. Eng. Software, vol. 13, no. 5-6, pages 303–312, 1991.

[Giles 2002] M.B. Giles and E. Süli. *Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality*. Acta numerica, vol. 11, pages 145–236, 2002.

[Gou 2018] J. Gou, X. Yuan and X. Su. *Adaptive mesh refinement method based investigation of the interaction between shock wave, boundary layer, and tip vortex in a transonic compressor*. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, vol. 232, no. 4, pages 694–715, 2018.

[Gourvitch 2004] N. Gourvitch, G. Rogé, I. Abalakin, A. Dervieux and T. Kozubskaya. *A tetrahedral-based super convergent scheme for aeroacoustics*. RR-5212, INRIA, May 2004.

[Gousseau 2011] P Gousseau, Bert Blocken and GJF Van Heijst. *CFD simulation of pollutant dispersion around isolated buildings: On the role of convective and turbulent mass fluxes in the prediction accuracy*. Journal of Hazardous Materials, vol. 194, pages 422–434, 2011.

[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.

[Guillard 1999] H. Guillard and C. Viozat. *On the behaviour of upwind schemes in the low Mach number limit*. Computers & Fluids, vol. 28, no. 1, pages 63 – 86, 1999.

[Harten 1983] A. Harten, P.D. Lax and B. Van Leer. *On upstream differencing and godunov-type schemes for hyperbolic conservation laws*. SIAM Revue, vol. 25, no. 1, pages 35–61, 1983.

[Hartman 2010] R. Hartman and P. Houston. *Error estimation and adaptive mesh refinement for aerodynamic flows*. In ADIGMA-A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications, pages 339–353. Springer, 2010.

[Hascoet 2013] L. Hascoet and V. Pascual. *The Tapenade Automatic Differentiation tool: principles, model, and specification*. ACM Transactions on Mathematical Software (TOMS), vol. 39, no. 3, page 20, 2013.

[Hay 2006] A. Hay and M. Visonneau. *Error estimation using the error transport equation for finite-volume methods and arbitrary meshes.* International Journal of Computational Fluid Dynamics, vol. 20, no. 7, pages 463–479, August 2006.

[Hecht 2008] F. Hecht. *Mesh generation and error indicator.* Summer school: More efficiency in finite element methods, 2008.

[Hu 2016] Y. Hu, C.F. Wagner, S.R. Allmaras, M.C. Galbraith and D.L. Darmofal. *Application of High-order Adaptive Methods to Reynolds-Averaged Navier-Stokes Test Cases.* AIAA Journal, vol. 54, no. 9, 2016.

[Huang 2010] W. Huang, L. Kamenski and X. Li. *A new anisotropic mesh adaptation method based upon hierarchical a posteriori error estimates.* J. Comp. Phys., vol. 229, pages 2179–2198, 2010.

[Huang 2018] A.C. Huang, S.R. Allmaras, M.C. Galbraith and D.L. Darmofal. *Well-posed Subsonic Inflow/Outflow Boundary Conditions for the Navier-Stokes Equations.* In 2018 AIAA Aerospace Sciences Meeting, page 0361, 2018.

[Ito 2002] Y. Ito and K. Nakahashi. *Unstructured mesh generation for viscous flow computations.* Proceedings of the 11th International Meshing Roundtable, pages 367–377, 2002.

[Jameson 1987] A. Jameson and S. Yoon. *Lower-Upper implicit schemes with multiple grids for the Euler equations.* AIAA Journal, vol. 25, no. 7, pages 929–935, 1987.

[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.

[Kalitzin 2005] G. Kalitzin, G. Medic, G. Iaccarino and P. Durbin. *Near-wall behavior of RANS turbulence models and implications for wall functions.* Journal of Computational Physics, vol. 204, no. 1, pages 265 – 291, 2005.

[Kim 2006] H. Kim and K. Kim. *Shape optimization of three-dimensional channel roughened by angled ribs with RANS analysis of turbulent heat transfer.* International Journal of heat and mass transfer, vol. 49, no. 21-22, pages 4013–4022, 2006.

[Knopp 2006] T. Knopp, T. Alrutz and D. Schwamborn. *A grid and flow adaptive wall-function method for RANS turbulence modelling.* Journal of Computational Physics, vol. 220, no. 1, pages 19 – 40, 2006.

[Koren 1993] B. Koren. *A robust upwind discretization method for advection, diffusion and source terms.* In Numerical methods for advection-diffusion problems. Vieweg, 1993.

[Kraft 2010] Edward Kraft. *Integrating Computational Science and Engineering To Re-Engineer the Aeronautical Development Process.* In 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, page 139, 2010.

[Lacasse 2004] David Lacasse, Eric Turgeon and Dominique Pelletier. *On the judicious use of the k–ε model, wall functions and adaptivity.* International Journal of Thermal Sciences, vol. 43, no. 10, pages 925–938, 2004.

[Larsson 2014] J. Larsson and Q. Wang. *The prospect of using large eddy and detached eddy simulations in engineering design, and the research required to get there.* Phil. Trans. R. Soc. A, vol. 372, no. 2022, page 20130329, 2014.

[Layton 2002] W. Layton, H. K. Lee and J. Peterson. *A defect-correction method for the incompressible Navier-Stokes equations.* Applied Mathematics and Computation, vol. 129, no. 1, pages 1–19, June 2002.

[Lee 2011] W.T. Lee. *Tridiagonal matrices: Thomas algorithm.* MS6021, Scientific Computation, University of Limerick, 2011.

[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.

[Levy 2014] D.W. Levy, K.R. Laflin, E.N. Tinoco, J.C. Vassberg, M. Mani, B. Rider, C.L. Rumsey, R.A. Wahls, J.H. Morrison, O. Brodersen*et al. Summary of data from the fifth computational fluid dynamics drag prediction workshop.* Journal of Aircraft, vol. 51, no. 4, pages 1194–1213, 2014.

[Li 2003] X. Li, M.S. Shephard and M.W. Beal. *Accounting for curved domains in mesh adaptation.* Int. J. Numer. Meth. Engng, vol. 58, pages 247–276, 2003.

[Liu 2008] D. Liu and P. Lin. *A numerical study of three-dimensional liquid sloshing in tanks.* Journal of Computational physics, vol. 227, no. 8, pages 3921–3939, 2008.

[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.

[Loseille 2007a] 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 18th AIAA Computational Fluid Dynamics Conference, AIAA Paper 2007-4186, Miami, FL, USA, Jun 2007.

[Loseille 2007b] 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.

[Loseille 2008] A. Loseille. *Adaptation de maillage anisotrope 3D multi-échelles et ciblée à une fonctionnelle pour la mécanique des fluides. Application à la prédiction haute-fidélité du bang sonique.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2008. (in French).

[Loseille 2010] A. Loseille, A. Dervieux and F. Alauzet. *Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations.* J. Comp. Phys., vol. 229, pages 2866–2897, 2010.

[Loseille 2011a] A. Loseille and F. Alauzet. *Continuous mesh framework. Part I: well-posed continuous interpolation error.* SIAM J. Numer. Anal., vol. 49, no. 1, pages 38–60, 2011.

[Loseille 2011b] A. Loseille and F. Alauzet. *Continuous mesh framework. Part II: validations and applications.* SIAM J. Numer. Anal., vol. 49, no. 1, pages 61–86, 2011.

[Loseille 2011c] A. Loseille and R. Löhner. *Boundary layer mesh generation and adaptivity.* In 49th AIAA Aerospace Sciences Meeting, AIAA Paper 2011-894, Orlando, FL, USA, Jan 2011.

[Loseille 2013] A. Loseille and R. Löhner. *Cavity-Based Operators for Mesh Adaptation.* In 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, AIAA Paper 2013-0152, Dallas, TX, USA, Jan 2013.

[Luo 1998] H. Luo, J.D. Baum and R. Löhner. *A fast, matrix-free implicit method for compressible flows on unstructured grids.* J. Comp. Phys., vol. 146, pages 664–690, 1998.

[Luo 2001] H. Luo, J.D. Baum and R. Löhner. *An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids.* Comput. & Fluids, vol. 30, pages 137–159, 2001.

[Marcum 1982] D. Marcum and N. Weatherill. *Unstructured grid generation using iterative point insertion and local reconnection.* AIAA Journal, vol. 33, no. 9, pages 1619–1625, 1982.

[Marcum 1996] D.L. Marcum. *Adaptive unstructured grid generation for viscous flow applications.* AIAA Journal, vol. 34, no. 8, pages 2440–2443, 1996.

[Marcum 1998] D.L. Marcum. *Unstructured Grid Generation Using Automatic Point Insertion and Local Reconnection.* In The Handbook of Grid Generation, Edited by J.F. Thompson, B. Soni, and N.P. Weatherill, chapitre 18, pages 1–31. CRC Press, 1998.

[Marcum 2014] David Marcum and Frédéric Alauzet. *Aligned metric-based anisotropic solution adaptive mesh generation.* Procedia Engineering, vol. 82, pages 428–444, 2014.

[Mavriplis 1998] D. Mavriplis. *On convergence acceleration techniques for unstructured meshes.* In 29th AIAA, Fluid Dynamics Conference, page 2966, 1998.

[Mavriplis 2009] D.J. Mavriplis, J.C. Vassberg, E.N. Tinoco, M. Mani, O.P. Brodersen, B. Eisfeld, R.A. Wahls, J.H. Morrison, T. Zickuhr, D. Levy*et al.* *Grid quality and resolution issues from the drag prediction workshop series.* Journal of Aircraft, vol. 46, no. 3, pages 935–950, 2009.

[Menier 2014] V. Menier, A. Loseille and F. Alauzet. *CFD validation and adaptivity for viscous flow simulations.* In 7th AIAA Theoretical Fluid Mechanics Conference. AIAA, 2014.

[Menier 2015] V. Menier. *Mesh Adaptation For the High Fidelity Prediction of Viscous Phenomena and Their Interactions. Application to Aeronautics.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2015.

[Menter 1993] F. Menter. *Zonal Two Equation k-w Turbulence Models For Aerodynamic Flows.* In 23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference, volume 1993, Feb 1993.

[Mirebeau 2010] J-M. Mirebeau. *Optimal meshes for finite elements of arbitrary order.* Constructive approximation, vol. 32, no. 2, pages 339–383, 2010.

[Miron 2014] P. Miron, J. Vétel and A. Garon. *On the use of the finite-time Lyapunov exponent to reveal complex flow physics in the wake of a mechanical valve.* Experiments in fluids, vol. 55, no. 9, page 1814, 2014.

[Norton 2006] T. Norton and D. Sun. *Computational fluid dynamics (CFD)–an effective and efficient design and analysis tool for the food industry: a review.* Trends in Food Science & Technology, vol. 17, no. 11, pages 600–620, 2006.

[Olivier 2011] G. Olivier. *Anisotropic metric-based mesh adaptation for unsteady CFD simulations involving moving geometries.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2011.

[Park 2010] M.A. Park and J.R. Carlson. *Turbulent output-based anisotropic adaptation.* In 48th AIAA Aerospace Sciences Meeting, AIAA Paper 2010-0168, Orlando, FL, USA, Jan 2010.

[Park 2016] M.A. Park, A. Loseille, J. Krakos, T.R. Michal and J.J. Alonso. *Unstructured grid adaptation: Status, potential impacts, and recommended investments towards CFD 2030.* In 46th AIAA Fluid Dynamics Conference, page 3323, 2016.

[Pierce 2004] Niles A. Pierce and Michael B. Giles. *Adjoint and defect error bounding and correction for functional estimates.* Journal of Computational Physics, vol. 200, no. 2, pages 769 – 794, 2004.

[Piperno 1998] S. Piperno and S. Depeyre. *Criteria for the design of limiters yielding efficient high resolution TVD schemes.* Comput. & Fluids, vol. 27, no. 2, pages 183–197, 1998.

[Pirzadeh 1996] S. Pirzadeh. *Three-dimensional unstructured viscous grids by the advancing-layers method.* AIAA journal, vol. 34, no. 1, pages 43–49, 1996.

[Ranut 2014] P. Ranut, G. Janiga, E. Nobile and D. Thévenin. *Multi-objective shape optimization of a tube bundle in cross-flow.* International Journal of Heat and Mass Transfer, vol. 68, pages 585–598, 2014.

[Reuther 1999] J.J. Reuther, A. Jameson, J. Alonso, M.J. Rimlinger and D. Saunders. *Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1.* Journal of aircraft, vol. 36, no. 1, pages 51–60, 1999.

[Roache 1998] Patrick J Roache. Verification and validation in computational science and engineering, volume 895. Hermosa Albuquerque, NM, 1998.

[Roe 1981] P.L Roe. *Approximate Riemann solvers, parameter vectors, and difference schemes.* Journal of Computational Physics, vol. 43, no. 2, pages 357 – 372, 1981.

[Rogé 2008] G. Rogé and L. Martin. *Goal-oriented anisotropic grid adaptation Adaptation de maillage anisotrope orientée objectif.* Comptes Rendus Mathématiques, vol. 346, no. 19-20, pages 1109–1112, 2008.

[Rumsey 2011] C.L. Rumsey, JP. Slotnick, M. Long, RA. Stuever and TR. Wayman. *Summary of the first AIAA CFD high-lift prediction workshop.* Journal of Aircraft, vol. 48, no. 6, pages 2068–2079, 2011.

[Rumsey 2018] C.L. Rumsey, J.P. Slotnick and A.J. Sclafani. *Overview and Summary of the Third AIAA High Lift Prediction Workshop.* In 2018 AIAA Aerospace Sciences Meeting, page 1258, 2018.

[Sahni 2009] O. Sahni, K.E. Jansen, C.A. Taylor and M.S. Shephard. *Automated adaptive cardiovascular flow simulations.* Engineering with Computers, vol. 25, no. 1, page 25, 2009.

[Selmin 1996] V. Selmin and L. Formaggia. *Unified construction of finite element and finite volume discretizations for compressible flows.* Int. J. Numer. Meth. Engng, vol. 39, pages 1–32, 1996.

[Seo 2010] J.H. Seo, D.M. Seol, J.H. Lee and S.H. Rhee. *Flexible CFD meshing strategy for prediction of ship resistance and propulsion performance.* International Journal of Naval Architecture and Ocean Engineering, vol. 2, no. 3, pages 139–145, 2010.

[Sharov 1997] D. Sharov and K. Nakahashi. *Reordering of hybrid unstructured grids for Lower-Upper Symmetric Gauss-Seidel computations.* AIAA Journal, vol. 36, no. 1, pages 484–486, 1997.

[Sharov 2000] D. Sharov, H. Luo, J.D. Baum and R. Löhner. *Implementation of unstructured grid GMRES+LU-SGS method on shared-memory, cache-based parallel computers.* AIAA Paper, vol. 2000-0927, 2000.

[Sini 1996] Jean-François Sini, Sandrine Anquetin and Patrice G Mestayer. *Pollutant dispersion and thermal effects in urban street canyons.* Atmospheric environment, vol. 30, no. 15, pages 2659–2677, 1996.

[Skinner 2017] S.N. Skinner and H. Zare-Behtash. *State-of-the-art in aerodynamic shape optimisation methods.* Applied Soft Computing, 2017.

[Slotnick 2014] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie and D. Mavriplis. *CFD vision 2030 study: A path to revolutionary computational aerosciences: NASA.* Rapport technique, Nasa, 2014.

[Soto 2003] O. Soto, R. Löhner and F. Camelli. *A linelet preconditioner for incompressible flow solvers.* International Journal of Numerical Methods for Heat & Fluid Flow, vol. 13, no. 1, pages 133–147, 2003.

[Souliez 2002] F.J. Souliez, L.N. Long, P.J. Morris and A. Sharma. *Landing gear aerodynamic noise prediction using unstructured grids.* International Journal of Aeroacoustics, vol. 1, no. 2, pages 115–135, 2002.

[Spalart 1992] P.R. Spalart and S.R. Allmaras. *A one-equation turbulence model for aerodynamic flows.* In 30th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-92-0439, Reno, NV, USA, Jan 1992.

[Spalart 2000] P.R. Spalart. *Strategies for turbulence modelling and simulations.* International Journal of Heat and Fluid Flow, vol. 21, no. 3, pages 252–263, 2000.

[Sun 2017] Y. Sun and H. Smith. *Review and prospect of supersonic business jet design.* Progress in Aerospace Sciences, vol. 90, pages 12–38, 2017.

[Vassberg 2002] J. Vassberg, P. Buning and C. Rumsey. *Drag Prediction for the DLR-F4 Wing/Body using OVERFLOW and CFL3D on an Overset Mesh.* In 40th AIAA Aerospace Sciences Meeting & Exhibit, page 840, 2002.

[Vassberg 2003] J. Vassberg, M. DeHaan and T. Sclafani. *Grid generation requirements for accurate drag predictions based on OVERFLOW calculations.* In 16th AIAA computational fluid dynamics conference, page 4124, 2003.

[Venditti 2000] D.A. Venditti and D. Darmofal. *Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow.* vol. 164, pages 204–227, 10 2000.

[Venditti 2002a] D.A. Venditti. *Grid Adaptation for Functional Outputs of Compressible Flow Simulations.* PhD thesis, Massachusetts Institute of Technology, 2002.

[Venditti 2002b] D.A. Venditti and D.L. Darmofal. *Grid adaptation for functional outputs: application to two-dimensional inviscid flows.* J. Comp. Phys., vol. 176, no. 1, pages 40–69, 2002.

[Venditti 2003] D.A. Venditti and D.L. Darmofal. *Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows.* J. Comp. Phys., vol. 187, no. 1, pages 22–46, 2003.

[Vétel 2009] J. Vétel, A. Garon and D. Pelletier. *Lagrangian coherent structures in the human carotid artery bifurcation.* Experiments in fluids, vol. 46, no. 6, pages 1067–1079, 2009.

[Viozat 1997] C. Viozat. *Implicit Upwind Schemes for Low Mach Number Compressible Flows.* Rapport technique RR-3084, INRIA, January 1997.

[Woeber 2017] C.D. Woeber, E.J. Gantt and N.J. Wyman. *Mesh generation for the nasa high lift common research model (hl-crm).* In 55th AIAA Aerospace Sciences Meeting, page 0363, 2017.

[Yano 2011] M. Yano, J.M. Modisette and D.L. Darmofal. *The importance of mesh adaptation for higher-order discretizations of aerodynamic flows.* In AIAA conference paper, numéro 2011-3852, 2011.

[Yano 2012a] M. Yano. *An Optimization Framework for Adaptive High-Order Discretization of Partial Differential Equations on Anisotropic Simplex Meshes.* PhD thesis, Massachusetts Institute of Technology, 2012.

[Yano 2012b] M. Yano and D.L. Darmofal. *An Optimization-Based Framework for Anisotropic Simplex Mesh Adaptation.* Journal of Computational Physics, vol. 231, no. 22, pages 7626–7649, 2012.

[Zienkiewicz 1992a] O.C. Zienkiewicz and J.Z. Zhu. *The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique.* Int. J. Numer. Meth. Engng, vol. 33, no. 7, pages 1331–1364, 1992.

[Zienkiewicz 1992b] O.C. Zienkiewicz and J.Z. Zhu. *The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity.* Int. J. Numer. Meth. Engng, vol. 33, no. 7, pages 1365–1380, 1992.

3D anisotropic mesh adaptation for Reynolds Averaged Navier-Stokes simulations

Abstract:

The fast and reliable simulation of turbulent flow using Reynolds Averaged Navier Stokes (RANS) models is a major financial issue for many industries. Paradoxically, although they have become the norm today, the accuracy of these simulations depends greatly on the quality of the mesh used. With the increasing complexity of geometries and simulated flows, as well as requirements in terms of fidelity, the generation of appropriate meshes has become a key link in the chain of computation. Yet it is still based on the subjective experience of a qualified engineer. Beyond the obstacle that this human involvement represents for the automation of the computation chain in question, it is impossible to determine clear and universal rules for the generation of a mesh. This is all the more true when one is confronted with a complex flow. We show in this thesis the ability of modern numerical schemes to simulate turbulent flows on fully unstructured meshes generated automatically using mesh adaptation methods. We present the implementation of different versions of the Spalart-Allmaras model as well as the numerical choices guaranteeing a sufficient robustness of the solver in order to not require a structured boundary layer. We then introduce the error analysis necessary to propose different error estimators for mesh optimization. This methodology is tested on various external aerodynamic and turbomachinery test cases and compared to traditional mesh generation methods. We show the ability of mesh adaptation methods to automatically generate optimal mesh sizes for RANS simulations on realistic and complex geometries.

Keywords: CFD, RANS mesh adaptation, metric-based mesh adaptation, continuous mesh framework, adjoint-based mesh adaptation