Unstructured Mesh Generation and Adaptation

Adrien Loseille

Unstructured Mesh Generation and Adaptation

Adrien Loseille*

October 13, 2016

Abstract

We first describe the well established unstructured mesh generation methods as involved in the computational pipeline, from geometry definition to surface and volume mesh generation. These components are always a preliminary and required step to any numerical computations. From an historical point of view, the generation of fully unstructured mesh generation in 3D has been a real challenge so as to the design of robust and accurate second order schemes on such unstructured meshes. If the issue of generating volume meshes for geometries of any complexity is now mostly solved, the emergence of robust numerical schemes on unstructured meshes has paved the way to adaptivity. Indeed, unstructured meshes in contrast with structured or block structured grids have the necessary flexibility to control the discretization both in size and orientation.

In the second part, we review the main components to perform adaptative computations: (i) anisotropic mesh prescription *via* a metric field tensor (ii) anisotropic error estimates, and (iii) anisotropic mesh generation. For each component, we focus on a particularly simple method to implement. In particular, we describe a simple but robust strategy for generating anisotropic meshes. Each adaptation entity, *ie* surface, volume or boundary layers, relies on a specific metric tensor field. The metric-based surface estimate is then used to control the deviation to the surface and to adapt the surface mesh. The volume estimate aims at controlling the interpolation error of a specific field of the flow.

Several 3D examples issued from steady and unsteady simulations from systems of hyperbolic laws are presented. In particular, we show that despite the simplicity of the introduced adaptive meshing scheme a high level of anisotropy can be reached. This includes the direct prediction of the sonic boom of an aircraft by computing the flow from the cruise altitude to the ground, the interaction between shock waves and boundary layer, or the prediction of complex unsteady phenomena in 3D.

keywords: Unstructured mesh generation; Anisotropic mesh adaptation; Metric-based error estimates; Surface approxiamtion; Euler equations; Navier-Stokes equations; Local remeshing; Sonic Boom Prediction; Blast; Boundary-layer/shock interaction.

Contents

*INRIA Saclay-Ile de France, France, (Adrien.Loseille@inria.fr).

Introduction

For flows involved in aerospace, naval, train and automotive industries or more generally in Computational Fluids Dynamics (CFD), the numerical prediction of a physical phenomenom follows the computational pipeline of Fig 1. From a continuous description of the geometry, a surface then a volume mesh are generated, see Fig 2. This mesh is used as a discrete support to solve a set partial differential equations (PDEs) by using any typical second order accurate numerical schemes [1, 24, 45, 81]. When unstructured meshes are used, the meshing and computation steps have reached a great level of maturity and automaticity, allowing to quickly modify the design and run a new simulation, even for highly complex geometries [12, 14, 37, 48, 55, 68, 69, 70]. During the mesh generation process, the sizing of the elements [11] is either based on a user *a priori* knowledge of the flow or is induced by the geometry. To take into account the whole flow features evaluated at the computation step while keeping automaticity, mesh adaptivity is required.

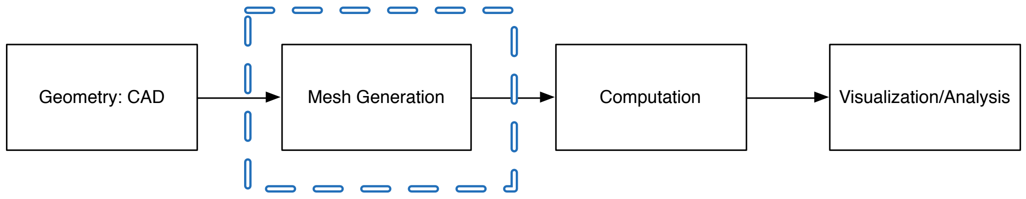

Figure 1: Computational pipeline.

Indeed, we observe that the solutions of non linear system of PDEs like the Euler of Navier-Stokes equations have complex features and multiscale phenomena: shocks waves, boundary

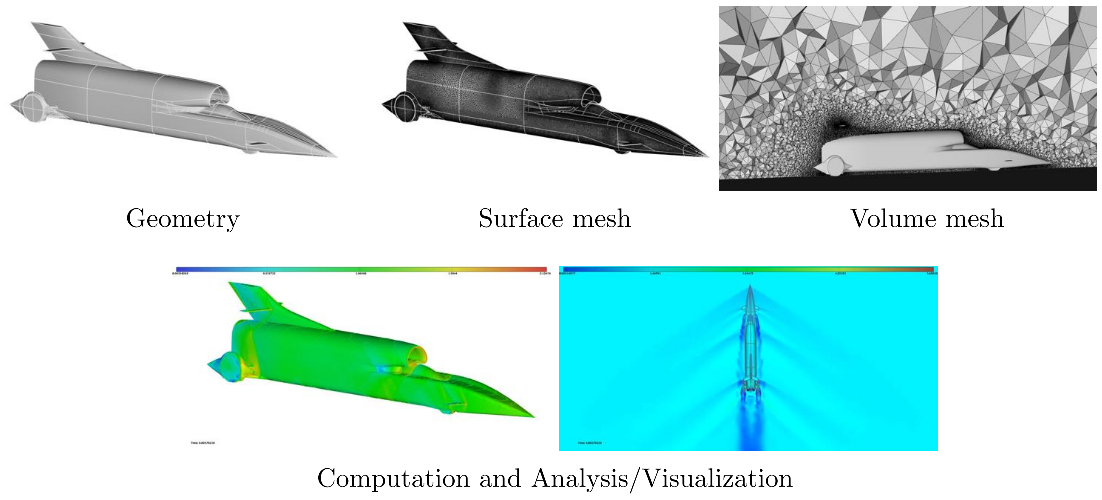

Figure 2: Illustration of the computational pipeline on the Bloodhound© supersonic car

layers, turbulence, ... When dealing with complex geometries, all these features are present in the flow field and interact with each other. It is then hardly impossible to design a tailored mesh to capture all these phenomena. To capture accurately them automatically, we typically use specific mesh adaptation procedures. We can distinguish: (i) isotropic and structured grids for turbulent flows, (ii) anisotropic meshes for shock capturing with an anisotropic ratio of the order of $O(1 : 100 - 1000)$ and (iii) highly stretched quasi-structured meshes with a ratio of $O(1 : 10^4 - 10^6)$ for boundary-layers. Many numerical examples have proved that the performance of a numerical scheme is bounded by the quality and the features of the discretization. For instance, we prefer anisotropic meshes to capture accurately shocks [61] while we use cartesian grids at a turbulent regime to allow high-order capturing of vortices. In the vicinity of bodies, quasi-structured grids are employed to capture the boundary layer in viscous simulations [14, 68]. If all these methods have now reached a good level of maturity, they are generally studied on their own. Consequently it seems difficult to handle together all the optimal meshes for all these phenomena. In this chapter, we will focus on one simple solution to generate all these kinds of meshes within a common framework. In this framework, the requirements on the mesh (for sizes, shapes and orientations) are expressed in term of a field of metric tensors and dedicated quality functions. We will consider metric fields issued from interpolation error, surface geometric approximation, and boundary-layer model. These fields are then used as a continuous support to drive the adaptation. From a practical point of view, simple anisotropic local operators as edge collapse, point insertion, edge swapping and point smoothing are recursively used to modify and improve the mesh. Note that each operator is monitored by a quality function to ensure that a quality mesh is outputted. This requirement is important to ensure the stability and enhance the performance of the flow solver.

Outline. The paper is decomposed as follows. In Section 1, we describe the main steps involved in generating a first mesh for complex geometries in an unstructured context. In Section 2, we recall the main concepts of metric-based mesh adaptation. We then define various metric-field expressions used for surface, volume, boundary layer and error control. In Section 3, we describe the algorithms used to generate an anisotropic mesh with respect to a prescribed metric. In Section 4, we briefly comment the adaptive loop, and we illustrate the previous concepts on both

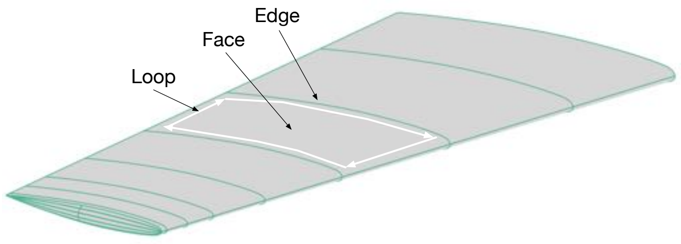

Figure 3: Topology hierarchy (Face, Loop, Edge) of the continuous representation of model using the Boundary REPresentation (BREP).

steady and unsteady simulations.

1 An introduction to unstructured mesh generation

We quickly describe and illustrate on simple examples the basic principles underlying the generation of unstructured meshes for complex geometries. For a complete description, we refer to the following monographs [32, 35, 52, 53].

1.1 Surface mesh generation

In industrial applications, the definition of the computational domain (or of a design) is provided by a continuous description composed by a collection of patches using a CAD (Computer Aided Design) system. If several continuous representation of a patch exist *via* an implicit equation or a solid model, we focus on the boundary representation (BREP). In this description, the topology and the geometry are defined conjointly. For the topological part, a hierarchical description is used from top level topological objects to lower level objects, we have:

$$\text{model} \longrightarrow \text{bodies} \longrightarrow \text{faces} \longrightarrow \text{loops} \longrightarrow \text{edges} \longrightarrow \text{nodes}$$

Each entity of upper level is described by a list of entities of lower level. This is represented in Figure 3 for an Onera M6 model, where a face, a loop and corresponding edges are depicted. Note that most of the time, only the topology of a face is provided, the topology between all the faces (patches) need to be recovered. This peace of information is needed to have a watertight valid surface mesh on output for the whole computational domain. This step makes the surface mesh generation of equal difficulty as volume mesh generation and have been shown to be not trivial [8].

For node, edge, and face, a geometry representation is also associated to the entity. For node, it is generally the position in space, while for edge and face a parametric representation is used. It consists in defining a mapping from a bounded domain of $\mathbb{R}^2$ onto $\mathbb{R}^3$ such that $(x, y, z) = \sigma(u, v)$ where (u, v) are the parameters. Generally, σ is a NURBS function (Non-uniform rational B-spline) as it is a common tool in geometry modeling and CAD systems [76]. From a conceptual point of view, meshing a parametric surface consists in meshing a 2D domain in the parametric space. However, surface mesh generation is not as naive as it seems, as several issues are faced to get a valid surface mesh:

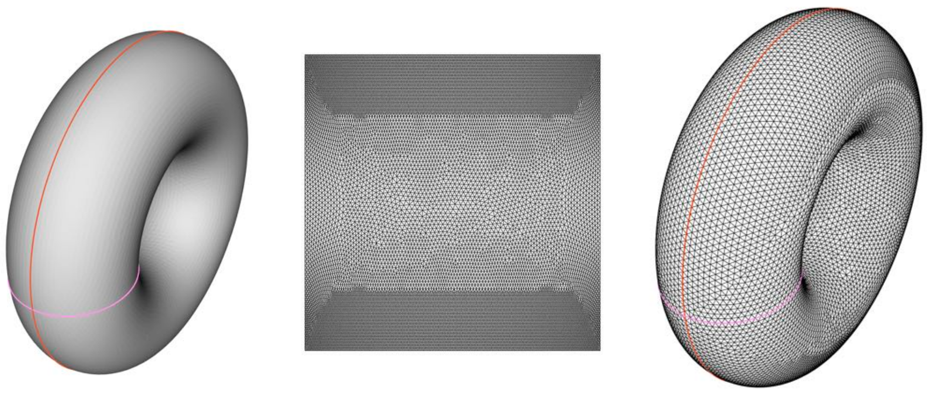

Figure 4: From left to right, CAD of a torus with 2 edges and one face, 2D mesh in the parametric space, and mapped uniform surface mesh.

- The mapping function is not bijective, *i.e.,* an infinite number of parameters values may have the same value in $\mathbb{R}^3$;
- A valid mesh in the parametric space may be invalid when mapped to 3D as σ is not necessary monotone;
- Having a uniform mesh in $\mathbb{R}^3$ requires to have a highly anisotropic adapted mesh in $\mathbb{R}^2$ due to the length distortion imposed by σ;
- The typical CAD queries (normal, tangent planes, principal curvatures) are based on the derivatives of σ may have undefined behaviors especially near the boundaries of the parametric space.

We illustrate this on the mesh of torus composed of two edges and one face, see Figure 4. We notice that if the mesh in $\mathbb{R}^3$ is perfectly uniform, it is not the case in the parametric space.

For further readings, the aforementioned issues of CAD parameterizations and their consequences for adaptivity are discussed in [75]. Robust meshing of NURBS surface is studied in [12] and implementation details are provided in [48].

1.2 Volume mesh generation

Once the generation of the surface mesh is completed, a volume mesh is generated to fill the domain with a tetrahedra. The surface mesh then becomes an input but also a constraint as all the input triangles have to match a face of the tetrahedral mesh. Two different approches have emerged and have proved to be robust to the complexity of the geometry: the frontal and the Delaunay methods.

The frontal approach is the easiest to understand in its principles. The process starts from the surface mesh that defines an initial front (a set of faces). From this front, a set of optimal points are created such that for each face of the front, an optimally shaped element would be created. This set of points is then checked and filtered to avoid collision and overlapping of faces. A reduced set of points is then inserted one point at a time and the front is updated. The same procedure is repeated until the whole domain is filled. The pros of this approach is that the shape

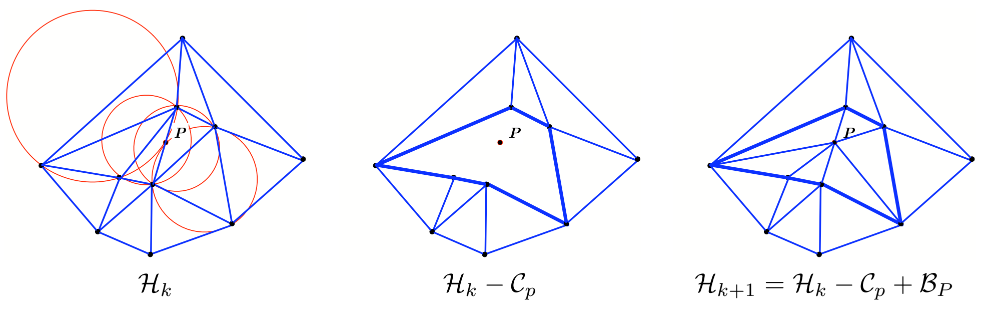
P
P
P
$\mathcal{H}_k$
$\mathcal{H}_k - \mathcal{C}_p$
$\mathcal{H}_{k+1} = \mathcal{H}_k - \mathcal{C}_p + \mathcal{B}_P$

Figure 5: Illustration of the incremental Delaunay insertion of a point in a mesh.

of elements can be controlled and different kinds of meshes can be obtained by modifying the optimal point procedure: cartesian core, iso-tetrahedra, ..., see [53] for more details. If meshes of very high quality are obtained when starting from isotropic surface meshes, the critical steps is in the closure of the front. Indeed, there is no guarantee that the procedure will end up with an empty front. This weakness tends to increase when anisotropic triangles are present in the initial surface mesh. We refer to [54] for an updated description of the frontal approach.

The second approach is the constrained Delaunay. It starts from an initial simple mesh of a box surrounding the surface mesh (composed of six tetrahedra). We then have the following steps:

(i) Insert the points of the surface mesh in the current mesh;

(ii) Recover the boundary corresponding to the initial surface mesh (list of edges and faces);

(iii) Fill the interior of the domain by inserting internal points;

(iv) Optimize the mesh with the smoothing of points and the swap of edges and faces.

Contrary to the frontal approach, a valid 3D mesh is always kept through the entire process. This is due to the insertion procedure based on a iterative process, see Figure 5. Once Step (i) is completed, some faces or edges of the initial mesh may not be present in the current mesh, a boundary recovery is used. It is generally used in enforcing these entities by applying successively or randomly standard optimization operators as the swap of edges and faces [36]. In addition, some theoretical and constructive proofs exist to show that this procedure can succeed to generate a mesh, see [38, 82]. The most critical step is the second one. However, if we accept to modify the initial surface mesh, this procedure can always succeed to output a volume mesh with a (slightly) modified surface mesh. Consequently, this approach is more robust than the frontal approach. The procedure is illustrated in Figure 6.

Note that a lot of hybrid approaches are a combinaison of both. The frontal creation of points can be used with Delaunay insertion, or the closure of the front can used a complete constrained Delaunay approach. For the two core methods, a simple example comparing both approaches is depicted in Figure 7.

2 Metric-based mesh adaptation

If unstructured meshes have been employed primarily to handle complex geometries, their great flexibility allows us to consider anisotropic mesh adaptation. The intent of adaptivity is then to

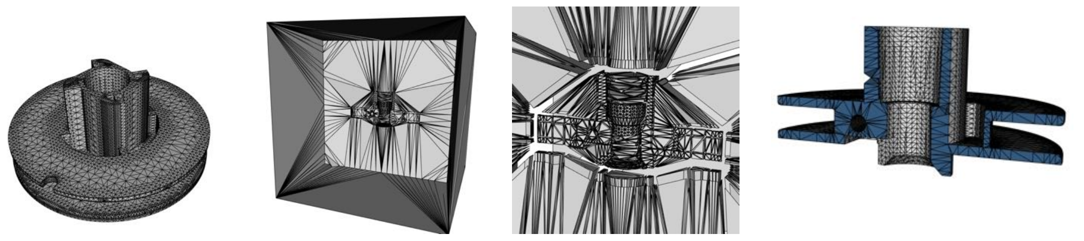

Figure 6: Constrained Delaunay method. From left to right, initial surface mesh, volume mesh after insertion of the surface points, volume mesh after boundary recovery and final mesh by removing element connected to the initial mesh of the surrounding box.

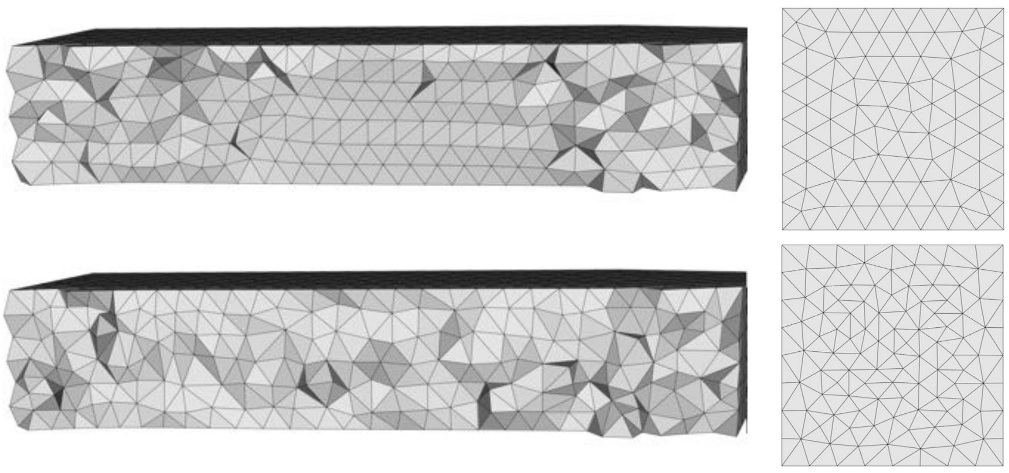

Figure 7: Cuts in a volume mesh filled with the frontal method (top left) and Delaunay insertion (bottom left) and right, 2d square domain filled with frontal (top right) and Delaunay (top bottom).

optimize the ratio between the level of accuracy and the CPU time to run a simulation. The expected gain is mostly motivated by the physical features of the flow, especially for systems of hyperbolic laws where the solutions have strong anisotropic components. It is then clear that using uniform meshes is not optimal (for the distribution of the degrees of freedom) to reach a given level of accuracy. Two examples of flows with anisotropic features are given in Figure 8 with a supersonic flow and the vorticity behind a business jet.

To perform anisotropic mesh adaptation, we have to define the following: (i) a directional error estimate, (ii) a way to prescribe the desired sizes and orientations (iii) and finally a set of mesh modification operators to generate anisotropic meshes. In this section, we introduce the **metric-based** approach where continuous and discrete tensor fields are used to handle (i)-(iii). The key idea is to generate a uniform mesh, **a unit mesh**, with respect to a Riemmannian metric space. More precisely, the geometric quantities as length, volume, angle, quality, . . . , are then evaluated in this space instead of using the standard Euclidean space.

sonic boom

vorticity in wake

Figure 8: Examples of phenomena with strong anisotropic features concentrated in small regions of the domain: shock waves (left), and vorticity (right).

2.1 Metric tensors in mesh adaptation

A metric tensor field of Ω is a Riemannian metric space denoted by $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$, where $\mathcal{M}(\mathbf{x})$ is a 3×3 symmetric positive definite matrix. Taking this field at each vertex $\mathbf{x}_i$ of a mesh $\mathcal{H}$ of Ω defines the discrete field $\mathcal{M}_i = \mathcal{M}(\mathbf{x}_i)$. If N denotes the number of vertices of $\mathcal{H}$, the linear discrete metric field is denoted by $(\mathcal{M}_i)_{i=1\dots N}$. As $\mathcal{M}(\mathbf{x})$ and $\mathcal{M}_i$ are symmetric definite positive, they can be diagonalized in an orthonormal frame, such that

$$\mathcal{M}(\mathbf{x}) = {}^t\mathcal{R}(\mathbf{x})\Lambda(\mathbf{x})\mathcal{R}(\mathbf{x}) \text{ and } \mathcal{M}_i = {}^t\mathcal{R}_i\Lambda_i\mathcal{R}_i,$$

where $\Lambda(\mathbf{x})$ and Λ_i are diagonal matrices composed of strictly positive eigenvalues $\lambda(\mathbf{x})$ and λ_i and $\mathcal{R}$ and $\mathcal{R}_i$ orthonormal matrices verifying ${}^t\mathcal{R}_i = (\mathcal{R}_i)^{-1}$. Setting $h_i = \lambda_i^{-2}$ allows to define the sizes prescribed by $\mathcal{M}_i$ along the principal directions given by $\mathcal{R}_i$. Note that the set of points verifying the implicit equation ${}^t\mathbf{x}\,\mathcal{M}_i\,\mathbf{x} = 1$ defines a unique ellipsoid. This ellipsoid is called the unit-ball of $\mathcal{M}_i$ and is used to represent geometrically $\mathcal{M}_i$.
The two fundamental operations in a mesh generator are the computation of length and volume. The length of an edge $\mathbf{e} = [\mathbf{x}_i, \mathbf{x}_j]$ and the volume of an element K are continuously evaluated in $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ by:

$$\ell_{\mathcal{M}}(\mathbf{e}) = \int_0^1 \sqrt{{}^t\mathbf{e}\;\mathcal{M}(\mathbf{x}_i + t\,\mathbf{e})\;\mathbf{e}}\,\mathrm{d}t \text{ and } |K|_{\mathcal{M}} = \int_K \sqrt{\det(\mathcal{M}(\mathbf{x}))}\,\mathrm{d}\mathbf{x}$$

From a discrete point view, the metric field needs to be interpolated [32] to compute approximate length and volume. For the volume, we consider a linear interpolation of $(\mathcal{M}_i)_{1\dots N}$ and the following edge length approximation is used:

$$|K|_{\mathcal{M}} \approx \sqrt{\det\left(\frac{1}{4}\sum_{i=1}^{4}\mathcal{M}_i\right)}\,|K| \text{ and } \ell_{\mathcal{M}}(\mathbf{e}) \approx \sqrt{{}^t\mathbf{e}\,\mathcal{M}_i\,\mathbf{e}}\,\frac{r-1}{r\ln(r)}, \qquad (1)$$

where $|K|$ is the Euclidean volume of K and r stands for the ratio $\sqrt{{}^t\mathbf{e}\,\mathcal{M}_i\,\mathbf{e}}/\sqrt{{}^t\mathbf{e}\,\mathcal{M}_j\,\mathbf{e}}$. The approximated length arises from considering a geometric approximation of the size variation along end-points of $\mathbf{e}$: $\forall t \in [0,1]\; h(t) = h_i^{1-t}\,h_j^t$.
The task of the adaptive mesh generator is then to generate a unit-mesh with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\boldsymbol{\Omega}}$. A mesh is said to be unit when it is only composed of unit-volume elements and

unit-length edges. Practically, these two requirements are combined in a quality function computed in the metric field. A mesh $\mathcal{H}$ is unit with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ when each tetrahedron $K \in \mathcal{H}$ defined by its list of edges $(\mathbf{e}_i)_{i=1\ldots6}$ verifies:

$$\forall i \in [1,6], \quad \ell_{\mathcal{M}}(\mathbf{e}_i) \in \left[\frac{1}{\sqrt{2}}, \sqrt{2}\right] \quad \text{and} \quad Q_{\mathcal{M}}(K) \in [\alpha, 1] \;\; \text{with} \;\; \alpha > 0\,, \tag{2}$$

with:

$$Q_{\mathcal{M}}(K) = \frac{36}{3^{\frac{1}{3}}} \frac{|K|_{\mathcal{M}}^{\frac{2}{3}}}{\sum_{i=1}^{6} \ell^2_{\mathcal{M}}(\mathbf{e}_i)} \in [0,1]. \tag{3}$$

A classical and admissible value of α is 0.8. This value arises from some discussions on the possible tessellation of $\mathbb{R}^3$ with unit-elements [57]. The $\sqrt{2}$ and $1/\sqrt{2}$ factors to control the length of edges are used to avoid to cycle during the remeshing step. If a long edge is split, the two new edges should not be considered too small, in order to avoid an infinite sequence of insertions and collapses.

There exist a large set of adaptive mesh generators that uses a metric-tensor as an input to generate anisotropic meshes. Let us cite `Bamg` [43] and `BL2D` [49] in 2D, `Yams` [29] for discrete surface mesh adaptation and `EPIC` [72], `Feflo.a` [62], `Forge3d` [22], `Refine 1/2` [46], `Gamanic3d` [34], `MadLib` [21], `MeshAdap` [51], `Mmg3d` [25], `Mom3d` [83], `Tango` [16], `LibAdaptivity` [74] and `Pragmatic` [79] in 3D.

2.2 Techniques for enhancing robustness and performance

The metric field provided has a direct, albeit complex, impact on the quality of the resulting mesh. A smooth and well-graded metric field makes the generation of the anisotropic mesh generation easier and generally improves the final quality. We consider two techniques that tend to give a substantial positive impact on the quality of the resulting mesh: The **anisotropic mesh gradation** tends to smooth the metric field, while the **Log-Eucidean** interpolation allows to properly define metric tensors interpolation, thereby preserving the anisotropy even after a numerous number of interpolations.

Anisotropic mesh gradation. The mesh gradation is a process that smoothes the initial metric field that is generally noisy as it is derived from discrete data. Gradation strategies for anisotropic meshes are available in [4, 50]. From a continuous point of view, the mesh gradation process consists in verifying the uniform continuity of the metric field:

$$\forall (\mathbf{x},\mathbf{y}) \in \Omega^2 \;\; \|\mathcal{M}(\mathbf{y}) - \mathcal{M}(\mathbf{x})\| \leq C \|\mathbf{x} - \mathbf{y}\|_2,$$

where C is a constant and $\|.\|$ a matrix norm. This requirement is far more complex that imposing only the continuity of $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$. From a practical point of view, this done that by ensuring that for all couples $(\mathbf{x}_i, \mathcal{M}_i)$ defined on $\mathcal{H}$ verify:

$$\forall (\mathbf{x}_i, \mathbf{y}_j) \in \mathcal{H}^2 \quad \mathcal{N}(\|\mathbf{x}_i - \mathbf{y}_j\|_2)\,\mathcal{M}_i \cap \mathcal{M}_j = \mathcal{M}_j \text{ and } \mathcal{N}(\|\mathbf{x}_i - \mathbf{y}_j\|_2)\,\mathcal{M}_j \cap \mathcal{M}_i = \mathcal{M}_i,$$

where $\mathcal{N}(.)$ is a matrix function defining a growth factor and $\cap$ is the classical metric intersection based on simultaneous reduction [32]. This standard algorithm has $O(N^2)$ complexity. Consequently, less CPU-intensive correction strategies need to be devised; we refer to [4] for some suggestions. Note that bounding the number of corrections to a fixed value is usually sufficient to correct the metric field near strongly anisotropic areas as the shocks. Two options are considered giving either an isotropic growth or an anisotropic growth:

$$\mathcal{N}(d_{ij})\,\mathcal{M}_i = \begin{pmatrix} \eta_1(d_{ij})\,\lambda_1 & & \\ & \eta_2(d_{ij})\,\lambda_2 & \\ & & \eta_3(d_{ij})\,\lambda_3 \end{pmatrix}$$

with

$$(i)\ \eta_k(d_{ij}) = (1 + \sqrt{{}^t\mathbf{e}_{ij}\,\mathcal{M}_i\,\mathbf{e}_{ij}}\ \log(\beta))^{-2} \ \text{ or } \ (a)\ \eta_k(d_{ij}) = (1 + \lambda_k\, d_{ij}\ \log(\beta))^{-2}, \tag{4}$$

where $d_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|_2$, $\mathbf{e}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and β the gradation parameter > 1. The isotropic growth is given by law (i) while the anisotropic by law (a). Note that (i) is identical for all directions, contrary to anisotropic law (a) that depends on each eigenvalue along its principal direction. In the sequel, we use the gradation to smooth the transition between the various metric fields: surface and volume, surface and boundary layers.

Log-Euclidean framework and applications. After each point insertion or during the computation of edge-lengths, a metric field must be interpolated. Interpolation schemes based on the simultaneous reduction [32] lack several desirable theoretical properties. For instance, the unicity is not guaranteed. A framework introduced in [10] proposes to work in the logarithm space as if one were in the Euclidean one. Consequently, a sequence of n metric tensors can be interpolated in any order while providing a unique metric. Given a sequence of points $(\mathbf{x}_i)_{i=1\ldots k}$ and their respective metrics $\mathcal{M}_i$, then the interpolated metric in $\mathbf{x}$ verifying

$$\mathbf{x} = \sum_{i=1}^{k} \alpha_i\,\mathbf{x}_i,\ \text{with}\ \sum_{i=1}^{k} \alpha_i = 1,\ \text{is}\ \mathcal{M}(\mathbf{x}) = \exp\left(\sum_{i=1}^{k} \alpha_i \ln(\mathcal{M}_i)\right). \tag{5}$$

On the space of metric tensors, logarithm and exponential operators are acting on metric's eigenvalues directly:

$$\ln(\mathcal{M}_i) = {}^t\mathcal{R}_i\ \ln(\Lambda_i)\,\mathcal{R}_i \quad \text{and} \quad \exp(\mathcal{M}_i) = {}^t\mathcal{R}_i\ \exp(\Lambda_i)\,\mathcal{R}_i.$$

Numerical experiments confirm that using this framework during interpolation allow to preserve the anisotropy. Note that the evaluation of length given by (1) corresponds to the Log-Eucldiean interpolation between the two metrics of the edge extremities.

2.3 Metric-based error estimates

From the previous concepts, metric-based error estimates are well suited for the generation of anisotropic meshes. We focus on this set of estimates in the sequel. We then describe in more details the case of the interpolation error as it is the easiest to implement.

2.3.1 A (quick) review of metric-based estimates

A first set of methods is based on the minimization of the interpolation error of one or several sensors depending on the CFD solution [2, 5, 19, 26, 30, 44, 62, 86]. Given a numerical solution W_h, a solution of higher regularity $R_h(W_h)$ is recovered, so that the following interpolation error estimate [20, 58] holds:

$$\|R_h(W_h) - \Pi_h R_h(W_h)\|_{L^p} \le N^{-\frac{2}{3}} \left(\int_\Omega \det\left(|H_{R_h(W_h)}|\right)^{\frac{p}{2p+3}}\right)^{\frac{2p+3}{3p}} \tag{6}$$

where $H_{R_h(W_h)}$ is the Hessian of the recovered solution and N an estimate of the desired number of nodes, and Π_h the piecewise linear interpolate of a function. If anisotropic mesh prescription is naturally deduced in this context, interpolation-based methods do not take into account the PDE itself. However, in some simplified context and assumptions (elliptic PDE, specific recovery operator), we have:

$$\|W - W_h\| \leq \frac{1}{1-\alpha} \|R_h(W_h) - \Pi_h R_h(W_h)\| \text{ with } \alpha > 1\,,$$

so that good convergence to the exact solution may be observed [61]. Indeed, if $R_h(W_h)$ is a better approximate of W in the following meaning:

$$\|W - W_h\| \ \leq \ \frac{1}{1-\alpha} \|R_h(W_h) - W_h\| \ \text{ where } \ 0 \leq \alpha < 1,$$

and if the reconstruction operator R_h has the property:

$$\Pi_h R_h(W_h) = W_h,$$

we can then bound the approximation error of the solution by the interpolation error of the reconstructed function $R_h(W_h)$:

$$\|W - W_h\| \leq \frac{1}{1-\alpha} \|R_h(W_h) - \Pi_h R_h(W_h)\|\,.$$

Note that from a practical point of view, $R_h(W_h)$ is never recovered, only its first and second derivatives are estimated. Standard recovery techniques include least-square, L^2-projection, green formula or the Zienkiewicz-Zhu recovery operator. A numerical review of H_R operators is given in [85].

A second set of methods tends to couple adaptivity with the assessment of the numerical prediction of the flow. Goal-oriented optimal methods [39, 46, 59, 77, 87] aims at minimizing the error committed on the evaluation of a scalar functional. A usual functional is the observation of the pressure field on an observation surface γ:

$$|j(W) - j_h(W_h)| \quad \text{with} \quad j(W) = \int_\gamma \left(\frac{p - p_\infty}{p_\infty} \right)^2,$$

where W and W_h are the solution and the numerical solution of the set of PDEs, respectively. They do take into account the features of the PDE, through the use of an adjoint state that gives the sensitivity of W to the observed functional j. In order to solve the goal-oriented mesh optimization problem, an *a priori* analysis depending on the numerical scheme is needed to restrict to the main asymptotic term of the local error, see [15] for the Euler equations. If a super-convergence of $|j(W) - j_h(W_h)|$ may be observed in some cases [40, 41], goal-oriented optimal methods are specialized for a given output, and in particular do not provide a convergent solution field. Indeed, the convergence of $\|W - W_h\|$ is not predicted. In addition, if the observation of multiple functionals is possible (by means of multiple adjoint states), the optimality of the mesh and the convergence properties of the approximation error may be lost.

In each case, the aforementioned adaptive strategies address specifically one goal. Consequently, it is still a challenge to find an adaptive framework that encompass all the desired requirements: anisotropic mesh prescription, asymptotic optimal order of convergence, assessment of the convergence of the numerical solution to the continuous one, control of multiple functionals of interest, ... One current field of research is based on the design of a norm-oriented or multi-functionals mesh adaptation, which takes into account the PDE features, and produces an approximate solution field which does converge to the exact one. This is done by estimating a residual term $\Pi_h W - W_h$. This term naturally arises when the functional of interest is the norm $\|\Pi_h W - W_h\|_{L^2}$. The estimate is then used as a functional with the standard goal-oriented approach. To do so, it is necessary to derive some correctors that estimate the implicit error. This approach requires the knowledge of the numerical method at hands along with an adjoint solver corresponding to set of equations being solved. Consequently, we can observed functional of interest that is the difference between the exact and the numerical solutions. In addition, multiple functionals of interest can be observed simultaneously. For instance, the norm-functional can be:

$$(\mathrm{drag}(W) - \mathrm{drag}(W_h))^2 + (\mathrm{lift}(W) - \mathrm{lift}(W_h))^2.$$

By linearizing the right-hand side (RHS), we see that the estimate (corrector) for the norm-functional depends only of $\Pi_h W - W_h$ and produces a single left-hand-side for the goal-oriented estimation. More details on these approaches can be found in [18, 42, 60].

2.4 Controlling the interpolation error

Controlling the linear interpolation error of a given flow field allows to derive a very simple anisotropic metric-based estimate. Interpolation estimate is the first introduced in the pioneering work [19] by equi-distributing the interpolation error in $\mathbf{L}^\infty$ norm. Here, we prefer to control the $\mathbf{L}^p$ norm of the interpolation error. Such control allows to recover the order of convergence of the scheme for flows with shocks and to capture all the scales of the numerical solution [61]. Given a numerical solution W_h (density, pressure, Mach number, ...), the point-wise metric tensor minimizing (6) is given by:

$$\mathcal{M}_{\mathbf{L}^p}(W_h) = \det(|H_R(W_h)|)^{\frac{-1}{2p+3}}\, |H_R(W_h)|, \tag{7}$$

where $|H_R(W_h)|$ is deduced from $H_R(W_h)$ by taking the absolute value of the eigen-values of $H_R(W_h)$. In the sequel, the interpolation error is controlled in $\mathbf{L}^2$ norm exclusively, while the H_R operator is based on the double $\mathbf{L}^2$ projection [85]. For the numerical examples, we will use the complexity to control the level of accuracy. The complexity is defined by $C(\mathcal{M}) = \int_\Omega \sqrt{\det(\mathcal{M})}$. Imposing a complexity of N leads to the following scaling of the metric:

$$\mathcal{M}_{\mathbf{L}^p}(W_h, N) = \left(\frac{N}{\int_\Omega \det(|H_R(W_h)|)^{\frac{p+1}{2p+3}}} \right) \det(|H_R(W_h)|)^{\frac{-1}{2p+3}}\, |H_R(W_h)|. \tag{8}$$

For time dependent problems, we use an extension of the multi-scale approach [7]. The process may be summarized as follows. The whole time frame $[0, t_f]$ is split in n_t sub-intervals:

$$[0, t_f] = \bigcap_{i=1}^{n_t} [t_i, t_{i+1}], \text{ with } t_1 = 0 \text{ and } t_{n_t} = t_f.$$

Then, the main idea consists in deriving n_t meshes $(\mathcal{H}_i)_{i=1,n_t}$ that minimize the interpolation error on the solution u defined on Ω:

$$\text{Find } (\mathcal{H}^i_{opt})_{i=1,n_t} = \min \sum_{i=1}^{n_t} \int_{t_i}^{t_{i+1}} \int_\Omega |W - \Pi_h W|^p \,\mathrm{d}\Omega\, \mathrm{d}t \text{ for all } i \in [1, n_t]. \tag{9}$$

The solution of this problems gives a sequence of metric tensor fields $(\mathcal{M}_i)_{i=1,nt}$ for each sub interval $[t_i, t_{i+1}]$. The continuous problem is then solved using a calculus of variations. From a practical point of view, on a time interval $[t_i, t_{i+1}]$, the flow solver outputs a sequence of solutions every $\Delta t = (t_{i+1} - t_i)/N$. From this sequence, a maximal or mean hessian $\widetilde{H}_i$ is recovered [7] accounting for the error for the sub-window time frame. Then, once all $\widetilde{H}_i$ are recovered, a global normalization is applied for the whole time frame $[0, t_f]$ to derive $(\mathcal{M}_i)_{i=1,nt}$, see Figure 26.

A detailed review of metric-based estimates for steady and unsteady problems can be found in [6].

2.5 Geometric estimate for surfaces

Controlling the deviation to a surface has been studied in previous works, see [13, 28, 31] for anisotropic remeshing. We recall that the surface remeshing is done by considering only discrete data, either inherited by the CAD or recovered directly from the discrete mesh. Prior to surface remeshing, normals and tangents are then assigned to each boundary point. We denote by $\mathbf{n}_i$ the normal of the vertex $\mathbf{x}_i$. As in [28], a quadratic surface model is computed locally around a surface point $\mathbf{x}_i$. Starting from the topological neighbors of $\mathbf{x}_i$, the coordinates of each point are mapped onto the local orthonormal Frenet frame $(\mathbf{u}_i, \mathbf{v}_i, \mathbf{n}_i)$ centered in $\mathbf{x}_i$. Vectors $(\mathbf{u}_i, \mathbf{v}_i)$ lie in the orthogonal plane to $\mathbf{n}_i$. We denote by $(u_j, v_j, \sigma_j) = ({}^t\mathbf{x}_j.\mathbf{u}_i, {}^t\mathbf{x}_j.\mathbf{v}_i, {}^t\mathbf{x}_j.\mathbf{n}_i)$ the new coordinates of vertex $\mathbf{x}_j$. $\mathbf{x}_i$ is set as the new origin so that $(u_i, v_i, \sigma_i) = (0, 0, 0)$. The surface model consists in computing by a least squares approximation a quadratic surface:

$$\sigma(u, v) = au^2 + bv^2 + cuv, \text{ where } (a, b, c) \in \mathbb{R}^3. \tag{10}$$

The least squares problem gives the solution to $\min_{(a,b,c)} \sum_j |\sigma_j - \sigma(u_j, v_j)|^2$, where j is the set of neighbors of $\mathbf{x}_i$. Note that 3 neighbors points are necessary to recover the surface model. Finally, if the degree of $\mathbf{x}_i$ is d and the linear system is:

$$A\,X \,=\, B \iff \begin{pmatrix} u_1^2 & v_1^2 & u_1 v_1 \\ \vdots & \vdots & \vdots \\ u_d^2 & v_d^2 & u_d v_d \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_d \end{pmatrix}.$$

The least square formulation consists in solving ${}^tA\,A = {}^tA\,B$. From this point, one may applied the surface metric given in [28]. We propose here a simplified version. We can first remark that the orthogonal distance from the plane $\mathbf{n}_i^\perp$ onto the surface is given by $\sigma(u, v)$ by definition. The trace of $\sigma(u, v)$ on $\mathbf{n}_i^\perp$ is a function that gives directly the distance to the surface. The 2D surface metric $\mathcal{M}_S^{2D}$ such that the length $\ell_{\mathcal{M}_S^{2D}}((u, v))$ is constant equal to ε is easy to find starting from the diagonalization of the quadratic function (10). Geometrically, it consists in finding the maximal area metric included in the level-set ε of the distance map. We assume that $\mathcal{M}_S^{2D}$ admits the following decomposition:

$$\mathcal{M}_S^{2D} = (\bar{\mathbf{u}}_S, \bar{\mathbf{v}}_S) \begin{pmatrix} \lambda_{1,S} & 0 \\ 0 & \lambda_{2,S} \end{pmatrix} {}^t(\bar{\mathbf{u}}_S, \bar{\mathbf{v}}_S), \text{ with } (\bar{\mathbf{u}}_S, \bar{\mathbf{v}}_S) \in \mathbb{R}^{2\times 2}.$$

If we want to achieve the same error as the initial mesh, we compute $\varepsilon = \min_j |\sigma(u_j, v_j)|$ among the neighbors of $\mathbf{x_i}$. The anisotropic 2D metric achieving an ε error becomes:

$$\mathcal{M}_S^{2D}(\varepsilon) = \frac{1}{\varepsilon} \mathcal{M}_S^{2D}.$$

The final 3D surface metric in $\mathbf{x}_i$ is:

$$\mathcal{M}_S(\varepsilon) = (\mathbf{u}_S, \mathbf{v}_S, \mathbf{n}_i) \begin{pmatrix} \dfrac{\lambda_{1,S}}{\varepsilon} & 0 & \\ 0 & \dfrac{\lambda_{2,S}}{\varepsilon} & 0 \\ 0 & 0 & h_{max}^{-2} \end{pmatrix} {}^t(\mathbf{u}_S, \mathbf{v}_S, \mathbf{n}_i), \tag{11}$$

$$\text{with} \left\{ \begin{array}{l} \mathbf{u}_S = \bar{\mathbf{u}}_S(1)\mathbf{u}_i + \bar{\mathbf{u}}_S(2)\mathbf{v}_i, \\ \mathbf{v}_S = \bar{\mathbf{v}}_S(1)\mathbf{u}_i + \bar{\mathbf{v}}_S(2)\mathbf{v}_i. \end{array} \right.$$

The parameter h_{max} is initially chosen very large (e.g. 1/10 of the domain size). This normal size is corrected during various steps. A first anisotropic gradation using (4)(i) is applied on surface edges only. The surface metric is then intersected with any computation metrics as given by (7). These two steps set automatically a proper element size in the normal direction. Note different local surface estimates can be derived depending on the local information available, see [88].

During the mesh adaptation process, the previous procedure is not applied independently on each current mesh to be adapted. On the contrary, the surface metric is computed once on a fixed background mesh. This metric is then interpolated on each adapted mesh in the course of the iterative process. This tends to maintain a consistent gap with respect to the true geometry.

2.6 Boundary layers metric

Boundary layers mesh generation has been devised to capture accurately the speed profile around a body during a viscous simulation. The width of the boundary layer depends on the local reynolds number [53]. So far, the generation of the boundary layer grids has been carried out by an extrusion of the initial surface along the normals to the surface or by local modification of the mesh [68]. Note that using the normals as sole information requires several enrichments to obtain a smooth layers transition on complex surfaces [14]. In this chapter, we consider a simple approach that is naturally compatible with anisotropic adaptation procedures. The idea consists in representing the boundary layer mesh by a continuous metric field.

The distance to the body is computed using classical algorithms of level-set methods [53]. This step can be done quickly and has generally a complexity of $O(N \ln(N))$ where N is the number of points in the current mesh. (Furthermore, note that from a practical point of view, this function is evaluated only in the vicinity of the body). To control the size in the tangential directions, a metric is recovered from the current surface mesh or a background mesh. It takes advantage of the Log-Euclidean framework. Starting from an elements $(K)_{P\in K}$ of vertex P, the unique surface metric tensor $\mathcal{M}_K$ (for which K is unit) is computed by solving the following 6×6 linear system:

$$(S) \left\{ \begin{array}{l} \ell^2_{\mathcal{M}_K}(\mathbf{e_1}) = 1 \\ \dots \\ \ell^2_{\mathcal{M}_K}(\mathbf{e_6}) = 1\,. \end{array} \right. \tag{12}$$

where $(\mathbf{e}_i)_{i=1,6}$ are elements edges. (S) has a unique solution as long as the volume of K is not null. The logarithm of each metric is computed so that a classical Euclidean mean weighted by the elements' area is done. Finally, the body point metric $\mathcal{M}_P$ is mapped back using the exponential operator:

$$\mathcal{M}_P = \exp\left(\frac{\sum_{P\in K} |K| \ln(\mathcal{M}_K)}{\sum_{P\in K} |K|} \right).$$

The final boundary layers metric is based for a continuous exponential law of the form $h_0 \exp(\alpha\phi(.))$, where h_0 is the initial boundary layer size and α the growing factor. For a volume point $\mathbf{x}_i$, the boundary layers metric depends on the body point P_i for which the minimum distance is reached. The following operations conclude this step:

1. Compute the local Frenet frame $(\mathbf{u}_i, \mathbf{v}_i, \nabla\Phi(\mathbf{x}_i))$ associated with $\nabla\Phi(\mathbf{x}_i)$
2. Set the size in the normal direction to $h_{\mathbf{n}_i} = h_0 \exp(\alpha\, \Phi(\mathbf{x}_i))$, the sizes in the orthogonal plane to:
$$h_{\mathbf{u}_i} = \left({}^t\mathbf{u}_i\, \mathcal{M}_{P_i}\, \mathbf{u}_i\right)^{-2} \text{ and } h_{\mathbf{v}_i} = \left({}^t\mathbf{v}_i\, \mathcal{M}_{P_i}\, \mathbf{v}_i\right)^{-2},$$
3. The final metric is given by:
$$\mathcal{M}_{bl}(\mathbf{x}_i) = {}^t(\mathbf{u}_i, \mathbf{v}_i, \nabla\Phi(\mathbf{x}_i)) \begin{pmatrix} h_{\mathbf{u}_i}^{-2} & & \\ & h_{\mathbf{v}_i}^{-2} & \\ & & h_{\mathbf{n}_i}^{-2} \end{pmatrix} (\mathbf{u}_i, \mathbf{v}_i, \nabla\Phi(\mathbf{x}_i)). \tag{13}$$

The key idea is again to simplify the coupling by using only metric tensor fields. Indeed, taking all together the viscous and un-viscous contributions simply consist in intersecting the corresponding metric tensor fields.

3 Algorithms for generating anisotropic meshes

This section describes the local operators used to adapt the mesh once one or several tensor fields are provided on input. We then describe additional operators used to optimize the mesh and to guarantee an optimal time step for unsteady simulations.

3.1 Insertion and collapse

To generate a unit-mesh in a given metric field $(\mathcal{M}_i)_{i=1\ldots N}$, two operations are recursively used: edge collapse and point insertion on edge.
The starting point for the insertion of a new point on an edge $\mathbf{e}$ is the shell of $\mathbf{e}$ composed of all elements sharing this edge. Each element of the shell is then divided into two new elements. The new point is accepted if each new tetrahedron has a positive volume. When a point is inserted on an boundary edge, either a linear approximation of the surface is used or a query to the CAD. The newly inserted point is created at the mid-edge point in the metric. To compute it, we first evaluate the size of the current edge with respect to the two end-points $(A, \mathcal{M}_A)$ and $(B, \mathcal{M}_B)$:
$$\ell_{\mathcal{M}_A} = \sqrt{{}^tAB\, \mathcal{M}_A\, AB} \text{ and } \ell_{\mathcal{M}_B} = \sqrt{{}^tAB\, \mathcal{M}_B\, AB}.$$
If $\ell_{\mathcal{M}_A}$ equals $\ell_{\mathcal{M}_B}$, the mid-edge point in the metric is the geometric mid-point $\frac{1}{2}(A+B)$. When they differ, we need to solve the non-linear problem in $t \in [0,1]$ arising for the length approximation of (1):
$$\text{Find } t \text{ such that} \frac{1}{2} = \ell_{\mathcal{M}_A} \frac{r^t - 1}{\log(r)} \text{ with } r = \frac{\ell_{\mathcal{M}_B}}{\ell_{\mathcal{M}_A}}. \tag{14}$$
We use a dichotomy approach to solve (14), the mid-point is then $(1-t)\,A + t\,B$.
The edge collapse starts from the ball of the vertex to be deleted. Again, for the deletion of points inside the volume, the only possible rejection is the creation of a negative volume element. A special care is also required to avoid the creation of an element that already exists, see Figure 9

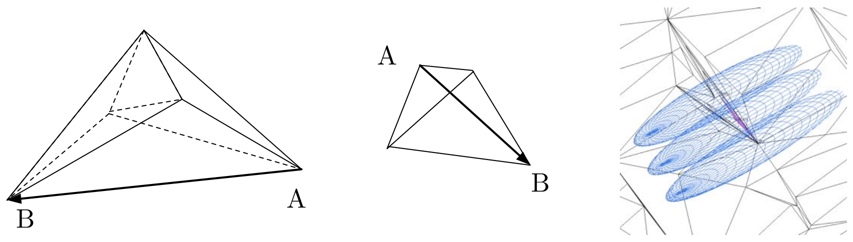

Figure 9: Left and middle, volume and surface collapse of edge AB leading to the creation of an element that already exists . Right, Example where an edge is recursively refined to get a unit-length without checking the length requirement in the edge's orthogonal direction; the configuration may lead to edges acting as a barrier for future refinement.

(left and middle). The rejections are more complicated in the case of a surface point. We first avoid each collapse susceptible to modify the topology of the object. This is simply done by assigning an order on each surface point types: corner, ridge (line), inside surface. The collapse can also be rejected if the normal deviation between old and new normals becomes too large. Currently, if $\mathbf{n}$ denotes the normals to an old face, we allow the collapse if each new normal $\mathbf{n}_i$ verifies ${}^t\mathbf{n}\,\mathbf{n}_i > \cos(\pi/4)$. Note that the control to the surface deviation is given by the surface metric and so it does not need to be handled directly in the collapse operation.
With these operations, the core of the adaptive algorithm consists in scanning each edge of the current mesh and, depending on its length, creating a new point on the edge or collapsing the edge. An edge is declared too small or too large according to the bounds given in (2). Without any more considerations, such adaptive mesh generator is known to be not efficient and to require a lot of CPU consuming optimizations as point smoothing and edge swapping. This inefficiency is simply due to the locality of these operations. Comparing to an anisotropic Delaunay kernel [25], when an edge needs to be refined, the metric lengths along the orthogonal directions are controlled by the creation of the cavity. Consequently, in one shot, the area of refinement must be large. With the present approach, the size is controlled along one direction only (along the edge being scanned). Consequently, one can reach intractable configurations where the same initial edge is refined successively to get the desired size whereas the sizes in the other directions get worse. A typical configuration is depicted in Figure 9 (right).

A simple way to overcome this major drawback is to use the quality function (3) together with the unit-length check. This supplementary check can be done at no cost since a lot of information can be re-used: the volume is already computed, as well as the length of the edges. By simply computing the quality function, we give to these operators the missing information on the orthogonal directions of the current scanned edge. For the collapse, to decide which vertex is deleted from the edge, the qualities of the two configurations are compared and the best one is kept. For an optimal performance, two parameters are added in the rejection cases of insertion and collapse: a relative quality tolerance $q_r \geq 1$ and a global quality tolerance q_a. Indeed, it seems particularly interesting not to try to implement a full descent direction by imposing the quality to increase on each operation. We prefer to allow the quality to decrease in order to get out of possible local minima. Consequently, a new configuration of elements is accepted if:

$$q_r\, Q_{\mathcal{M}}^{ini} \leq Q_{\mathcal{M}}^{new} \quad \text{and} \quad Q_{\mathcal{M}}^{new} < q_a,$$

where $Q_{\mathcal{M}}^{ini}$ is the worse element quality of the initial configuration and $Q_{\mathcal{M}}^{new}$ is the worse quality of the new configuration. This approach is similar to the simulated annealing global optimization technique [47]. Note that the current version does not fully implement the classical metropolis algorithm where the rejection is based on a random probability. To ensure the convergence of the algorithm, the relative tolerance q_r is decreased down to 1 after each pass of insertions and collapses. At the end of the process, the absolute tolerance q_a is set up to the current worse quality among all elements.

3.2 Optimizations and enhancement for unsteady simulations

In addition to the quality-driven insertion and collapse, we use standard anisotropic mesh optimization techniques such as edges and faces swaps and point smoothing in order to increase the level of anisotropy and the quality of the mesh. By improving the overall quality, they usually improve the stability of the flow solver as well. For unsteady simulations, we add an additional control parameter in order to ensure that an optimal time step is provided in the adaptive mesh.

Swaps of edges and faces are standard mesh modifications operators, see [27, 32]. In the context of anisotropic remeshing, theses operators are simply monitored by anisotropic quality (3). Once the topological and geometrical validity of a swap is verified (positive volume and valid new configurations), it is actually performed only if the quality of the new configuration is strictly lower that the initial quality. We use an improvement factor $q_r = 0.95$ for all the numerical examples.

The point smoothing is also a popular simple operator [32]. It consists in computing a new optimal position of a vertex to improve the quality of the surrounding elements. The main difficulty is the computation of the optimal position. In our case, we want to optimize the length distribution as well. Consequently, for an edge PP_i with metric $\mathcal{M}(P)$ and $\mathcal{M}(P_i)$, the optimal point position of P is approximated in a Riemannian way by computing :

$$\theta = 1 - \log\left(\frac{\ell_{\mathcal{M}}(P)}{\ell_{\mathcal{M}}(P) - \log(r)}\right)\frac{1}{\log(r)} \text{ with } \left\{\begin{array}{lcl} \ell_{\mathcal{M}}(P) & = & \sqrt{PP_i\mathcal{M}(P)PP_i}, \\ \ell_{\mathcal{M}}(P_i) & = & \sqrt{PP_i\mathcal{M}(P_i)PP_i}, \\ r & = & \ell_{\mathcal{M}}(P)/\ell_{\mathcal{M}}(P_i). \end{array}\right.$$

The formula arises from seeking the optimal size to get a unit-edge length along PP_i:

$$\int_0^\theta \ell_{\mathcal{M}}(P)^{t-1}\ell_{\mathcal{M}}(P_i)^{-t}\mathrm{d}t = 1.$$

Then the optimal position for P from P_i is :

$$P_{opt_i} = P + \theta\, P_iP.$$

This procedure is repeated with all the neighboring vertices of P:

$$P_{opt} = \alpha P + \frac{1-\alpha}{n_P}\left(\sum_{i=1}^{n_P} P_{opt_i}\right) \text{ with } \alpha \in [0,1]$$

If P_{opt} generates positive volume elements and improves the final quality, P is moved to this new position. In case of rejection, a greater value of α is considered starting with $\alpha = 0.2$. Note that the metric of P is interpolated at the new position to evaluate the new quality. When a surface point is moved, it is also projected back to the surface and the surface deviation is check in a similar way as for the insertion and collapse operators.

For unsteady simulations, the mesh adaptation becomes critical as the CPU time of the simulation depends on the quality of the worse element. Indeed, when an explicit time stepping is used, the minimal time step governs the speed of the simulation. Consequently, the minimal size (or height) generated during the remeshing process may impact drastically the CPU time.

If the generated size if 0.01 of the minimal target, then the while CPU time will be multiplied by 100. To overcome this issue, we add an additional control to the quality based on the height of the tetrahedra. We start from the definition of the minimal height of a tetrahedron:

$$h^2 = \frac{1}{3}\,\frac{V}{S_{max}}, \tag{15}$$

where h is the minimal height, V the volume and S_{max} the maximal area of the faces. For each provided metric, we consider then the regular tetrahedron of side h_1, h_2, h_3, where $(h_i)_i$ are the unit lengths along the eigenvectors of the metric. Then, assuming that the sizes may be in the range $[\frac{1}{\sqrt{2}}\,h_i, \sqrt{2}h_i]$, see (2), we can estimate the global minimal height h_{tar} using (15). A mesh modification is then rejected if the minimal height of the new set of tetrahedra is lower than h_{tar} and the minimal height of the initial set of elements. Numerical experiments have proven that this additional constraint does not have a negative impact on the level of anisotropy while preserving an optimal CPU time step.

4 Adaptive algorithm and numerical illustrations

The previous mesh adaptation strategy is used inside an adaptive loop that couples the error estimations, the mesh adaptation and the flow solver. In this section, we give some additional details on the components that are not relative to the local remeshing. We then first validate the full adaptive approach on a supersonic wing-body configuration and a transonic ONERA m6 wing. For each case, the adapted numerical solution is compared with experiments. Then, we consider the direct sonic boom prediction of a complex aircraft. The adaptive strategy is then applied to the prediction of boundary layer/shock interaction. Finally, we consider unsteady simulations with the double Mach reflection and a blast prediction.

4.1 Adaptive loop

The complete adaptive algorithm for steady simulations is composed of the following steps.

1. Compute the flow field (i.e. converge the flow solution on the current mesh);
2. Compute the metric estimates: surface, volume, boundary layers, etc.
3. Generate a unit mesh with respect to these metric fields;
4. Re-project the surface mesh onto the geometry using the CAD data or a fixed background mesh;
5. Interpolate the flow solution on the new adapted mesh;
6. Goto 1.

For Step 1., two flow solvers have been used in the numerical section. The first one, `FEFLO` [53], works on unstructured grids with finite element discretization of space and edge-based data structures. The Galerkin edge-fluxes are replaced by numerically consistent fluxes, typically given by approximate Riemann solvers (van Leer, Roe, HLLC, ...) with limited variables (van Leer, van Albada, ...). The second flow solver is `WOLF` [5] and it uses a mixed Finite Element/Finite Volume discretization with a MUSCL extrapolation. Both codes have been verified to be second order accurate on smooth flows and second order accurate for flows with shocks by using adaptivity [61, 62]. The flow solvers use an implicit LU-SGS scheme [66, 71]. `FEFLO` is used for simulations 4.4

and 4.5, `WOLF` for simulations 4.2, 4.3 and 4.6. For the unsteady simulations, an explicit time stepping based on Runge-Kutta schemes is used and the explicit control of the height of the tetrahedra of Section 3.2 is activated.

For Step 4., if the surface approximation ε is small enough with respect to the minimal metric size controlling the interpolation error, the simple smoothing procedure usually succeeds to directly move the point onto the geometry. For more complex cases, with boundary layer or when the surface approximation is low, most advanced operators like the cavity-based operators [64] are needed.

For all the simulations, we use a 8-processors 64-bits MacPro with an IntelCore2 chipsets with a clockspeed of 2.8GHz with 32Gb of RAM. The flow solver is multi-threaded while the local remeshing is serial. The final metric field (multi-scale, surface, boundary-layer) is always smoothed by using isotropic gradation law (4)(i), with a parameter of 1.2.

To evaluate the level of anisotropy, we use the anisotropic ratio and anisotropic quotients of an element. Both measures are uniquely defined by computing the metric solution of (12), and then by evaluating the following quantities from its eigenvalues with $h_i = \lambda_i^{-\frac{1}{2}}$:

$$r = \frac{\max_{i=1,3} h_i}{\min_{i=1,3} h_i} \text{ and } q_i = \frac{h_i^3}{h_1\, h_2\, h_3}$$

Anisotropic quotients measure the gain with respect to an isotropic mesh adaptation, in particular, they increase when two anisotropic directions exist.

For all cases, the initial meshes are generated by using either a constrained Delaunay approach for simulations 4.2 and 4.3 and a frontal approach for simulations 4.4, 4.5 and 4.6. Note that only the material described in the previous sections are used. The interpolation error in $\mathbf{L}^2$ norm is used in addition to the surface metric, metric smoothing and boundary layer metric described in Section 2. The algorithm used to generate the meshes exactly fits the procedures given in Section 3.

4.2 A wing-body configuration

The first example is a supersonic flows around the 4th wing-body configuration described in [67]. The planform of the model and the corresponding CAD are depicted in Figure 10. The inflow is a at Mach 1.68 with a lift of 0.15. We observe the pressure below the aircraft at a distance $R = 3.1\,L$ where L is the reference length of the aircraft (here 17.52 cm). Experimental data are available at this distance, see [67]. The adaptive process is based on metric (7) coupled with the surface metric (11) with $\varepsilon = 0.001$. The simulation is composed of 3 steps at the following complexity : 25 000, 50 000 and 75 000 with 5 sub-iterations at a fixed complexity yielding to a total of 15 iterations. We control the interpolation error of the Mach number in $\mathbf{L}^2$ norm. The final mesh is here composed of 283 625 vertices and 1 582 309 tetrahedra. The worst volume quality is 0.05 and the worst surface quality is 0.11 The average anisotropic ratio is 61 and the mean anisotropic quotient is 2711. 92 % and 99.9 % of the volume and surface edges respectively are unit. The total CPU time for this run is 61 mn. This case features 3 strong shocks that are well and early captured by the adaptive process, see Figure 11 for comparisons with experiments.

4.3 Transonic flow around a M6 wing

We consider a flow around the Onera-M6 wing. The initial surface mesh and the CAD are depicted in Figure 12. The flow condition is Mach 0.8395 with an angle of attack of 3.06 degrees. The scope of the example is to validate the interaction between the surface metric controlled with $\varepsilon = 0.001$ and the $\mathbf{L}^p$ metric. For this simulation, the following sequence of complexities

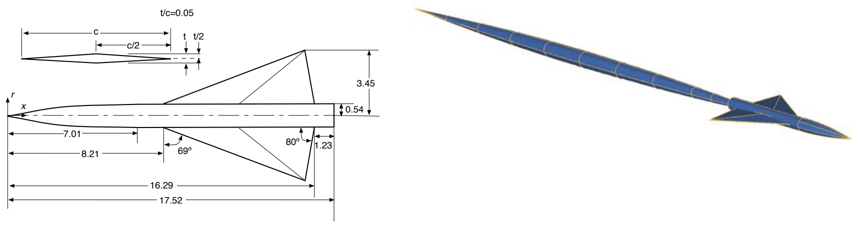

Figure 10: Wing-body example: Left, planform of the wing-body model. Right, CAD of the model equipped with a parabolic sting to emulate experiment apparatus.

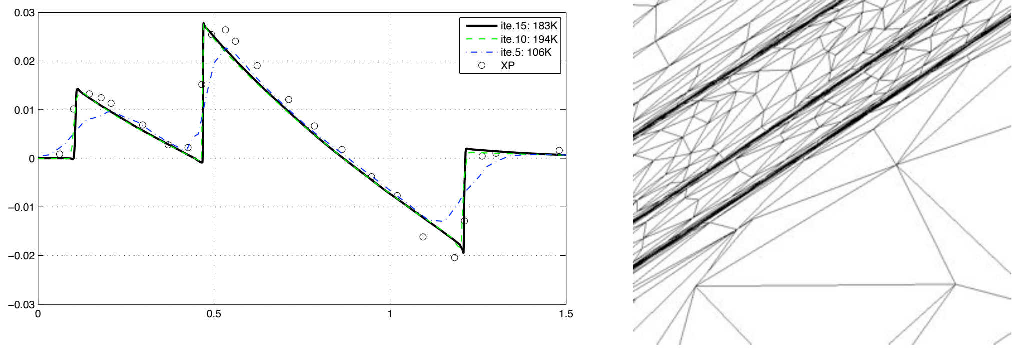

Figure 11: Wing-body example: Left, normalized pressure signature $\frac{p-p_\infty}{p_\infty}$ at $R/L = 3.6$ for the final meshes for each fixed complexity. Right, closer view of the final anisotropic adaptive mesh near the observation line.

for (8) is chosen: 20 000, 40 000 and 80 000, with 5 steps at a fixed complexity. The $\mathbf{L}^2$ norm of the interpolation error of the Mach number is controlled. The final mesh is composed of 222 561 vertices and 1 247 227 tetrahedra with a mean anisotropic ratio of 43 and a mean anisotropic quotient of 1662. The total CPU time for this simulation is 42 mn, with 55 % spent in the flow solver and 45 % in the remeshing, interpolation and error estimate. For the final mesh, the worst quality is 0.11 for the volume and 0.25 for the surface. The strong shocks on the surface of the wing are depicted in Figure 13. C_p extractions along two sections are given in Figure 15. In Figure 14, we observe the anisotropic volume meshes near the wake of the wing and near the shocks on the wing surface. Note that if the shock-dominated features of the flow are perfectly captured, we also capture smooth features as the wing tip vortex, see Figure 16. The amplitude of the flow variables in the wake are 2 orders of magnitude lower than the magnitude in the shock.

4.4 Direct sonic boom simulation

We consider in this example the accurate prediction of the pressure signal below the SSBJ design provided by Dassault-Aviation. The length of the aircraft is $L = 43$ m while the distance of observation from the aircraft is denoted by R. The initial surface mesh is depicted in Figure 17. The aircraft is put in a 10 km domain as depicted in Figure 17 (right). The initial mesh was generated automatically by using an advancing-front technique [55]. The size ratio in the initial

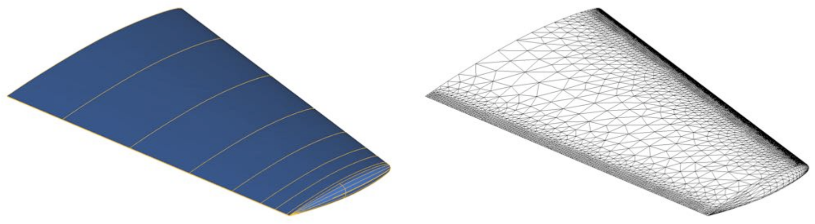
Figure 12: M6 wing example: Left, CAD of the geometry, right, a view of the initial surface mesh of the wing.

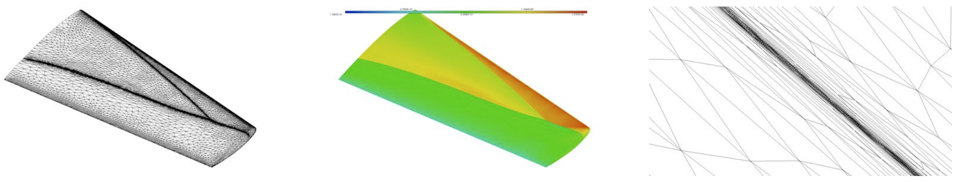
Figure 13: M6 wing example: From left to right, anisotropic surface mesh, Mach iso-values, closer view near the second shock.

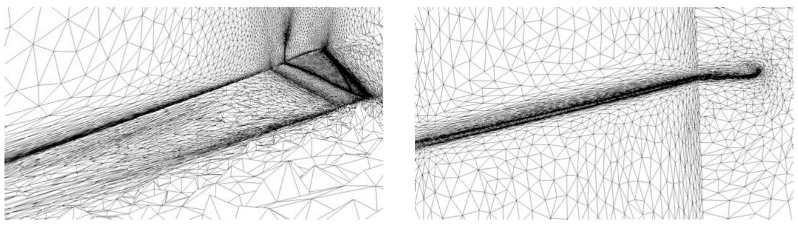
Figure 14: M6 wing example: From left to right, anisotropic capturing of the wake and of the shock, closer view in the wake, closer view around the shock.

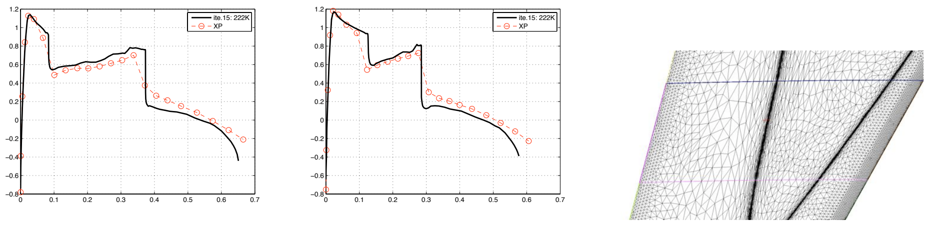

Figure 15: M6 wing example: Left and middle, comparisons between experimental values of $C_p = (p - p_\infty)/q_\infty$ for the second and third upper sections of the wing. Right, closer view of the mesh near upper sections 2 and 3.

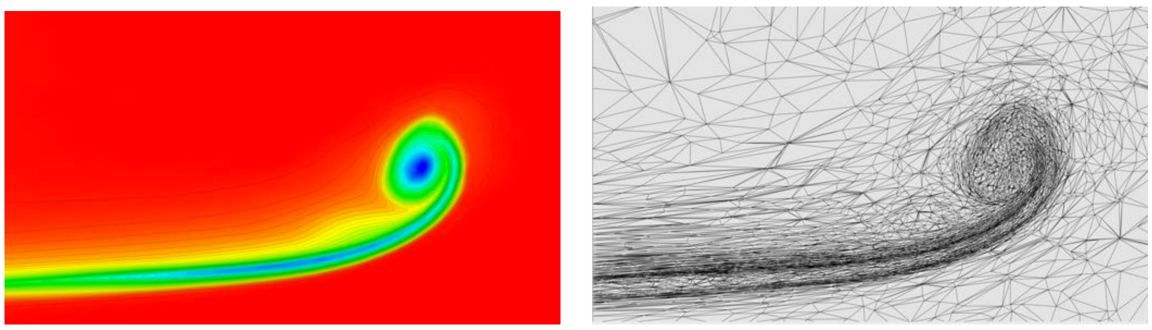
Figure 16: M6 wing example: Wing tip vortex 8 body-length behind the wing.

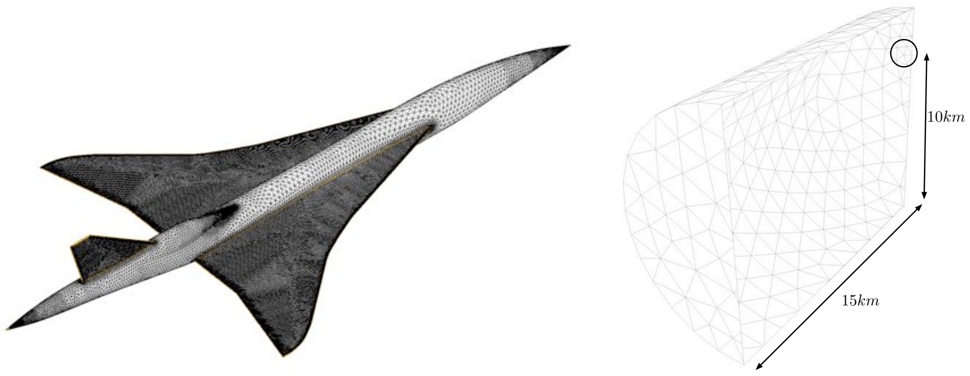

Figure 17: SSBJ example: Left, initial surface mesh of the SSBJ geometry, right, computational domain with the position of the aircraft in the domain.

mesh is $h_{min}/h_{max} = 1e^{-9}$ and the volume of the elements ranges from $5.4e^{-11}$ to $4.7e^{10}$. The flow condition is Mach number 1.6 with an angle of attack of 3 degrees. Our intent is to observed the pressure field for various R up to 9 km. This corresponds to a ratio R/L of about 243. According to the flow conditions, for $R = 9$ km, the length of the propagation of the shock waves emitted by the SSBJ is actually around 15 km.

The interpolation error on the Mach number in $\mathbf{L}^2$ norm is controlled and the surface is controlled with (11) and $\varepsilon = 0.001$. The strategy employed here is based on 30 adaptations at the following complexities: 80 000 , 160 000, 240 000, 400 000, 600 000 and 800 000. Each step is composed of 5 sub-iterations at a fixed complexity. The final mesh is composed of 3 299 367 vertices and 19 264 402 tetrahedra only. The average anisotropic ratio is 1907 while the mean anisotropic quotient is 50 3334. All the scales involved in this simulation are depicted in Figure 22. This example shows that a very high level of anisotropy is reached using unstructured mesh adaptation. Indeed, it is at least one order of magnitude higher than in the previous examples. Local refinement allows to keep a maximum accuracy and enables to generate quality anisotropic meshes. We mention that for each generated mesh, the worst element quality computed with (3) is always below 0.02 for the volume and below 0.05 for the surface while the percentage of unit elements is always greater than 90 %. In addition, the flow solver still converges on such meshes leading to accurate pressure signatures for $R/L \approx 250$. Anisotropic ratios and quotients for the whole sequence of meshes are reported in Table 1. They are increasing along the iterations. This shows that the accuracy across the shocks is increasing while the sizes in the anisotropic

Table 1: SSBJ example: Properties of each final adapted mesh : mean anisotropic ratio, mean anisotropic quotient, number of vertices and number of tetrahedra for each complexity. The last column gives the cumulative CPU time.

Iteration	Complexity	Ratio	Quotient	# Vertices	# Tet.	CPU time
5	80 000	200	10 964	432 454	2 254 826	1 h 10 mn
10	160 000	383	30 295	608 369	3 294 197	2 h 54 mn
15	240 000	698	81 129	1 104 910	6 243 462	6 h 9 mn
20	400 000	1 089	177 295	1 757 865	10 125 724	11 h 15 mn
25	600 000	1 575	340 938	2 572 814	14 967 820	18 h 47 mn
30	800 000	1 907	503 334	3 299 367	19 264 402	28 h 35 mn

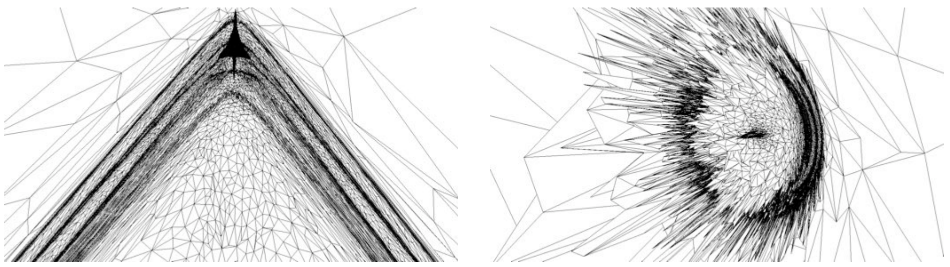

Figure 18: SSBJ example: Left, cut in the final adapted mesh 10 m below the aircraft. Right, cut 10 m behind the aircraft showing how anisotropic tetrahedra are aligned with the Mach cones.

directions are decreasing at a lower rate. The fact that the anisotropic quotient is increasing simply shows that there exist two anisotropic directions. Note that using (7) avoid to prescribe a minimal size during the adaptation leading to even stronger anisotropy. This property is due to the sensitivity property of (7) given by the local normalization term $\det(|H_R(u_h)|)^{\frac{-1}{2p+3}}$. An example of the scales of the solution is given by the pressure extractions in Figure 20 at $R = 5$ km and $R = 9$ km. Indeed, the normalized pressure signal at $R = 9$ km is of order $8e^{-4}$ while the magnitude is around $3e^{-2}$ at $R = 43$ m. Consequently, we can expect for the volume interpolation error to have a magnitude ratio of $(10^2)^3$. It is then necessary to guarantee that the error estimate detects such small amplitudes even in the presence of large amplitudes. This example demonstrates that using (7) complies with this requirement allowing to detect automatically all the scales of the solution, see Figure 20. Several cuts in the symmetry plane are depicted in Figure 19. At $R = 5$ km, we still distinguish 3 separated shocks waves and at $R = 9$ km only two shock waves are separated leading the classical N-wave signature. These features are even more emphasized on the pressure signatures in Figure 20.

The total CPU time is around 28 h 35 mn. 75 % of the CPU time is spent in the flow solver and 35 % in the remeshing, interpolation and error estimate. Note that accurate signatures at $R = 5$ km are already obtained after 11 h of CPU (corresponding to the 20th iteration) as depicted in Figure 20. We give in Table 1 the full sequence of CPU times. The first three steps provides an accurate signal for $R/L < 20$ and below.

4.5 Boundary layer shock interaction

We apply this strategy to study shock/boundary layer interaction. The test case is depicted in Figure 23. The shock waves are generated by a double wedge wing at Mach 1.4 with an angle attack of 0 degree and a Reynolds number of $3.4\,10^6$. Only the plate is treated as a viscous body. We solve the set of the Reynolds-average Navier-Stokes equations with Baldwin Lomax turbulence model. The final adapted mesh and the Mach number iso-values are depicted

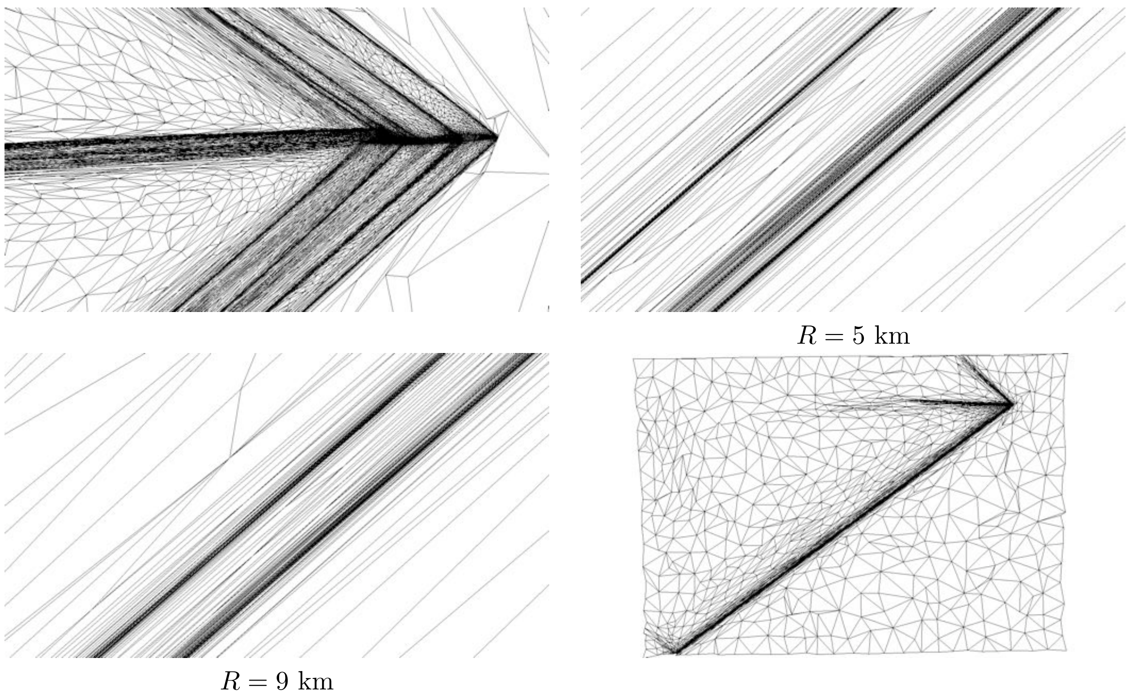

Figure 19: SSBJ example: From top to bottom, from left to rigth, cut in the final anisotropic mesh close to the aircraft, closer view of the mesh 5 km below the aircraft, closer view at 9 km below the aircraft, global view of the mesh.

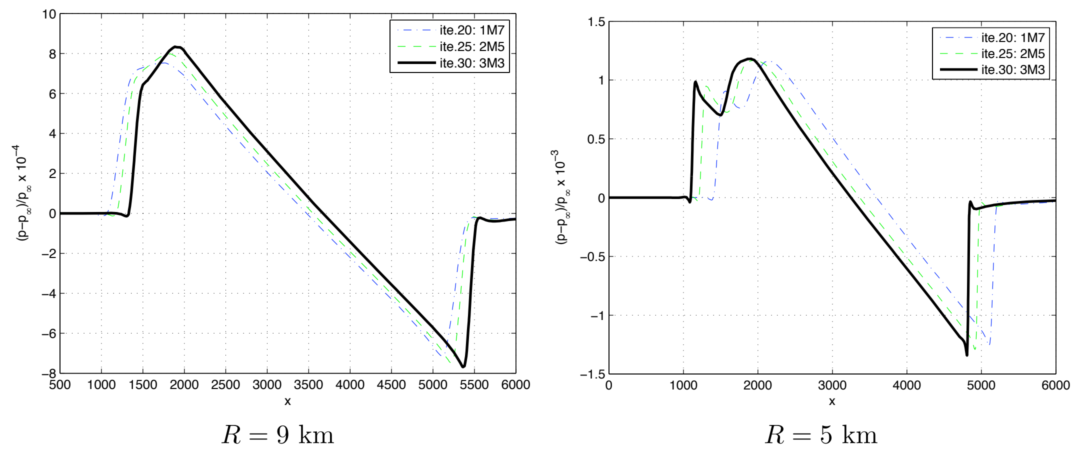

Figure 20: SSBJ example: Left, pressure signature at $R = 9$ km for the final meshes corresponding the last three complexities. Right, pressure signature at $R = 5$ km. The legend reports the number of vertices of each mesh in million (Iteration 20, 25 and 30). The pressure curves are deliberately shifted for visibility.

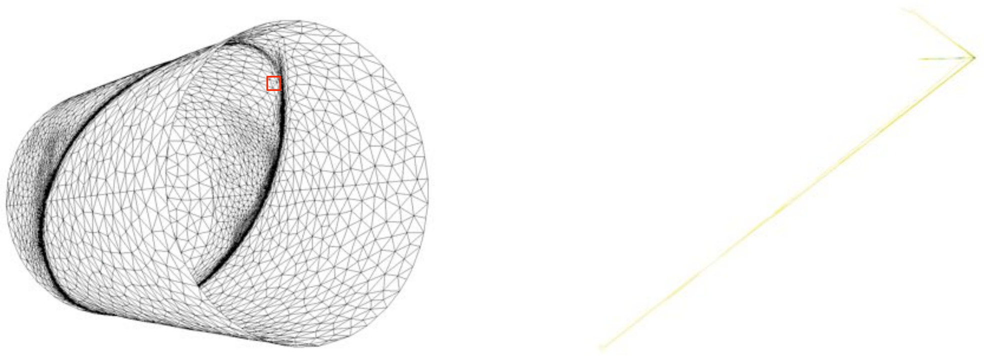

Figure 21: SSBJ example: Left, adapted surface mesh, the red square shows the position of the SSBJ. Right, Mach number iso-lines on the symetry plane $y = 0$.

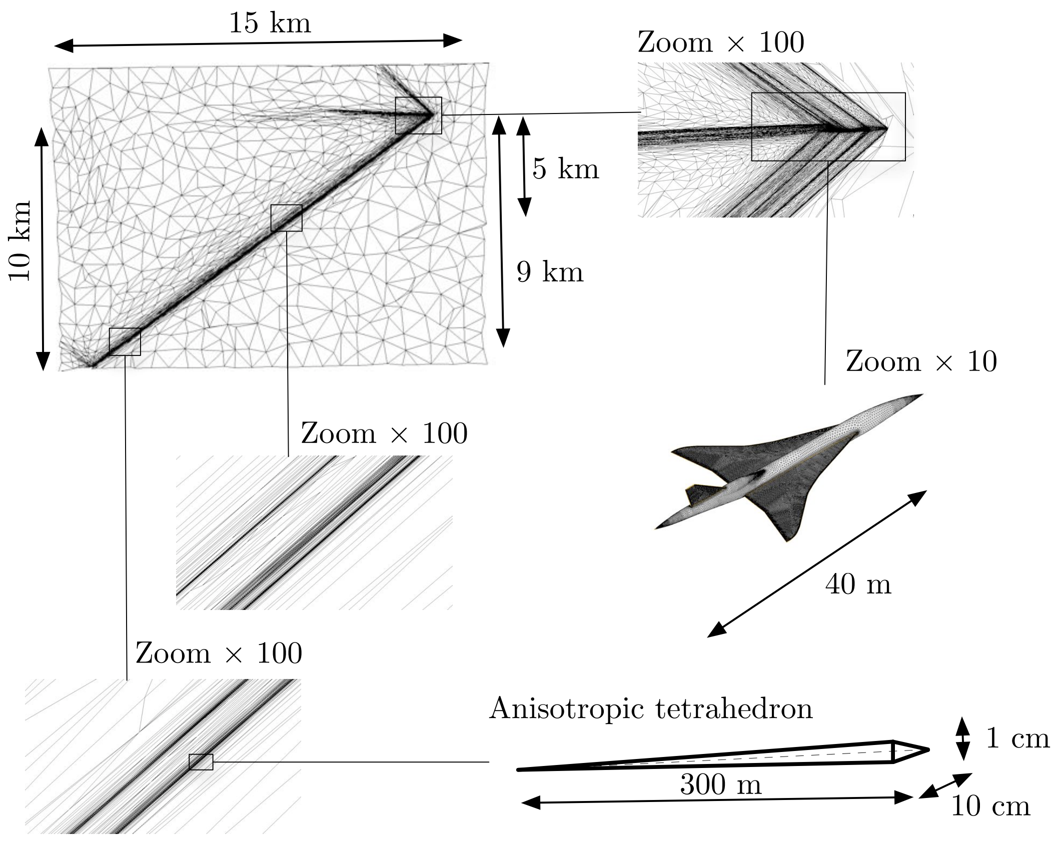

Figure 22: SSBJ example: Example of the scales of anisotropic elements reached in this simulation. An typical anisotropic element in the path of the shock has sizes of the order of 300 m , 10 cm and 1 cm

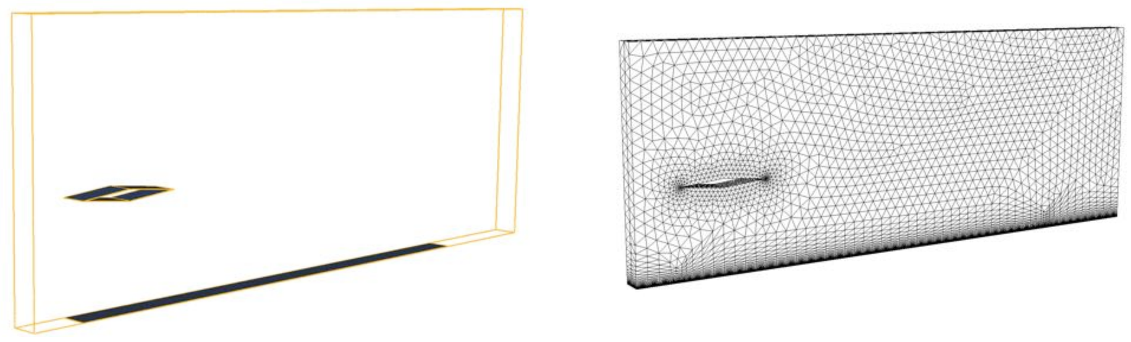
Figure 23: Shock/boundary layer interaction example: from left to right, computational domain and initial surface mesh

in Figure 23, closer views around the two shocks are depicted in Figure 24. The final mesh is composed of 280 000 vertices and 1.3 millions tetrahedra and is obtained after 20 iterations with a complexity of 10 000. In this example, we control the interpolation error on the Mach number coupled with boundary layer metric (13). The initial mesh is used as the background mesh to compute (13) with parameters $h_0 = 10^{-6}$ and $\alpha = 1.2$. As the boundary-layer metric is intersected with the interpolation error based metric, the resulting complexity is naturally greater. The unstructured boundary layer mesh height is around 10^{-7} near the plate. The average anisotropic ratio is grater than 10^6 and the average ratio around 500. The worst surface element quality is 0.03 and 0.002 for the volume. For the final mesh, the minimal size in the unstructured layer is of the order of 10^{-7}. As shown in Figure 24 (bottom), we successfully capture the typical bubbles and re-circulations at the intersection between the shocks and the boundary layer.

This simulation leads to the following observations. Generating a semi-structured boundary layer mesh extruded from the surface mesh gives only the require accuracy for the smaller layers. Indeed, the distance of the bottom of the shock from the viscous plate is around 10^{-3} whereas the initial height of the uniform boundary layer mesh was at 0.2. Consequently, the example emphasizes the difficulty of capturing these phenomena only with a *an priori* fixed quasi-structured boundary layer mesh. In addition, this approach is completely generic and robust and can handle complex geometries. However, if the shock/boundary layer interaction is automatically handled, the impact of having a fully unstructured mesh is not yet analyzed in term of solution accuracy and solver stability in the viscous area. Consequently, it seems also interesting to derive a method to generate structured mesh for the smaller layers (at least) while preserving (upper) anisotropic refinements. Metric-orthogonal and metric-aligned anisotropic mesh adaptation are possible solutions to generate highly anisotropic meshes whith quasi-structured elements [56, 63].

4.6 Double Mach reflection and blast prediction

The first unsteady case is double Mach reflection. This simulation starts from a 2-state initialization of a shock wave impacting a ramp. The density, speed and pressure for the right side is $(5.71, 9.76, 0, 0, 116.5)$ and $(1, 0, 0, 0, 1)$ for the left side, the shock wave propagates along the x-direction. The total physical time of the simulation is 0.18s. For this simulation, the time frame $[0, 0.18]$ is divided in 30 sub-time windows with 5 fixed point iterations and 21 metric intersections for each sub-time window. We control the $\mathbf{L}^2$ norm of the density interpolation error. The simulation CPU time is 8h55m, 80% is spent in the flow solver and 20% mesh adaptation. The

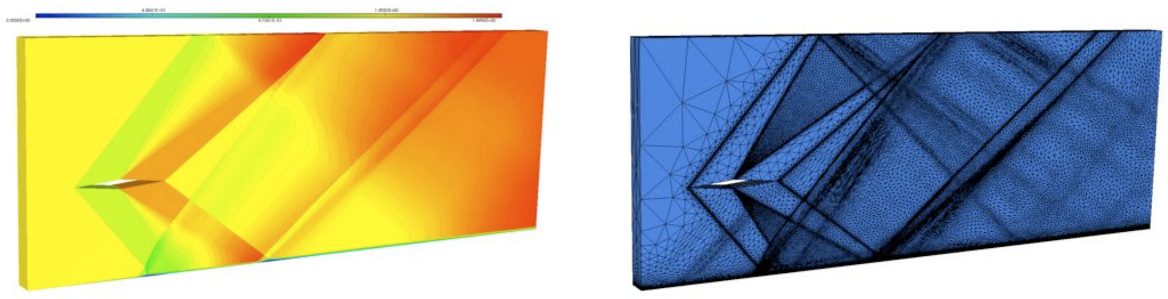

Figure 24: Shock/boundary layer interaction example: from left to right, Mach umber iso values and final adapted surface mesh.

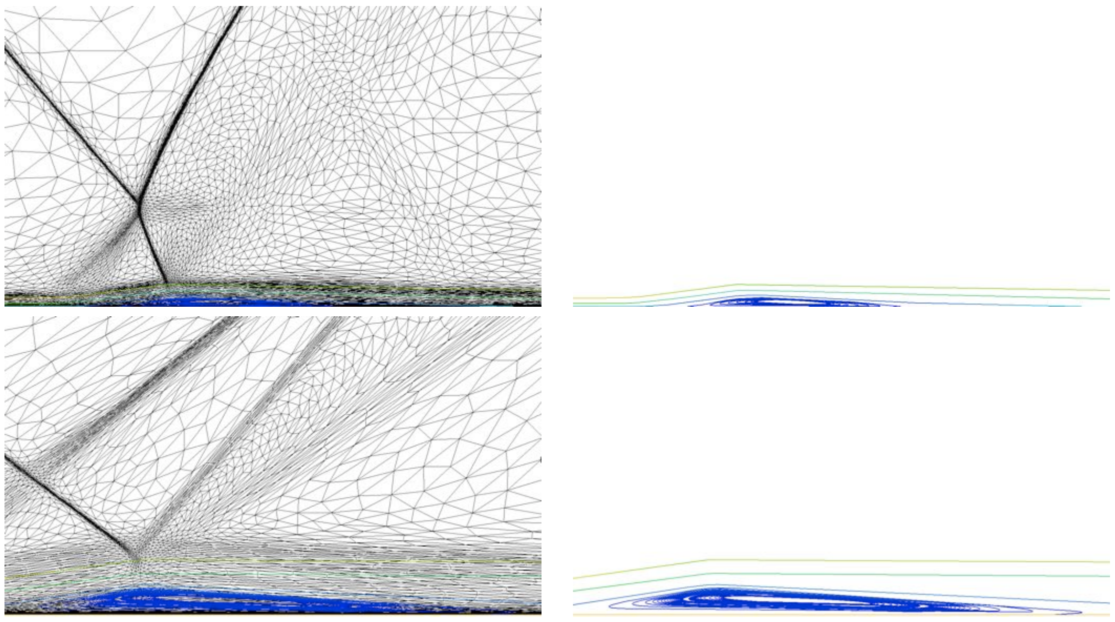

Figure 25: Shock/boundary layer interaction example: Stream lines of the velocity in two bubbles creating from the the interaction shock and boundary layer.

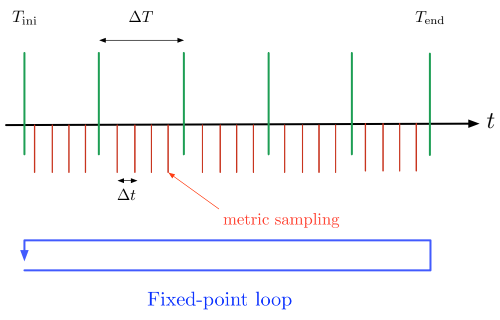

Figure 26: Unsteady adaptation algorithm: a fixed mesh is generated for sub-time window by sampling the solution at different time steps.

final mesh is composed of 235 095 vertices, 1 310 082 tetrahedra and 5 864 boundary faces. The mesh at final time and density iso-values are depicted in Figure 27. We can see that the contact discontinuity is impacting the ramp and that the generated vortices are pushing forward the initial front shock. If this phenomenon is usually observed in 2D simulation [89], its observation on 3D geometry is more complex. Moreover, the thickness of the adaptation is due to the fixed point strategy as the mesh is adapted for all the times step belonging to a sub-time frame.

We then consider a blast propagation on a more complex geometry: the US Capitol. Applying successfully an anisotropic adaptive simulation on it is challenging as it features many complex details as many columns, cupola, ... A classic load is considered, see Figure 28. The final physical time is 0.1 s. The whole time frame has been divided into 20 time slots of 0.005 s. The flow solver outputs density field every 0.0005 s. The final anisotropic mesh for the time frame $[0.05, 0.055]$ is depicted in Figure 28. The interpolation error on the density is controlled in $\mathbf{L}^2$ norm in space and time. The mesh is composed of almost 200 000 vertices for a total CPU time of 8 hours.

5 Conclusion

The standard computational pipeline have been described for complex geometries and unstructured mesh generation from CAD to surface and volume mesh generation. For adaptivity, we have described the basic principles of anisotropic mesh adaptation based on metric tensor fields: concept of unit mesh, metric interpolation, metric smoothing, ... A simple to implement but robust local remeshing strategy have been detailed. It allows to adapt different components of the flows. For each adaptation, we use a dedicated metric field issued from various estimates: surface curvatures, interpolation errors, distance to a body, ... Numerical examples show the robustness of the method to (i) reduce solver diffusion and (ii) reach a high level of anisotropy that is hardly tractable with a structured approach or a global remeshing method.

This chapter has covered only the basic processes that are required to reach a high level of anisotropy and recover a second order accuracy in space when simulating flows with shocks. It is important to mention that each component is crucial to gain all the benefit of adaptivity. Any improvement in one component may improve the whole process. We refer to [75] for a detailed discussion on the current issues of unstructured mesh adaptation. Mesh generation and adaptation is still an active field of research and many topics are not discussed in this chapter. This concerns the generation of boundary layer grids [14, 17, 33], the design of metric-aligned or metric-orthogonal grids [56, 63], the design of very high-order error estimates [91], the generation

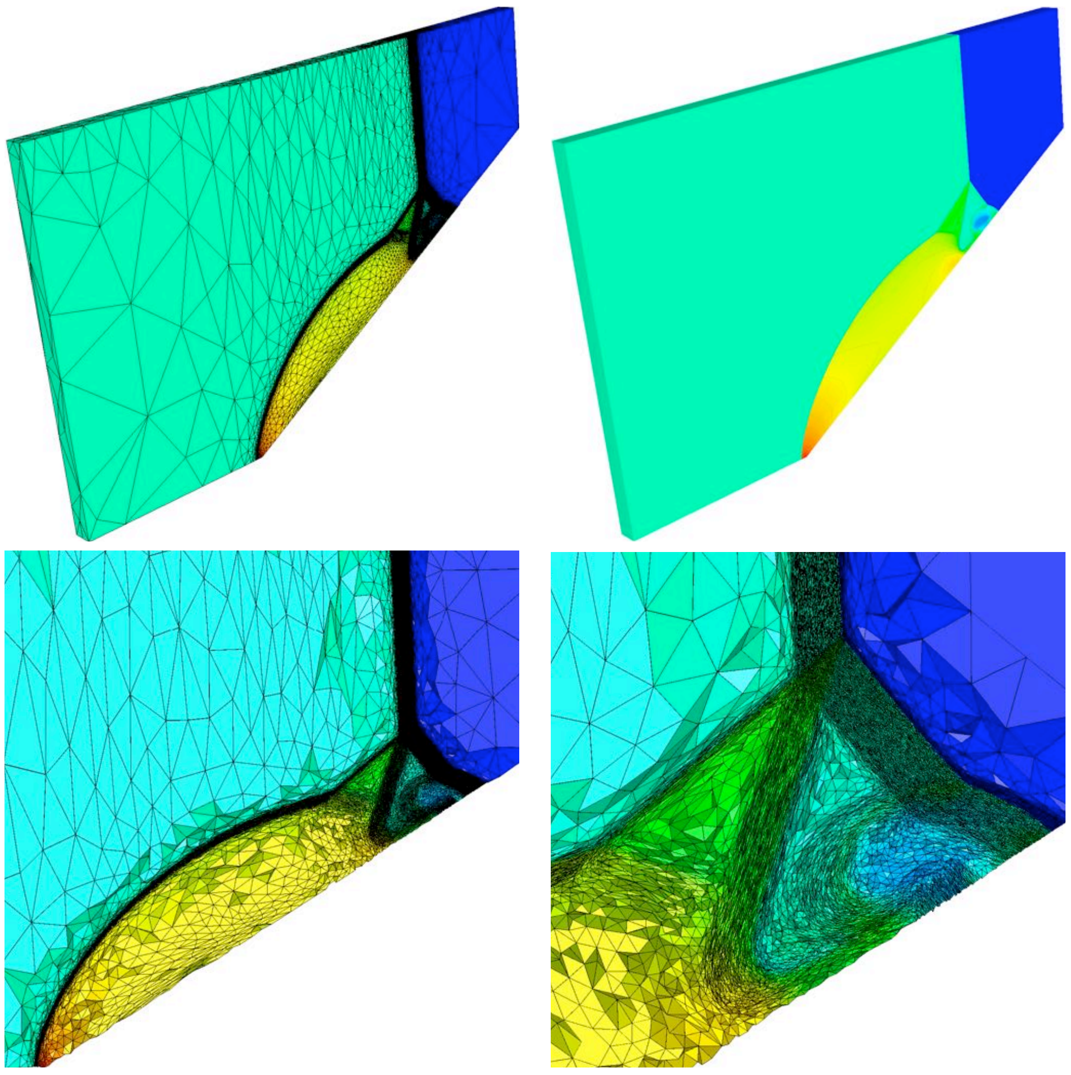

Figure 27: Double Mach reflection at final time : final adapted surface mesh (top left), density iso-values (top right), cut in the volume mesh (bottom left) and closer view near the contact discontinuity and vortex shock interaction (bottom right).

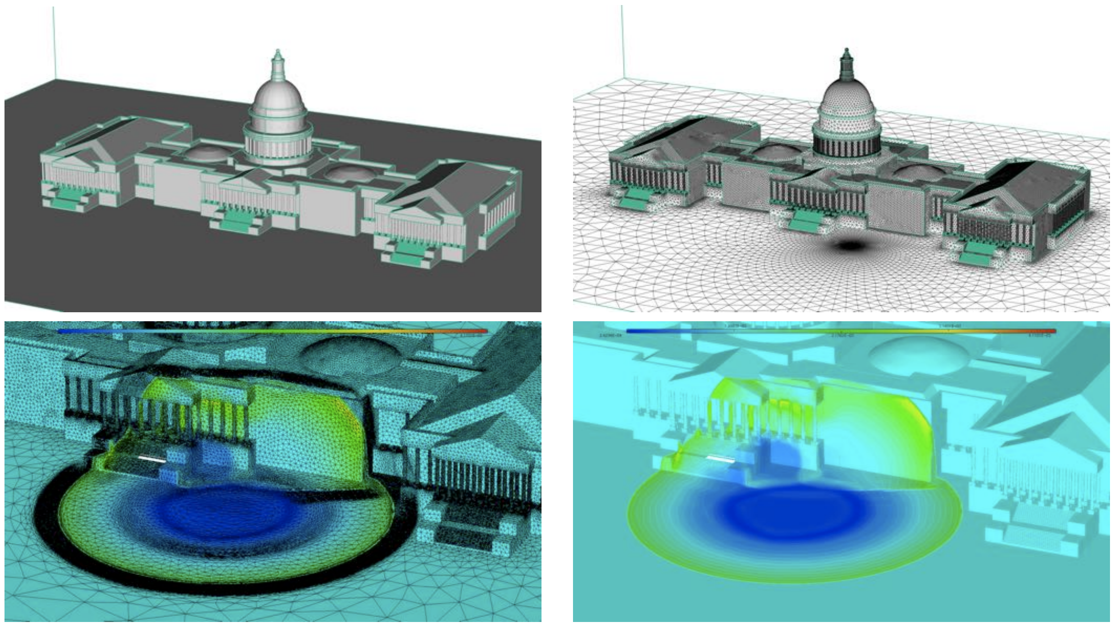

Figure 28: Top, the US capitol CAD (left) and initial surface mesh (right) with the initial blast location. Bottom, anisotropic surface mesh at $t = 0.025$s (left) and the density solution iso-values (right).

of high-order curved meshes [3, 80, 84, 90] and parallel (adaptive) mesh generation [9, 23, 65, 73, 78].

References

[1] R. Abgrall, *Toward the ultimate conservative scheme: Following the quest*, Journal of Computational Physics, 167 (2001), pp. 277 – 315.

[2] R. Abgrall, H. Beaugendre, and C. Dobrzynski, *An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques*, Journal of Computational Physics, 257, Part A (2014), pp. 83 – 101.

[3] R. Abgrall, C. Dobrzynski, and A. Froehly, *A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems*, International Journal for Numerical Methods in Fluids, 76 (2014), pp. 246–266.

[4] F. Alauzet, *Size gradation control of anisotropic meshes*, Finite Elements in Analysis and Design, (2009). published online.

[5] F. Alauzet and A. Loseille, *High order sonic boom modeling by adaptive methods*, J. Comp. Phys., 229 (2010), pp. 561–593.

[6] F. Alauzet and A. Loseille, *A decade of progress on anisotropic mesh adaptation for computational fluid dynamics*, Computer-Aided Design, 72 (2016), pp. 13 – 39. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[7] F. Alauzet and G. Olivier, *Extension of Metric-Based Anisotropic Mesh Adaptation to Time-Dependent Problems Involving Moving Geometries*, American Institute of Aeronautics and Astronautics, 2014/03/26 2011.

[8] A. Alleaume, *Automatic Non-manifold Topology Recovery and Geometry Noise Removal*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 267–279.

[9] A. Alleaume, L. Francez, M. Loriot, and N. Maman, *Automatic tetrahedral out-of-core meshing*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 461–476.

[10] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, *Log-Euclidean metrics for fast and simple calculus on diffusion tensors*, Magnetic Resonance in Medicine, 56 (2006), pp. 411–421.

[11] R. Aubry, S. Dey, K. Karamete, and E. Mestreau, *Smooth anisotropic sources with application to three-dimensional surface mesh generation*, Engineering with Computers, 32 (2016), pp. 313–330.

[12] R. Aubry, S. Dey, E. Mestreau, B. Karamete, and D. Gayman, *A robust conforming NURBS tessellation for industrial applications based on a mesh generation approach*, Computer-Aided Design, 63 (2015), pp. 26 – 38.

[13] R. Aubry, G. Houzeaux, and M. Vzquez, *A surface remeshing approach*, International Journal for Numerical Methods in Engineering, 85 (2011), pp. 1475–1498.

[14] R. Aubry and R. Löhner, *Generation of viscous grids at ridges and corners*, International Journal for Numerical Methods in Engineering, 77 (2009), pp. 1247–1289.

[15] A. Belme, A. Dervieux, and F. Alauzet, *Time accurate anisotropic goal-oriented mesh adaptation for unsteady flows*, J. Comp. Phys., 231 (2012), pp. 6323–6348.

[16] C. Bottasso, *Anisotropic mesh adaption by metric-driven optimization*, Int. J. Numer. Meth. Engng, 60 (2004), pp. 597–639.

[17] C. Bottasso and D. Detomi, *A procedure for tetrahedral boundary layer mesh generation*, Engineering Computations, 18 (2002), pp. 66–79.

[18] G. Brèthes, O. Allain, and A. Dervieux, *A mesh-adaptative metric-based full multigrid for the poisson problem*, International Journal for Numerical Methods in Fluids, 79 (2015), pp. 30–53. fld.4042.

[19] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau, *Anisotropic unstructured mesh adaptation for flow simulations*, Int. J. Numer. Meth. Fluids, 25 (1997), pp. 475–491.

[20] L. Chen, P. Sun, and J. Xu, *Optimal anisotropic meshes for minimizing interpolation errors in L^p-norm*, Math. Comp., 76 (2007), pp. 179–204.

[21] G. Compère, J.-F. Remacle, J. Jansson, and J. Hoffman, *A mesh adaptation framework for dealing with large deforming meshes*, Int. J. Numer. Meth. Engng, 82 (2010), pp. 843–867.

[22] T. Coupez, *Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing*, Journal of Computational Physics, 230 (2011), pp. 2391 – 2405.

[23] T. Coupez, H. Digonnet, and R. Ducloux, *Parallel meshing and remeshing*, Applied Mathematical Modelling, 25 (2000), pp. 153 – 175. Dynamic load balancing of mesh-based applications on parallel.

[24] P.-H. Cournède, B. Koobus, and A. Dervieux, *Positivity statements for a Mixed-Element-Volume scheme on fixed and moving grids*, European Journal of Computational Mechanics, 15 (2006), pp. 767–798.

[25] C. Dobrzynski and P. Frey, *Anisotropic delaunay mesh adaptation for unsteady simulations*, in Proc. of 17th Int. Meshing Rountable, Springer, 2008, pp. 177–194.

[26] J. Dompierre, M. Vallet, M. Fortin, Y. Bourgault, and W. Habashi, *Anisotropic mesh adaptation: towards a solver and user independent CFD*, in AIAA 35th Aerospace Sciences Meeting and Exhibit, AIAA-1997-0861, Reno, NV, USA, Jan 1997.

[27] L. A. Freitag and C. Ollivier-Gooch, *Tetrahedral mesh improvement using swapping and smoothing*, International Journal for Numerical Methods in Engineering, 40 (1997), pp. 3979–4002.

[28] P. Frey, *About surface remeshing*, in Proceedings of the 15th International Meshing Roundtable, Springer, 2000, pp. 123–136.

[29] ——, `Yams`*, a fully automatic adaptive isotropic surface remeshing procedure*, RT-0252, INRIA, 2001.

[30] P. Frey and F. Alauzet, *Anisotropic mesh adaptation for* CFD *computations*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 5068–5082.

[31] P. Frey and H. Borouchaki, *Surface meshing using a geometric error estimate*, Int. J. Numer. Meth. Engng, 58 (2003), pp. 227–245.

[32] P. Frey and P.-L. George, *Mesh generation. Application to finite elements*, ISTE Ltd and John Wiley & Sons, 2nd ed., 2008.

[33] R. Garimella and M. Shephard, *Boundary layer mesh generation for viscous flow simulations*, Int. J. Numer. Meth. Fluids, 49 (2000), pp. 193–218.

[34] P. George, `Gamanic3d`*, adaptive anisotropic tetrahedral mesh generator*, Technical Note, INRIA, 2003.

[35] P. George and H. Borouchaki, *Delaunay triangulation and meshing : application to finite elements*, Hermès Science, Paris, Oxford, 1998.

[36] ——, *Back to edge flips in 3 dimensions*, in Proceedings of the 12th International Meshing Roundtable, Santa Fe, NM, USA, 2003, pp. 393–402.

[37] P. George, F. Hecht, and E. Saltel, *Fully automatic mesh generator for 3d domains of any shape*, Impact of Comuting in Science and Engineering, 2 (1990), pp. 187–218.

[38] P. L. George, H. Borouchaki, and E. Saltel, *ultimate robustness in meshing an arbitrary polyhedron*, International Journal for Numerical Methods in Engineering, 58 (2003), pp. 1061–1089.

[39] M. Giles and E. Suli, *Adjoint methods for PDEs: a posteriori error analysis and post-processing by duality*, in Acta Numerica, Cambridge University Press, 2002, pp. 145–236.

[40] M. B. Giles, *On adjoint equations for error analysis and optimal grid adaptation in CFD*, Tech. Rep. NA-97/11, Oxford, 1997.

[41] M. B. Giles and N. Pierce, *Improved lift and drag estimates using adjoint euler equations*, AIAA Paper , 1999-3293 (1999).

[42] R. Hartmann, *Multitarget error estimation and adaptivity in aerodynamic flow simulations*, SIAM Journal on Scientific Computing, 31 (2008), pp. 708–731.

[43] F. Hecht, `BAMG`*: bidimensional anisotropic mesh generator*, available from http://www-rocq.inria.fr/gamma/cdrom/www/bamg/eng.htm, INRIA-Rocquencourt, France, 1998.

[44] W. Huang, *Metric tensors for anisotropic mesh generation*, J. Comp. Phys., 204 (2005), pp. 633–665.

[45] C. Johnson, U. Navert, and J. Pitkaranta, *Finite element methods for linear hyperbolic problems*, Math. Comp., 69 (1985), pp. 25–39.

[46] W. Jones, E. Nielsen, and M. Park, *Validation of 3D adjoint based error estimation and mesh adaptation for sonic boom reduction*, in 44th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2006-1150, Reno, NV, USA, Jan 2006.

[47] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680.

[48] P. Laug, *Some aspects of parametric surface meshing*, Finite Elements in Analysis and Design, 46 (2010), pp. 216 – 226. Mesh Generation - Applications and Adaptation.

[49] P. Laug and H. Bourochaki, `BL2D-V2`*, mailleur bidimensionnel adaptatif*, RR-0275, INRIA, 2003.

[50] X. Li, J.-F. Remacle, , N. Chevaugeon, and M. S. Shephard, *Anisotropic mesh gradation control*, in Proceedings of the 13th International Meshing Roundtable, Williamsburg, VA, USA, 2004.

[51] X. L. Li, M. S. Shephard, and M. W. Beall, *3D anisotropic mesh adaptation by mesh modification*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 4915–4950.

[52] D. S. Lo, *Finite Element Mesh Generation*, CRC Press, 2015.

[53] R. Löhner, *Applied CFD techniques*, Wiley, New-York, 2001.

[54] R. Löhner, *Recent advances in parallel advancing front grid generation*, Archives of Computational Methods in Engineering, 21 (2014), pp. 127–140.

[55] R. Löhner and P. Parikh, *Three-dimensionnal grid generation by the advancing-front method*, Int. J. Numer. Meth. Fluids, 8 (1988), pp. 1135–1149.

[56] A. Loseille, *Metric-orthogonal anisotropic mesh generation*, Procedia Engineering, 82 (2014), pp. 403 – 415.

[57] A. Loseille and F. Alauzet, *Continuous mesh framework. Part I: well-posed continuous interpolation error*, SIAM J. Numer. Anal., 49 (2011), pp. 38–60.

[58] ——, *Continuous mesh framework. Part II: validations and applications*, SIAM J. Numer. Anal., 49 (2011), pp. 61–86.

[59] A. Loseille, A. Dervieux, and F. Alauzet, *Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations*, J. Comp. Phys., 229 (2010), pp. 2866–2897.

[60] A. Loseille, A. Dervieux, and F. Alauzet, *Anisotropic Norm-Oriented Mesh Adaptation for Compressible Flows*, American Institute of Aeronautics and Astronautics, 2016/06/20 2015.

[61] A. Loseille, A. Dervieux, P. Frey, and F. Alauzet, *Achievement of global second-order mesh convergence for discontinuous flows with adapted unstructured meshes*, in 37th AIAA Fluid Dynamics Conference, AIAA Paper 2007-4186, Miami, FL, USA, Jun 2007.

[62] A. Loseille and R. Löhner, *Adaptive anisotropic simulations in aerodynamics*, in 48th AIAA Aerospace Sciences Meeting, AIAA Paper 2010-169, Orlando, FL, USA, Jan 2010.

[63] A. Loseille, D. L. Marcum, and F. Alauzet, *Alignment and orthogonality in anisotropic metric-based mesh adaptation*, American Institute of Aeronautics and Astronautics, 2016/06/18 2015.

[64] A. Loseille and V. Menier, *Serial and parallel mesh modification through a unique cavity-based primitive*, in Proceedings of the 22nd International Meshing Roundtable, X. Jiao and J.-C. Weill, eds., 2013.

[65] A. Loseille, V. Menier, and F. Alauzet, *Parallel generation of large-size adapted meshes*, Procedia Engineering, 124 (2015), pp. 57 – 69.

[66] H. Luo, J. Baum, and R. Löhner, *A fast, matrix-free implicit method for compressible flows on unstructured grids*, J. Comp. Phys., 146 (1998), pp. 664–690.

[67] L.W. Hunton and R.M. Hicks and J.P. Mendoza, *Some effects of wing planform on sonic boom*, TN. D-7160, Nasa, 1973.

[68] D. L. Marcum, *Adaptive unstructured grid generation for viscous flow applications*, AIAA Journal, 34 (1996), pp. 2440–2443.

[69] ———, *Efficient generation of high-quality unstructured surface and volume grids*, Engrg. Comput., 17 (2001), pp. 211–233.

[70] D. Mavriplis, *An advancing front delaunay triangulation algorithm designed for robustness*, J. Comp. Phys., 117 (1995), pp. 90–101.

[71] V. Menier, A. Loseille, and F. Alauzet, *Multigrid Strategies Coupled with Anisotropic Mesh Adaptation*, American Institute of Aeronautics and Astronautics, 2016/06/18 2015.

[72] T. Michal and J. Krakos, *Anisotropic mesh adaptation through edge primitive operations*, AIAA Paper , 2011-0159 (2011).

[73] A. Ovcharenko, K. Chitale, O. Sahni, K. E. Jansen, and M. S. Shephard, *Parallel Adaptive Boundary Layer Meshing for CFD Analysis*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 437–455.

[74] C. C. Pain, A. P. Umpleby, C. R. E. de Oliveira, and A. J. H. Goddard, *Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 3771–3796.

[75] M. Park, J. Krakos, T. R. Michal, A. Loseille, and J. Alonso, *Unstructured Grid Adaptation: Status, Potential Impacts, and Recommended Investments Toward CFD Vision 2030*, American Institute of Aeronautics and Astronautics, 2014/03/26 2011.

[76] L. Piegl and W. Tiller, *The NURBS Book (2Nd Ed.)*, Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[77] P. Power, C. Pain, M. Piggott, F. Fang, G. Gorman, A. Umpleby, and A. Goddard, *Adjoint a posteriori error measures for anisotropic mesh optimization*, Computers & Mathematics with Applications, 52 (2006), pp. 1213–1242.

[78] J.-F. Remacle, V. Bertrand, and C. Geuzaine, *A two-level multithreaded delaunay kernel*, Procedia Engineering, 124 (2015), pp. 6 – 17.

[79] G. Rokos, G. J. Gorman, K. E. Jensen, and P. H. J. Kelly, *Thread parallelism for highly irregular computation in anisotropic mesh adaptation*, CoRR, abs/1505.04694 (2015).

[80] O. Sahni, X. J. Luo, K. E. Jansen, and M. S. Shephard, *Curved boundary layer meshing for adaptive viscous flow simulations*, Finite Elem. Anal. Des., 46 (2010), pp. 132–139.

[81] C. W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.

[82] H. Si, *Tetgen, a delaunay-based quality tetrahedral mesh generator*, ACM Trans. Math. Softw., 41 (2015), pp. 11:1–11:36.

[83] A. Tam, D. Ait-Ali-Yahia, M. P. Robichaud, M. Moore, V. Kozel, and W. G. Habashi, *Anisotropic mesh adaptation for 3D flows on structured and unstructured grids*, Comput. Methods Appl. Mech. Engrg., 189 (2000), pp. 1205–1230.

[84] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts, *Robust untangling of curvilinear meshes*, J. Comput. Phys., 254 (2013), pp. 8–26.

[85] M.-G. Vallet, C.-M. Manole, J. Dompierre, S. Dufour, and F. Guibault, *Numerical comparison of some hessian recovery techniques*, Int. J. Numer. Meth. Engng, 72 (2007), pp. 987–1007.

[86] Y. Vasilevski and K. Lipnikov, *Error bounds for controllable adaptive algorithms based on a hessian recovery*, Computational Mathematics and Mathematical Physics, 45 (2005), pp. 1374–1384.

[87] D. A. Venditti and D. L. Darmofal, *Grid adaptation for functional outputs: application to two-dimensional inviscid flows*, J. Comp. Phys., 176 (2002), pp. 40–69.

[88] A. Vlachos, J. Peters, C. Boyd, and J. L. Mitchell, *Curved pn triangles*, in Proceedings of the 2001 Symposium on Interactive 3D Graphics, I3D '01, New York, NY, USA, 2001, ACM, pp. 159–166.

[89] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, Journal of Computational Physics, 54 (1984), pp. 115 – 173.

[90] Z. Q. Xie, R. Sevilla, O. Hassan, and K. Morgan, *The generation of arbitrary order curved meshes for 3d finite element analysis*, Comput. Mech., 51 (2013), pp. 361–374.

[91] M. Yano and D. L. Darmofal, *An optimization-based framework for anisotropic simplex mesh adaptation*, J. Comput. Phys., 231 (2012), pp. 7626–7649.