r-adaptive algorithms for supersonic flows with high-order Flux Reconstruction methods

Firas Ben Ameur[a,b,*], Joachim Balis[a], Ray Vandenhoeck[a,c], Andrea Lani[a,b], Stefaan Poedts[b,d]

[a] *Von Karman Institute for Fluid Dynamics, Waterloosesteenweg 72, 1640, Sint-Genesius-Rode, Belgium*
[b] *Centre for Mathematical Plasma-Astrophysics, KU Leuven, Celestijnenlaan 200B, B-3001 Leuven, Belgium*
[c] *Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300, B-3001, Leuven, Belgium*
[d] *Institute of Physics, University of Maria Curie-Skłodowska, ul. Radziszewskiego 10, PL-20-031 Lublin, Poland*

Abstract

The present paper addresses the development and implementation of the first r-adaptive mesh refinement (r-AMR) algorithm for a high-order Flux Reconstruction solver. The r-refinement consists on nodal re-positioning while keeping the number of mesh nodes and their connectivity frozen. The developed algorithm is based on physics-driven spring-analogies, where the mesh can be seen as a network of fictitious springs. While AMR increases the local mesh density, the high-order Flux Reconstruction method potentially provides a more accurate detection of complex flow features over relatively coarser mesh, when compared to low-order methods. In this work, a concise overview of the Flux Reconstruction method and spring-based AMR techniques will be given, followed by some promising results of r-AMR applied to benchmark high-order steady-state supersonic flow simulations.

Keywords: Adaptive mesh refinement, r-AMR, Spring Analogy, Flux Reconstruction, Cartesian meshes, Hypersonic flows

1. Introduction

The vast majority of state-of-the-art Computational Fluid Dynamic (CFD) codes tackling high-speed flows rely upon low-order solvers, including Finite Volume (FV), Finite Difference (FD) and Finite Element (FE) schemes. Such methods generally offer a second-order accuracy in space for applications featuring shock waves. By virtue of their robustness, reliability and ease of implementation, these low-order schemes are nowadays widespread in the industry as well as in the academic world. However, when applied to certain types of problems (e.g. aeroacoustics and vortex-dominated flows), they are not sufficiently accurate due to their high numerical dissipation [1]. Recently, high-order methods have drawn more attention among researchers in the field of CFD thanks to their increased computational efficiency “per degree of freedom" (DOF) over their low-order counterparts [2] along with their lower numerical dissipation. Amidst all developed FE-type methods, the Flux Reconstruction (FR)/ Correction Procedure via Reconstruction (CPR) formulation is amongst the newest framework. Originally developed by Huynh for 1D advection problems in [3], the FR method is a framework which allows to develop new high-order schemes while being simple and computationally efficient, particularly on graphical processor units (GPU) [4, 5].Despite those advantages, the FR method suffers from the same pacing items as the other high-order methods such as the slow convergence to steady state and the lack of robust shock capturing methods [6]. When trying to approximate a discontinuity (such as a jump in a physical property due to a shock) with a polynomial representation, the Gibbs phenomenon causes a decrease in accuracy and leads to the appearance of spurious oscillations which can trigger instabilities in the numerical computations [7]. To solve this issue, different limiting and artificial viscosity (AV) strategies have been developed as

*Corresponding author
Email addresses: firas.benameur@vki.ac.be (Firas Ben Ameur), joachim.balis@aeronomie.be (Joachim Balis), ray.vandenhoeck@vki.ac.be (Ray Vandenhoeck), andrea.lani@kuleuven.be (Andrea Lani), stefaan.poedts@kuleuven.be (Stefaan Poedts)

summarized in [8]. Although such shock capturing schemes alleviate the oscillations introduced by a discontinuity, they generally do not ensure that the flow variables such as the pressure will remain positive at every point of the computational domain and at every time step. Different research groups have been developing methods and numerical techniques for high-order positivity preservation. More details about positivity preservation can be found in [9] for scalar conservation laws, [10, 11] for compressible Euler equations, [8] for FR compressible Navier-Stokes flow equations and [12] for FR thermochemical nonequilibrium flow equations.
Suitable Adaptive Mesh Refinement (AMR) algorithms that are well developed for other FE-type methods and could ease shock capturing for FR methods are currently lacking. The refinement procedure in itself can be split in 3 categories: *p-*, *h-*, and *r-refinement*. The p-refinement consists in locally increasing the order of the polynomial describing the solution and is particularly efficient when the solution to approximate is smooth [13]. The h-refinement method locally refines the mesh by adding grid points where needed and is generally preferred when discontinuities such as shocks are present [14]. Finally, the r-refinement procedure consists in moving the grid points such that the nodal density is increased in the zones of interest. r-refinement is particularly suitable to parallel architectures since it does not require load balancing algorithms [15, 16]. Over the last decade, considerable effort has been devoted to couple high-order methods to AMR strategies. In 2010, Premasuthan, Liang and Jameson tested an approach in a high-order Spectral Difference (SD) context where local refinement was performed in zones where the artificial viscosity was non-zero [17]. The same year, Li, Premasuthan and Jameson developed and applied several h- and p- refinement techniques in the same SD framework [18]. More recently, most of the research has been focused on the coupling of h- and p- refinement with high-order methods. Coupled hp AMR for hybridized and standard Discontinuous Galerkin (DG) methods have been investigated in the framework of compressible flows [19, 20]. Similarly, a novel p-adaptation method for high-order DG methods was lately developed to compute aerodynamic force coefficients [21]. In the correction procedure via reconstruction (CPR) formulation, an h-adaptation approach was successfully implemented and tested on supersonic cases [22]. In the high-order FR framework, h-refinement methods combined with artificial diffusivity (AD) have been applied to capture shocks in transonic and supersonic problems [23, 24] and to resolve discontinuities in magnetohydrodynamics systems [25].
In our previous work [26, 15], a robust and efficient AMR algorithm for FV has been developed and already applied to a wide selection of steady-state test cases. The existing FV-AMR algorithm works on triangles, quadrilateral and tetrahedral cells, is parallel and physics-independent, letting the user decide which monitor physical quantity to use for driving the adaptation according to the application. In addition, in [8, 12], a state-of-the-art fully implicit high-order FR code has been implemented and used to solve Euler/Navier-Stokes equations, where its structure is extremely modular and can be easily coupled to arbitrary sets of advection-diffusion-reaction Partial Differential Equations (PDE).
In this work, we present the development of novel AMR algorithms within the FR framework and its application to a number of representative test cases featuring steady-state supersonic flows on meshes with quadrilateral cells.
The resulting solver has been implemented within the COOLFluiD platform [27, 28] (a world-class open source framework for multi-physics modeling and high-performance computing, particularly focused on simulating hypersonic flows [29, 30], radiation [31], laboratory [32] and space plasmas [33, 34, 35, 36, 37]).
The paper is structured as follows:
- Sec.2 reviews the state-of-the-art of FR method, following the work of Vandenhoeck and Lani [8];
- Sec.3 describes spring-based AMR, following the work of Ben Ameur and Lani [15, 26];
- Sec.4 presents a new method, denominated Vertex springs;
- Sec.5 shows some results for the Vertex springs method;
- Sec.6 presents the main novelty of our work: a method denominated Sub-cell order-dependent spring analogy;
- Sec.7 shows very promising results for the latter, on both linear and curvilinear meshes, in comparison with some analytical, h-refinement and high-order state-of-the-art results. - Sec.10 covers the main aspects of the implementation of FR-AMR;

2. Flux Reconstruction Solver

The FR method was originally developed by Huynh in [3]. It is a compact high-order FE-type method that offers a unifying formulation for various classes of high-order schemes (e.g. nodal Discontinuous Galerkin, Spectral Finite Difference). The formulation of the FR approach proposed by Huynh is applicable to deal with advection-diffusion

problems on structured and unstructured grids [38].
This section presents a brief review of the Flux Reconstruction method in one dimension, including the following main algorithmic steps, taken from [8]:

1. Consider solving the one-dimensional advection-diffusion problem within an arbitrary domain Ω, given by:

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0 \quad \text{with} \quad x \in \Omega = [x_L, x_R] \quad \text{and} \quad f = f\left(u, \frac{\partial u}{\partial x}\right). \tag{1}$$

   In Eq.1, x is the one dimensional coordinate, t is the time variable, $u(x,t)$ is a conservative variable, and f is the flux which is a scalar for the 1D case. As the FR method is classified as a FE-type method, the first step of the procedure involves partitioning the spatial domain Ω into a finite number N of non-overlapping, non-empty, open sub-domains Ω'_n such that:

$$\Omega'_n = \{x \mid x_n < x < x_{n+1}\} \quad \text{with} \quad x_1 = x_L \quad \text{and} \quad x_{N+1} = x_R. \tag{2}$$

2. The exact solution u of the conservation law (Eq.1) is approximated by a continuous function u_n^δ within each element Ω_n. This function u_n^δ corresponds to a polynomial of degree P within Ω_n and is identically zero on $\Omega \setminus \Omega_n$, while being generally discontinuous across elements. The approximate solution polynomial u^δ within the entire domain Ω is constructed as:

$$u^\delta = \sum_1^N u_n^\delta \tag{3}$$

   In the same manner, the exact flux f is approximated by a polynomial of degree $P+1$ within each sub-domain Ω_n.

3. Each element Ω_n together with the approximate solution u_n^δ and flux f_n^δ inside it, is transformed to a standard reference element $\Omega_S = \{\xi \mid -1 \leq \xi \leq 1\}$, where all computations are performed. The solution u_n^δ within each sub-domain Ω_n can then be obtained by solving the transformed conservation equation 1 within the reference element Ω_S:

$$\frac{\partial \hat{u}^\delta}{\partial t} + \frac{\partial \hat{f}^\delta}{\partial \xi} = 0 \quad \text{with} \quad \xi \in \Omega_S, \quad \hat{u}^\delta = \hat{u}^\delta(\xi,t) = J_n u_n^\delta(\Theta_n(\xi),t) \quad \text{and} \quad \hat{f}^\delta = \hat{f}^\delta(\xi,t) = f_n^\delta(\Theta_n(\xi),t), \tag{4}$$

   where J_n represents the Jacobian of the mapping $\Theta_n(\xi)$.

4. The approximate solution $\hat{u}^\delta$ is defined as a degree P polynomial based on a Lagrange polynomial basis related to the set of $P+1$ distinct solution points ξ_i within Ω_S, where the two end-points of the standard element Ω_S, i.e. $\xi = 1$ and $\xi = -1$, are referred to as the flux points.

5. This step consists of determining a common interface solution $\hat{u}^{\delta I}$ at the boundaries of the standard element Ω_S, i.e. $\xi = \pm 1$, where the flux points of two neighboring elements coincide.

$$u^{\delta I} = \frac{u^{\delta -} + u^{\delta +}}{2} \tag{5}$$

   where $u^{\delta -}$ and $u^{\delta +}$ are respectively the approximate solutions evaluated at the left and right boundaries.

6. A corrected solution gradient $\hat{q}^\delta$ is computed. $\hat{q}^\delta$ is defined in terms of two correction functions $g_L(\xi)$ and $g_R(\xi)$ of degree $P+1$, which approximate the zero function within the reference element Ω_S, and satisfy the following boundary conditions: [39]

$$g_L(-1) = 1, \qquad g_R(-1) = 0, \qquad g_L(1) = 0, \qquad g_R(1) = 1. \tag{6}$$

   Furthermore, the correction function for the right boundary $g_R(\xi)$ is defined as the reflection of the correction function $g_L(\xi)$ with respect to the vertical axis in order to ensure symmetry of the correction process $g_L(\xi) = g_R(-\xi)$.

The corrected solution gradient $\hat{q}^\delta$ is then computed via the following expression:

$$\hat{q}^\delta = \frac{\partial \hat{u}^\delta}{\partial \xi} + (\hat{u}_L^{\delta I} - \hat{u}_L^\delta)\frac{dg_L}{d\xi} + (\hat{u}_R^{\delta I} - \hat{u}_R^\delta)\frac{dg_R}{d\xi}, \tag{7}$$

where $\hat{u}_L^{\delta I}$ and $\hat{u}_R^{\delta I}$ are the common interface solutions at the left and right boundaries of Ω_S, and $\hat{u}_R^\delta = \hat{u}^\delta(1)$ and $\hat{u}_L^\delta = \hat{u}^\delta(-1)$ are the values of the approximate solution polynomial at the left and right boundaries. The Vincent-Castonguay-Jameson-Huynh (VCJH) scheme is used to define the exact polynomial expression of g_L and g_R, as found in [40].

7. On the reference element Ω_S, the approximate discontinuous flux $\hat{f}^{\delta D}(\xi)$ is calculated. Based on $\hat{u}^\delta$ and $\hat{q}^\delta$, the approximate discontinuous flux $\hat{f}^{\delta D}$ is constructed as a polynomial of degree P in the same manner as the approximate solution $\hat{u}^\delta$.

8. The transformed common interface fluxes $\hat{f}^{\delta I}$ at the flux points of the standard element Ω_S, i.e. $\xi = \pm 1$, is calculated. $\hat{f}_L^{\delta I}$ and $\hat{f}_R^{\delta I}$ are denoted as the common interface flux at the left and right end of the standard element, respectively. In the case of the Navier-Stokes equations, interface flux schemes are used to compute the advective and diffusive interface flux, which can found in [8].

9. The flux must be continuous across element boundaries in order to obtain a conservative scheme. Therefore, a correction flux function $\hat{f}^{\delta C}$ is added to the approximate discontinuous flux $\hat{f}^{\delta D}$ in order to obtain a continuous flux $\hat{f}^\delta$. The correction flux $\hat{f}^{\delta C}$ is a polynomial of degree $P + 1$ defined as:

$$\hat{f}^{\delta C} = (\hat{f}_L^{\delta I} - \hat{f}_L^{\delta D})h_L(\xi) + (\hat{f}_R^{\delta I} - \hat{f}_R^{\delta D})h_R(\xi), \tag{8}$$

where $\hat{f}_L^{\delta D} = \hat{f}^{\delta D}(-1)$ and $\hat{f}_R^{\delta D} = \hat{f}^{\delta D}(1)$ are the values of the approximate discontinuous flux at the left and right boundaries. $h_L(\xi)$ and $h_R(\xi)$ represents $P + 1$ correction functions equivalent to the correction functions $g_L(\xi)$ and $g_R(\xi)$ with the same properties expressed in Eq.6. The continuous flux $\hat{f}^\delta$ within the standard element Ω_S is then written as the sum of the approximated discontinuous flux and the correction flux.

10. The divergence of the total flux $\hat{f}^\delta$ at each solution point ξ_i is computed. This step is relatively straightforward as the derivatives of both the Lagrange polynomials and correction functions are exactly known.

11. Finally, a time marching strategy is applied to advance the approximate solution $\hat{u}^\delta$ in time using the following expression:

$$\frac{d\hat{u}_i^\delta}{dt} = -\frac{\partial \hat{f}^\delta}{\partial \xi}(\xi_i). \tag{9}$$

The extension of the 1D philosophy of the FR method to deal with 2D quadrilateral and 3D hexahedral elements is straightforward. To this end, a tensor product basis is defined based on the one-dimensional Lagrange polynomials on the reference element, as shown in [41].
Since the FR method can obtain arbitrary high orders of accuracy simply by increasing the order of the polynomials that represent the approximated solution, the method is inherently compact and does not require a wide stencil at higher orders. This fact goes hand in hand with r-refinement since the latter also requires only compact stencils.

3. Spring-Based Adaptive Mesh Refinement

In [15], the authors have proposed and developed, within a Finite Volume framework, a r-refinement adaptation algorithm based on a user-defined flow field variable, e.g. density or pressure, that relies upon the solution of pseudo-elastic systems associated to the given mesh. The same philosophy is applicable to the FR framework. Consequently, a brief overview is given in the following.
Let $n \in \mathbb{N}$ be the number of the nodes in a mesh $\mathcal{M}$ and let $\mathbf{P}$ be the set of the nodes positions inside $\mathcal{M}$. Let $\mathbf{L}$ be the incidence matrix defined as:

$$L_{ij} = \begin{cases} 1, & \text{if nodes } i \text{ and } j \text{ are edge-connected} \\ 0, & \text{otherwise.} \end{cases}$$

To achieve the equidistribution condition of r-refinement, the line integral I, expressed in Eq.10, must be constant.

$$I = \int_0^1 W(\mathbf{r}(s)) \cdot r'(s) ds = \text{constant}, \tag{10}$$

where $\mathbf{r}(s) = \mathbf{P_i} + s(\mathbf{P_j} - \mathbf{P_i})$ is the parametrization of the edge connecting nodes i and j in function of the parameter $s \in [0, 1]$. Eq.10 is the solution of the Euler-Lagrange equation to the minimization of the energy which reads:

$$E_{ij} = L_{ij} \int_0^1 W(\mathbf{r}(s))(\mathbf{P_j} - \mathbf{P_i})^2 ds, \tag{11}$$

Considering the weight function as a constant, the energy equation can be re-written in a form analogous to the classical spring potential energy equation. In Eq.12, we can identify W_{ij} as the stiffness coefficient between nodes i and j and an equilibrium spring length set to zero.

$$E_{ij} = L_{ij} W_{ij} (\mathbf{P_j} - \mathbf{P_i})^2, \tag{12}$$

The simplest optimization problem depends on finding the equilibrium positions between two adjacent nodes in the mesh $\mathcal{M}$ based on a network of springs:

$$\frac{\partial E}{\partial \mathbf{P}} = 0 \qquad \& \qquad \frac{\partial^2 E}{\partial \mathbf{P}^2} > 0. \tag{13}$$

The analytic Jacobian is defined as:

$$\frac{\partial E_{ij}}{\partial \mathbf{P_i}} = -2 L_{ij} W_{ij} (\mathbf{P_j} - \mathbf{P_i}) = 0. \tag{14}$$

After simplifying the constant and collecting the contributions of each node, we obtain:

$$\sum_{j=1}^{n} L_{ij} W_{ij} (\mathbf{P_j} - \mathbf{P_i}) = 0. \tag{15}$$

The resulting linear system, i.e. the pseudo-elastic system, to be solved using the Generalized Minimal RESidual (GMRES) algorithm complemented by a parallel Additive Schwartz Preconditioner as provided by the PETSc toolkit [42, 43, 44, 45], is expressed in Eq.16.

$$A_{[n \times n]} \mathbf{P}_{[n \times 1]} = b_{[n \times 1]}, \tag{16}$$

where

$$A_{ij} = \begin{cases} -L_{ij} W_{ij}, & if\ i \neq j \\ \sum_{j=1}^{n} L_{ij} W_{ij}, & if\ i = j. \end{cases}$$

and $\mathbf{b}$ contains the non-homogeneous terms of the Dirichlet or the Neumann boundary conditions. As a result, the nodal re-positioning obeys to the following relation:

$$\mathbf{P}^{k+l} = (1 - \omega)\mathbf{P}^k + \omega \mathbf{P}_{new}, \tag{17}$$

where $\mathbf{P}^k$ is the old node position at iteration k, $\mathbf{P}^{k+l}$ is the new *relaxed* node position at iteration $k + l$, with $l > 0$ and $\mathbf{P}_{new}$ is computed from Eq.16. Within this work, we will mainly target the linear spring analogy. To achieve that goal, the physics-driven linear stiffness can be used based on the selected flow variables gradients. Eq.18 expresses the linear spring stiffness coefficient, k_{ij}^L, between two edge-connected nodes i and j with a respective nodal state U_i and U_j. The weight function introduced in a previous section is described as:

$$W_{ij} = k_{ij}^L = |U_j - U_i|. \tag{18}$$

Choosing such stiffnesses ensure that the r-refinement algorithm is physics-driven because the spring constants will vary depending on the local physical properties of the flow. Nevertheless, supersonic flows display high variations of

physical variables over narrow regions and, for the nodes located in the vicinity of shocks, the values of the spring constants computed with Equation 18 can reach high values. For that reason, in our implementation, these values are truncated and bounded thanks to a P^2 algorithm, which is based on the use of quantiles (the p-quantile of a distribution is defined as the value below which 100 % of the distribution lies) [46]. The P2 algorithm gets all the spring constants one after the other and dynamically computes a user-defined minimum and maximum quantile. Afterwards, the AMR algorithm ensures that all the stiffnesses remain between these two bounds.

3.1. Boundary nodes

Two types of the boundary conditions are defined within our r-AMR algorithms:

1. Dirichlet (i.e. locked node) where the node position is kept constant:

$$P_i^m = P_i^0,$$

2. Neumann (i.e. moving node in boundary) where only the tangential displacement is allowed. Mathematically, this implies:

$$\frac{\partial(\mathbf{P_i} \cdot \mathbf{n_i})}{\partial \mathbf{x}} = 0,$$

where $\mathbf{n_i}$ is the boundary face normal vector.

4. Vertex Spring approach for linear meshes

Note: Hereby and after, the authors will embrace the convention proposed by Bassi and Rebay in [47]: each cell is denoted *PmQn* where *m* stands for the order of the polynomial reconstruction inside the cell and *n* denotes the order of the geometrical approximation of the cell faces.

Following the steps of FV-AMR developed in [15], the linear spring coefficient is computed as the state gradient of the geometric nodes and placed all along the edge connecting the nodes i & j as shown in Fig.1.

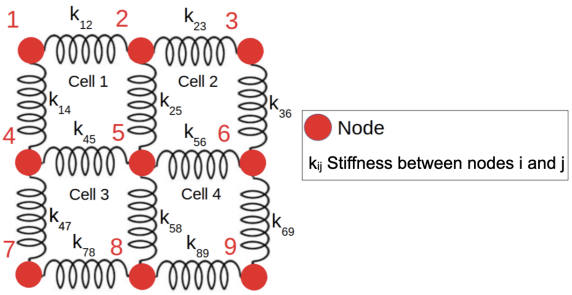

Figure 1: Vertex spring network of 2D quadrilaterals for FR

In order to evaluate the nodal states used to compute the linear spring stiffness in Eq.18, the P^{th}-order polynomial describing the approximated solution is extrapolated, thanks to the Lagrange polynomials, to the corner nodes as shown in Fig.2. To this end, the nodal state at a node i is evaluated as the average of the contributions of each solution polynomial inside elements attached to the considered node i. Eq.19 corresponds to the latter approach.

$$U_i = \frac{1}{N} \sum_{n=1}^{N} \hat{u}_n^{\delta}(x_i), \tag{19}$$

where $\hat{u}_n^{\delta}$ is the approximated solution polynomial within the elements n containing the node i, $\boldsymbol{x_i}$ is the vector of the coordinates of the node i and N represents the number of edges attached to the node i as shown in Fig.1.

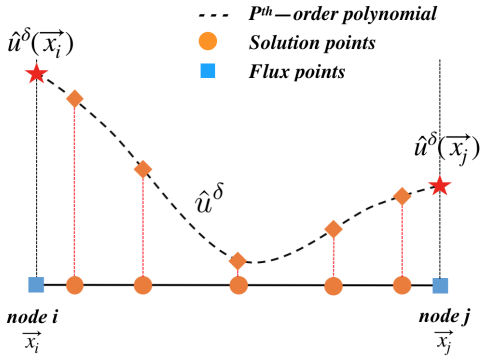

Figure 2: Evaluation of the nodal states

5. Results : Vertex spring approach for linear meshes for FR-AMR Simulations

The results of the Vertex Spring method are obtained for the steady wedge channel flow based on the FR-AMR parameters in App.12. The test case conditions are presented in Tab.1 and Tab.2, while the test case geometry and the corresponding structured mesh are shown in Fig.3.

Table 1: 2D wedge – Flow characteristics

Physical Model	M	ρ [-]	ρu [-]	ρv [-]	ρE [-]
Perfect gas	2	1	2.36643	0	5.3

Table 2: 2D wedge – Mesh characteristics

Dimensions	Type	# Elements	BC 1	BC 2	BC 3	BC 4
2D	Quadrilateral	3710	Inlet	Outlet	Symmetry	no-slip wall

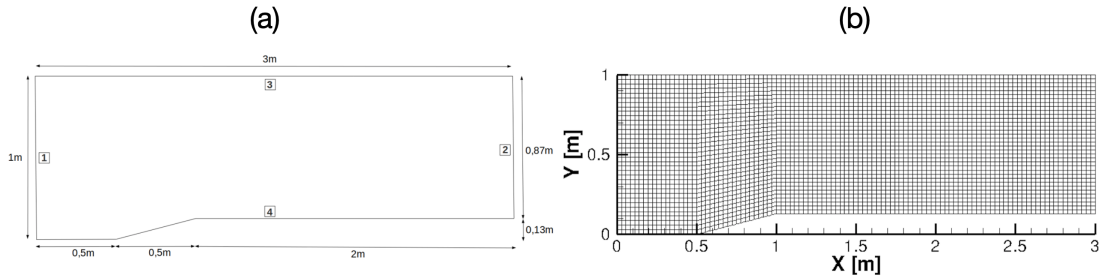

Figure 3: (a) 2D wedge geometry – (b) Initial mesh

As shown in Fig.4, in the final adapted mesh, the oblique shock, the expansion wave and their reflections are perfectly resolved for the different orders, mainly $P1$ (i.e. second order of accuracy in space), $P2$ (i.e. third order of accuracy in space) and $P3$ (i.e. fourth order of accuracy in space). An improved accuracy is obtained by combining the AMR algorithms and the FR solver, as demonstrated in Fig.5.

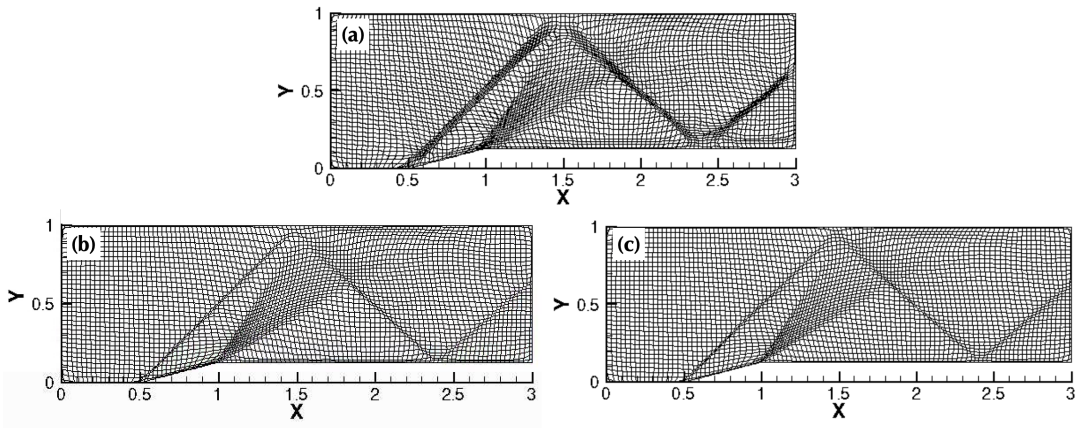

Figure 4: (a) P1 Adapted Mesh – (b) P3 Adapted Mesh – (c) P3 adapted mesh

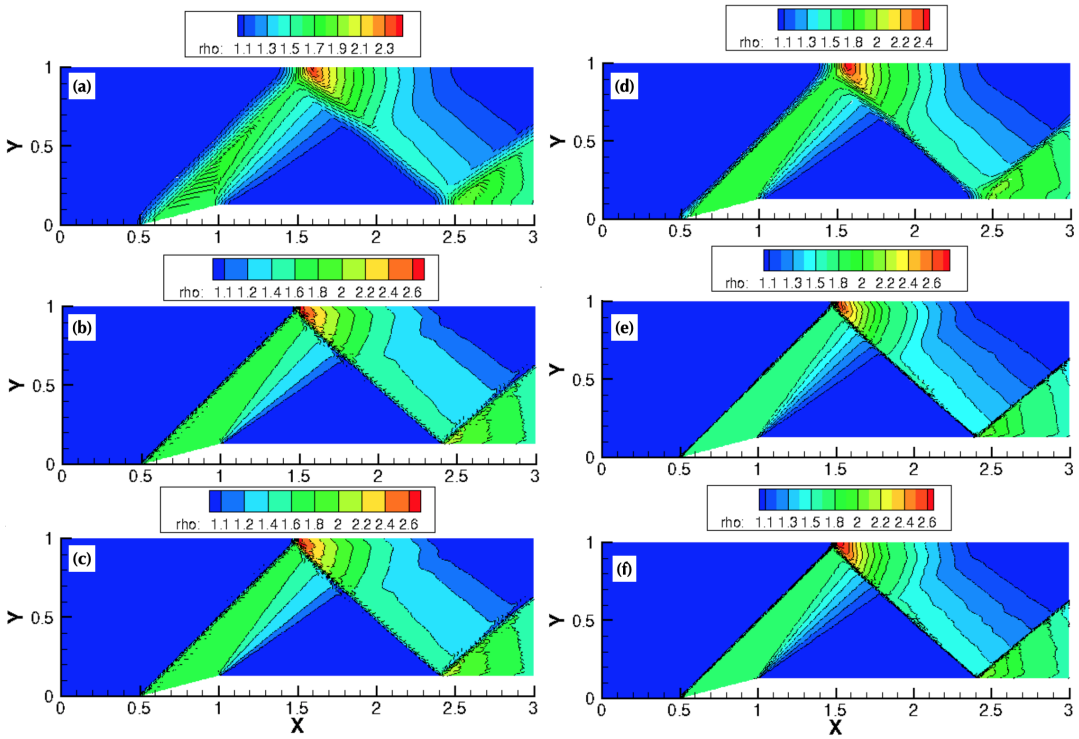

Figure 5: (a→c) P1→P3 non adapted density contours – (d→f) P1→P3 adapted density contours

6. Sub-cell order-dependent spring analogy

6.1. Motivation

The FR method is based on the knowledge of the state values in the solution points. These solution points are always interior to the element and are therefore different from the corner nodes. If we consider the use of a coarse mesh across a shock (which should be possible since we are using a high-order method), it could happen that the state values obtained at the nodes are very close to each other, when applying the Vertex Spring approach. To illustrate this

possibility, let us consider a P3 polynomial reconstruction in a 1D reference element as shown in Fig.6. This graph displays the variations of a normalized density. Starting with the known state values at the 4 solution points (located at the Gauss-Legendre quadrature points), a Lagrangian fit was computed and the state values at the boundaries (i.e. flux points) were obtained by extrapolation.

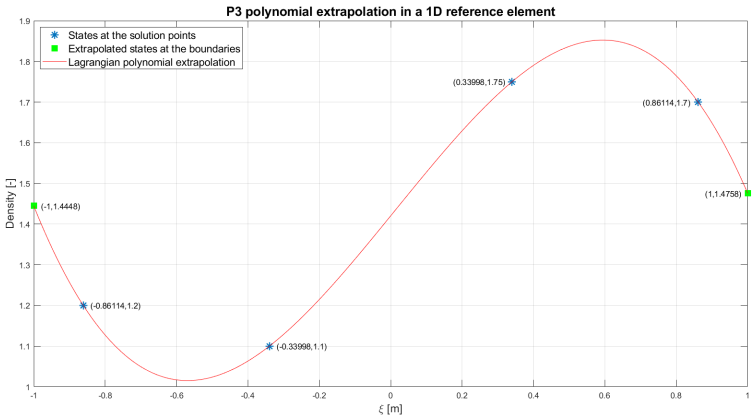

Figure 6: P3 polynomial reconstruction in a 1D reference element with the solution points located at the Gauss-Legendre quadrature points

The normalized density profile seems to indicate the presence of a physical phenomenon causing a high-gradient in a narrow region because of the abrupt density increase between the two center solution points. However, when looking at the densities extrapolated at the boundaries, the values appear to be very close to each other. The ratio of the maximum density variation inside the cell and of the density variation between the flux points is indeed equal to $\frac{1.75-1.1}{1.4758-1.4448} \approx 23.4$. This ratio, which could be infinite if the polynomial fit leads to equal state values at both flux points, shows that basing the AMR on the boundary points (or on the corner nodes in 2D) can result in a huge lack of information. The main consequence is that the mesh is much stiffer than it should as the physical gradients (which are the main driver of the refinement process) are not detected properly. Therefore, the AMR tends to be less effective and does not capture all the characteristic complexity of supersonic flow phenomena. This problem is further illustrated in Fig.7 which represents the loss of information caused by a negligence of the internal behavior of the polynomial reconstruction. Whereas the sum of the gradients between each flux and solution point (purple lines) is 1.7225, the gradient seen with the flux points only (yellow lines) is 0.031. As these values are equal to the physics-based linear spring constant through Eq.18, the mesh will be much more flexible in the purple case than in the yellow one. In order to reach this flexibility, we further developed the Vertex Spring approach and we adapted the spring analogy to take into account these internal variations by exploiting the order of the polynomial reconstruction. The novel method is denominated : *Sub-cell order-dependent spring analogy*.

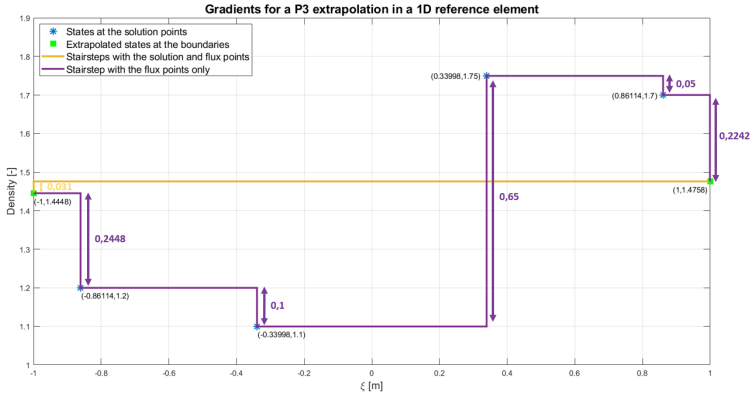

Figure 7: Gradients for a P3 polynomial reconstruction in a 1D reference element

6.2. Concept

To accomplish this goal, we decided to exploit the properties of the FR method and to build a novel sub-cell order-dependent spring analogy. It is illustrated in Fig.8 for P2. This figure depicts the edge linking two nodes and shows where the new springs are placed: between the flux points and the nodes.

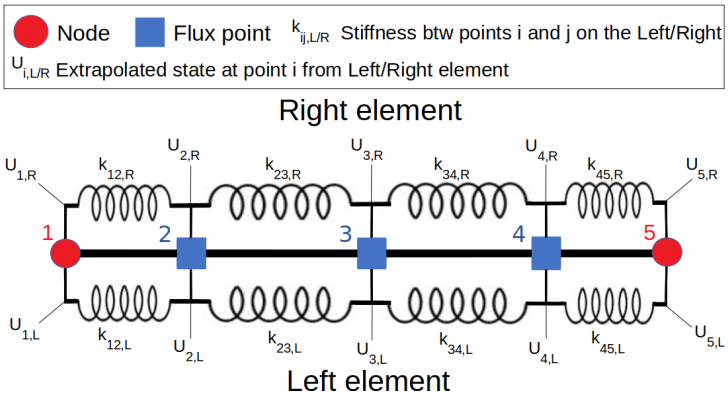

Figure 8: Sub-cell order-dependent spring analogy, face linking node 1 and 2 (red dots) with 3 flux points (blue squares) for a P2 reconstruction

Our key idea is based on considering a network of springs located at both sides of a face. Indeed, the polynomial reconstruction does not impose the continuity of the solution across the elements and, therefore, the state values are generally different at each side of the face, i.e. $U_{i,\text{Left}} \neq U_{i,\text{Right}}$. In order to take into account this discontinuity, the stiffnesses of the springs on the left side of the face are computed based on the state values obtained at the left of the face only. This assertion holds true for the stiffnesses of the springs located on the right side of the face as they depend on the right state values only.

In COOLFluiD, the flux points are not geometric entities and cannot be moved independently as the nodes can. Therefore, they cannot be included in the stiffness matrix $\mathbf{A}$ (see Eq.16). As a result, equivalent spring constants between each pair of edge-connected nodes need to be computed from the stiffnesses illustrated in Fig.8. To do so, we should note that, on each side of the face, the springs are connected end-to-end, i.e. in series. A look at the Newton's laws of motion yields that the equivalent stiffness k_{eq} of springs placed in series with a stiffness k_i should be computed as: $\frac{1}{k_{\text{eq}}} = \sum_i \frac{1}{k_i}$.

However, the physics-based linear spring constant of Eq.18 behaves inversely with respect to a mechanical spring. Indeed, when the gradient of the monitored field variable U increases, the linear stiffness increases, tends to make the mesh more flexible and thus the nodes will move more easily. As a consequence, we compute the equivalent stiffness on the left of a face $k_{\text{eq,L}}$ as $k_{\text{eq,L}} = \sum_i^N k_{i,i+1,\text{L}}$ where N equals the number of springs placed in series ($N = 4$ for a P2 extrapolation) and $k_{i,i+1,\text{L}}$ is the stiffness of the spring placed between point i and $i+1$ on the left side of the face. The same procedure is followed for the right side of the face.

The total stiffness between two edge-connected nodes can finally be computed. Although the equivalent spring of stiffness $k_{\text{eq,L}}$ and the equivalent spring of stiffness $k_{\text{eq,R}}$ appear to be parallel to each other, their contributions are summed. Because an inner edge of a quadrilateral element is shared by two elements (left and right), its total stiffness k_{eq} is computed as $k_{\text{eq}} = k_{\text{eq,L}} + k_{\text{eq,R}}$.

The main benefits of this method are its sub-cell resolution and its order dependency. Indeed, it tends to increase the flexibility of the mesh as the order of the polynomial extrapolation grows (because the number of spring contributions increases with the order). This is very important as we would like to use high-order polynomial reconstructions (to improve the accuracy per DOF) with coarse grids (to decrease the computational cost).

6.3. Extension to curved meshes

6.3.1. Motivation

When studying flows over curved bodies (such as cylinders), Q1 meshes produce unsatisfying solutions. This is due to the discrepancy between the physical boundary which is curved and its numerical description which is straight.

As illustrated in Fig.9, the Q1 mesh is not capable of fitting the curved boundary properly. As already reported in [48, 49], if the order of the numerical method increases and becomes highly different from the order of the geometrical approximation of the boundary (P4Q1 for instance), the quality of the physical results decreases significantly.

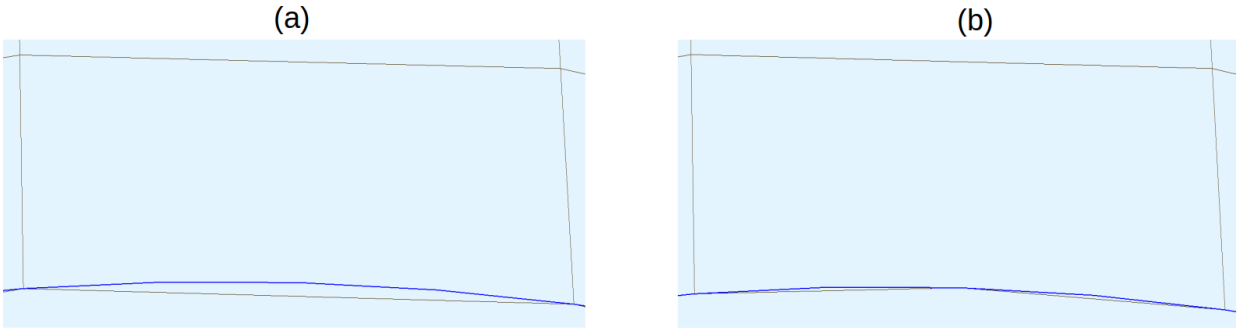

Figure 9: Approximation of the dark blue curved boundary with: (a) Q1, (b) Q2

6.3.2. Connectivity information: Q2 meshes

While Q1 meshes have 4 nodes per 2D quadrilateral (at the 4 corners of the element), Q2 meshes have 9 nodes per cell: 4 at the corners, 4 at the midpoint of each edge and finally 1 at the center of the cell. The extra-nodes at the midpoint of each face allow for a parabolic description of the edge (i.e. 2^{nd} order). However, the flux points (and the solution points) are placed at the same locations as for Q1 meshes, as shown in Fig.10 for a simplified mesh of 4 quadrilateral elements and a P3 reconstruction. In our case, the solution points remain thus at the Gauss-Legendre quadrature points. In this same Fig.10, we introduce the following nomenclature to clarify our subsequent explanations: the 4 nodes that were already present in Q1 meshes are called *corner nodes*, the 4 grid points which are located at the middle of each edge are called *middle nodes* and the node located at the center of the cell is called *center node*. Let us specify that this classification and numbering is only fictitious as of now. Indeed, these nodes are all COOLFluiD geometric entities and possess the same characteristics.

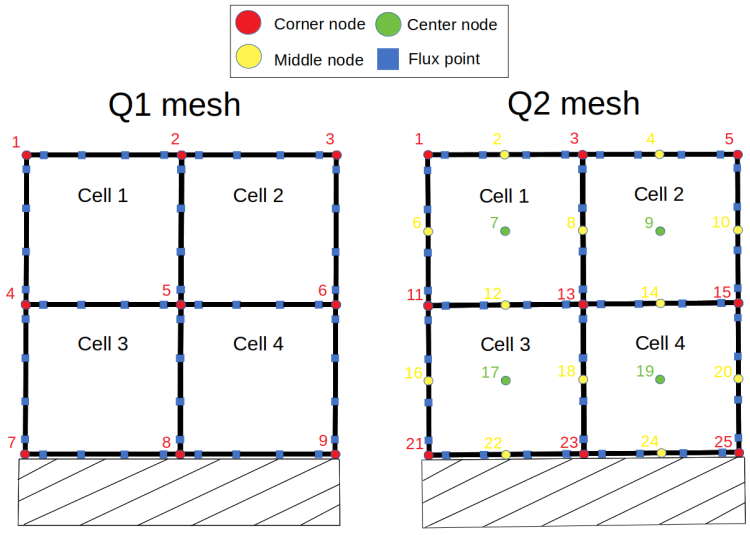

Figure 10: Location of the nodes and of the flux points for P3Q1 and P3Q2

Indeed, we would like to identify which nodes are edge-connected to which nodes in order to place our fictitious springs at the correct locations. The added complexity is that we now have a center node that is not located on one

edge and we have middle nodes that are edge-connected to two corner nodes. As an example, we illustrate the Q2 connectivity for an inner mesh element: we consider that each center node is connected to the 4 middle nodes belonging to the same element. Reciprocally, we assume that each middle node is linked to the two center nodes of the two neighboring cells to which it is connected. Finally, we consider that each middle node is linked to the two corner nodes belonging to the same face. These considerations are illustrated in Fig.11 where the gray segments show the new connectivity between corner and middle nodes.

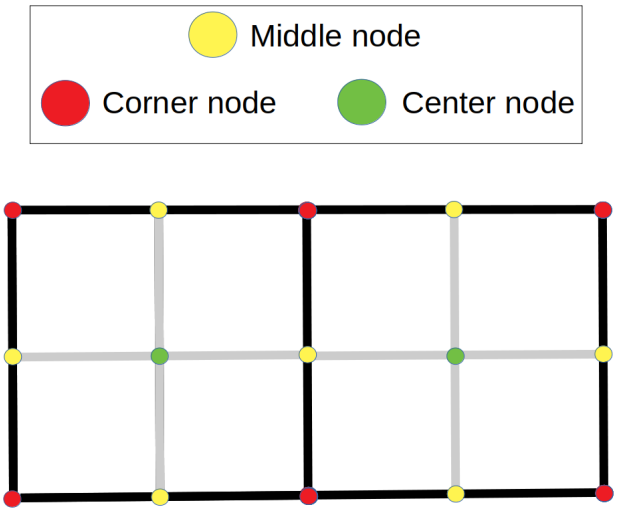

Figure 11: Nodal connectivity for Q2 grids

Now that the connectivity information are established within the setup phase and stored in multimaps, we can compute the stiffnesses between the edge-connected nodes. The same idea of order-dependent spring concept given is used for the nodes located on faces. However, we should remember that we do not want to place springs between corner nodes (as we did before in Q1) but rather between middle and corner nodes. From the set of the flux points of a face, we have to discriminate the ones that are located on the part of the face we are dealing with. An illustration of this consideration can be seen in Fig.12: if we wish to place a spring between middle node (labeled by the number 4) and corner node (labeled by the number 7), we should only take the two flux points located in between (i.e. the right flux points labeled 5 and 6).

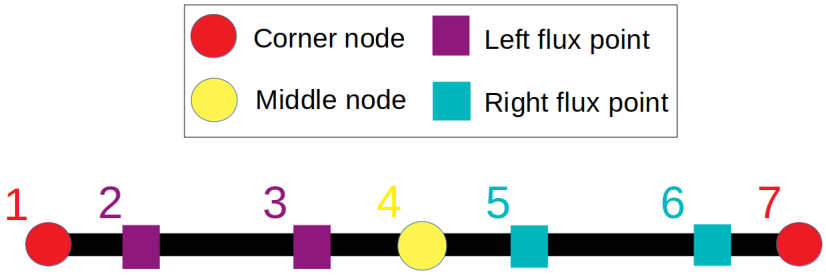

Figure 12: Discrimination of left and right flux points with respect to the middle node

To do so, the FR-AMR algorithm compares the distances between the flux points and the nodes and determine if the considered flux points are located between the two nodes of interest. Then, for all the flux points which satisfy this latter condition, a sorting algorithm is applied in order to determine how these flux points are oriented with respect to

the two edge-connected nodes. For the connection between center and middle nodes, the situation is more complex as the order-dependent spring concept is not applicable. Indeed, no flux points are located inside the cell and the solution points cannot be forced to be on the gray segments shown in Fig.11. Instead, a geometry-based spring linear stiffness is placed between each center and middle node given by Eq.20

$$k = \frac{1}{d_{ij}}, \tag{20}$$

where d_{ij} denotes the distance between the center node i and the middle node j. These extra-springs should alleviate the risk that the corner nodes exit the cells to which they belong and insures that the middle nodes do not enter inside the cells and cross the opposite edges.
Concerning the boundary treatments of the middle nodes, we distinguish 2 cases:

1. For Q2 meshes on straight lines, as shown in Fig.13-a, we deal with the middle node as it was a regular boundary node. Nevertheless, all boundary points are subject to a factor that multiplies its resulting spring stiffness, in order to rigid its movements.

2. For Q2 meshes on curved lines, as shown Fig.13-b, we propose an ad hoc solution. As the sub-cell resolution is also present on a curved boundary, we approximate the second order line between two consecutive points as a straight line and we deal with as piecewise linear boundary springs.

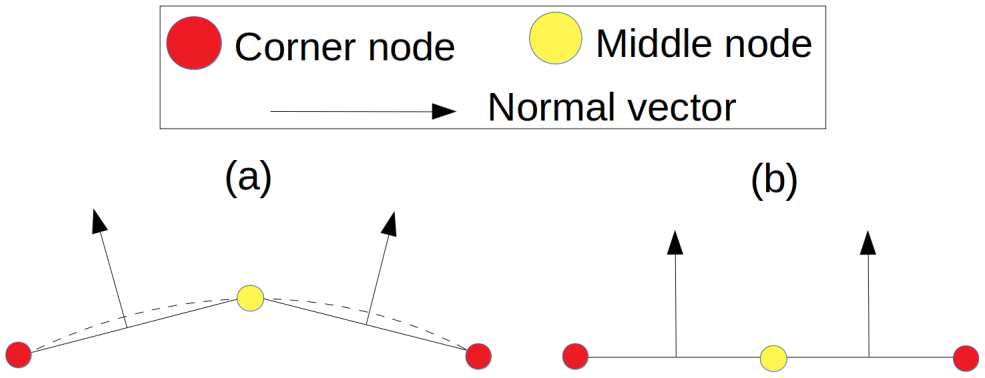

Figure 13: Simplified illustration of the Q2 boundary edge, (a) Q2 straight line, (b) Q2 curved line

7. Results : Sub-cell order-dependent spring analogy FR-AMR Simulations

7.1. Wedge channel: Q1 meshes

The density fields and final meshes obtained with the FR-AMR solver are given in Fig.14. For all orders, the mesh adapts itself properly to the phenomena encountered in the wedge channel. We can indeed see that the mesh is adequately refined around the shocks, the expansion waves and their reflections. Moreover, we can note that the boundary nodes are also attracted by the zones of high-gradient.

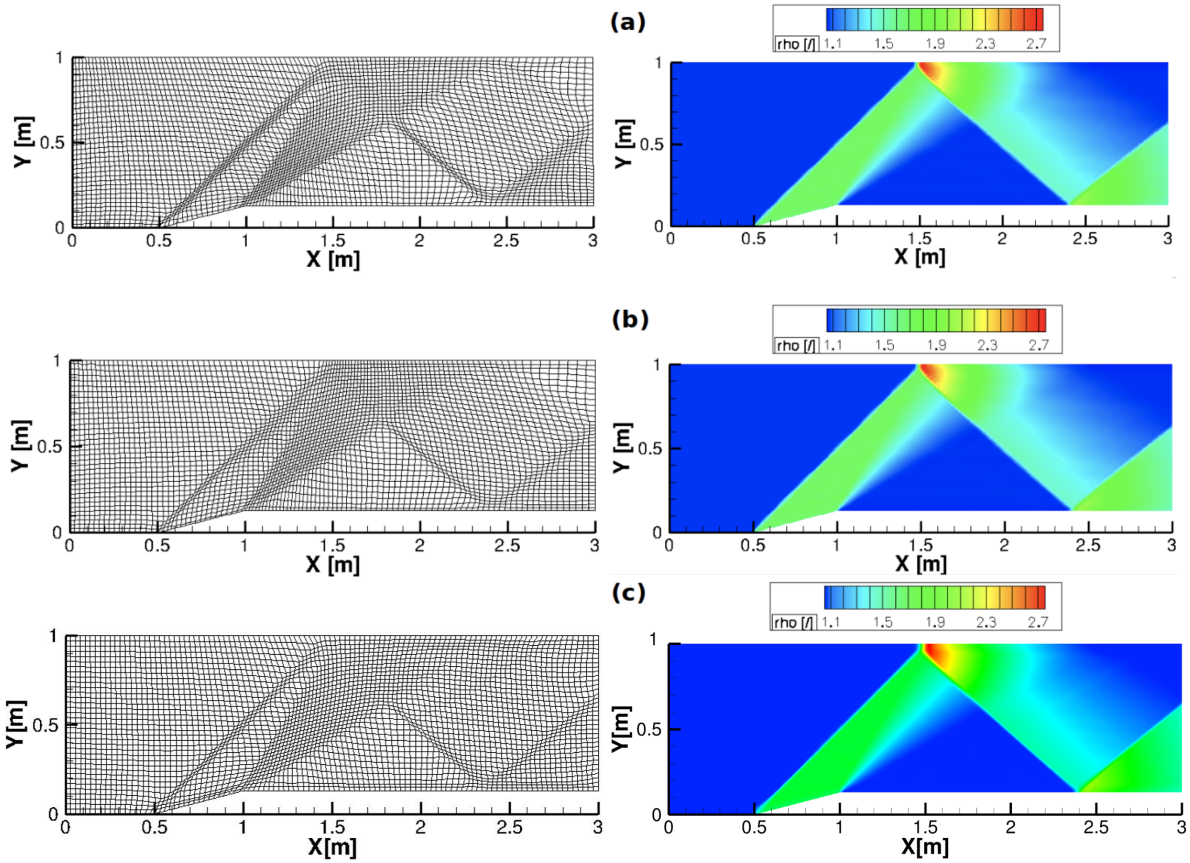

Figure 14: Final meshes and density field solution: (a) P1, (b) P2, (c) P3

In order to determine the analytical solution and shock position, we compute the density ratio across the first oblique shock. Indeed, with an upstream Mach number $M_1 = 2$, a deflection angle $\delta = \tan \frac{0.13}{0.5}$ and the oblique shock relationships [50], we can determine the shock angle θ based on Eq.21:

$$\tan \delta = 2 \cot \theta \left[\frac{M_1^2 \sin^2 \theta - 1}{M_1^2(\gamma + \cos 2\theta) + 2} \right] \tag{21}$$

This equation can be solved for θ with the function *fminsearch* of *Matlab* and gives an analytical shock angle $\theta_{an} = 44.7767°$. Tab.3 shows a statistically computed shock angle for P1 and P3 based on chosen points taken from the shock line. We observe that we are very close to the analytical solution.

θ_{an}	θ_{P1}	θ_{P3}	θ_{P1} deviation	θ_{P3} deviation
44.7767°	45.27°	45.13°	1.1%	0.789%

Table 3: Shock angles

Using the analytical shock angle θ_{an}, we determine the ratio of densities through the oblique shock [50] :

$$\frac{\rho_2}{\rho_1} = \frac{(\gamma + 1)M_1^2 \sin^2 \theta_{an}}{(\gamma - 1)M_1^2 \sin^2 \theta_{an} + 2} = 1.7047 \tag{22}$$

Let us assess the advantages brought by the use of the adaptive mesh refinement procedure and by an increase of the simulation order. To do so, the density fields are shown for P1, P2 and P3 with and without mesh adaptation in Fig.15

on a line section $Y = 0.5[m]$ within the interval $X = [0.85, 1.35][m]$. Taking into account that the shock starts from the bottom wedge (i.e. at a coordinate $\{0.5, 0\}[m]$) and develops with an angle $\theta_{an} = 44.7767°$, the jump in density on a line section $Y = 0.5[m]$ should numerically appear at a coordinate $X = 0.5 + \frac{0.5}{tan(\theta_{an})} = 1.0039[m]$.
Following the same procedure, we extract the density lines for P1, P2 and P3 solution fields for Y=0.3m.

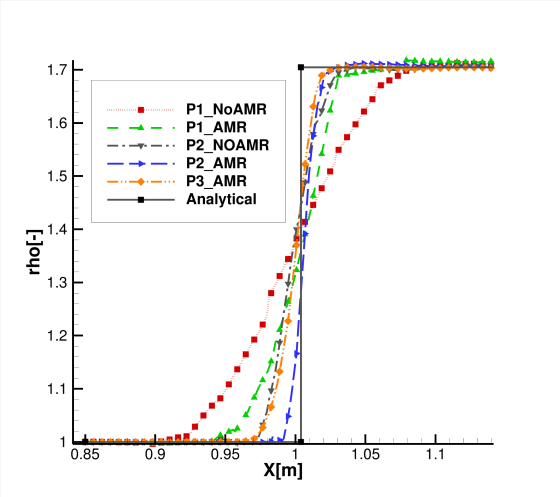

Figure 15: Comparison of the density fields obtained around the oblique shock for P1, P2 and P3 with respect to the analytical solution at Y=0.5m

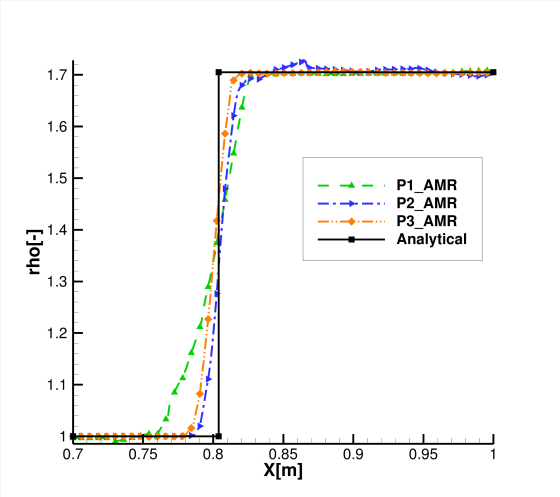

Figure 16: Comparison of the density fields obtained around the oblique shock for P1, P2 and P3 with respect to the analytical solution at Y=0.3m

The results obtained in the latter figure show a considerable advantage brought by the AMR procedure: the shock (which is characterized by the density jump) is steeper with the adapted mesh than without and is also much better resolved when increasing the simulation order.
Interestingly, we remark that for both line solution (i.e. Fig.15 and Fig.16), P2 perform the best before the shock with respect to the other depicted orders. Nevertheless, in the post shock region, P3 is the closest to the analytical solution and stabilizes perfectly on the analytical solution.

7.1.1. Comparison with literature results

In order to assess the reliability of the results obtained with the FR-AMR algorithm, a slightly modified wedge channel is considered. Its length is increased to 5 m and the angle of the wedge is reduced to 10°. However, the inlet and boundary conditions are the same as the ones of the previous subsection. Thanks to these geometry changes, the wedge is now exactly the same as the one studied by Ripley et al. in [51].
Two Q1 meshes coarser than M_f, i.e. the very fine h-adapted mesh used by the reference solution in [51], are considered to study this new wedge: a mesh of 2040 elements (called M_m) and another mesh of 845 elements (called M_c). Both grids are depicted in Fig.17. Simulations with and without AMR are performed for P1, P2 and P3 on M_m and for P4 on M_c. The parameters of these simulations are given in Tab.12 and Tab.13 in Appendix 14.

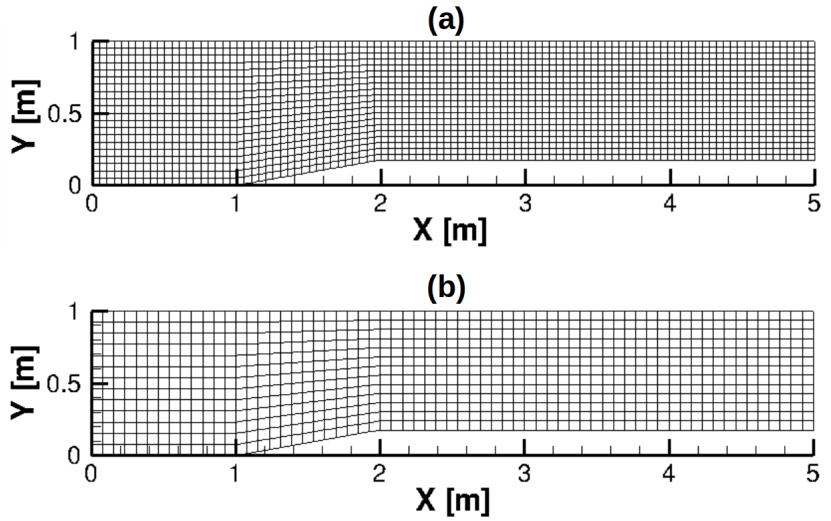

Figure 17: Initial meshes: (a) Middle mesh with 2040 cells M_m, (b) Coarse mesh with 845 cells M_c

The final meshes obtained with the r-refinement procedure for P1, P2, and P3 on M_f and for P4 on M_c are given in Fig.18 and are compared to the final mesh of 50837 cells obtained with h-refinement in the reference paper [51]. Although the mesh of the reference paper contains many more elements than the ones used here (almost 25× the number of elements of M_m and more than 60× the number of cells of M_c), the final grids obtained by the FR-AMR algorithms target the zones to refine much better. Indeed, whereas the h-refinement process leads to a refinement in regions where there are no physical gradients (such as before the first shock), our r-refinement procedure captures very well the shocks and the expansion waves along with their reflections.

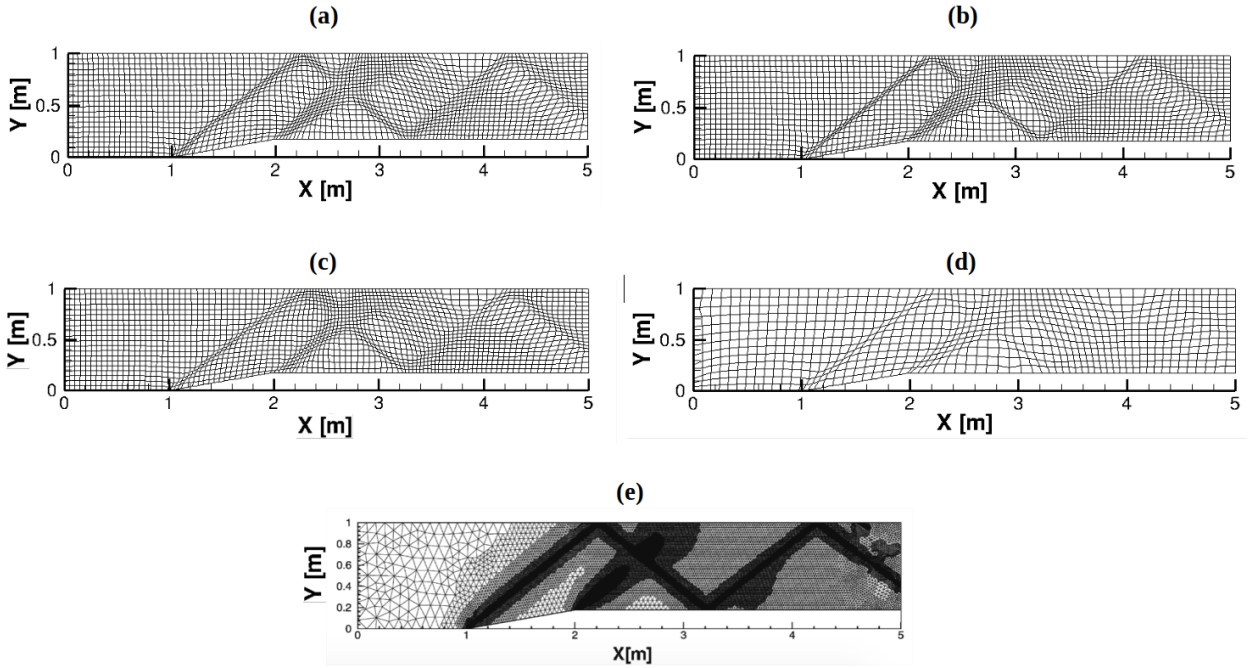

Figure 18: Final meshes: (a) M_m - P1, (b) Final M_m - P2, (c) Final M_m - P3, (d) Final M_c - P4. (e) Last level of h-adaptation - mesh with 50837 cells, taken from [51]

The normalized pressure contours obtained for P1, P2, P3 and P4 with the AMR algorithm are compared in Fig.19 with the solutions obtained without AMR. In particular, with the simulation parameters given in Tab.12, the pressure contours appear to be much smoother for P3 thanks to the AMR process. Similarly, the oblique shock is better captured for P4 with the AMR process than without (for which it is more diffusive).

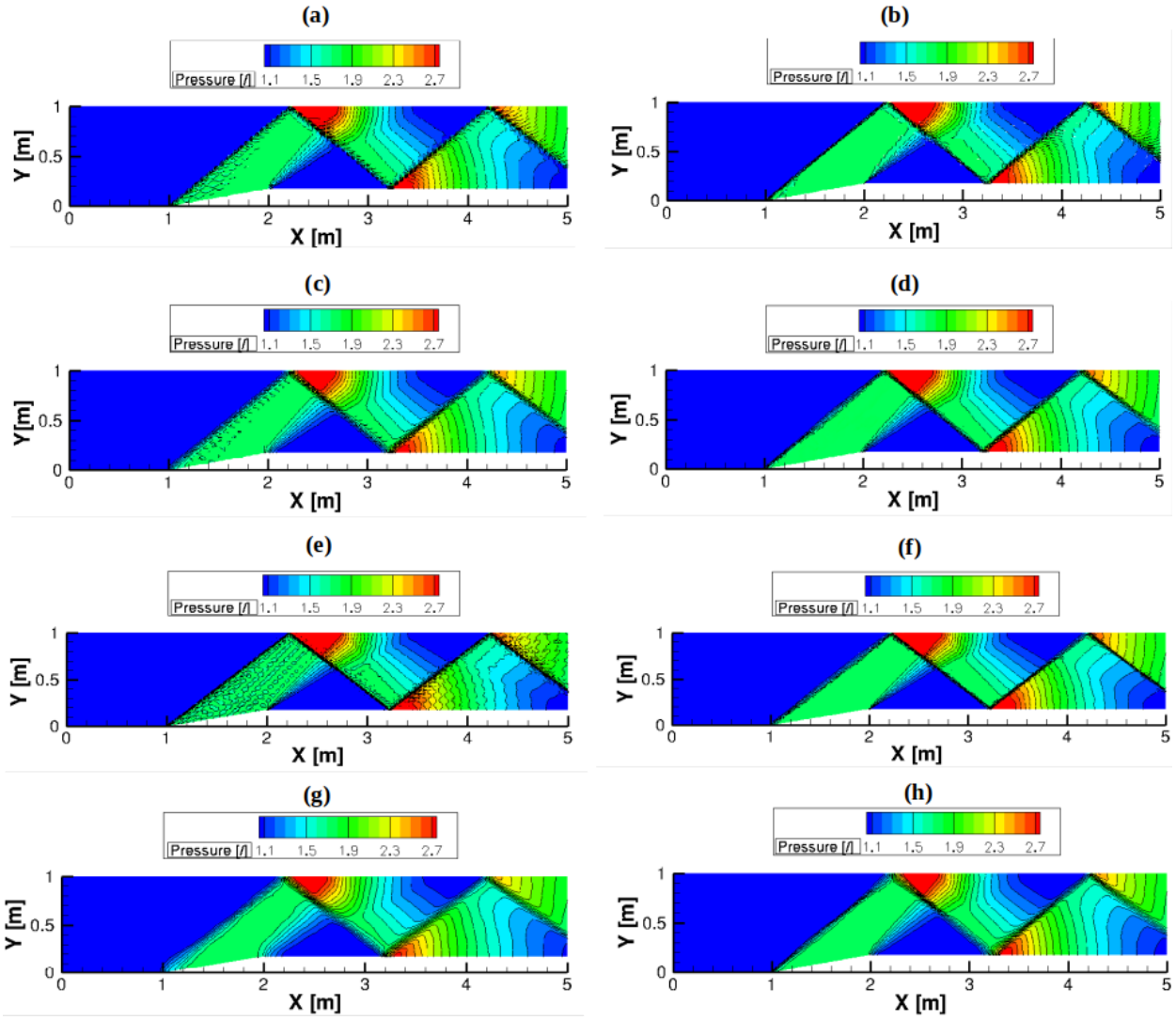

Figure 19: Pressure contours and isobar lines obtained with the FR solver. On M_m: (a) P1 no AMR, (b) P1 AMR, (c) P2 no AMR, (d) P2 AMR, (e) P3 no AMR, (f) P3 AMR. On M_c: (g) P4 no AMR, (h) P4 AMR

In Fig.20, the isobar lines obtained for P1, P2, P3 and P4 with AMR are compared with the reference ones given in [51]. The isobar plot of P3 AMR on M_m is particularly well in agreement with the reference one although the number of cells used here is much smaller. The isobar lines of P4 AMR on M_c are also very similar to the reference plot.

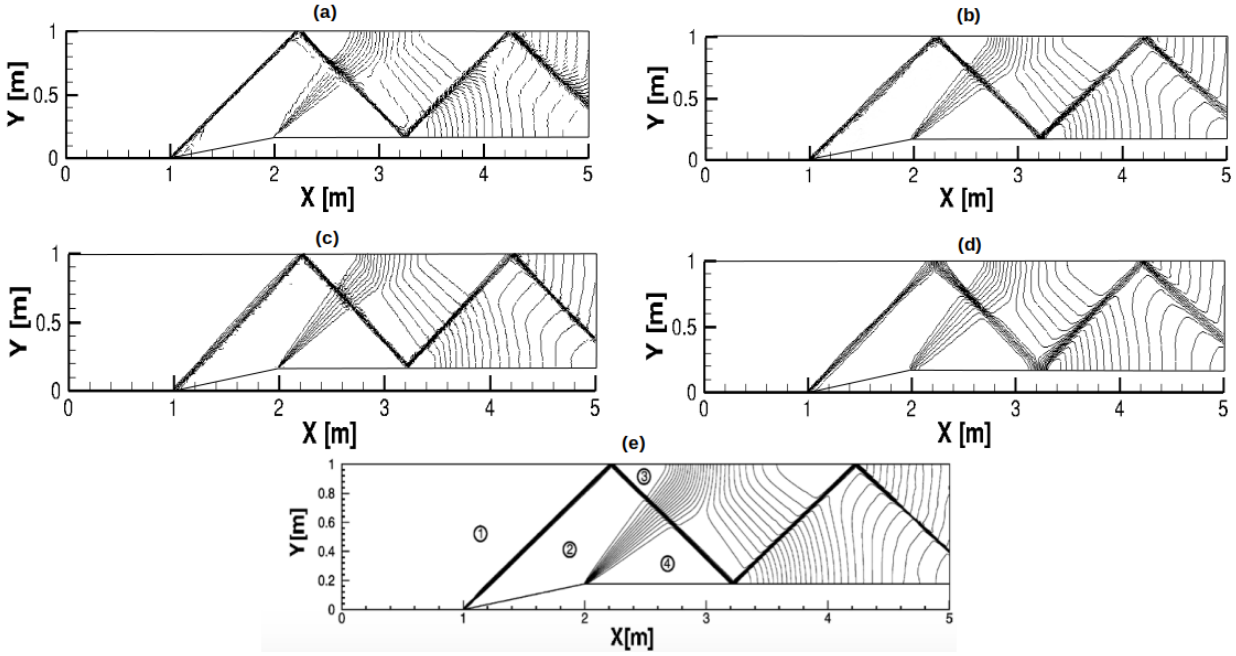

Figure 20: Isobar lines obtained with the FR solver. On M_m: (a) P1 AMR, (b) P2 AMR, (c) P3 AMR. On M_c: (d) P4 AMR. (e) Reference isobar lines - mesh with 50837 cells, taken from [51]

The Mach numbers computed at each state point defined in Fig.20 (e) with our FR-AMR algorithm are compared with the analytical solutions and with the values obtained by [51] in Tab.4. As expected, the Mach number predictions get better when the order increases with the same mesh M_m (cf. the columns of P1, P2 and P3). Another observation is that the results obtained with P4 on M_c are the closest to the analytical solutions. This is particularly interesting if we take into account the number of degrees of freedom of each simulation. For FR simulations, the number of degrees of freedom is determined by the number of solution points. In the case of 2D quadrilaterals, it is given by $N_{\text{DOF}} = N_{\text{elements}} \cdot (P+1)^2$ where P is the order of the polynomial reconstruction. For $P3$ on M_m, we have $N_{\text{DOF}} = 32640$ whereas for $P4$ on M_c, we have $N_{\text{DOF}} = 21125$. Hence, although the number of degrees of freedom is smaller for the $P4$ simulation, the results obtained are more accurate. This confirms the advantage of increasing the order of the solution over h-refinement and the better accuracy per DOF of higher-order FR over lower-order FR already highlighted in [8].

State point	**Analytical Mach [-]**	**Ref. Mach [-]**	**Ref. % difference**	**P1 Mach [-]** M_m	**P2 Mach [-]** M_m	**P3 Mach [-]** M_m	**P4 Mach [-]** M_c
1	2	2	0	2	2	2	2
2	1.65	1.639	0.7	1.6405	1.6425	1.6442	1.6456
3	1.3	1.287	1.0	1.2875	1.2905	1.2943	1.2955
4	2	1.985	0.8	1.9907	1.9909	1.9910	1.9923

Table 4: Comparison of the Mach numbers of the analytical solution, the reference results [51] and our FR-AMR simulations

In order to quantify the improvements brought by each method, we propose two assessment parameters. Firstly, we define the gain in accuracy with respect to the reference results:

$$\text{Gain} = \frac{\text{Ma [-] Sim} - \text{Ma [-] Ref}}{\text{Ma [-] Sim}} \times 100.$$

Secondly, we define the difference with respect to the analytical solution:

$$\text{Diff} = \frac{\text{Ma [-] Analytical } - \text{Ma [-] Sim}}{\text{Ma [-] Analytical}} \times 100.$$

These values are given in Table 5 for the P4 AMR simulations performed on M_c, where the relative difference with respect to the analytical solution appears to considerably decrease for the state point 4. We can observe that the gain percentage is always positive leading to an increased accuracy with respect to the reference solution while using 60× coarser mesh and up to 2.4× lower DOF.

State point	**Analytical Mach [-]**	**Ref. Mach [-]**	**Ref. % difference**	**P4 Mach [-] M_c**	**Diff. % with P4**	**Gain % with P4**
1	2	2	0	2	0	0
2	1.65	1.639	0.7	1.6456	0.398	0.269
3	1.3	1.287	1.0	1.2955	0.658	0.344
4	2	1.985	0.8	1.9923	0.366	0.386

Table 5: Improvements obtained with P4 AMR on M_c

7.1.2. Convergence

Figure 21 illustrates the convergence history of the different FR-AMR simulations over the long wedge. Every simulation shows a similar pattern: the oscillations in the residual during the AMR phase are followed by a drop within the convergence behavior. This illustrates particularly well the choice that has to be made in the value of the relaxation factor ω. Indeed, it appears that the simulation does not converge while the AMR process is activated. Hence, a trade-off has to be found: if ω is too low, the number of required AMR iterations has to increase and thus convergence is delayed whereas, if ω is too high, the number of AMR iterations can decrease but risks of instabilities appear and a crash of the simulation becomes very likely. Thus, as already discussed in [15], a value of 0.05 is generally chosen.

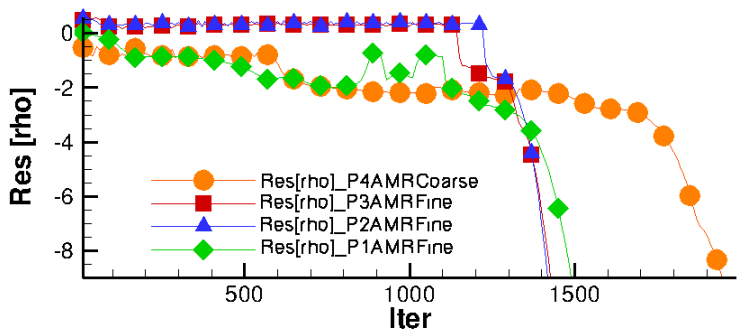

Figure 21: Convergence history for the FR-AMR simulations

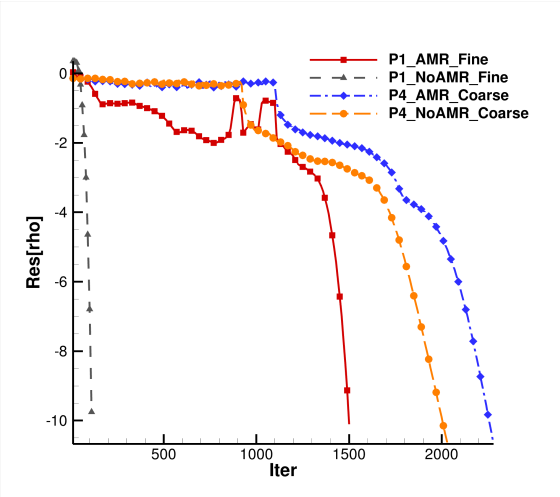

Figure 22: Convergence history for P1 and P4 with and without r-AMR over the long wedge

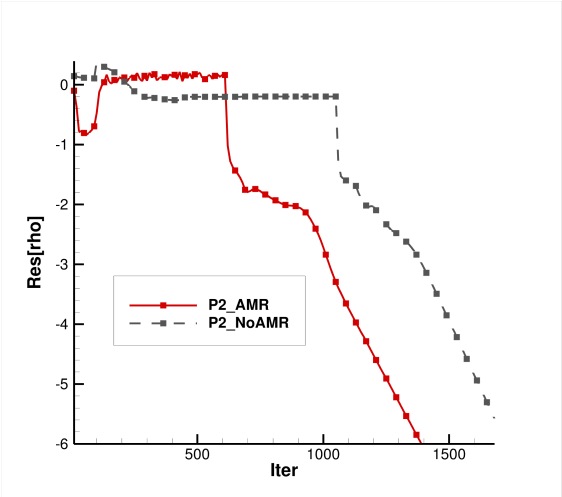

Figure 23: Convergence history for P2 with and without r-AMR over the first wedge

Figure 22 shows the convergence history of P1 and P4 simulations with and without r-AMR on the long wedge. We can observe in general that the rate of convergence of P1 simulations is faster than the P3 one. Also, as the order of the FR solver increases, the gap between the convergence history with and without r-AMR tends to decrease. Nevertheless, we can observe in the Fig.23 that the rate convergence of P2 with r-AMR is faster than the one with a fixed mesh and same order. In fact, Fig.23 shows a quasi-straight line in the convergence history of P2 without r-AMR. The residuals stagnate resulting from the continuously increasing artificial viscosity (AV) within a stable shock. This procedure should stop and freeze the AV value, as soon as the shock is stable. As a result, the residuals drop and we obtain a faster convergence rate. Actually, we are more interested in quantifying the penalty in the computational cost due to the AMR procedure. To do so, we compute the wall time per iteration by dividing the total time of the simulation by the number of iterations, both with and without r-refinement. To obtain the extra cost caused by the r-refinement procedure, let us introduce the factor C such that :

$$C = \frac{\text{Wall time per iteration with r-AMR}}{\text{Wall time per iteration without r-AMR}} \tag{23}$$

The values of C for the simulations performed in P1, P2, P3 and P4 on the wedge channel flow are respectively equal to 1.055, 1.069, 1.047 and 1.012. This factor is close to 1 for each simulation, leading to the conclusion that the relative computational cost per iteration of our r-refinement algorithm with respect to the overall time per iteration of the FR simulation is small. Indeed, the maximum computational time penalty is of 6.9% for P2.
Let us note that, after P2, the extra cost of our r-AMR-FR algorithm decreases with the order. This can be explained by the fact that the cost for solving the pseudo-elastic system vanishes and becomes negligible as the polynomial order increases with respect to the FR computations on the wedge channel. For more information about the performance and the scalability with the simulation order of the FR solver, we refer the reader to [8]. As the parallelization of the FR-AMR algorithm is handled by the HPC infrastructure of the COOLFluiD platform and has not been modified throughout this work, we encourage the reader to consult [28] for more details about the implementation.

7.2. Circular bump : Q2 meshes

The geometry of the 2D inviscid bump is shown in Fig.24. The bump is circular with a thickness equal to 4% of the channel width and is described by a second-order polynomial (Q2 mesh). The bump has been generated by creating a circle arc whose center coordinates are $\{1.5, -3.105\}$ [m] and radius is 3.145 m. The imposed flow conditions are given in Tab.6 and correspond to an inlet Mach number $M = 1.4$. The types of boundary conditions are the same as for the wedge channel flow.

M [-]	ρ [-]	ρu [-]	ρv [-]	ρE [-]
1.4	1.0	1.656502339	0.0	3.872

Table 6: 2D bump inlet flow characteristics

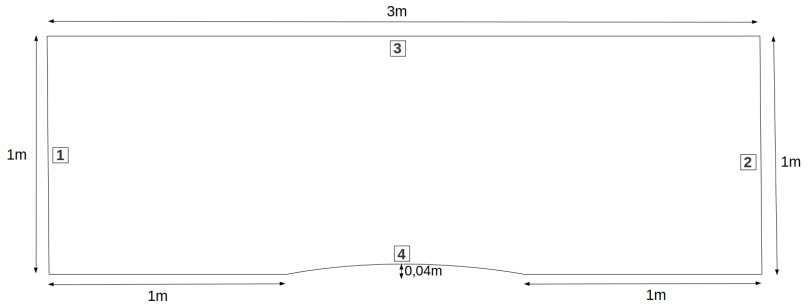

Figure 24: 2D bump geometry

The two meshes used by Premasuthan et al. in [17] are considered here: a fine mesh of 4800 cells called M_f and a coarse mesh of 1200 cells called M_c, which makes the number of nodes of M_c half the number of nodes of M_f in both the x- and y- directions. While M_f describes the bump surface with 40 cells in the tangential direction, M_c only uses 20 elements for this purpose. These two meshes are depicted in Fig.25.

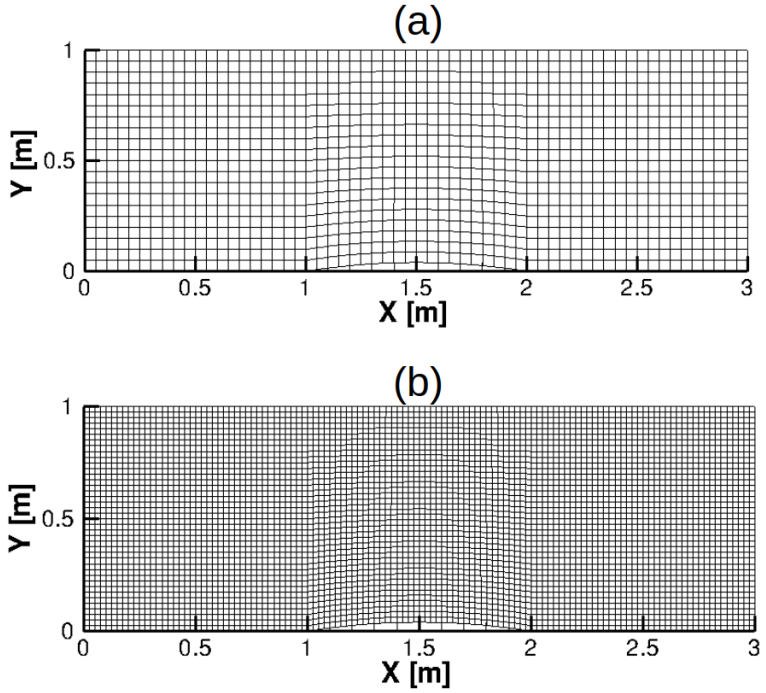

Figure 25: 2D bump initial meshes: (a) Coarse mesh with 1200 cells M_c, (b) Fine mesh with 4800 cells M_f

Simulations with P2 AMR and P3 AMR were performed on both meshes. The final meshes and pressure contours obtained with the simulation parameters given in Tab.14 and Tab.15 in Appendix 15 are shown for M_c in Fig.26 and for M_f in Fig.27.

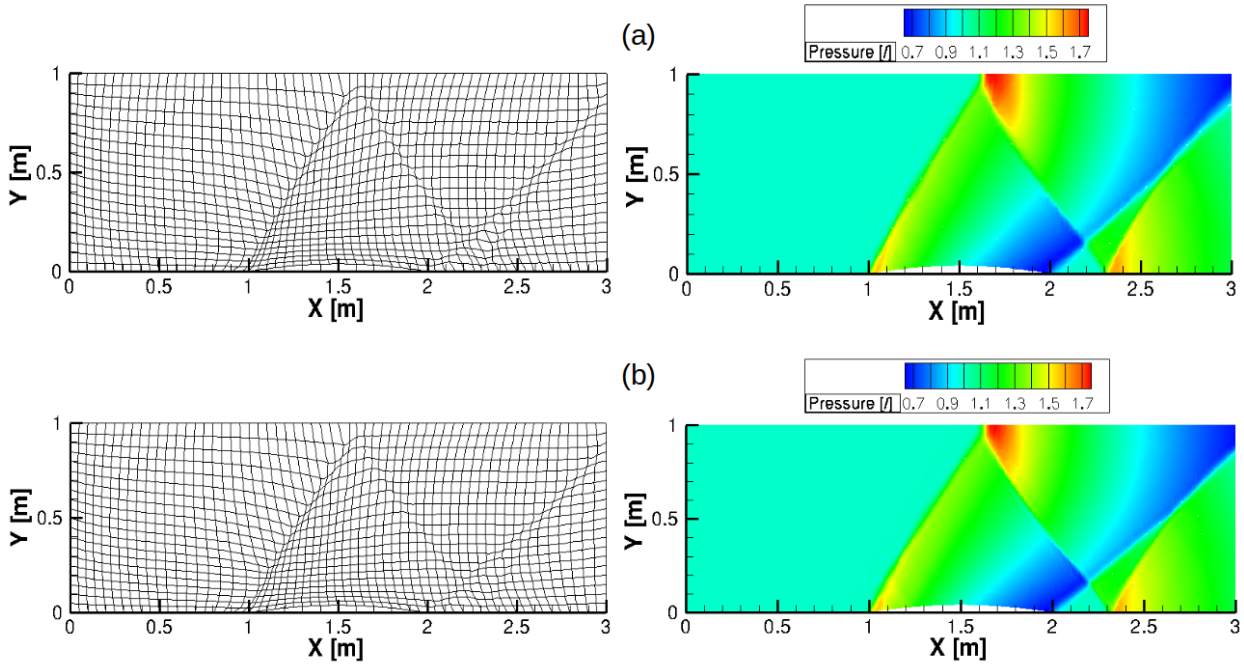

Figure 26: Final meshes and pressure field on the coarse mesh M_c: (a) P2 AMR, (b) P3 AMR

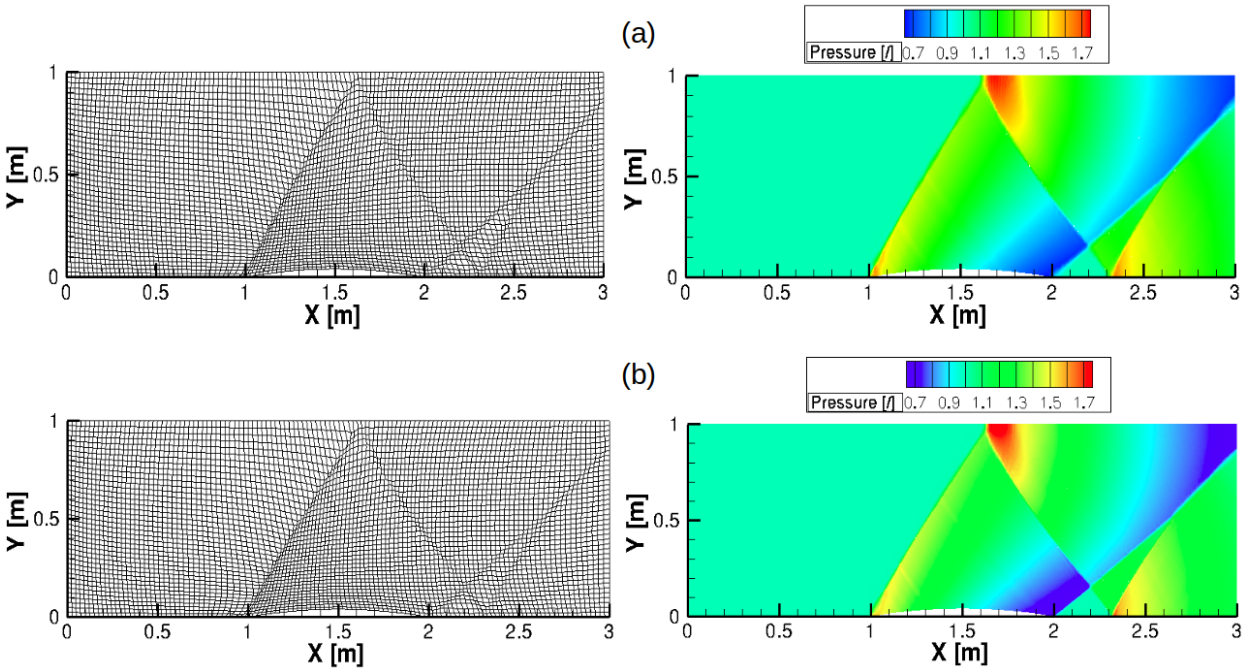

Figure 27: Final meshes and pressure field on the fine mesh M_f: (a) P2 AMR, (b) P3 AMR

The obtained pressure fields match the ones already reported in [17]: the oblique shock created at the leading edge of the bump is reflected on both walls whilst the expansion waves are created all over the bump length. Moreover, the final mesh confirms the proper functioning of the novel AMR algorithm for Q2 meshes as the mesh resolution is

larger around the zones of high-gradient. The region after the bump, which displays complex interactions between reflections of the oblique shock and expansion waves, is also correctly refined. Fig.28 shows the pressure along the line of equation $Y = 0.4m$ for our simulations (P2 and P3 on M_c and M_f) as well as for the reference solution (Spectral Difference with artificial viscosity taken from [52]). The latter figure shows that our solutions agree with the reference one and the shock are narrowly resolved without noteworthy Gibbs phenomena.
One of the main advantages of the FR-AMR is that our method (at least for P3) was able to detect the last pressure jump that lies close to the joining point of the two oblique shocks on the coarse mesh as shown in Fig.29. Fig.30 shows already the improved capabilities of the FR-AMR with the P2 solution over the fine mesh as the pressure jumps are narrower and better resolved. Similar conclusions can be drawn out from Fig.31 – 33.

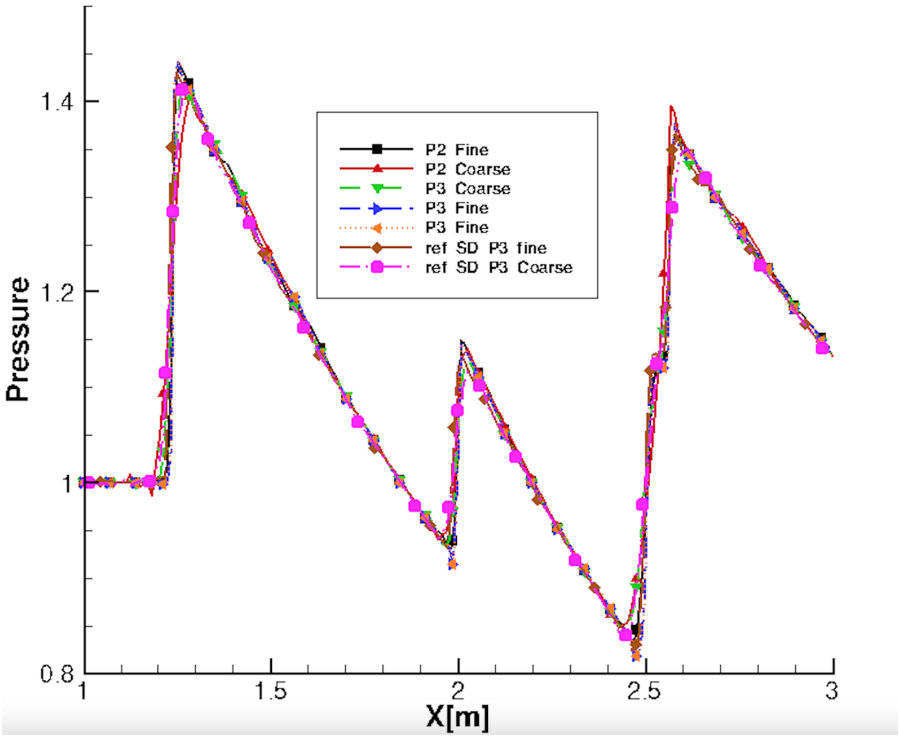

Figure 28: Pressure along the line $Y = 0.4m$ for P2 and P3 on the coarse and the fine mesh as the P3 SD reference solution on the same meshes

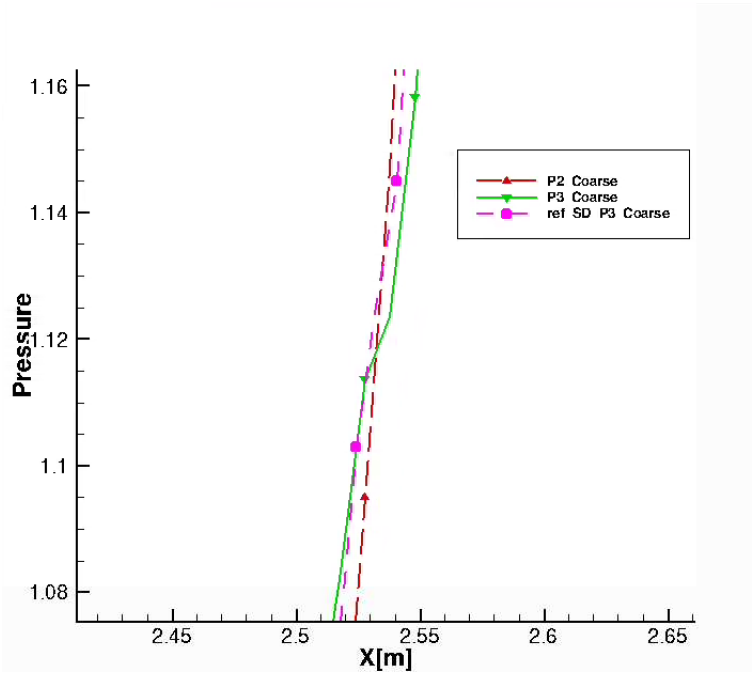

Figure 29: Zoom around the joining point of the two oblique shocks

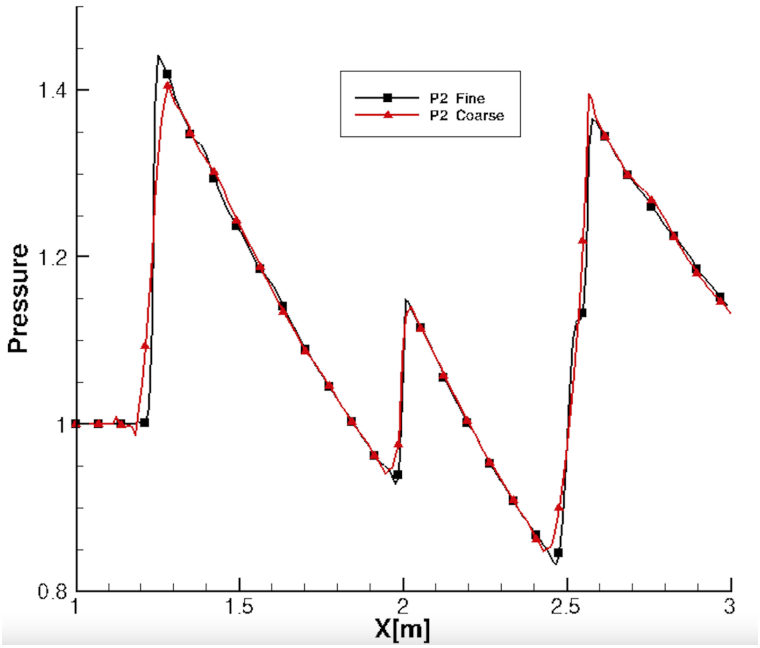

Figure 30: Line pressure for P2 on the coarse and fine meshes

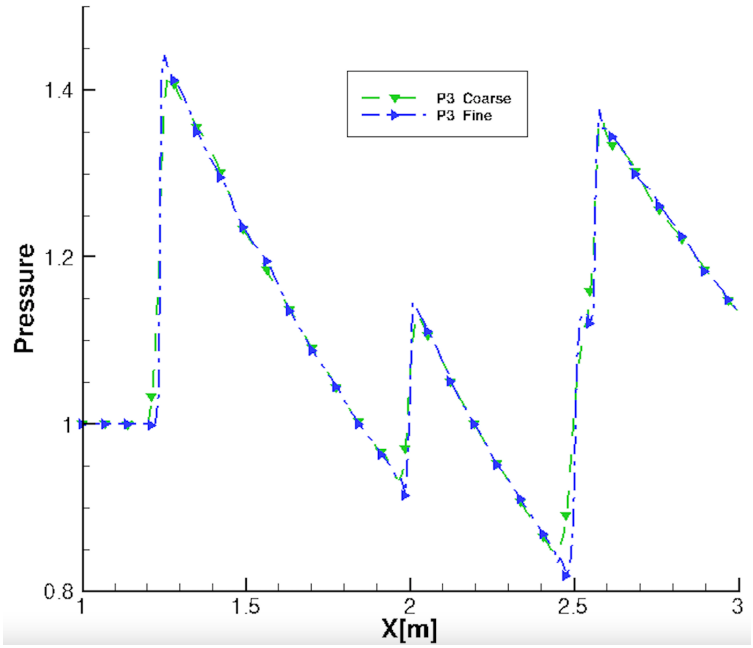

Figure 31: Line pressure for P3 on the coarse and fine meshes

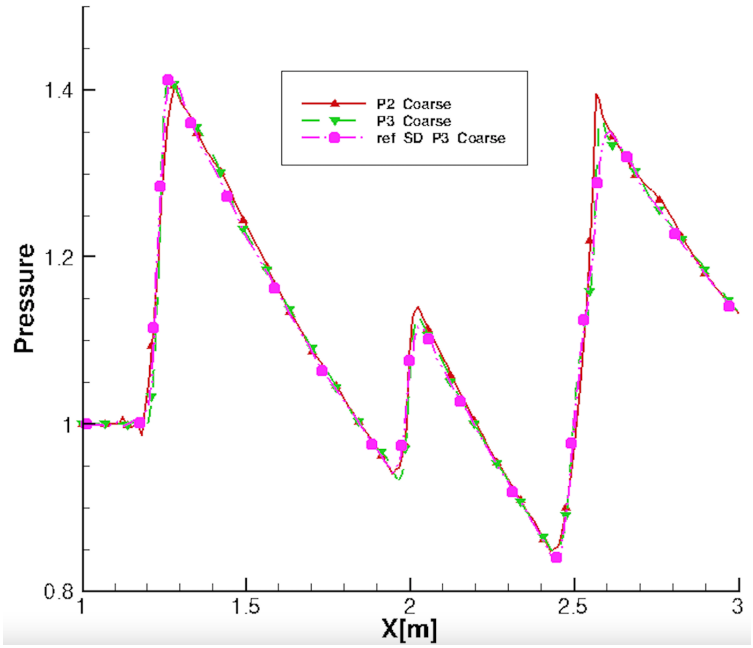

Figure 32: Line pressure for P2, P3 and reference SD P3 on the coarse mesh

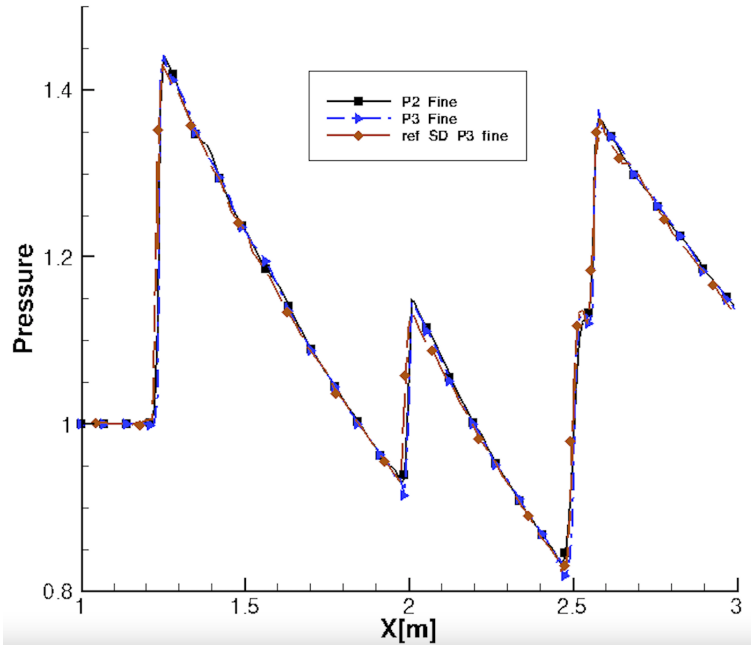

Figure 33: Line pressure for P2, P3 and reference SD P3 on the fine mesh

8. Conclusion

A novel r-AMR algorithm has been developed and, in combination with an existing Flux Reconstruction CFD solver, successfully applied to 2D benchmark test cases. The proposed sub-cell order-dependent spring analogy shows its promising results with respect to the vertex spring method and literature results, relying upon FV with h-refinement and SD with artificial viscosity. In addition, as our previous work in [15], the FR-AMR is developed as a standalone module that can be coupled, with minor efforts, to any high-order CFD/plasmas solvers. Finally, this work contributed to the development of the first in its kind spring-based r-AMR for a high-order FR framework.

9. Acknowledgment

The research of Firas Ben Ameur and Ray Vandenhoeck is supported by SB PhD fellowship 1SC3519N and 1S19918N, respectively, of the FWO.

References

[1] P. E. Vincent, A. Jameson, Facilitating the adoption of unstructured high-order methods amongst a wider community of fluid dynamicists, Mathematical Modelling of Natural Phenomena 6 (3) (2011) 97–140. doi:10.1051/mmnp/20116305.

[2] H. T. Huynh, Z. J. Wang, P. E. Vincent, High-order methods for computational fluid dynamics: A brief review of compact differential formulations on unstructured grids, Computers & Fluids 98 (2014) 209–220. doi:10.1016/j.compfluid.2013.12.007.

[3] H. T. Huynh, A flux reconstruction approach to high-order schemes including discontinuous galerkin methods, 18th AIAA Computational Fluid Dynamics Conferencedoi:10.2514/6.2007-4079.

[4] P. E. Vincent, F. D. Witherden, A. M. Farrington, G. Ntemos, B. C. Vermeire, J. S. Park, A. S. Iyer, Pyfr: Next-generation high-order computational fluid dynamics on many-core hardware (invited), 22nd AIAA Computational Fluid Dynamics Conferencedoi:10.2514/6.2015-3050.

[5] F. D. Witherden, B. C. Vermeire, P. E. Vincent, Heterogeneous computing on mixed unstructured grids with pyfr, Computers & Fluids 120 (2015) 173–186. doi:10.1016/j.compfluid.2015.07.016.

[6] R. Abgrall, M. Ricchiuto, High-order methods for cfd, Encyclopedia of Computational Mechanics Second Edition (2017) 1–54doi:10.1002/9781119176817.ecm2112.

[7] V. K. Chakravarthy, D. Chakraborty, Damping numerical oscillations in hybrid solvers through detection of gibbs phenomenon, International Journal for Numerical Methods in Fluids 84 (12) (2017) 699–714. doi:10.1002/fld.4367.
[8] R. Vandenhoeck, A. Lani, Implicit high-order flux reconstruction solver for high-speed compressible flows, Computer Physics Communications 242 (2019) 1–24. doi:10.1016/j.cpc.2019.04.015.
[9] X. Zhang, C.-W. Shu, On maximum-principle-satisfying high order schemes for scalar conservation laws, Journal of Computational Physics 229 (9) (2010) 3091–3120. doi:10.1016/j.jcp.2009.12.030.
[10] X. Zhang, C.-W. Shu, On positivity-preserving high order discontinuous galerkin schemes for compressible euler equations on rectangular meshes, Journal of Computational Physics 229 (23) (2010) 8918–8934. doi:10.1016/j.jcp.2010.08.016.
[11] X. Zhang, C.-W. Shu, Positivity-preserving high order discontinuous galerkin schemes for compressible euler equations with source terms, Journal of Computational Physics 230 (4) (2011) 1238–1248. doi:10.1016/j.jcp.2010.10.036.
[12] F. Ben Ameur, R. Vandenhoeck, A. Lani, High-order flux reconstruction scheme for thermo-chemical nonequilibrium high-speed flows, AIAA Scitech 2019 Forum, AIAA 2019-1391 doi:10.2514/6.2019-1391.
[13] T. J. Baker, Mesh adaptation strategies for problems in fluid dynamics, Finite Elements in Analysis and Design 25 (3-4) (1997) 243–273. doi:10.1016/s0168-874x(96)00032-7.
[14] D. S. McRae, r-refinement grid adaptation algorithms and issues, Computer Methods in Applied Mechanics and Engineering 189 (4) (2000) 1161–1182. doi:10.1016/s0045-7825(99)00372-2.
[15] F. Ben Ameur, A. Lani, r-adaptive algorithms for high-speed flows and plasma simulations, Computer Physics Communications 261 (2021) 107700. doi:https://doi.org/10.1016/j.cpc.2020.107700.
URL https://www.sciencedirect.com/science/article/pii/S0010465520303441
[16] M. Cho, S. Jun, r-adaptive mesh generation for shell finite element analysis, Journal of Computational Physics 199 (1) (2004) 291–316. doi:10.1016/j.jcp.2004.02.007.
[17] S. Premasuthan, C. Liang, A. Jameson, Computation of flows with shocks using spectral difference scheme with artificial viscosity, 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Expositiondoi:10.2514/6.2010-1449.
[18] Y. Li, S. Premasuthan, A. Jameson, Comparison of adaptive h and p refinements for spectral difference methods, 40th Fluid Dynamics Conference and Exhibitdoi:10.2514/6.2010-4435.
[19] M. Woopen, A. Balan, G. May, J. Schütz, A comparison of hybridized and standard dg methods for target-based hp-adaptive simulation of compressible flow, Computers & Fluids 98 (2014) 3–16. doi:10.1016/j.compfluid.2014.03.023.
[20] A. Balan, M. Woopen, G. May, Adjoint-based hp -adaptivity on anisotropic meshes for high-order compressible flow simulations, Computers & Fluids 139 (2016) 47–67. doi:10.1016/j.compfluid.2016.03.029.
[21] D. Ekelschot, D. Moxey, S. Sherwin, J. Peiró, A p-adaptation method for compressible flow problems using a goal-based error indicator, Computers & Structures 181 (2017) 55–69. doi:10.1016/j.compstruc.2016.03.004.
[22] L. Shi, Z. J. Wang, Adjoint-based error estimation and mesh adaptation for the correction procedure via reconstruction method, Journal of Computational Physics 295 (2015) 261–284. doi:10.1016/j.jcp.2015.04.011.
[23] J. Yang, B. Zhang, C. Liang, Y. Rong, A high-order flux reconstruction method with adaptive mesh refinement and artificial diffusivity on unstructured moving/deforming mesh for shock capturing, Computers & Fluids 139 (2016) 17–35. doi:10.1016/j.compfluid.2016.03.025.
[24] X. Zhang, C. Liang, J. Yang, A high-order flux reconstruction/correction procedure via reconstruction method for shock capturing with space-time extension time stepping and adaptive mesh refinement, 55th AIAA Aerospace Sciences Meetingdoi:10.2514/6.2017-0519.
[25] J. Yang, C. Liang, A high-order flux reconstruction adaptive mesh refinement method for magnetohydrodynamics on unstructured grids, International Journal for Numerical Methods in Fluids 86 (3) (2017) 231–253. doi:10.1002/fld.4415.
[26] F. Ben Ameur, A. Lani, Physics-based mesh fitting algorithms for hypersonic flows simulations, AIAA Scitech 2019 Forum, AIAA 2019-1997 doi:10.2514/6.2019-1997.
[27] D. Kimpe, A. Lani, T. Quintino, S. Poedts, S. Vandewalle, The coolfluid parallel architecture, in: B. Di Martino, D. Kranzlmüller, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 520–527.
[28] A. Lani, N. Villedie, K. Bensassi, L. Koloszar, M. Vymazal, S. M. Yalim, M. Panesi, Coolfluid: an open computational platform for multi-physics simulation and research, in: 21st AIAA Computational Fluid Dynamics Conference, 2013, p. 2589.
[29] J. G. Mena, R. Pepe, A. Lani, H. Deconinck, Assessment of heat flux prediction capabilities of residual distribution method: Application to atmospheric entry problems, Communications in Computational Physics 17 (3) (2015) 682–702.
[30] M. Panesi, A. Lani, T. Magin, J. Molnar, O. Chazot, H. Deconinck, Numerical investigation of the non equilibrium shock-layer around the expert vehicle, in: 18th AIAA Computational Fluid Dynamics Conference, 2007, p. 4317.
[31] P. D. Santos, A. Lani, An object-oriented implementation of a parallel monte carlo code for radiation transport, Computer Physics Communications 202 (2016) 233–261.
[32] W. Zhang, A. Lani, M. Panesi, Analysis of non-equilibrium phenomena in inductively coupled plasma generators, Physics of Plasmas 23 (7) (2016) 073512.
[33] A. A. Laguna, N. Ozak, A. Lani, H. Deconinck, S. Poedts, Fully-implicit finite volume method for the ideal two-fluid plasma model, Computer Physics Communications 231 (2018) 31–44.
[34] A. A. Laguna, A. Lani, N. N. Mansour, H. Deconinck, S. Poedts, Effect of radiation on chromospheric magnetic reconnection: reactive and collisional multi-fluid simulations, The Astrophysical Journal 842 (2) (2017) 117.
[35] Y. G. Maneva, A. A. Laguna, A. Lani, S. Poedts, Multi-fluid modeling of magnetosonic wave propagation in the solar chromosphere: effects of impact ionization and radiative recombination, The Astrophysical Journal 836 (2) (2017) 197.
[36] A. A. Laguna, A. Lani, H. Deconinck, N. Mansour, S. Poedts, A fully-implicit finite-volume method for multi-fluid reactive and collisional magnetized plasmas on unstructured meshes, Journal of Computational Physics 318 (2016) 252 – 276. doi:https://doi.org/10.1016/j.jcp.2016.04.058.
URL http://www.sciencedirect.com/science/article/pii/S0021999116301139

[37] A. Lani, M. S. Yalim, S. Poedts, A gpu-enabled finite volume solver for global magnetospheric simulations on unstructured grids, Computer Physics Communications 185 (10) (2014) 2538–2557.
[38] H. T. Huynh, High-order methods including discontinuous galerkin by reconstructions on triangular meshes, 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Expositiondoi:10.2514/6.2011-44.
[39] P. Castonguay, D. Williams, P. Vincent, M. López, A. Jameson, On the development of a high-order, multi-gpu enabled, compressible viscous flow solver for mixed unstructured grids, 20th AIAA Computational Fluid Dynamics Conferencedoi:10.2514/6.2011-3229.
[40] P. Castonguay, P. E. Vincent, A. Jameson, A new class of high-order energy stable flux reconstruction schemes for triangular elements, Journal of Scientific Computing 51 (1) (2011) 224–256. doi:10.1007/s10915-011-9505-3.
[41] H. Huynh, A flux reconstruction approach to high-order schemes including discontinuous galerkin methods, AIAA paper 4079 (2007) 2007.
[42] S. Abhyankar, J. Brown, E. M. Constantinescu, D. Ghosh, B. F. Smith, H. Zhang, Petsc/ts: A modern scalable ode/dae solver library, arXiv preprint arXiv:1806.01437.
[43] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc Web page, http://www.mcs.anl.gov/petsc (2018).
URL http://www.mcs.anl.gov/petsc
[44] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.9, Argonne National Laboratory (2018).
URL http://www.mcs.anl.gov/petsc
[45] S. Balay, W. D. Gropp, L. C. McInnes, B. F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A. M. Bruaset, H. P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.
[46] R. Jain, I. Chlamtac., The P2 Algorithm for Dynamic Calculation of Quantities and Histograms Without Storing Observations, Eastern Research Laboratory, Digital Equipment Corporation, 1985.
[47] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible navier–stokes equations, Journal of Computational Physics 131 (2) (1997) 267–279. doi:10.1006/jcph.1996.5572.
[48] L. Krivodonova, M. Berger, High-order accurate implementation of solid wall boundary conditions in curved geometries, Journal of Computational Physics 211 (2) (2006) 492–512. doi:10.1016/j.jcp.2005.05.029.
[49] X. Luo, M. Shephard, J.-F. Remacle, Influence of geometric approximation on the accuracy of higher order methods.
[50] J. E. A. John, T. G. Keith, Gas dynamics, 3rd edition.
URL https://www.pearson.com/us/higher-education/program/John-Gas-Dynamics-3rd-Edition/PGM17109.html
[51] R. C. Ripley, F. S. Lien, M. M. Yovanovich, Adaptive mesh refinement of supersonic channel flows on unstructured meshes, Int. J. Comput. Fluid Dyn 18 (2) (2004) 189–198.
[52] S. Premasuthan, C. Liang, A. Jameson, Computation of flows with shocks using the spectral difference method with artificial viscosity, i: Basic formulation and application, Computers Fluids 98 (2014) 111 – 121. doi:https://doi.org/10.1016/j.compfluid.2013.12.013.
URL http://www.sciencedirect.com/science/article/pii/S0045793013004933
[53] A. Papen, R. Vandenhoeck, Development of an Implicit High-Order Flux Reconstruction Solver for Compressible Flows on Unstructured Grids, Master's thesis, KU Leuven (2017).
[54] Y.-Y. Tsui, T.-C. Wu, A pressure-based unstructured-grid algorithm using high-resolution schemes for all-speed flows, Numerical Heat Transfer, Part B: Fundamentals 53 (1) (2008) 75–96.
[55] J. Fürst, On the implicit density based openfoam solver for turbulent compressible flows, in: EPJ Web of Conferences, Vol. 143, EDP Sciences, 2017, p. 02027.

10. Implementation of the FR-AMR solver

This appendix will give a concise overview about the implementation of the FR-AMR solver within the COOLFluiD platform. The implementation methodology mainly follows the steps of our previous work in [15] and the main differences are presented below.

10.1. General structure

Our r-AMR-FR algorithms interface the COOLFluiD's FR solver based on some AMR-controlling parameters, specified in the input file, the mesh topology and the solution polynomial

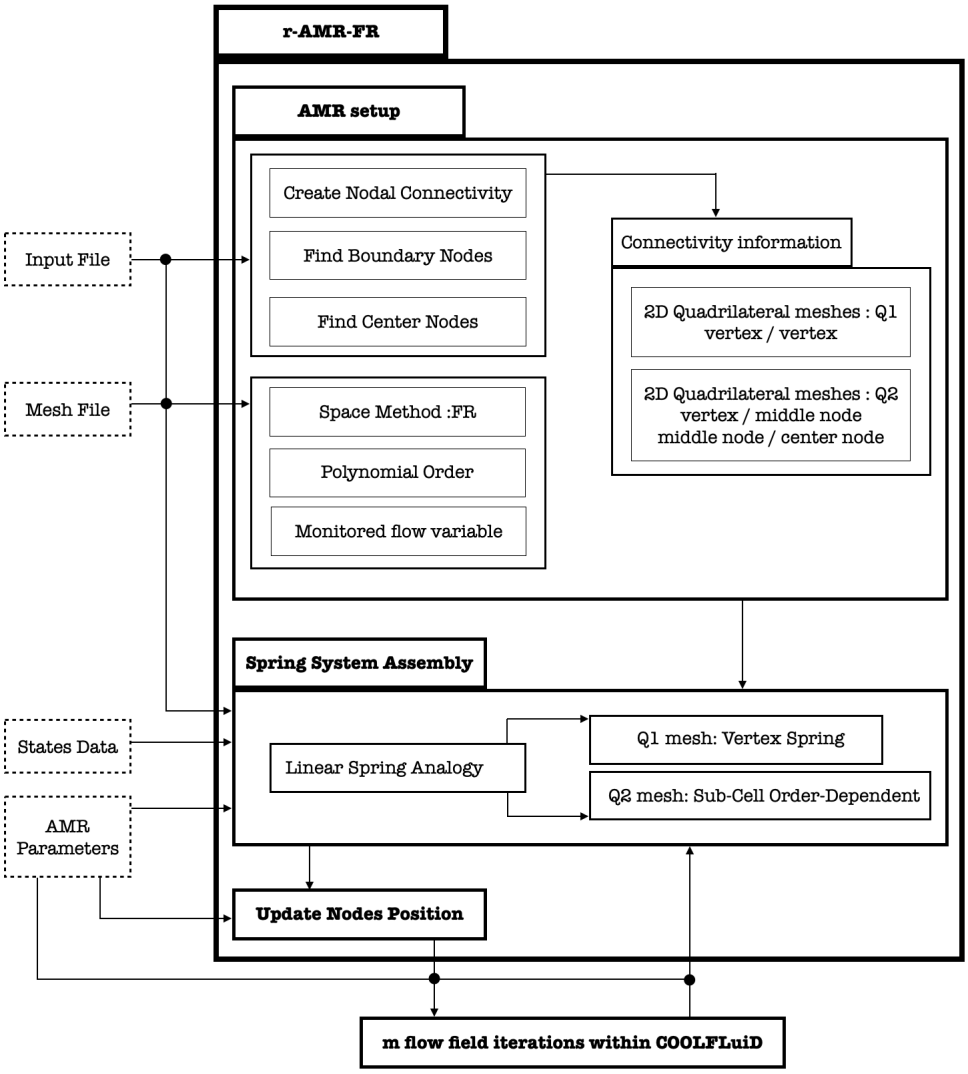

Figure 34: Overview about the r-AMR-FR implementation

10.2. General connectivity information

The FR data structure of COOLFluiD is organized in connectivity tables referred as Topological Region Sets (TRSs). These TRSs include the geometric entities (elements, faces and nodes) and their connectivity. Several types of TRS exist, in this paper we will only focus on meshes with quadrilateral cells:

- Elements TRS which links each cell to its corner nodes (4 for a quadrilateral) and to its faces (4 for a quadrilateral).
- Inner faces TRS which links each inner face (i.e. each face which is not at a boundary) to its nodes.
- Boundary faces TRS which links each boundary face to its nodes. It is similar to the inner faces TRS but there is one TRS per type of boundary condition applied to the boundary faces. For instance, the boundary faces to which Dirichlet conditions are applied, are separated from the ones to which Neumann conditions are imposed.

In each TRS, the faces are separated in different categories corresponding to their orientation. To define a face orientation, we should remember that all the FR algorithm takes place in a 2D reference domain $\Omega_S = \{[\xi, \eta]| -1 \leq \xi \leq$

$1 \cap -1 \leq \eta \leq 1\}$. In this reference domain, the solution points, the flux points, the nodes and the faces have a local numbering as illustrated in Figure 35. The orientation of a face depends on the local numbering of a face in the two neighboring elements to which it belongs. For instance, if the face number is 0 in its left element and 1 in its right element, it will have an orientation of 1, as shown in Figure 36.

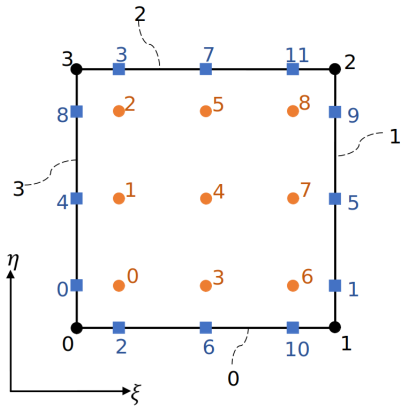

Figure 35: Local numbering in the reference element for a 2D quadrilateral element and a P2 reconstruction [53]

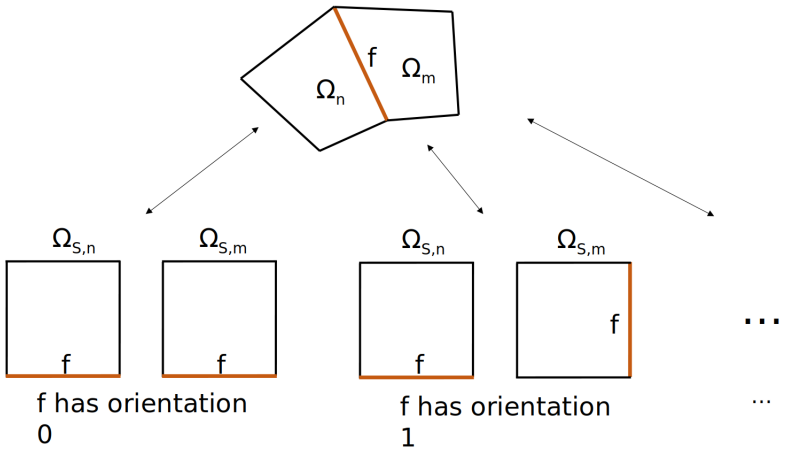

Figure 36: Examples of a face orientation for a 2D quadrilateral [53]

Our final objective is to place springs on the edges of the cells (a.k.a. faces in 2D). Consequently, we would like to be able to obtain a list of all the edge-connected nodes for each node of the mesh. As we are dealing with r-AMR, the process which is chosen here consists in building a nodal connectivity only once at the setup phase of the simulation in order to gain computational time. This approach is ideal since the connectivity with r-refinement is fixed for the whole simulation. To do so, the node IDs are placed into multimaps which are containers that associate an arbitrary number of values to a specific key. A part of the connectivity function is given in Algorithm 1. The idea here is to process all the faces with a given orientation, to obtain their 2 nodes and to add their connectivity information into multimaps.
The same procedure is repeated by looping on each boundary face TRS and on each element TRS. The final output of this whole procedure is four multimaps: the first one called *mapNodeNode* allows to deduce all the edge-connected nodes from a given node. The second called *mapNodeFace* gives all the faces (inner and boundary) to which a chosen node belongs. The third multimap called *mapNodeTRS* allows to obtain all the boundary TRS from a given node.

11. Transonic circular bump

A transonic flow simulation is performed over a circular bump, shown in Fig.37, with a thickness-to-chord ratio equals to 10%. The chosen Mach number is equal to 0.675. The outlet pressure boundary condition is set to a value corresponding to an isentropic Mach number $M_{is,out} = 0.675$. The imposed inlet values are depicted in Tab.7.

M [-]	ρ [-]	ρu [-]	ρv [-]	ρE [-]
0.675	1.0	0.79867077071	0.0	2.818937499

Table 7: 2D transonic bump inlet flow characteristics

The obtained $P2Q2$ Mach number contours with sub-cell order dependent spring analogy are shown in Fig.38. The contours are in a good agreement with the reference Mach contours taken from [54] and presented in Fig.39(left) while using a $\times 3.41$ coarser mesh. An other comparison is made with a solution taken from [55] and shown in Fig.39(right). We obtain similar Mach contours while using approximately $\times 1.56$ coarser mesh. In addition, the variation of the Mach number along the circular wall is depicted in Fig.40. The steep gradient of the embedded shock is well resolved and matches the one reported in the literature [55].

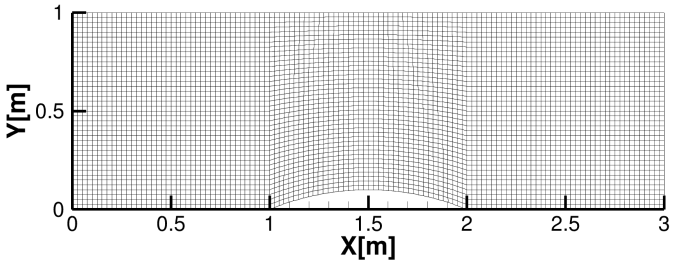

Figure 37: Initial mesh

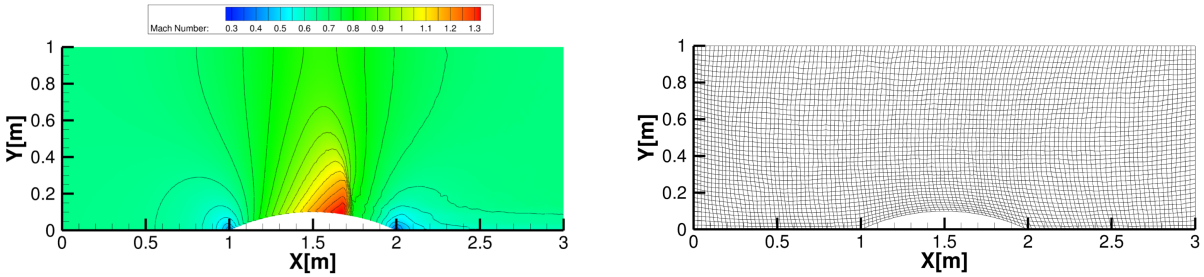

Figure 38: Final adapted mesh (left) and Mach contours for P2 (right)

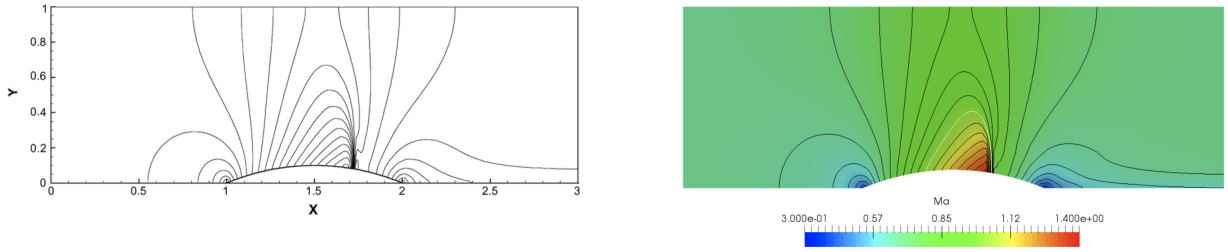

Figure 39: Reference solution taken from [54]: Mach Contours of the reference solution, using ×3.41 finer mesh (left) Reference solution taken from [55]: Mach Contours and solution field, using ×1.56 finer mesh (right

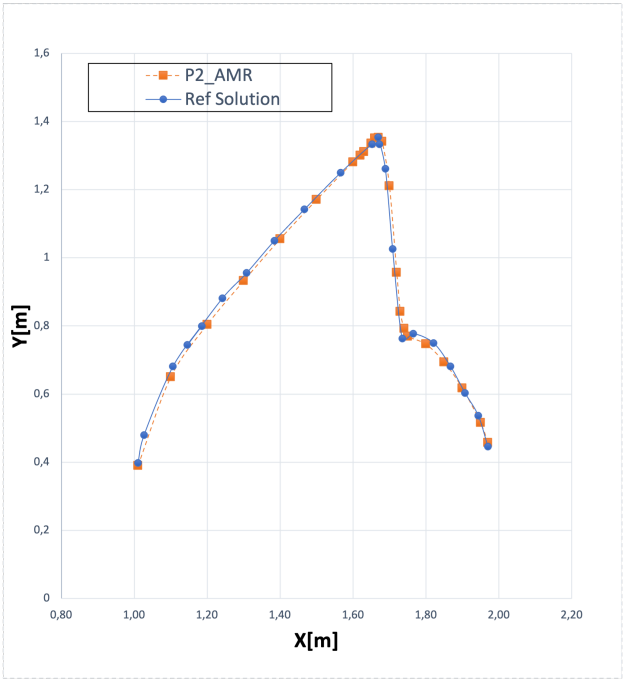

Figure 40: Mach number variation along the circular wall

12. Wedge channel – Vertex Spring

Parameters	P1 & P1AMR	P2 & P2AMR	P3 & P3AMR
κ	0.5	0.5	0.5
c	0.025	0.025	0.025
S_0	-3	-3	-4

Table 8: Simulation parameters for the 2D wedge channel – Vertex Spring

AMR parameters	P1AMR	P2AMR	P3AMR
Start AMR Iter	0	0	0
Stop AMR Iter	7000	7000	7000
Process rate	20	20	20
Min percentile	0.20	0.20	0.20
Max percentile	0.65	0.65	0.65
l_0 [m]	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
R_b	0.1	0.1	0.1

Table 9: AMR parameters for the 2D wedge channel – Vertex Spring

13. First wedge channel – Sub-cell order depend spring

Parameters	P1 & P1AMR	P2 & P2AMR	P3 & P3AMR
CFL	5	5	5
κ	0.5	0.5	0.5
c	0.025	0.025	0.025
S_0	-3	-3	-4

Table 10: Simulation parameters for the first wedge channel – Sub-cell order depend spring

AMR parameters	P1AMR	P2AMR	P3AMR
Start AMR Iter	0	0	0
Stop AMR Iter	1000	500	8000
Process rate	20	10	10
Min percentile	0.20	0.20	0.20
Max percentile	0.65	0.65	0.65
l_0 [m]	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
R_b	0.1	0.1	0.1

Table 11: AMR parameters for the first wedge channel – Sub-cell order depend spring

14. Long wedge channel – Sub-cell order dependent spring

Parameters	P1 & P1AMR	P2 & P2AMR	P3 & P3AMR	P4 & P4AMR
CFL	20	15	10	10
κ	0.5	0.5	0.5	0.5
c	0.025	0.025	0.025	0.025
S_0	-2	-3	-4	-4.5

Table 12: Simulation parameters for the long wedge channel

AMR parameters	P1AMR	P2AMR	P3AMR	P4AMR
Start AMR Iter	100	100	100	100
Stop AMR Iter	1000	1000	1000	1000
Process rate	20	20	20	20
Min percentile	0.20	0.20	0.20	0.20
Max percentile	0.65	0.65	0.65	0.65
l_0 [m]	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
R_b	0.1	0.1	0.1	0.1

Table 13: AMR parameters for the long wedge channel– Sub-cell order dependent spring

15. Circular bump

Parameters	P2AMR M_c	P3AMR M_c	P2AMR M_f	P3AMR M_f
CFL	20	15	20	10
κ	1	1	1	1
c	0.1	0.1	0.1	0.1
S_0	-3	-4	-3	-4

Table 14: Simulation parameters for the supersonic circular bump

AMR parameters	P2AMR M_c	P3AMR M_c	P2AMR M_f	P3AMR M_f
Start AMR Iter	300	300	300	300
Stop AMR Iter	2000	2000	2000	2500
Process rate	15	15	15	15
Min percentile	0.30	0.30	0.30	0.30
Max percentile	0.65	0.65	0.55	0.55
l_0 [m]	10^{-3}	10^{-3}	10^{-3}	10^{-3}
R_b	0.1	0.1	0.1	0.1

Table 15: AMR parameters for the supersonic circular bump

16. Pseudo-Algorithms

Input: Set of mesh cells and faces
Output: Multimap (*Key*: face ID, *Values*: connected Nodes IDs)

```
1  multimap mapFaceNode.size ← nbFaces ;
2  geoDataFace ← faceBuilder.getDataGeometricEntity ;
3  for orient ← 0 to nbrFaceOrients do
4      faceStartIdx ← innerFacesStartIdxs[orient] ;
5      faceStopIdx ← innerFacesStartIdxs[orient + 1] ;
6      for faceID ← faceStartIdx to faceStopIdx do
7          geoDataFace.idx ← faceID ;
8          face ← faceBuilder.buildGeometricEntity ;
9          faceNodes ← face.getNodes ;
10         for I ← 0 to faceNodes.size do
11             mapFaceNode.insert(faceID, faceNodes[i].getLocalID) ;
12 mapFaceNode.Sortkeys ;
13 return mapFaceNode
```

Algorithm 1: Inner faces connectivity information for 2D quadrilateral meshes

Input: Set of unlocked boundary TRS, set of boundary nodes
Output: Multimap (*Key*: movingBoundaryNodeID, *Values*: averageNormal)

```
1  multimap mapNodeIDNormal ;
2  multimap mapFaceIDNormal ;
3  for iTrs ← 0 to unlockedBoundaryTRSs.size do
4      nbFaces ← unlockedBoundaryTRSs[iTrs].getNbFaces ;
5      for i ← 0 to nbFaces do
6          faceID ← face.getID ;
7          faceNodes ← face.getNodes ;
8          coord[1] ← faceNodes[1].getCoordinates ;
9          coord[2] ← faceNodes[2].getCoordinates ;
10         mapFaceIDNormal.insert(faceID, computeNormal(coord[1], coord[2]))
11 for iNode ← 0 to boundaryNodes.size do
12     nodeID ← boundaryNodes[iNode].getID ;
13     connFaces ← boundaryNodes[iNode].getConnectedFaces ;
14     normal1 ← mapFaceIDNormal(connFaces[0]) ;
15     normal2 ← mapFaceIDNormal(connFaces[1]) ;
16     cos ← InnerProd(normal1, normal2) ;
17     if cos < ϵ then
18         averageNormal ← computeAverage(normal1, normal2) ;
19         mapNodeIDNormal.insert(nodeID, averageNormal) ;
20 return mapNodeIDNormal
```

Algorithm 2: Determination of moving boundary nodes

Input: Two edge-connected nodes $node1$ and $node2$
Output: A sub-cell order-dependent linear stiffness

faceID ← $getFaceInCommonID(node1, node2)$;
geoDataFace ← $faceBuilder.getDataGE$;
geoDataFace.idx ← $faceID$;
face ← $faceBuilder.buildGE()$;
states[Left] ← $cells[Left].getStates()$;
states[Right] ← $cells[Right].getStates()$;
for $iFlx \leftarrow 0$ **to** $nbrFaceFlxPnts - 1$ **do**
    statesFlxPnt[iFlx][Left] ← $extrapolateToFlxPnt(states[Left], iFlx)$;
    statesFlxPnt[iFlx][Right] ← $extrapolateToFlxPnt(states[Left], iFlx)$;
node1State[Left] ← $extrapolateToNode1(states[Left]))$;
node1State[Right] ← $extrapolateToNode1(states[Right]))$;
node2State[Left] ← $extrapolateToNode2(states[Left]))$;
node2State[Right] ← $extrapolateToNode2(states[Right]))$;
flxPntCoords ← $face.computeCoordFromLocalCoord(flxLocalCoords)$;
fluxVector ← $flxPntCoords[nbrFaceFlxPnts - 1] - flxPntCoords[0]$;
nodalVector ← $Node2.getCoordinates - Node1.getCoordinates$;
if $notAligned(FluxVector, NodalVector)$ **then**
    $actualStatesFlxPnt \leftarrow StatesFlxPnt.inverseOrder$;
else
    $actualStatesFlxPnt \leftarrow StatesFlxPnt$;
$k^L \leftarrow computeStiffness(Node1State, Node2State, actualStatesFlxPnt)$;
faceBuilder.releaseGE() ;
return k^L ;

Algorithm 3: Sub-cell order-dependent spring constants for Q1

Input: Set of mesh cells, faces and nodes
Output: Multimap (*Key*: nodeID, *Values*: edge-connected nodeIDs)

```
1  multimap mapNodeNode.size ← · nbNodes ;
2  nbCells ← number of mesh cells ;
3  for i ← 0 to nbCells − 1 do
4      nbFaces ← number of faces in cell[i] ;
5      for j ← 0 to nbFaces − 1 do
6          nbNodes ← number of nodes in face[j] ;
7          for k ← 0 to nbNodes − 1 do
8              nodeID ← faceNodes[k].getLocalID ;
9              if NeighborCell(nodeID) == 1 & NeighborFace(nodeID) == 1 then
10                 isBoundaryMiddlePoint(nodeID) ← true ;
11             if NeighborCell(nodeID) == 1 & isBound(nodeID) ← true then
12                 isBoundaryCornerPoint(nodeID) ← true ;
13             if NeighborCell(nodeID) == 1 & NeighborFace(nodeID) == 2 then
14                 isDomainCornerPoint(nodeID) ← true ;
15             if NeighborCell(nodeID) == 2 & isBound(nodeID) ← false then
16                 isMiddlePoint(nodeID) ← true ;
17             if NeighborCell(nodeID) == 4 then
18                 isCornerPoint(nodeID) ← true ;
19             mapNodeNode.insert(nodeID, findNeighborsID(nodeID)) ;
20 for i ← 0 to nbNodes − 1 do
21     nodeID ← nodes[i].getLocalID ;
22     if isInsideCell[nodeID] ← true then
23         isCenterNode(nodeID) ← true ;
24         mapNodeNode.insert(nodeID, findNeighborsID(nodeID)) ;
25 mapNodeNode.Sortkeys ;
26 return mapNodeNode
```

Algorithm 4: Connectivity information for Q2 meshes of 2D quadrilaterals

Input: Two edge-connected nodes *node*1 and *node*2
Output: A Q2 sub-cell order-dependent linear stiffness

1 **if** *InsideCell(node1) & InsideCell(node2)* ← *false* **then**
2 **do** lines 1-14 of Algorithm 3 ;
3 **for** $iFlx \leftarrow 0$ **to** $nbrFaceFlxPnts - 1$ **do**
4 Node1FluxVector ← $flxPntCoords[iFlx] - Node1.getCoordinates$;
5 Node2FluxVector ← $flxPntCoords[iFlx] - Node2.getCoordinates$;
6 NodalVector ← $Node2.getCoordinates - Node1.getCoordinates$;
7 **if** $Node1FluxVector.norm2() \;\&\; Node2FluxVector.norm2() < NodalVector.norm2()$ **then**
8 DistanceNode1Flux[iFlx] ← $Node1FluxVector.norm2()$;
9 nbrInsideFlxPnts +=1 ;
10 $ActualStatesFlxPnt \leftarrow Sort(StatesFlxPnt, DistanceNode1Flux)$;
11 $k^L \leftarrow computeStiffness(Node1State, Node2State, ActualStatesFlxPnt)$;
12 $faceBuilder.releaseGE()$;
13 **if** $InsideCell(node1) \;||\; InsideCell(node2) \leftarrow true$ **then**
14 geoDataCell ← *cellBuilder.getDataGE* ;
15 geoDataCell.idx ← *cellID* ;
16 cell ← *cellBuilder.buildGE* ;
17 cellStates ← *cell.getStates* ;
18 Node1State ← $extrapolateToNode1(cellStates)$;
19 Node2State ← $extrapolateToNode2(cellStates)$;
20 $k^L \leftarrow computeStiffness(Node1State, Node2State)$;
21 cellBuilder.releaseGE ;
22 **return** k^L ;

Algorithm 5: Sub-cell order-dependent spring constants for Q2 meshes