Feature-based and goal-oriented anisotropic mesh adaptation for RANS applications in aeronautic and aerospace

F. Alauzet*, L. Frazza

*INRIA Saclay Île-de-France, Projet Gamma, 1, rue Honoré d’Estienne d’Orves, 91126 Palaiseau, France

Abstract

The scope of this paper is to demonstrate the viability of unstructured anisotropic mesh adaptation for commercial aircraft drag and high-lift prediction studies. The main achievement of this work is to demonstrate that mesh-independent certified numerical solutions can be obtained thanks to anisotropic mesh adaptation and that it is possible to run high-fidelity CFD on unstructured adapted meshes composed only of tetrahedra which is fundamental to design robust meshing process for complex geometries. It also points out the early capturing property of the solution-adaptive process in the sense that accurate output functional values are obtained on relatively coarse adapted meshes. On a more practical point of view, this paper demonstrates how mesh adaptation, thanks to its automation, is able to generate meshes that are extremely difficult to envision and almost impossible to generate manually, leading to highly accurate numerical solutions. Moreover, as the process can start from any coarse initial mesh, it greatly simplifies the overall meshing process. This study also analyze the influence of different strategies in the mesh adaptation algorithm and in the error analysis which are key components of the process. Several error estimates are considered: feature-based ones which are based on the standard multiscale $L^p$ interpolation error estimate and goal-oriented ones to control the error on output functionals which rely on an accurate computation of the adjoint state. The adjoint problem proves to be a stiff problem for RANS equations, failing to converge the adjoint state to machine zero may impact negatively the adaptive process. The maturity of the solution-adaptive process is demonstrated on numerous drag and high-lift prediction cases. It has also excelled in sonic boom and turbomachine applications.

Keywords: Anisotropic mesh adaptation, Finite volume - finite element flow solver, Reynolds Average Navier-Stokes, aeronautics, feature-based error estimate, goal-oriented error estimate, drag prediction, high-lift prediction.

1. Introduction

Computation fluid dynamics (CFD) is now widespread in industry and routinely used by engineers. This is particularly true in aeronautics and aerospace where CFD is used to develop new digital design. High-fidelity and certification of numerical solutions are of utmost importance to avoid misconception or worse catastrophic events because of a bad numerical design with poor performances. Despite the progress of numerical simulation and its large distribution in the industry, accurate and certified numerical solutions of the Navier-Stokes equations are still not a solved problem. Indeed, uncertainties or errors can come from several sources: i) modeling error (for example via turbulence models or initial conditions), ii) discretization error (due to the mesh), iii) geometry error (due to the CAD representation of the design) and iv) the numerical scheme (dissipation and dispersion) and its implementation errors in the considered software. They are inherent to all numerical simulations. In this work, we assume that there is no modeling errors (here we consider the Reynolds Average Navier-Stokes model), we have the correct geometry representing the design, and that the flow solver has been verified and validated using the Software Verification and Validation (V&V) process [74, 80]. But, under these assumptions, if one considers a complex physical problem on a complex geometry, it does not guarantee to obtain the correct solution because of discretization errors. Indeed, this kind of error is becoming preponderant because numerical simulations are more and more complex at the same time from a geometric point of view and from a physical point of view. This paper proposes an automatic methodology, well-founded mathematically, to control and reduce errors coming from the discretization of the problem.

The generation of meshes for accurate and reliable numerical simulations for modern Reynolds-Averaged Navier-Stokes (RANS) applications is a time consuming and tedious task. Traditional processes in the aerospace industry for CFD simulations rely heavily on thumb rules or the experience and intuition of a skilled engineer to predict and adapt the mesh prescription to the
flow. Following meshing guidelines slows down the mesh generation process leading to a prohibitive cost in CPU time in the numerical simulation pipeline. This lack of automation is an impediment for many applications where the numerical simulation is a part of a whole. The inadequacy of this practice even for geometries frequently encountered in engineering practice has been highlighted in studies of the AIAA CFD Drag Prediction Workshops [69] and AIAA High-Lift Prediction Workshops [84, 85]. These studies suggest that the range of scales present in turbulent flows cannot be adequately resolved using meshes generated following what is considered to be the present best practices. Moreover, when dealing with complex geometries and the associated complexity of the simulated flow patterns, it is almost impossible for a user to anticipate the various physical features involved and prescribe an appropriate mesh1. A-priori refinements of large zones are used instead, leading to the unnecessary consumption of large amounts of resources. It is now fundamental to propose automatic numerical process where manual intervention are removed from the loop, and numerical process for which we can certify the numerical solution under the considered geometrical and physical model, in the sense that the discretization error does not anymore influence the prediction. In that case, we talk about mesh-independent numerical solution. In this regard, anisotropic mesh adaptation is considered as the future, as stated in the NASA report “CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences” [88] and the study dedicated to mesh adaptation [76].

This is why mesh adaptation strategies that automatically analyze the flow solutions and adapt the mesh to it can bring a significant improvement in terms of CPU times, memory consumption and accuracy through error control. This will bring automation, independence of the simulation to the initial generated mesh and, importantly, will remove the human from the numerical simulation pipeline [29, 37]. However, these strategies in 3D have been limited for long time to inviscid applications [6, 7, 59] due to the lack of appropriate error estimates and mesh requirements in RANS applications [76], in particular due the extreme anisotropy obtained near the geometry for such adapted 3D meshes. An important remark is that in the case of turbulent flows with a high Reynolds number, the anisotropy of the physical phenomena is such that adaptation methods like AMR [1, 17] or unstructured isotropic mesh [18, 35, 51] are in practice unusable because meshing adequately the boundary layer would be prohibitive. Anisotropic mesh adaptation methods are essential.

On the solver side, it is generally acknowledged that it is not possible to run CFD high-fidelity simulations in aeronautics using meshes composed only of tetrahedra. This is a misunderstanding on several points. This is mainly due to the existence of a boundary layer in the near wall region. Indeed, this physical phenomena is extremely localized (very thin with respect to the geometry) and highly anisotropic. Fortunately, we know where it occurs, close to the wall, therefore it was possible to generate highly anisotropic pseudo-structured meshes in that region using the surface mesh of the geometry as support [11, 34, 39, 41, 50, 67, 83]. Unconsciously, setting-up such a process is similar to the anisotropic mesh adaptation methodology: these pseudo-structured boundary layer meshes are fitting the boundary layer physics in term of sizes (small enough to capture the boundary layer) and in directions (following the anisotropy of the boundary layer). Unfortunately other physical phenomena that are localized and highly anisotropic (such as shock wave, wake, shear layer, tip vortices, ...) are far from the surface, difficult to locate a priori, and with no geometrical structure from which a specific adaptation of the mesh can be created. Consequently, these physical phenomena are generally not accurately captured (or missed) because of a poor spatial discretization, or require tedious time consuming and expensive approaches. We can also say that the boundary layer region is not optimally discretized with classical methods because, for instance, it becomes difficult to take into account its thickness which growth along the geometry or because there is a shock / boundary layer interaction. Similarly, classic laws used to determine the growth of the layers of element cannot encompass the various behaviors in the boundary layers.

Another argument against meshes composed only of tetrahedra is topological. As we already know the location of the boundary layer and the fact that we can use the surface mesh as a starting point to generate the mesh in the boundary layer region, it is possible to generate a pseudo-structured mesh in particular by following the surface normal which allows potentially more precise gradients (at least in the normal direction). However, as can be seen in Figure 1 (top middle picture), the construction of classical (median) finite volume cell on such quasi-structured tetrahedral mesh introduces a directional bias that is found in the solution. Hence the conclusion to use multi-element meshes composed of tetrahedra, prisms, pyramids, hexahedra or to change the finite volume cells for vertex-centered methods [13, 36] (see top right picture of Figure 1). But, the first choice does not treat properly geometric singularities (corners, ridges, ...), junction of boundary layer (wing-body junction, ...) and boundary layer mesh collision in enclosed regions [8, 11]. The second choice is difficult to apply in 3D in a general context because of the specific splitting of hexahedra into tetrahedra. The other possibility is to use completely unstructured adapted meshes to no longer have this bias, see Figure 1 (bottom middle picture).

One of the main achievement of this work is to demonstrate that mesh-independent certified numerical solutions can be obtained thanks to anisotropic mesh adaptation and that it is possible to run high-fidelity CFD on unstructured adapted meshes composed only of tetrahedra.

This work proposes a metric-based anisotropic mesh adaptation strategy based on unstructured meshes composed only of

1The American Institute of Aeronautics and Astronautics.  
2If we can, it means somehow that we already know the solution of the problem and the numerical simulation is unnecessary.
tetrahedra which is able to automatically manage complex geometries and technological effects, and adapt the mesh in size and in direction with respect to the characteristic of the flow. Starting from an initial coarse non-adapted mesh which is generated without any \textit{a priori} knowledge on the solution, metric-based anisotropic mesh adaptation consists in iteratively modifying the computational mesh in order to better capture all the flow features and to obtain an improved numerical solution for a given number of degrees of freedom. The size and the orientation of the elements of the mesh being given by an error estimate which evaluates the error in the computation of the solution due to the discretization. This process can be combined with a convergence study by incrementing gradually the size of the mesh, and thus giving the capability to check the independence of the numerical solution to the domain discretization.

If anisotropic mesh adaptation has proven its reliability for inviscid flows \cite{6, 59}, additional challenges remained to be solved to have the full gain of adaptivity for turbulent flows. This paper presents significant improvements have been made on mesh adaptation for RANS problems. These improvements have concerned all stages of the mesh adaptation loop:

- The flow solver with improvement in accuracy, convergence and robustness, thanks to a numerical scheme based on the mixed finite element - finite volume discretization of the equations, a new low dissipative limiter, a careful differentiation of the terms in the implicit resolution, and a specific strategy for the pseudo-transient continuation method. This numerical scheme does not require structured mixed elements boundary layer meshes and is able to compute accurately aerodynamic forces on highly anisotropic unstructured tetrahedral meshes, which reduces the quality requirement on the meshing side.

- The adaptive remesher by designing a robust cavity based remeshing operator which is able to handle highly anisotropic unstructured tetrahedral meshes and robust vertex projections on the CAD surface when considering such meshes. We refer to the following references for more details \cite{54, 58, 62, 64}.

- The development of an improved goal-oriented error estimate for RANS, it relies on an accurate computation of the adjoint state and its second derivatives. And, the best choice of the $L^p$ norm for the multiscale error estimate when feature-based anisotropic mesh adaptation is considered for RANS simulations.

Figure 1: Illustration of the shape of the finite volume cells on a tetrahedra pseudo-structured mesh (top), an unstructured mesh (middle), and an anisotropic adapted mesh (bottom). Left, view of the surface meshes. Middle, finite volume cells build with the rule of median. Right, finite volume containment cells \cite{13, 36} designed for structured meshes.
• The adjoint solution proves to be a stiff problem for RANS equations (drag and high-lift applications). Failing to converge the adjoint state to machine zero (at least ten orders of magnitude) may impact negatively the goal-oriented adaptive process. Therefore, a new strategy with a stronger preconditioner has been proposed to solve efficiently the adjoint state on large scale applications such as realistic drag or high-lift configurations.

• Clever mesh adaptation algorithm which also enables mesh-convergence studies.

One of the key component in the solution-adaptive process concerns the computation of the error estimate. Thus, on the numerical side, we analyze the influence of different strategies in the error analysis. We consider two kind of error estimates: feature-based [6] and goal-oriented [59, 16] error estimates. The feature-based mesh adaptation based on the multiscale error estimate is geometric and focuses on the best adapted mesh representing the solution. Mathematically, these adapted meshes minimize the interpolation error in $L^p$ norm of a given sensor for given number of degrees of freedom. This error estimate is easy to use as it only requires the computation of second derivatives of the chosen sensor. The goal-oriented error estimate minimizes the error of a given engineering output functional for given number of degrees of freedom. It is more complex to use as it requires an adjoint state. We will consider the laminar goal-oriented error estimate of [16] and the new RANS goal-oriented error estimate of this paper. These comparisons will be made on drag prediction and high-lift prediction applications in two and three dimensions for a variety of geometries.

To conclude this introduction, this publication brings several contributions in the field of numerical simulations for aeronautic applications:

• we prove that it is possible to compute high-fidelity turbulent solutions (using RANS model) on unstructured meshes composed only of tetrahedra while most of flow solvers requires multi-elements or structured meshes. This is of utmost importance because tetrahedra mesh generator are able to mesh complex geometries automatically and manage easily the variation of element size in the mesh

• thanks to anisotropic mesh adaptation, the meshing process is greatly simplified, fully automatic and able to handle complex geometries. Moreover, it is optimal w.r.t the considered error estimate leading the very accurate numerical solutions as the mesh fits to the physics without any a priori knowledge on the solution

and as regards the numerical results, the contributions are

• we demonstrate that the solution-adaptive process always converge toward the same solution whatever the considered initial mesh (which simplifies drastically the initial mesh generation process) and whatever the considerer error estimate

• early capturing is achieved with advanced error estimate, we are thus able to obtain very accurate predictions with small size meshes

• we are able to achieve mesh convergence to guarantee that the computed numerical solution is independent of the mesh (at a low threshold), i.e., discretization error in the solution does not impact the prediction (at a low threshold).

The paper is outlined as follows. Section 2 described the physical model which is given by the RANS equations using the Spalart-Allmaras turbulence model. We state the aerodynamic output functionals that will be used in the numerical study. Section 3 presents the anisotropic mesh adaptation algorithm and the strategy to automatically perform mesh-convergence study. A specific mesh adaptation algorithm is provided for high-lift predictions studies at high-angle of attack. Section 4 describes the numerical methods implemented in the flow solver. A new low dissipative limiter is given and we insist on the implicit differentiation of the adjoint state to machine zero (at least ten orders of magnitude) may impact negatively the goal-oriented adaptive process. In Section 5 the description of the adjoint solver and the differentiation of the aerodynamic output functionals are provided. Section 6 recalls the two error estimates that are considered in this work: the feature-based and the goal-oriented error estimates. Finally, the benefits using metric-based anisotropic mesh adaptation for RANS aeronautic simulations are pointed out on drag and high-lift predictions applications in Sections 8 and 9.

2. Modelling equations

2.1. The Navier-Stokes equations

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

\[
\begin{align*}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0, \\
\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p &= \nabla \cdot \mathbf{T}, \\
\frac{\partial (\rho E)}{\partial t} + \nabla \cdot ((\rho E + p) \mathbf{u}) &= \nabla \cdot (\mathbf{T} \cdot \mathbf{u}) + \nabla \cdot (\lambda \nabla T),
\end{align*}
\]
where $\rho$ denotes the density ($kg/m^3$), $u$ the velocity ($m/s$), $E$ the total energy per mass ($m^2.s^{-2}$), $p$ the pressure ($N/m^2$), $T$ the temperature ($K$), $\mu$ the laminar dynamic viscosity ($kg/(m.s)$) and $\lambda$ the laminar conductivity. $T$ the stress tensor:

$$T = \mu \left( (\nabla \otimes u + (\nabla \otimes u)^\top) - \frac{2}{3} \nabla \cdot u \mathbf{I} \right) = \mu \tau,$$

and, in 3D, $u = (u, v, w)$ and

$$\nabla \cdot u = \begin{bmatrix} u_x + v_y + w_z & 0 & 0 \\ 0 & u_x + v_y + w_z & 0 \\ 0 & 0 & u_x + v_y + w_z \end{bmatrix},$$

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 $\mu$ and $\lambda$ as a function of the dimensional temperature $T$ is defined by Sutherland’s law:

$$\mu = \mu_\infty \left( \frac{T}{T_\infty} \right)^{\frac{1}{2}} \left( \frac{T_\infty + Su}{T + Su} \right) \quad \text{and} \quad \lambda = \lambda_\infty \left( \frac{T}{T_\infty} \right)^{\frac{1}{2}} \left( \frac{T_\infty + Su}{T + Su} \right),$$

where $Su = 110$ is the Sutherland temperature and the index $\infty$ denotes reference quantities. The relation linking $\mu$ and $\lambda$ is expressed from the Prandtl laminar number:

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

where $C_p$ is the specific heat at constant pressure.

In the case of the Reynolds Average Navier-Stokes (RANS) numerical simulations, the Navier-Stokes equations are completed by a turbulence model defined by one or more equations (here the Spalart-Allmaras model given in Section 2.2), and the laminar dynamic viscosity $\mu$ (resp. the laminar conductivity $\lambda$) are replaced in the above equations by the sum between the laminar and the turbulent dynamic viscosity (resp. conductivity): $\mu + \mu_t$ (resp. $\lambda + \lambda_t$). The turbulent dynamic viscosity $\mu_t$ is given by the turbulence model and the turbulent conductivity $\lambda_t$ is expressed from the Prandtl turbulent number:

$$Pr_t = \frac{\mu_t C_p}{\lambda_t} \quad \text{with} \quad Pr_t = 0.9 \quad \text{for (dry) air}.$$

### 2.2. Turbulence modeling: the Spalart-Allmaras one equation model

According to the standard approach to turbulence modeling based upon the Boussinesq hypothesis, the turbulence is modeled with an eddy viscosity $\mu_t$, which is added to the laminar (or dynamic) viscosity $\mu$. The dynamic viscosity is usually taken to be a function of the temperature, whereas $\mu_t$ is obtained using a turbulence model. Here we choose the Spalart-Allmaras one equation turbulence model [90] given by the following equation:

$$\frac{\partial \tilde{v}}{\partial t} + u \cdot \nabla \tilde{v} = c_{b1}[1 - f_{\delta 2}]\tilde{S}\tilde{v} - \left[ c_{b1}f_{\delta 0} - \frac{c_{b1}}{\kappa} f_{\delta 2} \right] \left( \frac{\tilde{v}^2}{\delta} \right) + \frac{1}{\sigma} \left[ \nabla \cdot ((\nu + \tilde{v})\nabla \tilde{v}) + c_{b1} \nabla \|\nabla \tilde{v}\|^2 \right] + f_{\delta 1} \Delta u^2,$$

where $\tilde{v}$ 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., $f_{\delta 1} = 0$. Moreover, some implementations also ignore the $f_{\delta 2}$ term as it is argued that if the trip is not included, then $f_{\delta 2}$ is not necessary [30]. 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 Volume – Finite Element method. Indeed, Equation (3) can be decomposed into the following terms:

$$\frac{\partial \rho \tilde{v}}{\partial t} + \mathbf{u} \cdot \nabla \rho \tilde{v} = \underbrace{c_{b1} \tilde{S} \rho \tilde{v}}_{\text{convection}} - \underbrace{c_{b1} f_{\delta 0} \rho \left( \frac{\tilde{v}^2}{\delta} \right)}_{\text{production}} - \underbrace{\frac{\rho}{\sigma} \nabla \cdot ((\nu + \tilde{v})\nabla \tilde{v})}_{\text{destruction}} + \underbrace{\frac{c_{b1} \rho \sigma}{\sigma} \|\nabla \tilde{v}\|^2}_{\text{diffusion}}.$$

Note 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 Aupeix [22]. The turbulent eddy viscosity is computed from:

$$\mu_t = \rho \tilde{v} f_{\delta 1},$$

where

$$f_{\delta 1} = \frac{\chi^3}{\chi^3 + c_{\nu 1}} \quad \text{and} \quad \chi = \frac{\tilde{v}}{\nu} \quad \text{with} \quad \nu = \frac{\mu}{\rho}. $$
Additional definitions are given by the following equations:

\[ f_{i2} = 1 - \frac{\chi}{1 + \chi f_1} \quad \text{and} \quad \hat{S} = \|\nabla \times \mathbf{u}\| + \frac{\bar{\nu}}{\kappa^2 d^2} f_{i2} \]

\( d \) 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

\[
\begin{align*}
\sigma &= \frac{2}{3}, & c_{b1} &= 0.1355, & c_{b2} &= 0.622, & \kappa &= 0.41, \\
\epsilon_{w1} &= \frac{c_{b1}}{\kappa} + \frac{1 + \epsilon_{b2}}{\sigma}, & c_{w2} &= 0.3, & c_{w3} &= 2, & c_{s1} &= 7.1.
\end{align*}
\]

Finally, the function \( f_w \) is computed as:

\[
 f_w = g \left( \frac{1 + c_{w3}^6}{8^{\frac{1}{6}} + c_{w3}^6} \right)^{1/6} \quad \text{with} \quad g = r + c_{w2}(\hat{r} - r) \quad \text{and} \quad r = \min \left( \frac{\bar{\nu}}{\kappa \epsilon^{\frac{1}{2}}} \cdot 10 \right).
\]

In three dimensions, we consider the QCR version of the Spalart-Allmaras turbulence model, described in [89], which modified the turbulent stress tensor:

\[
\mathcal{T} = \mu \tau + \mu_t (\tau - \tau_{QCR}),
\]

with the \( \tau_{QCR} \) given in [89]. According to the NASA Turbulence Modeling Resource, the considered version is the SA-noft2-QCR2000.

### 2.3. Vector form of the Reynolds-averaged Navier-Stokes system

The considered Reynolds-averaged Navier-Stokes (RANS) system is rewritten under a vector form:

\[ \mathbf{W}_t + \nabla \cdot \mathbf{F}^E = \nabla \cdot \mathbf{F}^V + \mathbf{F}^S, \]

where \( \mathbf{W} \) is the nondimensionalized conservative variables vector:

\[ \mathbf{W} = (\rho, \rho u, \rho v, \rho w, \rho E, \rho \bar{v})^T. \]

\( \mathbf{F}^E \) is the convective (Euler) fluxes vector:

\[
\mathbf{F}^E(\mathbf{W}) = \left( \mathbf{F}^E_1(\mathbf{W}), \mathbf{F}^E_2(\mathbf{W}), \mathbf{F}^E_3(\mathbf{W}) \right) = (\rho \mathbf{u}, \rho u \mathbf{u} + \rho \mathbf{e}_1, \rho v \mathbf{u} + \rho \mathbf{e}_2, \rho w \mathbf{u} + \rho \mathbf{e}_3, \mathbf{u}(\rho E + p), \rho \bar{v} \mathbf{u})^T,
\]

with \( (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \) the canonical basis. \( \mathbf{F}^V \) is the viscous fluxes vector:

\[
\mathbf{F}^V(\mathbf{W}) = \left( \mathbf{F}^V_1(\mathbf{W}), \mathbf{F}^V_2(\mathbf{W}), \mathbf{F}^V_3(\mathbf{W}) \right) = (0, \mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \mathbf{T} \cdot \mathbf{u} + \lambda \nabla T, \frac{\rho}{\sigma} (\nu + \bar{v}) \nabla \bar{v})^T,
\]

where the stress tensor \( \mathbf{T} = (\mu + \mu_t) \left[ (\nabla \otimes \mathbf{u}) + (\nabla \otimes \mathbf{u})^T - \frac{2}{3} \nabla \| \mathbf{u} \|^2 \right] \) has for components:

\[ \mathbf{T}_{11} = (\mu + \mu_t) \frac{2}{3} (2u_x - v_y - w_z), \quad \mathbf{T}_{12} = (\mu + \mu_t) (u_y + v_x), \quad \mathbf{T}_{13} = (\mu + \mu_t) (u_z + w_x), \quad \ldots \]

or its modified version given by the Spalart-Allmaras QCR model, see Relation (5). \( \mathbf{F}^S(\mathbf{W}) \) is the source terms fluxes, i.e., the diffusion, production and destruction terms from the Spalart-Allmaras turbulence model:

\[ \mathbf{F}^S(\mathbf{W}) = \left( 0, 0, 0, 0, \frac{c_{b2} \rho}{\sigma} \| \nabla \bar{v} \|^2 + \rho c_{b1} \hat{S} \bar{v} - c_{w1} f_w \rho \left( \frac{\bar{v}}{\bar{d}} \right)^2 \right)^T. \]

### 2.4. Aerodynamic cost functionals

In this work, we will analyze the drag, lift and moment cost functions to evaluate the accuracy and the mesh convergence of the mesh adaptation process. These coefficients will also be used as functional of interest for the goal-oriented mesh adaptation and will be differentiated in Section 5.1 to define the right-hand side of the adjoint state system. The computation of these coefficients requires three geometric parameters: \( S_{ref} \) the reference surface area, \( L_{ref} \) the reference surface length, and \( G_{ref} \) the reference surface barycenter. It also needs as reference state \( \mathbf{W}_\infty \) the state at the far-field boundary.
We first define the pressure and the skin friction coefficient vectors defined locally on the wetted-surface. The pressure coefficient vector is given by:

\[
C_p(x) = \frac{p(x) - p_{\infty}}{\frac{1}{2} \rho_{\infty} |u_{\infty}|^2} \mathbf{n}(x),
\]

where \(\frac{1}{2} \rho_{\infty} |u_{\infty}|^2\) the far-field dynamic pressure and \(\mathbf{n}\) is the outward normal to the surface. The skin friction coefficient vector is

\[
C_f(x) = \frac{\tau_w(x)}{\frac{1}{2} \rho_{\infty} |u_{\infty}|^2}
\]

where \(\tau_w(x) = (\mu + \mu_t) (\mathbf{t}(x) \cdot \mathbf{n}(x)) = \mu (\mathbf{t}(x) \cdot \mathbf{n}(x))\)

where \(\tau_w\) is the local wall shear stress with \(\mathbf{t}\) defined as in Equation (2). We recall that we have \(\mu_t = 0\) at walls. These local coefficient vectors are used to plot \(C_p\) and \(C_f\) profiles on wings.

Then, we can evaluate the aerodynamic coefficient vector by integration on the considered body \(S\):

\[
\begin{pmatrix}
C_x \\
C_y \\
C_z
\end{pmatrix} = \frac{1}{S_{ref}} \int_S \left( C_p(x) + C_f(x) \right) \, dx. \tag{9}
\]

But, this vector does not take into account the angle of attack of the considered flow. To this end, we consider the orientation of the body in space:

\[
\begin{pmatrix}
\text{Drag} \\
\text{Slip} \\
\text{Lift}
\end{pmatrix} = \text{Rot}(\theta, \alpha, \sigma) \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix},
\]

where \(\text{Rot}(\theta, \alpha, \sigma)\) is the rotation matrix defined by the angle of attack \(\alpha\), the angle of side slip \(\sigma\) and the angle of roll \(\theta\). For instance, if the span is along \(y\) and the symmetry plane is \(xz\), the rotation matrix is:

\[
\text{Rot}(\theta, \alpha, \sigma) = R_\theta^y R_\sigma^x R_\alpha^z
\]

with

\[
R_\theta^y = \begin{pmatrix}
1 & 0 & 0 \\
0 & \cos \theta & \sin \theta \\
0 & -\sin \theta & \cos \theta
\end{pmatrix}, \quad R_\sigma^x = \begin{pmatrix}
\cos \alpha & 0 & -\sin \alpha \\
0 & 1 & 0 \\
\sin \alpha & 0 & \cos \alpha
\end{pmatrix} \quad \text{and} \quad R_\alpha^z = \begin{pmatrix}
\cos \sigma & \sin \sigma & 0 \\
-\sin \sigma & \cos \sigma & 0 \\
0 & 0 & 1
\end{pmatrix}.
\]

Similarly, the moment coefficient vector is given by:

\[
\begin{pmatrix}
C_l \\
C_m \\
C_n
\end{pmatrix} = \frac{1}{S_{ref} L_{ref}} \int_S \left( \mathbf{g} \times (C_p(x) + C_f(x)) \right) \, dx,
\]

where \(\mathbf{g}\) are the coordinates of the body gravity center \(G_{ref}\). Then, the aerodynamic moment coefficient vector is obtained considering the orientation of the body in space:

\[
\begin{pmatrix}
\text{Roll} \\
\text{Pitch} \\
\text{Yaw}
\end{pmatrix} = \text{Rot}(\theta, \alpha, \sigma) \begin{pmatrix} C_l \\ C_m \\ C_n \end{pmatrix}.
\]

3. Anisotropic mesh adaptation algorithm with mesh-convergence analysis

Mesh adaptation is a non-linear problem where the couple formed by the mesh and the solution needs to be converged at the same time. Therefore an iterative process is required which is usually achieved by means of a mesh adaptation loop starting from an initial mesh \(\mathcal{H}_0\), an initial solution \(W_0\), an initial adjoint state \(W_0^r\) if goal-oriented mesh adaptation is considered, and a given mesh complexity \(C\) (the continuous counterpart of the mesh size, see Section 6).

At each step of the mesh adaptation loop, a metric tensor \(M_i\) is computed from the triple \((\mathcal{H}_i, W_i, W_i^r)\) and the given mesh complexity \(C\), using the selected error estimate, see Section 6. Metric tensor field \(M_i\) contains information on sizes and directions of the elements of the adapted mesh we seek. This information is then used by the remeshing to generate a new adapted mesh \(\mathcal{H}_{i+1}\) [64]. Then \(W_i\) is interpolated on \(\mathcal{H}_{i+1}\) to obtain \((W_0^0)_{i+1}\) which is then used as a restart solution for the next flow solution of the mesh adaptation loop [3]. In the case of goal-oriented mesh adaptation, the adjoint state \(W_i^r\) can be also interpolated on the new mesh \(\mathcal{H}_{i+1}\) to obtain \((W_r^0)_{i+1}\) which is used as a restart for the next adjoint solution. In practice, as we are using local
CFL (see Section 4.2.7) and as we restart from a configuration close to the previous step, we also interpolate the local CFL. Like this we restart the computation at high CFL and we let the automatic CFL law (section 4.2.7) manage the future evolution of the local CFL. Restarts are important to not waste time in the adaptive process and reuse at maximum the previous work done. This iterative process is depicted by the step 1 while loop in Algorithm 1.

The convergence criteria of step 1(f) is up to the expectations of the user, it specifies when the couple mesh-solution is considered as converge for the current complexity in the process. In this work, for aeronautic applications, we consider that the solution is converged at the given complexity if the lift coefficient, the pressure component of the drag and viscous component of the drag are not varying by a given percentage \( \epsilon \) for three consecutive iterations. Usually, we choose \( \epsilon \) between 0.001 (i.e., 0.1%) and 0.01 (i.e., 1%).

Algorithm 1: General mesh adaptation algorithm with mesh-convergence analysis

| Initial mesh \( \mathcal{H}_0 \), solution \( W^0 \), adjoint \( W^\star_0 \), and complexity \( C^0 \) |
| --- Outer loop to perform the convergence study |
| while \( C^j \leq C^{\text{max}} \) do |
| --- Inner loop to converge the mesh adaptation at fixed complexity |
| 1. while \( i \leq n_{\text{adap}} \) do |
| (a) Compute optimal metric for the considered error estimate and complexity \( \Rightarrow \mathcal{M}_i \) |
| (b) Generate new adapted mesh \( \Rightarrow \mathcal{H}_i \) |
| (c) Interpolate primal and adjoint states on the new mesh \( \Rightarrow (W^0)_i \) and \( (W^\star)_i \) |
| (d) Compute primal state \( \Rightarrow W_i \) |
| (e) Compute adjoint state \( \Rightarrow W^\star_i \) |
| (f) if (convergence check) then |
| \( i = n_{\text{adap}} + 1 \) |
| else |
| \( i = i + 1 \) |
| fi |
| done |
| 2. \( \mathcal{H}_0^{i+1} = \mathcal{H}_{n_{\text{adap}}}; \ W_0^{i+1} = W_{n_{\text{adap}}}; \ (W^\star)_0^{i+1} = (W^\star)_{n_{\text{adap}}}; \ C^{i+1} = \alpha \cdot C^j \) |
| done |

In the context of a mesh convergence analysis this adaptation loop has to be repeated for several increasing mesh complexities \( [C^j] \). An efficient strategy consists in converging the couple mesh/solution for a given complexity and reuse the final mesh, solution and adjoint state (and local CFL field) to initialize the next computations at an increased mesh complexity. Such a process enables a multiscale resolution of the flow by solving large scale features on coarse adapted meshes (at the smallest complexities) and the fine scale features of the flow on fine adapted meshes (at the largest complexities). This acts like a "multigrid effect" and enables faster convergence on fine adapted meshes. This process is represented by the outer while loop in Algorithm 1. At each outer loop iteration, the complexity is increase by a factor \( \alpha \). In this work, we have set \( \alpha = 2 \) to multiply the mesh size by a factor 2 when increasing the complexity. We have found that it is very advantageous to converge on the smallest complexities because of lot of work is done in converging the solution and these iterations are inexpensive in comparison to the largest complexities.

4. RANS flow solver

Reynolds-Averaged Navier-Stokes (RANS) numerical simulations are considered with the Spalart-Allmaras turbulence model. Wolf is a vertex-centered (flow variables are stored at vertices of the mesh) mixed Finite Volume - Finite Element Navier-Stokes solver on unstructured meshes composed of triangles in 2D and tetrahedra in 3D [6, 26, 27, 70]. The time integration considers an implicit temporal discretization. At each time step, the linear system of equations is approximately solved using a Symmetric Gauss-Seidel (SGS) implicit solver, and local time stepping and local CFL to accelerate the convergence to steady state. We observed that it is very important to exactly differentiate all the terms as it greatly improves the flow solver convergence. For the turbulence model, the Spalart-Allmaras is loosely-coupled to the mean-flow equations, where the mean-flow and turbulence model equations are relaxed in an alternating sequence.
4.1. Spatial discretization

The spatial discretization of the fluid Equations (1) and (3) is based on an hybrid Finite Volume - Finite Element formulation on unstructured meshes. The convective terms are solved by the Finite Volume method on the dual mesh composed of median cells. It uses a HLLC approximate Riemann solver [14] to compute the fluxes at the cell interfaces. Second order space accuracy is achieved through a piecewise linear interpolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure [47] which uses a particular edge-based formulation with upwind elements. A specific low dissipation scheme is considered using combination of centered (edge gradient) and upwind gradients (element gradient). A dedicated slope limiter is employed to damp or eliminate spurious oscillations that may occur in the vicinity of discontinuities. The viscous terms are solved by the $P^1$ Galerkin Finite Element Method (FEM) which provides second order accuracy.

Let $\mathcal{H}$ be a mesh of domain $\Omega$, 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 $\Omega_h$ (see Figure 2) 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, the median cells are considered.

In 2D, the median cell $C_i$ associated with vertex $P_i$ is built using the segments of medians of all triangles in the ball of $P_i$ (i.e., the set of triangles containing $P_i$), see Figure 2. 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) $G_{ij}, G_{jf}, G_{kf}$, 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 median cell $C_i$ associated with vertex $P_i$ is the union of all hexahedra of the tetrahedra surrounding $P_i$. The equations are integrated on each cell $C_i$ (using the Green formula):

$$\frac{1}{|C_i|} \int_{\partial C_i} \frac{dW_i}{dt} + \mathbf{F}_i = \mathbf{S}_i + \mathbf{Q}_i + \mathbf{\Gamma}_i,$$

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:

$$\mathbf{F}_i = \int_{\partial C_i} F(W_i) \cdot \mathbf{n}_i \, dy, \quad \mathbf{S}_i = \int_{\partial C_i} S(W_i) \cdot \mathbf{n}_i \, dy, \quad \mathbf{Q}_i = \int_{C_i} Q(W_i) \, d\Omega, \quad \mathbf{\Gamma}_i = \int_{\partial C_i} G(W_i) \, dy,$$

where $\mathbf{n}_i$ is the outer normal to the cell surface $\partial C_i$, and $F$, $S$ and $Q$ are respectively the convective, viscous and source terms flux functions as defined previously in Relations (6), (7) and (8). $G$ holds for the boundary flux function specified to impose the desired boundary condition.

![Figure 2: Median cell in 2D (left) and 3D (right).](image)

4.1.1. Convective fluxes discretization

The integration of the convective fluxes $\mathbf{F}$ of Equation (10) is done by decomposing the cell boundary $\partial C_i$ into many facets. These facets are grouped and associated to the edges joining vertex $P_i$ to its neighbors, as shown in 2D in Figure 3 where we have
a bi-segment for each edge. In 3D, we will have several facets for each edge. The facets associated to each edge are averaged to yield a single mean face per edge:

\[
\mathbf{F}_i = \int_{\partial C_i} F(W_i) \cdot \mathbf{n}_i \, dy \approx \sum_{P_f \in \mathcal{V}(P)} \int_{\partial C_{ij}} F_{i|\partial C_{ij}} \cdot \mathbf{n}_i \, dy
\]

where \( \mathcal{V}(P) \) is the set of all neighboring vertices linked by an edge to \( P \), and \( F_{i|\partial C_{ij}} \) represents the constant value of \( F(W) \) at interface \( \partial C_{ij} \). 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 \( \Phi_{ij} \) at the interface \( \partial C_{ij} \) is achieved by the resolution of a one-dimensional Riemann problem in the direction of the normal \( \mathbf{n}_j \) by means of an approximate Riemann solver. Here, we consider HLLC approximate Riemann solver proposed by Batten and al. \[14\]:

\[
\Phi_{ij}^{\text{convective}} = \Phi_{ij}^{\text{HLLC}}(W_i, W_j, \mathbf{n}_{ij}) = \int_{\partial C_{ij}} \mathbf{n}_i \, dy \quad \text{where} \quad \mathbf{n}_{ij} = \int_{\partial C_{ij}} \mathbf{n}_i \, dy.
\]

![Figure 3: Illustration of finite volume cell construction in 2D: two neighboring cells \( C_i \) and \( C_j \) and the upwind triangles \( K_i \) and \( K_j \) associated with edge \( P_iP_j \).](image)

**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 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 \( F(W_i^*) \) or \( F(W_j^*) \). Consequently, the HLLC flux is given by:

\[
\Phi_{ij}^{\text{HLLC}}(W_i, W_j, \mathbf{n}_{ij}) = \begin{cases} 
F(W_i) \cdot \mathbf{n}_{ij} & \text{if } S_i > 0 \\
F(W_i^*) \cdot \mathbf{n}_{ij} & \text{if } S_i \leq 0 < S_M \\
F(W_j^*) \cdot \mathbf{n}_{ij} & \text{if } S_M \leq 0 \leq S_j \\
F(W_j) \cdot \mathbf{n}_{ij} & \text{if } S_j < 0
\end{cases}
\]

\( W_i^* \) and \( W_j^* \) are evaluated as follows. We denote by \( \eta = \mathbf{u} \cdot \mathbf{n} \). Assuming that \( \eta_i^* = \eta_j^* = \eta_k^* = S_M \), the following evaluations are proposed \[14\] (the subscripts \( i \) and \( j \) are omitted for clarity):

\[
W^* = \frac{1}{S - S_M} \begin{pmatrix} 
\rho (S - \eta) \\
\rho E(S - \eta) + p^* S_M - p\eta
\end{pmatrix}
\]

where

\[
p^* = \rho (S - \eta)(S_M - \eta) + p.
\]

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

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

and the acoustic wave speeds based on the Roe’s average are:

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

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. From our experience, the approximate HLLC Riemann solver has proved to be a lot more robust that the approximate Roe Riemann solver.
**Second order accurate version.** The MUSCL type reconstruction method has been designed to increase the order of accuracy of the scheme [47]. The idea is to use extrapolated values \( W_{ij} \) and \( W_{ji} \) instead of \( W_i \) and \( W_j \) at the interface \( \partial C_{ij} \) 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:

\[
\Phi_{ij}^{\text{convective}} = \Phi_{ij}^{\text{HLLC}}(W_{ij}, W_{ji}, n_{ij}),
\]

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

\[
W_{ij} = W_i + \frac{1}{2} (\nabla W)_j \cdot \overrightarrow{P_i P_j} \quad \text{and} \quad W_{ji} = W_j + \frac{1}{2} (\nabla W)_i \cdot \overrightarrow{P_j P_i}.
\]

In contrast to the original MUSCL approach and most of the vertex-centered finite volume scheme, the approximate “slopes” \((\nabla W)_{ij}\) and \((\nabla W)_{ji}\) are defined for each edge. In other words, we can qualify this numerical scheme as a multi-slope finite-volume scheme unlike classical single-slope finite-volume schemes, which results in a more accurate scheme because it uses more directional information. This scheme is obtained using a combination of centered and upwind gradients in order to build low dissipation second order numerical scheme [45] that are called \( \beta \)-scheme. Even lower dissipation schemes can be obtained using also nodal gradients [27]. The centered gradient, which is related to edge \( P_i P_j \), is implicitly\(^3\) defined along edge \( P_i P_j \) by the relation:

\[
(\nabla W)_{ij}^{C} \cdot \overrightarrow{P_i P_j} = W_j - W_i \quad \text{and} \quad (\nabla W)_{ji}^{C} \cdot \overrightarrow{P_j P_i} = W_i - W_j.
\]

Upwind and downwind gradients, which are also related to edge \( P_i P_j \), are computed according to the definition of upwind and downwind triangles/tetrahedra of edge \( P_i P_j \). These triangles/tetrahedra are respectively denoted \( K_{ij} \) and \( K_{ji} \). \( K_{ij} \) (resp. \( K_{ji} \)) is the unique triangle/tetrahedron of the ball of \( P_i \) (resp. \( P_j \)) the opposite face of which is crossed by the line defined by the edge \( P_i P_j \), see Figures 3 and 4.

The upwind and downwind gradients are then the \( P^1 \)-Galerkin gradient of the considered elements. More precisely, we denote by \( \varphi_k \in V_h \) the basis function of vertex \( P_k \) where \( V_h \) is the approximation space associated with the \( P^1 \) Lagrange finite element. The \( P^1 \)-Galerkin gradient of triangle/tetrahedron \( K \) is given by

\[
(\nabla W)|_K = \sum_{P_k \in K} W_k \nabla \varphi_k|_K.
\]

This gradient at the element is constant. Upwind and downwind gradients are then defined for vertices \( P_i \) and \( P_j \) as:

\[
(\nabla W)|_{ij}^C = (\nabla W)|_{K_{ij}} \quad \text{and} \quad (\nabla W)|_{ji}^C = (\nabla W)|_{K_{ji}}.
\]

Parametrized gradients are built by introducing the \( \beta \)-scheme:

\[
(\nabla W)^{\beta}_{ij} \cdot \overrightarrow{P_i P_j} = (1 - \beta)(\nabla W)^{C}_{ij} \cdot \overrightarrow{P_i P_j} + \beta(\nabla W)^{C}_{ji} \cdot \overrightarrow{P_j P_i},
\]

\[
(\nabla W)^{\beta}_{ji} \cdot \overrightarrow{P_j P_i} = (1 - \beta)(\nabla W)^{C}_{ji} \cdot \overrightarrow{P_j P_i} + \beta(\nabla W)^{C}_{ij} \cdot \overrightarrow{P_i P_j},
\]

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 \( \beta = 1 \).

\(^3\)In the sense that we never evaluate \( (\nabla W)^{C}_{ij} \) and we know it only through \( W_j - W_i \) in the direction \( \overrightarrow{P_i P_j} \).
Fourth-order numerical dissipation: V4-scheme. The most accurate $\beta$-scheme is obtained for $\beta = 1/3$ [45, 27]. 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:

\[
(W^{\nabla}V^4)_{ij} \cdot \overrightarrow{P_iP_j} = \frac{2}{3} (W^{\nabla}V^4)_{ij} \cdot \overrightarrow{P_iP_j} + \frac{1}{3} (W^{\nabla}V^4)_{ij} \cdot \overrightarrow{P_iP_j}
\]

A sixth-order numerical dissipation scheme is given in [6, 27].

Dealing with edges composed of boundary vertices. For some of the edges having one or two vertices on the boundary, it can happen that there is no upwind element. In that case, the upwind gradient and the V4 gradient are undefined, thus an approximate upwind gradient should be found. Here, we choose to approximate the upwind gradient using the centered gradient and the nodal gradient of the considered vertex. Other methods can be designed.

The nodal gradient is computed using the Clément’s $L^2$-projection local operator [23]. For a vertex $P_i$, we denote by $S_i$ the stencil of basis function $\varphi_i$, i.e., $S_i = \text{supp} \varphi_i$, which is geometrically the ball of elements of $P_i$ in the $P_i$ case. The idea is to find, in a $L^2$-norm sense, the best constant gradient on $S_i$ approximating the piecewise constant field by elements (i.e., the constant gradients at the elements). Following [23], we obtain the following gradient reconstruction at vertex $P_i$:

\[
(W^{\nabla}V^4)_{P_i} = \frac{\sum_{K_j \in P_i} |K_j| (W^{\nabla}V^4)_{K_j}}{\sum_{K_j \in P_i} |K_j|} = \frac{\sum_{K_j \in P_i} |K_j| (W^{\nabla}V^4)_{K_j}}{\sum_{K_j \in P_i} (d+1)|C_j| (W^{\nabla}V^4)_{K_j}},
\]

where $(W^{\nabla}V^4)_{K_j}$ is $P_i$-Galarkin gradient of element $K_j$ given by Relation (11), and $|K_j|$ and $|S_i|$ denote the area/volume of element $K_j$ and stencil $S_i$, respectively. For finite volume cells build using the rules of median, we have $|S_i| = (d+1)|C_i|$. We observe that the best nodal gradient in a $L^2$-norm sense is the area-weighted (volume-weighted) average of the elements gradients of the vertex ball. The approximate upwind gradient is then defined by

\[
(W^{\nabla}V^4)_{ij} \approx 2 (W^{\nabla}V^4)_{P_i} - (W^{\nabla}V^4)_{ij}.
\]

Limiter function. MUSCL schemes are not monotone and can be a source of spurious oscillations especially in the vicinity of discontinuities [26]. 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 [38] 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 $(W^{\nabla}V^4)_{ij}$. The choice of the limiting function is crucial as it directly affects the numerical dissipation of the scheme and the convergence of the simulation.

$\beta$-scheme requires specific limiter functions with three entries because we have at hand three different gradient: the centered, the upwind and the low dissipation gradients. Piperno et al. have extended the Van Albada limiter to $\beta$-scheme [82, 21]. The superbbee limiter has been extended by Koren-Dervieux in [45, 26]. In this work, we consider a new limiter which is as smooth as the Piperno limiter and as low dissipative as the Koren-Dervieux limiter which is expressed in a factorized form:

\[
(W^{\nabla}V^4)_{ij} = \frac{\nabla W^C}{\nabla W^U} \psi \left( \frac{\nabla W^C}{\nabla W^U} \right), \quad \text{where} \quad R = \frac{\nabla W^C}{\nabla W^U},
\]

\[
\psi \left( \frac{\nabla W^C}{\nabla W^U} \right) = \begin{cases}
\frac{3}{R} - \frac{6}{2} + 19 & \text{if } R < \frac{1}{2}, \\
\frac{1}{2} - \frac{3}{2} + \frac{6}{1} + 11 & \text{if } \frac{1}{2} \leq R \leq 2, \\
1 + \left( \frac{3}{2} \frac{1}{R-1} + \frac{1}{R-1} \right)^3 & \text{if } R \geq 2
\end{cases}
\]

which can be also re-written only as a function of $\nabla W^C$ and $\nabla W^U$:

\[
(W^{\nabla}V^4)_{ij} = \frac{\nabla W^U}{\frac{1}{3} + \frac{2}{3} R} \psi \left( \frac{\nabla W^U}{\psi} \right) = \frac{\nabla W^U}{\psi} \psi \left( \frac{\nabla W^U}{\psi} \right).
\]

The proof of this new limiter and an analysis of its efficiency are given in Appendix A. The analysis emphasizes clearly the benefits of using low dissipation schemes and low dissipation limiters notably for their early capturing properties.
Discretization of the Spalart-Allmaras convection term. The convective term of Equation (4) can be discretized using a classical linear advection of the turbulent variable \( \rho \tilde{v} \). But, here we consider a nonlinear approach that has been proposed in [46]:

\[
\Phi_p^{\text{convective}}(W_{ij}, W_{ji}, \mathbf{n}_j) = \Phi_p^{\text{convective}}(W_{ij}, W_{ji}, \mathbf{n}_j) \begin{cases} \\
\vec{v}_i & \text{if } \Phi_p^{\text{convective}}(W_{ij}, W_{ji}, \mathbf{n}_j) > 0 \\
\vec{v}_j & \text{otherwise}
\end{cases}
\]

where \( \Phi_p^{\text{convective}} \) of the density variable is computed with the HLLC approximate Riemann solver described above. It has been proven that this scheme preserves the maximum principle for the convected turbulent variable.

4.1.2. Viscous terms discretization

The viscous terms are discretized using the \( P^1 \) Finite Element method (FEM) which is second order accurate. We need to evaluate

\[
\mathbf{S}_i = \sum_{P_i \in \mathcal{V}(P)} \int_{\partial C_i} S_i(W) \cdot \mathbf{n} \, d\Omega,
\]

where \( \partial C_{ij} \) is the common interface between cells \( C_i \) and \( C_j \) build using the rules of median. Let \( \varphi_i \) be the \( P^1 \) Finite Element basis function associated with vertex \( P_i \), we have:

\[
\int_K \nabla \varphi_i \, d\Omega = - \int_{\partial C \cap K} \mathbf{n} \, d\Omega,
\]

which states the equivalence between the Finite Element and the Finite Volume for linear solutions. As the solution is represented on the \( P^1 \) basis, \( S_i(W) \) (which comes from a gradient) is assumed constant by parts on each element \( K \), for instance (in 3D):

\[
\mathcal{T}_{xy}|_K = (\mu_l|_K + \mu_t|_K) \left( \frac{\partial u}{\partial y}|_K + \frac{\partial v}{\partial x}|_K \right) = \left( \sum_{P \in K} \frac{\mu_j + \mu_k}{4} \right) \left( \sum_{P \in K} \left( \frac{\partial \varphi_j}{\partial y} + \frac{\partial \varphi_k}{\partial x} \right) \right),
\]

where \( \mu|_K \) (resp. \( \mu_t|_K \)) is the mean value of \( \mu \) (resp. \( \mu_t \)) on the element. Then, we obtain

\[
\mathbf{S}_i = \sum_{P_i \in \mathcal{V}(P)} \int_{\partial C_{ij}} S_i(W) \cdot \mathbf{n} \, d\Omega = \sum_{K, P_i \in K} S_i(W)|_K \cdot \int_{\partial C \cap K} \mathbf{n} \, d\Omega = - \sum_{K, P_i \in K} \int_K S_i(W)|_K \cdot \nabla \varphi_i \, d\Omega.
\]

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

\[
\int_K \nabla \varphi_i \cdot \nabla \varphi_j \, d\Omega = |K| \nabla \varphi|_K \cdot \nabla \varphi|_K.
\]

In this expression, \( \nabla \varphi|_K \) is the constant gradient of basis function \( \varphi_i \) associated with vertex \( P_i \) in element \( K \). This discretization is justified because the characteristic time associated with the diffusive terms is large as compared to the characteristic time associated with the hyperbolic (convective) terms.

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 assemble them. Given an element \( K = (P_i, P_j, P_k, P_l) \) we have the partial flux associated with each vertex

\[
\Phi_i^{\text{viscous}}|_K(W_i, W_j, W_k, W_l) = \int_{\partial C \cap K} S_i(W)|_K \cdot \mathbf{n} \, d\Omega = - \int_K S_i(W)|_K \cdot \nabla \varphi_i \, d\Omega,
\]

computed as mentioned previously.

If the QCR version of the Spalart-Allmaras turbulence model, then the stress tensor \( \mathcal{T} \) should be modified accordingly, see Equation (5).

Discretization of the Spalart-Allmaras dissipation term. The Spalart-Allmaras dissipation term of Equation (4) is also discretized with the FEM:

\[
\Phi_i^{\text{dissipation}}|_K(W_i, W_j, W_k, W_l) = |K| \frac{1}{\sigma_p} \left( |\vec{v}|_K + |\vec{v}_{ij}|_K \right) \nabla \vec{v}|_K \cdot \nabla \varphi_i|_K.
\]

4.1.3. Source terms discretization

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

\[
\Phi_i^{\text{source}} = \int_{C_i} Q_i(W_i) \, d\Omega = |C_i| Q_i(W_i),
\]

where \( |C_i| \) is the volume of cell associated with vertex \( P_i \) and the source term \( Q_i(W_i) \) is given by the last line of the source term vector of Relation (8).
4.1.4. Boundary conditions discretization

Boundary conditions are imposed boundary element-wise, e.g. a boundary flux is computed for each boundary edge/face and assembled to each of its vertices. Consequently, even if a vertex-center scheme is considered, consistent boundary conditions are obtained because the type of conditions can be multiple around a vertex. For aeronautic computations, three different boundary conditions are involved: no-slip boundary condition for walls, symmetry plane (or slip) boundary condition and far-field boundary condition.

No-slip boundary condition. For no-slip boundary conditions, a null velocity \( \mathbf{u} = 0 \) and a turbulent variable equal to \( \tilde{\nu} = 0 \) are strongly enforced at each iteration. Consistently, we impose \( \Phi_{\nu} = 0, \Phi_{\nu} = 0, \) and \( \Phi_{\chi} = 0 \) at the boundary. The energy flux is fixed according to the desired temperature behavior: for an adiabatic wall it is null and for an isothermal wall the energy variable is enforced similarly to the velocity.

Symmetry plane boundary condition. Symmetry plane conditions are the same as slip wall boundary conditions. For this boundary condition, we impose weakly:

\[
\mathbf{u} \cdot \mathbf{n} = 0. \tag{15}
\]

To this end, we compute the flux between the inner state \( W_i \) and its mirror state \( \bar{W}_i \) defined by:

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

and using the HLLC approximate Riemann solver:

\[
\Phi_i^{\text{Sym}}(W_i) = \sum_{f_j \in \partial P_i} \mathbf{q}_{HLLC}(W_i, \bar{W}_i, n_{f_j}) = \sum_{f_j \in \partial P_i} (0, p_i^*, \mathbf{n}_{f_j}, 0)^T,
\]

where \( f_j \ni P_i \) is the list of boundary faces containing \( P_i \), and \( p_i^* \) is defined according to the considered Riemann solver [6]. If Condition (15) is verified then \( W_i = \bar{W}_i \) and \( \Phi_i^{HLLC}(W_i, \bar{W}_i) = F(W_i) \). Moreover, as \( W_i \) verifies Relation (15), \( F(W_i) \) simplifies to:

\[
\Phi_i^{\text{Sym}}(W_i) = \sum_{f_j \ni P_i} (0, p_i, \mathbf{n}_{f_j}, 0)^T.
\]

Therefore, if the desired condition is satisfied, then the boundary flux reduced to its well known commonly used form. For the turbulent variable we have \( \Phi_{\nu}^{\text{Sym}} = 0 \) at slip/symmetry boundary.

Far-field boundary condition. Far field boundary conditions are imposed weakly. The far-field state \( W_\infty \) is defined by the flow conditions of the problem at hand. The flux at the far-field boundary is computed using the HLLC approximate Riemann solver:

\[
\Phi_i^{\text{FarField}}(W_i) = \sum_{f_j \ni P_i} \mathbf{q}_{HLLC}(W_i, W_\infty, \mathbf{n}_{f_j}).
\]

For the turbulent variable, we use Relation (13) to compute the boundary flux where \( \Phi_{\nu}^{\text{convec}} \) is obtained from the above flux and \( \tilde{\nu} \) is replace by \( \bar{\nu}_\infty \). We choose \( \bar{\nu}_\infty = 3 \nu_\infty \) as suggested in [90].

4.2. Implicit time integration

There are two iterative strategies to approach a solution of a nonlinear system: i) the Newton method and ii) pseudo-transient continuation method. The Newton method does not use pseudo-time terms (i.e., the mass matrix), which is equivalent to using an infinite CFL number. It makes it very difficult to use for ill-conditioned linear system. The pseudo-transient continuation method using pseudo-time marching is assumed to be more robust, because it increases the diagonal dominance of the matrix. A nice discussion about these methods is given in [75]. Because we deal with complex 3D geometries and highly anisotropic adapted mesh, the convergence to steady state is achieved using the pseudo-transient continuation method with an implicit time integration. The implicit temporal discretization considers the backward Euler time-integration scheme as high-order accuracy in time is not required for steady flows. At each time step, the linear system of equations is approximately solved using a Symmetric Gauss-Seidel (SGS) implicit solver and local time stepping to accelerate the convergence to steady state. A Newton method based on the SGS relaxation is very attractive because it can be written with an edge-based data structure which can be efficiently parallelized. From our experience, we have made the following - crucial - choices to solve the compressible Navier-Stokes equations:
• the SGS relaxation iterates until the residual of the linear system is reduced by one or two orders of magnitude
• the choice of the renumbering also impacts strongly the convergence of the non-linear system. While Hilbert-type (space filling curve) renumbering is very efficient for cache misses and memory contention [5, 87], Breadth-first search renumbering proves to be more effective for the convergence of the implicit method and the overall efficiency
• we found very advantageous to exactly differentiate the convective terms, the viscous terms, the source terms and the boundary conditions. Exact differentiation greatly improves the flow solver convergence. The only (unfortunate) approximation is the MUSCL which is not differentiated because it changes the matrix pattern
• to achieve high efficiency, automation and robustness in the resolution of the non-linear system of algebraic equations to steady-state, it is mandatory to have a clever strategy to specify the CFL. In Rõlfl, the CFL evolution depends on the non-linear convergence and on the evolution of the solution at each time step. Like this there is no adhoc CFL law prescribed by the user, the evolution of the CFL is automatically managed by the flow solver. A similar strategy has been developed in USB3D [75] (NASA). Furthermore, we have found very advantageous to use a local CFL for each vertex.

4.2.1. Implicit system

Once the equations have been discretized in space, a set of ordinary differential equations in time is obtained. For an implicit time integration, the semi-discretized RANS system becomes:

\[
\frac{|C_i|}{h^2} \delta W_i = -F_i^{n+1} + S_i^{n+1} + Q_i^{n+1} + \Gamma_i^{n+1},
\]

where \( \delta W_i = W_i^{n+1} - W_i^n \), \(|C_i|\) is the area/volume of the finite volume cell associated with vertex \( P_i \) and \( \delta t^n \) is the local time step at iteration \( n \) for vertex \( P_i \) given by:

\[
\delta t = \text{CFL} \frac{h^2}{h(c + ||u||) + 2 \frac{C_p}{\rho} \left( \frac{\mu}{\text{Pr}} + \frac{\mu_t}{\text{Pr}_t} \right)} = \text{CFL} \frac{h^2}{h(c + ||u||) + 2 \lambda + \lambda_t}
\]

where \( C_p \) is the specific heat at constant pressure, \( \text{Pr} \) and \( \text{Pr}_t \) are the laminar and turbulent Prandtl numbers, \( \mu \) and \( \mu_t \) are the laminar and turbulent dynamic viscosities and, \( \lambda \) and \( \lambda_t \) are the laminar and turbulent dynamic conductivity. In the above relation, \( h \) is the representative mesh size for vertex \( P_i \) which is taken as the smallest height of all the elements surrounding \( P_i \).

The implicit system is obtained by linearization of the semi-discretized RANS system with respect to the conservative variables \( W \). However, the computation of the Jacobian of the second order convective flux \( \frac{\partial F_i^n}{\partial W_j} \) introduces an extra-difficulty as it involves the second order ball of vertex \( P_i \), as seen above, while the linearization of the other terms only involve the first order ball. This will enlarge drastically the pattern of the matrix which will not be edge-based anymore. It will lead to a large memory and CPU overhead. For all these reasons, it is approximated by the Jacobian of the first order convective flux \( \frac{\partial F_i^n}{\partial W_j} \) which only depends on the first order ball (the other Jacobian terms are kept unchanged). After linearization of the RHS, it becomes:

\[
\left( \left[ \frac{|C_i|}{h^2} I_d + \frac{\partial F_i^n}{\partial W_i} - \frac{\partial S_i^n}{\partial W_i} - \frac{\partial Q_i^n}{\partial W_i} - \frac{\partial R_i^n}{\partial W_i} \right] \delta W_i + \sum_{j \in \mathcal{V}(i)} \left( \frac{\partial F_i^n}{\partial W_j} - \frac{\partial S_i^n}{\partial W_j} - \frac{\partial Q_i^n}{\partial W_j} - \frac{\partial R_i^n}{\partial W_j} \right) \delta W_j \right) = -F_i^n + S_i^n + Q_i^n + \Gamma_i^n.
\]

where \( P_j \in \mathcal{V}(i) \) is the set of vertices connected to vertex \( P_i \) by an edge. As the RHS still consider the second order convective flux, this acts as an approximated Jacobian but it is independent of the residual and thus does not affect the spatial order of the scheme. 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. This linearized system can be written in vector form:

\[
A^n \delta W^n = \mathbf{R}^n \quad \text{with} \quad A^n = \frac{|C_i|}{h^2} I_d - \frac{\partial R_i^n}{\partial W} \quad \text{and} \quad \delta W^n = W^{n+1} - W^n,
\]

where \( \mathbf{R}^n = -F^n + S^n + Q^n + \Gamma^n \) and \( \mathbf{R}^n = -\mathbf{F}^n + \mathbf{S}^n + \mathbf{Q}^n + \mathbf{I}^n \). We will decompose the matrix \( A \) into three parts: the diagonal part \( \mathbf{D} \), the lower part \( \mathbf{L} \) and the upper part \( \mathbf{D} \) such that: \( A = \mathbf{L} + \mathbf{D} + \mathbf{U} \).

\[4\] In 3D, instead of an average of twenty entries per matrix line, it will increase to an average of a hundred.
When solving the RANS equations, the following choice has been made. In the linear system, the Spalart-Allmaras is loosely-coupled to the mean-flow equations, where the mean-flow and turbulence model equations are relaxed in an alternating sequence. In other words, at each solver iteration, we are solving independently two linear systems, one for the mean flow where each entry is a block of size $5 \times 5$ in 3D and one for the turbulence model where each entry is a single value when considering the Spalart-Allmaras model. Because of this choice, we have chosen to differentiate the turbulent system with respect to $\tilde{v}$ and not with respect to $\tilde{\rho}$ according to Equation (4). Thus the turbulent linear system is computing $\delta \rho'$ and the mass matrix diagonal term is multiplied by $\rho'$ (in other word we consider $\rho$ constant in the turbulent system).

In the following, we quickly describe the strategy to compute the exact Jacobians.

4.2.2. Convective fluxes Jacobians

We recall that

$$F^{n+1}_i = \sum_{P \in V(i)} \Phi^{HLC}_{ij}(W_{ij}^{n+1}, W_{ji}^{n+1}, n_{ij}),$$

where $W_{ij}^{n+1}$ and $W_{ji}^{n+1}$ are obtained, for the second-order scheme, through MUSCL extrapolation and gradients limitation from the stencil of neighbors. This will enlarge the edge-based pattern of the matrix. To avoid it, the linearization of the second-order convective fluxes is approximated by the exact linearization of the first-order convective fluxes:

$$\tilde{F}^{n+1}_i = \sum_{P \in V(i)} \Phi^{HLC}_{ij}(W_{ij}^{n+1}, W_{ji}^{n+1}, n_{ij}).$$

The linearization of the convective fluxes reads:

$$\Phi^{HLC}_{ij}(W_{ij}^n, W_{ji}^n, n_{ij}) = \Phi^{HLC}_{ij}(W_i^n, W_j^n, n_{ij}) + \frac{\partial \Phi^{HLC}_{ij}(W_i^n, W_j^n, n_{ij})}{\partial W_i} \delta W_i + \frac{\partial \Phi^{HLC}_{ij}(W_i^n, W_j^n, n_{ij})}{\partial W_j} \delta W_j.$$ 

Assuming $i < j$, the first derivative term contributes to matrix diagonal $D(i,i)$ and the second derivative term contributes to matrix upper part $U(i,j)$. As $\Phi^{HLC}_{ij} = -\Phi^{HLC}_{ji}$, minus the second term contributes to matrix diagonal $D(j,j)$ and minus the first term contributes to matrix lower part $L(j,i)$. Some works suggest to used extrapolated values in these Jacobians to improve the Jacobian approximation. All the details on the differentiation of the HLLC approximate Riemann solver can be found in [15].

The linearization of the convective fluxes of the turbulent system is simply:

$$\frac{\partial \Phi^{\text{convective}}}{\partial \tilde{\rho}}(W_i^n, W_j^n, n_{ij}) = \begin{cases} \Phi^{\text{convective}}(W_i^n, W_j^n, n_{ij}) & \text{if } \Phi^{\text{convective}}(W_i^n, W_j^n, n_{ij}) > 0 \\ 0 & \text{otherwise} \end{cases}$$

and similarly for the other derivatives.

4.2.3. Viscous terms Jacobians

Let $K = (P_i, P_j, P_k, P_l)$ be a tetrahedron, we now linearize the viscous flux terms $S_{ij}$:

$$S_{ij}^{n+1} = -\sum_{K \in P_{ij}} \Phi^{\text{viscous,L}}_{i,j,K}(W_{ij}^{n+1}, W_{ji}^{n+1}, W_k^n, W_{ij}^n).$$

To simplify the notations we omit the four entries of $\Phi^{\text{viscous,L}}_{i,j,K}$, the linearization is then:

$$\Phi^{\text{viscous,L,+1}}_{i,j,K} = \Phi^{\text{viscous,L}}_{i,j,K} + \frac{\partial \Phi^{\text{viscous,L}}_{i,j,K}}{\partial W_i} \delta W_i + \frac{\partial \Phi^{\text{viscous,L}}_{i,j,K}}{\partial W_j} \delta W_j + \frac{\partial \Phi^{\text{viscous,L}}_{i,j,K}}{\partial W_k} \delta W_k + \frac{\partial \Phi^{\text{viscous,L}}_{i,j,K}}{\partial W_{ij}} \delta W_{ij}.$$ 

where the derivative terms contributes respectively to the matrix diagonal and the matrix extra-diagonal terms. These terms are obtained by deriving terms of Relation (7) that are discretized with respect to the $P_i$ Finite Element basis function $\varphi_i$ as described in Section 4.1.2. For instance, the term given by Relation (14) in Section 4.1.2 is derived as:

$$\frac{\partial T_{xy}}{\partial W_i} = (\mu_k + \mu_l)$$

$$\left( \begin{array}{cc} -\frac{1}{\rho_j} \left( \rho u_i \frac{\partial \varphi_j}{\partial y} + \rho v_j \frac{\partial \varphi_i}{\partial x} \right) \\ \frac{1}{\rho_j} \frac{\partial \varphi_j}{\partial y} \\ \frac{1}{\rho_j} \frac{\partial \varphi_i}{\partial x} \\ 0 \end{array} \right) = \begin{array}{c} \frac{\partial T_{xy}}{\partial W_j} \\ (\mu_k + \mu_l) \end{array}.$$
If the QCR version of the Spalart-Allmaras turbulence model, then the modified stress tensor should be differentiated.

For the Spalart-Allmaras dissipation term, we have:

\[
\frac{\partial \phi_{\text{diss}}}{\partial \psi_i} = |K| \rho_i \left( \frac{1}{\sigma} \frac{\partial}{\partial \psi_i} \nabla \psi \cdot \nabla \psi + (\psi + \bar{\psi}) \nabla \psi \cdot \nabla \psi \right)
\]

and

\[
\frac{\partial \phi_{\text{diss}}}{\partial \psi_j} = |K| \rho_j \left( \frac{1}{\sigma} \frac{\partial}{\partial \psi_j} \nabla \psi \cdot \nabla \psi + (\psi + \bar{\psi}) \nabla \psi \cdot \nabla \psi \right).
\]

### 4.2.4. Source terms Jacobians

For the Spalart-Allmaras equation, the source term is the sum of production (\(Q_{\text{prod}}\)), destruction (\(Q_{\text{dest}}\)) and diffusion (\(Q_{\text{diss}}\)) terms. For convergence efficiency, it is very important to fully differentiate all the terms and to avoid alterations of the differentiation like clippings to ensure the positivity of diagonal terms. The production and the destruction terms only contribute to the diagonal. As the diffusion term involves gradients, it contributes to the diagonal and extra-diagonal matrices.

For the full linearization of the production term, we have:

\[
\frac{\partial Q_{\text{prod}}}{\partial \psi_i} = c_{\text{w}} \rho_i \left( \bar{S} + \bar{v}_i \frac{\partial S}{\partial \psi_i} \right).
\]

Seeing that \(\frac{\partial \psi}{\partial \psi_i} = \frac{1}{\psi_i}\), we have:

\[
\frac{\partial \bar{S}}{\partial \psi_i} = \frac{f_{\psi_2}}{k^2 \psi_i^2} + \frac{\bar{v}_i}{\psi_i} \frac{\partial f_{\psi_2}}{\partial \psi_i} ; \quad \frac{\partial f_{\psi_2}}{\partial \psi_i} = \frac{-\frac{1}{\psi_i} + \chi^2 \frac{\partial f_{\psi_2}}{\partial \psi_i}}{(1 + \chi f_{\psi_2})^2} ; \quad \frac{\partial f_{\psi_2}}{\partial \psi_i} = \frac{3 \chi^2 c_{\psi_2}^2}{\psi_i (\chi^3 + c_{\psi_2})^2}.
\]

For the full linearization of the destruction term, we have:

\[
\frac{\partial Q_{\text{dest}}}{\partial \psi_i} = c_{\text{w}} \rho_i \left( 2 f_{\psi_2} \bar{v}_i + \frac{\partial f_{\psi_2}}{\partial \psi_i} \right) = \frac{c_{\text{w}} \rho_i \bar{v}_i}{\psi_i} \left( 2 f_{\psi_2} + \frac{\partial f_{\psi_2}}{\partial \psi_i} \right)
\]

We denote \(G_{\text{lin}} = \frac{1 + c_{\psi_2}^2}{g^2 + c_{\psi_2}^2}\) and we have:

\[
\frac{\partial G_{\text{lin}}}{\partial \psi_i} = \frac{6 g \frac{\partial \psi}{\partial \psi_i} (1 + c_{\psi_2}^2)}{(g^2 + c_{\psi_2}^2)^2} ; \quad \frac{\partial f_{\psi_2}}{\partial \psi_i} = c_{\text{lin}} \frac{\partial G_{\text{lin}}}{\partial \psi_i} \left( 1 - \frac{g^2}{g^2 + c_{\psi_2}^2} \right) ; \quad \frac{\partial g}{\partial \psi_i} = \left( 1 + c_{\psi_2} (6 g^2 - 1) \right) \frac{\partial \psi}{\partial \psi_i} ; \quad \frac{\partial \psi}{\partial \psi_i} = \bar{S} - \bar{v}_i \frac{\partial \bar{S}}{\partial \psi_i}.
\]

For the full linearization of the diffusion term, as it involves gradients, we get Jacobian contributions through the neighboring elements (vertices). Assuming the nodal gradient is obtained by \(L^2\) projection, this source term reads in 3D:

\[
Q_{\text{diss}}^i(W) = \frac{c_{\psi_2}}{\sigma} \rho_i \left( \nabla \psi_i \right)^2 = \frac{c_{\psi_2}}{\sigma} \rho_i \left( \sum_{K, \beta h_i} |K| \left( \sum_{l \in h_i} \bar{v}_i \nabla \psi_i \right) \right)^2,
\]

and it is differentiated as

\[
\frac{\partial Q_{\text{diss}}^i(W)}{\partial \psi_i} = \frac{c_{\psi_2}}{\sigma} \rho_i \left( 2 \nabla \psi_i \cdot \nabla \psi_i \right) \quad \text{and} \quad \frac{\partial Q_{\text{diss}}^i(W)}{\partial \psi_j} = \frac{c_{\psi_2}}{\sigma} \rho_i \left( 2 \nabla \psi_i \cdot \nabla \psi_j \right).
\]

leading to the following diagonal and extra-diagonal contributions on each element:

\[
\frac{\partial Q_{\text{diss}}^i}{\partial \psi_i} = 2 \frac{c_{\psi_2}}{\sigma} \rho_i \left( \nabla \psi_i \right)^2 \quad \text{and} \quad \frac{\partial Q_{\text{diss}}^i}{\partial \psi_j} = 2 \frac{c_{\psi_2}}{\sigma} \rho_i \left( \nabla \psi_j \right)^2.
\]

### 4.2.5. Boundary conditions Jacobian

Similarly to the other terms, the boundary conditions are exactly differentiated.
**No-slip boundary condition.** For no-slip boundary conditions, nothing is done for \( \rho \) and \( \rho E \) as there is no flux. Dirichlet (enforced variables) boundary conditions, such as the velocity in no-slip boundary conditions are enforced strongly 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

\[
\begin{bmatrix}
* & * & * & * & * \\
* & * & * & * & * \\
* & * & * & * & * \\
0 & \ldots & 0 & 1 & 0 \\
* & * & * & * & * \\
\end{bmatrix}
\begin{bmatrix}
\delta W_{m} = 0 \\
\delta W \\
\delta W \\
\delta W \\
\delta W \\
\end{bmatrix}
= \begin{bmatrix}
* \\
* \\
* \\
0 \\
* \\
\end{bmatrix}.
\]

**Slip boundary condition.** We have seen that for slip boundary conditions, the flux is computed between the inner state and its mirror state. Following the chain rule, the exact differentiation is:

\[
\frac{\partial \Phi_{\text{Sym}}}{\partial W_{i}}(W_{j}) = \sum_{j \neq p, k} \left( \frac{\partial \Phi_{\text{HLLC}}}{\partial W_{i}}(W_{j}, n_{j}) \frac{\partial W_{i}}{\partial W_{j}} \right) + \sum_{j \neq p} \left( \frac{\partial \Phi_{\text{HLLC}}}{\partial W_{i}}(W_{j}, n_{j}) \frac{\partial W_{i}}{\partial W_{j}} \right).
\]

where we have:

\[
\frac{\partial W_{i}}{\partial W_{j}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 - n_{e}^{2} & -n_{e} & -n_{e} & 0 \\
0 & -n_{e} & 1 - n_{e}^{2} & -n_{e} & 0 \\
0 & -n_{e} & -n_{e} & 1 - n_{e}^{2} & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Or we can directly use \( \frac{\partial p^{*}}{\partial W_{i}} \) or \( \frac{\partial p}{\partial W_{i}} \) depending on the considered formulation of the boundary condition.

**Far-field boundary condition.** The far-field boundary condition is easy to differentiate because it is given by the flux computed between the inner state and prescribed infinite state. We immediately get:

\[
\frac{\partial \Phi_{\text{farField}}}{\partial W_{i}}(W_{j}) = \sum_{j \neq p, k} \frac{\partial \Phi_{\text{HLLC}}}{\partial W_{i}}(W_{j}, W_{\infty}, n_{j}).
\]

### 4.2.6. Solving the linear system: SGS Relaxation

The linearized system obtained in the previous sections and given by Relation (17) is solved at each flow solver iteration. In practice, we ask the user to provide a maximal number of iterations \( k_{\text{max}} \) (usually 20) and a targeted order of magnitude by which the relative residual of the system must be decreased (usually 0.01). The iteration is stopped when this targeted relative residual or the maximal iteration is reached. To solve the linear system, the considered method is the SGS relaxation (Symmetric Gauss-Seidel) based on the Lower-Upper Symmetric Gauss-Seidel (LU-SGS) implicit solver initially introduced by Jameson [42] and fully developed by Sharov et al. [87]. The SGS relaxation is very attractive because it uses an edge-based data structure of the code and can be efficiently parallelized [5, 87].

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 \( \delta W^{m,k} \) so that \( A^{n}_{+} \delta W^{m,k} \rightarrow R^{n}. \) As the system \((L^{n} + D^{n} + U^{n}) \delta W^{m,k} = R^{n}\) 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:

\[
(D^{n} + L^{n}) \delta W^{m,k+1} + U^{n} \delta W^{m,k} = R^{n}.
\]

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

\[
(D^{n} + L^{n}) (\delta W^{m,k+1} + U^{n} \delta W^{m,k}) = (D^{n} + L^{n} + U^{n}) \delta W,
\]

\[
(D^{n} + L^{n}) (\delta W^{m,k+1} - \delta W) = -(U^{n}) (\delta W^{m,k} - \delta W),
\]

\[
(\delta W^{m,k+1} - \delta W) = -(D^{n} + L^{n})^{-1} (U^{n}) (\delta W^{m,k} - \delta W).
\]

The convergence of Equation (18) is thus directly related to the largest eigenvalue of \((D^{n} + L^{n})^{-1} (U^{n})\). The SGS is obtained with a two steps fixed point, alternatively fixing \( U^{n} \delta W^{m,k} \) as in Equation (18) and then \( L^{n} \delta W^{m,k} \) the same way.
We first zero the unknown: \( \delta W^0 = 0 \). Then, \( k_{\text{max}} \) sub-iterations are made using forward and backward sweeps:

\[
(D^n + L^n) \delta W^{n,k+1/2} = R^n - U^n \delta W^{n,k} \\
(D^n + U^n) \delta W^{n,k+1} = R^n - L^n \delta W^{n,k+1/2} .
\]

or rewritten point-wise with block matrix operations:

\[
\delta W^{n,k+1/2} = (D^n)^{-1}(R^n - \sum_{j \in U(i)} L^n_{ij} \delta W^{n,k+1/2} - \sum_{j \in L(i)} U^n_{ij} \delta W^{n,k}) \\
\delta W^{n,k+1} = (D^n)^{-1}(R^n - \sum_{j \in U(i)} U^n_{ij} \delta W^{n,k+1} - \sum_{j \in L(i)} L^n_{ij} \delta W^{n,k+1/2}) .
\]

where \( U(i) \) (resp. \( L(i) \)) is the set of vertices with an index lower (resp. upper) than \( i \). For one sub-iteration, the SGS method is equivalent to the LU-SGS method.

### 4.2.7. Automatic CFL Law

In most of the cases, steady flow simulations are started from a uniform flow field\(^5\). In that case, even if implicit scheme

or unconditional stable, we cannot start the pseudo-transient continuation method at very high CFL. Simulations are started at low CFL (around 1) and a CFL law is used to increase it gradually. Many CFL laws exist in the literature - linear growth, geometric growth, residual based, constant ramp, ... - but these adhoc laws generally require parameters that are difficult to establish optimally because they depend on the considered flow, the geometry and the size of the mesh. In other words, they depend too much on user’s data. But, they are mandatory to achieve fast convergence in solving non-linear equations. If the objective is robustness, then the parameters are chosen to have a gentle growth of the CFL which results in a loss of efficiency. On the other hand, if the objective is efficiency, then the parameters can be chosen so that it increases more quickly but the computation may not converge or even diverge, we are losing robustness.

To avoid these issues, in the strategy we propose, the CFL is automatically managed by the flow solver depending on the non-linear convergence of the flow solver and the change in the solution at each vertex.

In the literature, we found three interesting strategies to automatically control the CFL law.

Luke et al. proposed a new approach \([66]\) based on bounding the primitive variables, \( \rho \), \( p \) and \( T \), variations at each flow solver iteration. More precisely, at each iteration, an initial large time step is set to each vertex which is the maximal allowed time step by the user. Then, this time step is locally truncated such that the change in \( \rho \), \( p \) and \( T \) at each vertex are below a user given percentage \( \eta \). To this end, the change in primitive variables during a given interval of time has to be estimated because the local time step at each vertex has to be known before assembling the linear system as the local time step is used to compute the mass matrix. A way to accomplish this is to first solve an explicit time-integration step to describe a functional relationship between time and primitive variable.

This method is efficient as each vertex is progressing with its own time step and no CFL law is involved. But, that choice is made from an estimation before the linear system resolution using an explicit time integration thus the time step may not be the optimal one, and there is no time step control according to the behavior convergence or divergence of the method, i.e., the non-linear residual.

Another attractive approach has been proposed by Burgess and Glasby \([20]\) which couples under-relaxation coefficient and dynamic CFL. Here, the solution is analyzed at each step after solving the linear system and before updating the solution. First, the change in primitive variables at each vertex, \( \rho \) and \( p \), is controlled by a user given percentage \( \eta \) and defines a global under-relaxation coefficient \( \omega^f \) at each step of the process:

\[
\omega^f = \min_{\rho, p, T} (\omega^f) .
\]

This global coefficient is applied to the solution update:

\[
W^{n+1} = W^n + \omega^f \delta W .
\]

Then, the CFL value is updated depending on the global under-relaxation coefficient:

\[
CFL^{n+1} = \begin{cases} 
0.1 \text{CFL}^n & \text{if } \omega^f < 0.1 \\
\text{CFL}^n & \text{if } 0.1 \leq \omega^f < 1 \\
\alpha \text{CFL}^n + \beta & \text{if } \omega^f = 1 
\end{cases}
\]

where they choose \( \alpha = 1 \) and \( \beta = 1 \) for a linear increase or \( \alpha = 2 \) and \( \beta = 0 \) for a geometric increase. This adaptive CFL, thus time step, is attractive because it is based on the behavior of the pseudo-transient continuation method. To improve even more the robustness of the method, they propose to set the solution update to zero when the value of \( \omega^f \) is less than 0.1, i.e., if \( \omega^f < 0.1 \) then \( \omega^f = 0 \).

---

\(^5\)In industry, for some applications, low fidelity models are first used and served as initialization for the higher fidelity CFD model to speed-up the process.
This approach is extremely robust because if the Newton’s method diverges, the current step is cancelled and the time step, via the CFL, is automatically reduced. But the considered criterium is global and hence one bad vertex in the mesh can kill the overall efficiency by not allowing the CFL to grow.

More recently, Pandya et al. proposed a nice approach, called HANIM [75], having some similarities with the method in Wolf. The non-linear iterative method is controlled with three hierarchical levels: i) failure of the preconditioned linear system solver, ii) realizability (i.e., positivity) of the solution update, and iii) non-linear control trying to find the optimal update. Depending on the failure or the success of these checks, the global CFL will be decreased (by a factor 10) and the increment reset or increased (by a factor 2) and the solution updated. In that case, the CFL evolution is based on both the solution and the pseudo-transient continuation method behavior. But, again this is a global CFL control and one bad vertex in the mesh can kill the overall efficiency by not allowing the CFL to grow which puts a lot of constraint on the mesh generation process.

We consider an hybrid method trying to have the efficiency and the robustness of the above presented methods. The CFL growth law is geometric:

\[ CFL^{n+1} = \alpha CFL^n, \]

where \( \alpha \) is the growth geometric coefficient usually taken between 1.01 and 1.1. On main difference with the above methods is that we considered a local CFL at each vertex to remove constraint on the mesh generation side and to not be stopped because of one bad vertex in the mesh. This local CFL is automatically controlled by:

- a global CFL control based on the behavior of the linear and non-linear solve, i.e., the pseudo-transient continuation method
- a local CFL control based on the behavior of the solution at each increment.

If we start from a uniform flow field, then the initial CFL is always set to 0.1 and a geometrical growth between 1.01 and 1.1 is considered (for a \( \alpha \) value equal 1.01, 1.025, and 1.1 the CFL is multiplied by 10 each 230, 93, and 24 iterations, respectively). If we are in the solution-adaptive process, we restart from the previous local CFL that has been interpolated from the previous mesh to the new adapted mesh, see Section 3. This can be done because we restart from the previous solution interpolated on the new adapted mesh. This reduces importantly the runtime of the simulation because restarts are done at high CFL values.

To illustrate the benefits of restarting the simulation with an interpolated local CFL, we consider the HLCRM 2D case. We consider the configuration at 16 degrees of angle of attack, and an adapted mesh composed of 151 821 vertices (obtained at the forth iteration of that complexity). The convergence of the density residual is shown in Figure 5. We observe that the residual drops to \( 10^{-6} \) in 66 iterations with the CFL restart instead of 370 iterations when the CFL is reset. In terms of efficiency the targeted residual is reached in 27 seconds instead of 98 which represents almost a factor 4 of CPU time reduction.

![Figure 5: Comparison of the flow solver convergence on the HLCRM 2D case for an adapted mesh composed of 150K vertices. The blue curve is the density residual convergence when the simulation is re-started with a CFL reset to 0.1 for each vertex. The green curve is the density residual convergence when the simulation is re-started with a local CFL interpolated at each vertex from the previous run.](image)

**Global CFL control based on the non-linear convergence.** We have observed, generally on very fine adapted meshes, that a non-convergence (by at least an order of magnitude) of the linear system residual can lead to a stall of the non-linear residual or even worse a divergence of the non-linear residual. This occurs at high CFL values when the linear system becomes too stiff.
Therefore, it is fundamental to have an automatic control of the global CFL evolution (which is a lot more powerful than ad hoc manual CFL law). A first option to solve this issue is to apply a global CFL reduction if the targeted residual of the linear system is not reached with the maximal number of SGS relaxations. But, we found more advantageous to apply this global CFL reduction if the non-linear residual is increasing.

More precisely, after each solution increment, we analyze if the global non-linear residual and log-residual are increasing (see Section 4.2.8 for the definition of the log-residual). If it is the case, then a global CFL reduction is applied, i.e., we divide the CFL of all the vertices by \( a^2 \), \( a \) being the CFL growth geometric coefficient. This control is absolutely necessary and it perfectly manages the evolution of the CFL.

**Local CFL control based on solution realizability.** The under-relaxation of the increments consists in the reducing locally the increment vector by an under-relaxation coefficient \( \omega \) if the new solution value becomes negatives or close to zero for the density, the pressure, and the turbulent variable. For instance, the smallest admissible value being given by \( \varepsilon W_{\text{ref}} \), then if the new value at vertex \( P_i \) is below this value then the under-relaxation coefficient at vertex \( P_i \) is given by:

\[
\omega_i^t = \frac{W_i^n - \varepsilon W_{\text{ref}}}{\delta W_i^n}.
\]

We are doing the same if the increment is too large, we reduce it by a coefficient \( \omega_j \) if the increment in density or pressure or turbulent variable is larger than \( \beta W_i^n \) (\( \beta \) being the maximal growth allowed at each iteration, it is a percentage). The final local under-relaxation coefficient \( \omega_i \) will be the smallest of all the constraints:

\[
W_i^{n+1} = W_i^n + \omega_i^n \delta W_i^n.
\]

Then, we set-up an evolution law of the local CFL depending on the obtained under-relaxation coefficient \( \omega_i \):

\[
CFL_i^{n+1} = \begin{cases} 
0.1 \, CFL_i^n & \text{if } \omega_i^n \leq 0.1 \\
CFL_i^n & \text{if } 0.1 < \omega_i^n < 1 \\
\alpha \, CFL_i^n & \text{if } \omega_i^n = 1
\end{cases}
\]

where \( \alpha \) is the geometric law growth coefficient. This adaptive local CFL, thus time step, is attractive because it is based on the local behavior of the solution. This helps the convergence to steady state by controlling locally the CFL.

To improve even more the robustness of the method, Burgess and Glasby [20] propose to set the solution update to zero when the value of \( \omega \) is less than 0.1 (they are using global values). In our case, we do it locally: if \( \omega_i < 0.1 \) then \( \omega_i = 0 \) or equivalently \( \delta W_i^n = 0 \). We observe that this reset of the increment improves the convergence of all the variables and the robustness of the flow solver.

**Remark 4.1.** All the proposed components, such as the local CFL adaptation, the under-relaxation of nonlinear solution updates, the control of the non-linear convergence, are intertwined and critically important for the efficiency and robustness of the method.

**Remark 4.2.** This global strategy controlling the CFL can be seen as a method where we locally increase the diagonal dominancy of the matrix only for points with very low or negative eigen values. The main interest is that this correction is only local in the matrix.

### 4.2.8. Residual computation

In a mesh adaptation context, it is crucial to guarantee the convergence of the solver so that no time is wasted on intermediate meshes. We have observed that the global non-linear residual value may be polluted by few vertices (sometime a unique vertex) with a high residual value. In other words, a few vertices that are not converging hide the overall convergence. This is particularly true when shocks are present in the flow field and limiter are used (see next section). This is also true for highly anisotropic adapted meshes where a few vertices may keep high residual value.

The classical formula to compute the non-linear residual is:

\[
Res = \| R \|_2 = \left( \sum_{P_i \in M} R_i^2 \right)^{1/2},
\]

where \( R_i \) is the local residual at vertex \( P_i \) (see Section 4.2.1). This non-linear residual value can be normalized to make it independent of the considered units. Two approaches are possible. The common used one is to normalize with respect to the residual obtained from the farfield uniform flow field. The second one is to normalize with respect to the highest residual obtained during the simulation. In a solution-adaptive process context the main difficulty is that meshes are changing during the process...
and we want a consistant residual computation. In our case, we store the overall maximum residual of the process and use it to normalize the non-linear residual.

However, representing the nodal residual field with this single scalar value leads to a huge loss of information. When we analyze Equation (19), it is clear that a large residual value at one vertex will hide the overall convergence of the simulation. For this reason, we have introduced an histogram of the residual at each vertex. This histogram enables us to see the overall convergence of the solution. But, we also want a scalar value which is representative of this global non-linear residual convergence and less sensitive to a few bad points. To this end, we propose to use a geometric (pseudo-)norm based on the geometric mean which is also called log-average, this is why we will call it log-average non-linear residual or, in short, log-residual:

\[
LogRes = \left( \prod_{P \in \mathcal{H}} R_i \right)^{\frac{1}{N}} = \exp \left( \frac{1}{N} \sum_{P \in \mathcal{H}} \log(R_i) \right),
\]

(20)

where \(N\) is the number of vertices in the mesh. Note that the second expression is a lot more appropriate to compute numerically the \(\text{LogRes}\). To normalize the log-residual expression, we use the maximal local residual of the simulation:

\[
LogRes = \left( \prod_{P \in \mathcal{H}} \frac{R_i}{R_{\text{max}}} \right)^{\frac{1}{N}} = \exp \left( \frac{1}{N} \sum_{P \in \mathcal{H}} \ln \left( \frac{R_i}{R_{\text{max}}} \right) \right) \quad \text{where} \quad R_{\text{max}} = \max_{n \in [0,T]} \left( \max_{P \in \mathcal{H}}(R_i) \right),
\]

like this all the normalized local residual are between 0 and 1. In a solution-adaptive process context, we store the overall maximum local residual of the process and use it to normalize.

This log-residual can also be seen as a statistical residual obtained from the residual histogram. Indeed, this histogram gives useful indications on the convergence and we can determine either all the nodes are still not converged or a few points are not converging. We can compute an approximation of this log-residual using the residual histogram. Instead of using the local residual, we compute a scalar residual by a statistical distribution by counting the number of vertices \(n_j\) with a residual in between \(10^{-j}\) and \(10^{-j+1}\), with \(j \in \mathbb{N}\) (because we use a normalized local residual). In practice, we take \(j \in [0, 15]\). The log-residual can be approximated by:

\[
\text{LogRes} \approx \exp \left( \sum_{j} \frac{n_j}{N} \ln(10^{-j}) \right) = \exp \left( \sum_{j} \ln \left( 10^{-j} \right) \right) = \prod_{j} 10^{-\frac{n_j}{N}} = 10^{-\left( \sum_j \frac{n_j}{N} \right)}. \]

From the histogram, we are also able to compute standard deviation. This is why we think that the log-residual is a lot more appropriate to check the non-linear convergence of the solution.

4.2.9. Limiter Freezing

A major drawback of limiters is that they can prevent the solver from converging by creating limit cycles so that the solution shows very little temporal evolution. This behavior is mainly due to the interaction between the approximate Jacobian used in the implicit solver and small variations in the gradients which induce larger variations in the fluxes through the limiter usage. The usual option to get rid of these fluctuations in the solution is to "freeze" the limiters by keeping the gradients to a given value at some point during the simulation. We prefer to use a different option where we bound the gradient values for the rest of the simulation, in other word the value of the gradient is updated only if it is smaller than the previous value. At worse, the gradient value will converge toward zero and the method will locally drop to first order.

The main difficulty in freezing the limiters is to know when to freeze them which makes this process ad hoc. If it is too early and the solution is not enough converged then it will impact negatively the numerical prediction. If it is too late then CPU time is wasted in the convergence process. In general, limiters are frozen after a given (large) number of iterations leading to a waste of CPU time. The key idea to solve this issue is to have a better view of the overall convergence of the solution and to freeze the limiters when the solution is enough converged. To this end, we propose to use the log-residual as a sensor for the convergence of the non-linear residual because it is less sensitive to the non-convergence of a few vertices. The limiter freezing is then activated when the log-residual is below a given threshold.

4.2.10. Numerical illustraton

To illustrate the benefits of the automatic CFL control and the limiter freezing based on the log-residual threshold, we consider the transonic RAE2822 2D case of Section 8.1. This case is particularly difficult because it has a strong shock on the wing upper part. We analyze the convergence of the non-linear residual on an adapted mesh composed of 153,096 vertices (obtained at the third iteration of that complexity). As we are in the adaptive process, we restart from the previous solution and not from a uniform
one. We consider the geometric CFL growth law with $\alpha = 1.05$. For the convergence, the targeted non-linear residual is $10^{-6}$ and the targeted non-linear log-residual is $3 \times 10^{-11}$. The flow solver stops when one of these values is attained. The limiter freezing is activated when the non-linear log-residual is below $3 \times 10^{-9}$.

Four cases are considered that are summarized in Table below:

<table>
<thead>
<tr>
<th></th>
<th>Automatic CFL Control</th>
<th>Limiter Freezing</th>
<th>Restart CFL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Case 2</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Case 3</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Case 4</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

The convergence of the density residual (left) and the density log-residual (right) are shown in Figure 6. Again, restarting from the previous local CFL field instead of resetting the CFL improves the efficiency of the flow solver (blue curve vs green curve). By comparing the green and the red curves, we observe the effect of the limiter freezing (just before iteration 200): the residual drops quickly but not the log-residual meaning that the limiter freezing acts mainly on few vertices with high residual values. We also observe how it helps to continue to decrease the residual and the log-residual. In the last case (cyan curve) where the CFL increases continuously, we note that at some point the residual increases and then stagnate, underlying the importance of the automatic CFL control.

![Figure 6: Comparison of the flow solver convergence on the RAE2822 2D case of Section 8.1 for an adapted mesh composed of 150K vertices.](image)

5. **Adjoint solver**

The error estimate used in the goal-oriented mesh adaptation presented in Section 6 requires the computation of the numerical adjoint to obtain the sensitivity of the considered output functional $J$ to the flow. We denote by $\mathbf{R}$ the nonlinear residual operator representing the system of equations and $\mathbf{W}$ the exact solution such that $\mathbf{R}(\mathbf{W}) = 0$. We can linearize the nonlinear residual operator as

$$\mathbf{R}(\mathbf{W}) \approx \mathbf{R}(\mathbf{W}_h) + \frac{\partial \mathbf{R}}{\partial \mathbf{W}} (\mathbf{W} - \mathbf{W}_h),$$

where $\mathbf{W}_h$ is the discrete solution. Symbolically, we can invert the above relation leading to:

$$(\mathbf{W} - \mathbf{W}_h) \approx \left(\frac{\partial \mathbf{R}}{\partial \mathbf{W}}\right)^{-1} \mathbf{R}(\mathbf{W}_h).$$

Similarly, we consider an output function $J$ which can be linearized as:

$$J(\mathbf{W}) \approx J(\mathbf{W}_h) + \frac{\partial J}{\partial \mathbf{W}} (\mathbf{W} - \mathbf{W}_h).$$

Combining the above relations, we get:

$$J(\mathbf{W}) \approx J(\mathbf{W}_h) + \left(\frac{\partial J}{\partial \mathbf{W}}\right) \left(\frac{\partial \mathbf{R}}{\partial \mathbf{W}}\right)^{-1} \mathbf{R}(\mathbf{W}_h) = J(\mathbf{W}_h) + \mathbf{W}^* \mathbf{R}(\mathbf{W}_h)$$
where we have introduced $W^*$ the adjoint state which is solution of the following linear system:

$$
\left( \frac{\partial R}{\partial W} \right)^T W^* = \left( \frac{\partial J}{\partial W} \right)^T.
$$

Numerically this consists in computing the sensitivity of a given functional $J(W)$ to the local nodal residual $R_i(W)$, through the linearization of the numerical problem:

$$
\frac{\partial J}{\partial R} = \frac{\partial J}{\partial W} \left( \frac{\partial R}{\partial W} \right)^{-1} = \left( \frac{\partial R}{\partial W} \right)^{-T} \frac{\partial J}{\partial W}.
$$

Consequently, the adjoint matrix is the transpose of the implicit matrix without the mass matrix. The assembling of the adjoint state which is solution of the following linear system:

$$
\left( \frac{\partial R}{\partial W} \right)^T W^* = \left( \frac{\partial J}{\partial W} \right)^T \iff A^* W^* = \left( \frac{\partial J}{\partial W} \right)^T.
$$

In the following sections, we provide the differentiation of aeronautic output functionals that can be used to define the right-hand side (RHS) of the above linear system. Then, we give the choices made to solve the adjoint linear system.

### 5.1. Aeronautic cost function for goal-oriented mesh adaptation

In Section 2.4, we have provided output functionals (drag, lift, slip and moments) used to evaluate aeronautic design. Each of these functionals can be used to define the right-hand side of the adjoint System (21). The differentiation of these functionals relies on the differentiation of the pressure coefficient $C_p(x)$ and skin friction coefficient $C_f(x)$ vectors.

The differentiation of the pressure coefficient is easy, it is given by:

$$
\frac{\partial C_p}{\partial W}(x) = \frac{\partial p}{\partial W}(x) \frac{\mathbf{n}(x)}{2\rho_\infty ||\mathbf{u}_w||^2} \quad \text{with} \quad \frac{\partial p}{\partial W} = (\gamma - 1) \left( \frac{||\mathbf{u}||^2}{2} , \mathbf{u} , 1 \right)^T.
$$

The differentiation of the skin friction coefficient is more complex because it involves gradients. We have:

$$
\frac{\partial C_f}{\partial W}(x) = \frac{\partial r_w}{\partial W}(x) \frac{1}{2\rho_\infty ||\mathbf{u}_w||^2} \quad \text{with} \quad \frac{\partial r_w}{\partial W} = \frac{\partial (\mu + \mu_t)}{\partial W} (\tau(x) \cdot \mathbf{n}(x)) + (\mu + \mu_t) \left( \frac{\partial r}{\partial W}(x) \cdot \mathbf{n}(x) \right).
$$

We now express each differential. For the dynamic viscosity, the differentiation is obtained from the Sutherland’s law:

$$
\frac{\partial \mu}{\partial W} = \frac{\partial \mu}{\partial T} \frac{\partial T}{\partial W} \quad \text{with} \quad \frac{\partial \mu}{\partial T} = \mu \frac{T + 3\mathbf{s}_u}{2 \mathbf{T}(T + \mathbf{s}_u)} \quad \text{and} \quad \frac{\partial T}{\partial W} = \frac{1}{\rho c_v} \left( \frac{||\mathbf{u}||^2 - E}{\mathbf{u}} , 1 \right)^T.
$$

Indeed, we have:

$$
\frac{\partial \mu}{\partial T} = \mu \left( \frac{T + \mathbf{s}_u}{2 \mathbf{T}(T + \mathbf{s}_u)} \right)^2 \left( \frac{2}{2 \mathbf{T}^2 - \mathbf{T}} \right) = \mu \left( \frac{T + \mathbf{s}_u}{2 \mathbf{T}(T + \mathbf{s}_u)} \right)^2 \left( \frac{3}{2 \mathbf{T} - \mathbf{T}^2 - \mathbf{s}_u} \right) = \mu \left( \frac{T + 3\mathbf{s}_u}{2 \mathbf{T}(T + \mathbf{s}_u)} \right).
$$

The turbulent viscosity $\mu_t$ is zero at walls, but its differentiation can be non null. Let us check that it is effectively zero. For the turbulent viscosity, we recall that the residual has been linearize with respect to $\tilde{r}$ thus $\tilde{r}$ should be considered as an independent variable. The goal-oriented error estimate presented in Section 6 does not yet involve the turbulent viscosity, thus we only differentiate with respect to the conservative variables:

$$
\frac{\partial \mu_t}{\partial W} = \frac{\partial (\rho v_i)}{\partial W} = \frac{\partial (f_{i1} \rho \tilde{r}^3)}{\partial W} = \rho \tilde{r} \frac{\partial f_{i1}}{\partial W} + f_{i1} \tilde{v} \frac{\partial \rho}{\partial W}.
$$

The second term is trivial, let us express the differentiation of $f_{i1}$:

$$
\frac{\partial f_{i1}}{\partial W} = \frac{\partial \left( \frac{\chi}{\chi^3 + c_{i1}} \right)}{\partial W} = \frac{3 \chi^2 \frac{\partial \chi}{\partial W} \left( \chi^3 + c_{i1} \right) - 3 \chi^2 \frac{\partial \chi}{\partial W} \chi^3}{\left( \chi^3 + c_{i1} \right)^2} = \frac{3 \chi^2 c_{i1}^3}{\left( \chi^3 + c_{i1} \right)^2} \frac{\partial \chi}{\partial W}.
$$
with \( \chi = \hat{\varphi}/\varphi \) which is differentiated as:

\[
\frac{\partial \chi}{\partial W} = \frac{\hat{\varphi}}{\varphi^2} \frac{\partial \varphi}{\partial W} \quad \text{and} \quad \frac{\partial \varphi}{\partial W} = \frac{1}{\rho} \frac{\partial \mu}{\partial W} - \frac{\mu}{\rho^2} \frac{\partial \rho}{\partial W}.
\]

The final expression is:

\[
\frac{\partial \mu}{\partial W} = -\frac{3 \rho \varphi^2 \chi \hat{\varphi} c_l^3}{(\chi^3 + c_l^3)^2} \frac{\partial \nu}{\partial W} + f_{st} \frac{\partial \rho}{\partial W} = -\frac{3 \chi^4 c_l^3}{(\chi^3 + c_l^3)^2} \frac{\partial \mu}{\partial W} + \left( \frac{3 \chi^3}{(\chi^3 + c_l^3)^2} + f_{st} \hat{\varphi} \right) \frac{\partial \rho}{\partial W}.
\]

As \( \hat{\varphi} \) is zero at walls, we deduce that \( \frac{\partial \mu}{\partial W} = 0 \) at walls. Consequently, we obtain exactly the same expression for \( \frac{\partial C_f}{\partial W} \) when the QCR version of the Spalart-Allmaras turbulence model is considered.

It remains to evaluate the differentiation of the stress tensor \( \tau \) which involves gradients and is discretized with a Finite Element approach (see 4.1.2 for details). We proceed exactly the same as for the viscous Jacobians presented in Section 4.2.3, except that the viscosity is evaluated at the considered vertex. We note that \( \frac{\partial \tau}{\partial W} \) gives contributions to the considered vertex but also to its neighbors. Therefore, the vector \( \frac{\partial C_f}{\partial W} \) is not null at the wall and also for all internal vertices connected to the wall.

Finally, to get the differentiation of the drag or the lift, it is sufficient to apply the rotation matrix given in Section 2.4. Namely, the aerodynamic coefficient vector is given by Relation (9), hence, as the integration surface does not change,

\[
\frac{\partial}{\partial W} \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix} = \frac{1}{S_{Ref}} \int_S \left( \frac{\partial}{\partial W} C_f(x) + \frac{\partial}{\partial W} C_f(x) \right) \, dx,
\]

where the differentiation of the pressure and friction coefficients have been developed above. The rotation \( \text{Rot}(\theta, \alpha, \sigma) = R_\alpha^t R_\sigma^t R_\gamma^t \) is then applied to the vector

\[
\frac{\partial}{\partial W} \begin{pmatrix} \text{Drag} \\ \text{Lift} \end{pmatrix} = \text{Rot}(\theta, \alpha, \sigma) \frac{\partial}{\partial W} \begin{pmatrix} C_x \\ C_y \\ C_z \end{pmatrix},
\]

to obtain the differentiation of the drag, lift and slip coefficients.

**Remark 5.1.** The error estimate proposed in Section 6 does not consider the turbulent variable \( \hat{\varphi} \) in its formulation (it is implicitly taken into account via \( \mu \)), but if the error estimation involves the turbulent variable then it will require to solve the fully coupled system and to differentiate the cost functions w.r.t to \( \hat{\varphi} \).

### 5.2. Boundary adjoint correction

As stated above, a formal description of the nodal value of the adjoint is the sensitivity of a given functional to a correction or perturbation of local nodal residual. In that respect, Dirichlet boundary conditions have a specific behavior. Indeed, as we impose a given field value at these nodes, their residual is discarded and plays no role in the computation. There is thus no dependency of the considered functional to these nodes and formally the adjoint should be zero at these boundaries. In practice, as the adjoint is computed with the transpose of the Jacobian matrix whose values have been corrected to impose the Dirichlet boundary conditions (see Section 4.2.5), the adjoint reflects the sensitivity of the functional to these boundary values. In consequence, the adjoint field is discontinuous at Dirichlet boundary conditions such as no-slip boundaries. However, in the context of error estimation for mesh adaptation, we assume the adjoint to be a continuous inner function. Therefore, at Dirichlet boundary conditions, we replace its value by an extrapolation from the interior of the domain using a quadratic least square fitting.

### 5.3. Solving the adjoint linear system

Goal-oriented error estimates rely on an accurate computation of the adjoint state that proves to be a stiff problem for RANS equations. Failing to converge the adjoint linear system to machine zero may impact negatively the adaptive process, eg. we observe noise in the adapted mesh. In our applications, we generally converge the linear system by twelve orders of magnitude to obtain an accurate adjoint state for mesh adaptation. But, this linear system is a lot more stiffer than the flow solver one because there is no mass matrix, i.e. it is equivalent to set a CFL equal to infinity. For inviscid flows, we didn’t see any issue in solving that linear system using a restarted GMRES preconditioned with LUSGS relaxation. However, we have observed that the adjoint linear system is a lot harder to converge for RANS. In fact, the finer is the adapted mesh the stiffer is the linear system.
At first, we have increased the Krylov space size but on highly-resolved anisotropic adapted meshes it was frequent that the linear system did not converge (e.g., the residual was reduced by three to five orders of magnitude) after 10,000 iterations and a Krylov space size of 1,000! Such Krylov space sizes have stringent memory requirements which make them unusable in practice.

The key to solve this issue was to consider a stronger preconditioner: a restarted GMRES preconditioned with SGS relaxation. For the preconditioner, we perform several passes of SGS (generally between 20 and 40 passes). In that case, we observe that the linear system is converging when considering Krylov space of size up to 200.

To illustrate the necessity of using a stronger preconditioner, we consider the computation of the adjoint state for three examples:

- the RAE2822 airfoil test case of Section 8.1 on an adapted mesh composed of approximately 600,000 vertices,
- the 2D HLCRM multi-element airfoil test case (the 2D version of Section 9.2) on an adapted mesh composed of approximately 600,000 vertices,
- and, the ONERA M6 wing test case of Section 8.2 on an adapted mesh composed of approximately 1,800,000 vertices.

For all the cases, the Krylov space size is set to 200 and the targeted residual is $10^{-12}$. The maximum number of iterations is set to 10,000 but the computation is stopped if the GMRES residual stalls. We compare four different preconditioners: block Jacobi, LUSGS, SGS with 20 sweeps (backward and forward), and the SGS with 40 sweeps. The results are shown in Figure 7.

The GMRES stalls with block Jacobi preconditioner for the two 2D cases and converges very slowly for the 3D case (it will stall on a finer adapted mesh). A similar behavior is observed for the GMRES with LUSGS preconditioner even if it is slightly better than the block Jacobi. For all the cases, the GMRES with the SGS preconditioner is converging. We observe that the stiffer the system (i.e., the more resolved is the mesh) the more SGS passes are required to keep an efficient preconditioner. Similar conclusions are obtained on complex 3D geometries such as the HLCRM of Section 9.2.

Figure 7: Illustration of the convergence of the adjoint linear system. From left to right, convergence of the adjoint linear system for four preconditioner for the RAE2822 airfoil of Section 8.1, the 2D HLCRM multi-element airfoil and the ONERA M6 wing of Section 8.2. Top, convergence with respect to the number of iterations and bottom with respect to the CPU time.
6. RANS error estimates

In the mesh adaptation process, the metric field \( (M(x))_{\Omega} \) used to prescribe the new adapted mesh \( \mathcal{H} \) is automatically deduced from the actual solution or from the actual solution and adjoint state with different error estimates [7]. The goal is to find the optimal mesh \( \mathcal{H}_{opt} \) which minimizes the given error model \( E(\mathcal{H}) \) for a fixed number of elements \( C(\mathcal{H}_{opt}) = N \):

\[
\mathcal{H}_{opt} = \arg\min_{C(\mathcal{H})=N} E(\mathcal{H}).
\]

This problem can be analytically solved by recasting it in the continuous mesh theoretical framework [56, 57]: the goal is to find the optimal continuous mesh \( M_{opt} \) which minimizes the given continuous error model \( E(C(\mathcal{M})) \) for a fixed continuous mesh complexity \( C(\mathcal{M}) = N \):

\[
M_{opt} = \arg\min_{C(\mathcal{M})=N} E(C(\mathcal{M})).
\]

The continuous mesh complexity is the continuous counterpart of the discrete mesh size (number of points or elements) and is used to prescribed the mesh size during the adaptation process. It is given by relation [55, 7]

\[
C(\mathcal{M}) = \int_{\Omega} \sqrt{|\det M|} \, d\Omega.
\]

There is a direct relationship between the prescribed metric complexity and the number of elements of the generated mesh. If a unit mesh \( \mathcal{H} = \cup K \) (where the \( K \) are the tetrahedra of mesh \( \mathcal{H} \)) is generated with respect to \( (M(x))_{\Omega} \) [55, 7], then the continuous mesh complexity and the mesh size are linked in 3D by:

\[
C(\mathcal{M}) \approx \sum_k \sqrt{|\det M_k|} |K| \approx \sum_k \frac{\sqrt{7}}{12} = \frac{\sqrt{7}}{12} \times nt,
\]

where \( |K| \) is the volume of \( K \), \( M_k \) is the average metric at element \( K \), and \( nt \) is the number of elements of the mesh.

In the following, we recall the multiscale error estimate used for feature-based mesh adaptation. It is based on a control of the interpolation error of a given sensor in \( L^p \)-norm. We also recall the goal-oriented error estimate based on the \textit{a priori} analysis proposed in [16] which proved to be very efficient for laminar flows. Then, we propose a new \textit{a priori} goal-oriented error analysis which turns out to be more accurate for RANS simulations. In the numerical result sections, we will analyze the influence of the different error estimates on the accuracy of the solution-adaptive simulations.

6.1. Multiscale error estimate

The most natural and straight forward approach is to control the interpolation error of a sensor field \( u = f(W) \) [21, 29, 55] which is defined from solution field \( W \). Given a continuous sensor \( u \), it is represented by its discrete nodal values on the mesh \( u_i = u(x_i) \) and its piecewise linear representation \( \Pi_h u \) on mesh \( \mathcal{H} \). The \( L^p \)-norm of the interpolation error of the sensor field \( u \) is stated as

\[
E_{L^p}(\mathcal{H}) = \left( \int_{\Omega} |u - \Pi_h u|^p \right)^{1/p}.
\]

Feature-based mesh adaptation generates discrete adapted meshes that minimizes the global interpolation error of the given sensor field \( u \) for the considered number of elements. Under certain assumptions, we can prove that this approach also controls the approximation error [61]. The analytical expression of the optimal continuous mesh \( M_{opt} \) that minimizes the interpolation error in \( L^p \)-norm of sensor \( u \) for a given complexity \( N \) is [6]:

\[
M_{opt}(x) = N^{1/2} \left( \int_{\Omega} |\det(H_u(x))|^{1/2} \, d\Omega \right)^{-1/2} \det(H_u(x))^{-1/2} |H_u(x)|,
\]

where \( d \) is the space dimension and \( H_u \) is the Hessian of the sensor \( u \) computed using a double \( L^2 \)-projection method, see Section 7.3. Then, to generate the associated adapted mesh, a unit mesh is generated with respect to this metric field, see Section 7.1.

For inviscid flows, many studies have pointed out that controlling the \( L^2 \)-norm of the interpolation error of the chosen sensor is the most appropriate choice [7, 60]. But recently, Park and Balan [78] have pointed out benefits of controlling the interpolation error in \( L^1 \)-norm for the ONERA M6 RANS case instead of using the \( L^p \)-norm because this norm choice captures and adapts quicker the boundary layer region. Therefore, in this paper, we consider the \( L^1 \)-norm of the interpolation error of the Mach sensor field as feature-based error estimates. Controlling the interpolation error of the Mach sensor field will tend to also control the interpolation error of all primitive variables.
6.2. Goal-oriented error estimate for laminar flows

Feature-based adaptation is efficient to control the interpolation error of a sensor field but it ignores the non-linear dependency of the solution \( W \) to the mesh and the considered system of equations. It improves the overall solution but ignores the ultimate goal of the simulation. In our case, we are interested in computing the main aerodynamic coefficient of the solution. This is achieved by considering goal-oriented error estimates where the non-linear dependency is taken into account by using the adjoint state \( W' \). The goal is to minimize the approximation error in the computation of a given scalar functional \( J(W) \):

\[
E_{\text{fcol}}(\mathcal{H}) = |J(W) - J(W_h)| = |\delta J|.
\]

To do so, various approaches have been proposed to take into account the sensitivity of the functional \( J \) to the local residual with the discrete adjoint. Namely, as the discrete numerical solution \( W_h \) is sought to cancel the residual \( R(W_h) = 0 \), any error in the computation of the residual \( \delta R \) will lead to an error in the computed solution

\[
\delta W = \left( \frac{\partial R}{\partial W} \right)^{-1} \delta R,
\]

where \( \frac{\partial R}{\partial W} \) holds for the standard Jacobian of the residual. This error in the solution leads to an error in the computation of the output functional

\[
\delta J \approx \frac{\partial J}{\partial W} \delta W \approx \frac{\partial J}{\partial W} \left( \frac{\partial R}{\partial W} \right)^{-1} \delta R,
\]

where \( W^* = \left( \frac{\partial R}{\partial W} \right)^{-T} \frac{\partial J}{\partial W} \) is the discrete adjoint. The local error in the residual \( \delta R \) is then related to the local mesh size by different means \([31, 43, 59, 78, 79, 93]\).

We use here a different goal-oriented error analysis to analytically derive optimal metric field from the adjoint and primal fields. This a priori analysis has been done for inviscid flows and has shown excellent results \([59]\). The same analysis has been extended to laminar viscous flows by taking into account viscous terms and shown very good results \([16]\). The main advantage of these estimates in comparison to other goal-oriented error estimates is that the anisotropy of the mesh appears naturally using the continuous mesh framework \([56, 57]\) and they provide directly the analytical expression of the optimal mesh. In our error estimation problem, the approximation error on the functional can be decomposed into an implicit error term and an interpolation error term:

\[
|\delta J| = |J(W) - J(W_h)| = |\frac{\partial J}{\partial W}, W - W_h| \leq |\frac{\partial J}{\partial W}, W - \Pi_h W| + |\frac{\partial J}{\partial W}, \Pi_h W - W_h|.
\]

The interpolation error term is easily estimated using Relation (22). For the implicit term we demonstrate \([16, 59]\) that:

\[
|\frac{\partial J}{\partial W}, W_h - \Pi_h W| \approx (\Psi(W) - \Psi_h(\Pi_h W), \Pi_h W^*) ,
\]

where \( \Psi \) and \( \Psi_h \) are the continuous and discrete operators of the considered problem. We use the Finite Element framework to accurately analyze the implicit error term involved in the Navier-Stokes equations.

We denote by \( F^E(W) \) the convective fluxes given by Relations (6) and \( F^V(W) \) the viscous fluxes deduced from Relations (7), e.g. \( S(W) = V \cdot F^V(W) \). In \([16]\), we have proved the following error estimation:

\[
|\frac{\partial J}{\partial W}, W_h - \Pi_h W| = \int_{\Omega} \left| W^*_h \right| |V \cdot (F^E(W) - F^E(\Pi_h W)) - V \cdot (F^V(W) - F^V(\Pi_h W))| \, d\Omega
\]

\[
\leq \int_{\Omega} \left| W^*_h \right| |F^E(W) - \Pi_h F^E(W)| \, d\Omega
\]

\[
+ \sum_i \int_{\Omega} G\left(W, \rho u(W^*)\right) |T - \Pi_h T| \, d\Omega + \sum_i \int_{\Omega} G\left(W, \rho u(W^*)\right) |u_i - \Pi_h u_i| \, d\Omega,
\]

where \( T \) is the temperature, \( u_i \) the components of the velocity vector, and \( \rho u(W^*) \) is the spectral radius of the Hessian of the adjoint state, i.e., the largest eigen value in absolute value. The functions \( G_T \) and \( G_u \) are given in 3D by:

\[
G_u = (\mu + \mu_t) \left( 20 |\rho u(\rho u^*)| + 2 |\rho u(\rho v^*)| + 2 |\rho u(\rho w^*)| + \left( 20 |u| + 2 |v| + 2 |w| \right) |\rho u(\rho E^*)| + \frac{5}{3} |\omega_{u,v} - \omega_{u,v^*}| \right)
\]

\[
G_v = (\mu + \mu_t) \left( 2 |\rho u(\rho u^*)| + 20 |\rho u(\rho v^*)| + 2 |\rho u(\rho w^*)| + \left( 20 |u| + 20 |v| + 2 |w| \right) |\rho u(\rho E^*)| + \frac{5}{3} |\omega_{u,v} - \omega_{u,v^*}| \right)
\]

\[
G_w = (\mu + \mu_t) \left( 2 |\rho u(\rho u^*)| + 2 |\rho u(\rho v^*)| + 20 |\rho u(\rho w^*)| + \left( 20 |u| + 2 |v| + 20 |w| \right) |\rho u(\rho E^*)| + \frac{5}{3} |\omega_{u,v} - \omega_{u,v^*}| \right)
\]

\[
G_T = 18 (\lambda + \lambda_t) |\rho u(\rho E^*)|
\]
with $\mu$ (resp. $\mu_t$) the (resp. turbulent) viscosity, $\lambda$ (resp. $\lambda_t$) the (resp. turbulent) heat conduction and the $\omega$ vector defined by:

$$\nabla u \times \nabla \rho E^* = \omega_u = \left( \omega_{u_1}, \omega_{u_3}, \omega_{u_z} \right)^T,$$

$$\nabla v \times \nabla \rho E^* = \omega_v = \left( \omega_{v_1}, \omega_{v_3}, \omega_{v_z} \right)^T,$$

$$\nabla w \times \nabla \rho E^* = \omega_w = \left( \omega_{w_1}, \omega_{w_3}, \omega_{w_z} \right)^T.$$

We observe that the above error estimate of the implicit error is a weighted sum of interpolation error in $L^1$-norm (involving 19 interpolation error terms in 3D). To control the approximation error on $J$ we also sum the 5 terms of the interpolation error of Relation (23). Thus, we can directly apply to this sum the formulation of the optimal continuous mesh, Relation (22), to find the analytical expression of the optimal goal-oriented continuous mesh.

**Remark 6.1.** Goal-oriented error estimates rely on an accurate computation of the adjoint state that proves to be a stiff problem for RANS equations. Failing to converge the adjoint state to a sufficiently precise may impact negatively the adaptive process. The use of a strong preconditioner as stated in Section 5 is fundamental to converge the adjoint state.

6.3. Improved goal-oriented error estimate for RANS flows

In [71], a different analysis of $|J(W) - J(W_h)|$ has been proposed which has proved to be more efficient to predict the drag of a turbulent flow around the ONERA M6 wing than the above laminar goal-oriented error estimate. But, an approximation was made by using only the Hessian of the Mach number variable. However, it is possible to combine both analysis to derive a sharper RANS goal-oriented error estimate. In the following, we give a formal analysis of the error estimate using the vector form of the Navier-Stokes equations. In this formal analysis some simplifications are done but it gives the spirit of the proof of the error estimate. The complete proof is provided in Appendix C where the terms are expended for a rigorous mathematical analysis.

In our error estimation problem, the approximation error on the functional can be decomposed into an implicit error term and an interpolation error term:

$$\delta J = J(W) - J(W_h) = \left( \frac{\partial J}{\partial W} , W - W_h \right) = \left( \frac{\partial J}{\partial W} , W - \Pi_h W \right) + \left( \frac{\partial J}{\partial W} , \Pi_h W - W_h \right).$$

(26)

The interpolation error term is already a sum of weighted interpolation error on the conservative variables. We have to analyze the implicit error term. To this end, we start from Relation (24) which becomes for the Navier-Stokes equations in vector form:

$$\left( \frac{\partial J}{\partial W} , W_h - \Pi_h W \right) \approx \int_{\Omega} \left( \nabla \cdot (\mathcal{F}^E(W) - \mathcal{F}^E(\Pi_h W)) - \nabla \cdot (\mathcal{F}^V(W) - \mathcal{F}^V(\Pi_h W)) \right) \cdot W^* \ \text{d} \Omega .$$

The aim of the error analysis is to transform the above expression in a sum of weighted interpolation error on the conservative variables. Then, we will be able to apply the continuous mesh framework to obtain the analytical expression of the optimal metric field.

For the inviscid fluxes, we first integrate by part omitting boundary terms:

$$\int_{\Omega} \left( \nabla \cdot (\mathcal{F}^E(W) - \mathcal{F}^E(\Pi_h W)) \right) \cdot W^* \ \text{d} \Omega \approx - \int_{\Omega} \left( \sum_i \left( \mathcal{F}_i^E(W) - \mathcal{F}_i^E(\Pi_h W) \right) \nabla_i W^* \right) \ \text{d} \Omega ,$$

and then we linearize the convective fluxes with respect to the conservative solution vector $W$:

$$\int_{\Omega} \left( \nabla \cdot (\mathcal{F}^E(W) - \mathcal{F}^E(\Pi_h W)) \right) \cdot W^* \ \text{d} \Omega \approx - \int_{\Omega} \left( \sum_i \left( \left( \frac{\partial \mathcal{F}_i^E}{\partial W} \right) (W - \Pi_h W) \nabla_i W^* \right) \right) \ \text{d} \Omega$$

$$\approx - \int_{\Omega} \left( \sum_i \left( \frac{\partial \mathcal{F}_i^E}{\partial W} \right) \nabla_i W^* \right) (W - \Pi_h W) \ \text{d} \Omega .$$

(27)

For this formal error analysis, viscous fluxes should be seen as functions of the gradient of $W$ and not as function of $W$, that we re-write:

$$\mathcal{F}_i^V(\nabla W) = \mathcal{F}_i^V(W).$$

Thus, other terms that are not gradients are considered as constant. This is a strong simplification but this formal analysis will give the idea of the proof of the error estimate. In Appendix C, the complete proof is provided where the terms are expended for a more rigorous mathematical analysis. Formally, the error analysis of the viscous term is the following:

$$- \int_{\Omega} \left( \nabla \cdot (\mathcal{F}^V(W) - \mathcal{F}^V(\Pi_h W)) \right) \cdot W^* \ \text{d} \Omega = \int_{\Omega} \left( \nabla \cdot (\mathcal{F}^V(W) - \mathcal{F}^V(\Pi_h W)) \right) \cdot W^* \ \text{d} \Omega$$

$$\approx \int_{\Omega} \left( \sum_i \left( \mathcal{F}_i^V(\nabla W) - \mathcal{F}_i^V(\nabla \Pi_h W) \right) \nabla_i W^* \right) \ \text{d} \Omega .$$

(29)
after integration by part (omitting boundary terms). Then, we linearize each viscous fluxes term \( \mathcal{F}_i^V \) with respect to \( \nabla W \) as:

\[
\tilde{\mathcal{F}}_i^V (\nabla W) \approx \sum_j \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial W_{x_j}} \right)^T W_{x_j} = \sum_j \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial W_{x_j}} \right)^T \partial W_{x_j}
\]

and:

\[
\tilde{\mathcal{F}}_i^V (\nabla \Pi_b W) \approx \sum_j \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial \Pi_b W_{x_j}} \right)^T W_{x_j} = \sum_j \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial \Pi_b W_{x_j}} \right)^T \partial \Pi_b W_{x_j}.
\]

Therefore, after integration by part (omitting boundary terms). Then, we linearize each viscous fluxes term \( \mathcal{F}_i^V \) with respect to \( \nabla W \) as:

\[
\tilde{\mathcal{F}}_i^V (\nabla W) \approx \sum_j \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial W_{x_j}} \right)^T W_{x_j} = \sum_j \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial W_{x_j}} \right)^T \partial W_{x_j}
\]

and:

\[
\tilde{\mathcal{F}}_i^V (\nabla \Pi_b W) \approx \sum_j \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial \Pi_b W_{x_j}} \right)^T W_{x_j} = \sum_j \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial \Pi_b W_{x_j}} \right)^T \partial \Pi_b W_{x_j}.
\]

where \( \{W_{x_i}\}_{i,j} \) are second derivative vectors of all the adjoint conservative variables. As we have:

\[
\left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial W_{x_i}} \right)^T (W - \Pi_b W) \cdot (W_{x_i}) = \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial W_{x_i}} \right)^T (W_{x_i}) \cdot (W - \Pi_b W).
\]

We finally get:

\[
\int_{\Omega} \nabla \cdot (\tilde{\mathcal{F}}^V (W) - \mathcal{F}^V (\Pi_b W)) \cdot W^* d\Omega \approx - \int_{\Omega} \left( \sum_i \left( \frac{\partial \tilde{\mathcal{F}}_i^V}{\partial W_{x_i}} \right)^T (W - \Pi_b W) \cdot (W_{x_i}) \right) (W - \Pi_b W) d\Omega.
\]

From now, we omit the tilde on the viscous flux function. Finally, we obtain the following estimation of the implicit error:

\[
\frac{\partial J}{\partial W} (W_h - \Pi_b W) \approx \int_{\Omega} \sum_i \frac{\partial \mathcal{F}_i^E}{\partial W} \nabla x_i W^* + \sum_{j} \left( \frac{\partial \mathcal{F}_j^V}{\partial \nabla x_j W} \right)^T (W_{x_i}) (W - \Pi_b W) d\Omega. \tag{28}
\]

Now, we can express the approximation error on the output function using Equation (26):

\[
J(W) - J(W_h) \approx \int_{\Omega} \left( \frac{\partial J}{\partial W} - \sum_i \frac{\partial \mathcal{F}_i^E}{\partial W} \nabla x_i W^* - \sum_{j} \left( \frac{\partial \mathcal{F}_j^V}{\partial \nabla x_j W} \right)^T (W_{x_i}) \right) (W - \Pi_b W) d\Omega.
\]

which is bounded by:

\[
|J(W) - J(W_h)| \leq \int_{\Omega} \left| - \frac{\partial J}{\partial W} + \sum_i \frac{\partial \mathcal{F}_i^E}{\partial W} \nabla x_i W^* + \sum_{j} \left( \frac{\partial \mathcal{F}_j^V}{\partial \nabla x_j W} \right)^T (W_{x_i}) \right| |W - \Pi_b W| d\Omega = \int_{\Omega} \sum_i |G(W_i)| |W_i - \Pi_b W| d\Omega. \tag{29}
\]

This error estimate is a weighted sum of interpolation errors in \( L^1 \)-norm on the conservative variables where the weights depend on the gradient and the hessian of the adjoint state and on the convective and viscous fluxes Jacobians. As one can see, this is quite different from error estimate (25). For this sharper error estimate, we tolerate compensation of terms in the weight of the interpolation error. This is a major difference with the error estimate of the previous section [16]. Moreover, in 3D, it involves only 5 weighted interpolation error terms instead of 19 + 5 with the previous error estimate. Thus, we can directly apply this sum the formulation of the optimal continuous mesh, Relation (22), to find the analytical expression of the optimal goal-oriented continuous mesh. For a practical implementation, we give the expression of the weights for each conservative variable. The proof to find these weights is provided in Appendix C where the terms are expended for a rigorous mathematical analysis.
2D practical implementation. In 2D, the weights are for each conservative variable:

$$G(\rho) = -\frac{\partial j}{\partial \rho} + \sum_i \frac{\partial F_i^E}{\partial \rho} \cdot \nabla_s W^* - u f_{\rho u}(W, H_W^*) - v f_{\rho v}(W, H_W^*) - (E - u^2 - v^2) f_{\rho E}(W, H_W^*)$$

$$G(\rho u) = -\frac{\partial j}{\partial \rho u} + \sum_i \frac{\partial F_i^E}{\partial \rho u} \cdot \nabla_s W^* + f_{\rho u}(W, H_W^*) - u f_{\rho E}(W, H_W^*)$$

$$G(\rho v) = -\frac{\partial j}{\partial \rho v} + \sum_i \frac{\partial F_i^E}{\partial \rho v} \cdot \nabla_s W^* + f_{\rho v}(W, H_W^*) - v f_{\rho E}(W, H_W^*)$$

$$G(\rho E) = -\frac{\partial j}{\partial \rho E} + \sum_i \frac{\partial F_i^E}{\partial \rho E} \cdot \nabla_s W^* + f_{\rho E}(W, H_W^*)$$

with

$$f_{\rho u}(W, H_W^*) = \frac{1}{3} \left( \frac{\mu + \mu_i}{\rho} \left( 4 (\rho u^*)_{xx} + 3 (\rho u^*)_{yy} + 3 (\rho u^*)_{zz} + (\rho v^*)_{xy} + 4 u (\rho E^*)_{xx} \right) + 4 u (\rho E^*)_{xy} + 3 u (\rho E^*)_{yy} + 3 u (\rho E^*)_{zz} - 5 \omega_{vz} + 5 \omega_{uv} \right)$$

$$f_{\rho v}(W, H_W^*) = \frac{1}{3} \left( \frac{\mu + \mu_i}{\rho} \left( 3 (\rho v^*)_{xx} + 4 (\rho v^*)_{yy} + 3 (\rho v^*)_{zz} + (\rho u^*)_{xy} \right) + 3 v (\rho E^*)_{xx} + 4 v (\rho E^*)_{yy} + w (\rho E^*)_{yz} + 3 v (\rho E^*)_{yz} + 5 \omega_{ux} - 5 \omega_{uv} \right)$$

$$f_{\rho E}(W, H_W^*) = \frac{1}{3} \left( \frac{\mu + \mu_i}{\rho} \left( 3 (\rho w^*)_{xx} + 3 (\rho w^*)_{yy} + 4 (\rho w^*)_{zz} + (\rho w^*)_{xy} \right) + 3 w (\rho E^*)_{xx} + 3 w (\rho E^*)_{yy} + v (\rho E^*)_{yz} + 4 w (\rho E^*)_{yz} - 5 \omega_{uy} + 5 \omega_{ux} \right)$$

where the terms $\frac{\partial j}{\partial W}$ have been provided in Section 5.1, $\omega$ given by Relation (C.5) and the Jacobians of the convective fluxes given in Appendix B.

3D practical implementation. In 3D, the weights are for each conservative variable:

$$G(\rho) = -\frac{\partial j}{\partial \rho} + \sum_i \frac{\partial F_i^E}{\partial \rho} \cdot \nabla_s W^* - u f_{\rho u}(W, H_W^*) - v f_{\rho v}(W, H_W^*) - w f_{\rho w}(W, H_W^*) - (E - u^2 - v^2 - w^2) f_{\rho E}(W, H_W^*)$$

$$G(\rho u) = -\frac{\partial j}{\partial \rho u} + \sum_i \frac{\partial F_i^E}{\partial \rho u} \cdot \nabla_s W^* + f_{\rho u}(W, H_W^*) - u f_{\rho E}(W, H_W^*)$$

$$G(\rho v) = -\frac{\partial j}{\partial \rho v} + \sum_i \frac{\partial F_i^E}{\partial \rho v} \cdot \nabla_s W^* + f_{\rho v}(W, H_W^*) - v f_{\rho E}(W, H_W^*)$$

$$G(\rho w) = -\frac{\partial j}{\partial \rho w} + \sum_i \frac{\partial F_i^E}{\partial \rho w} \cdot \nabla_s W^* + f_{\rho w}(W, H_W^*) - w f_{\rho E}(W, H_W^*)$$

$$G(\rho E) = -\frac{\partial j}{\partial \rho E} + \sum_i \frac{\partial F_i^E}{\partial \rho E} \cdot \nabla_s W^* + f_{\rho E}(W, H_W^*)$$

where we have

$$f_{\rho u}(W, H_W^*) = \frac{1}{3} \left( \frac{\mu + \mu_i}{\rho} \left( 4 (\rho u^*)_{xx} + 3 (\rho u^*)_{yy} + 3 (\rho u^*)_{zz} + (\rho v^*)_{xy} + 4 u (\rho E^*)_{xx} \right) + 4 u (\rho E^*)_{xy} + 3 u (\rho E^*)_{yy} + 3 u (\rho E^*)_{zz} - 5 \omega_{vz} + 5 \omega_{uv} \right)$$

$$f_{\rho v}(W, H_W^*) = \frac{1}{3} \left( \frac{\mu + \mu_i}{\rho} \left( 3 (\rho v^*)_{xx} + 4 (\rho v^*)_{yy} + 3 (\rho v^*)_{zz} + (\rho u^*)_{xy} \right) + 3 v (\rho E^*)_{xx} + 4 v (\rho E^*)_{yy} + w (\rho E^*)_{yz} + 3 v (\rho E^*)_{yz} + 5 \omega_{ux} - 5 \omega_{uv} \right)$$

$$f_{\rho w}(W, H_W^*) = \frac{1}{3} \left( \frac{\mu + \mu_i}{\rho} \left( 3 (\rho w^*)_{xx} + 3 (\rho w^*)_{yy} + 4 (\rho w^*)_{zz} + (\rho w^*)_{xy} \right) + 3 w (\rho E^*)_{xx} + 3 w (\rho E^*)_{yy} + v (\rho E^*)_{yz} + 4 w (\rho E^*)_{yz} - 5 \omega_{uy} + 5 \omega_{ux} \right)$$

$$f_{\rho E}(W, H_W^*) = \frac{1}{3} \left( \frac{\mu + \mu_i}{\rho} \left( 3 (\rho w^*)_{xx} + 3 (\rho w^*)_{yy} + 4 (\rho w^*)_{zz} + (\rho w^*)_{xy} \right) + 3 w (\rho E^*)_{xx} + 3 w (\rho E^*)_{yy} + v (\rho E^*)_{yz} + 4 w (\rho E^*)_{yz} - 5 \omega_{uy} + 5 \omega_{ux} \right)$$

where the terms $\frac{\partial j}{\partial W}$ have been provided in Section 5.1, $\omega$ given by Relation (C.6) and the Jacobians of the convective fluxes given in Appendix B.
Remark 6.2. In this work, we do not have considered the turbulent equation in the goal-oriented error estimate. This is left for future developments. Note that in this simplified version, the turbulence model is taken into account implicitly throughout $\mu_i$ and $\Lambda_i$.

7. Anisotropic mesh adaptation

7.1. The local adaptive remesher: Feflo.a

We give a brief overview of the AMG/Feflo.a meshing algorithm that is used as the local adaptive remesher. The main features are:

- it is metric-based and uses the concept of unit mesh,
- the volume and the surface meshes are adapted simultaneously in order to keep a valid 3D mesh throughout the entire process. This guarantees the robustness of the complete remeshing step,
- it relies on a single cavity operator capable of automatically managing a combination of generalized standard operators in one go (insertion, collapse, swap of edges and faces) at once,
- it is capable of handling extremely anisotropic meshes,
- the surface geometry can be either represented by the CAD or a $P^3$ surface mesh model. Each time a vertex is inserted or moved on the surface then it is projected on it using one of these two geometric models.

For a complete description, we refer to [65, 53]. Other local adaptive remeshers exist such as EPIC [72] or Refine [43].

Metric-based and unit-mesh concept. AMG/Feflo.a is a generic purpose adaptive mesh generator dealing with 2D, 3D and surface mesh generation. It belongs to the class of metric-based mesh generator [63, 25, 49, 28, 72] which aims at generating a unit mesh with respect to a prescribed metric field $\mathcal{M}$. A mesh is said to be unit when composed of almost unit-length edges and unit-volume element. The length of an edge $AB$ in $\mathcal{M}$ is evaluated with:

$$\ell_{\mathcal{M}}(AB) = \int_0^1 \sqrt{AB \mathcal{M}((1-t)A + tB)AB} \, dt,$$

while the volume is given by $|K|_{\mathcal{M}} = \sqrt{\det \mathcal{M}[K]}$, where $|K|$ is the Euclidean volume of $K$. From a practical point of view, the volume and length requirements are combined into a quality function defined by:

$$Q_{\mathcal{M}}(K) = \frac{36 \sum_{i=1}^6 \ell_{\mathcal{M}}^2(e_i)}{|K|_{\mathcal{M}}^3} \in [1, \infty],$$

where $e_i$ with $i=1,6$ are the edges of element $K$. A perfect element has a quality of 1.

Cavity-based operators. A complete mesh generation or mesh adaptation process usually requires a large number of operators: Delaunay insertion, edge-face-element point insertion, edge collapse, point smoothing, face/edge swaps, etc. Independently of the complexity of the geometry, the more operators are involved in a remeshing process, the less robust the process may become. Consequently, the multiplication of operators implies additional difficulties in maintaining, improving and parallelizing a code. In [65], a unique cavity-based operator has been introduced which embeds all the aforementioned operators. This unique operator is used at each step of the process for surface and volume remeshing.

The cavity-based operator is inspired from incremental Delaunay methods [19, 92, 40] where the current mesh $\mathcal{H}_k$ is modified iteratively through sequences of point insertion. The insertion of a point $P$ can be written:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - C_P + B_P,$$

where, for the Delaunay insertion, the cavity $C_P$ is the set of elements of $\mathcal{H}_k$ such that $P$ is contained in their circumsphere and $B_P$ is the ball of $P$, i.e., the set of new elements having $P$ as vertex. These elements are created by connecting $P$ to the set of the boundary faces of $C_P$.

In [65], each meshing operator is equivalent to a node (re)insertion inside a cavity. For each operator, we just have to define judiciously which node $P$ to (re)insert and which set of volume and surface elements will form the cavity $C$ where point $P$ will be reconnected:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - C + R_P.$$  \hspace{1cm} (31)

Note that if $\mathcal{H}_k$ is a valid mesh (only composed of elements of positive volume) then $\mathcal{H}_{k+1}$ will be valid if and only if $C$ is connected (through internal faces of tetrahedron) and $R_P$ generates only valid elements.
Features of the serial remesher. The use of the previous cavity-based operators allows us to design a remeshing algorithm that has a linear complexity in time with respect to the required work (sum of the number of collapses and insertions). On a typical laptop computer Intel Core i7 at 2.7 GHz, the speed for the (cavity-based) collapse is around 20,000 points removed per second and the speed for the insertion is also around 20,000 points or equivalently 120,000 elements inserted per second. Both estimates hold in an anisotropic context [53]. In addition, the complexity to compress the mesh to remove destroyed entities (points, elements) depends on the number of destroyed entities (and not on the number of current entities). A rule of thumb is then to make sure that each loop used in the remeshing phase has a complexity proportional to the required work rather than a complexity proportional to the size of the current mesh. In many cases, it is sufficient to replace complete loop over the entities with a loop on a front of entities; this front being updated dynamically during the underlying process. If these modifications have a little impact on medium size meshes, they appear to be a drastic bottleneck for very large meshes or when the process is run in parallel with a a priori metric-based static load balancing.

7.2. Solution field interpolation

Nothing special is done when the solution field is interpolated from one mesh to another to be used as restart. Vertices from the new mesh are localized in the elements of the previous mesh and a $P^1$ interpolation based on the barycentric coordinates is applied. More details on the localization can be found in [3].

7.3. Hessian recovery

The above error estimates require to compute second derivatives - Hessian - of some variables. The Hessian at each vertex is computed using the Clément’s $L^2$-projection local operator [23]. From our experience, this method is as accurate than least square type method but a lot more robust when dealing with highly anisotropic meshes. It is a two step process.

First, nodal gradients of the considered variable $u$ are computed exactly in the same way as in the flow solver (see Section 4.1.1):

$$\nabla u(P_i) = \frac{\sum_{j \in P_i} [K_j] |\nabla u|_{K_j}}{\sum_{j \in P_i} |K_j|} ,$$

where $P_i$ is a vertex of the mesh, $|K_j|$ denotes the area/volume of element $K_j$ and the $P^1$-Galerkin gradient at the element $(\nabla u)|_{K_j}$ is given by Relation (11). This gives us a piecewise linear representation of the gradient field of the considered variable.

Second, we re-apply the same operator to each component of the gradient field to obtain components of the Hessian. For instance,

$$u_{xx}(P_i) = \frac{\sum_{j \in P_i} [K_j] |\nabla u_{xx}|_{K_j}}{\sum_{j \in P_i} |K_j|}, \quad u_{xy}(P_i) = \frac{\sum_{j \in P_i} [K_j] |\nabla u_{xy}|_{K_j}}{\sum_{j \in P_i} |K_j|} \ldots$$

The important point here is that cross derivatives are obtained by two different computations and there is no reason that numerically we obtain $u_{xy} = u_{yx}$. Therefore, to enforce the symmetry we take $\frac{1}{2}(u_{xy} + u_{yx})$ for the cross derivatives values.

7.4. Operations on metric fields

The manipulation of metric fields is often omitted but the choice of these operations is crucial to obtain very high anisotropy (ratio up to 1:10,000). We recall quickly two of them that are mandatory.

7.4.1. Metric field interpolation

In practice, the metric field is only known discretely at the background mesh vertices, i.e., the previous adapted mesh on which the metric field has been computed. 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 metric field interpolation is used each time a new vertex is created by the local remesher to set a metric at that vertex. Interpolating metric on the background mesh (and not on the current modified mesh) is crucial to have an accurate representation of the metric field and to preserve the very high anisotropy. Several interpolation schemes have been proposed in [4] which are based on the simultaneous reduction. The main drawback of these approaches is that they do not preserve the very high anisotropy of the metric field. On the other hand, the metric interpolation based on the log-Euclidean framework [10, 52] proves to better behave in this context as demonstrated in [72].

We first define the notion of metric logarithm and exponential:

$$\ln(M) := \mathcal{R} \ln(\Lambda) \mathcal{R} \quad \text{and} \quad \exp(M) := \mathcal{R} \exp(\Lambda) \mathcal{R},$$

where $\ln(\Lambda) = \text{diag}(\ln(\lambda_i))$ and $\exp(\Lambda) = \text{diag}(\exp(\lambda_i))$. We can now define the logarithmic addition $\oplus$ and the logarithmic scalar multiplication $\odot$:

$$M_1 \oplus M_2 := \exp(\ln(M_1) + \ln(M_2)) \quad \text{and} \quad \alpha \odot M := \exp(\alpha \cdot \ln(M)) = M^\alpha .$$

33
The logarithmic addition is commutative and coincides with matrix multiplication whenever the two tensors \( M_1 \) and \( M_2 \) commute in the matrix sense. The space of metric tensors, supplied with the logarithmic addition \( \otimes \) and the logarithmic scalar multiplication \( \circ \) is a vector space. Let \((x_i)_{i=1..k}\) be a set of vertices and \((M(x_i))_{i=1..k}\) their associated metrics. Then, for a point \( x \) of the domain such that:

\[
x = \sum_{i=1}^{k} \alpha_i x_i \quad \text{with} \quad \sum_{i=1}^{k} \alpha_i = 1,
\]

the interpolated metric is defined by:

\[
M(x) = \bigoplus_{i=1}^{k} \alpha_i \otimes M(x_i) = \exp \left( \sum_{i=1}^{k} \alpha_i \ln(M(x_i)) \right).
\]  

This interpolation is commutative. Moreover, it has been demonstrated in [10] that this interpolation preserves the maximum principle, i.e., for an edge \( pq \) with endpoints metrics \( M(p) \) and \( M(q) \) such that \( \det(M(p)) < \det(M(q)) \) then we have \( \det(M(p+t pq)) < \det(M(q)) \) for all \( t \in [0,1] \).

### 7.4.2. Metric field gradation

Metric fields may have huge variations or may be quite irregular when evaluated from numerical solutions that present discontinuities or steep gradients. This makes the generation of a unit mesh difficult or impossible, thus leading to poor quality anisotropic meshes. Generating high-quality anisotropic meshes requires to smooth the metric field by bounding its variations in all directions. It also helps flow solver convergence. In the anisotropic context, the mesh gradation consists in reducing in all directions the size prescribed at any points if the variation of the metric field is larger than a fixed threshold [2]. We adopt the following continuous vision of the anisotropic mesh gradation control. Each point of the domain defines a metric field in the whole domain by growing its metric at a rate given by the desired gradation coefficient. These fields define well-graded smooth continuous meshes over the domain and represent the size constraint imposed by each point in the entire domain. Then, each point of the domain has to take into account these constraints to guarantee a metric field with a smooth variation controlled by the size gradation.

Because the metric gradation reduces the variation (gradient) of the metric field, the complexity of the new graded metric field is always higher than the complexity of the initial metric field. Depending on the gradation coefficient and the property of the original, the complexity can be increased significantly. This is troublesome for convergence study analysis. To avoid this issue, the new graded metric field is rescaled to the original complexity at the end of the metric gradation procedure.

**Spanning a metric field.** Let \( p \) be a point of a domain \( \Omega \) supplied with a metric \( M(p) \) and \( \beta \) the specified gradation. Two laws have been proposed to govern the metric growth in the domain [2]. In the first one, the metric growth is homogeneous in the Euclidean metric field defined by \( M(p) \). As a results, the shape of the metric is kept unchanged while growing. This law conserves the same anisotropic ratio. The second metric growth is homogeneous in the physical space, i.e., the classical Euclidean space. In this case, the eigenvalues are growing separately and differently, the shape of the metric is not more preserved. This law gradually makes the metric more and more isotropic as it gradually propagates in the domain.

From our experience, we suggest to mix these two laws to achieve an efficient metric gradation algorithm. For this new law, a growth factor is associated independently with each eigenvalue of \( M \):

\[
\eta^i_t(px) = \left( 1 + \sqrt{\ell(px) \ln(\beta)} \right) + \left( 1 + \ell(px) \ln(\beta) \right)^{i-1} \quad \text{for} \quad 0 \leq t \leq 1 \quad \text{and} \quad i = 1, \ldots, 3,
\]

where \( \beta \) is the gradation coefficient. The growth metric at \( x \) is given by:

\[
M_p(x) = 3^{RN(px)}R \text{ where } RN(px) = \begin{pmatrix} \eta^1_t(px) & 0 & 0 \\ 0 & \eta^2_t(px) & 0 \\ 0 & 0 & \eta^3_t(px) \end{pmatrix}.
\]  

From the point \( p \), a well-graded smooth continuous mesh is defined all over domain \( \Omega \) by:

\[
(M_p(x))_{x \in \Omega} \text{ with } M_p(x) = 3^{RN(px)}R.
\]

In the numerical examples, we always consider \( t = \frac{1}{8} \).

**Metric reduction.** The reduced metric at a given point \( x \) of domain \( \Omega \) is given by the strongest size constraint imposed by the metric at \( x \) and by the spanned metrics (parametrized by the given size gradation) of all the other points of the domain at \( x \):

\[
\widehat{M}(x) = \bigcap_{p \in \Omega} M_p(x) \cap M(x).
\]

It is done using the metric intersection operator [2, 12].
**Practical implementation.** In the case of a mesh $\mathcal{H}$ supplied with a discrete metric field given at its vertices, each vertex provides a metric for all the other vertices that imposes its size constraints in all directions. The metric reduction is thus performed with all mesh vertices. Unfortunately, we face a quadratic complexity algorithm. To avoid this, the mesh gradation problem is approximated with a linear complexity algorithm based on mesh edges. More precisely, the size correction is performed edge by edge. Let $pq$ be an edge of the mesh with endpoints metrics $M(p)$ and $M(q)$. We define the grown metrics $M_G(p)$ and $M_G(q)$ at both extremities of the edge which are given by Relation (33). Then, the reduction is performed at each vertex by a metric intersection. The information is propagated in the whole domain thanks to an iterative algorithm depicted in Algorithm 2. To accelerate the edge treatment, a dynamic list of the edge to be examined is considered.

It is important to note that the considered edge-based propagation of the size constraints assumes that the size growth law is transitive. It is the case for the linear law on $h$ but not for the other ones. Unfortunately, the size growth of the linear law variation over the domain is too slow. Therefore, it is all the more important to be able to design other laws. To achieve a pseudo-exponential size variation law, the idea is to increase the gradation coefficient $\beta$ while getting farther from a vertex. As the restriction to the $h$-linear size variation law is due to the edge-based algorithm, the pseudo-exponential size variation law is obtained by proposing a modified edge-based algorithm with a dynamic gradation coefficient $\alpha$. The information is then propagated in the whole domain through iterative Algorithm 2. In this algorithm, storing the reduced metric in $M^{\text{new}}$ at each step implies that the size constraint associated with each vertex is propagated layer by layer, e.g. a vertex first impacts is first-order ball, then at iteration two its ball of order two, etc. Consequently, a vertex imposes a size gradation factor of $\beta$ to its ball of order one, $\alpha \beta$ to its ball of order two, ..., $\alpha^n \beta$ to its ball of order $n$, ... In other words, this algorithm simulates an exponential size variation law by applying the $h$-linear size variation law with a dynamic increasing gradation coefficient while gradually getting further from a vertex.

In all the exemples, we set $\alpha = 1.5$ and $\beta = 1.5$. This choice is a good trade off between smoothing the metric field and a quite fast metric size variation (meaning smaller mesh size).

---

**Algorithm 2 Metric Field Size Gradation**

$M^{\text{new}} = M$

While correction = 1

- correction = 0
- Copy $M^{\text{new}}$ current metric field $M$
- Loop over the edge of $\mathcal{H}$

Let $pq$ be the current edge;

1. Grow vertices metric to both edge extremities:
   $M(q) \rightarrow M_G(q)$ and $M(p) \rightarrow M_G(p)$;
2. Apply the reduction process to each vertex:
   $M^{\text{new}}(p) = M^{\text{new}}(p) \cap M_G(p)$ and $M^{\text{new}}(q) = M^{\text{new}}(q) \cap M_G(q)$;
3. If one metric is modified then correction = 1;

EndEdgeLoop

- Increase the size gradation factor to $\beta = \alpha \beta$ with $\alpha > 1$;

EndWhile
8. Drag prediction application

We first illustrate the efficiency of the solution-adaptive process on drag prediction application. Two test cases are considered: the 2D transonic RAE2822 airfoil and the 3D transonic ONERA M6 wing. For each case, we will compare the efficiency of the different error estimates presented in this paper. We will also compare the benefits of using a lower dissipation limiter.

8.1. 2D transonic RAE2822 airfoil

The first drag prediction application is a flow over a transonic 2D RAE2822 airfoil nearing buffet. It is an interesting case because the dominant flow features are a strong adverse pressure gradient, shock/boundary layer interaction, and separation downstream of the shock. The CAD geometry of the RAE2822 airfoil has been reconstructed from the geometric data given in [24]. Experimental measurements can be found in [24]. However, experimental comparison has to be taken with care as the experiment was not axisymmetric (we should run 3D) and no corrections have been made to take into account the walls of the test section of the wind tunnel while CFD is “free air”. Several studies have been done to try to give CFD conditions to match the experiment but it proves to be a very complex problem. A detailed study is done in [33] pointing out these difficulties.

We consider the freestream flow conditions propose by NASA:

<table>
<thead>
<tr>
<th>Mach number</th>
<th>Angle-of-Attack</th>
<th>Reynolds number</th>
<th>Temperature (K)</th>
<th>Reference length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.729</td>
<td>2.31</td>
<td>$6.5 \times 10^6$</td>
<td>300</td>
<td>1</td>
</tr>
</tbody>
</table>

As regards the mesh adaptation, we compare four error estimates:

- the feature-based error estimate controlling the interpolation of the local Mach number in $L^2$-norm and in $L^4$-norm given by Relation (22)
- the laminar goal-oriented error estimate given by Relation (25) with the drag coefficient as output functional
- the new viscous goal-oriented error estimate given by Relation (29) with the drag coefficient as output functional.

For each error estimate we perform a maximum of $n_{adap} = 20$ mesh adaptation iterations at each fixed complexity and for the convergence study we consider eight complexities:

$\{4000, 8000, 16000, 32000, 64000, 128000, 256000, 512000\}$.

We choose $\varepsilon = 0.001$ as threshold to exit the mesh adaptation loop at each complexity. In other word, if the variation of the total drag, pressure drag, viscous drag and total lift is less than 0.1% on three consecutive adapted meshes then we consider the solution converged at that complexity.

We start the convergence study with an initial coarse mesh composed of 9 363 vertices and 17 980 triangles, see Figure 8. This is a coarse inviscid mesh without any boundary layer or any specific refinement for viscous flows, thus very easy and quick to generate. The final adapted mesh size and the total number of adaptation iterations at each complexity obtained with the viscous goal-oriented error estimate are given in Table 1.

![2D RAE2822 airfoil. Initial mesh composed of 9 363 vertices and 17 980 triangles.](image)
Figure 9 shows the solution Mach number field (right) and the adapted mesh (left) obtained with the viscous goal-oriented error estimate for a complexity of 32,000 leading to an adapted mesh composed of 37,961 vertices. All the flow features for this transonic configuration are automatically captured by the error estimate and appropriately meshed by the mesh adaptation process. The boundary layer, the wake and the shock regions are highly resolved and we clearly see the anisotropy of the mesh. A close-up view of the boundary layer - shock interaction region (bottom pictures) is shown to highlight the advantages of using an automatic process to adapt the mesh: the foot of the shock is meshed isotropically at the proper place with a nice size transition toward the boundary layer region, and the mesh adapts accordingly to the evolution of the boundary layer thickness. We observe

7https://www.grc.nasa.gov/WWW/wind/valid/raetaf/raetaf01/raetaf01.html

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Nbr of vertices</th>
<th>Nbr of adap</th>
<th>$C_D$</th>
<th>$C_p$</th>
<th>$C_f$</th>
<th>$C_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>4,939</td>
<td>20</td>
<td>1.214012e-02</td>
<td>6.637217e-03</td>
<td>5.502906e-03</td>
<td>6.968416e-01</td>
</tr>
<tr>
<td>8000</td>
<td>9,511</td>
<td>20</td>
<td>1.207169e-02</td>
<td>6.352798e-03</td>
<td>5.718892e-03</td>
<td>7.219147e-01</td>
</tr>
<tr>
<td>16,000</td>
<td>18,980</td>
<td>9</td>
<td>1.204567e-02</td>
<td>6.298474e-03</td>
<td>5.747194e-03</td>
<td>7.298444e-01</td>
</tr>
<tr>
<td>32,000</td>
<td>37,961</td>
<td>4</td>
<td>1.207148e-02</td>
<td>6.313946e-03</td>
<td>5.757535e-03</td>
<td>7.334015e-01</td>
</tr>
<tr>
<td>64,000</td>
<td>75,491</td>
<td>4</td>
<td>1.207376e-02</td>
<td>6.302692e-03</td>
<td>5.771070e-03</td>
<td>7.348984e-01</td>
</tr>
<tr>
<td>128,000</td>
<td>151,484</td>
<td>6</td>
<td>1.209880e-02</td>
<td>6.316332e-03</td>
<td>5.782471e-03</td>
<td>7.360597e-01</td>
</tr>
<tr>
<td>256,000</td>
<td>300,723</td>
<td>3</td>
<td>1.212413e-02</td>
<td>6.333168e-03</td>
<td>5.790957e-03</td>
<td>7.369866e-01</td>
</tr>
<tr>
<td>512,000</td>
<td>605,665</td>
<td>3</td>
<td>1.214145e-02</td>
<td>6.344023e-03</td>
<td>5.797425e-03</td>
<td>7.374950e-01</td>
</tr>
</tbody>
</table>

Table 1: 2D RAE2822 airfoil. Statistics for the viscous goal-oriented mesh adaptation.

Figure 9: 2D RAE2822 airfoil. Adapted mesh (left) and solution Mach number field (right) obtained with the viscous goal-oriented error estimate for a complexity of 32,000 leading to an adapted mesh composed of 37,961 vertices.
that the mesh is also adapted before the shock above the wing and on the sonic line before the leading edge. These regions are characteristic line of the adjoint solution that are adapted by the error estimate. Note that the process will adapt automatically to any flow conditions.

Let us compare the results obtained with the four error estimates, to this end the mesh convergence of the total drag and all its components, and the total lift are presented in Figure 14. We first notice that both goal-oriented error estimates and the feature-based error estimates in $L^4$-norm are converging toward the same drag and lift values which demonstrates the independence of the solution to the error estimate. It tends to show that we obtain a mesh independent solution. There is a slight discrepancy for the prediction of the feature-based error estimates in $L^2$-norm with a difference of 0.3 drag count on the $C_p$ and 0.2 lift count. We also observe that some error estimates converge faster toward the final target value. The faster the convergence the better (more efficient) the error estimate. We call this property early capturing in the following. Clearly the viscous goal-oriented error estimate is superior to all the other error estimates. The drag prediction is excellent even on the coarse grid, there is only 1 drag count difference between all complexities. Numbers of this convergence for the viscous goal-oriented error estimate are given in Table 1. But, this is a bit of luck because of compensation between the $C_p$ and $C_f$. If we observe the lift, $C_p$ and $C_f$, we still note that it provides more accurate prediction on coarse adapted meshes than the other error estimates. We note that the error estimate of [16] does not give accurate prediction for adapted meshes with less than 40,000 vertices. However, for larger complexities it gives the same results as the new viscous error estimate. This better behavior of the new error estimates on coarse mesh may
come from the fact that it uses a sharper bound of the error and that it involves less interpolation error terms (4 instead of 11+4 in 2D).

In [78], Park and Balan pointed out that controlling the interpolation error in $L^4$-norm is clearly a better choice than the $L^2$-norm. Our results lead to the same conclusion. Even if, both error estimates converge toward similar values, better prediction are obtained on coarse adapted meshes with the $L^4$-norm and at convergence it gives the same prediction as the one with goal-oriented error estimates. The solution is slower to converge with the $L^2$-norm because the boundary layer region is not enough refined on coarse adapted meshes.

If we compare more precisely the viscous goal-oriented error estimate and the feature-based error estimate in $L^4$-norm, we note that there is near one drag count difference in the $C_p$ for adapted meshes between 20 000 vertices and 150 000 vertices while on the last two complexities the feature-based recovers the $C_p$ value of the goal-oriented error estimate. On these intermediate meshes the shock is slightly shifted upfront which explains the difference in $C_p$ and lift predictions. The same shock position is obtained on the last two complexities, see the $C_p$ profile comparison in Figure 11.

Figure 10 compares the $C_p$ profile obtained with the viscous goal-oriented error estimate with the wind tunnel experimental data [24]. As stated above, we do not expect fitting to the experimental data because we do not take into account the wind tunnel walls and 3D effects. A detailed study on the difficulties to match the experimental data is given in [33]. This $C_p$ profile still give a good idea of the obtained results.

For this comparison, the final solution at five different complexities (4 000, 8 000, 16,000, 32 000 and 512 000) are plotted. It points out the convergence of the $C_p$ profile with the mesh size. On the left picture, $C_p$ profile are almost indistinguishable between complexities 16,000, 32 000 and 512 000. A close-up view on the shock region (right picture) shows a slight shift of the shock to the right during the mesh-convergence. As regards the comparison with the experiment data, we obtain a shock position which is slightly shifted to the left like most of the CFD results found in the literature.

In the bottom pictures, we compare the final solution given by the four error estimates. The $C_p$ profile obtained with the two goal-oriented error estimates and the feature-based error estimate in $L^4$-norm are indistinguishable. We note that the shock is slightly shifted to the left for the feature-based error estimate in $L^2$-norm which explains the slight discrepancy in the drag and lift prediction.

We now analyze the convergence of the aerodynamic coefficients for the viscous goal-oriented error estimate throughout the whole mesh-convergence analysis. Figure 15 presents the evolution of the total drag, the pressure drag, the viscous drag and the total lift for each computation in red (i.e., all predictions given by all runs at all complexities) and the final retained value obtained for each complexity in blue. We clearly observe that a lot is done on coarse adapted meshes (which are cheap runs) while almost the minimum number of iterations is done on the finer adapted meshes. Here, three adaptation iterations because we required one thousandth of variation over three consecutive adapted meshes. Thus, converging on coarse adapted meshes is advantageous and enables early capturing of the solution. We also observed that there is almost no variation of the prediction at large complexities showing that the solution-adaptive process is stable. The convergence of the non-linear log-residual of the density variable is shown in Figure 13. The same convergence behavior is observed for all the variables.

Finally, we compare the results obtained using the Piperno limiter [81, 82] and the new limiter proposed in this paper (denoted Gamma). To this end, we analyze the convergence of the aerodynamic coefficients using the viscous goal-oriented error estimate. The results are presented in Figure 12. The advantages of the lower dissipation limiter are clearly visible, in particular on the $C_p$ plots on coarse adapted meshes, thus favoring the early capturing property of the process. Note that both limiters give similar prediction on the finest adapted mesh.
Figure 12: 2D RAE2822 airfoil. Mesh convergence of the total drag $C_D$, the pressure drag $C_p$, the viscous drag $C_f$, and the total lift $C_L$ for the viscous goal-oriented error estimate using the Piperno limiter and the Gamma limiter.

Figure 13: 2D RAE2822 airfoil. Illustration of the non-linear log-residual convergence of the flow solver throughout the solution-adaptive process.
Figure 14: 2D RAE2822 airfoil. Mesh convergence of the total drag $C_D$, the pressure drag $C_p$, the viscous drag $C_f$, and the total lift $C_L$ (from top to bottom) for each error estimate. Left, global view, and right, close-up view for adapted meshes above 10,000 vertices.
Figure 15: 2D RAE2822 airfoil. Mesh convergence history of the total drag $C_D$, the pressure drag $C_p$, the viscous drag $C_f$, and the total lift $C_L$ (from top to bottom) for the viscous goal-oriented error estimate throughout the whole mesh-convergence analysis. In red, the convergence of the aerodynamic coefficient at each complexity and, in blue, the global convergence of the aerodynamic coefficient by retaining the final value for each complexity.
8.2. 3D transonic ONERA M6 wing

The second drag prediction application is the well-known transonic ONERA M6 wing test case. We consider the usual freestream flow conditions for this geometry where a $\lambda$-shock occurs on the upper part of the wing:

<table>
<thead>
<tr>
<th>Mach number</th>
<th>Angle-of-Attack</th>
<th>Reynolds number</th>
<th>Temperature (K)</th>
<th>Reference surface</th>
<th>Reference length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.84</td>
<td>3.06</td>
<td>$14.6 \times 10^6$</td>
<td>300</td>
<td>1.15315</td>
<td>0.801673</td>
</tr>
</tbody>
</table>

We start the convergence study with an initial coarse mesh composed of 19462 vertices and 95593 tetrahedra, see Figure 16. This is a coarse inviscid mesh without any boundary layer or any specific refinement for viscous flows, thus very easy and quick to generate. The surface mesh is just adapted to the geometric property of the wing.

As regards the mesh adaptation, we compare three error estimates:

- the feature-based error estimate controlling the interpolation of the local Mach number in $L^2$-norm and in $L^4$-norm given by Relation (22)

- the new viscous goal-oriented error estimate given by Relation (29) with the drag coefficient as output functional.

For the two feature-based error estimate, we also compare the results obtained with the Piperno limiter and the new limiter described in Section 4.1.1. For each error estimate, we perform a maximum of $n_{adap} = 10$ mesh adaptation iterations at each fixed complexity and we choose $\varepsilon = 0.01$ as threshold to exit the mesh adaptation loop at each complexity. We consider nine complexities for the convergence study:

$$\{10000, 20000, 40000, 80000, 160000, 320000, 640000, 1\,280\,000, 2\,560\,000\}.$$ 

These complexities lead to adapted meshes composed approximately of a number of vertices equal to two times the complexity and a number of tetrahedra equal to twelve times the complexity. For instance, the final adapted mesh obtained with the goal-oriented error estimate at complexity 1 280 000 is composed of 2 493 056 vertices, 14 355 570 tetrahedra, and 338 628 (surface) triangles.

Similar conclusions to the RAE2822 case can be drawn by analyzing the mesh convergence of the aerodynamic coefficients (drag, $C_p$ and $C_f$) given in Figure 18. Thus, we will give a less detailed analysis.

The new limiter outperforms the Piperno limiter on coarse adapted meshes showing the benefits of using less dissipative limiter to obtain accurate prediction on coarse adapted meshes. At convergence similar predictions are obtained.

At convergence, all error estimates give the same predictions which shows the consistency of the solution-adaptive process and the obtention of mesh-independent solutions. Again, the feature-based error estimate using the $L^4$-norm is clearly a better choice than the $L^2$-norm because it quickly captures the boundary layer region. Overall, the viscous goal-oriented error estimate performs better but it requires the computation of the adjoint. Therefore, if the adjoint state is not available, feature-based error estimate using the $L^4$-norm is a good choice for drag prediction.

A frequently asked question is why the goal-oriented error estimate would refine the shock region while the shock does not appear in the adjoint sensitivity. The answer is simply that the gradient of the adjoint is not zero in that region, thus it gives a weight to the error estimate (even if small) which will still ask to refine the shock region. However if we compare the mesh size in the shock region, see Figure 17 for a comparison in the symmetry plane, between the goal-oriented and the feature-based in $L^4$-norm error estimates, we clearly note that this region is a lot less refine using the goal-oriented error estimate, so does the wake region.

Figure 16: 3D ONERA M6 wing. Initial mesh composed of 19462 vertices and 95593 tetrahedra.
The convergence of the drag, $C_p$ and $C_f$ coefficients for the viscous goal-oriented error estimate throughout the whole mesh-convergence analysis is shown in the right column of Figure 18. Red crosses represent all the runs and blue crosses the final prediction at each complexity. Once more, we note that a lot is done on coarse adapted meshes which are inexpensive runs. It is thus interesting to perform a maximum number of adaptations until the convergence of the aerodynamic coefficients. On the contrary, on fine adapted meshes all run provide a similar prediction and we can perform a minimum of number of adaptations. Thus, converging on coarse adapted meshes is advantageous and enables early capturing of the solution.

We now illustrate the convergence of the process for the goal-oriented error estimate. To this end, we consider the final adapted meshes obtained at four different complexities: 20 000, 80 000, 320 000, and 1 280 000. Figures 19, 20, 21, and 22 show the surface adapted mesh on the upper part of the wing, the density field on the wing and the symmetry plane, the volume adapted mesh for the cut plane $y = 0.8$, and the Mach number field in the cut plane $y = 0.8$, respectively.

We note how the adapted meshes are more and more refined when the complexity is increased resulting, for instance, in a drastic reduction of the shocks thickness. The thickness of the wake and the boundary layer is almost the same except for the coarsest adapted mesh composed of only 43 000 vertices. This is also visible on the Mach solution field evolution. The density field shows how the shocks are spread out by the boundary layer pointing out that the shock boundary layer interaction is well-captured. In conclusion, it results in a very accurate solution prediction, and as previously, all the flow features are automatically captured by the solution-adaptive process.

The adapted surface meshes of wing are shown to illustrate the high degree of refinement and anisotropy on the surface of the wing, in particular at the leading edge where the mesh is extremely refined and where a high anisotropy is requested. The largest the complexity the more the anisotropy. Generating such adapted meshes and being able to project vertices on the CAD with this high level of anisotropy requires a very robust process in the adaptive local remesher.

Figure 17: 3D ONERA M6 wing. Adapted meshes in the symmetry plane for the viscous goal-oriented error estimate (left) and the feature-based error estimate in $L^4$-norm (right) for a complexity of 320 000. We note that the goal-oriented mesh adaptation puts less refinement in the shock and wake region.
Figure 18: 3D ONERA M6 wing. Left, for each error estimate, convergence of the total drag $C_D$ (top), the pressure drag $C_P$ (middle), and the viscous drag $C_F$ (bottom). Right, convergence history of the total drag $C_D$ (top), the pressure drag $C_P$ (middle), and the viscous drag $C_F$ (bottom) for the viscous goal-oriented error estimate throughout the whole mesh-convergence analysis. In red, the convergence of the aerodynamic coefficient at each complexity and, in blue, the global convergence of the aerodynamic coefficient by retaining the final value for each complexity.
Figure 19: 3D ONERA M6 wing. Surface adapted mesh on the upper part of the wing obtained with the viscous goal-oriented error estimates. From top to bottom and from left to right, final adapted meshes for complexities 20,000, 80,000, 320,000, and 1,280,000.

Figure 20: 3D ONERA M6 wing. Density field on the wing surface and the symmetry plane obtained with the viscous goal-oriented error estimates. From top to bottom and from left to right, final adapted meshes for complexities 20,000, 80,000, 320,000, and 1,280,000.
Figure 21: 3D ONERA M6 wing. Cut plane ($y = 0.8$) of the volume adapted mesh obtained with the viscous goal-oriented error estimates. From top to bottom and from left to right, final adapted meshes for complexities 20,000, 80,000, 320,000, and 1,280,000.

Figure 22: 3D ONERA M6 wing. Mach number field in the cut plane ($y = 0.8$) of the volume adapted mesh obtained with the viscous goal-oriented error estimates. From top to bottom and from left to right, final adapted meshes for complexities 20,000, 80,000, 320,000, and 1,280,000.
9. High-lift prediction application

In this numerical section, we consider high-lift prediction applications. The first case is a well-known 2D case, a multi-element airfoil, to emphasize which benefits can be expected in three-dimensions. The second case is the NASA High-Lift Common Research Model (HL-CRM) geometry considered during the 3rd AIAA CFD High-Lift Prediction Workshop.

9.1. 2D multi-element airfoil

The first high-lift prediction application is a subsonic 2D multi-element airfoil. We consider the freestream flow conditions close to the one of the 3rd AIAA CFD High-Lift Prediction Workshop:

<table>
<thead>
<tr>
<th>Mach number</th>
<th>Angle-of-Attack</th>
<th>Reynolds number</th>
<th>Temperature (K)</th>
<th>Reference length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.175</td>
<td>16.21</td>
<td>$15.1 \times 10^6$</td>
<td>300</td>
<td>1</td>
</tr>
</tbody>
</table>

As regards the mesh adaptation, we compare four error estimates:

- the feature-based error estimate controlling the interpolation of the local Mach number in $L^2$-norm and in $L^4$-norm given by Relation (22)
- the two presented goal-oriented error estimates: the laminar goal-oriented error estimate given by Relation (25) and the viscous goal-oriented error estimate given by Relation (29).

For each error estimate, we perform a maximum of $n_{adap} = 20$ mesh adaptation iterations and we choose $\epsilon = 0.01$ as threshold to exit the mesh adaptation loop at each complexity. We consider eight complexities for the convergence study:

$$\{4000, 8000, 16000, 32000, 64000, 128000, 256000, 512000\}.$$  

We start the convergence study with an initial coarse mesh composed of 12,263 vertices and 23,950 triangles, see Figure 23. This is a simple inviscid mesh without any boundary layer or any specific refinement for viscous flows, thus very easy and quick to generate.

Figures 24 and 25 show the solution Mach number field and the adapted mesh obtained with the viscous goal-oriented error estimate for a complexity of 64,000 leading to an adapted mesh composed of 74,371 vertices. We observe that the error estimate and the mesh adaptation capture automatically all the flow features of this high-lift configuration:

- wakes of the slat and main wing create shear layers that are are highly resolved and propagated over the airfoil
- wake - boundary layer merging and the flow separation over the flap is properly meshed and capture
- the wake is highly resolved and well propagated in the domain
- separated cove flows are accurately captured which is very important for aero-acoustic applications.

All of this is done automatically by the mesh adaptation process without any human intervention. This clearly demonstrates the benefits in automation of solution-adaptive process and it is the way to remove the human from the loop for the meshing stage which can be source of errors. Note that the process will adapt automatically to any flow conditions which is advantageous for large change in the angle-of-attack for high-lift configuration study.

We now compare the four error estimates presented in this paper, the convergence of the lift and of the drag are presented in Figure 26.

First, for the feature-based error estimates, we observe that initially the flow is fully detached. The flow is detached for meshes with 40,000 vertices or less for the feature-based error estimates in $L^2$-norm (in red), and for meshes with 10,000 vertices or less for the feature-based error estimates in $L^4$-norm (in green). It is interesting to see here that when the mesh size is increased (i.e., more resolution) then the adaptive process is able to re-attach the flow and jump to the correct solution. Even if multiple solutions for the RANS equations exist [44], this is a numerical evidence that the mesh adaptive process can correct itself and jump to another solution when the mesh resolution is increased. This is illustrated in Figure 27 where the final adapted meshes for the feature-based error estimates in $L^2$-norm at complexities 32,000 and 128,000 are shown. Note that the switch from one solution to another is smooth for the laminar goal-oriented error estimate.

Second, we now compare the accuracy of the different error estimates. As pointed out by Park and Balan for drag applications [78], we also observe that controlling the interpolation error in $L^4$-norm (in green) is clearly a better choice than the $L^2$-norm

---

8https://hiliftpw.larc.nasa.gov
Figure 23: 2D multi-element airfoil. Initial mesh composed of 12,263 vertices and 23,950 triangles.

Figure 24: 2D multi-element airfoil. Solution Mach number field (top) and adapted mesh (bottom) obtained with the viscous goal-oriented error estimate for a complexity of 64,000 leading to an adapted mesh composed of 74,371 vertices.
(in red) for high-lift applications. The flow is re-attached faster (on coarser meshes) and the lift converge quicker to the correct value. This is mainly because the feature-based error estimate in $L^4$-norm increases the mesh resolution in the boundary layer region faster than the $L^2$-norm. The result with the laminar goal oriented error estimate (in blue) given by Relation (25) is a bit disappointing. Even if this adaptive process converge toward the correct solution and is better than the feature-based error estimate in $L^2$-norm, this estimate is comparable to the feature-based error estimate in $L^4$-norm and does not bring any benefits. This may be attributed to the over-estimation of the error we make by taking the spectral radius of the Hessian of the adjoint state and by taking the absolute value of each term in the weights. This points out that a sharper error estimate is required. On the other hand, the viscous goal-oriented error estimate (in pink) given by Relation (29) surpasses the other error estimates. The computed flow is immediately attached, even on the coarser adapted mesh composed of around 4,500 vertices. We have an early capturing of the solution and a good estimate of the lift value with only 40,000 vertices. This is clearly the best choice while the feature-based error estimate in $L^2$-norm is an attractive strategy for flow solver without adjoint solution. The same conclusion arises for the drag, see Figure 26 (top right). However, it is important to note that all error estimates converge toward the same lift and drag values (even if it may require larger meshes) which proves that the process is consistent.

Finally, we analyze the convergence of the lift value for the viscous goal-oriented error estimate throughout the whole mesh-convergence analysis. Figure 26 (bottom) presents the evolution of the lift for each computation in red (i.e., each adaptation at each complexity) and the final retained value obtained for each complexity in blue. We clearly observe that a lot is done on coarse adapted meshes (which is cheap) while almost the minimum number of iterations is done on the finer adapted meshes. Thus converging on coarse meshes is advantageous and enable early capturing of the solution.
Figure 26: 2D multi-element airfoil. Top, for each error estimate, convergence of the total lift $C_L$ (left) and the total drag $C_D$ (right). Bottom, convergence history of the total lift value $C_L$ for the viscous goal-oriented error estimate throughout the whole mesh-convergence analysis. In red, the convergence of the total lift at each complexity and, in blue, the global convergence of the total lift by retaining the final lift value for each complexity.

Figure 27: 2D multi-element airfoil. Adapted mesh obtained with the feature-based error estimate in $L^2$-norm for a complexity of 32 000 (left) and a complexity of 128 000 (right).
9.2. High-lift version of NASA’s Common Research Model (HL-CRM) geometry

We present our solution-adaptive computations on the high-lift version of the NASA CRM (HL-CRM) geometry used for the 3rd AIAA CFD High Lift Prediction Workshop [86] (HLPW3). We consider the geometry with the full chord flap gap which corresponds to the case 1b of the workshop with the following flow conditions:

<table>
<thead>
<tr>
<th>Mach number</th>
<th>Angle-of-Attack</th>
<th>Reynolds number</th>
<th>Temperature (K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>16</td>
<td>3.26e6</td>
<td>288.15</td>
</tr>
</tbody>
</table>

The case is considered in SI units. The geometry has been transformed in meters, so in that case the reference length is 7.00532 m and the reference surface is 191.84477 m².

For the mesh adaptation, we have retained the two best error estimates of the previous study:

- the feature-based error estimate controlling the interpolation of the local Mach number in $L^4$-norm
- the viscous goal-oriented error estimate given by Relation (29).

For each error estimate, we perform a maximum of $n_{\text{adap}} = 20$ mesh adaptation iterations at each fixed complexity and we choose $\varepsilon = 0.01$ as threshold to exit the mesh adaptation loop at each complexity. We consider five complexities for the convergence study:

{320 000, 640 000, 1 280 000, 2 560 000, 5 120 000}.

We start the convergence study with an initial mesh composed of 229 263 vertices, 726 852 tetrahedra and 384 868 triangles on the surface, see Figure 28. This is a very coarse inviscid mesh without any boundary layer mesh or any specific refinement for viscous flows, in other words no meshing guidelines, thus very easy and quick to generate. We choose to start from this coarse and clearly unresolved mesh to illustrate the non-dependency of the solution-adaptive process to the initial data.

For the given complexities, we end-up with final adapted meshes of size between 0.7M vertices and 10.2M vertices. We compare the results obtained with the solution-adaptive process to all the results obtained during the workshop on the coarse (8M vertices), the medium (26M vertices), the fine (70M vertices) and the x-fine (206M vertices) meshes. In Figures 31, 32, 33 and 34, we compare the lift and drag prediction results obtained with the solution-adaptive process (blue stars) with respect to all the HLPW3 results (red squares). For the lift prediction with the viscous goal-oriented error estimate (Figures 31 and 32), we observe that a result similar to the fine grid is obtained with a 2.7M vertices adapted mesh, and the 5.3M and 10.2M vertices adapted meshes provides results similar to the x-fine mesh that has been run by only five participants. The drag prediction requires 5.3M and 10.2M vertices adapted meshes to get a good value. In conclusion, we achieve the same accuracy as meshes done with the best practice meshing guidelines but with 20 times less vertices. For the lift prediction with the feature-based error estimate (Figures 33 and 34), we observe that we obtain a similar prediction as the workshop fine mesh with 10.2M vertices adapted mesh. The feature-based is clearly less efficient than the goal-oriented in this case but similar results as best practice meshes is obtained with 7 times less vertices.

The convergence of the lift value for the viscous goal-oriented error estimate throughout the whole mesh-convergence analysis is shown in Figure 29 (left). It shows the evolution of the lift for each computation in red (i.e., each adaptation at each complexity) and the final retained value obtained for each complexity in blue. Like in 2D, we note that a lot is done on coarse adapted meshes

Figure 28: HL-CRM case. Initial coarse and clearly unresolved mesh only composed of 229 263 vertices, 726 852 tetrahedra and 384 868 triangles on the surface.

52
(which is cheap) while the minimum number of iterations is done on the finer adapted meshes. Again, converging on coarse meshes is advantageous and enable early capturing of the solution. Figure 29 (right) compares the convergence of the lift for the solution-adaptive process (red and blue lines) with respect to all workshop entries (green lines). This again emphasizes the early capturing of the lift value on coarse adapted meshes in comparison to the meshes used for the workshop.

Now, to illustrate the differences between the workshop best practice meshes and the obtained adapted meshes, we show comparisons of cuts in the medium and fine meshes of the high-lift prediction workshop and the 5M vertices adapted mesh obtained with the viscous goal-oriented error estimate. We also show the local Mach number of the solution for each of these cuts to display - like in 2D - all the high-lift flow features that are automatically captured by the solution-adaptive process.

Figure 30 displays a global view of the HL-CRM with a cut plane at $x = 50$ to emphasize the wake refinement. The medium and fine meshes of the workshop are refined isotropically in a large rectangular region hoping that the wake will be there for all angles of attack. The adapted mesh focus only in the current wake and refine that region appropriately and anisotropically. For instance, we can clearly see the refinement of the four tip vortices coming from the main wing and the two flaps.

Figures 35 and 36 exhibit close-up views of the slat and the flap for the cut plane $y = 15.5$. It points out the difficulty for classical meshing methods to mesh complex geometric details, here a small gap in the geometry where two boundary layer meshes collide. We clearly see the poor mesh quality between the two colliding boundary layer meshes and the over-connected big elements that join them. We observe that the adapted mesh focuses in the boundary layer which is very thin, the wakes of the slat and main wing that create shear layers, the wake - boundary layer merging and the flow separation over the flap, and also the separated cove flows. It is evident that meshing guidelines ask for a large number of layers in the boundary layer mesh to have some refinement far from the body in order to capture some of the shear layer. But, a shear layer further above the body will be prohibitive to capture with such strategy.

Figures 37 and 38 show two details of the geometry that may impact the overall flow: the tip vortex emitted by the slat and the gap between the two flaps. First, in Figure 37, we observe that the adapted mesh refine a lot more the surface mesh and this mesh is highly anisotropic. Moreover, ridges of the geometry are a lot more refine because they can be at the intersection of two boundary layer or source of detached vortices. We can also see in the adapted mesh and in the solution the wing tip vortex of the slat that interacts with the boundary layer of the main wing and detach the flow there. Similarly, in Figure 38, we accurately capture the vortex that runs over the gap and that completely detach the flow in that region. The workshop meshes are very coarse in that region, and for sure do not capture that physical phenomenon, thus it is not clear if the partially sealed gap effect has been correctly accounted during the workshop.

In conclusion, the more complex the geometry the more efficient should be the adaptive process as it able to automatically and accurately capture all the physical features associated with these geometry details. Moreover, these geometric details are very hard to mesh accurately with classical mesh generation methods. Again, this clearly demonstrates the benefits in automation of mesh-adaptive solution platform and it is the way to remove the human from the loop for the meshing stage which can be source of errors. Note that the same results and conclusions are obtained for the 8° angle of attack case.
Figure 30: HL-CRM 16° case. Comparison of the meshes for the cut plane $x = 50$. Top, the workshop medium (left) and fine (right) meshes. Bottom, the 5M vertices adapted mesh obtained with the viscous goal-oriented error estimate (left) and the associated local Mach number solution (right).
Figure 31: HL-CRM 16° case. Comparison of the lift prediction results obtained with the mesh-adaptive solution platform using the viscous goal-oriented error estimate (blue stars) with respect to all the HLFW3 results (red squares) obtained on the coarse (top left), medium (top right), fine (bottom left) and x-fine (bottom right) meshes.
Figure 32: HL-CRM 16° case. Comparison of the drag prediction results obtained with the mesh-adaptive solution platform using the viscous goal-oriented error estimate (blue stars) with respect to all the HL-PW3 results (red squares) obtained on the coarse (top left), medium (top right), fine (bottom left) and x-fine (bottom right) meshes.
Figure 33: HL-CRM 16° case. Comparison of the lift prediction results obtained with the mesh-adaptive solution platform using the feature-based ($L^4$-norm) error estimate (blue stars) with respect to all the HLPW3 results (red squares) obtained on the coarse (top left), medium (top right), fine (bottom left) and x-fine (bottom right) meshes.
Figure 34: HL-CRM 16° case. Comparison of the drag prediction results obtained with the mesh-adaptive solution platform using the feature-based ($L^1$-norm) error estimate (blue stars) with respect to all the HLPW3 results (red squares) obtained on the coarse (top left), medium (top right), fine (bottom left) and x-fine (bottom right) meshes.
Figure 35: HL-CRM 16° case. Comparison of the meshes for the cut plane \( y = 15.5 \) (near the flap). Top, the workshop medium (left) and fine (right) meshes. Bottom, the 5M vertices adapted mesh obtained with the viscous goal-oriented error estimate (left) and the associated local Mach number solution (right).

Figure 36: HL-CRM 16° case. Comparison of the meshes for the cut plane \( y = 15.5 \) (near the slat). Top, the workshop medium (left) and fine (right) meshes. Bottom, the 5M vertices adapted mesh obtained with the viscous goal-oriented error estimate (left) and the associated local Mach number solution (right).
Figure 37: HL-CRM 16° case. Comparison of the meshes for a cut plane in the region where the slat tip vortex interact with the main wing. Top, the workshop medium (left) and fine (right) meshes. Bottom, the 5M vertices adapted mesh obtained with the viscous goal-oriented error estimate (left) and the associated local Mach number solution (right).

Figure 38: HL-CRM 16° case. Comparison of the meshes for the cut plane $x = 38$ (near the gap between the flaps). Top, the workshop medium (left) and fine (right) meshes. Bottom, the 5M vertices adapted mesh obtained with the viscous goal-oriented error estimate (left) and the associated local Mach number solution (right).
10. Conclusion and Future Works

This paper has challenged some established principles for CFD in aeronautic and aerospace, and shown that they were based on bad arguments. It has demonstrated that high-fidelity predictions can be obtained with unstructured meshes composed only of tetrahedra. It is not absolutely necessary to generate boundary layer meshes composed of prisms and / or hexahedra. The key point is to use unstructured anisotropic mesh adaptation where the error in the numerical solution is controlled by an error estimate. For the solution-adaptive process, it has been demonstrated that the final results are independent of the initial mesh, the process can correct itself to converge toward the correct solution, different error estimates and different numerical schemes converge toward the same solution, and that we can achieve mesh-converge results even in 3D for complex flows. Thanks to mesh adaptation the meshing process is greatly simplified, optimal and fully automatic.

In this regards, this paper brought several novelties and major improvements in the design of solution-adaptive process for the RANS equations.

On the solver side, we have presented a mixed Finite Element - Finite Volume formulation which is robust and accurate on highly anisotropic adapted meshes. We have proposed a new limiter for the V4 numerical scheme which proved to be a lot less dissipative than previous limiters. We have pointed out the importance of exact differentiation for the solver convergence when an implicit scheme is designed. And, we have proposed an efficient strategy to solve the linear system at each iteration which is based on Symmetric Gauss-Seidel (SGS) relaxation and an automatic CFL law control. A new residual computation has been proposed to better perceive the convergence of the flow solver on anisotropic adapted meshes.

The adjoint problem turns out to be a stiff problem for RANS such that a GMRES solver with a weak preconditioner was not able to converge whatever the Krylov space size. To solve this issue, we proposed to use a preconditioner based on SGS relaxation which proved to be efficient if a large number of sweeps is performed.

On the mesh adaptation side, we have recalled all the ingredients that are necessary to set-up an efficient solution adaptive platform. We have presented a new goal-oriented error estimate which is sharper than the previous one presented in [16]. It relies on a control of the interpolation error of the conservative variables weighted by a function of the state and the derivatives of the adjoint state.

We have also proposed mesh adaptation algorithms enabling mesh convergence study. In these algorithms, previous solution and CFL fields are interpolated and used as restart to speed-up the overall process and make it efficient.

This solution-adaptive process has been successfully applied to drag and high-lift prediction applications in 2D and 3D. All the presented error estimates have been compared. The feature-based error estimate in $L^2$-norm has proved to be a better choice than the $L^4$-norm. It is the same conclusion as the one of Park and Balan on drag prediction applications. For goal-oriented error estimates, the new version given in this paper is clearly more efficient than the first laminar version. Overall the viscous goal-oriented error estimate is a better choice but it requires a strong adjoint solver as the one presented in this work. If no adjoint is available, then the feature-based error estimate in $L^2$-norm is a good choice. We have also shown the benefits of converging the process on coarse adapted meshes for accurate prediction even on very coarse meshes. It enables early capturing of the solution.

We have also demonstrated that significant gain can be achieved on complex 3D configurations such as the high-lift version of NASA’s Common Research Model (HL-CRM) proposed in the 3rd AIAA CFD High Lift Prediction Workshop. As regards the accuracy of the obtained results with the mesh adaptation process, the benefits are clear using the goal-oriented error estimate. The lift prediction agrees with the ones obtained during the workshop on the x-fine meshes composed of 206M vertices but here a mesh size is between 5M and 10M vertices to reach that accuracy, i.e., the size of the workshop coarse mesh ! For the feature-based error estimate, the prediction is a bit lower and corresponds to the prediction on the fine workshop mesh. And we have revealed that it also captures accurately all flow features and importantly all flow features that are created by geometric details. Thus the more complex the geometry the more efficient should be the process.

As regards the perspective of this work, several improvements are envisioned to make the process even better.

On the flow solver side, the key is the flow solver convergence as stated in the NASA report "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences" [88]. An improved differentiation of the second order convective fluxes and the limiter should help, and it has already been demonstrated, considering a tight coupling between the Navier-Stokes equations and the turbulence model should improve the convergence of the flow solver [73].

At present, we don’t use restart for the adjoint problem which makes it time consuming on very fine adapted meshes. We loose all the benefits of the previous work. Reformulating the adjoint problem as a pseudo-transient one should solve this issue.

For the error estimate part, we aim at adding the turbulence model in the goal-oriented error estimate which requires a tight coupling between the Navier-Stokes equations and the turbulence model. And, an improved error estimate can be designed based on the non-linear corrector proposed in [32].

Seeing the efficiency of the actual process, the extension to URANS can be foreseen following the time-accurate mesh adaption process presented in [9]. In a longer term, we need to extend this work to higher-order flow solver and to tackle higher physical fidelity with DES or LES problems.
Acknowledgements

The authors want to acknowledge Adrien Loseille for providing software AMG/Fefflo.a.

This work was partially supported by funding from Safran Tech with technical monitors Frédéric Feyel and Vincent Brunet and The Boeing Company with technical monitor Todd Michal.

References


62


63
Appendix A. New limiter for the V4-scheme

In the edge-based numerical scheme for the convective terms presented in Section 4.1, we bring us back to local 1D problem (edge by edge) where an approximate Riemann problem is solved. To increase the spatial accuracy, extrapolated values are considered at the interface between two finite volume cells using a MUSCL like approach. This scheme is not monotone and can be source of spurious oscillations especially in the vicinity of discontinuities. To guarantee the TVD or the LED properties, limiting functions are used during the extrapolation.

We recall the notations for a MUSCL scheme in 1D. We consider a uniform discretization of the segment [0, L] in N intervals of length Δx (L = NΔx). We denote by \{x_i\}_{i=0,N} the set of vertices of the 1D mesh, thus x_{i+1} − x_i = Δx, \forall i. For a centered scheme, the extrapolated value is given by:

\[ u_{i+\frac{1}{2}} = u_i + \frac{1}{2}(\nabla u_i)^C \cdot (x_{i+1} - x_i) = u_i + \frac{1}{2}(u_{i+1} - u_i) = u_i + \frac{1}{2}Δ^C. \]

If we consider a convection from left to right, then an extrapolated value for an upwind scheme is:

\[ u_{i+\frac{1}{2}} = u_i + \frac{1}{2}(\nabla u_i)^U \cdot (x_{i+1} - x_i) = u_i + \frac{1}{2}(u_i - u_{i-1}) = u_i + \frac{1}{2}Δ^U. \]

To guarantee the TVD property of the 1D scheme, a limiting function ψ is introduced to maintain the monotonicity:

\[ u_{i+\frac{1}{2}} = u_i + \frac{1}{2}ψ(R)(u_i - u_{i-1}) = u_i + \frac{1}{2}ψ(R)Δ^U \quad \text{with} \quad R = \frac{u_{i+1} - u_i}{u_i - u_{i-1}} = \frac{Δ^C}{Δ^U} \quad \text{and} \quad ψ(R) = 0 \quad \text{if} \quad R < 0. \]

Famous limiters are the MinMod limiter ψ_{mm} and the van Albada limiter ψ_{va}:

\[ ψ_{mm}(R) = \min(1,R) \quad \text{and} \quad ψ_{va}(R) = \frac{R^2 + R}{R^2 + 1}. \]

Appendix A.1. Limiter for β-scheme

β-scheme controls the upwinding through parameter β. In that case, the pseudo-gradient is given by:

\[ Δ^β = (1 - β)Δ^C + βΔ^D. \]

The β-scheme is spatially at least second order accurate. For β = 0, we obtain a centered scheme, and for β = 1 a fully upwind scheme. For β = \frac{1}{2}, we have a half-upwind scheme. And, for β = \frac{1}{3}, we have a third order scheme in the case of the linear advection [45]. For more complex equations, like the compressible Euler equations, it provides a low dissipation (a 4th order dissipation) second order scheme. This is the V4-scheme considered in this work. The main difficulty with that scheme is the construction of appropriate limiters because now we have three pseudo-gradients in action: Δ^C, Δ^U, Δ^V with Δ^V values which are always between the values of Δ^C and Δ^U. Therefore, a MinMod like limiter will never choose the V4 extrapolation.

Koren [45] has proposed (it was also considered by Dervieux [26]) a compressive limiter with three entries (derived from the Superbee) for the V4-scheme:

\[ ψ_{3e}(R) = \min \left( 2, 2R, \frac{1}{3} + \frac{2}{3}R \right). \]

In that case, if the V4 pseudo-gradient is lower than two times the centered and the upwind pseudo-gradient, then it will be used for the extrapolation. The problem with this limiter is that it is derived from the SuperBee limiter and it is only C^0 continuous. Thus, it is problematic to achieve the steady state convergence. Piperno has proposed a method to extend smooth limiter such as Van Albada limiter to β-scheme.

The Piperno limiter

We recall that \( R = Δ^C / Δ^U \) and \( Δ^{Llim} = ψ(R)Δ^U \). Piperno states five criteria for the construction of limiter for upwind scheme [81, 82]. In particular (the 4th and 5th criteria), Piperno reminds us that, when advecting a square wave, the 0-scheme (centered) oscillate for \( R = +∞ \) but not for \( R = 0 \) thus we must have \( ψ(R)/R → 1 \) as \( R → 0^+ \) (as \( ψ(R)/R → 0 \) as \( R → +∞ \)), and the 1-scheme (fully upwind) oscillate for \( R = 0 \) but not for \( R = +∞ \) thus we must have \( ψ(R) → 1 \) as \( R → +∞ \). This is exactly what does the MinMod limiter but for \( R \) lower or greater than 1.

Therefore, the idea of the Piperno limiter is to: i) use the low dissipation pseudo-gradient when \( R ≈ 1 \), ii) converge toward the centered gradient when \( R → 0 \), and iii) converge toward the upwind gradient when \( R → +∞ \). This is equivalent to the properties:

\[ ψ(R) → 1 \quad \text{when} \quad R → +∞ \quad \text{and} \quad ψ(R)/R → 1 \quad \text{when} \quad R → 0. \]
Moreover, we want a smooth limiter for convergence to steady state (the 2nd criteria), to this end we enforce a $C^2$ continuity. To obtain the Piperno limiter, it is interesting to rewrite the limiting function for the V4-scheme under this form:

$$\psi(R) = \left(\frac{1}{3} + \frac{2}{3} R\right) \cdot \psi^*(R),$$

as we can clearly see the region where the pseudo-gradient is low dissipative, this is when $\psi^*(R) = 1$. For the following, it is also interesting to perform the mapping $r = 1/R = \Delta^U / \Delta^C$ and rewrite it as:

$$\phi(r) = \left(\frac{1}{3} + \frac{2}{3} r\right) \cdot \psi^*(r).$$

According to [81, 82], for $R \in [1, \infty]$ or $r \in [0, 1]$, $\psi^*(r)$ should verify the following conditions:

- the first condition is that we want the low dissipative pseudo-gradient for $r = R = 1$ leading to $\psi^*(R = 1) = \psi^*(r = 1) = 1$,
- the second condition is that we want the $C^1$ continuity (the continuity of the first derivative), thus for $r = R = 1$ we should have $(\psi^*)'(R = 1) = (\psi^*)'(r = 1) = 0$ (because we have an extrema here),
- the third condition is $\psi^*(R) \rightarrow 1$ when $R \rightarrow +\infty$ or equivalently $\phi(r = 0) = 1$. Thus, when $r = 0$ we have $\psi^*(r = 0) = \frac{3r}{r + 2} = 0$,
- the fourth condition is that the curve is flat at infinity thus $\psi'(R) \rightarrow 0$ when $R \rightarrow +\infty$ or equivalently $\phi'(r = 0) = 0$. Thus, we have
  $$-\frac{2}{3r^2} \psi^*(0) + \frac{r + 2}{3r} (\psi^*)'(0) = 0 \quad \Leftrightarrow \quad (\psi^*)'(0) = \frac{2}{3r^2} \frac{3r}{r + 2} \frac{3r}{(r + 2)^2} = \frac{3}{2} \quad \text{as} \quad r = 0$$
- The fifth condition is to have a $C^2$ continuity and a plateau for $r = R = 1$ (if we want to extend the high-order gradient region (see the new limiter) or to be almost flat near the high-order gradient region) which translates in $(\psi^*)''(1) = 0$.

To sum up, we want to find a polynomial $\psi^*(r)$ verifying:

$$\psi^*(0) = 0, \quad \psi^*(1) = 1, \quad (\psi^*)'(0) = \frac{3}{2}, \quad (\psi^*)'(1) = 0, \quad (\psi^*)''(1) = 0$$

leading to

$$\psi^*(r) = 1 + \frac{3}{2} r + (r - 1)^3 \quad \Rightarrow \quad \psi^*(R) = 1 + \frac{3}{2} R + (\frac{1}{R} - 1)^3$$

For $R \in [0, 1]$ or $r \in [1, \infty]$, the trick of Piperno was to propose a parametrized rational polynomial which verifies three of the mandatory conditions whatever the parameters defining the rational polynomial:

- the first condition is the same as above,
- the second condition is the same as above,
- the third condition is to have a $C^2$ continuity and a plateau for $r = R = 1$, i.e., the fifth condition as above
- the fourth condition is that for $r \rightarrow 0$ we have $\psi(R) / R \rightarrow 1 \quad \Leftrightarrow \quad \left(\frac{1}{3} R + \frac{2}{3}\right) \psi^*(R) \rightarrow 1$. In other word, we want for
  $$r \rightarrow \infty \quad \text{to have} \quad \left(\frac{1}{3} r + \frac{2}{3}\right) \psi^*(r) \rightarrow 1 \quad \Leftrightarrow \quad \psi^*(r) \rightarrow 3/r.$$

To this end, they proposed the following general rational function:

$$\phi^*(r) = \frac{3r^2 + ar + b}{(r - 1)^3 + 3r^2 + ar + b} = \frac{P(r)}{Q(r) + P(r)}$$

$$\Rightarrow (\phi^*)'(r) = \frac{P'(r)Q(r) - P(r)Q'(r)}{(Q(r) + P(r))^2},$$

$$\Rightarrow (\phi^*)''(r) = \frac{(P'(r)Q(r) - P(r)Q'(r))(P(r) + Q(r)) + 2(P(r)Q'(r) - P'(r)Q(r))(P'(r) + Q'(r))}{(Q(r) + P(r))^3}.$$

66
The trick is that for \( r = 1 \) the cubic term \( Q(r) \), its derivative \( Q'(r) \) and its second derivative \( Q''(r) \) cancel, thus the three first conditions are always satisfied whatever the parameters \( a \) and \( b \). And, the last trick is that when \( r \to \infty \) the numerator leading term is \( 3r^2 \) and the denominator leading term is \( r^3 \) and thus \( \phi(r) \to 3/r \) satisfying the fourth condition.

The parameters \( a \) and \( b \) have been found to maximize the CFL condition for explicit scheme (the condition on \( M \) in the paper), they found \( a = -6 \) and \( b = 19 \).

Finally, the Piperno limiter reads:

\[
\psi_{\text{P}}(R) = \left( \frac{1}{3} + \frac{2}{3}R \right) \left\{ \begin{array}{ll}
\frac{3 \frac{1}{R} - 6 \frac{1}{R} + 19}{(\frac{1}{R} - 1)^3 + 3 \frac{1}{R}^2 - 6 \frac{1}{R} + 19} = \frac{3 \frac{1}{R} - 6 \frac{1}{R} + 19}{\frac{1}{R}^3 - 3 \frac{1}{R} + 18} & \text{if } R < 1 \\
1 + (\frac{1}{2} \frac{1}{R} + 1)(\frac{1}{R} - 1)^3 & \text{if } R \geq 1.
\end{array} \right.
\]

We notice that the Koren limiter converges toward 2 instead of 1 when \( R \to 0 \) or \( R \to +\infty \) thus leading to under or over shoot near discontinuities, see Figure A.39. We also notice that the high-order gradient has a wider region for the Koren limiter than the Piperno one, see Figure A.39. An improvement of the Piperno limiter would be to enlarge this region, because we should have a smooth solution for \( R \sim 1 \) and discontinuous solution for \( R \to 0 \) and \( R \to +\infty \).

**Appendix A.2. New limiter for \( \beta \)-scheme**

To enlarge the region where the low dissipation pseudo-gradient is used, we proposed the following modifications of the Piperno limiter. We modify it such that the low dissipation pseudo-gradient is used between \( R = \frac{1}{k} \) and \( R = k \) this means a ratio of two between the centered and the upwind gradient (note that this ratio is \( 5/2 \) for the Koren limiter).

For \( R < \frac{1}{k} \), the use the same idea as for the Piperno limiter \( i.e., \) we want the cubic term to cancel for \( R = \frac{1}{k} \) or \( r = 2 \) thus the cubic term becomes \((r - 2)^3\). We keep \( P(r) \) unchanged.

For \( R > 2 \), we just shift the polynomial curve by 1 \( i.e., \) we use the mapping \( t = R + 1 \) which gives \( \frac{1}{R} = \frac{1}{r-1} \).

The new limiter reads:

\[
\psi_{\beta}(R) = \left( \frac{1}{3} + \frac{2}{3}R \right) \left\{ \begin{array}{ll}
\frac{3 \frac{1}{R} - 6 \frac{1}{R} + 19}{(\frac{1}{R} - 2)^3 + 3 \frac{1}{R}^2 - 6 \frac{1}{R} + 19} = \frac{3 \frac{1}{R} - 6 \frac{1}{R} + 19}{\frac{1}{R}^3 - 3 \frac{1}{R} + 11} & \text{if } R < \frac{1}{2} \\
1 + (\frac{3}{2} \frac{1}{R - 1} + 1)(\frac{1}{R - 1} - 1)^3 & \text{if } \frac{1}{2} \leq R \leq k \\
1 + (\frac{3}{2} \frac{1}{R - k} + 1)(\frac{1}{R - k} - 1)^3 & \text{if } R \geq k
\end{array} \right.
\]

In fact, we can parametrize this limiter like the Koren limiter to enforce the low dissipation pseudo-gradient in the region between \( R = \frac{1}{k} \) and \( R = k \). This may be very interesting for viscous flows and a nice feature is that for \( k = 1 \), it degenerates to the Piperno limiter.

For \( R \leq \frac{1}{k} \), we simply want the cubic term to cancel for \( R = \frac{1}{k} \) or \( r = k \) thus the cubic term becomes \((r - k)^3\).

For \( R \geq k \), we shift the curve by \( k - 1, i.e., t = R + k - 1 \). It is sufficient to replace the term \( \frac{1}{R} = r \) by \( \frac{1}{R + k - 1} = \frac{r}{r(k-1)r} \) to shift the curve.

The parametrizable new limiter reads:

\[
\psi_{\beta}(R) = \left( \frac{1}{3} + \frac{2}{3}R \right) \left\{ \begin{array}{ll}
\frac{3 \frac{1}{R} - 6 \frac{1}{R} + 19}{(\frac{1}{R} - 2)^3 + 3 \frac{1}{R}^2 - 6 \frac{1}{R} + 19} = \frac{3 \frac{1}{R} - 6 \frac{1}{R} + 19}{\frac{1}{R}^3 - 3 \frac{1}{R} + 11} & \text{if } R < \frac{1}{k} \\
1 + (\frac{3}{2} \frac{1}{R - k} + 1)(\frac{1}{R - k} - 1)^3 & \text{if } \frac{1}{k} \leq R \leq k \\
1 + (\frac{3}{2} \frac{1}{R - k} + 1)(\frac{1}{R - k} - 1)^3 & \text{if } R \geq k
\end{array} \right.
\]

or

\[
\psi_{\beta}(R) = \left( \frac{1}{3} + \frac{2}{3} R \right) \left\{ \begin{array}{ll}
1 + (\frac{3}{2} \frac{1}{R - (k-1)r} + 1)(\frac{r}{r - (k-1)r} - 1)^3 & \text{if } r < \frac{1}{k} \\
\frac{3r^2 - 6r + 19}{(r - k)^3 + 3r^2 - 6r + 19} & \text{if } \frac{1}{k} \leq r \leq k \\
1 & \text{if } r \geq k
\end{array} \right.
\]
Appendix A.3. Illustration of the limiters

We illustrate the behavior of the different limiting functions given above in Figure A.39 according the Sweby diagram [91]. The admissible limiter region for second-order TVD schemes is the region between the MinMod limiter and the SuperBee limiter (top left plot). We have also plotted the well-known van Alberda limiter.

We observe that the Koren limiter behaves like the SuperBee limiter when \( R < \frac{1}{4} \) or \( R > \frac{5}{2} \) (top right and bottom left pictures), and consider the low dissipation pseudo-gradient in between (bottom right picture). The Koren limiter converges toward 2 instead of 1 when \( R \to 0 \) or \( R \to +\infty \) violating two of the Piperno’s criteria. We also notice that the Koren limiter function is not smooth for \( R = \frac{1}{4} \) and \( R = \frac{5}{2} \).

The Piperno limiter behaves more like the van Alberda limiter, both curves are very close (top left picture). In comparison to the Koren limiter, we note that the \( R \)-region where the low dissipation pseudo-gradient is used is very thin (bottom left pictures). This leads us to think that this limiter will not exploit to much the low dissipation of the V4-scheme.

On the contrary, we observe that the new limiter has all the good properties: convergence toward 1 when \( R \to 0 \) or \( R \to +\infty \), the function is smooth everywhere and a large \( R \)-region where the low dissipation pseudo-gradient is used. Moreover, it is parametrizable and the user can control the size of the \( R \)-region where the low dissipation pseudo-gradient is used.

Figure A.39: Top, classical plots of flux limiter functions \( \phi(R) \) for several limiters. On this plot, the second-order TVD region lies between the MinMod and the SuperBee limiters. Bottom, plots of the function \( \frac{\phi(R)}{R} \) and \( \phi^+(R) \) illustrating the properties of each limiting function.

Appendix A.4. Accuracy analysis on a 2D CFD exemple

We consider a 2D turbulent simulations on an adapted (coarse) mesh. It is a RANS high-lift configuration at Mach 0.125, an angle of attack of 16.21° and a Reynolds number of 15.1 million. The geometry is the three-elements airfoil. The (coarse) adapted mesh is composed of 12229 vertices and 22048 triangles. This case is interesting because the boundary layer is not fully resolved and a more dissipative numerical scheme will obtain a detached flow while a less dissipative one should be able to re-attach the flow.

We compare the V3-scheme with the MinMod and the van Alberda limiters, the V4-scheme with the Piperno and the new limiters, and the V4-scheme without limiter (considered as the reference solution). Figure A.41 compares the \( C_p \) curves on the
slat, flap and main wings for these five choices. The $C_p$ curves clearly demonstrate that the V3-scheme is a lot less accurate than the V4-scheme. We also observe the huge benefits in using a less dissipative limiter MinMod vs VanAlbada, and Piperno vs. new limiter. This ascertainment is verified for the overall solution in Figure A.40 where the momentum solution is displayed. The more attached is the flow the more accurate is the numerical scheme. We notice that the V4-scheme with the Piperno limiter is doing better than the V3-scheme with the van Albada limiter. We also notice that the V4-scheme with new limiter provides a solution which is very close to the V4-scheme without limiter. This results is very important because in 3D, for complex geometry, it is almost impossible to run a case without limiter. Thus, low dissipative limiters are of utmost importance and source of huge savings.

**Figure A.40:** Three-elements airfoil in high-lift configuration. Comparison of the momentum solution for different schemes and limiter functions on a relatively coarse adapted mesh composed of 12,229 vertices. From top to bottom and left to right, we have: V3-scheme with MinMod limiter, V3-scheme with van Albada limiter, V4-scheme with Piperno limiter, V4-scheme with new limiter, and the V4-scheme without limiter.
Appendix B. Differentiation of the convective fluxes

For error estimate (29), we need the differentiation of the convective fluxes vector $F^E(W)$ w.r.t. the conservative variable vector $W$. In 2D, denoting $q^2 = u^2 + v^2$, we have the following relations:

$$
\frac{\partial F^E_1(W)}{\partial W} = \begin{pmatrix}
\frac{(y-1)}{2}q^2 - u^2 & -(y-3)u & -(y-1)v & 0 & 0 \\
-uv & v & u & 0 & 0 \\
(u-1)uv - yuE & yE - \frac{y-1}{2} (2u^2 + q^2) & -(y-1)uv & yu & 0
\end{pmatrix},
$$

$$
\frac{\partial F^E_2(W)}{\partial W} = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 \\
-uv & v & u & 0 & 0 \\
(y-1)uv - yuE & yE - \frac{y-1}{2} (2v^2 + q^2) & -(y-1)uv & yu & 0
\end{pmatrix},
$$

In 3D, denoting $q^2 = u^2 + v^2 + w^2$, we have the following relations:

$$
\frac{\partial F^E_1(W)}{\partial W} = \begin{pmatrix}
\frac{(y-1)}{2}q^2 - u^2 & -(y-3)u & -(y-1)v & 0 & 0 \\
-uv & v & u & 0 & 0 \\
-wu & w & 0 & u & 0 \\
(y-1)uv - yuE & yE - \frac{y-1}{2} (2u^2 + q^2) & -(y-1)uv & yu & 0
\end{pmatrix},
$$

$$
\frac{\partial F^E_2(W)}{\partial W} = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 \\
-uv & v & u & 0 & 0 \\
0 & w & v & 0 & 0 \\
(y-1)uv - yuE & yE - \frac{y-1}{2} (2v^2 + q^2) & -(y-1)uv & yu & 0
\end{pmatrix},
$$

$$
\frac{\partial F^E_3(W)}{\partial W} = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 \\
-uv & w & 0 & u & 0 \\
-vw & 0 & w & v & 0 \\
(y-1)uv - yuE & yE - \frac{y-1}{2} (2w^2 + q^2) & -(y-1)uv & yu & 0
\end{pmatrix}.
$$
Appendix C. Proof of the viscous weights in error estimate (29): analysis in conservative variables

In this section, we demonstrate how to obtain the viscous weights of Section 6.3. We seek for the detailed expression of the term:

$$\sum_{i,j} \frac{\partial F^V_i}{\partial x_j}\nabla_{x_i,x_j} W^r.$$ 

in Relation (29).

Appendix C.1. 2D error analysis in conservative variables ($\rho, \rho u, \rho v, \rho E$)

We start from:

$$\int_\Omega (\nabla \cdot (F^V(W) - F^V(\Pi_0 W))) \cdot W^r d\Omega = \int_\Omega \left( \left( \frac{\partial F^V_1(W)}{\partial x} - \frac{\partial F^V_1(\Pi_0 W)}{\partial x} \right) + \left( \frac{\partial F^V_2(W)}{\partial y} - \frac{\partial F^V_2(\Pi_0 W)}{\partial y} \right) \right) \cdot W^r d\Omega,$$

where the 2D viscous terms $F^V_1(W) = (F^V_1(W) - F^V_2(W))$ (omitting the turbulent equation) read:

$$F^V_1(W) = \begin{bmatrix} 0 \\ (\mu + \mu_t) \frac{2}{3}(2u - v) \\ (\mu + \mu_t)(u + v) \\ (\mu + \mu_t)(\frac{2}{3}2u - v, u + (u + v) v) + (\lambda + \lambda_t)T_\tau \end{bmatrix}, \quad F^V_2(W) = \begin{bmatrix} 0 \\ (\mu + \mu_t)(u + v) \\ (\mu + \mu_t)\frac{2}{3}(2v - u) \\ (\mu + \mu_t)(\frac{2}{3}(2v - u), u + (2v - u, v) + (\lambda + \lambda_t)T_\tau \end{bmatrix}.$$

After integrating by part (omitting boundary terms), it gives:

$$\int_\Omega (\nabla \cdot (F^V(W) - F^V(\Pi_0 W))) \cdot W^r d\Omega \approx - \int_\Omega \left( \left( \frac{\partial F^V_1(W)}{\partial x} - \frac{\partial F^V_1(\Pi_0 W)}{\partial x} \right) + \left( \frac{\partial F^V_2(W)}{\partial y} - \frac{\partial F^V_2(\Pi_0 W)}{\partial y} \right) \right) \cdot W^r d\Omega = T_{\rho u} + T_{\rho v} + T^1_{\rho E} + T^2_{\rho E}.$$

The RHS of this relation is the sum of 4 terms which are analyzed in the following. In this error analysis we aim at expressing these terms as weighted interpolation error on the conservative variables. To this end, we will use again an integration by part to move the derivatives involved in the viscous terms $F^V_1$ and $F^V_2$ to the adjoint terms and make interpolation error appears on the conservative variables. We have to develop these vectorial expressions for this algebraic manipulation. To express $F^V_1(W)$ and $F^V_2(\Pi_0 W)$, we have to rewrite these terms as functions of the conservative variables. It is important to note that we have:

$$u(W) = \frac{\rho u}{\rho} \quad \text{and} \quad (\Pi_0 W) = \frac{\Pi_0(\rho u)}{\Pi_0 \rho} \neq \Pi_0 u. \quad (C.1)$$

Analysis of term $T_{\rho u}$.

The first term is

$$T_{\rho u} = - (\mu + \mu_t) \int_\Omega \left( \left( \frac{4}{3} u(W) - \frac{2}{3} v(W) \right) - \frac{4}{3} u(\Pi_0 W) - \frac{2}{3} v(\Pi_0 W) \right) (\rho u') \cdot d\Omega.$$ 

We integrate by parts (omitting boundary terms), then we replace $u$ and $v$ terms by the above expression (C.1) leading to:

$$T_{\rho u} \approx (\mu + \mu_t) \int_\Omega \left( \frac{4}{3} \left( \frac{\rho u}{\rho} - \frac{\Pi_0(\rho u)}{\Pi_0 \rho} \right) (\rho u')_{xx} - \frac{2}{3} \left( \frac{\rho u}{\rho} - \frac{\Pi_0(\rho u)}{\Pi_0 \rho} \right) (\rho u')_{xy} + \left( \frac{\rho u}{\rho} - \frac{\Pi_0(\rho u)}{\Pi_0 \rho} \right) (\rho u')_{yy} + \left( \frac{\rho u}{\rho} - \frac{\Pi_0(\rho u)}{\Pi_0 \rho} \right) \left( \frac{\rho u}{\rho} - \frac{\Pi_0(\rho u)}{\Pi_0 \rho} \right) (\rho u')_{yy} \right) d\Omega.$$ 

These terms can be expressed in term of interpolation error of the conservative variables with the following development

$$\frac{\rho u}{\rho} - \frac{\Pi_0(\rho u)}{\Pi_0 \rho} = \frac{\rho u}{\rho} - \frac{\rho u + (\Pi_0(\rho u) - \rho u)}{\rho + (\Pi_0 \rho - \rho)} = \frac{\rho u}{\rho} - \frac{\rho u + (\Pi_0(\rho u) - \rho u)}{\rho + (\Pi_0 \rho - \rho)} \frac{1}{1 + \frac{\rho}{2}(\Pi_0 \rho - \rho)}.$$
and now using a Taylor expansion of \( \frac{1}{1+\delta (\Pi _\rho - \rho)} \) in the vicinity of 0 (we assume the error term is small enough) and by neglecting high order error terms, we obtain:

\[
\frac{\rho u}{\rho} - \frac{\Pi _\rho (\rho u)}{\Pi _\rho \rho} \approx \frac{\rho u}{\rho} - \frac{\rho u + (\Pi _\rho (\rho u) - \rho u)}{\rho} (1 - \frac{1}{\rho} (\Pi _\rho \rho - \rho)) = \frac{\rho u - \rho u}{\rho} - \frac{1}{\rho} (\Pi _\rho (\rho u) - \rho u) + \frac{\rho u}{\rho^2} (\Pi _\rho (\rho u) - \rho u) (\Pi _\rho \rho - \rho) \approx \frac{1}{\rho} (\rho u - \Pi _\rho (\rho u)) - \frac{\rho u}{\rho^2} (\rho - \Pi _\rho \rho),
\]

(C.2)

and a similar result is obtained for the \( \frac{\rho v}{\rho} \) term. We thus have the following error estimation coming from the first term:

\[
T_{pu^*} \approx \int _\Omega \left( \frac{(\rho u - \Pi _\rho (\rho u)) (\mu + \mu _t)}{\rho} \left( \frac{4}{3} (\rho u^*_x)^2 + (\rho u^*_y)^2 \right) + \frac{(\rho v - \Pi _\rho (\rho v)) (\mu + \mu _t)}{\rho} \left( \frac{1}{3} (\rho v^*)_x \right) - (\rho - \Pi _\rho (\rho u)) \frac{4}{3} (\rho u^*_x)^2 + u (\rho u^*_y) + \frac{1}{3} v (\rho u^*_y) \right) d\Omega.
\]

**Analysis of term \( T_{pv^*} \).**

The second term is

\[
T_{pv^*} = -(\mu + \mu _t) \int _\Omega \left( \left( (u(W) + v(W)) - (u(\Pi _W W) + v(\Pi _W W)) \right) (\rho v^*)_y + \left( \frac{4}{3} u(W) - \frac{2}{3} u(\Pi _W W) \right) - \left( \frac{4}{3} v(W) - \frac{2}{3} v(\Pi _W W) \right) \right) d\Omega.
\]

We use exactly the same analysis as above to end up with the following error estimation coming from the second term:

\[
T_{pv^*} \approx \int _\Omega \left( \frac{(\rho v - \Pi _\rho (\rho v)) (\mu + \mu _t)}{\rho} \left( \frac{1}{3} (\rho v^*)_x \right) + \frac{(\rho v - \Pi _\rho (\rho v)) (\mu + \mu _t)}{\rho} \left( \frac{4}{3} (\rho v^*_y)^2 \right) \right. - \left. (\rho - \Pi _\rho (\rho v)) \frac{4}{3} v (\rho v^*_y) + \frac{1}{3} u (\rho v^*_y) \right) d\Omega.
\]

**Analysis of term \( T_{pE^*}^1 \).**

The third term is

\[
T_{pE^*}^1 = -(\mu + \mu _t) \int _\Omega \left( \left( \frac{4}{3} u(W) - \frac{2}{3} u(\Pi _W W) \right) + \left( \frac{4}{3} v(W) - \frac{2}{3} v(\Pi _W W) \right) \right) d\Omega.
\]

This term is more complex to analyze because its sub-terms involve at the same time the variables \( u, v \) and their derivatives. Therefore, we cannot apply directly an integration by parts. In this expression, we find three kinds sub-terms.

- The first kind of sub-terms (of the form \( uu_x, vv_x, uu_y, vv_y \)) is easy to manipulate:

\[
\int _\Omega (u(W) u(W) - u(\Pi _W W) u(\Pi _W W)) (\rho E^*)_x d\Omega = \int _\Omega \left( \frac{1}{2} (u(W))^2 \right) d\Omega \approx \int _\Omega \frac{1}{2} (u(W))^2 (\rho E^*)_x d\Omega.
\]

- The second kind of the form \( uu_y, vv_x, uu_x, vv_y \) is easy to manipulate:

\[
\int _\Omega (u(W) u(W) - u(\Pi _W W) u(\Pi _W W)) (\rho E^*)_y d\Omega = \int _\Omega \left( \frac{1}{2} (u(W))^2 \right) d\Omega \approx \int _\Omega \frac{1}{2} (u(W))^2 (\rho E^*)_y d\Omega.
\]

- The third kind of the form \( uu_y, vv_x, uu_x, vv_y \) is easy to manipulate:

\[
\int _\Omega (u(W) v(W) - u(\Pi _W W) v(\Pi _W W)) (\rho E^*)_y d\Omega \approx \int _\Omega \frac{1}{2} (u(W))^2 (\rho E^*)_y d\Omega.
\]
To pursue, we use the following relation:

$$u(W)^2 - u(\Pi_b W)^2 = \left(\frac{\rho u}{\rho} - \frac{\Pi_b(\rho u)}{\Pi_b \rho}\right)^2 = \left(\frac{\rho u}{\rho} - \left(\frac{\Pi_b(\rho u)}{\Pi_b \rho} - \frac{\rho u}{\rho}\right)\right)^2$$

$$= \left(\frac{\rho u}{\rho} - \frac{\Pi_b(\rho u)}{\Pi_b \rho} - \frac{\rho u}{\rho}\right)^2 - 2 \left(\frac{\rho u}{\rho} - \frac{\Pi_b(\rho u)}{\Pi_b \rho}ight) \left(\frac{\Pi_b(\rho u)}{\Pi_b \rho} - \frac{\rho u}{\rho}\right)$$

$$= -\left(\frac{\rho u}{\rho} - \frac{\Pi_b(\rho u)}{\Pi_b \rho}\right)^2 + 2 \left(\frac{\rho u}{\rho}\right) \left(\frac{\Pi_b(\rho u)}{\Pi_b \rho} - \frac{\rho u}{\rho}\right)$$

and, using Relation (C.2), we get:

$$u(W)^2 - u(\Pi_b W)^2 \approx -\frac{1}{\rho^2}(\rho u - \Pi_b(\rho u))^2 - \left(\frac{\rho u}{\rho^2}\right)^2 - 2 \frac{\rho u}{\rho^3}(\rho u - \Pi_b(\rho u))(\rho - \Pi_b \rho)$$

$$+ 2 \frac{\rho u}{\rho^3}(\rho u - \Pi_b(\rho u)) - 2 \left(\frac{\rho u}{\rho^2}\right)^2 - (\rho - \Pi_b \rho),$$

which gives by neglecting the high-order error (square error terms) terms:

$$u(W)^2 - u(\Pi_b W)^2 \approx 2 \frac{u}{\rho}(\rho u - \Pi_b(\rho u)) - 2 \frac{u^2}{\rho^2}(\rho - \Pi_b \rho). \quad \text{(C.3)}$$

Finally, that kind of sub-term gives:

$$\int_{\Omega} (u(W) u_x(W) - u(\Pi_b W) u_x(\Pi_b W)) (\rho E^+)_x \, d\Omega \approx \int_{\Omega} \left(\frac{u}{\rho}(\rho u - \Pi_b(\rho u)) \rho E^+ x + \frac{u^2}{\rho^2}(\rho - \Pi_b \rho)(\rho E^+ x) \right) \, d\Omega$$

- The second kind of sub-terms (of the form $v u_x + u v_x$ or $v_x u + u v$) is analyzed in a similar way:

$$\int_{\Omega} \left((v(W) u_x(W) + v_x(W) u(W)) - (v(\Pi_b W) u_x(\Pi_b W) + v_x(\Pi_b W) u(\Pi_b W))\right) (\rho E^+) y \, d\Omega$$

$$= \int_{\Omega} (v(W) u_x(W)) y - (v(\Pi_b W) u_x(\Pi_b W)) y \, d\Omega$$

$$\approx \int_{\Omega} -(u(W) v(W) - u(\Pi_b W) v(\Pi_b W)) (\rho E^+) y \, d\Omega.$$

Now, we develop:

$$u(W) v(W) - u(\Pi_b W) v(\Pi_b W) = \frac{\rho u \rho v}{\rho} - \frac{\Pi_b(\rho u v)}{\Pi_b \rho} = \frac{\rho u \rho v}{\rho} - \left(\frac{\rho u}{\rho} + \left(\frac{\Pi_b(\rho u v)}{\Pi_b \rho} - \frac{\rho u}{\rho}\right)\right)$$

$$= \frac{\rho u \rho v}{\rho} - \frac{\rho u}{\rho} \left(\frac{\Pi_b(\rho u v)}{\Pi_b \rho} - \frac{\rho v}{\rho}\right) - \rho v \left(\frac{\Pi_b(\rho u v)}{\Pi_b \rho} - \frac{\rho u}{\rho}\right)$$

$$= \frac{\rho u \rho v}{\rho} - \frac{\Pi_b(\rho u v)}{\Pi_b \rho} + \rho v \left(\frac{\rho u}{\rho} - \frac{\Pi_b(\rho u v)}{\Pi_b \rho}\right)$$

and, we use Relation (C.2) to obtain:

$$u(W) v(W) - u(\Pi_b W) v(\Pi_b W) \approx \frac{\rho u \rho v}{\rho^3}(\rho v - \Pi_b(\rho v)) - \frac{\rho v(\rho u v)}{\rho^3} - \rho v \left(\rho u - \Pi_b(\rho u v)\right)$$

$$- \frac{1}{\rho^2}(\rho u - \Pi_b(\rho u v)) \left(\rho v - \Pi_b(\rho v)\right) + \frac{\rho v}{\rho^3}(\rho u - \Pi_b(\rho u v))(\rho - \Pi_b \rho)$$

$$+ \frac{\rho u}{\rho^3}(\rho v - \Pi_b(\rho v))(\rho - \Pi_b \rho) - \frac{(\rho u)(\rho v)}{\rho^3}(\rho - \Pi_b \rho)^2$$

which gives by neglecting the high-order error (square error terms) terms:

$$u(W) v(W) - u(\Pi_b W) v(\Pi_b W) \approx \frac{v}{\rho} (\rho u - \Pi_b(\rho u v)) + \frac{u}{\rho} (\rho v - \Pi_b(\rho v)) - 2 \frac{u v}{\rho^2}(\rho - \Pi_b \rho) \quad \text{(C.4)}$$
Finally that kind of sub-term gives:

\[
\int_\Omega \left( (v(W) u_t(W) + v_c(W) u_c(W)) - (v(\Pi_b W) u_t(\Pi_b W) + v_c(\Pi_b W) u_c(\Pi_b W)) \right) (\rho E^*)_x \ d\Omega \\
\approx \int_\Omega \left( -\frac{\nu}{\rho} (\rho u - \Pi_b (\rho u)) (\rho E^*)_x - \frac{\nu}{\rho} (\rho v - \Pi_b (\rho v)) (\rho E^*)_y + 2 \frac{\nu}{\rho} \left( \rho - \Pi_b \rho \right) (\rho E^*)_{xy} \right) \ d\Omega 
\]

• The third and last kind of sub-term is the most complex to analyze, it reads:

\[
T = \int_\Omega \left( (v_c(W) u(W) - v_c(\Pi_b W) u(\Pi_b W)) (\rho E^*)_x + (u_t(W) v(W) - u_t(\Pi_b W) v(\Pi_b W)) (\rho E^*)_y \right) \ d\Omega 
\]

First, we expand the relation by introducing interpolation error like terms:

\[
T = \int_\Omega \left( (v_c(W) u(W) - (v(W) + (v(\Pi_b W) - v(W))) (u(W) + (u(\Pi_b W) - u(W)))) (\rho E^*)_x \\
+ \left( u_t(W) v(W) - (u(W) + (u(\Pi_b W) - u(W))) (v(W) + (v(\Pi_b W) - v(W))) \right) (\rho E^*)_y \right) \ d\Omega 
\]

Noting that |u(\Pi_b W) - u(W)| << |u(W)| thus |u(\Pi_b W) - u(W)||(v(\Pi_b W) - v(W))| << |u(W)||(v(\Pi_b W) - v(W))|, if we develop the above expression, neglect high-order terms and simplify, then we end-up with:

\[
T = \int_\Omega \left( -v_c(W) (u(\Pi_b W) - u(W)) - u_t(W) (v(\Pi_b W) - v(W)) \right) (\rho E^*)_x \\
+ \left( -u_t(W) (v(\Pi_b W) - v(W)) - v_c(W) (u(\Pi_b W) - u(W)) \right) (\rho E^*)_y \ d\Omega 
\]

which gives after integration by parts (omitting boundary terms):

\[
T = \int_\Omega \left( -v_c(W) (u(\Pi_b W) - u(W)) (\rho E^*)_x + u_t(W) (v(\Pi_b W) - v(W)) (\rho E^*)_x \\
- u_t(W) (v(\Pi_b W) - v(W)) (\rho E^*)_y + v_c(W) (u(\Pi_b W) - u(W)) (\rho E^*)_y \\
+ u(W) (v(\Pi_b W) - v(W)) (\rho E^*)_{xy} + v(W) (u(\Pi_b W) - u(W)) (\rho E^*)_{xy} \right) \ d\Omega 
\]

and now using Relations (C.1) and (C.2) we finally get:

\[
T \approx \int_\Omega \left( \left( \rho u - \Pi_b (\rho u) \right) \left( \frac{1}{\rho} \left( v_c (\rho E^*)_x - v_c (\rho E^*)_y - v (\rho E^*)_{xy} \right) \right) \\
+ \left( \rho v - \Pi_b (\rho v) \right) \left( \frac{1}{\rho} \left( u_t (\rho E^*)_y - u_t (\rho E^*)_x - u (\rho E^*)_{xy} \right) \right) \\
- \left( \rho - \Pi_b \rho \right) \left( \frac{1}{\rho} \left( u (\rho E^*)_y - v (\rho E^*)_x - v (\rho E^*)_{xy} + v (\rho E^*)_y - u (\rho E^*)_{xy} \right) \right) \right) \ d\Omega 
\]

We define the vectors

\[
\omega_u = (\omega_{u,x}, \omega_{u,y}, \omega_{u,z}) = \nabla u \times \nabla \rho E^+ \quad \text{and} \quad \omega_v = (\omega_{v,x}, \omega_{v,y}, \omega_{v,z}) = \nabla v \times \nabla \rho E^+ \quad \text{Relation (C.5)}
\]

then this term can be re-written:

\[
T \approx \int_\Omega \left( \left( \rho u - \Pi_b (\rho u) \right) \left( \frac{1}{\rho} \left( \omega_{u,z} - v (\rho E^*)_{xy} \right) \right) \\
+ \left( \rho v - \Pi_b (\rho v) \right) \left( \frac{1}{\rho} \left( \omega_{u,z} - u (\rho E^*)_{xy} \right) \right) \\
- \left( \rho - \Pi_b \rho \right) \left( \frac{1}{\rho} \left( u (\omega_{u,z} - v (\rho E^*)_{xy}) + v (\omega_{u,z} - u (\rho E^*)_{xy}) \right) \right) \right) \ d\Omega 
\]
To finish the analysis of term $T_{\rho E}^1$, we combine all the sub-terms treated above:

$$
T_{\rho E}^1 \approx (\mu + \mu) \int_{\Omega} \left( (pu - \Pi_b(\rho u)) \left( \frac{1}{\rho} \frac{4}{3} u (\rho E)^{\ast x} + 2v(\rho E)^{\ast y} + u(\rho E)^{\ast y} - \frac{5}{3} v(\rho E)^{\ast y} - \frac{5}{3} \omega_{xc} \right) + (\rho v - \Pi_b(\rho v)) \left( \frac{4}{3} u (\rho E)^{\ast x} + 2v(\rho E)^{\ast y} + u(\rho E)^{\ast y} - \frac{5}{3} v(\rho E)^{\ast y} - \frac{5}{3} \omega_{xc} \right) - (\rho - \Pi_b(\rho)) \left( \frac{4}{3} u (\rho E)^{\ast x} + 2v(\rho E)^{\ast y} + u(\rho E)^{\ast y} - \frac{5}{3} v(\rho E)^{\ast y} - \frac{5}{3} \omega_{xc} \right) - (\rho - \Pi_b(\rho)) \left( \frac{4}{3} u (\rho E)^{\ast x} + 2v(\rho E)^{\ast y} + u(\rho E)^{\ast y} - \frac{5}{3} v(\rho E)^{\ast y} - \frac{5}{3} \omega_{xc} \right) \right) d\Omega
$$

where

$$
\omega_{xc} = \frac{2}{\rho} \left( \frac{\rho u^{\ast} + (\rho v)^{\ast}}{2\rho} \right) - \left( \frac{\Pi_b(\rho u)}{\Pi_b(\rho)} \right)^2 + \left( \frac{\Pi_b(\rho v)}{\Pi_b(\rho)} \right)^2
$$

and

$$
\omega_{xc} = \frac{2}{\rho} \left( \frac{\rho u^{\ast} + (\rho v)^{\ast}}{2\rho} \right) - \left( \frac{\Pi_b(\rho u)}{\Pi_b(\rho)} \right)^2 + \left( \frac{\Pi_b(\rho v)}{\Pi_b(\rho)} \right)^2
$$

The fourth term is

$$
T_{\rho E}^2 \approx (\lambda + \lambda) \int_{\Omega} \left( (T_s(W) - T_s(\Pi_b(W)) (\rho E)^{\ast x} + (T_s(W) - T_s(\Pi_b(W)) (\rho E)^{\ast y}) \right) d\Omega.
$$

After integrating by parts (omitting boundary terms) and using the fact that $T = \frac{\rho E}{\rho - 1} = \frac{\rho E}{\rho} + \frac{(\rho u)^2 + (\rho v)^2}{2\rho^2}$ we have:

$$
T_{\rho E}^2 \approx (\lambda + \lambda) \int_{\Omega} \left( \frac{\Pi_b(\rho E)}{\Pi_b(\rho)} - \frac{\rho E}{\rho} \right) \left( \frac{\Pi_b(\rho u)}{\Pi_b(\rho)} + \frac{\Pi_b(\rho v)}{\Pi_b(\rho)} \right) \left( \frac{\Pi_b(\rho u)}{\Pi_b(\rho)} + \frac{\Pi_b(\rho v)}{\Pi_b(\rho)} \right) \left( \frac{\Pi_b(\rho u)}{\Pi_b(\rho)} + \frac{\Pi_b(\rho v)}{\Pi_b(\rho)} \right) \right) d\Omega.
$$

Using the same method as for Relation (C.2), we have also:

$$
\frac{\rho E}{\rho} - \frac{\Pi_b(\rho E)}{\Pi_b(\rho)} \approx \frac{1}{\rho} (\rho E - \Pi_b(\rho E)) - \frac{\rho E}{\rho^2} (\rho - \Pi_b(\rho))
$$

We have also terms of the form:

$$
- \frac{1}{2} \left( \frac{(\rho u)^2}{\rho} - \frac{(\Pi_b(\rho u))^2}{\Pi_b(\rho)} \right) = - \frac{1}{2} \left( \frac{(\rho v)^2}{\rho} - \frac{(\Pi_b(\rho v))^2}{\Pi_b(\rho)} \right)
$$

Using Relation (C.3), we obtain:

$$
- \frac{1}{2} \left( \frac{(\rho u)^2}{\rho} - \frac{(\Pi_b(\rho u))^2}{\Pi_b(\rho)} \right) \approx - \frac{u}{\rho} (\rho u - \Pi_b(\rho u)) + \frac{u^2}{\rho} (\rho - \Pi_b(\rho))
$$

We thus have the following error estimation coming from the last term:

$$
T_{\rho E}^2 \approx \int_{\Omega} \left( - \frac{(\rho u - \Pi_b(\rho u)) (\lambda + \lambda)}{\rho} \left( \frac{\Pi_b(\rho u)}{\Pi_b(\rho)} + \frac{\Pi_b(\rho v)}{\Pi_b(\rho)} \right) \left( \frac{\Pi_b(\rho u)}{\Pi_b(\rho)} + \frac{\Pi_b(\rho v)}{\Pi_b(\rho)} \right) \left( \frac{\Pi_b(\rho u)}{\Pi_b(\rho)} + \frac{\Pi_b(\rho v)}{\Pi_b(\rho)} \right) \right) d\Omega.
$$
To finish the proof of the error estimate for the viscous terms, we gather the four terms:

\[
\int_{\Omega} \left( \nabla \cdot (F^V(W) - F^V(\Pi_0 W)) \right) \cdot W^{*} \, d\Omega \approx T_{p\nu} + T_{p\nu} + T_{pE} + T_{pE} \approx \int_{\Omega} \sum_{i=1}^{4} V(W_i) (W_i - \Pi_0 W_i) \, d\Omega
\]

where

\[
V(\rho) = -u f_{p\rho}(W, H_{w^*}) - v f_{p\rho}(W, H_{w^*}) - (E - u^2 - v^2) f_{pE}(W, H_{w^*})
\]

\[
V(\rho u) = f_{p\rho}(W, H_{w^*}) - u f_{pE}(W, H_{w^*})
\]

\[
V(\rho v) = f_{p\rho}(W, H_{w^*}) - v f_{pE}(W, H_{w^*})
\]

\[
V(\rho E) = f_{pE}(W, H_{w^*})
\]

with

\[
f_{p\rho}(W, H_{w^*}) = \frac{1}{3} \left( \frac{\mu + \mu_s}{\rho} \right) \left( 4 (\rho u^*)_{xx} + 3 (\rho u^*)_{xy} + (\rho v^*)_{xx} + 4 u(\rho E^*)_{xx} + v(\rho E^*)_{xy} + 3 u(\rho E^*)_{xy} - 5 \omega_{xz} \right)
\]

\[
f_{p\rho}(W, H_{w^*}) = \frac{1}{3} \left( \frac{\mu + \mu_s}{\rho} \right) \left( 3 (\rho v^*)_{xx} + 4 (\rho v^*)_{xy} + (\rho u^*)_{yy} + 4 u(\rho E^*)_{xx} + 3 v(\rho E^*)_{xy} + 4 v(\rho E^*)_{yy} + 5 \omega_{xz} \right)
\]

\[
f_{pE}(W, H_{w^*}) = \frac{(\lambda + \lambda_s)}{\rho} \left( (\rho E^*)_{xx} + (\rho E^*)_{yy} \right)
\]

and \(\omega\) given by Relation (C.5).

Appendix C.2. 3D error analysis in conservative variables \((\rho, \rho u, \rho v, \rho w, \rho E)\)

We start from:

\[
\int_{\Omega} \left( \nabla \cdot (F^V(W) - F^V(\Pi_0 W)) \right) \cdot W^{*} \, d\Omega
\]

\[
= \int_{\Omega} \left( \left( \frac{\partial F^V_1(W)}{\partial x} - \frac{\partial F^V_1(\Pi_0 W)}{\partial x} \right) + \left( \frac{\partial F^V_2(W)}{\partial y} - \frac{\partial F^V_2(\Pi_0 W)}{\partial y} \right) + \left( \frac{\partial F^V_3(W)}{\partial z} - \frac{\partial F^V_3(\Pi_0 W)}{\partial z} \right) \right) \cdot W^{*} \, d\Omega,
\]

where the 3D viscous terms \(F^V(W) = (F^V_1(W), F^V_2(W), F^V_3(W))\) (omitting the turbulent equation) read:

\[
F^V_1(W) = \begin{cases} 
0 \\
(\mu + \mu_s) \left( \frac{2}{3} (2 u_s - v_i - w_j) + (\mu + \mu_s) (u_i + v_i) + (\mu + \mu_s) (u_i + w_i) \right) \\
(\mu + \mu_s) \left( \frac{2}{3} (2 u_s - v_i - w_j) + (u_i + v_i) v + (u_i + w_i) w + (\lambda + \lambda_s) T_i \right)
\end{cases}
\]

\[
F^V_2(W) = \begin{cases} 
0 \\
(\mu + \mu_s) (u_i + v_i) \\
(\mu + \mu_s) \left( \frac{2}{3} (2 v_j - u_s - w_i) + (\mu + \mu_s) (v_i + w_i) \right) \\
(\mu + \mu_s) \left( (u_i + v_i) u + \frac{2}{3} (2 v_j - u_s - w_i) v + (v_i + w_i) w + (\lambda + \lambda_s) T_i \right)
\end{cases}
\]

\[
F^V_3(W) = \begin{cases} 
0 \\
(\mu + \mu_s) (u_i + w_i) \\
(\mu + \mu_s) (v_i + w_i) \\
(\mu + \mu_s) \left( \frac{2}{3} (2 w_j - u_s - v_i) + (\mu + \mu_s) (u_i + w_i) u + (v_i + w_i) v + \frac{2}{3} (2 w_j - u_s - v_i) w + (\lambda + \lambda_s) T_i \right)
\end{cases}
\]

76
Similarly, we have the following error estimation coming from the third term:

\[ \int_\Omega \left( \nabla \cdot (E^V(W) - \mathcal{F}^V(\Pi_bW)) \right) \cdot \nabla^* \d\Omega \]

\[ \approx - \int_\Omega \left( \left( E^V_1(W) - \mathcal{F}^V(\Pi_bW) \right) \cdot \frac{\partial \nabla^*}{\partial x} + \left( E^V_2(W) - \mathcal{F}^V(\Pi_bW) \right) \cdot \frac{\partial \nabla^*}{\partial y} + \left( E^V_3(W) - \mathcal{F}^V(\Pi_bW) \right) \cdot \frac{\partial \nabla^*}{\partial z} \right) \d\Omega \]

\[ = T_{pw^*} + T_{pv^*} + T_{pw^*} + T_{pw^*}^\perp + T_{pv^*}^\perp. \]

The RHS of this relation is the sum of 5 terms which are analyzed in the same way as for the 2D case. Expression of the terms \( T_{pw^*}, T_{pv^*}, T_{pw^*}, T_{pw^*}^\perp, T_{pv^*}^\perp \) are immediate generalization of the 2D case, we thus give directly the results. The main difficulty is again the third sub-term of term \( T_{pw^*}^\perp \) for which we provide a more detailed proof.

**Analysis of term \( T_{pw^*} \).**

Following the 2D proof, we have the following error estimation coming from the first term:

\[
T_{pw^*} \approx \int_\Omega \left( (\rho u - \Pi_b(\rho u)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{4}{3} (\rho u^*)_{xx} + (\rho u^*)_{yy} + (\rho u^*)_{zz} \right) \right. \\
+ \left. (\rho v - \Pi_b(\rho v)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{1}{3} (\rho u^*)_{xy} \right) \right. \\
+ \left. (\rho w - \Pi_b(\rho w)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{1}{3} (\rho u^*)_{x} \right) \right. \\
- \left. \text{big}(\rho - \Pi_b(\rho)) \left( \left( \frac{4}{3} u(\rho u^*)_{xx} + u(\rho u^*)_{yy} + u(\rho u^*)_{zz} + \frac{1}{3} v(\rho u^*)_{xy} + \frac{1}{3} w(\rho u^*)_{x} \right) \right) \right) \d\Omega.
\]

**Analysis of term \( T_{pv^*} \).**

Following the 2D proof, we have the following error estimation coming from the second term:

\[
T_{pv^*} \approx \int_\Omega \left( (\rho u - \Pi_b(\rho u)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{1}{3} (\rho v^*)_{yy} \right) \right. \\
+ \left. (\rho v - \Pi_b(\rho v)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{4}{3} v(\rho v^*)_{yy} + (\rho v^*)_{zz} \right) \right. \\
+ \left. (\rho w - \Pi_b(\rho w)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{1}{3} (\rho v^*)_{y} \right) \right. \\
- \left. \text{big}(\rho - \Pi_b(\rho)) \left( \left( \frac{1}{3} u(\rho v^*)_{yy} + v(\rho v^*)_{yy} + v(\rho v^*)_{zz} + \frac{4}{3} v(\rho v^*)_{yy} + \frac{1}{3} w(\rho v^*)_{y} \right) \right) \right) \d\Omega.
\]

**Analysis of term \( T_{pw^*} \).**

Similarly, we have the following error estimation coming from the third term:

\[
T_{pw^*} \approx \int_\Omega \left( (\rho u - \Pi_b(\rho u)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{1}{3} (\rho w^*)_{zz} \right) \right. \\
+ \left. (\rho v - \Pi_b(\rho v)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{1}{3} (\rho w^*)_{zz} \right) \right. \\
+ \left. (\rho w - \Pi_b(\rho w)) \left( \frac{\mu + \mu_i}{\rho} \right) \left( \frac{4}{3} (\rho w^*)_{zz} \right) \right. \\
- \left. \text{big}(\rho - \Pi_b(\rho)) \left( \left( \frac{1}{3} u(\rho w^*)_{zz} + v(\rho w^*)_{zz} + w(\rho w^*)_{zz} + \frac{4}{3} v(\rho w^*)_{zz} \right) \right) \right) \d\Omega.
\]
Analysis of term $T_{pE^c}^2$. 

The error estimation coming from the last term is simply:

$$T_{pE^c}^2 \approx \int_{\Omega} \left( - (\mu - \Pi_0(\mu)) (\lambda + \mu) \left[ \left( \frac{\lambda}{\rho} \right) E_{x}^2 + \left( \frac{\mu}{\rho} \right) E_{y}^2 + \left( \frac{\mu}{\rho} \right) E_{z}^2 \right] 
- (\rho \mu - \Pi_0(\rho \mu)) (\lambda + \mu) \left[ \left( \frac{\lambda}{\rho^2} \right) E_{x}^2 + \left( \frac{\mu}{\rho^2} \right) E_{y}^2 + \left( \frac{\mu}{\rho^2} \right) E_{z}^2 \right] 
+ (\rho \mu - \Pi_0(\rho \mu)) (\lambda + \mu) \left[ \left( \frac{\lambda^2}{\rho^2} \right) E_{x}^2 + \left( \frac{\mu^2}{\rho^2} \right) E_{y}^2 + \left( \frac{\mu^2}{\rho^2} \right) E_{z}^2 \right] \right) \rho \, d\Omega .$$

Analysis of term $T_{pE^c}^1$. 

In 2D, we have seen that this (more complex) term is composed of three sub-terms:

$$T_{pE^c}^1 = - (\mu + \mu_0) \int_{\Omega} \left( \left( \frac{4}{3} \left( \frac{\lambda}{\rho^2} \right) E_{x}^2 + \left( \frac{\mu}{\rho^2} \right) E_{y}^2 + \left( \frac{\mu}{\rho^2} \right) E_{z}^2 \right) \rho \, d\Omega 
+ \left( \frac{4}{3} \left( \frac{\mu}{\rho^2} \right) E_{y}^2 + \left( \frac{\mu}{\rho^2} \right) E_{z}^2 \right) \rho \, d\Omega \right) \rho \, d\Omega .$$

The first and second kind of sub-terms are exactly the same adding $w$-terms and $z$-terms. The only term to analyze, which is different from 2D, is the third one, i.e., the set of terms with the $\frac{4}{3}$ coefficient. In 3D, it reads:

$$T = \int_{\Omega} \left( \left( \frac{4}{3} \left( \frac{\mu}{\rho^2} \right) E_{x}^2 + \left( \frac{\mu}{\rho^2} \right) E_{y}^2 + \left( \frac{\mu}{\rho^2} \right) E_{z}^2 \right) \rho \, d\Omega 
+ \left( \frac{4}{3} \left( \frac{\mu}{\rho^2} \right) E_{y}^2 + \left( \frac{\mu}{\rho^2} \right) E_{z}^2 \right) \rho \, d\Omega \right) \rho \, d\Omega .$$
Like in 2D, we expand all the terms by introducing interpolation error like terms, then we cancel equal terms and we neglect high-order interpolation error term to end-up with:

\[
T = \int_{\Omega} \left( (v - \Pi_h W)(u(\Pi_h W) - u(W)) - u(W)(v(\Pi_h W) - v(W)) \right) (\rho E^+) \, dx + \left( w - \Pi_h W \right)(u(\Pi_h W) - u(W)) - u(W)(w(\Pi_h W) - w(W)) \right) (\rho E^+) \, dx + \left( u - \Pi_h W \right)(w(\Pi_h W) - w(W)) - w(W)(u(\Pi_h W) - u(W)) \right) (\rho E^+) \, dx + \left( v - \Pi_h W \right)(w(\Pi_h W) - w(W)) - w(W)(v(\Pi_h W) - v(W)) \right) (\rho E^+) \, dx \right) \, d\Omega,
\]

which gives after integration by parts (omitting boundary terms) and using Relations (C.1) and (C.2):

\[
T \approx \int_{\Omega} \left( (\rho u - \Pi_h (\rho u)) \frac{1}{\rho} \left( - \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z} - v(\rho E^+)_{xy} - w(\rho E^+)_{xz} \right) + (\rho v - \Pi_h (\rho v)) \frac{1}{\rho} \left( \frac{4}{3} \frac{\partial (\rho E^+)}{\partial x} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial y} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial z} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial y} \right) + (\rho w - \Pi_h (\rho w)) \frac{1}{\rho} \left( - \frac{\partial w}{\partial x} + \frac{\partial v}{\partial z} - v(\rho E^+)_{xz} - w(\rho E^+)_{yz} \right) \right) \, d\Omega,
\]

where we have define the vectors:

\[
\omega_x = (\omega_{u,x}, \omega_{v,y}, \omega_{w,z}) = \nabla u \times \nabla \rho E^+, \quad \omega_v = (\omega_{u,x}, \omega_{v,y}, \omega_{w,z}) = \nabla v \times \nabla \rho E^+, \quad \omega_w = (\omega_{u,z}, \omega_{v,z}, \omega_{w,z}) = \nabla w \times \nabla \rho E^+.
\]

We then obtain:

\[
T_{1(\rho E^+)} = (\mu + \mu_0) \int_{\Omega} \left( (\rho u - \Pi_h (\rho u)) \frac{1}{\rho} \left( \frac{4}{3} \frac{\partial (\rho E^+)}{\partial x} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial y} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial z} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial y} \right) + (\rho v - \Pi_h (\rho v)) \frac{1}{\rho} \left( \frac{4}{3} \frac{\partial (\rho E^+)}{\partial x} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial y} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial z} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial y} \right) + (\rho w - \Pi_h (\rho w)) \frac{1}{\rho} \left( \frac{4}{3} \frac{\partial (\rho E^+)}{\partial x} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial y} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial z} + \frac{1}{3} \frac{\partial (\rho E^+)}{\partial y} \right) \right) \, d\Omega,
\]

To finish the 3D proof of the error estimate for the viscous terms, we gather the five terms:

\[
\int_{\Omega} \left( \nabla \cdot (\mathcal{F}(W) - \mathcal{F}(\Pi_h W)) \right) : W^* \, d\Omega \approx T_{\rho u} + T_{\rho v} + T_{\rho w} + T_{\rho E^+} + T_{\rho E^+}^2 \approx \int_{\Omega} \sum_{i=1}^{5} V(W_i) (W_i - \Pi_h W_i) \, d\Omega
\]

where

\[
V(\rho) = -fu_{f}(W, H_{W'}) - v f_{f}(W, H_{W'}) - w f_{f}(W, H_{W'}) - (E - u^2 - v^2 - w^2) f_{f}(W, H_{W'})
\]

\[
V(\rho u) = f_{f}(W, H_{W'}) - u f_{f}(W, H_{W'})
\]

\[
V(\rho v) = f_{f}(W, H_{W'}) - v f_{f}(W, H_{W'})
\]

\[
V(\rho w) = f_{f}(W, H_{W'}) - w f_{f}(W, H_{W'})
\]

\[
V(\rho E) = f_{f}(W, H_{W'})
\]

79
with

\[ f_{\mu\mu}(W, H_{W^*}) = \frac{1}{3} \left( \mu + \mu_t \right) \left( 4 (\rho u^*)_{xs} + 3 (\rho u^*)_{ys} + 3 (\rho u^*)_{zs} + 4 u (\rho E^*)_{xs} + v (\rho E^*)_{ys} + 3 u (\rho E^*)_{zs} + 3 u (\rho E^*)_{ zs} - 5 \omega_{x,z} + 5 \omega_{y,z} \right) \]

\[ f_{\rho\rho}(W, H_{W^*}) = \frac{1}{3} \left( \mu + \mu_t \right) \left( 3 (\rho v^*)_{xs} + 4 (\rho v^*)_{ys} + 3 (\rho v^*)_{zs} + 3 (\rho v^*)_{zs} + 3 v (\rho E^*)_{xs} + u (\rho E^*)_{ys} + 4 v (\rho E^*)_{zs} + 3 v (\rho E^*)_{zs} + 3 v (\rho E^*)_{zs} + 5 \omega_{y,z} - 5 \omega_{x,z} \right) \]

\[ f_{\omega\omega}(W, H_{W^*}) = \frac{1}{3} \left( \mu + \mu_t \right) \left( 3 (\rho w^*)_{xs} + 3 (\rho w^*)_{ys} + 4 (\rho w^*)_{zs} + (\rho w^*)_{zs} + 3 w (\rho E^*)_{xs} + u (\rho E^*)_{ys} + 3 w (\rho E^*)_{zs} + v (\rho E^*)_{zs} + 4 w (\rho E^*)_{zs} + 5 \omega_{x,y} + 5 \omega_{x,z} \right) \]

\[ f_{\phi\phi}(W, H_{W^*}) = \frac{(\lambda + \lambda_t)}{\rho} \left( (\rho E^*)_{xs} + (\rho E^*)_{ys} + (\rho E^*)_{zs} \right). \]

and \( \omega \) given by Relation (C.6).