

Finite Volume Error Minimisation via the Adaptation of Voronoi Meshes from Transonic to Hypersonic Flows

B. Praud^{*,**}, A. Lemoine^{**}, L. Muscat^{*}, J. Breil^{*,**}

Corresponding author: brieuc.praud@cea.fr

^{*} CEA-CESTA, F-33114 Le Barp, France.

^{**} I2M, UMR 5295, F-33400 Talence, France.

Abstract: Hypersonic flows are very challenging for numerical methods. In this context, the mesh is as important as the numerical scheme to achieve accurate simulations. It has been shown that Voronoi meshes tend to dampen shock instabilities such as the *carbuncle* effect, thanks to the greater rotational invariance that their cells provide in numerical dissipation compared to their simplicial counterparts. Furthermore, a specific kind of Voronoi meshes, namely Centroidal Voronoi Tessellations, naturally arise as minimisers of the error of cell-centered Finite Volume schemes, providing a framework for mesh adaptation to increase numerical simulations accuracy while keeping the computational cost moderate. Several efficient algorithms have been developed to generate triangulations and are implemented in various meshing tools. Leveraging the geometrical duality between Voronoi tessellations and Delaunay triangulations, one can construct the Voronoi diagram from a triangulation thus rely on the same efficient core algorithm. Our aim is to develop a methodology to construct and adapt Voronoi meshes according to stiff compressible effects that occur when simulating hypersonic flows.

Keywords: Hypersonic flows, Euler equations, Carbuncle, Voronoi mesh, Mesh adaptation

Introduction

Block-structured grids has historically been preferred over unstructured meshes for hypersonic flow simulations as they provide both accuracy and computational efficiency. The generation of such meshes can however be challenging for complex geometries or flows with intricate features. Over the years, mesh adaptation has proven to be an extremely powerful tool for Computational Fluid Dynamics (CFD), by both reducing computational costs and improving solution accuracy [1]. This leap forward made unstructured meshes competitive for transonic and supersonic aerodynamic applications [2].

Mesh adaptation is mainly done on simplicial meshes (triangular meshes in 2D, tetrahedral meshes in 3D) or quad meshes as the predictive data structure that they rely on makes algorithms conceptually simpler than arbitrary meshes and also computationally more efficient. However, cell-centered Finite Volume (FV) solvers may misbehave on triangular meshes, especially at high speed flows, leading to numerical instabilities such as the carbuncle phenomenon [3]. It has been shown that the greater cell isotropy provided by Voronoi meshes tends to dampen this instability [4] and provide greater accuracy overall [5]. Voronoi diagrams also provide an adequate framework for mesh adaptation [6, 7] where quality meshes can be obtained as local minimisers of the so called Centroidal Voronoi Tessellation (CVT) energy function [8].

In this article, a full Voronoi mesh adaptation pipeline is described, including mesh generation, adaptation and CFD computations. The main contributions of this article are a new 2D Voronoi diagrams clipping strategy which is relatively simple to implement and integrates seamlessly with the mesh adaptation process (Section 1), showing how the concept of CVT naturally arises from the minimisation of the feature-based error estimation of the second-order accurate FV scheme (Section 2) and evaluate the suitability of the methodology for transonic and hypersonic inviscid flow computations (Section 3).

1. Voronoi diagrams and meshes

This section defines the concepts of Voronoi diagram and CVT, then describes the methodology used to build Voronoi meshes with an emphasis on the clipping stage.

1.1 Voronoi diagram definition

Let $\Omega \subset \mathbb{R}^d$ be a connected compact region, where $\mathbb{R}^d$ denote the d -dimensional Euclidean space. Let also $(x_i)_{i=1}^N \in \Omega^N$ be a set of distinct points. These points, called *seeds* or *generators*, define a tiling $(\Omega_i)_{i=1}^N$ of domain Ω where each cell Ω_i is defined as

$$\Omega_i = \{x \in \Omega \mid \forall j \in \llbracket 1, N \rrbracket \quad \|x - x_i\| \leq \|x - x_j\|\}.$$

This means that a cell Ω_i is the set of all points from Ω which are closer from generator x_i than from any other seed. While other mathematically equivalent definitions exist [9], this one emphasise the compactness property that this structure provide. An interesting special case of Voronoi diagrams is obtained when all seeds are collocated with the centroids of the cells that they generate. Such a diagram is called a CVT and is particularly well-suited for meshing applications.

1.2 Clipped Voronoi mesh generation

Voronoi diagram construction The Voronoi diagram is equivalently characterised by its duality from the Delaunay triangulation of its generators. This means that usual algorithms used for Delaunay triangulations generation, such as Bowyer-Watson's for which computationally efficient implementations have been developed [10], can be leveraged to construct the Voronoi diagram, provided that a post-processing dualisation step is performed. The pipeline is thus as follows: first generators are sampled inside the domain. Then, the convex hull of these generators is Delaunay-triangulated. Follows the dualisation step where each vertex from the triangulation leads to the creation of a Voronoi cell. Finally, as the domain boundary $\partial\Omega$ is not taken into account during the dualisation step, the constructed Voronoi diagram is unbounded thus needs to be clipped. This last step is the subject of next paragraph.

Clipping strategy Here a novel clipping strategy, applying only to the bidimensional case, is introduced. The reader can refer to [11] for a more general clipping strategy which is applicable to both 2D and 3D Voronoi diagrams but is more involved.

A multiply-connected polygon $\mathcal{P}$ approximating $\partial\Omega$ is considered. This linear approximation of the boundary is defined as a counter-clockwise exterior boundary $\mathcal{P}_0$ and possibly several clockwise-oriented polygons $\mathcal{P}_i$ that define inner boundaries. Each polygon $\mathcal{P}_i$ is also defined as a set of oriented edges $(e_i^k)_{k=1}^{n_i}$. Edge $e_{\mathcal{V}, \mathcal{W}}$ denotes the edge shared by cells $\mathcal{V}$ and $\mathcal{W}$: $e_{\mathcal{V}, \mathcal{W}} = \partial\mathcal{V} \cap \partial\mathcal{W}$. The clipping process, as described in Algorithm 1, relies on neighbor-to-neighbor propagation and a simplified special case of the Greiner-Hormann polygon clipping algorithm [12]. This polygon clipping algorithm works by connecting the circular linked list $L_{\mathcal{W}}$ of the vertices of the polygon to be clipped in counter-clockwise order, to the list L_e of the vertices of an oriented boundary fragment. This connection is made by inserting the intersection points stored in L_e at the appropriate location in $L_{\mathcal{W}}$ based on their curvilinear abscissa, as exemplified by Figure 1.

Note that the only new vertices introduced by this clipping stage are the intersections between a Voronoi cell edge and a boundary edge. As an entering point for a cell is also the exit point for the previous cell, the connectivity of the clipped mesh is easy to keep track of.

Depending on the initial discretisation of $\partial\Omega$, this clipping phase can create cells with many tiny faces that are useless from a computational standpoint and can even degrade the flow solver performance. A simple manner of avoiding such overdiscretisation is to save only the entering and the leaving points (the intersections), effectively skipping over the points of $\mathcal{P}_i$.

```

foreach  $\mathcal{P}_i \in \mathcal{P}$  do
    Find the cell  $\mathcal{V}$  containing the starting point of  $e_i^1$ ;
     $k \leftarrow 1$ ;
    /* Now a first intersection is found by leaving the current cell. */
    while  $e_i^k \cap \partial\mathcal{V} = \emptyset$  do
        |  $k \leftarrow k + 1$ ;
    end
     $k_0 \leftarrow k$ ;
    repeat
        /* Here  $e_i^k$  intersects the line segment  $e_{\mathcal{V},\mathcal{W}}$  shared by cell  $\mathcal{V}$  and
           its neighbor  $\mathcal{W}$ . */
        Store the cell  $\mathcal{W}$  as a cycling linked list  $L_{\mathcal{W}}$  (see Figure 1);
        Create another linked list  $L_e$ ;
        Insert vertex  $e_i^k \cap e_{\mathcal{V},\mathcal{W}}$  in  $L_e$ ;
        while  $e_i^k \cap (\partial\mathcal{W} \setminus e_{\mathcal{V},\mathcal{W}}) = \emptyset$  do
            | Insert the endpoint of  $e_i^k$  in  $L_e$ ;
            |  $k \leftarrow k + 1$ ;
            | if  $k > n_i$  then
            | |  $k \leftarrow 1$ ;
            | end
        end
        Insert vertex  $e_i^k \cap (\partial\mathcal{W} \setminus e_{\mathcal{V},\mathcal{W}})$  in  $L_e$ ;
        Connect list  $L_e$  to  $L_{\mathcal{W}}$ ;
        Follow the linked list from first intersection until looping back. This loop is the cell  $\mathcal{W}$ 
        clipped against  $\mathcal{P}_i$ ;
         $\mathcal{V} \leftarrow \mathcal{W}$ ;
    until  $k = k_0$ ;
end

```

Algorithm 1: Clipping procedure.

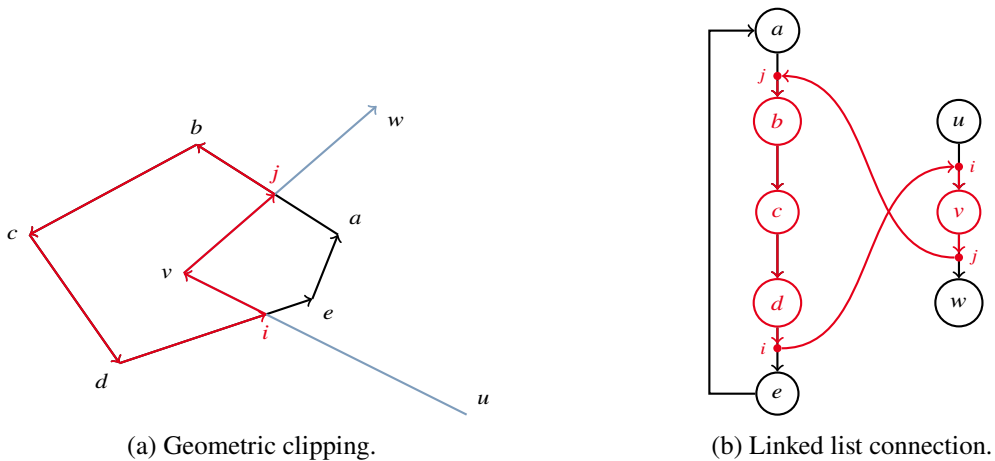


Figure 1: Cell clipping example. Cell $L_{\mathcal{W}} = (a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow a)$ is clipped against a boundary piece $L_e = (u \rightarrow v \rightarrow w)$ effectively creating two intersection points i and j .

As the clipping procedure needs to be performed at each iteration of the mesh optimisation process, it is required to be computationally efficient. This efficiency is provided by the neighbor-to-neighbor propagation strategy and the fact that the number of clipped cells is generally far below the total number of cells. This makes the clipping phase cost negligible compared to the actual Voronoi construction phase. Note also that a linear search is sufficient to find the first intersection, provided that the number of boundaries is significantly smaller than the number of cells.

Algorithm 1 can be extended in order to adress some edge cases:

- If a boundary $\mathcal{P}_i$ is fully contained within a cell then the algorithm will loop forever. This issue can be addressed either by exiting with failure if k came back to zero without any intersection being found, or by considering that the clipped mesh is the original mesh with the intersected cell removed.
- If two boundaries $\mathcal{P}_i$ and $\mathcal{P}_j$ are relatively close from each other, then a single cell can be clipped against both $\mathcal{P}_i$ and $\mathcal{P}_j$. This issue can be addressed by not doing the clipping phase on the fly but stack the linked lists instead, adding each new cut to the already existing lists. The clipping stage can be done afterward.
- Robust geometric predicates are used to perform intersection tests [13]. However edge cases such as intersections on edges endpoints are not considered because a random initialisation is used for generating initial Voronoi seeds.

1.3 L-BFGS optimisation

The Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm is considered to optimise the placement of the Voronoi seeds. This is the most common and efficient strategy to converge toward a CVT [8]. Given the L-BFGS descent direction, the step is given by a line search enforcing the strong Wolfe conditions [14]. In order to avoid generators leaving domain $\mathcal{P}$, each local displacement is clamped to the boundary. This effectively affects the descent direction and can break L-BFGS assumptions. The L-BFGS update is therefore skipped if the curvature condition does not hold for this restricted global displacement.

The clipping stage boundary simplification degrades the convergence of the algorithm. Therefore, this cleaning step is done at the very end, after the mesh optimisation.

2. Finite volume error estimation and minimisation

This subsection derives an approximation of the FV method reconstruction error and shows that minimising such estimate leads naturally to building a CVT.

2.1 Finite Volume error estimation

Let $(\Omega_i)_{i=1}^N$ be a mesh of the computational domain Ω . At a point $\mathbf{x}$ of cell Ω_i with centroid $\mathbf{x}_i$, the Taylor expansion of a C^2 -smooth scalar field u writes

$$u(\mathbf{x}) = u(\mathbf{x}_i) + (\mathbf{x} - \mathbf{x}_i) \cdot \nabla u(\mathbf{x}_i) + R_1(\mathbf{x}_i, \mathbf{x}), \quad (1)$$

where the remainder $R_1(\mathbf{x}_i, \mathbf{x})$ depending on the Hessian matrix $\mathbb{H}_u$ of field u is generally written as:

$$R_1(\mathbf{x}_i, \mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_i) \cdot \mathbb{H}_u(\mathbf{x}_i) (\mathbf{x} - \mathbf{x}_i) + o(\|\mathbf{x} - \mathbf{x}_i\|^2).$$

but can also write

$$R_1(\mathbf{x}_i, \mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_i) \cdot \mathbb{H}_u(\mathbf{x}) (\mathbf{x} - \mathbf{x}_i) + o(\|\mathbf{x} - \mathbf{x}_i\|^2).$$

as $\mathbb{H}_u(\mathbf{x}) = \mathbb{H}_u(\mathbf{x}_i) + o(1)$.

Considering a second-order accurate cell-centered FV scheme, the pointwise truncation error e_i that stems from the linear reconstruction over the cell Ω_i is thus

$$e_i(\mathbf{x}) = |R_1(\mathbf{x}_i, \mathbf{x})|, \quad (2)$$

which can be bounded as follows

$$|e_i(\mathbf{x})| \lesssim \frac{1}{2}(\mathbf{x} - \mathbf{x}_i) \cdot |\mathbb{H}_u|(\mathbf{x}) (\mathbf{x} - \mathbf{x}_i), \quad (3)$$

where the absolute value $|\mathbb{H}_u|$ of a matrix $\mathbb{H}_u$ is defined from its spectral decomposition $\mathbb{H}_u = \mathbb{P}^\top \text{diag}(\lambda, \mu) \mathbb{P}$ as $|\mathbb{H}_u| = \mathbb{P}^\top \text{diag}(|\lambda|, |\mu|) \mathbb{P}$.

This bound is optimal since equality is achieved when u is an elliptic function [15]. Therefore, the interpolation error can be reasonably modeled as

$$|e_i(\mathbf{x})| \approx \frac{1}{2}(\mathbf{x} - \mathbf{x}_i) \cdot \mathbb{M}(\mathbf{x}) (\mathbf{x} - \mathbf{x}_i) \quad (4)$$

where $\mathbb{M} = |\mathbb{H}_u|$.

Integrating over the domain Ω , the total error estimate in L^p norm writes

$$E_{L^p} = \left(\sum_{i=1}^N \int_{\Omega_i} |e_i(\mathbf{x})|^p d\mathbf{x} \right)^{\frac{1}{p}}. \quad (5)$$

Using Equation (4), the L^p norm error is estimated as

$$E_{L^p} \approx \frac{1}{2} \left(\sum_{i=1}^N \int_{\Omega_i} \|\mathbb{M}^{\frac{1}{2}}(\mathbf{x})(\mathbf{x} - \mathbf{x}_i)\|^{2p} d\mathbf{x} \right)^{\frac{1}{p}}, \quad (6)$$

where $\mathbb{M}^{\frac{1}{2}}(\mathbf{x}) = \mathbb{P}^\top(\mathbf{x}) \text{diag}(\sqrt{|\lambda(\mathbf{x})|}, \sqrt{|\mu(\mathbf{x})|}) \mathbb{P}(\mathbf{x})$.

2.2 Link to the CVT energy

Now the mesh that minimises (6) is an anisotropic CVT [16]. For simplicity of mesh construction and robustness of the numerical scheme, an isotropic approximation $\phi^2 \mathbb{I}_2$, where $\mathbb{I}_2$ is the identity matrix and ϕ is a scalar field, can be substituted to the tensor $\mathbb{M}$. The derivation of a suitable ϕ field is discussed in the next paragraph.

With this expression, one finally gets the following isotropic L^p error estimation:

$$E_{L^p}^{\text{iso}} \approx \frac{1}{2} \left(\sum_{i=1}^N \int_{\Omega_i} \phi^{2p}(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^{2p} d\mathbf{x} \right)^{\frac{1}{p}}. \quad (7)$$

One would want to build the mesh that minimises (7). This is a well-known optimal quantisation problem for which local minima belongs to a class of Voronoi diagram called stable CVTs [17], which can be obtained by gradient descent and quasi-Newton algorithms [18, 8]. The asymptotic result from [19] relates the mesh density field ϕ^2 of a CVT to the size of the cells as N tends to $+\infty$. Using this result, the following asymptotically equivalent error estimation can be derived:

$$E_{L^p}^{\text{iso,asymptotic}} \approx \frac{1}{2} \left(\sum_{i=1}^N \int_{\Omega_i} \phi^{\frac{4p}{p+1}}(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\mathbf{x} \right)^{\frac{1}{p}}. \quad (8)$$

This final estimation makes a direct link with the concept of CVT energy with $\phi^{\frac{4p}{p+1}}$ as weighting field.

Metric isotropisation The aim is to find a scalar field $\phi(\mathbf{x})$ such that

$$\int_{\Omega_i} \phi^{2p}(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^{2p} d\mathbf{x} = \int_{\Omega_i} \|\mathbb{M}^{\frac{1}{2}}(\mathbf{x})(\mathbf{x} - \mathbf{x}_i)\|^{2p} d\mathbf{x}.$$

As it is already assumed that variations of $\mathbb{M}$ are small over a cell and because ϕ will be constructed from pointwise values anyway, the problem is recasted into finding, for any given index i , a scalar ϕ_i such that

$$\int_{\Omega_i} \phi_i^{2p} \|\mathbf{x} - \mathbf{x}_i\|^{2p} d\mathbf{x} = \int_{\Omega_i} \|\mathbb{M}^{\frac{1}{2}}(\mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)\|^{2p} d\mathbf{x},$$

which is given by

$$\phi_i^{2p} = \frac{\int_{\Omega_i} \|\mathbb{M}^{\frac{1}{2}}(\mathbf{x}_i)(\mathbf{x} - \mathbf{x}_i)\|^{2p} d\mathbf{x}}{\int_{\Omega_i} \|\mathbf{x} - \mathbf{x}_i\|^{2p} d\mathbf{x}}.$$

In order to derive a clean closed form expression, the integration domain is considered to be a circle of radius R , which is a reasonable approximation as long as the Voronoi cells are well shaped (which is the case when converging toward a CVT). With that final assumption, the following result holds:

$$\phi_i^{2p} = \frac{1}{4^p} \sum_{k=0}^p \frac{(2k)!(2(p-k))!}{(k!(p-k)!)^2} \lambda^k \mu^{p-k}.$$

where λ and μ are the eigenvalues of matrix $\mathbb{M}$. See Appendix A for details. This is as a convex combination of the binomial terms of degree p of the eigenvalues of $\mathbb{M}$. In particular

$$\begin{aligned} p = 1 & \quad \phi_i^2 = \frac{1}{2}(\lambda + \mu), \\ p = 2 & \quad \phi_i^2 = \frac{1}{2\sqrt{2}} \sqrt{3\lambda^2 + 2\lambda\mu + 3\mu^2}, \\ p = +\infty & \quad \phi_i^2 = \max(\lambda, \mu). \end{aligned}$$

Hessian tensor reconstruction To perform the mesh adaptation, the Hessian tensor is reconstructed on a reference mesh using a double L^2 projection [20]. Note that the requirement on u being C^2 -smooth is only theoretical as constructing the Hessian matrix estimation of a discontinuous field will lead to an implicit smoothing of the mesh density near discontinuities, leading to smooth mesh gradation which is beneficial for numerical schemes. In fact, after the construction of ϕ^2 from the Hessian approximation, a smoothing step is added to control the variation of the field over the whole computational domain and improve mesh gradation and quality. This smoothing procedure is taken from [7] and is restated briefly. The smoothed monitor function $\tilde{\phi}$ is computed from the raw monitor function ϕ as

$$(I + \alpha(1 + \alpha)L) \tilde{\phi} = \phi$$

where L is the graph Laplacian matrix of the reference mesh and α is a user parameter (a value of 0 corresponds to no smoothing while $\alpha \rightarrow +\infty$ leads to a uniform mesh).

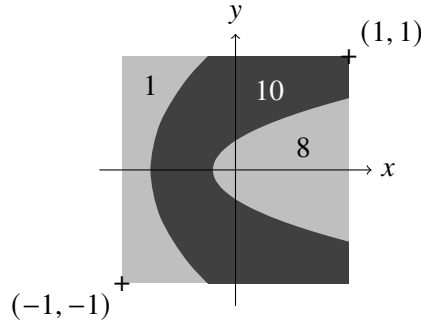


Figure 2: Two parabolic steps test case.

2.3 CFD coupling

Flow solver The Euler equations are solved using an in-house density-based cell-centered FV code. Fluxes are computed at the cell interfaces using the SLAU2 solver [21]. Second-order accuracy is achieved through the Monotone Upstream-centered Scheme for Conservation Laws (MUSCL) method [22] with the Barth-Jespersen limiter [23]. Steady-states numerical solutions are obtained via local time stepping.

Adaptation loop An initial uniform Voronoi mesh is constructed in order to perform the first computation. Note that one could also use inverse wall distance as the monitor function in order to accelerate the convergence of the adaptation loop [6]. For the next iterations, the monitor function is directly obtained from the Hessian tensor reconstruction. Each solution is projected onto the newly adapted mesh so that each CFD computation is initialised with a good guess of the final steady-state solution. To create the new mesh, generators are also sampled based on the previous mesh, which makes the overall adaptation loop quite efficient.

3. Numerical examples

Several test cases are provided to illustrate the versatility of the methodology.

Two parabolic steps An analytic test case is introduced with the aim of illustrating the effect of choosing different L^p norms.

The function is defined as

$$u = \begin{cases} 1 & \text{if } x \leq \frac{1}{2}y^2 - \frac{3}{4} \\ 10 & \text{if } \frac{1}{2}y^2 - \frac{3}{4} < x \leq 3y^2 - \frac{1}{5} \\ 8 & \text{otherwise} \end{cases}$$

and represented in Figure 2. It features two discontinuities of different amplitudes. This difference in amplitude is what makes the different L^p norms behave differently.

In the adaptation loop, instead of solving for a feature field via CFD computation, the analytical function u defined above is evaluated at the centroid of each cell. Then the Hessian tensor is reconstructed from these values as if it was a genuine physical field. It is finally smoothed with the parameter $\alpha = 1$.

For a fixed number of cells, the higher the p power, the more cells are brought toward the sharper discontinuity. This effect can be seen in Figure 3 where four meshes with $N = 10\,000$ cells were adapted according to different L_p norms. The small discontinuity is clearly visible in the L_1 and L_2 -adapted meshes and fades out in L_4 and L_8 -adapted meshes where cells are mainly focused toward the sharper discontinuity. Under the L_∞ norm, the adaptation process cannot converge. The choice of parameter p is therefore a tradeoff between capturing discontinuities as sharply as possible and capturing multiple flow

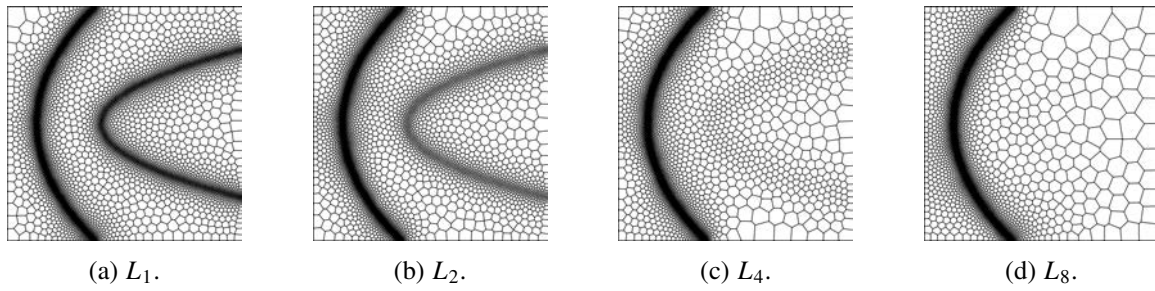


Figure 3: Meshes for several L^p norms (10 000 cells; 16 adaptations).

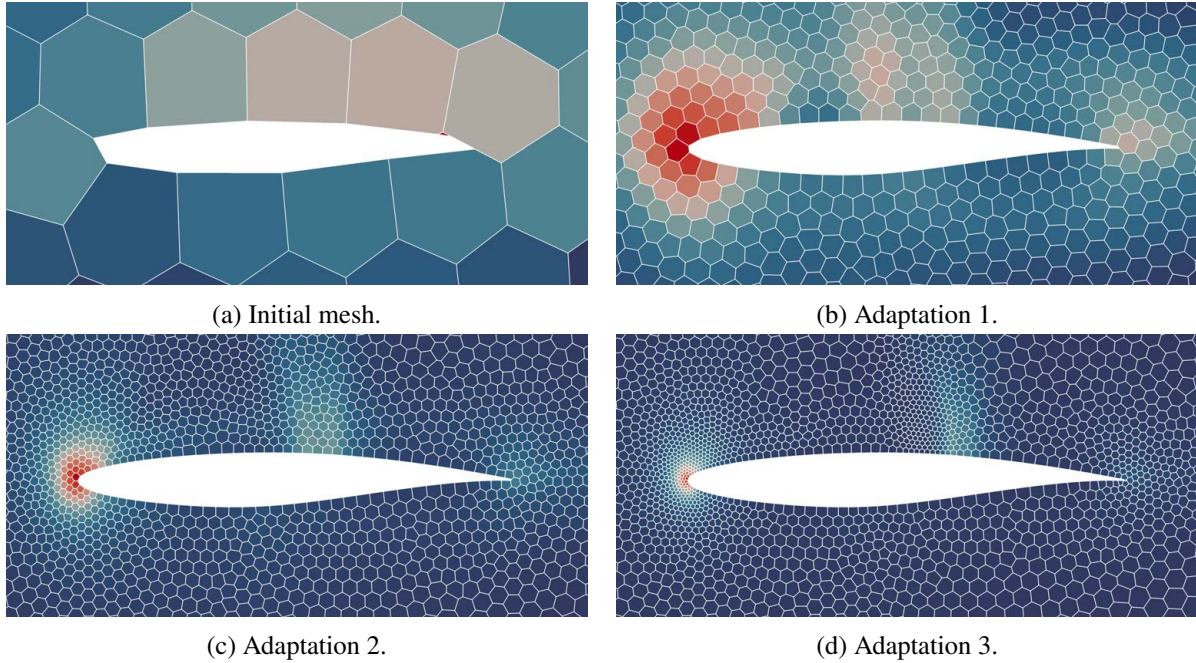


Figure 4: Near-wing view of the adapted meshes and smoothed monitor function (8000 cells per mesh).

features. A choice of $p = 2$ is considered a good tradeoff for inviscid computations, whereas $p = 4$ has been shown to perform better for viscous flows [2].

Transonic flow over the RAE2822 airfoil ($M_\infty = 0.729$) An inviscid transonic flow around the RAE2822 airfoil is considered. The free-stream Mach number is set to $M_\infty = 0.729$ and the angle of attack is $\alpha = 2.31^\circ$. The chord length is denoted by c . For this case and the others, a perfect gas is considered with a $\gamma = 1.4$ specific heats ratio.

The initially subsonic flow accelerates over the wing and reaches criticality at around 60% the chord length. The mesh illustrated by figure 5a is formed by 16 000 polygonal cells in a 10×10 chord lengths-wide square domain. It is obtained after 9 adaptations, starting from an initial uniform mesh with 8000 cells. The adaptation is done on the Mach number field in L^2 norm with a smoothing parameter set to $\alpha = 0.5$. To prove the robustness of the mesh adaptation process, the uniform near wing mesh is shown in Figure 4a as well as the first three adapted meshes obtained successively through the adaptation process in Figure 4. Blue to red colors corresponds to low to high values of the smoothed monitor function, giving locations to refine for the next adaptation step. It is shown that from a very coarse initial mesh (4 cells per chord length), the adaptation process is able to converge toward a well-suited mesh in few iterations.

The Mach number contour is provided in Figure 5b and shows that the shock is sharply captured near the airfoil. Upstream of the shock, the Mach number contours shows spurious oscillations due to mesh

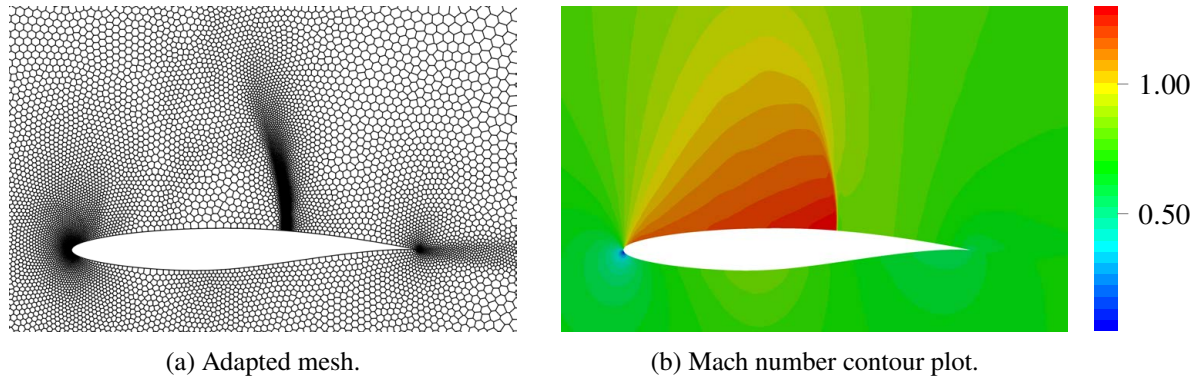


Figure 5: RAE2822 airfoil case (16 000 cells; 9 adaptations).

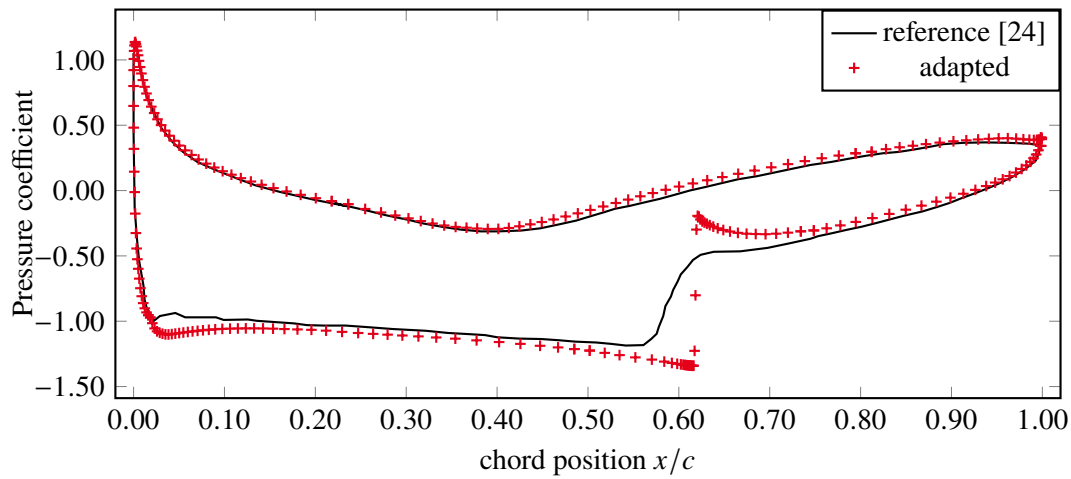


Figure 6: Pressure coefficient around the RAE2822 airfoil.

coarseness. If one would want to improve on the sharpness of the contour resolution, one can opt for a better balance between overall accuracy and shock resolution by considering the reconstruction error in L^1 norm. The pressure coefficient curve on the adapted mesh is shown in Figure 6. Compared to the reference computation [24], which is a 3rd-order accurate Discontinuous Galerkin (DG) computation done on a mesh formed by 20 816 triangles, Similar accuracy is achieved overall while providing a sharper shock ($x/c \approx 0.62$) and a better resolution near the stagnation point.

Supersonic flow through a duct with triangular bumps ($M_\infty = 2.0$) The supersonic flow at Mach $M_\infty = 2.0$ through a duct with two symmetric triangular bumps is considered. These bumps create shock waves that interact with each other. The full case specification can be found in [25]. The density plot along the line $y = 0.3$ is compared to the reference edge-based FV computation [25] performed on a uniform unstructured triangular grid with 11 250 nodes. This reference has the specificity of introducing slope limiters that are less dissipative than the more common Barth-Jespersen and Venkatakrishnan limiters. The adaptation of the 10 000 cells Voronoi mesh in Figure 7a is done in L_2 norm with density as the feature field and is obtained after three consecutive adaptations. As the flow is supersonic, no density gradient propagates back toward the inlet so that the raw monitor function is zero before the first shock and still quite close to zero after the smoothing step ($\alpha = 0.5$). This creates very large cells which do not degrade the accuracy of the numerical resolution as the density is constant throughout this region. Figure 7b shows the density contour plot. It can be seen in Figure 7a that small cells are created at the shocks and are even smaller in the areas where two shocks cross. The section plot at $y = 0.3$, presented in Figure 8, shows the same trend as the reference, obtained with the so called “R5” limiter, with improved shock resolution thanks to the adaptation process.

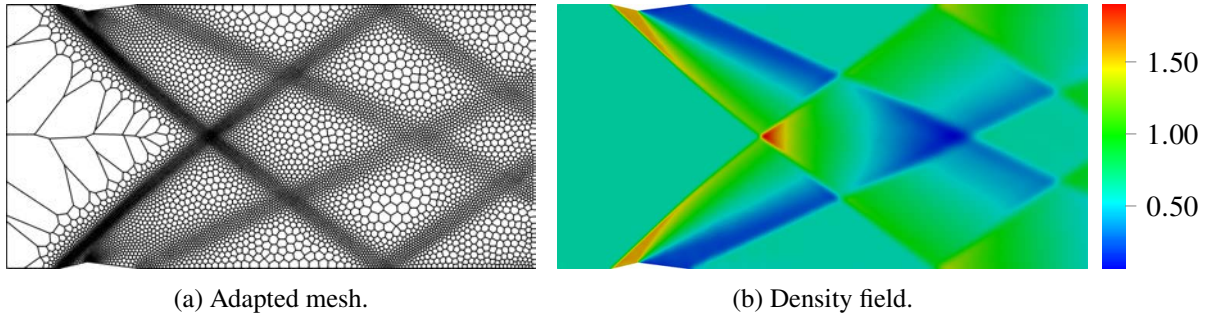


Figure 7: Supersonic duct with triangular bumps case (10 000 cells; 4 adaptations).

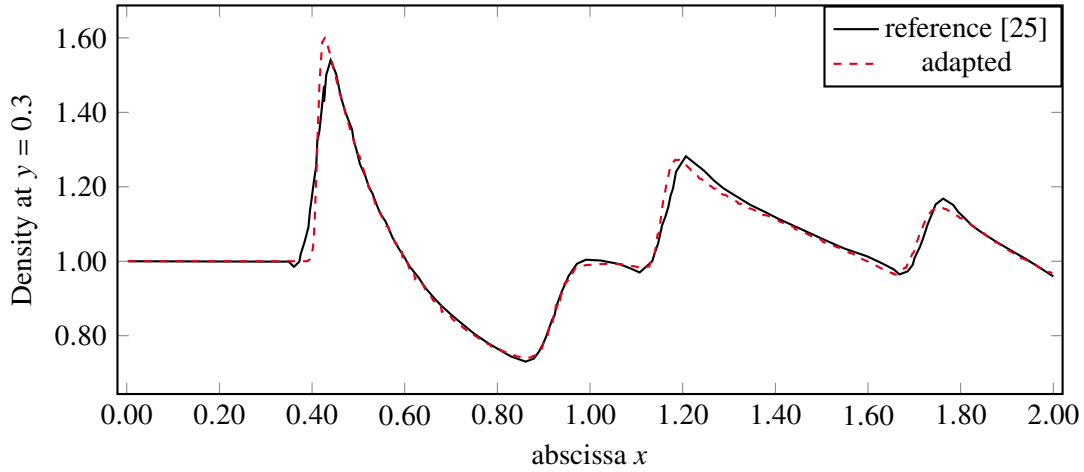


Figure 8: Section plot of the density at $y = 0.3$ for the supersonic duct.

Hypersonic flow around the double ellipse geometry ($M_\infty = 8.1$) A hypersonic flow ($M_\infty = 8.1$) around the double ellipse geometry is considered. The geometry specification can be found in [26]. The 16 000 cells mesh ($\alpha = 1$) and pressure field are given in Figure 9. Even if Voronoi meshes tend to damp the carbuncle effect, on coarse meshes, this effect has detrimental effect on pressure coefficient curves on the uniform mesh, see Figure 10. On the fourth mesh represented by Figure 9a however, the carbuncle phenomenon does not affect the pressure coefficient significantly anymore, while still being slightly visible on the pressure field in Figure 9b.

Hypersonic flow around a cylinder ($M_\infty = 17.6$) A $M_\infty = 17.6$ hypersonic flow is computed around a cylinder geometry. Despite the large carbuncle effect on the uniform mesh that can be seen on the pressure coefficient in Figure 12, like for the double ellipse case, the mesh adaptation process is still able to recover the physical solution after some adaptation steps. On the adapted mesh, the solution matches the analytical model from Lester Lees [27] closely near the stagnation point. This model is specifically formulated for high accuracy in this flow region but its validity degrades further downstream along the curvilinear abscissa. This explains the discrepancy with the numerical computations. The final mesh obtained after 3 adaptations at 32 000 cells ($\alpha = 1$) is visible in Figure 11a. The pressure field obtained on this mesh is given in Figure 11b and shows only a moderate carbuncle effect thanks to the Voronoi mesh and the adaptation process.

4. Conclusion and Future Work

A method for generating Voronoi meshes for arbitrary computational domains has been described. This method relies on four main steps: sampling of generators in the computational domain, constructing a

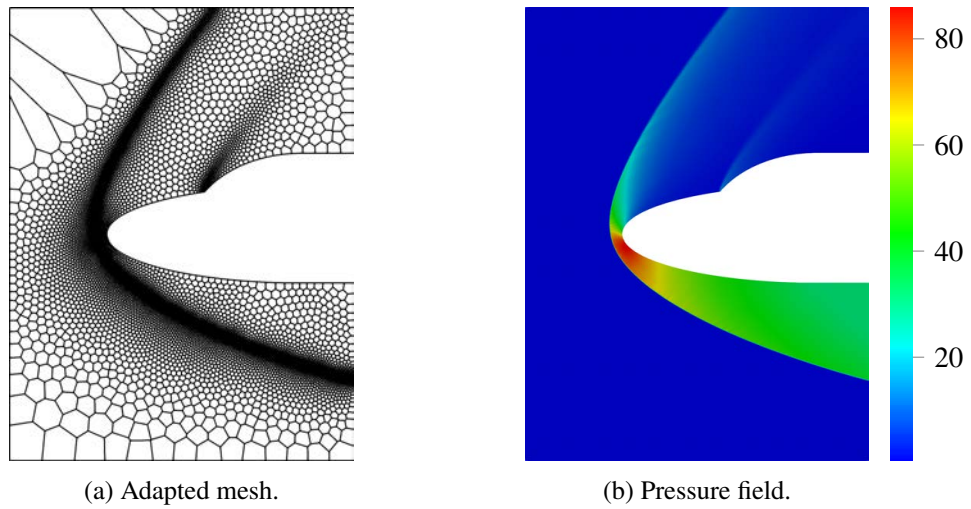


Figure 9: Double ellipse case (16 000 cells; 4 adaptations).

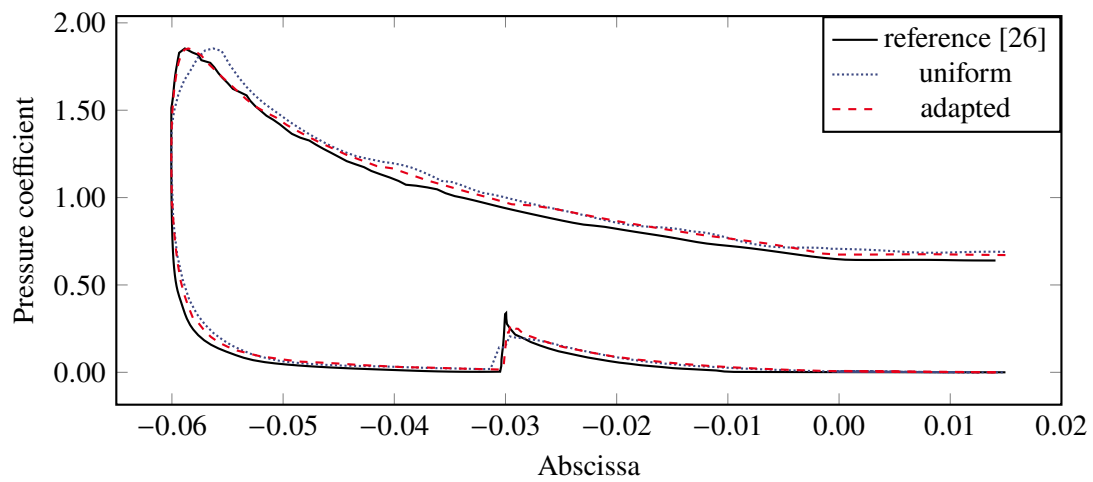


Figure 10: Pressure coefficient around the double ellipse.

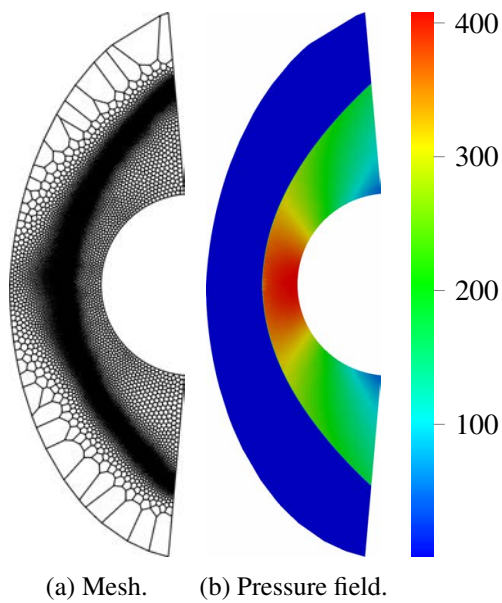


Figure 11: Cylinder (32 000 cells; 4 adaptations).

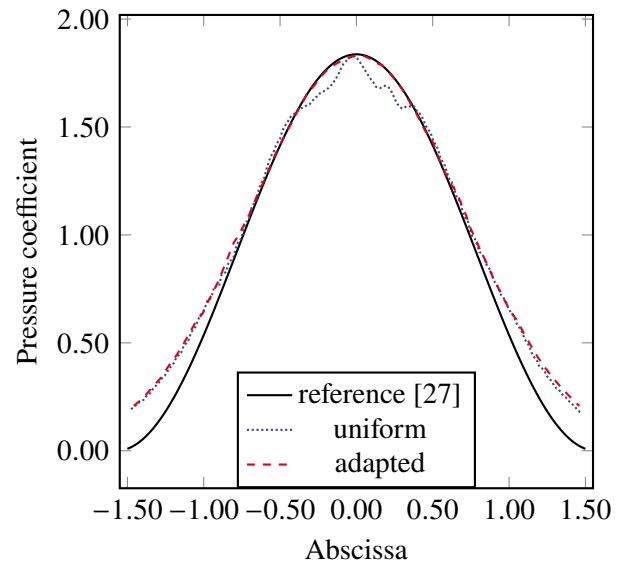


Figure 12: Pressure coefficient around the cylinder.

Delaunay triangulation from these seeds, dualising to obtain an unbounded Voronoi diagram and finally clipping to the domain boundary.

This mesh generation process is integrated into a mesh adaptation loop. Voronoi meshes have been shown to perform well across transonic, supersonic, and hypersonic flows over a variety of geometries. Moreover, their use can be mathematically justified over triangular or quadrilateral elements. From a very coarse initial mesh, the feature-based adaptation process is able to properly adapt to localised flow features to provide greater accuracy.

As a future work, the aim is to use direct Computer-Assisted Design (CAD) queries instead of using a fine-discretised version of the geometry in order to improve the fidelity of the whole process. Efforts will also go toward computing viscous flows by leaning toward anisotropic mesh generation or hybrid-structured meshes. Finally, the whole methodology, except the specific clipping strategy, generalises very well to 3D so that performance of this Voronoi mesh adaptation should be evaluated on 3D cases.

References

- [1] Guglielmo Vivarelli, Ning Qin, and Shahrokh Shahpar. A review of mesh adaptation technology applied to computational fluid dynamics. *Fluids*, 10(5):129, 5 2025.
- [2] Frédéric Alauzet and L. Frazza. Feature-based and goal-oriented anisotropic mesh adaptation for RANS applications in aeronautics and aerospace. *Journal of Computational Physics*, 439:110340, 8 2021.
- [3] Maurizio Pandolfi and Domenic D’Ambrosio. Numerical instabilities in upwind methods: Analysis and cures for the “carbuncle” phenomenon. *Journal of Computational Physics*, 166(2):271–301, 1 2001.
- [4] Laurent Muscat, Tony Marques, and Jérôme Breil. Mitigating carbuncle effects using Voronoi hybrid meshes. In *AIAA Aviation Forum and ASCEND 2024*, page 4506, 2024.
- [5] Emre Sozer, Aditya S. Ghate, Gaetan K. Kenway, Michael F. Barad, Victor C. Sousa, and Cetin C. Kiris. Evaluation of Voronoi meshes for large eddy simulations of high lift aerodynamics. In *AIAA SCITECH 2023 Forum*. American Institute of Aeronautics and Astronautics, 1 2023.
- [6] Brieuc Praud, Jérôme Breil, and Laurent Muscat. Polygonal mesh adaptation with cell size control for hypersonic flow simulations. In *Proceedings of the International Meshing Roundtable*. SIAM, 2025.
- [7] Wurigen Bo and Mikhail Shashkov. R-adaptive reconnection-based arbitrary Lagrangian Eulerian method. *Journal of Mathematical Study*, 48(2):125–152, 2015.
- [8] Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. On centroidal Voronoi tessellation—energy smoothness and fast computation. *ACM Transactions on Graphics (TOG)*, 28(4):1–17, 8 2009.
- [9] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 9 1991.
- [10] Célestin Marot, Jeanne Pellerin, and Jean-François Remacle. One machine, one minute, three billion tetrahedra. *International Journal for Numerical Methods in Engineering*, 117(9):965–990, 2019.
- [11] Dong-Ming Yan, Wenping Wang, Bruno Lévy, and Yang Liu. Efficient computation of clipped Voronoi diagram for mesh generation. *Computer-Aided Design*, 45(4):843–852, 2013.
- [12] Günther Greiner and Kai Hormann. Efficient clipping of arbitrary polygons. *ACM Transactions on Graphics (TOG)*, 17(2):71–83, 1998.
- [13] Jonathan Richard Shewchuk. Robust adaptive floating-point geometric predicates. In *Proceedings of the twelfth annual symposium on Computational geometry—SCG ’96*, pages 141–150. ACM Press, 1996.
- [14] Jorge J. Moré and David J. Thente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software*, 20(3):286–307, 9 1994.
- [15] Adrien Loseille and Frédéric Alauzet. Continuous mesh framework part I: well-posed continuous

- interpolation error. *SIAM Journal on Numerical Analysis*, 49(1):38–60, 2011.
- [16] Bruno Lévy and Yang Liu. L_p centroidal Voronoi tessellation and its applications. In *ACM SIGGRAPH 2010 papers*, pages 1–11. ACM, 7 2010.
 - [17] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.
 - [18] Masao Iri, Kazuo Murota, and Takao Ohya. A fast Voronoi-diagram algorithm with applications to geographical optimization problems. In *System Modelling and Optimization: Proceedings of the 11th IFIP Conference Copenhagen, Denmark, July 25–29, 1983*, pages 273–288. Springer, 1984.
 - [19] Qiang Du and Desheng Wang. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal for Numerical Methods in Engineering*, 56(9):1355–1373, 2003.
 - [20] Frédéric Alauzet. *Adaptation de maillage anisotrope en trois dimensions: Application aux simulations instationnaires en Mécanique des Fluides*. PhD thesis, Université des Sciences et Techniques du Languedoc (Montpellier II), 2003.
 - [21] Keiichi Kitamura and Eiji Shima. Improvements of simple low-dissipation AUSM against shock instabilities in consideration of interfacial speed of sound. In *Proceedings of the V European Conference on Computational Fluid Dynamics (ECCOMAS CFD 2010)*, page 1283, 6 2010.
 - [22] Bram Van Leer. Towards the ultimate conservative difference scheme. V. a second-order sequel to Godunov’s method. *Journal of Computational Physics*, 32(1):101–136, 7 1979.
 - [23] Timothy Barth and Dennis Jespersen. The design and application of upwind schemes on unstructured meshes. In *27th Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, 1 1989.
 - [24] Faheem Ahmed, Fareed Ahmed, Muhammad Affan, and Javed R. Larik. Numerical solution of 2D Euler equations for the transonic flow past over NACA0012 and RAE2822 airfoils using high order accurate Runge-Kutta Discontinuous Galerkin method. In *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST)*, pages 218–223. IEEE, 1 2014.
 - [25] Hiroaki Nishikawa. New unstructured-grid limiter functions. In *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, 1 2022.
 - [26] Jean-Antoine Désidéri, Roland Glowinski, and Jacques Périaux. *Hypersonic Flows for Reentry Problems: Volume II: Test Cases—Experiments and Computations Proceedings of a Workshop Held in Antibes, France, 22–25 January 1990*. Springer Science & Business Media, 2012.
 - [27] Lester Lees. Hypersonic flow. *Journal of Spacecraft and Rockets*, 40(5):700–735, 9 2003.

A. Metric isotropisation derivation

By changing the domain of integration to the eigenbasis of metric $\mathbb{M}(x_i)$, one gets

$$\phi_i^{2p} = \frac{\int_0^{2\pi} \int_0^R (r^2(\lambda \cos^2 \theta + \mu \sin^2 \theta))^p dr r d\theta}{\int_0^{2\pi} \int_0^R (r^2(\cos^2 \theta + \sin^2 \theta))^p dr r d\theta},$$

where λ and μ are the eigenvalues of matrix $\mathbb{M}$ (which are positives). This simplifies to

$$\phi_i^{2p} = \frac{1}{2\pi} \int_0^{2\pi} (\lambda \cos^2 \theta + \mu \sin^2 \theta)^p d\theta.$$

Using the binomial expansion of $(\lambda \cos^2 \theta + \mu \sin^2 \theta)^p$,

$$\phi_i^{2p} = \frac{1}{2\pi} \sum_{k=0}^p \binom{p}{k} \lambda^k \mu^{p-k} \int_0^{2\pi} \cos^{2k} \theta \sin^{2(p-k)} \theta d\theta,$$

then using the parity properties of sine and cosine functions,

$$\phi_i^{2p} = \frac{2}{\pi} \sum_{k=0}^p \binom{p}{k} \lambda^k \mu^{p-k} \int_0^{\frac{\pi}{2}} \cos^{2k} \theta \sin^{2(p-k)} \theta d\theta.$$

This integral is known as a first kind Euler integral for which a closed form expression can be derived:

$$\int_0^{\frac{\pi}{2}} \cos^{2k} \theta \sin^{2(p-k)} \theta d\theta = \frac{2\pi}{4^{p-1}} \frac{(2k)!(2(p-k))!}{p!k!(p-k)!}.$$

Injected in the previous expression, this leads to

$$\phi_i^{2p} = \frac{1}{4^p} \sum_{k=0}^p \frac{(2k)!(2(p-k))!}{(k!(p-k)!)^2} \lambda^k \mu^{p-k}.$$