# Anisotropic Mesh Adaptation for Transient Problems and Functional-Output Variables

Mostafa Najafiyazdi

Master of Engineering

Department of Mechanical Engineering

McGill University

Montreal, Quebec, Canada

Aug. 25, 2011


Copyright© by Mostafa Najafiyazdi, 2011.

# DEDICATION

I would like to dedicate this thesis to the two beautiful angles in my life who have made my life heaven,

My mother Shahrzad, who has always looked after me,

And my wife Atefeh, whose love is a godly gift.

# ACKNOWLEDGMENTS

I would like to thank my supervisor, Professor Wagdi G. Habashi, for his advices, guidance and support all throughout my Master's program. Studying and researching under his supervision has been an honour for me.

I am grateful for the generous and continuous contribution of NSERC (Natural Sciences and Engineering Research Council), Bombardier Aerospace (Foundation J-.Armand Bombardier), Bell Helicopter Textron and CAE Inc., under whose auspices this research is being carried out.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

Mesh adaptation is modification of an existing mesh as to accurately capture physical phenomena without increasing computational cost excessively. Among many challenges remained in this field, “unsteady mesh adaptation” and “functional-output mesh adaptation” are the two topics addressed in this work.

“Transient-Fixed-point” method is a new approach for unsteady mesh adaptation as an implicit coupling between the mesh and the solution. What differentiates transient-fixed-point from steady adaptation methods is the metric intersection through time. In this work a new scheme for transient metric intersection is proposed for orthogonal data preserving.

Another challenge in mesh adaptation is to control mesh modifications according to an integral function, known as functional-output, (e.g. lift, drag, moments, etc.) rather than the directly computed variables (e.g. velocity, pressure, Mach, etc.). Such control comes from a suitable edge-based error evaluation to be defined according to the chosen functional. One such definition is proposed in this work that drives surface mesh adaptation based on the chosen functional-output function.

# RÉSUMÉ

L’adaptation de maillage est la modification d'un maillage existant afin de saisir avec précision les phénomènes physiques sans augmentation excessive du coût de calcul. Parmi les nombreux défis qui demeurent dans ce domaine, l’adaptation de maillage de manière instable et l'adaptation de maillage selon un « functional-output » sont les deux thèmes abordés dans ce travail.

La méthode « Transient-Fixed-point » est une nouvelle approche pour l'adaptation de maillage de manière instable. Cette méthode consiste à formée un lien entre le maillage et la solution. Ce qui différencie la méthode « Transient-Fixed-point » de méthodes d'adaptation constante est l'intersection des métriques à travers le temps. Dans cet ouvrage, un nouveau régime pour l'intersection de métriques transitoire est proposé afin de préserver l’orthogonalité des données.

Un autre défi de l'adaptation de maillage est de contrôler les modifications apporté à la maille selon une fonction intégrale, connu sous le nom de « functional-output » (par exemple, la portance, la trainée, des moments, etc.) plutôt que de calculées directement les variables (par exemple vitesse, pression, Mach, etc.). Un tel contrôle provient d'une d'évaluation approprié de l’erreur des lisières à être définies en fonction des choix de « functional ». Une telle définition est proposée dans ce travail qui entraine l'adaptation de maillage de surface en fonction du« functional-output » choisi.

# Chapter 1 Introduction

In many numerical applications dealing with real-life problems, mesh adaptation is recognized as a complementary tool to achieve higher accuracy and lower overall computational time [1-5].

Mesh adaptation methods can be categorized from two different points of view: I) Regeneration vs. Modification, and II) Isotropic vs. Anisotropic.

Mesh adaptation can be performed as regeneration of mesh based on an error indicator computed on an original grid [6-14]. The error indicator is used to define a kernel (e.g. Delaunay kernel), which transforms the Euclidean space to a Riemannian space and performs the grid generation there. This transformation works perfectly for meshes fully constructed of tetrahedrons in 3D or triangles in 2D. Such regeneration ensures that Delaunay criterion is satisfied in the Riemannian space, which means that all edges or elements are almost equal in size (length or volume). However, in the Euclidean space it will result in a grid of non-uniform edges. Note that based on the definition of kernel (i.e. edge based or element based) the grid could be either isotropic or anisotropic.

The other approach is mesh modification. In this approach the grid is locally modified using four techniques: refinement, coarsening, edge swapping and node movement [1-3,15,16]. Based on how these operations are controlled, the adaptation could be isotropic or anisotropic. Habashi et al. believe that mesh modification has a major advantage over regeneration [3]. Mesh regeneration is costly with a constant overhead no matter how much the mesher-solver has approached a converged solution. But, mesh modification is less expensive, especially close to converged solution when there are only few modifications done by adaptor.

Another aspect of mesh adaptation is whether it is isotropic or anisotropic. Isotropic adaptations are based on error indicators defined on elements that do not contain any directional data. However, isotropic mesh adaptation can only be optimal with almost equal gradients in all spatial directions (e.g. diffusion

problems) [17-19]. In their results directional flow features such as shocks, boundary layers, wakes, slip lines and vortices will not be efficiently adapted and the number of elements required grows very rapidly by refinement.

The alternative approach is anisotropic adaptation allowing directional stretched and oriented elements [1-3,6,8,9,11,14,15,20-23]. In anisotropic mesh adaptations operations are performed on directional error indicators usually computed over edges of the mesh (ref. section 2.6). As the result, at areas of high curvature in the solution elements will be highly stretched so that a lot of flat elements lie side by side (Figure 1).

Therefore, the goal is to achieve a grid with equi-distribution of error estimation which is defined by Habashi *et al.* [3] as:

> "The error estimate has been equi-distributed when the error estimate is the same for all *edges* of a mesh."

Following operations are used to optimize a mesh with all edges having approximately the same error (Figure 2):

1. Edge refinement (*h*-refinement): When the Riemannian length (error estimate) of an edge is larger than a defined maximum threshold the edge is cut in half and new elements are created.
2. Edge or node coarsening: When the Riemannian length of an edge is smaller than minimum threshold or the Euclidian length is smaller than a user-defined minimum value, the edge is removed and elements are collapsed.
3. Edge swapping: An edge is swapped if there's an improvement in the Riemannian length of the edge (i.e. gets smaller).
4. Node movement (*r*-method): Node movement is the only smoothing operation during mesh adaptation. There are various global as well as local techniques for doing so. Global methods are based on solving a partial differential equation (e.g. Laplace) over the entire grid. However, since in this context the error computation as well as all other operations is performed locally, a local scheme is adopted for node movement.

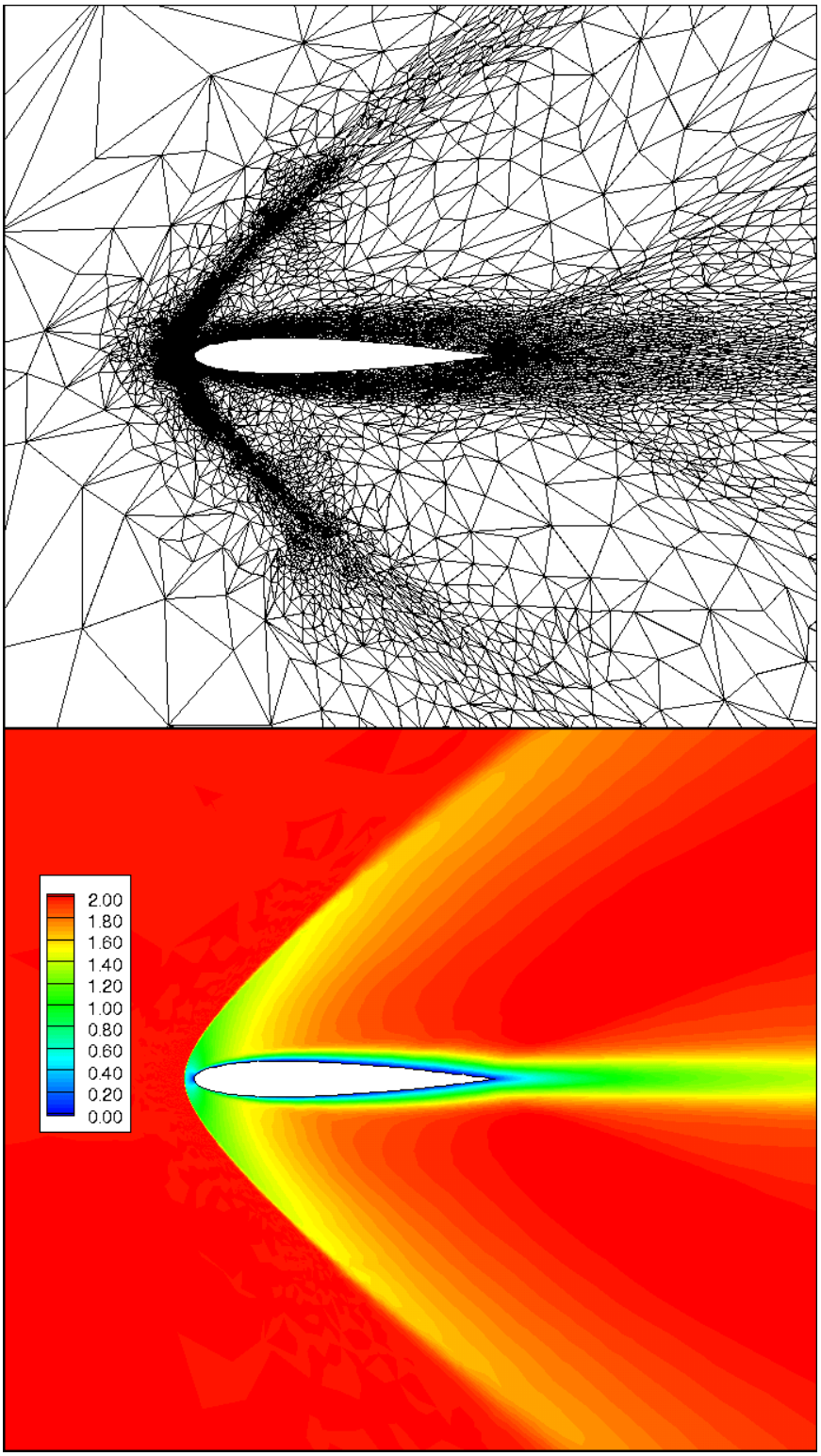



Figure 1: Anisotropic adapted mesh (top) and Mach contours (bottom) for an Euler Mach 2 flow over NACA0012 airfoil

In this work, spring analogy (Figure 3) method is used for its simplicity in implementation and no computational cost. In spring analogy, nodes are moved one at a time. To move each node, all its surrounding nodes are fixed and edges are considered to be springs under pre-tension or pre-compression.

The stiffness coefficient for each edge is defined as:

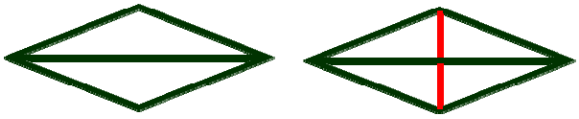
$$K_I\left(x_J\right)=\frac{e\left(x_I-x_J\right)}{L_{IJ}} \tag{1.1}$$

where $e\left(x_I-x_J\right)$ is the error associated to that edge and $L_{IJ}$ is its Euclidian length.

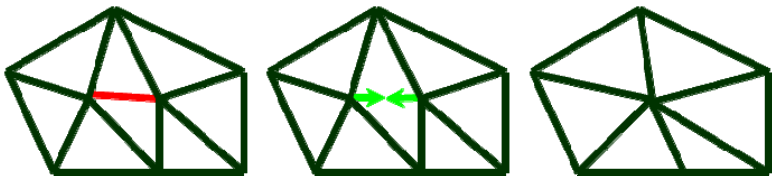
Putting the Euclidian length in denominator makes small edges stiffer in order to prevent the node to cross the boundaries of its surrounding elements and causing a degenerate element.

The ideal position of the node is where the energy of this spring system is minimized, i.e.:

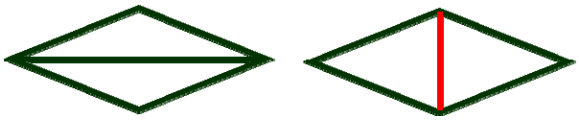
$$\min_{x_J\in\mathbb{R}^n} E\left(x_J\right)=\min_{x_J\in\mathbb{R}^n}\sum_{I=1}^{n}\left(x_I-x_J\right)^2 K_I\left(x_J\right) \tag{1.2}$$



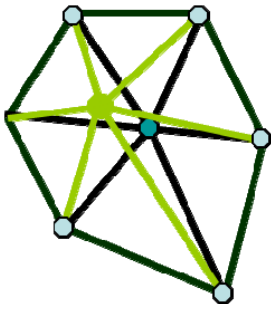
(a) Refinement



(b) Coarsening



(c) Swapping



(d) Node movement

Figure 2: Mesh adaptation operations

This minimization problem is equivalent to having zero net fore exerted on the node:

$$\sum_{I=1}^{n} F_I\left(x_J\right) = \sum_{I=1}^{n}\left(x_I - x_J\right) K_I\left(x_J\right) = 0 \qquad (1.3)$$

Equation (1.3) can be easily solved by an iterative fixed point method [3]:

$$x_J^{new} = x_J^{old} + \omega \frac{\sum_{I=1}^{n}\left(x_I - x_J^{old}\right) K_I\left(x_J^{old}\right)}{\sum_{I=1}^{n} K_I\left(x_J^{old}\right)} \qquad (1.4)$$

in which $\omega$ is a relaxation factor between zero and one.

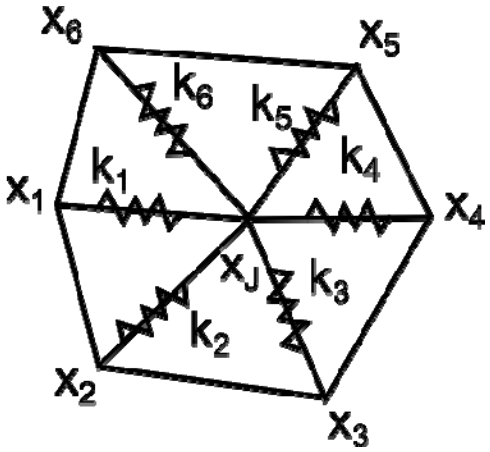



Figure 3: Spring analogy

### 1.1 Adaptor-solver coupling

The goal of mesh adaptation is to obtain improved meshes so that the coupled solver-adaptor converges to an optimal solution. The coupling is in the following way:

Consider a pair of mesh and a solution on this mesh $(H_n, S_n)$. The mesh is adapted based on error approximation of solution over this mesh and a new mesh $H_{n+1}$ is created. The solution $S_n$ is interpolated over this mesh and numerical computation is restarted over the new mesh from the interpolated solution. This iteration is continued until the solution is converged to a desirable level and does not change with mesh anymore.

In mesh adaptation, the choice of scalar to control the mesh plays a significant role in adaptation performance. For example, local Mach number is a very good choice for capturing shock waves and boundary layers, but not good enough to capture vortices in wakes or separation zones (as will be show in Chapter 3). An alternative choice could be scaled sum of all the variables [17,24] or the intersection of metrics associated with all or some of flow variables [12]. In this work, our focus is not on the choice of scalar or how multiple scalars are combined to a single one. The methods presented here are general and applicable to any choice of controlling scalar.

# Chapter 2 Error Estimation and Anisotropic Mesh Adaptation

This chapter focuses on error estimation techniques, as the heart of adaptive computational methods used in the context of mesh adaptation. The difference between *a priori* and *a posteriori* error approximations is briefly discussed and mathematical modeling details are explained for case of *a posteriori* error estimation. Moreover, error estimation application is discussed in mesh adaptation where the concept of anisotropy is introduced.

## 2.1 Error Estimation

All numerical computations are based on mathematical models, i.e. a system of governing equations (algebraic, ordinary or partial differential, integral, integro-differential), inequalities, boundaries and initial conditions, data (set of parameters, domain geometry). Such mathematical models might not be able to fully describe the actual characteristics of a physical phenomenon, which consequently introduce an error in the simulations. Another source of error is solving the mathematical models numerically which is caused by discretization of "continuous" equations and/or boundary as well as initial conditions, or because of round off errors arising from machine precision.

In this work, the term "error" is used for the second category (i.e. numerical deficit with respect to a given mathematical model) and the validity of governing equations and/or boundary conditions is excluded.

In any numerical computation, the error in solution is a function of the independent variables of the problem. For example for an unsteady CFD [1] simulation the independent variables are spatial coordinates (e.g. $x$ and $y$, or $r$ and $\theta$ in 2D[2] flows) and time $t$.

The error for a numerical simulation is simply defined as:

$$e = u - u_a \tag{2.1}$$

[1] Computational Fluid Dynamics
[2] 2-dimensional

where $u$ is the exact solution of the governing mathematical model and $u_a$ is the numerical approximation obtained by for example finite differences, finite volume or finite element methods. Surely, for models consisting of system of equations the error can be defined for any dependent variables for which the model is solved or for any other quantity based on obtained solution.

The magnitude of spatial approximation error and its bounds are measured by norms and semi-norms defined on the computational domain. Ciarlet [10] has proved that the approximation error is always bounded by the interpolation error of the discretized space:

$$\|u - u_a\| \leq C\|u - u_I\| \tag{2.2}$$

where $\|\cdot\|$ is an $L_P$ norm and $u_I$ is the interpolate of solution $u$ in the finite element space and $C$ is just a positive constant. Therefore, the approximation error is bounded by the interpolation error. Determining suitable bounds for interpolation error provides a bound on the actual error.

Defining bounds for interpolation error has been a topic of research for many years during which two different paths have been used: a priori and a posteriori estimates.

A priori estimates are based on mathematical theories that do not require information from the actual finite element solution and can be computed prior to the determination of the solution [10, 24, 25]. The advantage of these methods is that an approximation of error and its dependency on the domain discretization is at hand, even before the numerical computation is performed. But they are limited in their application as they highly depend on the discretization of the governing equations and the computational domain. Moreover, they usually are computed globally and provide an upper bound of interpolation error throughout the domain and ensure a certain level of convergence. Although suitable for determining convergence rate of a numerical scheme, it has little use for mesh adaptation in which error must be approximated locally at every point inside the computational domain.

A posteriori error estimates derive information from the finite element solution itself in order to approximate the interpolation error. Their only disadvantage rises from their need of computing the solution on a given discretized space. But, their strength is in providing point-wise information about the interpolation error, which is the key point for mesh adaptation. Therefore, in this thesis a posteriori estimates will be used.

### 2.2 A Posteriori Error Approximation

Precise estimates of error are usually difficult and/or very expensive to evaluate for highly complex problems. Moreover, since the error estimate is only an indication of relative error in the context of mesh adaptation, it is more suitable to use less precise and less expensive methods for error estimation. A simple but very efficient error estimator was introduced by Habashi *et al.* [3], which is described in details. This error estimator is based on finite element interpolation theory and Lagrange error bound. For linear elements the error term is proportional to the second derivative of a chosen scalar. Here, the derivation for a one-dimensional problem is illustrated.

Consider a continuous field $u(x)$ is approximated by a piecewise linear distribution $u_h(x)$ as shown in Figure 4. At every point in element *E* the error is defined as:

$$e_E(x) = u(x) - u_E^h(x) \tag{2.3}$$

where the approximate solution $u_E^h(x)$ can be expressed in terms of nodal values of the element *E*. Moreover, the solution value at point $I+1$ ( $u_{I+1}$ ) can be expanded as a Taylor series about node $I$ :

$$u_h(x) = \left(1 - \frac{x}{l_E}\right) u_I + \frac{x}{l_h} u_{I+1} \tag{2.4}$$

$$u_{I+1} = u_I + l_E \frac{du_I}{dx} + \frac{l_E^2}{2} \frac{d^2 u_I}{dx^2} + \cdots \tag{2.5}$$

Substituting equation (2.4) into equation (2.5) gives:

$$u_h(x) = u_I + x \frac{du_I}{dx} + \frac{x l_E}{2} \frac{d^2 u_E}{dx^2} + \cdots \tag{2.6}$$

Expansion of the exact solution $u(x)$ about the point $I$ gives:

$$u(x) = u_I + x\frac{du_I}{dx} + \frac{x^2}{2}\frac{d^2u_E}{dx^2} + \cdots \tag{2.7}$$

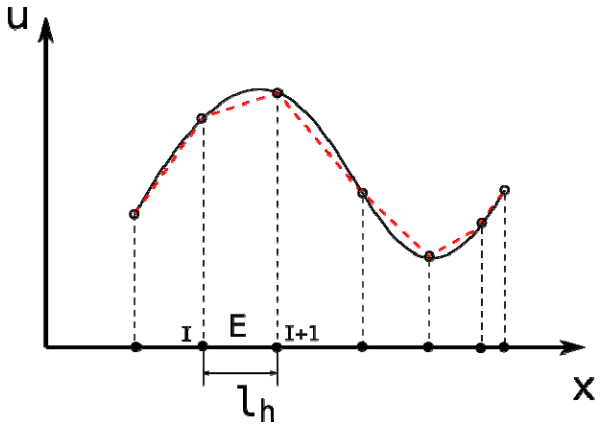



Figure 4: Schematic discrete solution on linear elements

By replacing equations (2.6) and (2.7) into equation (2.3) and neglecting higher order terms $O(\Delta x^3)$ one can see that:

$$e_E(x) \simeq \left(\frac{x^2}{2} - \frac{xl_E}{2}\right)\frac{d^2u_E}{dx^2} \tag{2.8}$$

So, the interpolation error for a one-dimensional element is proportional to the product of second derivative of the solution and the square of element length.

For 2D and 3D cases, the second derivative of solution is the Hessian matrix:

$$\mathbf{H}_h(x) = \begin{pmatrix} \frac{\partial^2 u_h}{\partial x^2} & \frac{\partial^2 u_h}{\partial x \partial y} & \frac{\partial^2 u_h}{\partial x \partial z} \\ \frac{\partial^2 u_h}{\partial y \partial x} & \frac{\partial^2 u_h}{\partial y^2} & \frac{\partial^2 u_h}{\partial y \partial z} \\ \frac{\partial^2 u_h}{\partial z \partial x} & \frac{\partial^2 u_h}{\partial z \partial y} & \frac{\partial^2 u_h}{\partial z^2} \end{pmatrix} \tag{2.9}$$

The second derivative of the solution in the direction of a given unit vector $\vec{\mathbf{e}}$ is defined as:

$$\frac{\partial^2 u_h}{\partial \vec{\mathbf{e}}^2} = \vec{\mathbf{e}}^T \mathbf{H}_h(x)\vec{\mathbf{e}} \tag{2.10}$$

Therefore, the error corresponding to an edge can be defined as follows:

$$e_{edge} = \int_0^1 \sqrt{\left(\vec{x}_I - \vec{x}_J\right)^T \mathbf{M}\left(\vec{x}_I - \vec{x}_J\right)} ds \tag{2.11}$$

where $\vec{x}_1$ and $\vec{x}_J$ are the position vectors of the two end points of the edge and $\mathbf{M}$ is a positive definite matrix obtained from $\mathbf{H}$. Note that for the definition in equation (2.11) the metric $\mathbf{M}$ must be positive definite.

Various definitions for metric **M** are proposed [3,4,10,12,26]. The fundamental definition of **M** is the same in all these. The Hessian matrix **H** can be decomposed into eigenvectors and eigenvalues matrices:

$$\mathbf{H} = \mathbf{R}^T \mathbf{\Lambda} \mathbf{R} \tag{2.12}$$

where **R** is the matrix of eigenvectors and **Λ** is the diagonal eigenvalue matrix. By taking the absolute value of eigenvalues, one can ensure that the resulting matrix is positive definite. Moreover, since the Hessian matrix is symmetric, then the metric defined in equation (2.13) can be used to define a Riemannian space.

$$\mathbf{M} = \mathbf{R}^T \left|\mathbf{\Lambda}\right| \mathbf{R} \tag{2.13}$$

For a symmetric positive definite metric $\mathbf{M}\left(\vec{x}\right)$ the arc length of a curve $\Im$ in the Riemannian space associated to it is given by:

$$d_M\left(\Im\right) = \int_0^1 \sqrt{\vec{\mathbf{s}}'(t)^T \mathbf{M}\left(\vec{\mathbf{s}}(t)\right)\vec{\mathbf{s}}'(t)} dt \quad \text{for } t \in [0,1] \tag{2.14}$$

where $\vec{\mathbf{s}}(t)$ is the parametric representation of the curve $\Im$.

Comparing equations (2.11) and (2.14) shows that the definition of error for an edge is actually its Riemannian length corresponding to the metric **M** defined in equation (2.13).

### 2.3 Geometrical Representation of Riemannian Metrics

Riemannian metrics (i.e. positive-definite symmetric matrices) have geometrical representatives in form of (hyper) ellipsoids. It means that in 2D a metric can be shown by an ellipse and in 3D by an ellipsoid centered at the origin. Here, for simplicity derivation is shown for a 2D case which is easily extendable for higher dimensions as well. Consider a $2 \times 2$ metric:

$$\mathbf{M}_2 = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \tag{2.15}$$

The ellipse corresponding to this metric can be defined as:

$$f(x,y)=(x,y)\mathbf{M}_2\begin{pmatrix}x\\y\end{pmatrix}=1 \tag{2.16}$$

$$ax^2+2bxy+cy^2=1 \tag{2.17}$$

By rewriting equation (2.17) in canonical form, one can show that the ellipse defined by this equation is an ellipse with major and minor axes parallel to eigenvectors, and with diagonals equal to the inverse of the square root of eigenvalues of the metric. Figure 5 shows a schematic drawing of such an ellipse. Geometrical representation of the metric shows the principal directions and magnitudes of error based on definition in equation (2.11). Note that higher values of error along an edge correspond to larger eigenvalues and smaller diagonals for the ellipse. So, a similar ellipse known as “error ellipse” can be defined whose axes are perpendicular to that of the metric ellipse. The length of diagonals of the error ellipse is equal to inverse of diagonals of the metric ellipse (Figure 6).

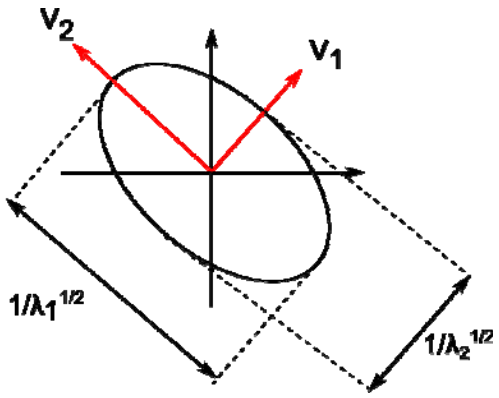



Figure 5: Schematic drawing of an ellipse corresponding to a metric

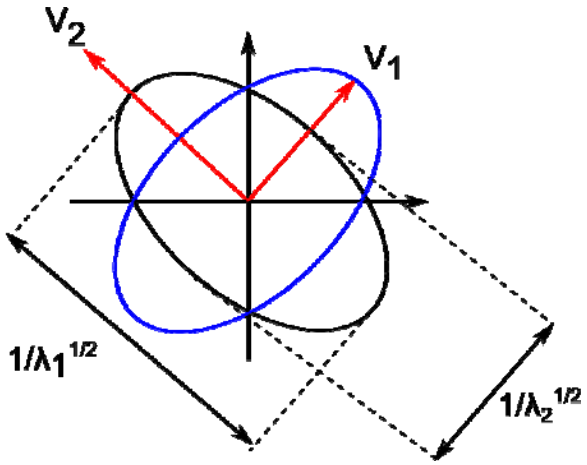



Figure 6: Error ellipse (blue) in comparison to metric ellipse (black)

### 2.4 Second Derivative (Hessian) Recovery

The first step in evaluating the error over each edge using equation (2.11) is to compute the Hessian matrix at every node. Usually solutions do not have second derivatives for the dependent variables. In order to have continuous second derivatives accurately a numerical scheme must be at least $4^{th}$ order in space which rarely is the case. Therefore, one needs to approximate the Hessian matrix using derivative recovery methods. In this chapter we show a few derivative recovery schemes that can be used for solutions over linear-element grids.

Derivative recovery methods (most of which are focused on recovery of first and second derivatives) are more or less classified in terms of global and local methods. Global methods require solving a linear system of equations on the whole computational domain while local methods define the problem locally and usually depend only on a few cells around a node. Global methods are seldom used in practice as they are believed to be computationally expensive.

Hessian recovery techniques are fairly new with main application in anisotropic mesh adaptation that needs directional error indicators. Most Hessian recovery methods are either applying gradient recovery twice or by extending them [24].

Recovery techniques have been mainly developed since the late 1990's. Zienkiewicz and Zhu [25] proposed smoothing the gradient using a local discrete least-squares fitting operator. This technique for error indicators shows super

convergent asymptotic exactness on structured grids, but not for unstructured grids [27, 28].

Another family of recovery techniques is based on the $L^2$-projection on a set of functions. These schemes are developed both globally [14, 29] and locally using the Clément interpolation operator [21, 30].

Few works are published on comparison of Hessian recovery methods. Buscaglia *et al.* [29] compared only their improved version of another technique [3,12]. Almeida *et al.* [21] compare four gradient recovery techniques by plotting some convergence graphs of gradients and Hessian errors. Vallet *et al.* [31] propose a methodology to compare recovery techniques for second-order derivatives from a piecewise linear approximation and compare four different methods, DLF[3], SLF[4], QF[5] and DL2P[6]. They believe that gradient extensions (DLF and QF) should be more accurate than the use of a gradient twice.

In this work, for Hessian recovery first the gradient vector is recovered at each node. A least-square fitting with a constant function at element centers (as sampling point) is used for recovering gradient. In this technique, nodal values are computed on a patch $T_I$ associated with a node $I$. This patch is considered to be the support of that node, i.e. the union of elements around it. This technique is actually the first step in DLF scheme (called Local Polynomial Expansion in [21]). Starting from:

$$du_I = \frac{\partial u}{\partial x}dx + \frac{\partial u}{\partial y}dy + \frac{\partial u}{\partial z} \tag{2.18}$$

one can rewrite it approximately for every cell as:

$$\Delta u_j = \left(\frac{\partial u}{\partial x}\right)_I \Delta x_j + \left(\frac{\partial u}{\partial y}\right)_I \Delta y_j + \left(\frac{\partial u}{\partial z}\right)_I \Delta z_j \tag{2.19}$$

where $\left(\Delta x_j, \Delta y_j, \Delta z_j\right)$ is the vector from the node to the center of $j$th element in the support (Figure 7).

---

[3] Double Linear Fitting
[4] Simple Linear Fitting
[5] Quadratic Fitting
[6] Double $L^2$-projection

To solve for $(\partial u/\partial x)_I$ , $(\partial u/\partial y)_I$ , and $(\partial u/\partial z)_I$ , a least-square problem can be formed:

$$\begin{aligned}
\sum_{j\in T_I}\Delta u_j\Delta x_j &= \sum_{j\in T_I}\left(\frac{\partial u}{\partial x}\right)_I \Delta x_j{}^2 + \sum_{j\in T_I}\left(\frac{\partial u}{\partial y}\right)_I \Delta y_j\Delta x_j + \sum_{j\in T_I}\left(\frac{\partial u}{\partial z}\right)_I \Delta z_j\Delta x_j \\
\sum_{j\in T_I}\Delta u_j\Delta y_j &= \sum_{j\in T_I}\left(\frac{\partial u}{\partial x}\right)_I \Delta x_j\Delta y_j + \sum_{j\in T_I}\left(\frac{\partial u}{\partial y}\right)_I \Delta y_j{}^2 + \sum_{j\in T_I}\left(\frac{\partial u}{\partial z}\right)_I \Delta z_j\Delta y_j \\
\sum_{j\in T_I}\Delta u_j\Delta z_j &= \sum_{j\in T_I}\left(\frac{\partial u}{\partial x}\right)_I \Delta x_j\Delta z_j + \sum_{j\in T_I}\left(\frac{\partial u}{\partial y}\right)_I \Delta y_j\Delta z_j + \sum_{j\in T_I}\left(\frac{\partial u}{\partial z}\right)_I \Delta z_j{}^2
\end{aligned} \tag{2.20}$$

Equation (2.20) can be written in matrix form:

$$\begin{bmatrix} \sum\Delta x_j{}^2 & \sum\Delta x_j\Delta y_j & \sum\Delta x_j\Delta z_j \\ \sum\Delta x_j\Delta y_j & \sum\Delta y_j{}^2 & \sum\Delta y_j\Delta z_j \\ \sum\Delta z_j\Delta x_j & \sum\Delta x_j\Delta y_j & \sum\Delta z_j{}^2 \end{bmatrix}\begin{bmatrix} \partial u/\partial x \\ \partial u/\partial y \\ \partial u/\partial z \end{bmatrix} = \begin{bmatrix} \sum\Delta u_j\Delta x_j \\ \sum\Delta u_j\Delta y_j \\ \sum\Delta u_j\Delta z_j \end{bmatrix} \tag{2.21}$$

Solving this system of equations gives a least-square approximation of gradient at every point.

If the same procedure is used for computing the second derivatives, the method is the DLF. However, DLF performs poorly when high gradients exist in the solution (e.g. shock waves or boundary layers) and results in highly oscillating values for the Hessian matrix.

In order to avoid such non-physical oscillations, a weak Galerkin weighted average technique can be used [15]. Second derivatives at a point can be defined in a weak Galerkin sense as:

$$\left.\frac{\partial^2 u}{\partial x_i{}^2}\right|_I = \frac{\int_{T_I}\sum_k N_k\frac{\partial^2 u_k}{\partial x_i{}^2}d\Omega}{\int_{T_I}\sum_k N_k d\Omega} \tag{2.22}$$

where $k$ loops over all nodes of surrounding elements and $N_k$ are shape functions at those nodes. Equation (2.22) can be integrated by parts:

$$\left.\frac{\partial^2 u}{\partial x_i{}^2}\right|_I = \frac{\int_{T_I}\sum_k N_k\frac{\partial^2 u_k}{\partial x_i{}^2}d\Omega}{\int_{T_I}\sum_k N_k d\Omega} = \frac{-\int_{T_I}\sum_k \frac{\partial N_k}{\partial x_i}\frac{\partial u_k}{\partial x_i}d\Omega + \oint_\Gamma N_k\frac{\partial u_k}{\partial x_i}d\Gamma}{\int_{T_I}\sum_k N_k d\Omega} \tag{2.23}$$

In this equation $\Omega$ is the interior of the support and $\Gamma$ represents its boundary (Figure 8).

Equation (2.23) can be broken in terms of summation of integrals over elements:

$$\left.\frac{\partial^2 u}{\partial x_i^{\,2}}\right|_I = \frac{\sum_{m\in T_I}\sum_{k=1}^{ndperl}\int_l N_k \frac{\partial^2 u_k}{\partial x_i^{\,2}} d\Omega}{\sum_{l\in T_I}\sum_{k=1}^{ndperl}\int_l N_k d\Omega} = \frac{\sum_{M\in T_I}\sum_{k=1}^{ndperl}\left(-\int_M \frac{\partial N_k}{\partial x_i}\frac{\partial u_k}{\partial x_i} d\Omega + \oint_m \frac{\partial u_k}{\partial x_i} N_k d\Gamma_m\right)}{V_T} \qquad (2.24)$$

where $M$ , $\Gamma_M$ and $V_T$ respectively represent elements, boundaries of elements (i.e. faces for a 3D element and edges for a 2D element), and total volume of the support. Note that during integration the boundary integral over internal facets of two adjacent elements will cancel each other out. This means that the boundary integral is only needed for boundary facets (if there are any).

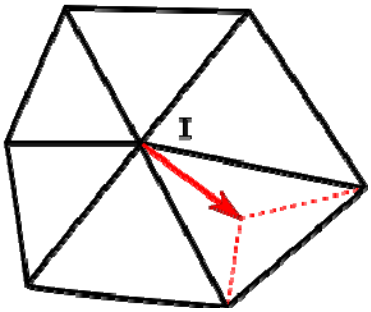



Figure 7: A sample chosen vector for one of elements around the node $I$

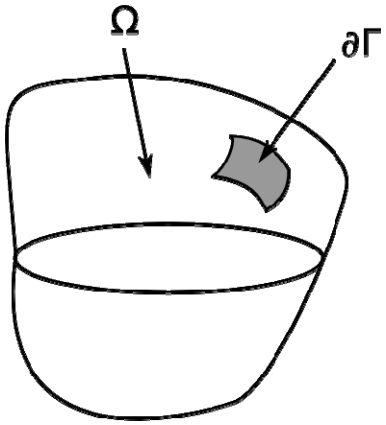



Figure 8: Schematic drawing of a support and its boundary

# Chapter 3 Unsteady Mesh Adaptation

The classical mesh adaptation algorithms for steady state problems are known to be inadequate for unsteady simulations [4,32]. First of all, in steady-state algorithms the mesh is always "lagging" with respect to the solution. A proposed remedy is to adapt the mesh so frequently that the evolving solution through a time step is contained in the suitably refined regions of the mesh [8,33,34] or to add a safety area around critical regions to be adapted similarly [17,35,36]. Secondly, frequent mesh adaptations might compensate for the problem to some extent, but will introduce a source of error due to interpolation of the solution from one mesh to another (unless adaptation is only based on refinement and/or coarsening or pure node movement using ALE methods). Interpolation errors prevent the simulation from being time-accurate, especially when they stack up during a large number of adaptations. Although a mesh adaptation with only refinement and/or coarsening would be free from this flaw, it cannot provide anisotropy, a powerful and important feature in mesh optimization for reducing computational costs while increasing accuracy [1,2,37].

Recently, a new scheme was proposed by Alauzet et al [4] which combines the classical mesh adaptation with a "Transient-Fixed-Point" scheme [32]. In this method for a chosen time interval one iterates between flow solver and mesh adaptation to obtain a converged mesh-solution couple, which means the variation of solution over two consecutively adapted meshes is smaller than a user-defined threshold. Therefore, the mesh is adapted to be suitable for the time interval containing several time steps of the numerical simulation by intersecting metrics computed from each (or some) of solutions stored at every (or some) time-steps. This proposed method seems to be promising because it eliminates the need for adapting frequently while minimizing the number of solution interpolations. Choosing a suitable adaptation interval is case dependant and not trivial. Alauzet recommends an interval in which 10 to 25 solution time steps have been saved (as a rule of thumb) [4,32]. If the interval is too large the effect of many physical phenomena will be combined which would result into either a very large mesh

due to many refinements or an isotropically adapted mesh (e.g. uniform adaptation in the wake of a vortex-shedding cylinder). On the other hand, if the interval is too small, many adaptations are needed which means expensive computational costs. A suitable interval can be chosen by investigating the rate of evolution of physical phenomena on the computational domain such as shedding vortices, shock wave reflections, boundary movement or deformation, and etc.

For an unsteady solution the solution at last time step of an adaptation interval must be used as a restart solution for the next period by interpolating it on the meh of next interval. Since this solution is common between both adaptation periods, meshes of both periods are already adapted for it. In other words, both meshes are good enough to ensure a minimum accuracy for interpolation in elements. Therefore, one can conclude that interpolating from one mesh onto the other will not cause interpolation errors larger than the target error for mesh adaptation. It means that the simulation will remain time-accurate with interpolation errors less than the threshold chosen for the adaptation at all computational nodes.

This new technique has been tested for a few 2D as well as 3D test cases [4] to show that such metric intersection provides a suitable mesh for the specified time period if a sufficient number of iterations is carried out between solver and mesh adaptor.

This work looks further into metric intersection schemes, particularly a modified version of the metric intersection scheme by McKenzie *et al*. [4]. The proposed method is based on intersecting metrics by using the ellipse representation of positive definite tensors and will ensure the exact intersection of any two ellipses (as the geometrical representation of metric tensors). The present work introduces a suitable transformation for error matrices to another geometrical space, where one of the error matrices becomes indifferent to direction and therefore the intersection preserves orthogonality of eigenvectors at all steps of the procedure. In other words, in this Riemannian transformed domain, eigenvalues of a matrix are identical, i.e. its ellipse representation will be a unit circle. This method of metric intersection ensures that one will achieve the exact intersection of as many

number of the matrices as desired in contrast to the technique proposed in [4], which does not guarantee this criterion.

A suitable metric intersection means a more accurate error approximation for unsteady problems, resulting in faster convergence towards the fixed-point of mesh-solution couple. In the following sections, the proposed approach is presented and discussed in details. We will first recall the classical mesh adaptation algorithm and review the transient-fixed-point scheme for unsteady problems.

In order to predict the temporal evolution of physical phenomena, transient-fixed point mesh adaptation combines the classical and fixed-point coupling schemes [32]. This iterative algorithm is a dual-loop procedure, where at each step of its outer loop an adaptation time interval $[t, t+\Delta t_{ad}]$ is chosen. From the solution at time $t$, numerical computations are done to reach $t+\Delta t_{ad}$ in a number of simulation time steps $N_s$ with time steps of $\Delta t_s < \Delta t_{ad}$. Using the solutions of these time steps, the mesh is adapted and the same time-interval simulation is performed. This internal loop is repeated until a convergence on the solution is achieved, i.e. until an $L^P$ norm of the variations of the solution at the final simulation step.

$$t = t + \Delta t_{ad} = t + \sum_{i=1}^{N_s} \Delta t_{s,i} \tag{3.1}$$

is lower than a specified threshold. Alauzet [4] suggests using the $L^1$ norm for computing the variations:

$$\frac{\left\| S_i - S_{i+1} \right\|_{L^1(\Omega)}}{\left\| S_{i+1} \right\|_{L^1(\Omega)}} \leq \varepsilon_{inner} \tag{3.2}$$

where $\Omega$ represents the computational domain, $S_i$ and $S_{i+1}$ are the solutions from two consecutively adapted meshes and $\varepsilon_{inner}$ is the threshold defined for the inner loop.

When the inner loop has converged, the algorithm goes to the next adaptation time interval $[t+\Delta t_{ad}, t+2\Delta t_{ad}]$ and the computations restarted from the interpolated solution at $t+\Delta t_{ad}$ onto the current mesh.

The key element in transient-fixed-point mesh adaptation is to combine the errors of all or some of solutions in a given adaptation interval for mesh optimization in order to get a mesh good enough for all simulation time steps of this particular time span. In other words, using intersection metrics for approximating the error ensures that for all the solutions of the adaptation period the interpolation error at all edges remains bounded by a user-defined target error (constrained by maximum and minimum Euclidean edge lengths). Therefore, although the adapted mesh is not “optimized” for every given step, it is, as Alauzet *et al*. [4] puts it, “pseudo-optimized” for the computation along this time lapse.

### 3.1 Alauzet *et al.* metric intersection method [4]

In order to compute the intersection of two positive-definite matrices $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$, a basis $\left\{\vec{e}_i\right\}$ is defined such that these two matrices are congruent to diagonal matrices in it. This basis is assumed to be the set of normalized eigenvectors of the diagonizable matrix $\boldsymbol{N} = \boldsymbol{M}_1^{-1}\boldsymbol{M}_2$. Eigenvalues of $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ associated with this basis can be computed using the Rayleigh formula:

$$\lambda_i = \vec{e}_i^{\,T}\boldsymbol{M}_1\vec{e}_i \qquad\qquad \mu_i = \vec{e}_i^{\,T}\boldsymbol{M}_2\vec{e}_i \tag{3.3}$$

By assuming $\boldsymbol{P} = \left(\vec{e}_i\right)$ as the matrix whose columns are these eigenvectors (i.e. eigenvector matrix of $\boldsymbol{N}$), the intersected metric is defined to be:

$$\boldsymbol{M}_{1\cap 2} = \boldsymbol{M}_1 \cap \boldsymbol{M}_2 = \left(\boldsymbol{P}^{-1}\right)^T \boldsymbol{D}\boldsymbol{P}^{-1} \tag{3.4}$$

where $\boldsymbol{D}$ is the diagonal matrix:

$$\boldsymbol{D} = diag\left(\max\left(\lambda_i, \mu_i\right)\right)_{i=1..n} \tag{3.5}$$

and $n$ specifies the dimension of the matrix.

### 3.2 McKenzie *et al.*'s metric intersection scheme [38]

McKenzie et al. studied the representation of metric tensors, their intersection and union in quadratic form of tensors. The quadratic form of a metric tensor is an ellipsoid, i.e. the locus of points equidistant from the origin in computed based on that metric [12, 26, 39, 40]. Consider an $n \times n$ symmetric positive definite matrix $\boldsymbol{M}$ (called *metric tensor*). The metric's geometric representation can be defined by expressing it as a quadratic form:

$$f\left(\vec{x}\right) = \vec{x}^{T} \boldsymbol{M} \vec{x} = 1 \tag{3.6}$$

Equation (3.6) is the locus of points at a distance of one from the origin in the space defined by the metric $\boldsymbol{M}$. These points form an ellipse for a $2 \times 2$ and an ellipsoid for a $3 \times 3$ metric.

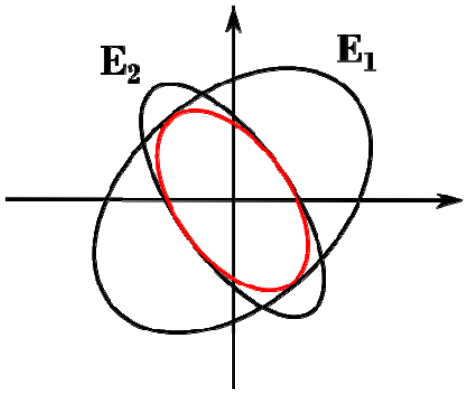



Figure 9: Metric intersection

Let $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ be two metric tensors (schematically drawn for a 2D example in Figure 9) whose ellipse representations are called $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$.

First, these two ellipses are transformed by a function $T$ so that ellipse $\boldsymbol{E}_1$ becomes a unit circle. McKenzie et al. define the transformation matrix as:

$$\boldsymbol{T} = \boldsymbol{\Lambda}_1^{-1} \boldsymbol{P}_1^{-1} \tag{3.7}$$

where $\boldsymbol{\Lambda}_1$ is the diagonal matrix of eigenvalues of metric $\boldsymbol{M}_1$ and $\boldsymbol{P}_1$ is the eigenvectors matrix. $\boldsymbol{T}$ transforms $\boldsymbol{M}_1$ into a unit circle of $\widehat{\boldsymbol{E}}_1$ and $\boldsymbol{E}_2$ into another ellipse $\widehat{\boldsymbol{E}}_2$. In order to find the metric form of the ellipse $\widehat{\boldsymbol{E}}_2$ a linear

system of equations should be solved. For simplicity, assume $\widehat{\boldsymbol{M}}_2$ is a $2\times 2$ metric. By choosing three points from the ellipse $\widehat{\boldsymbol{E}}_2$ we can write a linear system of three equations in terms of unknown arguments of $\widehat{\boldsymbol{M}}_2$ by substituting their coordinates in equation (3.6). The metric form of the unit ball is just the identity matrix. In this space, $\widehat{\boldsymbol{E}}_1$ is indifferent to directions since any two orthogonal directions can be chosen as its eigenvectors. Therefore, we can choose the eigenvectors of $\widehat{\boldsymbol{M}}_2$ as our basis in order to define the intersection matrix as:

$$\widehat{\boldsymbol{M}}_{1\cap 2} = \widehat{\boldsymbol{M}}_1 \cap \widehat{\boldsymbol{M}}_2 = \left(\widehat{\boldsymbol{P}}_2^{-1}\right)^T \left[\min\left(1, \hat{\lambda}_{2,i}\right)\right]_{i=1..n} \widehat{\boldsymbol{P}}_2^{-1} \tag{3.8}$$

where $\widehat{\boldsymbol{P}}_2$ is the eigenvector and $\hat{\lambda}_{2,i}$ are eigenvalues of $\widehat{\boldsymbol{M}}_2$. After computing the intersection metric $\widehat{\boldsymbol{M}}_{1\cap 2}$ the inverse transformation $\boldsymbol{T}^* = \boldsymbol{T}^{-1}$ is applied on the ellipse form of this matrix and again a linear system of equations is solved to find the metric $\boldsymbol{M}_{1\cap 2}$ in the original space.

This procedure is schematically shown in Figure 10 for a pair of 2D metric tensors. Since the intersection is found in a space where one of ellipses is actually a circle, projecting and truncating eigenvalues of the other metric makes sense. In other words, when one of the ellipses is a circle, the intersection is nothing but the biggest ellipse that is constrained to the circle and the ellipse. Since for the circle there's no directional preference, choosing the eigenvectors of $\widehat{\boldsymbol{M}}_2$ as the principal axes of the intersection is the perfect choice. Moreover, the intersected ellipse can only have eigenvalues between $1.0$ and those of $\widehat{\boldsymbol{M}}_2$. The choice of principal axes for projecting the two metrics and truncating eigenvalues is the important difference between this method and that of Alauzet. We will show later that Alauzet's method does not guarantee the orthogonality of chosen basis and therefore the corresponding eigenvalues computed for $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ cannot be directly compared for computing the intersection. Examples will be presented to illustrate this draw-back.

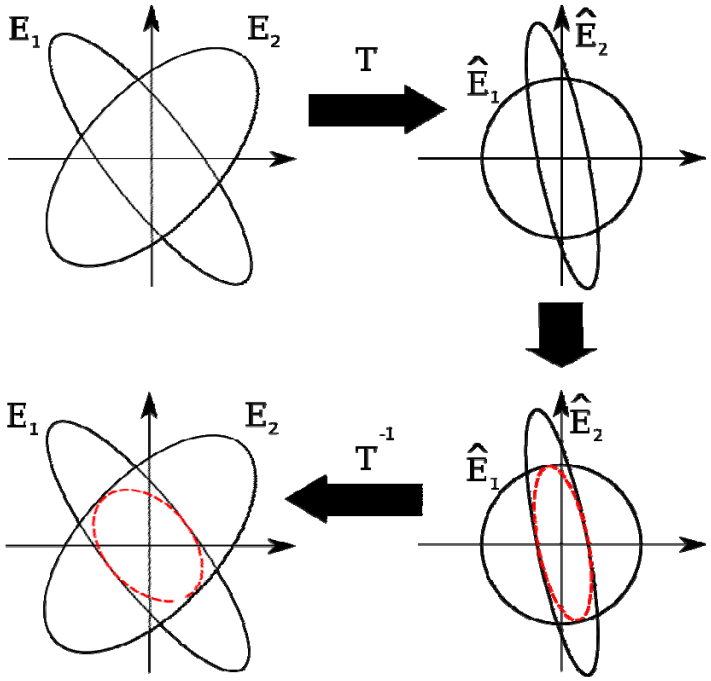



Figure 10: McKenzie *et al.*'s metric intersection algorithm

### 3.3 Modified McKenzie's intersection method [41]

McKenzie's proposed method shows promising characteristics and ensures orthogonality of directions along which the eigenvalues are truncated. However, the definition of the transformation for transforming the two ellipses $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$ is not consistent with the definition of error metric tensors. We will look at this issue in two dimensions for simplicity, but a very similar mathematical procedure can be used for higher dimensions. Let $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ be two $2\times 2$ metric tensors (symmetric positive definite) and assume that our purpose is to find a suitable transformation under which the ellipse $\boldsymbol{E}_1$ becomes a unit circle. The symbolic quadratic form of $\boldsymbol{E}_1$ can be written as:

$$f(x,y)=\begin{bmatrix}x & y\end{bmatrix}\boldsymbol{M}_1\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}x & y\end{bmatrix}\begin{bmatrix}a & b\\ b & c\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=ax^2+2bxy+cy^2=1 \qquad (3.9)$$

In order to show the relation between geometrical characteristics of ellipse $\boldsymbol{E}_1$ and eigenvalues and eigenvectors of $\boldsymbol{M}_1$, consider the special case of $\boldsymbol{E}_1$ being a horizontal ellipse, i.e. having its major axis in the *x*-direction and minor axis in the *y*-direction. This is equivalent to having $b=0$ in $\boldsymbol{M}_1$. Therefore, the quadratic form is simplified as:

$$f(x,y)=ax^2+cy^2=1 \qquad (3.10)$$

From equation (3.10), we can see that major and minor diagonals of the ellipse $\boldsymbol{E}_1$ are equal to $d_1=1/\sqrt{a}$ and $d_1=1/\sqrt{c}$. Moreover, eigenvalues of this special matrix are none other than $a$ and $c$. It implies that eigenvectors and inverse of square root of eigenvalues of the metric $\boldsymbol{M}_1$ define respectively the directions and size of major and minor diagonals of its ellipse representation. Now let's go back to the general form of $\boldsymbol{M}_1$ in equation (3.9) and assume that its eigenvalues are $\{\lambda_1,\lambda_2\}$ and its normalized eigenvectors are $\{\vec{e}_1,\vec{e}_2\}$. It can easily be deduced that the coordinate of points on the major and minor diagonals of the ellipse are:

$$\vec{p}_1=1/\sqrt{\lambda_1}\,\vec{e}_1 \qquad \vec{p}_2=1/\sqrt{\lambda_2}\,\vec{e}_2 \qquad (3.11)$$

So a suitable transformation should transform these two points to $\vec{q}_1 = (1,0)$ and $\vec{q}_2 = (0,1)$. We can write:

$$\boldsymbol{T}\vec{p}_1 = 1/\sqrt{\lambda_1}\,\boldsymbol{T}\vec{e}_1 \qquad\qquad \boldsymbol{T}\vec{p}_2 = 1/\sqrt{\lambda_2}\,\boldsymbol{T}\vec{e}_2 \tag{3.12}$$

Equation (3.12) can be rewritten in a single matrix form:

$$\boldsymbol{T}\left(\vec{p}_1, \vec{p}_2\right) = \begin{pmatrix} 1/\sqrt{\lambda_1} & 0 \\ 0 & 1/\sqrt{\lambda_2} \end{pmatrix} \boldsymbol{T}\left(\vec{e}_1, \vec{e}_2\right) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.13}$$

$$\overline{\boldsymbol{\Lambda}}\boldsymbol{T}\,\boldsymbol{P} = \boldsymbol{I} \tag{3.14}$$

where $\overline{\boldsymbol{\Lambda}} = 1/\sqrt{\boldsymbol{\Lambda}}$ and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues. So we can see that the correct definition of transformation matrix is:

$$\boldsymbol{T} = \overline{\boldsymbol{\Lambda}}^{-1}\boldsymbol{P}^{-1} \tag{3.15}$$

The difference between Eq. (3.5) and the original transformation (Eq. (3.7)) is in the definition of $\overline{\boldsymbol{\Lambda}}$ which is the inverse of square root of what is used in the original definition.

Moreover, since the geometrical diagonals are inversely proportional to the eigenvalues, and larger eigenvalues mean larger errors, the suitable form of truncation defined in equation (3.8)(mentioned in [38] that it might differ for error tensors from density tensors) is:

$$\widehat{\boldsymbol{M}}_{1\cap 2} = \left(\widehat{\boldsymbol{P}}_2^{-1}\right)^T \left[\max\left(1, \hat{\lambda}_{2,i}\right)\right]_{i=1..n} \widehat{\boldsymbol{P}}_2^{-1} \tag{3.16}$$

where $\widehat{\boldsymbol{P}}_2$ is the eigenvector matrix of $\widehat{\boldsymbol{M}}_2$ (the metric of transformed ellipse $\widehat{\boldsymbol{E}}_2$ and $\hat{\lambda}_{2,i}$ are eigenvalues of this metric. With these suitable definitions the modified McKenzie's scheme can be compared to Alauzet's method to study the effect of orthogonality-preserving in the former. Consider the following example for matrices that are defined to be symmetric and positive definite:

$$\boldsymbol{M}_1 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix} \qquad\qquad \boldsymbol{M}_2 = \begin{pmatrix} 3 & 0 \\ 0 & 0.5 \end{pmatrix} \tag{3.17}$$

Figure 11 shows the ellipse representation of these two metrics and the intersections computed using Alauzet's and the modified McKenzie's schemes. It can be clearly seen that Alauzet's method is not finding the perfect intersection.

Even analytically, there is no guarantee that the multiplication of two symmetric matrices results in a symmetric matrix.

For 2D it can be easily derived analytically:

$$\boldsymbol{N} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}\begin{pmatrix} d & e \\ e & f \end{pmatrix} = \begin{pmatrix} ad+be & ae+bf \\ bd+ce & be+cf \end{pmatrix} \tag{3.18}$$

Equation (3.18) shows that the product of two symmetric matrices is not necessarily symmetric unless $b(f-d) = e(c-a)$. It implies that such a scheme does not result in the true intersection of two matrices. In terms of error approximation it means that the computed values for errors along edges of the mesh are not accurate. We will show in following sections that its consequence is under-prediction of the error carpet in the computational domain.

However, the modified McKenzie's method proposed in this work transforms the two metrics so that one of them becomes indifferent to direction (Figure 10). Therefore the directionality of the intersected metric will come from the principle axes of the second transformed metric. Note that since the transformation and its inverse are nonlinear it does not mean that by transforming the metrics back the eigenvectors of intersected metric will be similar to that of the second metric. Figure 11 is a simple numerical example of this behaviour.

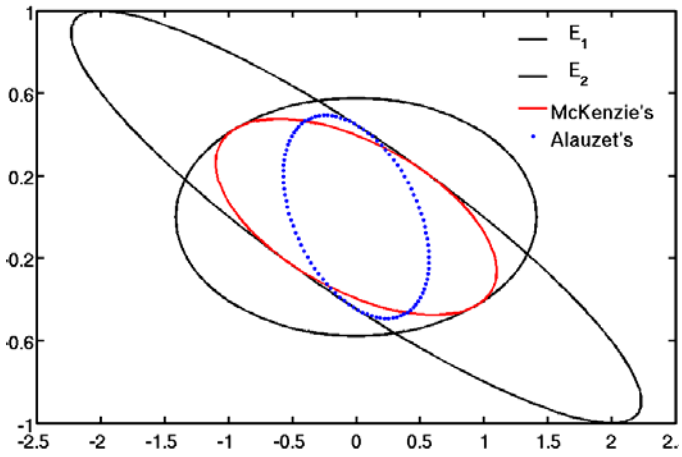



Figure 11: McKenzie's vs. Alauzet's intersected metrics

### 3.4 Unsteady Mesh Adaptation Benchmarks

A 2D mesh adaptation code (DADMOM[7]) is developed and foregoing methods are implemented in it to study their performance. For this purpose the first test case is a 2D blast problem in which simultaneous propagation of shock, contact discontinuity and expansion fan is expected. The second case is the classical problem of a 2D Mach 3 wind tunnel with a forward facing step [42, 43]. The last benchmark is a 0.3 Mach flow at 5000 Reynolds number passing over a four element airfoil taken from Omar *et al.* [44], whose leading edge slat is at $45^{\circ}$, the first trailing flap at $25^{\circ}$ and the second flap at $51.2^{\circ}$ with vortex shedding from leading edge slat and trailing edge flaps. In all these cases, the improved transient-fixed-point adaptation is used to study the impact of unsteady mesh adaptation on the solution. Moreover, carpets of approximated error using modified McKenzie's and Alauzet's schemes are computed at different iteration loops of mesh adaptation and presented side-by-side for comparison.

[7] Dynamic Anisotropic Data and Mesh Optimization Methods

### 3.4.1 2D Blast Problem

The first test case is the 2D blast problem. Computation domain is initialized with a uniform temperature and zero velocity distributions everywhere with high pressure and density jump in a small circular area at the center of computational domain with low values everywhere else. This test case is a simple extension of shock tube problem to two dimensions. This initial condition causes a shock wave and contact discontinuity propagates outwards while an expansion fan moves inwards. Figure 12 shows the initial condition over the originally created mesh. The flow solver FENSAP[45] is used for performing numerical simulations and is coupled with a 2D mesh adaptation code for coupling of solution and mesh. Table 1 shows the initial conditions for the problem. Computations are done for Euler equations under ideal gas assumption for the fluid. Mesh Adaptation parameters are listed in Table 2.

The total simulation time is $25.0\times10^{-5}\,s$ and is cut into five identical time periods of $5.0\times10^{-5}\,s$. This is the length of time step used for mesh adaptation, while the flow solver time step is chosen to be $1.0\times10^{-6}\,s$. This time step ensures that the CFL number in the smallest elements (for original and adapted meshes) is equal to or smaller than 0.8 to ensure numerical stability of the solver.

Therefore, for each adaptation time period 11 solutions saved at each time step of the simulation are combined.

Table 1: Initial conditions for blast problem

| | | | |
|---|---|---|---|
| High Pressure ($P_{high}$) | $100\,k$Pa | High Density ($\rho_{high}$) | $1.22\ kg/m^3$ |
| Low pressure ($P_{low}$) | $10\,k$Pa | Low Density ($\rho_{high}$) | $0.122\ kg/m^3$ |
| Radial position of initial pressure jump ($r_{jump}$) | 0.1 | Temperature ($T$) | $288\,K$ |

Table 2: Mesh adaptation parameters

| | | | |
|---|---|---|---|
| Target interpolation error | 0.015 | Maximum number of elements | 50,000 |
| Maximum Euclidean length | 0.5 | Maximum number of nodes | 50,000 |
| Minimum Euclidean length | $1.0\times10^{-8}$ | No. solutions combined | 10 |

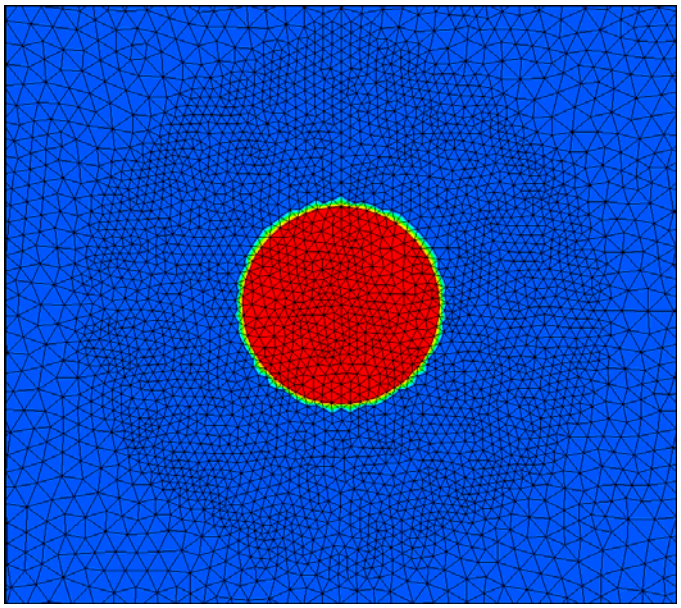


Figure 12: Original grid and density contours of pressure for initial condition

Then the mesh is adapted for the first time interval (i.e. $t \in \left[0, 1.0\times10^{-5}\right]$) and the error computed using the intersected metrics over time for Mach number (adaptation variable) using the modified McKenzie's as well as Alauzet's schemes over this initial mesh (Figure 13). It can be seen that Alauzet's error carpet is slightly lower than McKenzie's. It gets even worse when the second loop of computations and mesh adaptation is performed (Figure 14).

Because of well-posed behaviour of the problem (i.e. propagation of shock and expansion waves, and contact discontinuity) one could only compare the solutions at the last time step. Moreover, one would like to see the if intermediate interpolations from one mesh into another one would stack up to any significant value. Therefore, for this test case we have chosen to compare the solutions in the last time step (i.e. $5.0\times10^{-5}\,s$). Figure 15 and Figure 16 show the mesh, density contours and density plots for this insstant computed on the original, 1$^{st}$ and 2$^{nd}$ adapted meshes using the new transient-fixed-point method.

??? shows the summary of number of nodes and elements in the three meshes used. Note that in mesh adaptation it is not easy to maintain the exact same degrees of freedom in the domain. However, adaptation constraints (minimum edge size, maximum number of nodes & elements) are chosen to keep the size of the grids almost constant. Number of nodes and elements reported in ??? for adapted meshes are average values of meshes of the five adaptation intervals with negligible differences (i.e. maximum difference of 50 nodes and 120 elements).
After the 2nd mesh adaptation and 3rd flow solution the maximum change in solution on the modified McKenzie's adapted meshes is about 1.5% in comparison to a value of 14% for meshes adapted with Alauzet's scheme.

Table 3: Sizes of three meshes used for numerical simulations

| | Original Mesh | 1st adapted mesh | 2nd adapted mesh |
|---|---|---|---|
| Numer of nodes | 42,000 | 50,000 | 50,000 |
| Number of elements | 128,000 | 176,531 | 176,060 |

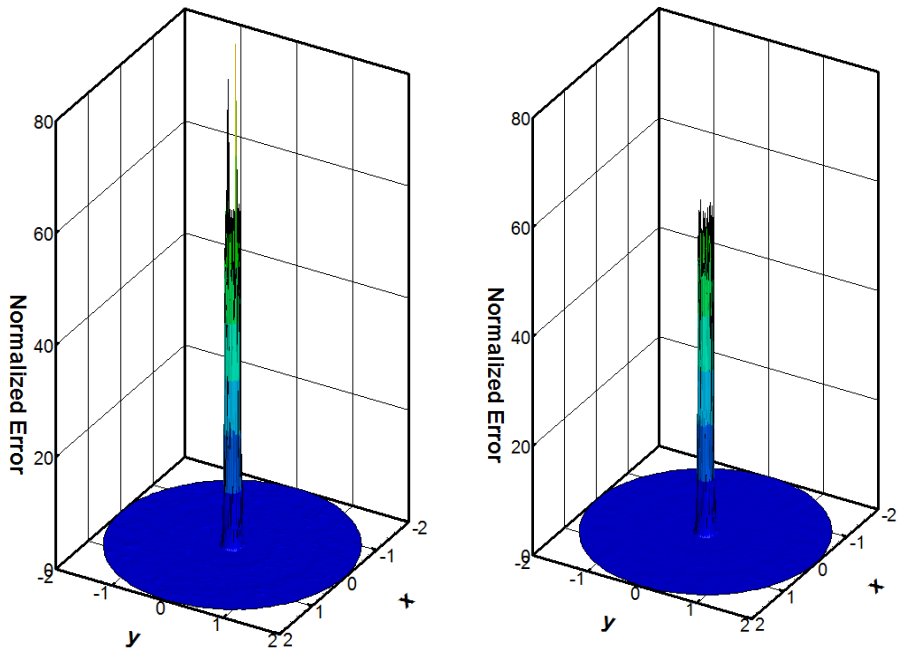



Figure 13: Normalized error carpets for $t \in \left[0, 1.0 \times 10^{-5}\right]$ using modified McKenzie's (left) and Alauzet's schemes (right) on the original mesh

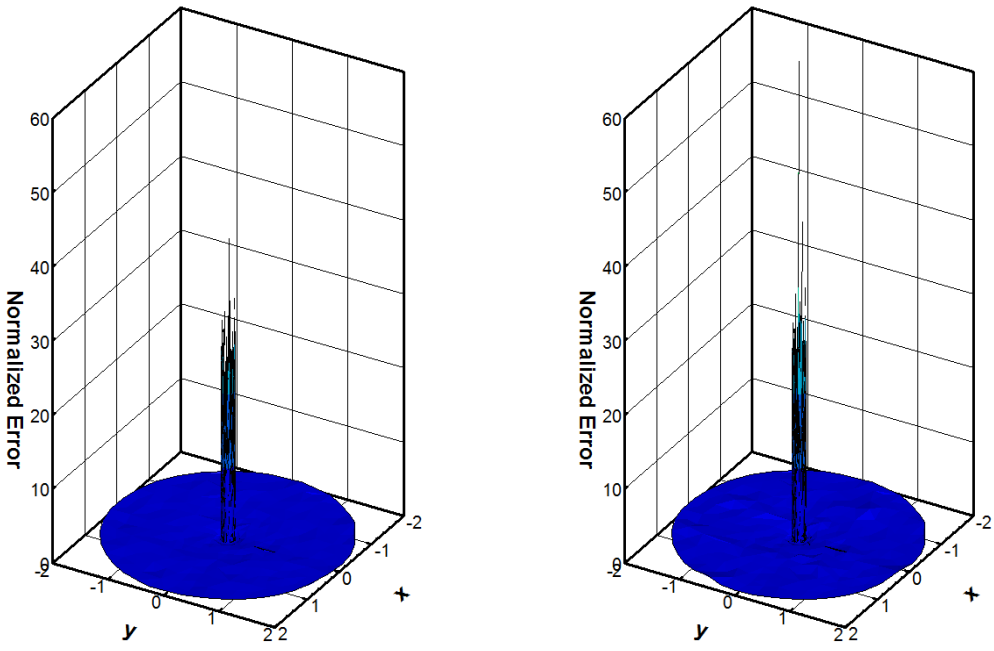



Figure 14: Normalized error carpets for $t \in \left[0, 1.0 \times 10^{-5}\right]$ using modified McKenzie's (left) and Alauzet's schemes (right) on the 1st adapted mesh

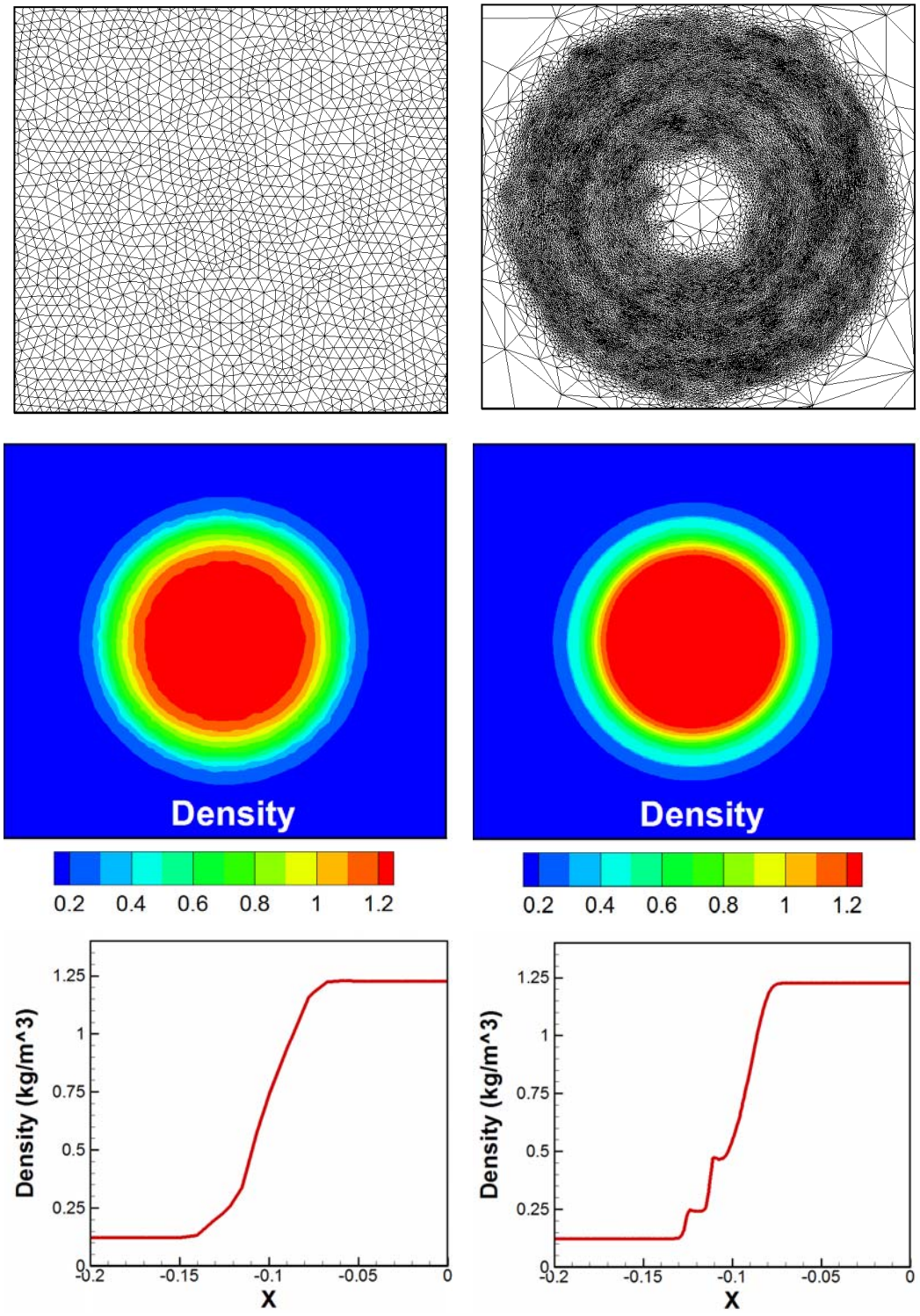



Figure 15: Mesh (top row), density contours (middle row) and density plots (bottom row) for original mesh (left) and 1st adapted mesh (right) in the last time step of simulation

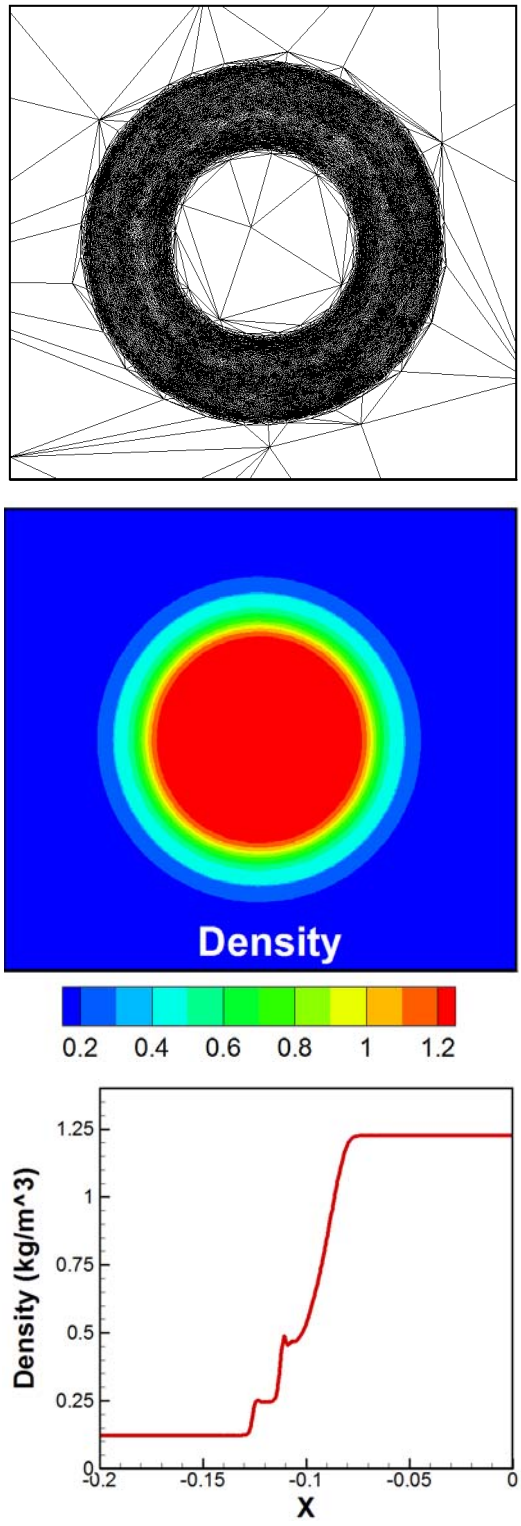



Figure 16: Mesh (top row), density contours (middle row) and density plots (bottom row) for 2nd adapted mesh in the last time step of simulation

### 3.4.2 Mach 3 wind tunnel over 2D forward facing step [43]

This test case was introduced by Emery [46] and accepted as a benchmark for unsteady flow simulations after a paper by Woodward and Colella [43]. The geometry of the tunnel, which is filled with an inviscid gas, is shown in Figure 17. The gas initially has a density 1.4 , pressure 1.0 and Mach number 3.0 everywhere. The inlet boundary condition is exactly the same as the initial condition. The initial triangular mesh is generated with maximum and minimum edge sizes of 0.05 and 0.01 respectively, resulting in 19556 elements and 1001 nodes (Figure 18).

The flow reaches its steady state at around $t = 12$ . However, similar to the original work [43] the flow is studied only until $t = 4$. Figure 19 and Figure 20 demonstrate Mach contours at different time instances over the initial mesh. A shock wave grows in front of the step and first reflects from top and then from bottom walls. Moreover, an expansion fan starts from the step corner towards downstream hitting shock reflections.

It can be seen from these solutions, especially from the last one that the relative large size of elements has the following negative effects:

1- Smearing shocks (i.e. a thickening discontinuity)
2- Very weak expansion fan (almost not visible)
3- Corrugated shocks (due to lack of nodes)

Solutions for this simulation were stored at time intervals of $t = 0.05$: 81 solutions in total, including the initial solution.

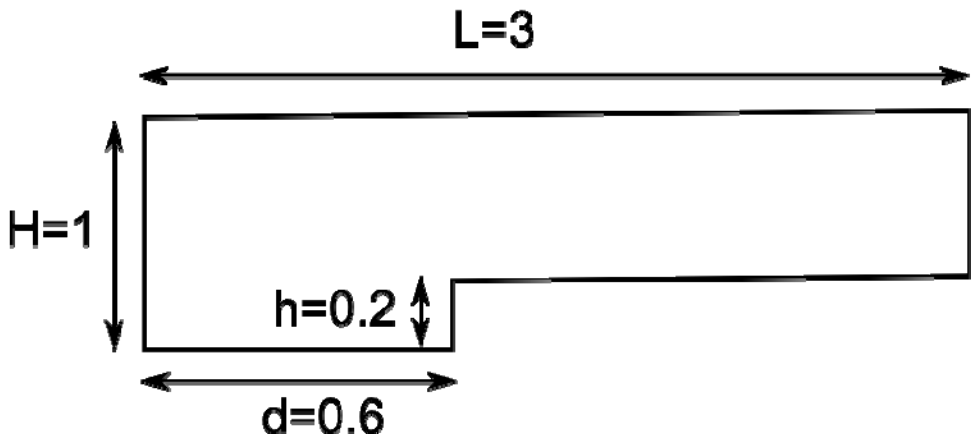


Figure 17: Geometry of Tunnel

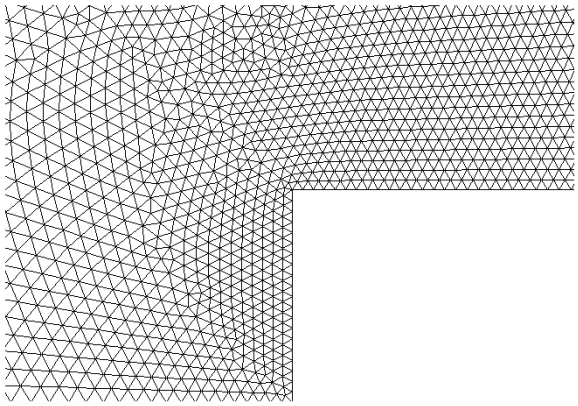


Figure 18: Initial Delaunay mesh (zoom at the corner of step)

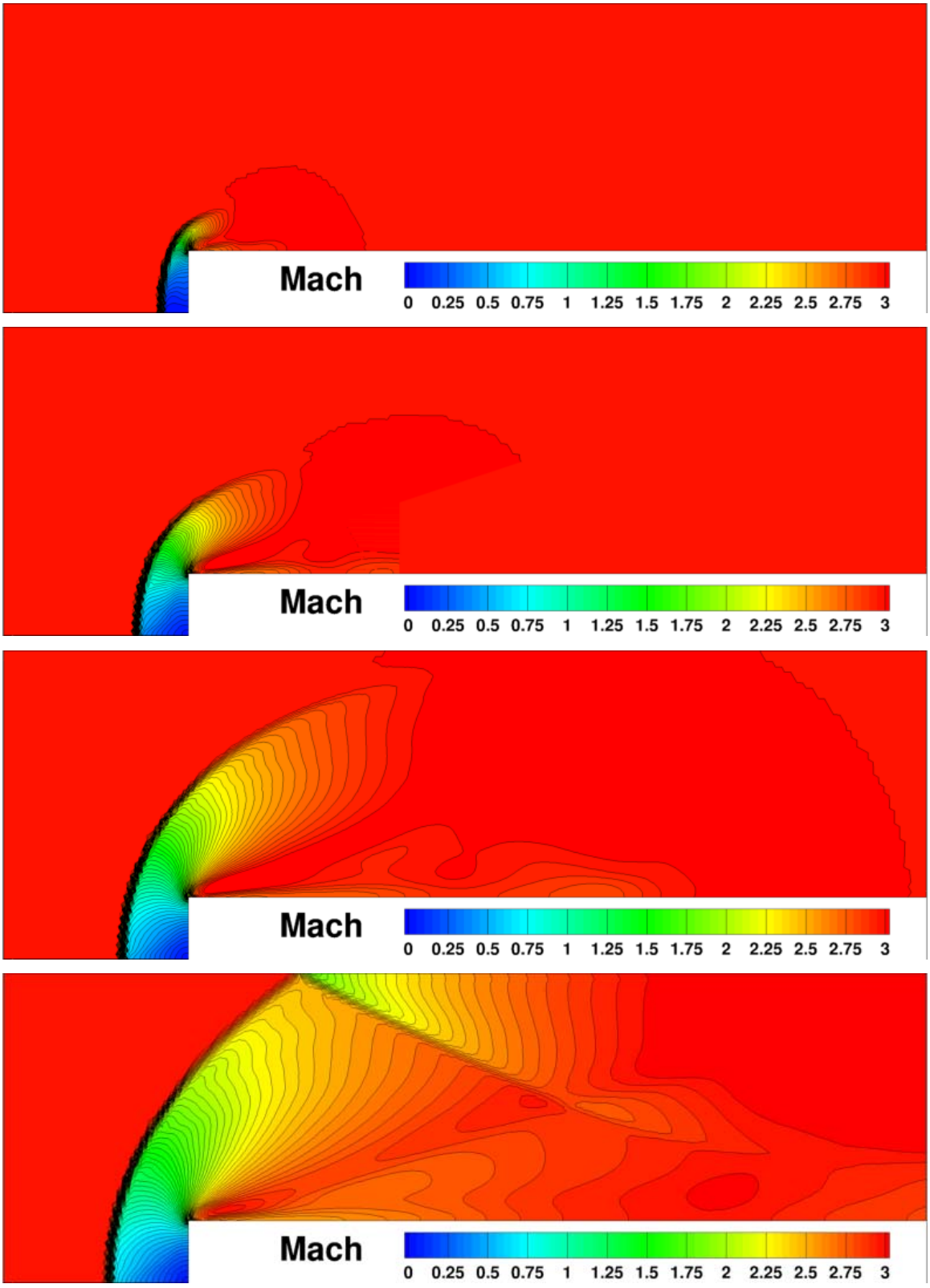



Figure 19: Solutions at $t = 0.1, 0.25, 0.5, 1.0$ over initial mesh.

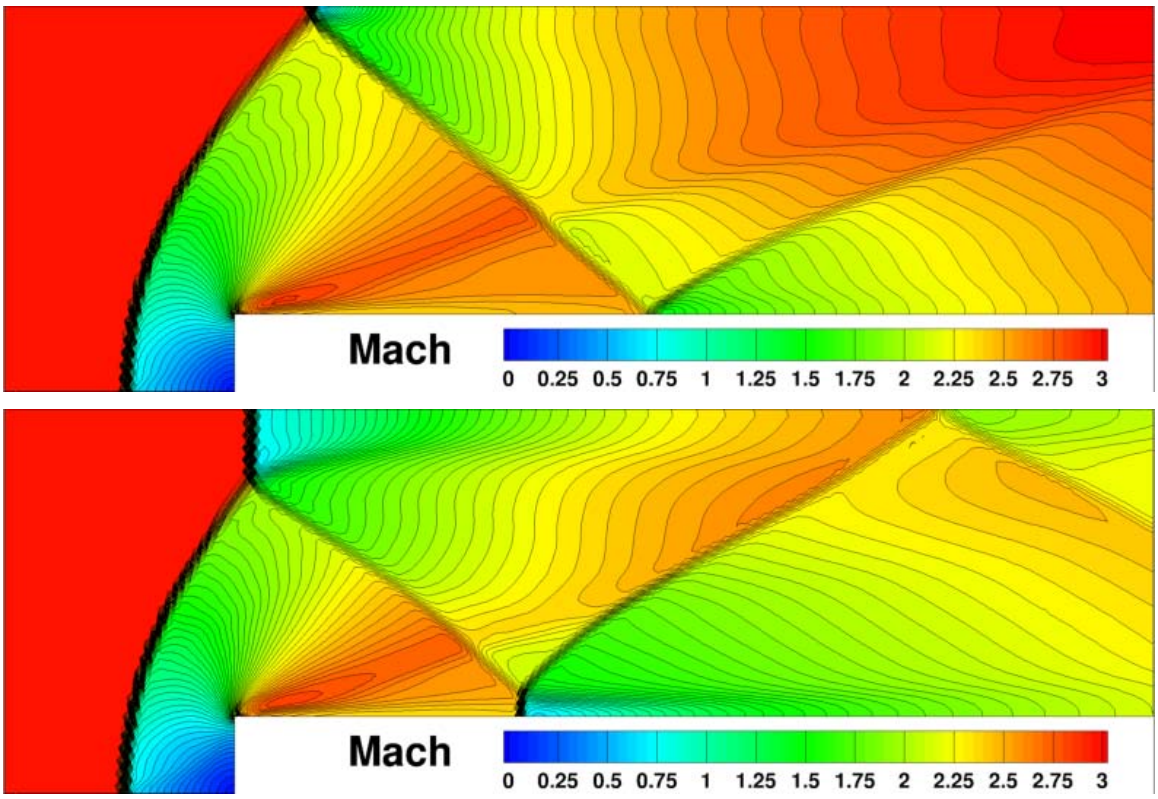



Figure 20: Solutions at $t = 2.0, 4.0$ over initial mesh.

For unsteady mesh adaptation, the simulation time is broken into 10 intervals. Each interval contains 9 solutions, with starting and end solutions shared with adjacent time intervals. Mach number is chosen as the scalar for error estimation and mesh adaptation. Each interval results in one adapted mesh. For the second cycle of flow simulation, the computations starts from time zero on the adapted mesh for the first interval up to $t = 0.4$. Then, the solution at that time is interpolated onto the mesh for the second interval on which the flow computation is continued. This procedure is repeated all through the 2nd simulation cycle (Figure 21-23).

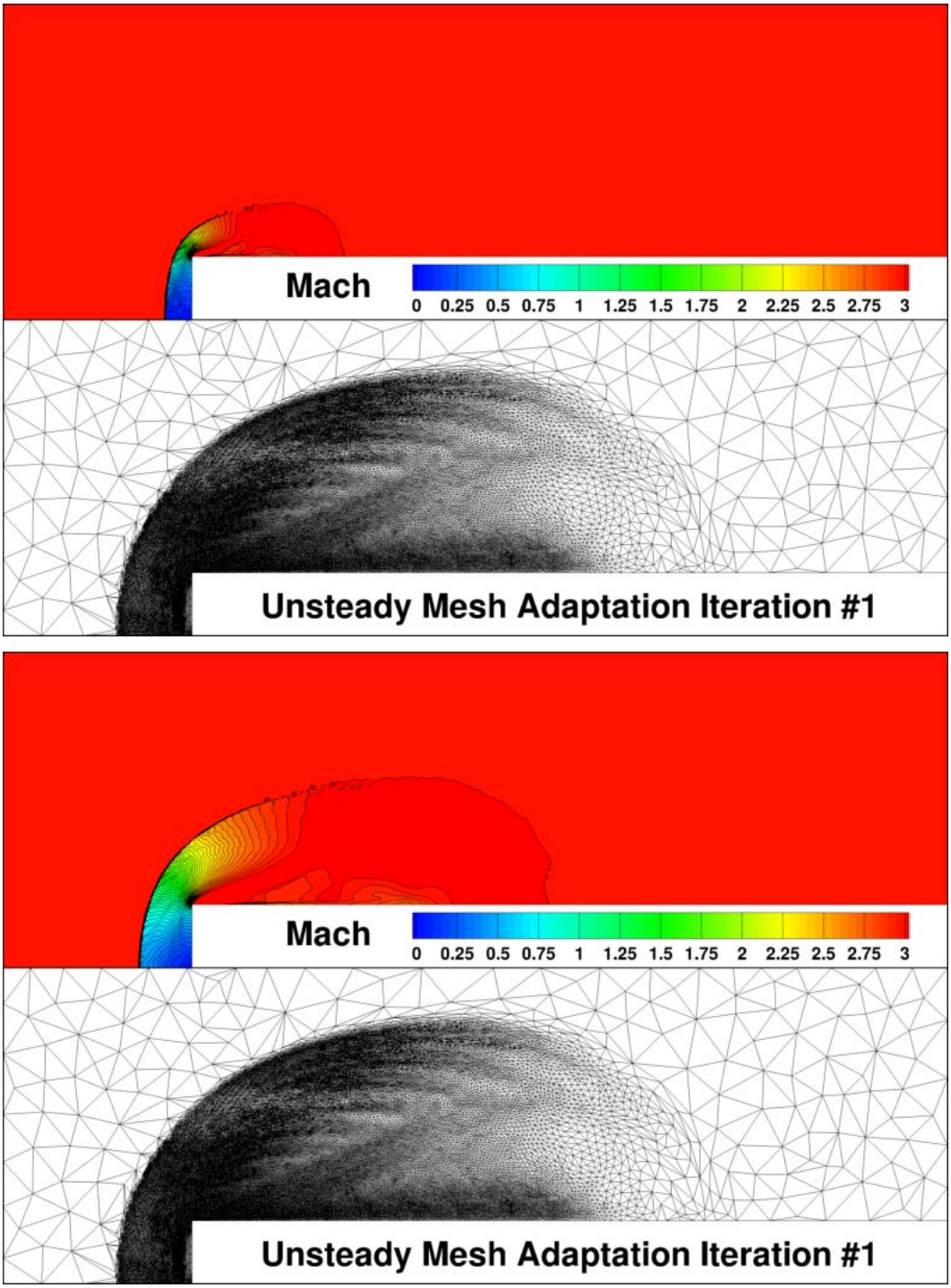



Figure 21: Adapted meshes and solutions for Mach 3.0 flow over forward facing step at $t = 0.1, 0.25$

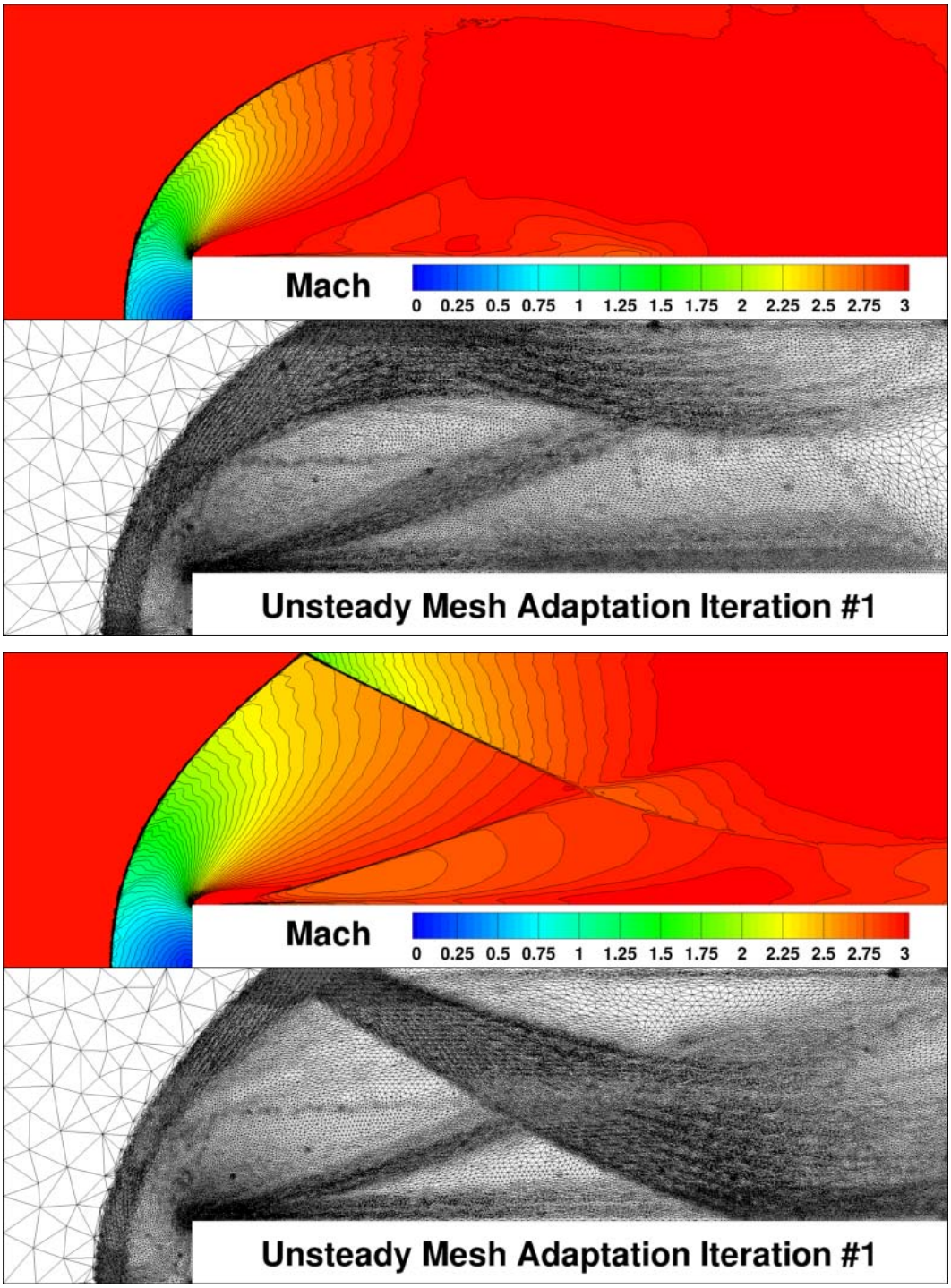



Figure 22: Adapted meshes and solutions for Mach 3.0 flow over forward facing step at $t = 0.5, 1.0$

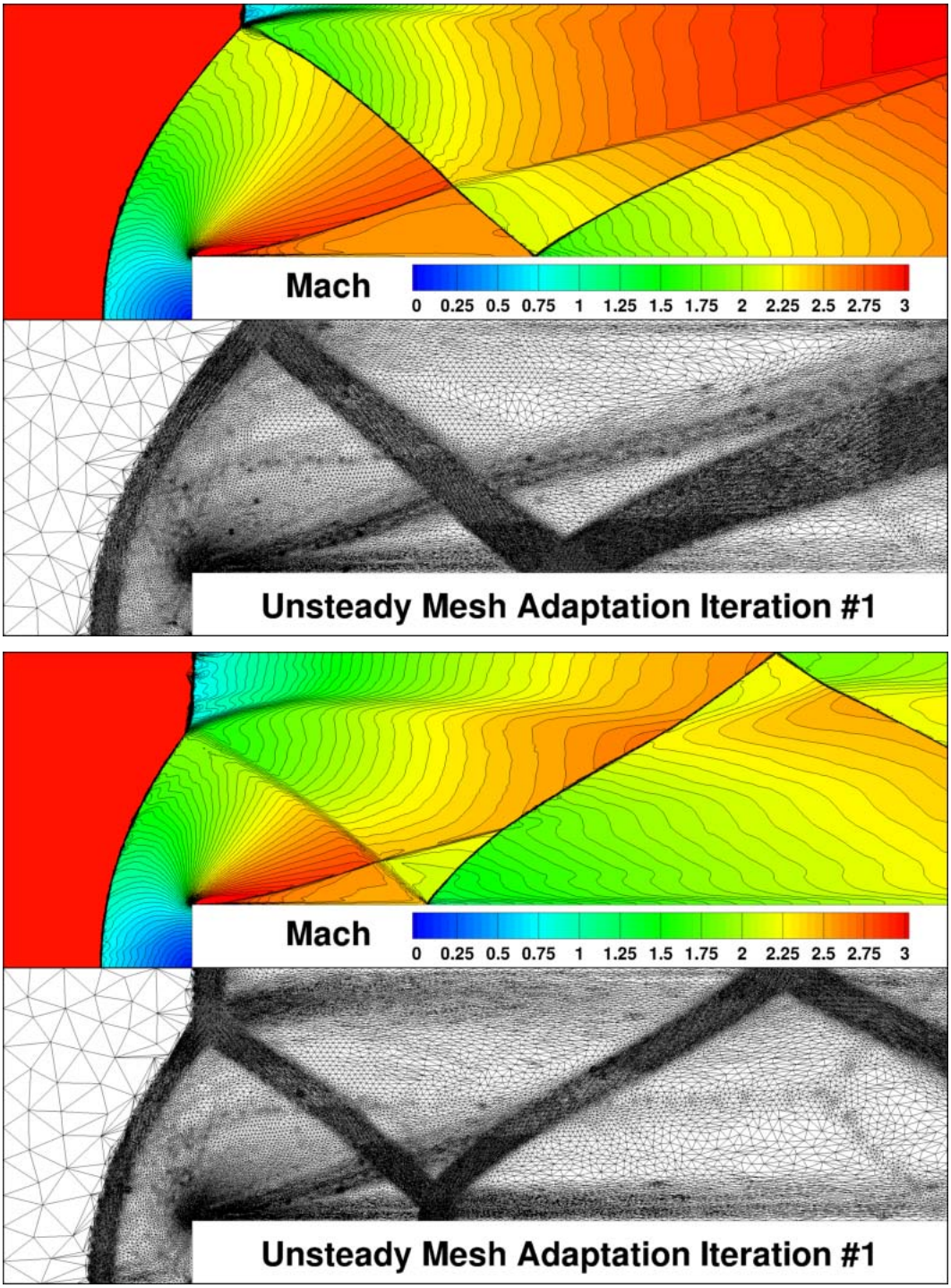



Figure 23: Adapted meshes and solutions for Mach 3.0 flow over forward facing step at $t = 2.0, 4.0$

The first thing to notice is that the mesh is fixed for one time interval (Figure 21Figure 23) and the evolution of grid from one time interval to another one. Each mesh is adapted to be suitable enough for all time steps of simulation for its

corresponding interval. The effect of mesh adaptation is very clear. Shocks have become very sharp and thin, the expansion fan is clearly visible and even the slip line which originates from the triple point of three shocks at the top is captured as well. This slip line e is created due to Kelvin-Helmholtz instability at the intersection of the main strong shock and upper Mach disk.

### 3.4.3 Two-dimensional viscous flow over multi-element airfoil

The last test case is viscous flow over a four-element airfoil [44], with leading edge slat at $45^{\circ}$, first trailing edge flap at $25^{\circ}$ and the second one at $51.2^{\circ}$. Table 4 shows all parameters set for flow simulation. The computational domain is 15 times the characteristic length ($L = 0.4\,m$) in every direction. Initially, the flow is set to be uniform everywhere. It is as if the airfoil is suddenly placed inside the airflow with the abovementioned conditions. An unsteady simulation shows a vortex shed from the slat's leading edge and that it is washed downstream. Moreover, behaving as a bluff body for the incoming flow, the flaps initiate vortex sheddings as well. Therefore, one can expect that the flow remains unsteady with (semi-) periodic vortex shedding from the trailing edge.

Table 4: Simulation conditions for the multi-element airfoil

| | | | |
|---|---|---|---|
| $Mach$ | 0.33 | $P_{\infty}$ | 101.325 $k$Pa |
| Re | 5000 | $T_{\infty}$ | $200\,K$ |
| $\alpha\left(AoA\right)$ | $0^{\circ}$ | $L$ | $0.4\,m$ |
| $t_{simulation}$ | $5.0\times10^{-5}\,s$ | $\Delta t$ | $1.0\times10^{-7}\,s$ |

This simulation is initially performed on an initial mesh with not many nodes in the wake (Figure 24). Figures 25 and 26 are snapshots of vorticity contours at different time instances. Due to low mesh density in the wake, the wake is not captured and after a while the flow seems to become steady with no vortex shedding. Vorticity magnitude is chosen as the scalar for error estimation and mesh adaptation for capturing boundary layer and vortices in the wake. One might try Mach number or velocity magnitudes, which are good choices for boundary layer, but the variations in these two variables are not strong enough in the wake to result in error estimates comparable to the boundary layer.

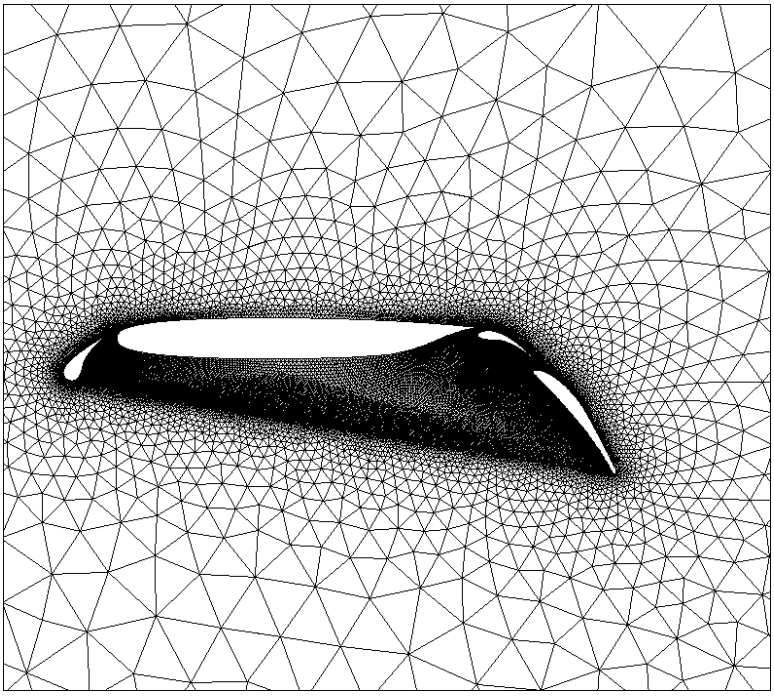
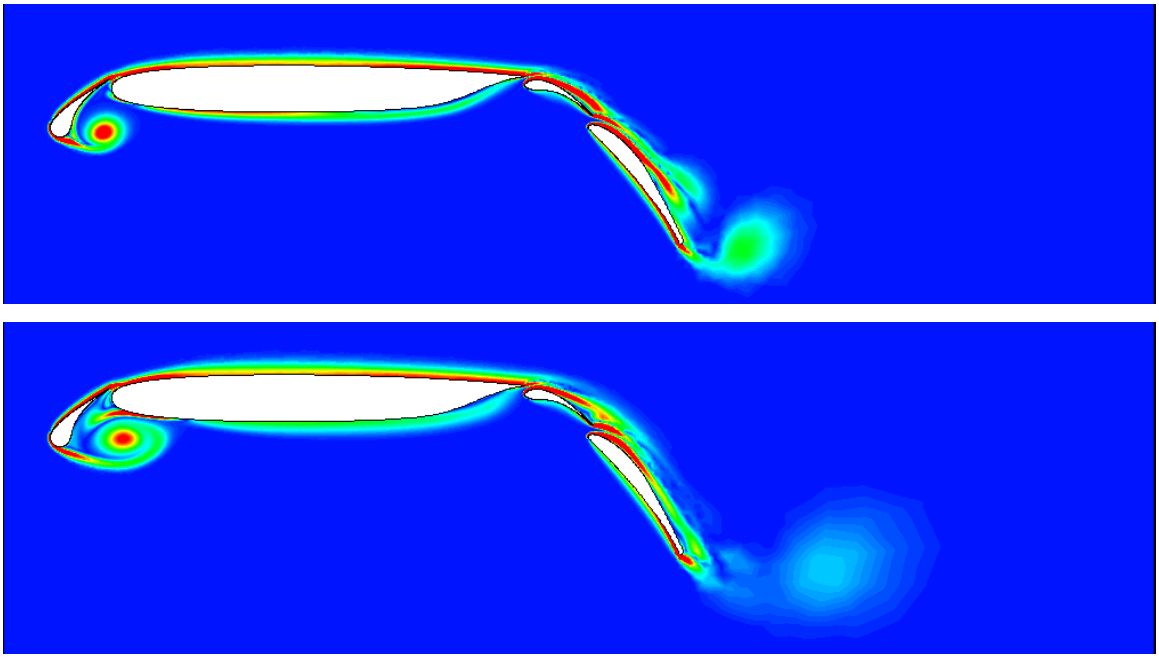


Figure 24: Initial grid generated around a multi-element airfoil



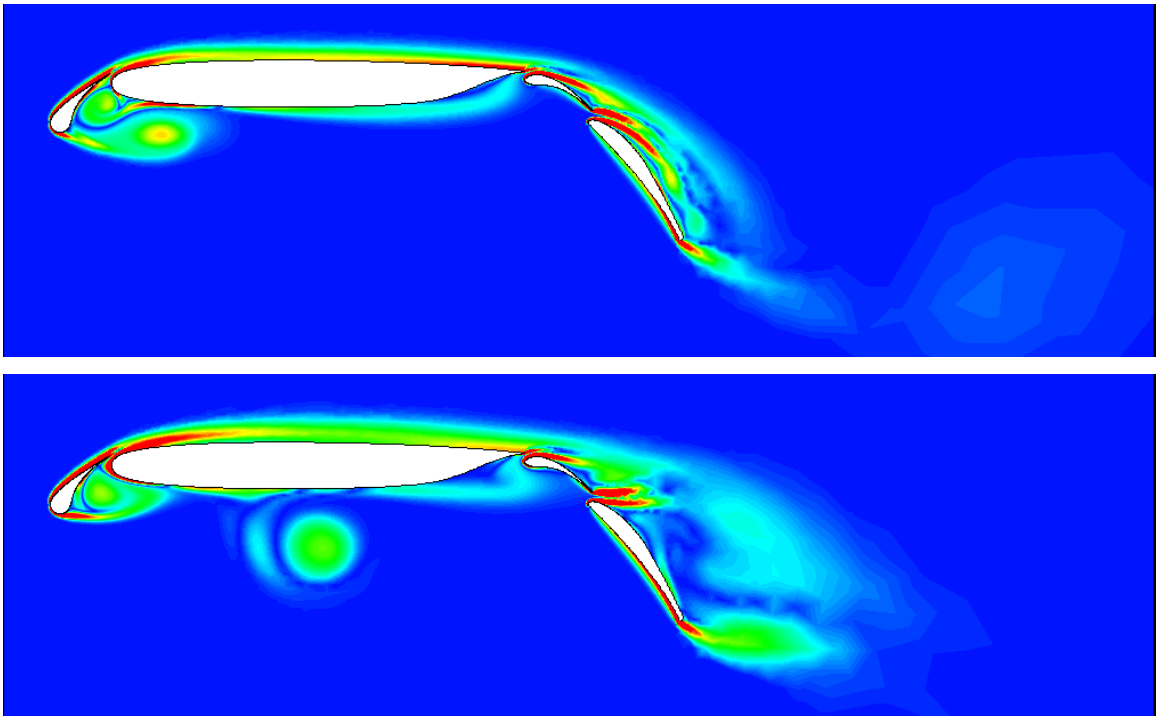
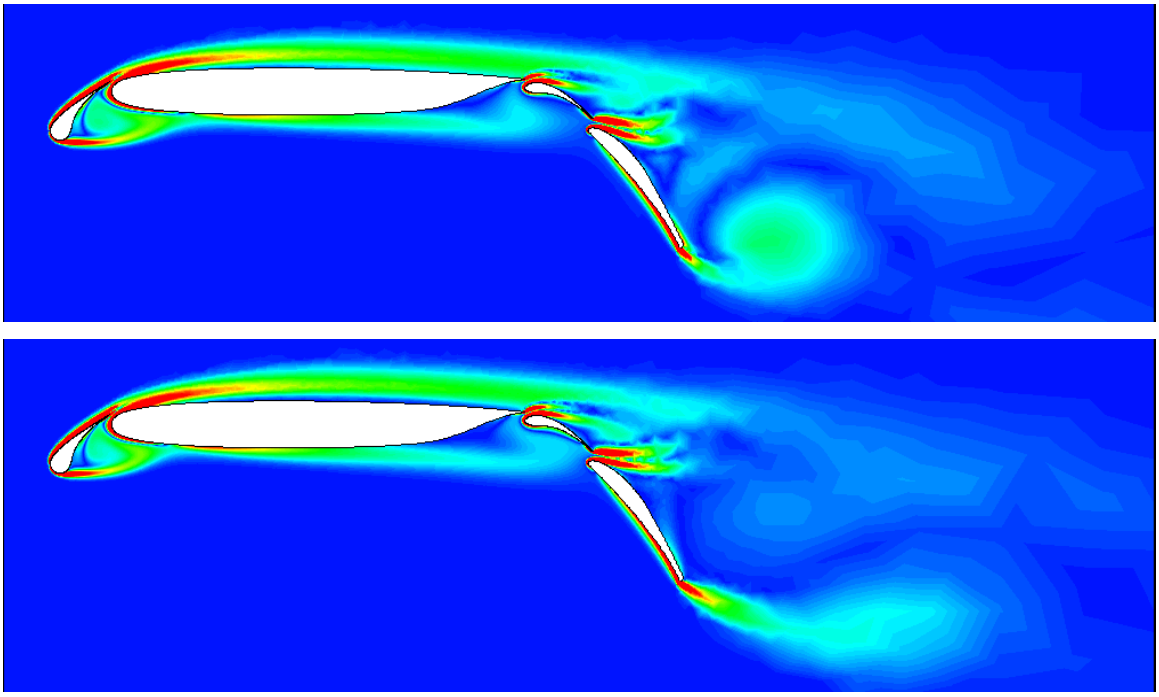


Figure 25: Vorticity contours for multi-element airfoil at time instances of

$$t = 1.0\times10^{-6}, 2.0\times10^{-6}, 4.0\times10^{-6}, 8.0\times10^{-6}$$



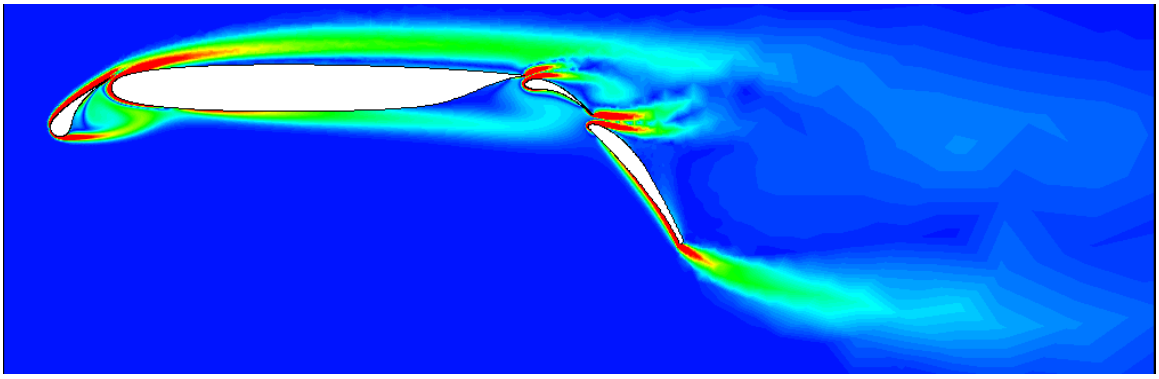


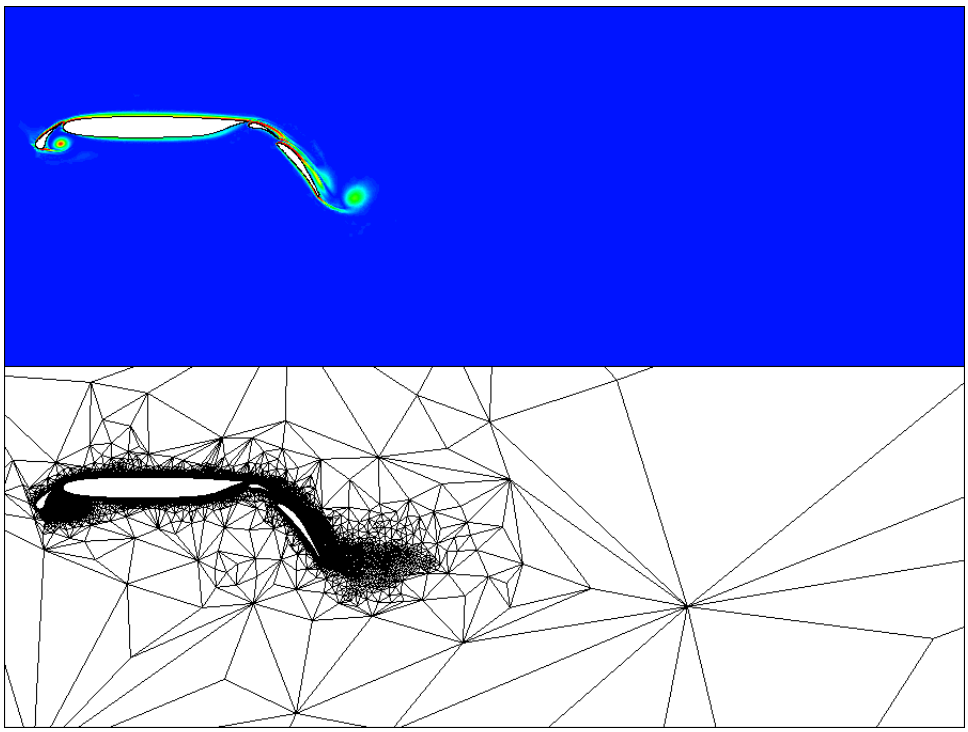
Figure 26: Vorticity contours for multi-element airfoil at time instances of $t = 1.6\times10^{-6}, 3.2\times10^{-5}, 5.0\times10^{-5}$

As the result, the mesh will be mostly adapted in the boundary layer and not the wake. A better choice is the magnitude of vorticity, which is well preserved in the wake for a long distance after the airfoil. Therefore, the vortices will be tracked and captured by error estimation and eventually mesh adaptation.

Moreover, the total simulation time is split into 25 intervals with 21 solutions for each interval. Note that the first and last solutions of each interval are, respectively shared with previous and next time intervals.

Figures 27-30 are the set of numerical results obtained over the adapted meshes after 2 cycles of mesh adaptation and flow simulation (in addition to the initial results obtained).

The first thing to notice is that unsteady mesh adaptation has helped the solver to simulate the true unsteady vortex shedding in the wake. Secondly, the mesh changes from one time interval to another one while staying fixed during any interval.



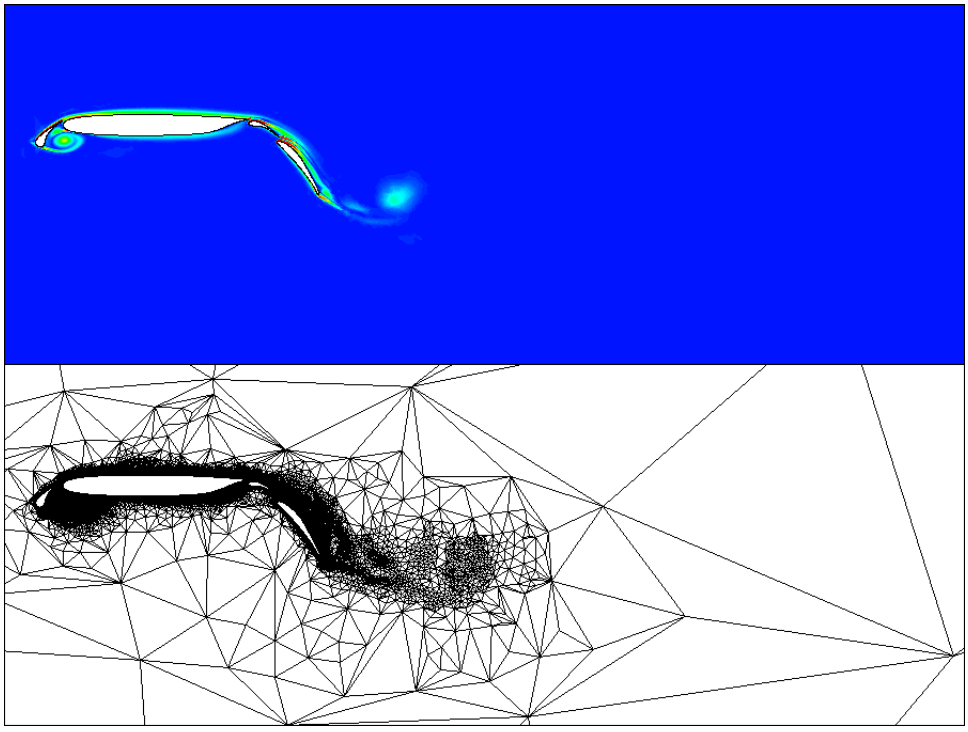
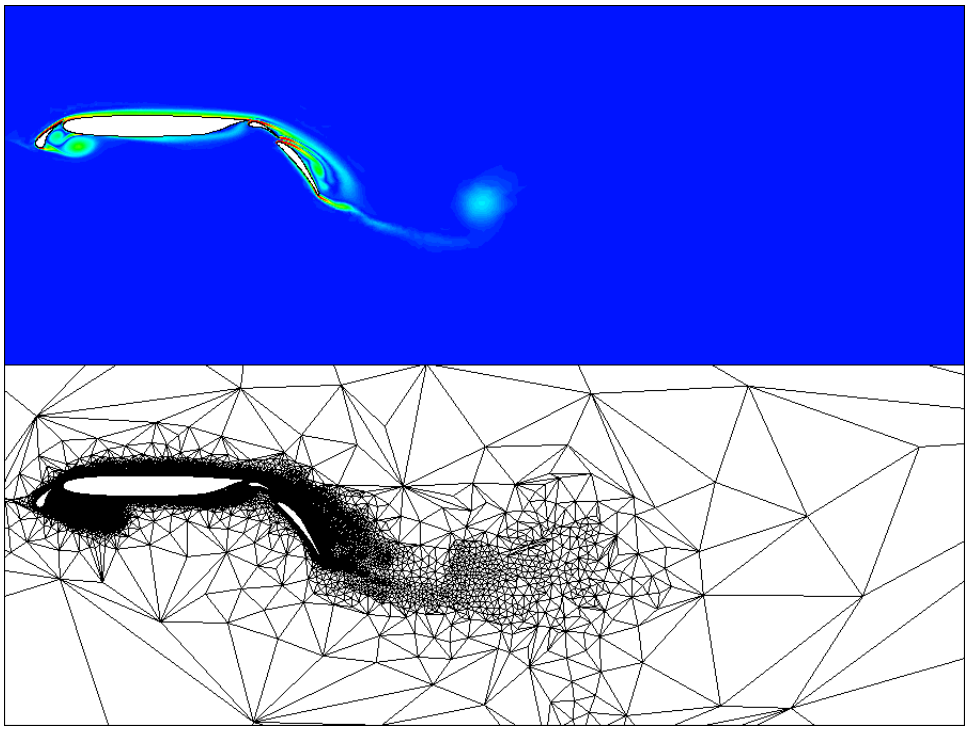


Figure 27: Adapted meshes and solutions for

multi-element airfoil at $t = 1.0\times10^{-6}, 2.0\times10^{-6}$



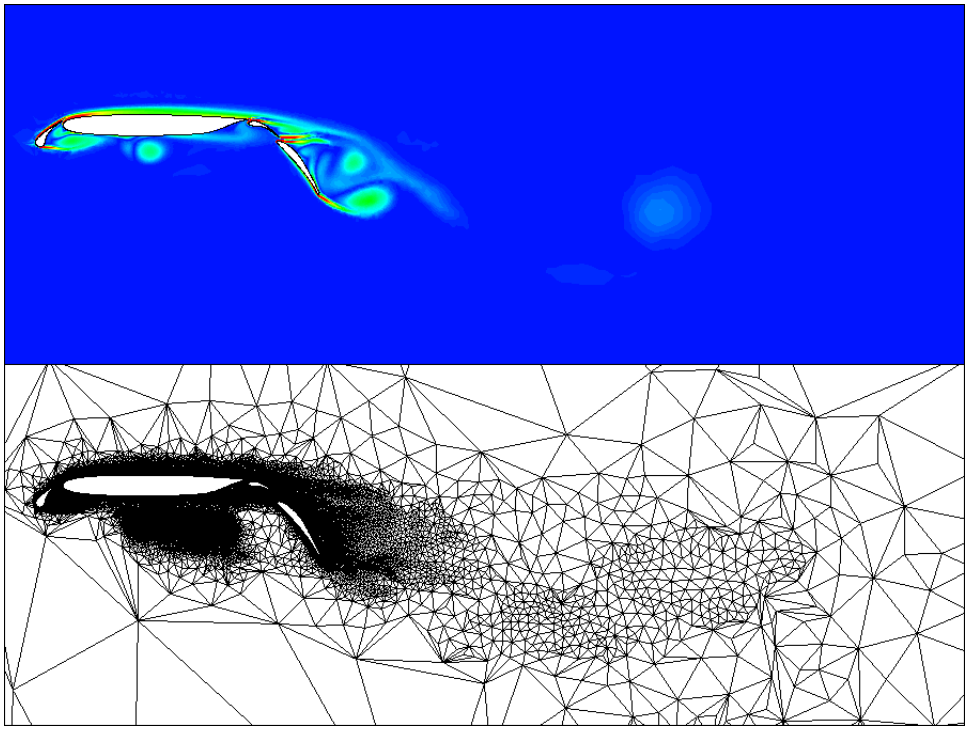
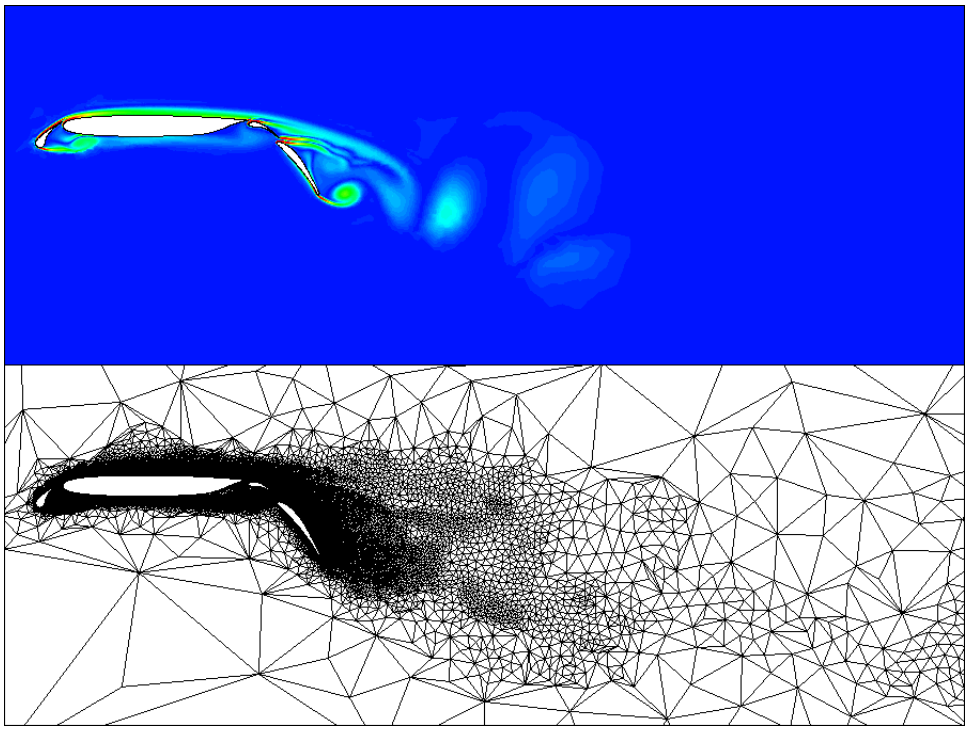


Figure 28: Adapted meshes and solutions for multi-element airfoil at $t = 4.0\times10^{-6}, 8.0\times10^{-6}$



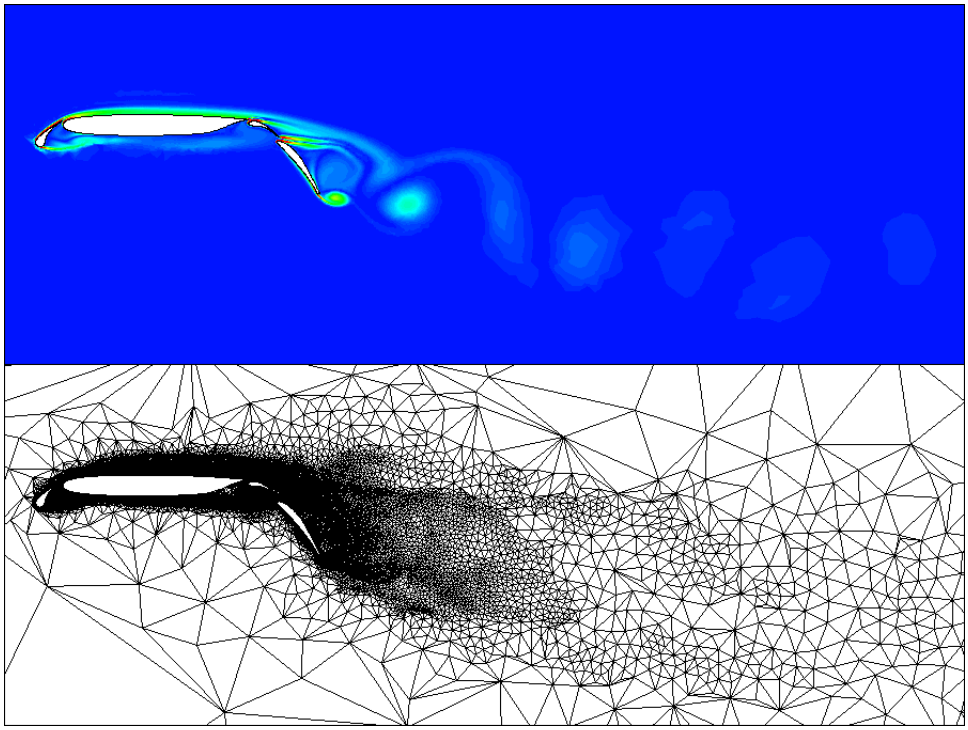


Figure 29: Adapted meshes and solutions for

multi-element airfoil at $t = 1.6\times10^{-6}, 3.2\times10^{-5}$

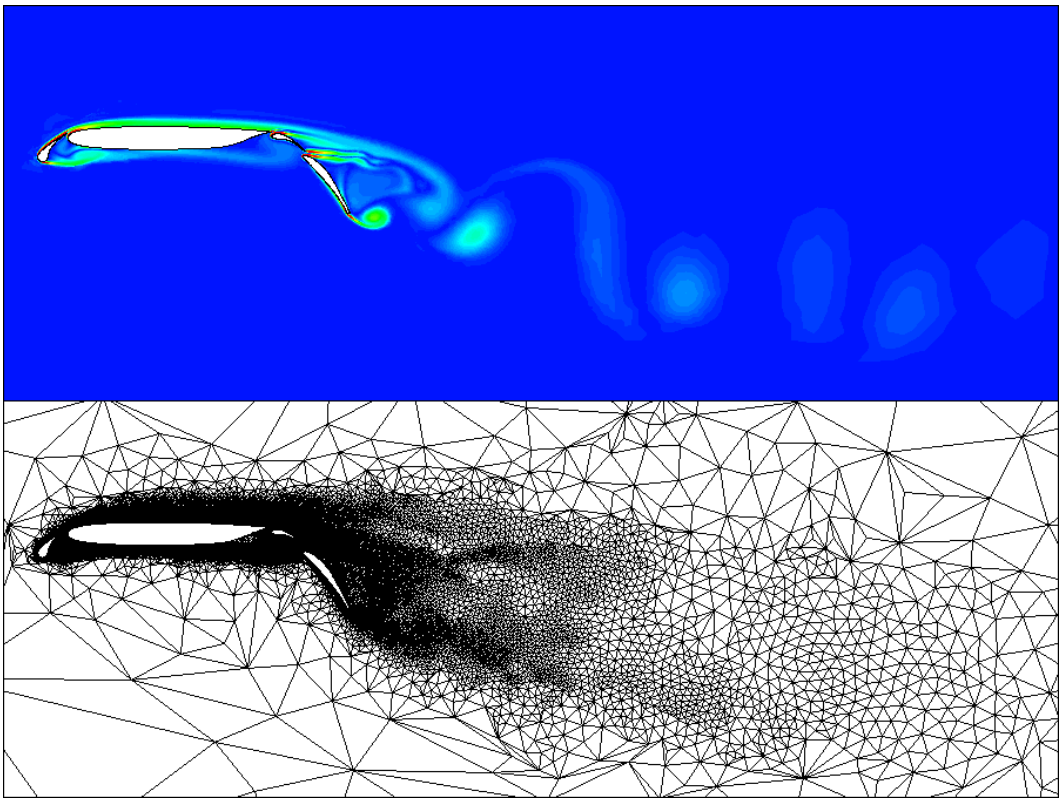


Figure 30: Adapted meshes and solutions for
multi-element airfoil at $t = 5.0 \times 10^{-5}$

### 3.5 Conclusion

In this chapter a new metric intersection method for unsteady mesh adaptation was proposed and implemented in the transient-fixed-point algorithm introduced by Alauzet *et al.* [32, 47]. In this new method, metric intersection is performed using the ellipsoidal representation of metrics by first transforming them into a space in which one of the metrics becomes indifferent to any direction (i.e. transforming its ellipse into a circle). The intersection is computed as the largest ellipse that can be fit inside the intersection zone. In this approach the intersection is performed to keep orthogonality of principal directions.

Using the proposed methods three 2D cases were simulated (blast wave problem, Mach 3.0 flow over forward facing step, and viscous flow over a four-element airfoil) using unsteady mesh adaptation. For the blast problem the error carpet was plotted for this method and compared to the Alauzet *et al.*'s [32] showing that the former computes error carpets more sharply. We believe that it is because our intersection method preserves the directionality of solution by computing the best principle directions for intersection metric.

### 3.6 Future Work

For unsteady mesh adaptation, one of the future works can be testing 3D unsteady benchmarks. Such benchmarks can be used to study the effect of unsteady mesh adaptation for predicting time-dependent flow characteristics, for example frequency of vortex shedding, moving position of separation point and amplitude of fluctuating lift and drag forces over surfaces with vortex shedding in their wake. One of such examples is the unsteady simulation of wing-tip vortex shedding, its effect on lift and drag and the vortex interaction with the tail.

Another topic as a future work for unsteady mesh adaptation is to use it as a tool for practical mesh independence study for transient flow problems. The goal is to reduce computational costs of intermediate cycles of adaptation-solver couple in

unsteady mesh adaptation. This topic is an ongoing project in CFD Lab at McGill University in which a Reduced Order Modeling (ROM) technique is coupled with unsteady simulation for enriching transient solutions used for unsteady mesh adaptation [48]. In this approach, for intermediate cycles higher time steps can be used for flow simulation with solutions at time steps that were skipped approximated via ROM.

# Chapter 4 Functional-Output Mesh Adaptation

Recently, a new approach for error estimation is studied that is more relevant for engineering applications and in which error made in prediction of integral quantities representing an engineering output is assessed. Examples of such engineering outputs include lift and drag forces, total heat flux, acoustics noise level, or total rate of ice formation on an aircraft wing during adverse atmospheric conditions. These integral outputs are known in the error estimation literature as functional-outputs. There have been significant volumes of research into *a posteriori* error estimation within the context of finite element methods for fluid dynamics using adjoint solutions. Becker, Rannacher and collaborators have developed an optimal control approach for output-based isotropic grid adaptation within a Galerkin finite element framework [49-51]. Other works by Patera, Peraire and their collaborators [52, 53], Suli and his colleagues [54, 55], Larson and Barth [56], Formaggia *et al.* [57] worked on implicit *a posteriori* error approximations for finite element methods. Most recent works which utilize adjoint solutions for functional-output adaptation belong to Venditti, Fidkowski and Darmofal [23, 58-60], Loseille, Dervieux and Alauzet [5], and Park and Carlson [61]. In these methods, the error indicator is related to the local residual error of the primary solution. The local residual error is computed using the linear-adjoint solution along with the primary solution. Therefore, these methods are computationally expensive. Moreover, the discretized adjoint equations are dependent on the numerical scheme implemented in the flow solver, and the technology is therefore not portable between codes.

An alternative approach is proposed by Remaki and Habashi [62] as Hermite-based mesh adaptation. They believe that in anisotropic adapted meshes highly stretched and large elements connected to smaller ones lead to loss of accuracy for functional outputs. Therefore, they propose an error indicator based on Hermitian interpolation used to control the elements size by computing integrals in a smoothly varying way from regions of high curvature to regions with low curvature. This indicator is applicable to any kind of error formulated by a metric.

The present work is a continuation of Remaki and Habashi's work in which several improvements are proposed. At first, a review is presented on the Hermite-based mesh adaptation. It is followed by a qualitative analysis of the method to identify points that can be improved. A new error indicator is defined, based on surface integrals of classical Hessian-based error, which is used for element size control the same way as is done in Hermite-based adaptation.

### 4.1 Hermite-based mesh adaptation [62]

Functional-output mesh adaptation aims at adapting the grid locallu so as to minimize the integral error of a given quantity ($I = \int_\Omega u\, dA$). In a Hermite-based approach, the edge-based form of this integral is used $I = \int_\Omega u\, ds$ . Let $E_{e,h} = \left| I_e - I_{e,h} \right|$ be the edge-based functional error. This value can be approximated by reconstructing a Hermitian profile along each edge and using it instead of the unknown exact solution:

$$E_{e,H} = \left| I_{e,H} - I_{e,h} \right| \approx \left| I_e - I_{e,h} \right| \tag{4.1}$$

In equation (4.1), subscript $I_H$ is the integral of a Hermitian profile reconstructed along an edge using solution nodal values ($u_I$ and $u_J$) and the recovered first derivatives of solution at these points. Figure 31 shows a 1-dimensional example of a Hermitian profile reconstructed using the nodal solution and derivative values. The derivatives can be computed using any gradient recovery technique [13, 21, 63, 64].

In classical Hessian-based methods, the metric **M** computed at very computational node could be represented by its ellipsoidal form as:

$$f(x, y, z) = (x, y, z)\mathbf{M}(x, y, z)^T \tag{4.2}$$

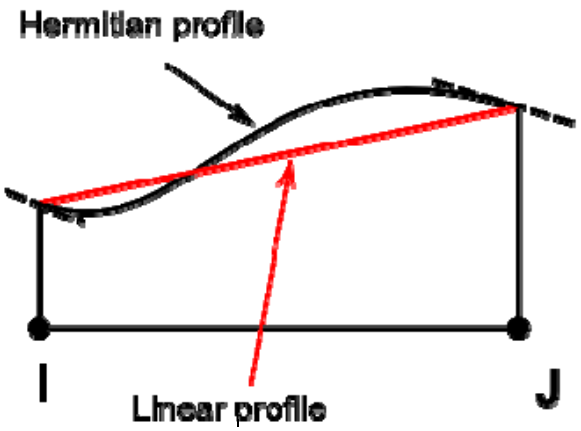



Figure 31: Schematic drawing of a Hermitian reconstructed profile along an edge

Equation (4.2) is an ellipse centered at origin which in a rotated frame of reference with axes aligned along the eigenvectors of metric **M**, the ellipsoid equation becomes:

$$\frac{\bar{x}^2}{\left(1/\sqrt{\lambda_1}\right)^2}+\frac{\bar{y}^2}{\left(1/\sqrt{\lambda_2}\right)^2}+\frac{\bar{z}^2}{\left(1/\sqrt{\lambda_3}\right)^2}=1 \tag{4.3}$$

where $(\bar{x},\bar{y},\bar{z})$ are the transformed coordinates $(\bar{x},\bar{y},\bar{z})=\mathbf{R}^T(x,y,z)$ by the transverse matrix of eigenvalues **R** of metric **M**.

The volume of this ellipsoid is computed based on the equatorial radii as:

$$V=\frac{4}{3}\pi\frac{1}{\sqrt{\lambda_1}}\times\frac{1}{\sqrt{\lambda_2}}\times\frac{1}{\sqrt{\lambda_3}} \tag{4.4}$$

For an ellipsoid two characteristic lengths known as stretching coefficients (or factors) may be defined:

$$S_1=\frac{\sqrt{\lambda_2}}{\sqrt{\lambda_1}} \qquad S_2=\frac{\sqrt{\lambda_3}}{\sqrt{\lambda_2}} \tag{4.5}$$

according to which the eigenvalues can be re-expressed:

$$\lambda_1=\left(\frac{5\pi}{4R_1^2R_2V}\right)^{2/3},\lambda_2=\left(\frac{5\pi R_1}{4R_2V}\right)^{2/3},\lambda_3=\left(\frac{5\pi R_1R_2^2}{4V}\right)^{2/3} \tag{4.6}$$

Now, the Hermitian-based functional error $E_{e,H}$ defined in equation (4.1) is used to rescale the ellipsoid by redefining the volume of ellipse as:

$$V_{E_{e,H}} \triangleq \frac{4}{3}\pi\left(\frac{1}{E_{e,H}}\right)^{3/2} \tag{4.7}$$

This new definition of volume is used to recompute eigenvalues by equation (4.6) and the metric $\mathbf{M} = \mathbf{R}^T\mathbf{\Lambda}\mathbf{R}$ ($\mathbf{\Lambda}$ is the diagonal matrix of new eigenvalues).

### 4.1.1 Drawbacks of Hermite-based error indicator

The beauty of Hermite reconstructed profile is in controlling the local values of error to ensure a smooth varying element size during mesh adaptation. However, it has its own drawbacks which are mentioned in this section.

The Hermite-based method is based on the edge-based definition of a functional, similar to how Hessian-based error estimation is defined. Let's look at the Hermite-based error for one edge. Let's rewrite equation (4.1) in terms of the solution scalar $u$:

$$E_{e,H} = \left|I_{e,H} - I_{e,h}\right| = \left|\int\left(u_H - u_h\right)ds\right| \tag{4.8}$$

Both linear $u_h$ and Hermitian $u_H$ profiles can be re-written using Taylor series expansion about one of nodes of the edge, let's say node $I$:

$$u_h\left(x\right) = u_I + s\left(\frac{\partial u}{\partial s}\right)_I \tag{4.9}$$

$$u_H\left(s\right) = u_I + s\left(\frac{\partial u}{\partial s}\right)_I + \frac{s^2}{2}\left(\frac{\partial^2 u}{\partial s^2}\right)_I + \frac{x^3}{6}\left(\frac{\partial^3 u}{\partial s^3}\right)_I \tag{4.10}$$

By substituting them into equation (4.8):

$$\int\left(u_H - u_h\right)ds = \int\left(\frac{s^2}{2}\left(\frac{\partial^2 u}{\partial s^2}\right)_I + \frac{s^3}{6}\left(\frac{\partial^3 u}{\partial s^3}\right)_I\right)ds \tag{4.11}$$

where $s$ is the local coordinate system along the edge. If equation (4.11) is split into two terms, the first term is related to Hessian-based error computed for the edge:

$$\int (u_H - u_h)\, ds = \frac{1}{2}\int (\vec{x}_J - \vec{x}_I)^T \left(\frac{\partial^2 u}{\partial s^2}\right)_I (\vec{x}_J - \vec{x}_I)\, ds + \int \frac{s^3}{6}\left(\frac{\partial^3 u}{\partial s^3}\right)_I ds \quad (4.12)$$

Therefore, the Hermite-based error indicator is re-scaling eigenvalues according to the Hessian-based error plus 3rd order term. So, one could argue that:

1- How come the normalized metric ellipse is re-scaled via a non-normalized value computed directly from the scalar?
2- What additional information does this method add to the Hessian-based error indicator?

Using the non-normalized scalar values is inconsistent with how metric was computed by normalizing the eigenvalues. Moreover, it is not proper to use the integral of the scalar to change the value of its second derivative.

In addition, the contribution of Hermitian-profile in enriching the information is adding an approximation of the 3rd order terms, but in the original work by Peraire [65] for 1-dimensional cases the contribution of 3rd order and above terms are neglected in comparison to the 2nd order term. Besides, the 3rd order term comes from a reconstructed Hermitian profile which itself was obtained using recovered gradients. The hierarchical levels of approximation are too many to believe that the 3rd term values are reliable.

Another point of debate is using an edge-based definition of the functional. For 2D cases the functional variables are actually curve integrals. But it means that the Hessian-based error indicator is already defined to compute the error for functionals, as it is an integration over the length of edges. So, for 2D problems the Hessian-based error indicator is actually the functional-output error.

For 3D problems, the functionals are surface integrals. Therefore, it would be more appropriate to use the surface integral definition for the error. In the next section, a new definition for functional-output error is proposed and the way it is used for controlling metric ellipsoid sizes (and consequently edge sizes) is improved.

### 4.2 Surface functional-output mesh adaptation

An alternative way to Hermite-based method is to compute a weighted average of surface integrals of classical Hessian-based error and translate it into a nodal value. To do so, an introduction to metric spaces is needed.

#### 4.2.1 Metric Space

A metric space is a space with a defined distance (called metric) notion, for example Euclidean spaces. In Euclidean space, distance is the length of the straight-line segment connecting two points. A non-Euclidean space can be seen as a transformation of Euclidean space using a transformation metric. Figure 32 shows an example of such transformation for a 2D Euclidean space to a continuous metric space. Therefore, the distance in such metric space is the length of the projected curve to be traveled from one point to another one (the projection of the line segment between these two points in Euclidean space into the metric space).

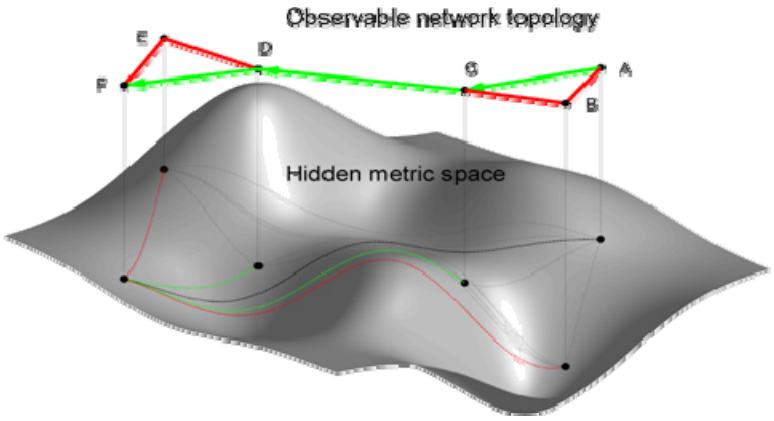



Figure 32: A 2D metric space example defined based on a 2D Euclidean space

Mathematically, a metric space is defined as an ordered pair $(S, d)$ where $S$ is a non-empty set and $d$ is a function on $S$ with following four properties:

1- $d(x, y) \geq 0$

2- $d(x, y) = 0 \Leftrightarrow x = y$

3- $d(x,y) = d(y,x)$

4- $d(x,z) \leq d(x,y) + d(y,x)$

In a discretized Euclidean space (a grid), the metric space defined by any metric defined at nodes will transform it into a discretized metric space (Figure 33).

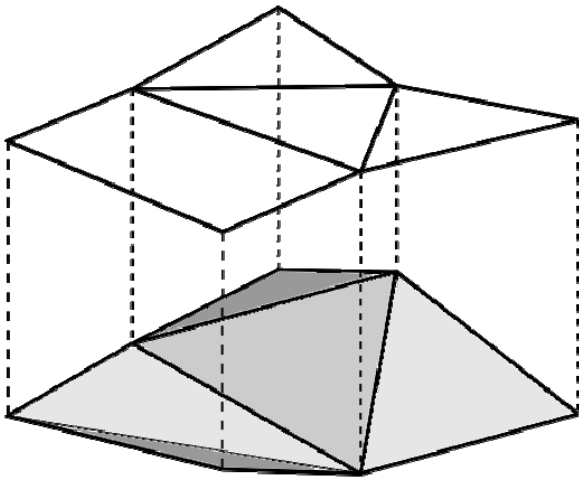


Figure 33: Discretized metric space defined by a metric
defined at nodes of a 2D grid

So, any straight line in Euclidean space is mapped into another straight line in metric space. Therefore, one can deduce that the surface integral of interpolation error over any element in Euclidean space (grid) is equivalent to the area of the mapped element in the metric space. Defining a surface integral of interpolation error is not as easy as what was done for edges in the form of equation (2.11) [65]. But, computing the area of mapped element in the metric space using the edge lengths in that space is trivial. For a triangular element in metric space the area can be computed as:

$$A_M = \sqrt{s(s-e_a)(s-e_b)(s-e_c)} \tag{4.13}$$

where $s = 0.5(e_a + e_b + e_c)$ and $e_a, e_b, e_c$ are respectively the lengths of each of the triangle's sides in the metric space. We recall from (2.11) that they are the Hessian-based errors associated with the corresponding edges of the grid in the Euclidean space.

### 4.2.2 Surface functional-output error indicator

Adopting the smoothing procedure from Hermite-based method, we will redefine the error indicator based on a varying the size of metric ellipsoids at each point. In order to have smoothly changing sizes for metric ellipsoids we will use the surface integral of Hessian-based error. Consider a node of the grid $I$ ; a magnifying operator is defined as:

$$f_I = \frac{\overline{A}_{M,I}}{\max\left(\overline{A}_M\right)} \tag{4.14}$$

where $\overline{A}_M$ is the normalized total area of elements around that node in the metric space, i.e.:

$$\overline{A}_{M,I} = \frac{\sum_{k=1}^{NbSurElem} A_{M,I,k}}{\overline{e}_I^2} \tag{4.15}$$

$\overline{e}_I$ is the average length of edges in metric space that are connected to node $I$. $A_{M,I,k}$ is the area of $k^{\text{th}}$ element in around the node $I$ in the metric space and $\max\left(\overline{A}_M\right)$ is the maximum of normalized areas for all nodes of a specific surface. The magnifying factor defined in equation (4.14) is used to scale the volume of metric ellipsoid of each surface node as (Ref. to equations (4.4) and (4.6)):

$$\hat{V}_I = f_I V_I \tag{4.16}$$

This magnified volume is substituted in the definition of eigenvalues in equation (4.6):

$$\lambda_1 = \left(\frac{5\pi}{4R_1^2 R_2 \hat{V}}\right)^{2/3}, \lambda_2 = \left(\frac{5\pi R_1}{4R_2 \hat{V}}\right)^{2/3}, \lambda_3 = \left(\frac{5\pi R_1 R_2^2}{4\hat{V}}\right)^{2/3} \tag{4.17}$$

resulting in higher error values on surface nodes with high curvature in the solution and smoothing the variation of error near these nodes.

The surface functional-output error is defined based on the modified metric:

$$\hat{\mathbf{\Lambda}} = \begin{pmatrix} \hat{\lambda}_1 & 0 & 0 \\ 0 & \hat{\lambda}_2 & 0 \\ 0 & 0 & \hat{\lambda}_3 \end{pmatrix} \tag{4.18}$$

$$\hat{\mathbf{M}} = \mathbf{R}^T \hat{\mathbf{\Lambda}} \mathbf{R} \tag{4.19}$$

$$\hat{e} = \int \sqrt{\left(\vec{x}_J - \vec{x}_I\right)^T \hat{\mathbf{M}} \left(\vec{x}_J - \vec{x}_I\right)} dx \tag{4.20}$$

Note that equation (4.19) is only defined for surface nodes.

### 4.3 Numerical Results

The test case chosen to study the effect of magnifying factor defined in equation (4.14) for surface nodes is an RAE2822 high-lift airfoil in transonic regime (Table 5). The airflow is simulated using Euler equations. As mentioned in previous chapter, surface functional-output error indicator is meaningful for full 3D cases. Therefore, a 2D-extruded geometry of RAE2822 airfoil is created and a full-tetrahedron mesh is generated over it using Octree method (Figure 34).

The Mach contours of solution computed over the initial mesh are presented in Figure 35. The shock on the upper surface of the airfoil is not well captured and very thick. So, mesh adaptation is needed to improve the quality of the solution especially at the shock.

In order to compare surface functional-output mesh adaptation with Hessian-based mesh adaptation, pressure coefficient is selected as the adaptation scalar and 4 cycles of mesh adaptation/flow simulation is performed for each of Hessian-based and functional-output adaptations.

Table 5: Parameters for RAE2822 transonic airfoil simulation for Euler flow

| Mach Number | Chord length | Pressure | Temperature |
|---|---|---|---|
| $Ma = 0.73$ | $L_c = 0.3809\,m$ | $P_\infty = 101.325\ k\text{Pa}$ | $T_\infty = 288\,K$ |

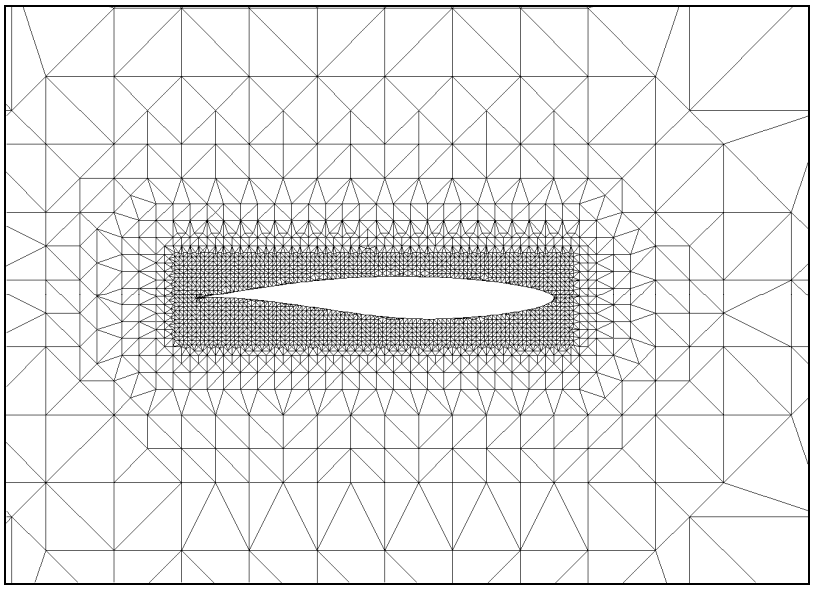


Figure 34: Full-tetrahedron initial mesh around RAE2822 airfoil using Octree method

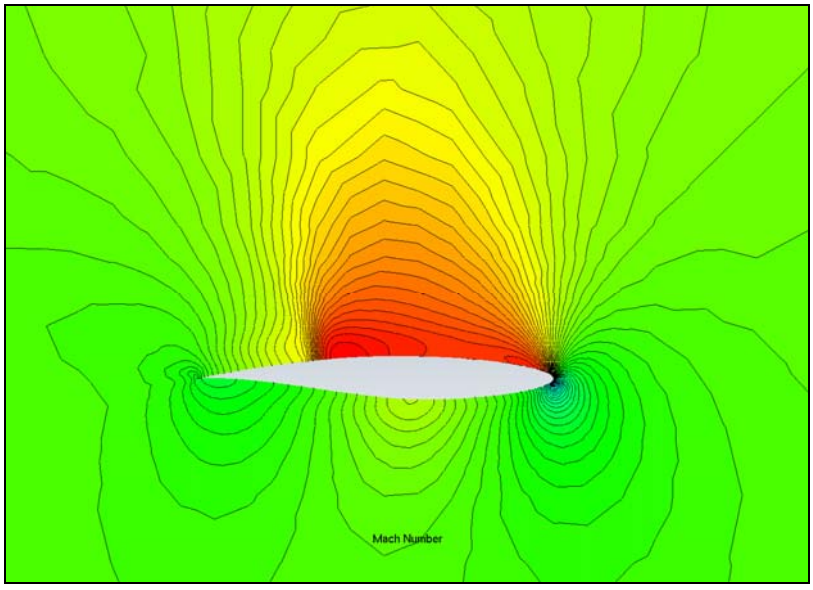
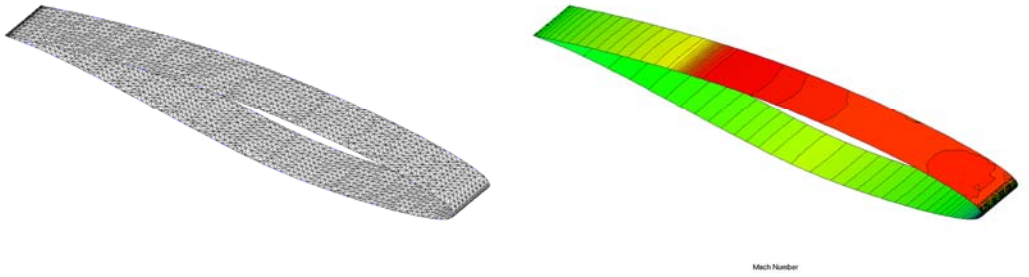
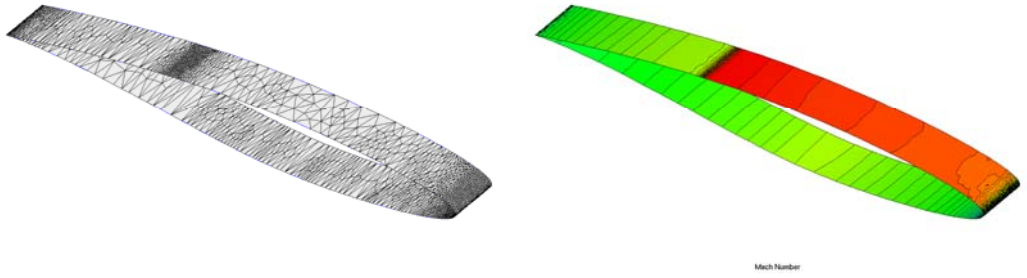


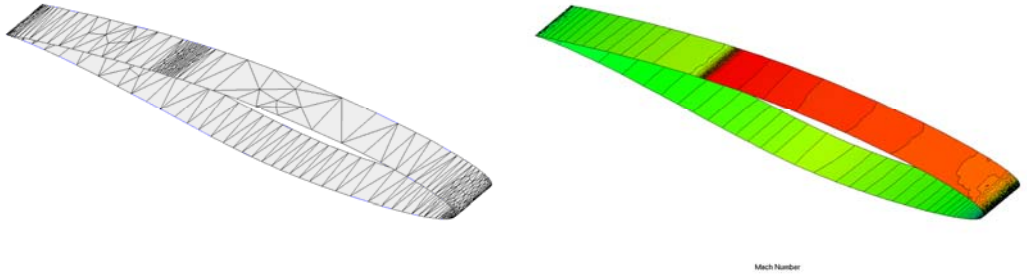

Figure 35: Mach contour levels around RAE2822 transonic airfoil




(a)

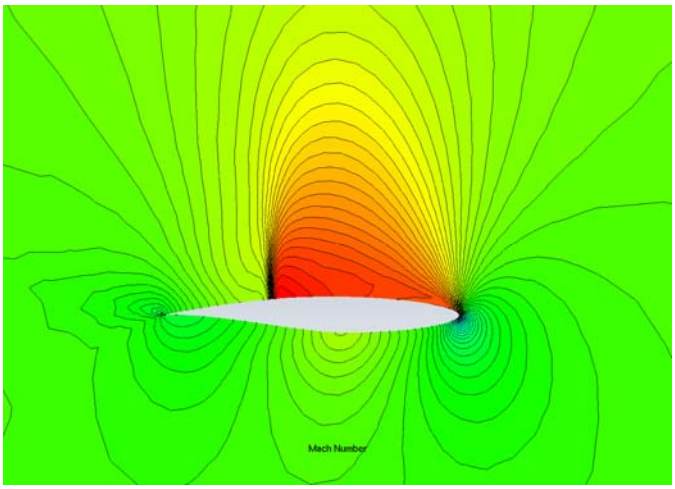



(b)

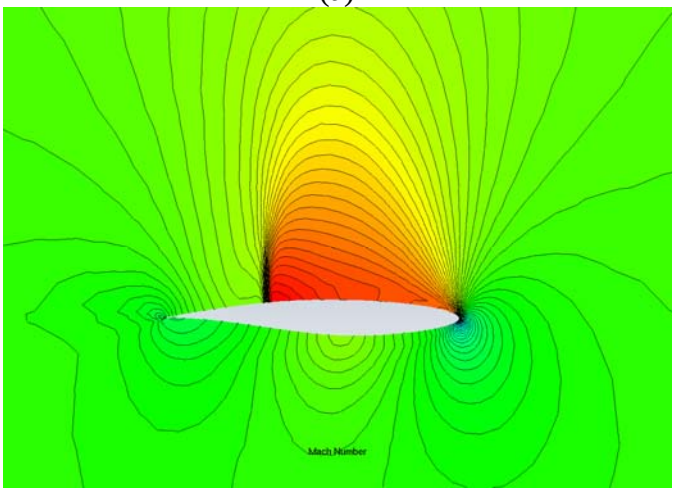



(c)

Figure 36: Mach contours on the airfoil surface for (a) initial mesh and after 4 cycles of (b) surface functional-output adapted mesh and (c) Hessian-based adapted mesh




(a)




(b)

Figure 37: Mach contours levels after 4 cycles of
(a) surface functional-output adapted mesh and
(b) Hessian-based adapted mesh

The initial solution and mesh, $4^{th}$ functional-output (FO) adapted mesh and $4^{th}$ Hessian-based adapted meshes and solutions over them are compared in Figure 36. It can be seen that the FO adaptation has resulted in a denser mesh and more smoothness in element size near shock position compared to Hessian-based adaptation. Figure 38 shows pressure contour plots obtained from solutions shown in Figure 36.

It can be seen that FO adaptation has resulted in a better shock quality (less over and under shoots). Moreover, FO adaptation has helped the solver to converge faster for capturing the position of the shock.

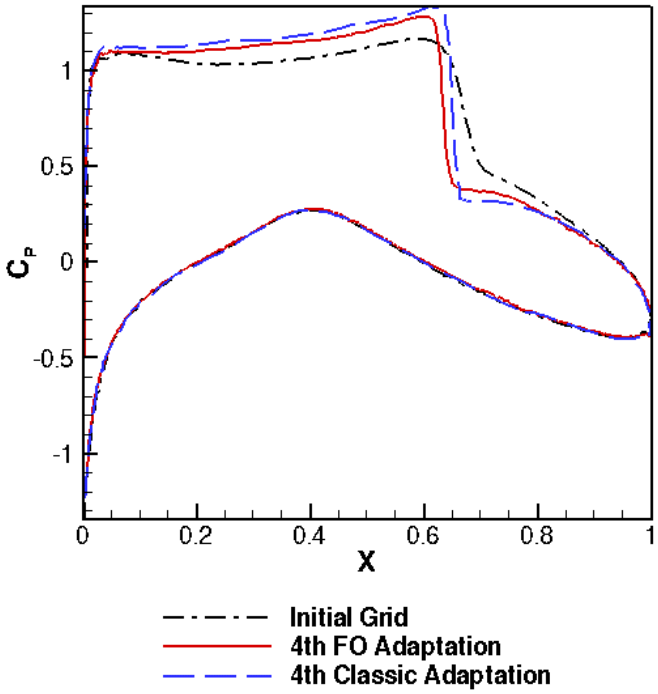



Figure 38: Pressure coefficient plot on the RAE2822 transonic airfoil

## 4.4 Conclusion

In this chapter a new definition for functional-output error indicator was presented. Based on surface integrals of Hessian-based error in the discretized metric space, a magnifying factor was defined to re-scale the volume of metric ellipsoid for each surface node. Numerical results for RAE2822 transonic airfoil show that this new definition can be promising for obtaining a mesh with smoother variation in element size near regions of high curvature in the solution. Moreover, with this error indicator a converged mesh-solution couple can be obtained faster.

## 4.5 Future Work

For functional-output mesh adaptation, the following future steps are suggested:

1- The effect of functional-output adaptation must be studied for full Navier-Stokes flows and the goal must be set to predict lift and drag coefficients more accurately.

2- The drag force is a bigger problem in comparison to the lift for separated flows due to difficulty in predicting the separation point. Functional-output adaptation based on a suitably chosen scalar can be a very useful tool to capture the separation point more accurately.

3- The test case presented in this work was a 2D-extruded geometry. Studying the efficiency of functional-output adaptation on full 3D geometries, e.g. 3D wings, full airplane, etc., is a further step in this field.

# LIST OF REFERENCES


[1]. Ait-Ali-Yahia, D., et al., *Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part II. Structured grids.* International Journal for Numerical Methods in Fluids, 2002. **39**(8): p. 657-673.

[2]. Dompierre, J., et al., *Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part III. Unstructured meshes.* International Journal for Numerical Methods in Fluids, 2002. **39**(8): p. 675-702.

[3]. Habashi, W.G., et al., *Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver-independent CFD. Part I: general principles.* International Journal for Numerical Methods in Fluids, 2000. **32**(6): p. 725-744.

[4]. Alauzet, F., P.J. Frey, and P.-L.G.B. Mohammadi, *3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations.* Journal Computational Physics, 2007. **222**: p. 592-623.

[5]. Loseille, A., A. Dervieux, and F. Alauzet, *Fully anisotropic goal-oriented mesh adaptation for 3D steadty Euler equations.* Journal of Computational Physics, 2010. **229**: p. 2866-2897.

[6]. Frey, P.J. and F. Alauzet, *Anisotropic mesh adaptation for CFD computations.* Computer Methods in Applied Mechanics and Engineering, 2005. **194**(48-49): p. 5068-5082.

[7]. Frey, P.J. and P.-L. George, *Mesh Generation: Application to Finite Elements*. 2nd edition ed2010: Wiley-ISTE.

[8]. Remacle, J.-F., et al., *Anisotropic adaptive simulation of transient flows using discontineous Galerkin methods.* Int. Jr. Num. Meth. Fluids, 2005. **62**: p. 899-923.

[9]. Borouchaki, H., et al., *Delaunay mesh generation governed by metric specifications. Part I. Algorithms.* Finite Elements in Analysis and Design, 1997. **25**(1-2): p. 61-83.

[10]. Borouchaki, H., et al., *Delaunay Mesh Generation Goverend by Metric Specifications Part I: Algorithms.* Finite Elements in Analysis and Design, 1997. **25**: p. 61-83.

[11]. Castro-Díaz, M.J., et al., *Anisotropic unstructured mesh adaption for flow simulations.* International Journal for Numerical Methods in Fluids, 1997. **25**(4): p. 475-491.

[12]. Castro-Díaz, M.J., et al., *Anisotropic unstructured mesh adaptation for flow simulation.* Int. J. Numer. Meth. Fluids, 1997. **25**: p. 475-491.

[13]. Hecht, F., *Métriques et indicateur d'erreur*, in *Mesh Course*2003, Ècole CEA-EDF-INRIA, INRA: Rocquencourt, France.

[14]. Vallet, M.-G., F. Hecht, and B. Mantel. *Anisotropic control of mesh generation based upon a Voroni type method*. in *Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*. 1991. Barcelona, Spain.

[15]. Tam, A., *An Anisotropic Adaptive Method for Solution of 3D Inviscid and Viscous Compressible Flows*, in *Department of Mechanical Engineering*1998, Concordia University: Montreal.

[16]. Tam, A., et al., *Anisotropic mesh adaptation for 3D flows on structured and unstructured grids.* Comput. Methods in Applied Mechechanical Engineering, 2000. **189**: p. 1205-1230.

[17]. Löhner, R. and J. Baum, *Adaptive h-refinement on 3D unstructured grids for transient problems.* International Journal of Numerical Methods in Fluids, 1992. **14**: p. 1407-1419.

[18]. Habashi, W.G., et al., *Certifiable Computational Fluid Dynamics Through Mesh Optimization.* AIAA Journal, 1998. **36**(5): p. 703-711.

[19]. Oden, J., T. Strouboulis, and P. Devloo, *Adaptive finite element methods for high-speed compressible flows.* Int. J. Numer. Meth. Fluids, 1987. **7**: p. 1211-1228.

[20]. Borouchaki, H., P.L. George, and B. Mohammadi, *Delaunay mesh generation governed by metric specifications Part II. Applications.* Finite Elements in Analysis and Design, 1997. **25**(1-2): p. 85-109.

[21]. Almeida, R., et al., *Adaptive finite element computational fluid dynamics using an anisotropic error estimator.* Computer Methods in Applied Mechanics and Engineering, 2000. **182**: p. 379-400.

[22]. Aubé, M.S., et al., *On the impact of anisotropic mesh adaptation on computational wind engineering.* International Journal for Numerical Methods in Fluids, 2009.

[23]. Venditti, D.A. and D.L. Darmofal, *Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows.* Journal Computational Physics, 2003. **187**: p. 22-46.

[24]. Löhner, R., *Applied CFD Techniques: An Introduction Based on Finite Element Methods*2001, Chichester, U.K.: Wiley.

[25]. Zienkiewicz, O.C. and J. Zhu, *A simple error estimator and adaptive procedure for practical engineering analysis.* International Journal for Numerical Methods in Engineering, 1987. **24**: p. 337-357.

[26]. Alauzet, F., *Adaptation de millage anisotrope en trois dimensions. Application aux simulations instationnaires en Mécanique des Fluids*, 2003, Université des Sciences et Techniques, France: Languedoc.

[27]. Bank, R. and J. Xu, *Asymptotically exact a posteriori error estimator, Part I: Grids with superconvergence.* SIAM Journal on Numerical Analysis, 2003. **41**: p. 2294-2312.

[28]. Bank, R. and J. Xu, *Asymptotically exact a posteriori error estimator, Part II: General unstructured grids.* SIAM Journal on Numerical Analysis, 2003. **41**: p. 2313-2332.

[29]. Buscaglia, G., et al., *On Hessian Recovery and anisotropic adaptivity*, in *4th ECCOMAS Computational Fluid Dynamics Conference*1998: Athenes, Greece. p. 403-407.

[30]. Alauzet, F. and P. Frey, *Estimateur d'erreur géométrique et métrique anistropes pour l'adaptation de maillage, Parie I: Aspects théoriques*, in *Technical Report 4759*2003, Institut National de Recherche en Informatique et en Atomatique: France.

[31]. Vallet, M.-G., et al., *Numerical comparison of some Hessian Recovery techniques.* International Journal for Numerical Methods in Engineering, 2007. **72**: p. 987-1007.

[32]. Alauzet, F., et al., *Transient fixed point based unstructured mesh adaptation.* International Journal for Numerical Methods in Fluids, 2003. **43**: p. 729-745.

[33]. Rausch, R., J. Batina, and H. Yang, *Spatial adaptation procedures on tetrahedral meshes for unsteady aerodynamic flow calculations.* AIAA Journal, 1992. **30**: p. 1243-1251.

[34]. Pain, C., et al., *Tetrahedral mesh optimisation and adaptivity for steay-state and transient finite element calculations.* Comput. Methods Appl. Mech. Eng., 2001. **190**: p. 3771-3796.

[35]. Löhner, R., *Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing.* Computing Systems in Engineering, 1990. **1**(2-4): p. 257-272.

[36]. Speares, W. and M. Berzins, *A 3D unstructured mesh adaptation algorithm for time-dependent shock dominated problems.* Int. Jr. Num. Meth. Fluids, 1997. **25**: p. 81-104.

[37]. Habashi, W.G., et al., *Anisotropic mesh adaptation: towards user-independent, mesh-independent, and solver-independent CFD. Part I: General Principles.* Int. Jr. Num. Meth. Fluids, 2000. **32**: p. 725-744.

[38]. McKenzie, S., et al., *On metric tensor representation, intersection, and union*, in *11th ISGG Conference on Numerical Grid Generation*2009: Montreal, QC, Canada.

[39]. Lee, C.K., *Automatic adaptive mesh generation using metric advancing front approach.* Eng. Computations, 1999. **16**(2): p. 230-263.

[40]. Xia, G., D. Li, and C.L. Merkle, *Anisotropic grid adaptation on unstructured meshes*, in *39th Aerospace Sciences Meeting and Exhibiti*2001, AIAA: Reno, NV.

[41]. Najafiyazdi, M. and W.G. Habashi. *Improved Transient-Fixed-Point Mesh Adaptation Using Orthogonality-Preserving Metric Intersection*. in *20th AIAA Computational Fluid Dynamics Conference*. 2011. Honolulu, Hawaii, US.

[42]. Ermy, A.E., *An evaluation of several differencing methods for inviscid fluid flow problems.* Journal of Computational Physics, 1968. **2**: p. 306-331.

[43]. Woodward, P. and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks.* Journal of Computational Physics, 1984. **54**: p. 115-173.

[44]. Omar, E., et al., *Two dimensional wind tunnel test of a NASA supercritical airfoil with various high lift systems*, 1973, NASA.

[45]. Habashi, W.G., et al., *FENSAP-ICE: A FULL-3D IN-FLIGHT ICING SIMULATION SYSTEM FOR AIRCRAFT, ROTORCRAFT AND UAVS*, in *24th Congress of International Council of the Aeronautical Sciences*2004: Yokohama, Japan

[46]. Emery, A., *An evaluation of several differencing methods for inviscid fluid flow problems.* Journal of Computational Physics, 1968. **2**: p. 306-311.

[47]. Alauzet, F., et al., *3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations.* Journal Computational Physics, 2007. **222**: p. 592-623.

[48]. Fossati, M. and M. Najafiyazdi, *Unsteady mesh adaptation via Reduced Order Modeling*, in *20th AIAA Computational Fluid Dynamics Conference*2011: Honolulu, Hawaii, US.

[49]. Bank, R. and R. Rannacher, *An optimal control approach to a posteropri error estimation in finite element methods,* in *Acta Numerica 2001*, A. Iserles, Editor 2001, Cambridge University Press: Cambrige.

[50]. Becker, R. and R. Rannacher. *Weighted a postereiori error control in finite element methods*. in *ENUMATH-97*. 1998. Heidelbereg, World Scientific, Singapore.

[51]. Braack, M. and R. Rannacher, *Adaptive finite element methods for low-Mach-number flows with chemical reactions*, 1999, von Larman Institute for Fluid Dynamics: VKI Lecture series.

[52]. Peraire, J. and A.T. Patera, *Bounds for linear-functional outputs of coercive partial ifferential equations: Local indicators and adadptive refinement*, in *Advances in Adaptive Computational Methodsd in Mechanics*, P. Ladevèz and J. Oden, Editors. 1998, Elsvier: Amsterdam.

[53]. Machiels, L., J. Peraire, and A.T. Patera, *A posteriori finite element output bounds for the incompressible Navire-Stokes euquations: application to a natural convection problem.* Journal of Computational Physics, 2001. **172**: p. 401-425.

[54]. Giles, M.B., et al., *Adaptive error control for finite element approximations of the elift and rag in a viscous flow*, 1997, Oxford Computing Laboratory: Oxford.

[55]. Houston, P. and E. Süli, *hp-adaptive discontinuous Galerkin finite element methos for first-order hyperbolic problems.* SIAM Journal of Scientific Computing, 2001. **23**(4): p. 1226-1252.

[56]. Larson, M.G. and T.J. Barth, *A posteriori error estimation for iscontinuous Galerekin approximations of hyberbolic systems*, 1999, NASA.

[57]. Formaggia, L., S. Micheletti, and S. Perotto. *Anisotropic mesh adaptation with application to CFD problems*. in *Figth World Congress on Computationa Mechanics*. 2002. Vienna: Austria.

[58]. Venditti, D.A. and D.L. Darmofal, *Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow.* Journal Computational Physics, 2000. **164**: p. 204-227.

[59]. Venditti, D.A. and D.L. Darmofal, *Grid Adaptation for Functional Outputs:Application to Two-Dimensional Inviscid Flows.* Journal Computational Physics, 2002. **176**: p. 40-69.

[60]. Fidkowski, K.J. and D.L. Darmofal. *Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics: Overview and Recent Results*. in *AIAA Aerospace Sciences Meeting and Exhibit*. 2009. Orlando, Florida.

[61]. Park, M.A. and J.-R. Carlson, *Turbulent Output-Based Anisotopic Adaptation*, in *48th AIAA Aerospace sciences Meeting Including the New Horizons Forum and Aerospace Exposition*2009: Orlando, Florida, USA.

[62]. Remaki, L. and W.G. Habashi, *A Hermite-Based Mesh Adaptation for Functional Outputs Improvement in Fluid Flow Simulation* AIAA Journal, 2009. **47**(8): p. 1965-1976.

[63]. Zienkiewicz, O.C. and J. Zhu, *The superconvergent patch recovery and a posteeriori error estimatees. Part I: The recovery technique.* International Journal for Numerical Methods in Engineering, 1992. **33**: p. 1331-1364.

[64]. Labbé, P. and A. Garon, *A robust implemntation of Zienkiewicz and Zhu's local patch recovery meethod.* Communications in Numerical Methods in Engineering, 1995. **11**: p. 427-434.

[65]. Peraire, J., et al., *Adaptive remeshing for compressible flows.* Journal Computational Physics, 1987. **72**(449-466).