Title

A coupled discontinuous Galerkin-Finite Volume framework for solving gas dynamics over embedded geometries

Authors

Gulizzi, Vincenzo

Almgren, Ann S

Bell, John B

Publication Date

2022-02-01

DOI

10.1016/j.jcp.2021.110861

A coupled discontinuous Galerkin-Finite Volume framework for solving gas dynamics over embedded geometries

Vincenzo Gulizzi[1], Ann S. Almgren[1], and John B. Bell[1]

[1]Center for Computational Sciences and Engineering (CCSE), Lawrence Berkeley National Laboratory MS 50A-3111, Berkeley, CA 94720, USA

Abstract

We present a computational framework for solving the equations of inviscid gas dynamics using structured grids with embedded geometries. The novelty of the proposed approach is the use of high-order discontinuous Galerkin (dG) schemes and a shock-capturing Finite Volume (FV) scheme coupled via an *hp* adaptive mesh refinement (*hp*-AMR) strategy that offers high-order accurate resolution of the embedded geometries. The *hp*-AMR strategy is based on a multi-level block-structured domain partition in which each level is represented by block-structured Cartesian grids and the embedded geometry is represented implicitly by a level set function. The intersection of the embedded geometry with the grids produces the implicitly-defined mesh that consists of a collection of regular rectangular cells plus a relatively small number of irregular curved elements in the vicinity of the embedded boundaries. High-order quadrature rules for implicitly-defined domains enable high-order accuracy resolution of the curved elements with a cell-merging strategy to address the small-cell problem. The *hp*-AMR algorithm treats the system with a second-order finite volume scheme at the finest level to dynamically track the evolution of solution discontinuities while using dG schemes at coarser levels to provide high-order accuracy in smooth regions of the flow. On the dG levels, the methodology supports different orders of basis functions on different levels. The space-discretized governing equations are then advanced explicitly in time using high-order Runge-Kutta algorithms. Numerical tests are presented for two-dimensional and three-dimensional problems involving an ideal gas. The results are compared with both analytical solutions and experimental observations and demonstrate that the framework provides high-order accuracy for smooth flows and accurately captures solution discontinuities.

Keywords: Embedded boundaries, Discontinuous Galerkin methods, Finite Volume methods, Shock-capturing schemes, *hp*-AMR

1 Introduction

In spite of the enormous amount of work on solving the equations of gas dynamics, there is still considerable interest in developing higher-order methods in complex domains. In general, computational methods for problems in irregular domains can be classified into those employing body-fitted meshes (either unstructured or structured with curvilinear coordinates) and those employing structured meshes with embedded boundaries (EB). Body-fitted mesh approaches use irregular cells where the boundaries of the mesh elements coincide with the boundaries of the domain; although very flexible for modeling complex geometrical features, these methods often require significant effort to generate high-quality elements. On the other hand, EB methods represent the domain as a distinguished interface in a uniform background grid. The EB approach offers a number of benefits, such as automatic grid generation, data storage and adaptive mesh refinement [1], while the regularity of the mesh elements allows one to use methods designed for uniform grids in most of the domain. However, additional considerations need to be addressed in the presence of an embedded geometry.

The are two main approaches to designing EB methods, diffuse-interface methods [2, 3, 4], where the embedded boundaries or interfaces are smeared over a finite-thickness region that is captured with a sufficiently fine mesh, and sharp-interface methods [5, 6, 7, 8, 9, 10], where the embedded boundaries or interfaces are represented as a lower dimensional surface. The latter is often the recommended approach if the flow details in proximity of the geometries are of interest. However, the sharp-interface methods suffer from the so-called *small-cell* problem, i.e. the presence of cells cut by the embedded geometry that have unacceptably small volume fractions, which induces overly restrictive time steps and ill-conditioned discrete operators if not properly treated [11].

Most EB discretizations are based on Finite Volume (FV) schemes [12, 13] that are relatively easy to implement and offer robust shock-tracking capabilities. In embedded-boundary FV approaches, several strategies have been explored to ameliorate the small-cell problem. Notable examples include cell-merging [14, 15, 16, 17, 18, 19, 20], flux-redistribution [21, 22, 23, 24], the h-box method [25, 26], the dimensionally-split Cartesian cut cell method [27] and state redistribution [28]. However, embedded-boundary FV methods are typically limited to second-order accuracy in space and, if high-order accuracy is desired, see e.g. [20], they require large stencil to perform the solution reconstruction.

Recently, the discontinuous Galerkin (dG) methods [29, 30, 31, 32] have gained popularity thanks to the flexible variational framework on which they are based. The dG methods represent the unknown fields in a suitably chosen space of basis functions, whose coefficients become the unknowns of the discrete problem. This approach has several highly desirable features for parallel computing such as high-order accuracy via compact stencils, a natural way to handle hp refinement, more general (e.g. polytopic) element shapes [33, 34] and, unlike high-order schemes based on continuous approximations, block-structured mass matrices. As a result, the dG method, typically in combination with a cell-merging technique, has been successfully employed in embedded-boundary numerical schemes for the solution of several scientific and engineering problems including elliptic partial differential equations [35, 36, 37, 38], reaction-diffusion equations [39], two-phase flow [40, 41], two-phase flow with moving boundaries [42], surface tension dynamics, free-surface flow and rigid body-fluid interaction in the incompressible regime [8, 9, 43], and solid mechanics of thin structures with cutouts [44, 45].

In the context of compressible flow, embedded-boundary dG methods are less thoroughly investigated. For two-dimensional (2D) problems, Qin and Krivodonova [6] presented a dG methods for inviscid gas dynamics, Müller et al.[46] developed a dG method that includes viscous terms and, recently, Geisenhofer et al.[47] presented a dG method for inviscid compressible flow where the formation of solution discontinuities was captured by adding a suitably-chosen artificial viscosity to the governing equations. The EB approaches mentioned above use Cartesian background grids. Fidkowski and Darmofal [48] presented a 2D dG method using triangular cut cells for compressible flow, whereas, to the best of our knowledge, the only EB method involving both two- and three-dimensional computations of compressible fluids was recently proposed by Xiao et al.[49], who used triangular and tetrahedral meshes as their background grid.

As seen in the brief literature review provided above, most embedded-boundary dG schemes are for elliptic and incompressible fluid flow equations, which do not generally involve the formation and evolution of solution discontinuities, unless sharp corners or cracks are present in the geometry. Conversely, the development of shock waves and contact discontinuities are key features of inviscid compressible flow. When discontinuities are present, higher-order methods, including dG methods, produce nonphysical oscillations, referred to as the Gibbs phenomenon. Controlling oscillations in higher-order methods requires the introduction of a *limiting* strategy. Although various shock-capturing approaches for dG methods exist in the literature, such as artificial-viscosity methods [50, 51], moment limiters [52] or shock-fitting deforming-mesh approaches [53], among others, shock-capturing methods for embedded-boundary dG schemes is still not a mature area.

The aim of this work is to present a computational framework for embedded-boundary approaches that provides high-accuracy resolution of the geometry, high-order accuracy in regions of smooth flows and shock-capturing capabilities in the presence of solution discontinuities for two- and three-dimensional problems. This is achieved by coupling high-order dG schemes with a shock-capturing FV scheme through an hp adaptive mesh refinement (hp-AMR) strategy featuring high-accuracy resolution of embedded geometries.

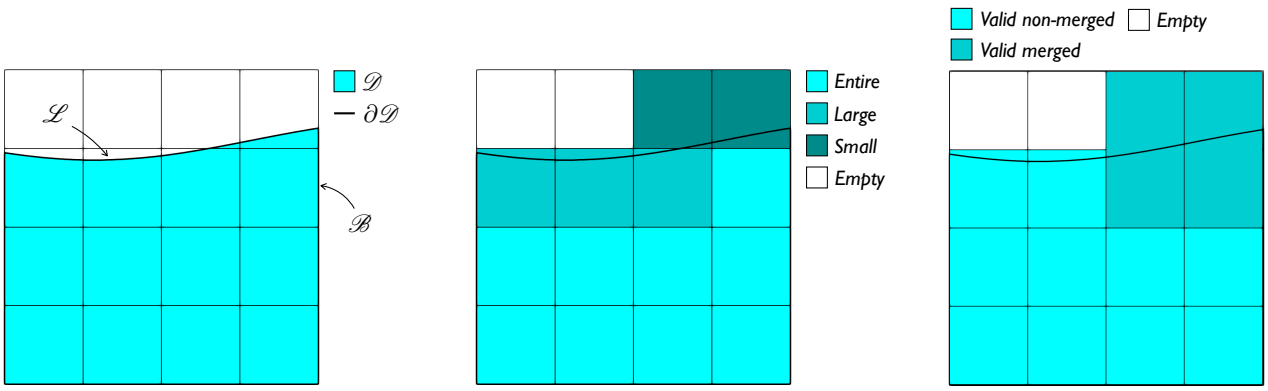
The paper is organized as follows: Sec.(2) introduces the considered initial-boundary value problem for inviscid gas dynamics; Sec.(3) describes the proposed framework that includes the construction of the implicitly-defined mesh, the formulation of the corresponding high-order dG schemes and the shock-capturing FV scheme, and the hp-AMR strategy; Sec.(4) presents a set of two- and three-dimensional numerical tests. Finally, Sec.(5) concludes the work and discusses potential further developments.

2 Governing equations

Inviscid gas dynamics is described by a system of first-order hyperbolic conservation laws given by

$$\partial_t \boldsymbol{U} + \partial_k \boldsymbol{F}_k = \boldsymbol{0}, \tag{1}$$

where ∂_t denotes the partial derivative with respect to the time t and ∂_k denotes the partial derivative with respect to the k-th coordinate x_k of the vector $\boldsymbol{x} = \{x_k\}$ in d-dimensional space. In Eq.(1), $\boldsymbol{U}$ is the $(d+2)$-dimensional



(a) (b) (c)

Figure 1: (a) Domain $\mathscr{D}$ and its boundary $\partial\mathscr{D} \equiv \mathscr{B} \cup \mathscr{L}$ implicitly defined in a 4×4 background grid. (b) Classification of the background cells of figure (a) on the basis of their volume fraction. (c) Classification of the background cells of figure (a) after cell-merging.

vector of conserved variables and $\boldsymbol{F}_k$ is the $(d+2)$-dimensional vector of the flux in the k direction defined as

$$\boldsymbol{U} \equiv \left\{ \begin{array}{c} \rho \\ \rho \boldsymbol{v} \\ \rho E \end{array} \right\} \quad \text{and} \quad \boldsymbol{F}_k \equiv \left\{ \begin{array}{c} \rho v_k \\ \rho v_k \boldsymbol{v} + p \boldsymbol{\delta}_k \\ (\rho E + p) v_k \end{array} \right\}, \tag{2}$$

where $\boldsymbol{\delta}_k$ is a d-dimensional vector containing 1 in its k-th component and 0 in all the others, ρ is the density, $\boldsymbol{v} = \{v_k\}$ is the velocity vector, E is the total energy and p is the pressure. The governing equations are closed by an equation of state. Here, we assume an ideal gas with ratio of specific heats γ, such that

$$p = (\gamma - 1)\rho \left(E - \frac{1}{2} v_k v_k \right). \tag{3}$$

The governing equations given in Eq.(1) are assumed to be valid for $\{t, \boldsymbol{x}\} \in \mathscr{I}_T \times \mathscr{D}$, where $\mathscr{I}_T \equiv [0, T]$ is the time interval, T is the final time, and $\mathscr{D} \subset \mathbb{R}^d$ is the spatial domain. The governing equations are supplemented by initial conditions and boundary conditions, which must be enforced on the boundary $\partial\mathscr{D}$ of the domain $\mathscr{D}$.

In Eqs.(1-3) and in the remainder of the paper, the indices k and l are used as subscripts to denote the component of a vector or an array; they take values in $\{1, \ldots, d\}$ and imply summation when repeated unless it is explicitly stated otherwise. Finally, we note that the symbol p will also be used to denote the order of the basis functions of dG schemes but its meaning will be clear from the context.

3 Coupled discontinuous Galerkin-Finite Volume framework

3.1 Implicitly-defined mesh

Let $\mathscr{D}$ be the domain, which we assume to be enclosed in a background rectangle $\mathscr{R} \supseteq \mathscr{D}$. We then let $\Phi : \mathscr{R} \to \mathbb{R}$ be a level set function such that $\mathscr{D}$ is implicitly defined as the region where Φ is negative, i.e. $\mathscr{D} \equiv \{\boldsymbol{x} \in \mathscr{R} : \Phi(\boldsymbol{x}) < 0\}$, and its boundary $\partial\mathscr{D}$ is defined as $\partial\mathscr{D} \equiv \mathscr{B} \cup \mathscr{L}$, where $\mathscr{B} \equiv \{\boldsymbol{x} \in \partial\mathscr{R} : \Phi(\boldsymbol{x}) < 0\}$ is the portion of the rectangle's outer boundary $\partial\mathscr{R}$ where Φ is negative, and $\mathscr{L} \equiv \{\boldsymbol{x} \in \mathscr{R} : \Phi(\boldsymbol{x}) = 0\}$ is the zero level of Φ. We then introduce a structured grid $\mathscr{G}_h \equiv \cup_{\boldsymbol{i}} \mathscr{C}^{\boldsymbol{i}} \subseteq \mathscr{R}$, where h denotes a characteristic mesh size, $\mathscr{C}^{\boldsymbol{i}}$ is a generic rectangular cell and $\boldsymbol{i} \equiv \{i_k\}$ is the d-tuple identifying the location of the cell within the grid. The cell $\mathscr{C}^{\boldsymbol{i}}$ is defined as $[x_1^{\boldsymbol{i}}, x_1^{\boldsymbol{i}} + h_1] \times \cdots \times [x_d^{\boldsymbol{i}}, x_d^{\boldsymbol{i}} + h_d]$, where $x_k^{\boldsymbol{i}}$ is the cell's lower corner in the k-th direction and h_k is the grid's mesh size in that direction.

When intersected with the domain $\mathscr{D}$, cells are characterized by their volume fraction ν: *entire* cells fall entirely inside $\mathscr{D}$ and have volume fraction $\nu = 1$; *empty* cells fall entirely outside $\mathscr{D}$ and have volume fraction $\nu = 0$; *partial* cells are cut by $\mathscr{L}$ and have volume fraction $0 < \nu < 1$. We then introduce a volume fraction threshold $\overline{\nu}$ to label each partial cell as either *large* if $\overline{\nu} < \nu < 1$ or *small* if $0 < \nu \leq \overline{\nu}$. As an example, Fig.(1a) shows a two-dimensional domain embedded in a 4×4 grid and Fig.(1b) shows the corresponding cell classification.

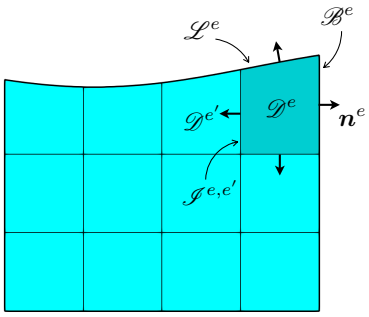
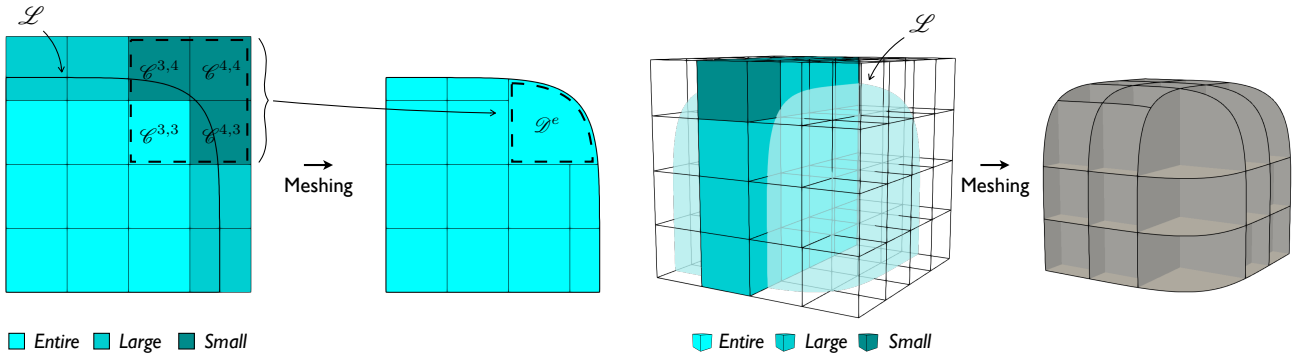


Figure 2: Implicitly-defined mesh of the domain shown in Fig.(1); the figure also highlights a mesh element $\mathscr{D}^e$ (in darker color) obtained from the merging between a small cell and an entire cell. See Figs.(1b) and (1c).



(a) (b)

Figure 3: Examples of (a) two-dimensional and (b) three-dimensional implicitly-defined meshes where multiple small cells are merged with the same valid cell.

The volume fraction threshold triggers the cell-merging strategy, whereby all cells labelled as small are merged with nearby cells. Several possibilities exist for implementing the cell-merging strategy. Here, a small cell is merged with one of its entire or large neighboring cells, henceforth collectively referred to as *valid* cells. The neighbor targeted for merging is selected according to its location with respect to the small cell and its volume fraction. Specifically, neighboring cells are queried in the following order: in 3D, we first consider the set of valid cells sharing a face with the small cell, second we consider the set of valid cells sharing an edge with the small cell and finally we consider the set of valid cells sharing a corner with the small cell. Within each set, the neighbors are ordered according to their volume fraction. Then, the target neighbor is the first cell in the first non-empty set. It follows that, in the choice of the target neighbor, a large cell sharing a face with the small cell has higher priority than an entire cell sharing an edge with the small cell. In 2D, the search starts from the cells that share an edge with a small cell.

Once the cell-merging procedure is applied, the cells of the grid $\mathscr{G}_h$ consist of *valid non-merged* cells, i.e. the entire and large cells that have not been targeted during the merging process, *valid merged* cells, i.e. the union of small cells and their merging large or entire neighbors, and empty cells. Such a classification for the background cells of Fig.(1b) is shown in Fig.(1c). Then, the implicitly-defined mesh $\mathcal{M}_h$ associated with $\mathscr{G}_h$ is defined by the intersection of the valid merged and non-merged cells of $\mathscr{G}_h$ and the domain $\mathscr{D}$, such that $\mathcal{M}_h \equiv \{\mathscr{D}^e\}$ and $\mathscr{D}^e$ is the e-th implicitly-defined mesh element. Figure (2) shows the implicitly-defined mesh associated with the grid of Fig.(1c) and highlights (in darker color) an implicitly-defined element $\mathscr{D}^e$ resulting from the merging of a small cell and an entire cell. The figure also shows the outer boundaries $\mathscr{B}^e$ and $\mathscr{L}^e$ of the e-th element, which stem from the intersection of the grid with $\mathscr{B}$ and $\mathscr{L}$, respectively, and the internal boundaries $\mathscr{I}^{e,e'}$ that the e-th element shares with its generic neighboring element e'. A few comments are worth pointing out:

- In the implementation of the implicit mesh, we do not explicitly merge cells. When a small cell and a valid cell have been merged, both the dG and the reconstructed FV representations of the solution on that merged cell are specified in terms of coefficients associated with the valid cell. Moreover, the geometry of the curved elements is never explicitly constructed or parameterized and is only inferred by the quadrature rules for domains and boundaries implicitly defined by the level set function.

- The cell-merging procedure needs to traverses the cells only once and is fully parallelizable. In fact, the volume fraction of the cells is computed prior to and does not change during the merging process; for example, after being merged, a small cell does not become a large cell eligible for merging with another small cell. Each small cell queries its neighboring cells and selects the most appropriate neighbor according to the procedure described above. Note that multiple small cells can target the same cell for merging; as a 2D example, Fig.(3a) shows a 4×4 grid where the small cells $\mathscr{C}^{4,3}$, $\mathscr{C}^{3,4}$ and $\mathscr{C}^{4,4}$ all merge with the cell $\mathscr{C}^{3,3}$ to define the implicitly-defined mesh element $\mathscr{D}^e$. A similar example in 3D is shown in Fig.(3b).

- In general, and especially for high volume fraction thresholds, it is possible that the merging algorithm does not find a suitable neighbor, i.e., given a small cell, all the cells in its 3×3 (in 2D) or its 3×3×3 (in 3D) neighborhood are small or empty; if this occurs, the simulation will terminate and require a smaller volume fraction threshold. However, in all the numerical tests considered here, a volume fraction threshold of 0.3 in 2D and a volume fraction threshold of 0.15 in 3D was found to be satisfactory. Other approaches could be implemented to deal with small cells that do not find suitable neighbors for merging. For example, one could consider merging multiple small cells together, thus increasing the effective mesh size of the elements, or one could lower the volume fraction threshold that triggers the merging and introduce an additional higher volume fraction thresholds that instead triggers a reduced polynomial order representation, thus improving the stability of the scheme. These approaches are, however, not considered here.

3.2 Discontinuous Galerkin schemes

Discontinuous Galerkin schemes are based on a weak form of the governing equations. For each element in the mesh we define a local polynomial basis on that element and extend it to vanish on all of the other elements. The collection of these functions for each element then forms a discontinuous basis. Here, it is natural to define the local basis functions as tensor-product polynomials since the discretization is constructed using Cartesian grids and the majority of the mesh elements are rectangular cells.

Let $\mathscr{D}^e$ be the implicitly-defined element associated with the grid cell $\mathscr{C}^{\boldsymbol{i}}$ (and all the small cells that are merged with $\mathscr{C}^{\boldsymbol{i}}$) and let $\mathcal{P}^e_{hp}$ be the space of tensor-product polynomials of degree p in the rectangular domain occupied

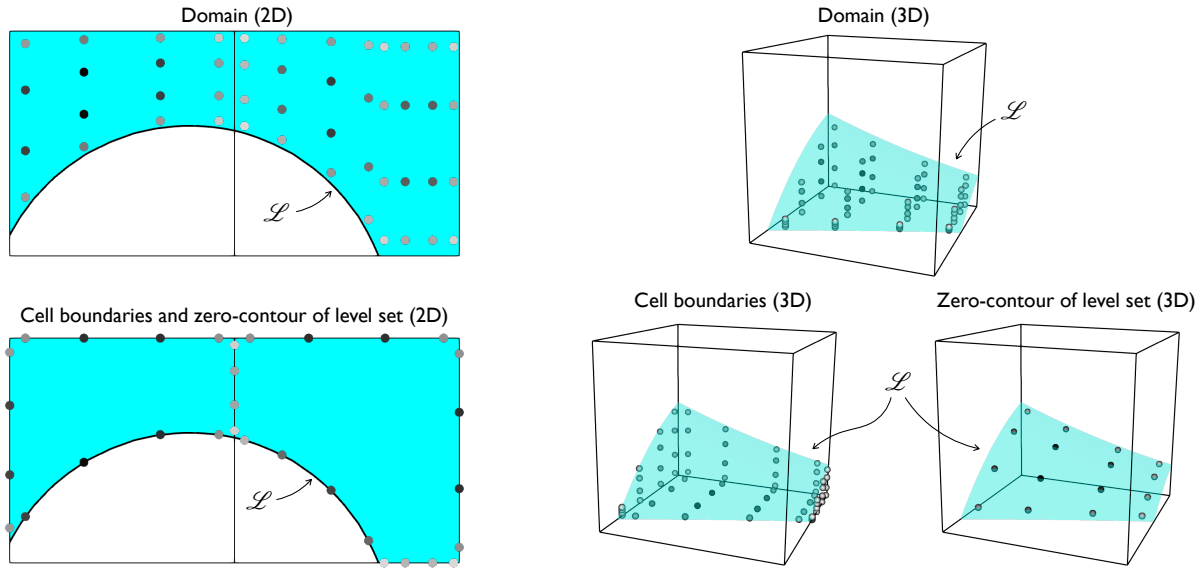


Figure 4: Examples of the location of the quadrature points obtained using the algorithm of Ref.[54]; in the images, the quadrature points are coloured according to their weight: light grays denote small weights while dark grays denotes large weights. Note that the quadrature weights are always strictly positive.

by $\mathscr{C}^{\boldsymbol{i}}$. Then, the space $\mathcal{V}_{hp}$ of discontinuous basis functions for the mesh $\mathcal{M}_h$ based on the grid $\mathscr{G}_h$ is

$$\mathcal{V}_{hp} \equiv \left\{ v : \mathscr{G}_h \to \mathbb{R} \mid v|_{\mathscr{D}^e} \in \mathcal{P}^e_{hp},\ \forall \mathscr{D}^e \in \mathcal{M}_h \right\}, \tag{4}$$

while the related space $\mathcal{V}^N_{hp}$ of discontinuous polynomials vector fields is $\mathcal{V}^N_{hp} \equiv (\mathcal{V}_{hp})^N$.

Following Cockburn et al.[29, 30], we integrate by parts in space to obtain the weak form of Eq.(1) as

$$\int_{\mathscr{D}^e} \boldsymbol{V}^\intercal \frac{\partial \boldsymbol{U}}{\partial t} \mathrm{d}V = \int_{\mathscr{D}^e} \frac{\partial \boldsymbol{V}^\intercal}{\partial x_k} \boldsymbol{F}_k \mathrm{d}V - \int_{\mathscr{B}^e \cup \mathscr{L}^e} \boldsymbol{V}^\intercal \widehat{\boldsymbol{F}}_n \mathrm{d}S - \sum_{e' \in \mathcal{N}^e} \int_{\mathscr{J}^{e,e'}} \boldsymbol{V}^\intercal \widehat{\boldsymbol{F}}_n \mathrm{d}S \quad \forall \mathscr{D}^e \in \mathcal{M}_h \text{ and } \boldsymbol{V} \in \mathcal{V}^{d+2}_{hp}. \tag{5}$$

In Eq.(5), $\mathcal{N}^e$ denotes the set of mesh elements that are neighbors of $\mathscr{D}^e$, and $\widehat{\boldsymbol{F}}_n$ denotes the so-called *numerical flux*, whose expression for hyperbolic equations is typically computed using an exact or an approximate Riemann solver [55, 56].

We now approximate $\boldsymbol{U}$ on each element as a linear combination of basis functions with time dependent coefficients. This can be written compactly as

$$\boldsymbol{U}|_{\mathscr{D}^e}(\boldsymbol{x}) = \boldsymbol{B}^e(\boldsymbol{x}) \boldsymbol{X}^e(t) \tag{6}$$

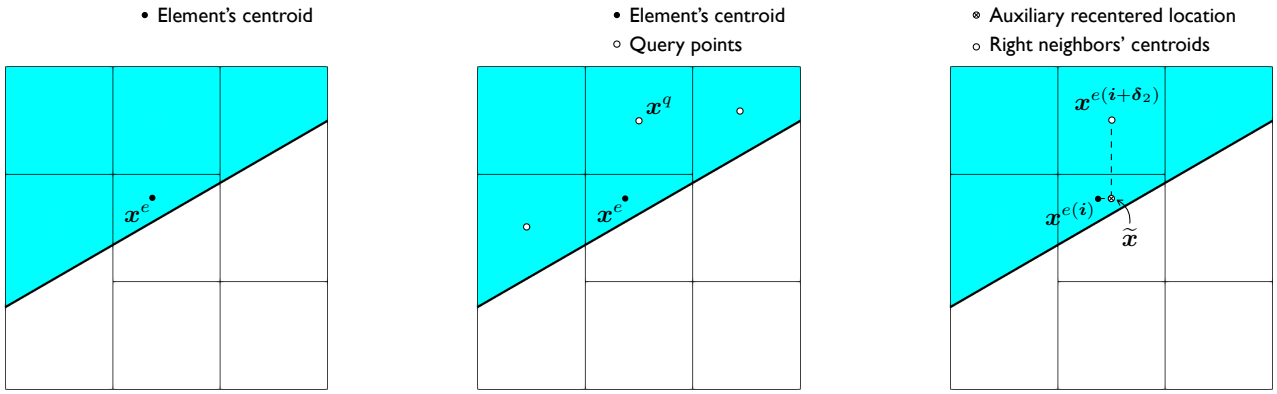
where $\boldsymbol{B}^e$ is an $N_u \times N_u N_p$ matrix where each column corresponds to a basis function in $\mathcal{V}_{hp}$ and $\boldsymbol{X}^e(t)$ is a vector of coefficients of length $N_u N_p$. Here $N_u = d+2$ is the number of unknowns in the system and $N_p = (1+p)^d$ is the number of polynomials in $\mathcal{P}^e_{hp}$. Substituting Eq. (6) into Eq.(5) and letting $\boldsymbol{V}$ range over the basis functions, we obtain the final semidiscrete evolution equation

$$\boldsymbol{M}^e \dot{\boldsymbol{X}}^e = \int_{\mathscr{D}^e} \frac{\partial \boldsymbol{B}^{e\intercal}}{\partial x_k} \boldsymbol{F}_k \mathrm{d}V - \int_{\mathscr{B}^e \cup \mathscr{L}^e} \boldsymbol{B}^{e\intercal} \widehat{\boldsymbol{F}}_n \mathrm{d}S - \sum_{e' \in \mathcal{N}^e} \int_{\mathscr{J}^{e,e'}} \boldsymbol{B}^{e\intercal} \widehat{\boldsymbol{F}}_n \mathrm{d}S, \tag{7}$$

where the superimposed dot denotes the derivative with respect to time, and $\boldsymbol{M}^e$ is the mass matrix of e-th element given by

$$\boldsymbol{M}^e \equiv \int_{\mathscr{D}^e} \boldsymbol{B}^{e\intercal} \boldsymbol{B}^e \ \mathrm{d}V. \tag{8}$$

The domain and boundary integrals appearing in Eqs.(7) and (8) are evaluated using high-order quadrature rules for implicitly-defined domains and boundaries. These quadrature rules are generated using the algorithm developed by Saye [54] and provide high-order accuracy in the presence of smooth embedded geometries. This is a key ingredient for the accuracy of the overall scheme. For example, Qin and Krivodonova [6] discuss the importance



(a) (b) (c)

Figure 5: Reconstruction and limiting for an element belonging to an implicitly-defined mesh in a 3×3 grid: (a) Location of the centroid $\boldsymbol{x}^e$ of the element where the reconstruction is to be performed; (b) Cloud of query points used in the least-square problem; (c) Auxiliary recentered location $\widetilde{\boldsymbol{x}}$ of the centroid $\boldsymbol{x}^{e(\boldsymbol{i})}$ such that $\widetilde{\boldsymbol{x}}$ is aligned with the centroid of the element $e(\boldsymbol{i}+\boldsymbol{\delta}_2)$.

of an accurate representation of embedded geometries and use curvature-aware boundary conditions to avoid the loss of accuracy due to the piecewise representation of the embedded boundaries. An example of the quadrature rules used here is provided in Fig.(4), which shows the distribution of the quadrature points for the domain and boundary integrals of 2D and 3D implicitly-defined elements.

3.3 Finite volume schemes

In a FV setting, the weak form of the governing equations can be derived by integrating Eq.(1) over a generic implicitly-defined element $\mathscr{D}^e \in \mathcal{M}_h$ and applying the divergence theorem with respect to the space variables to obtain

$$m^e \dot{\boldsymbol{U}}^e = -\int_{\mathscr{B}^e \cup \mathscr{L}^e} \widehat{\boldsymbol{F}}_n \mathrm{d}S - \sum_{e' \in \mathcal{N}^e} \int_{\mathscr{I}^{e,e'}} \widehat{\boldsymbol{F}}_n \mathrm{d}S, \tag{9}$$

where the superimposed dot again denotes the derivative with respect to time, $m^e \equiv \int_{\mathscr{D}^e} 1 \, \mathrm{d}V$ is the volume of $\mathscr{D}^e$, $\boldsymbol{U}^e$ represents the average of the conserved variables over $\mathscr{D}^e$, i.e. $\boldsymbol{U}^e \equiv \frac{1}{m^e} \int_{\mathscr{D}^e} \boldsymbol{U} \mathrm{d}V$, and the remaining symbols have the same meaning as those appearing in Eq.(5).

Similar to Eq.(7), high-order quadrature rules are employed to evaluate the right-hand side of Eq.(9) and maintain the high-order accuracy representation of the geometry for the FV scheme. However, unlike dG schemes, where the solution at any space location within the element is computed via its polynomial representation, Eq.(9) requires a suitable reconstruction and limiting strategy, which allows one to evaluate the solution, and thus the numerical flux $\widehat{\boldsymbol{F}}_n$, at the element's boundaries.

Although high-order reconstruction methods are available in the literature, the present approach is based on a linear representation of the solution, which leads to a scheme that is at most second-order accurate in space. This choice is mainly motivated by the fact that, within the adaptive mesh refinement algorithm, the FV grids will be fine-tuned to track and resolve the regions containing solution discontinuities, where high-order schemes would be reduced to first-order schemes by the limiter. The details of the limiter employed here are given in Sec.(3.3.1).

3.3.1 Reconstruction and limiting

In FV schemes, the typical reconstruction and limiting strategy is to compute the slopes of the primitive variables, perform the limiting of the slopes using characteristic variables and then compute the flux $\hat{\boldsymbol{F}}_n$ appearing in Eq.(9) using the reconstructed primitive variables. Here, we follow a slightly different approach, whereby the limiting is still performed using characteristics variables but the final reconstruction is performed on the conserved variables. This preserves the robustness of characteristics-based limiters but enables a more natural coupling with the dG schemes, where conserved variables are readily available at the elements' boundaries.

The vector $\widetilde{\boldsymbol{U}}^e$ of the reconstructed conserved variables at any point $\boldsymbol{x} \in \mathscr{D}^e$ is assumed to be given by

$$\widetilde{\boldsymbol{U}}^e = \boldsymbol{U}^e + \partial_k \boldsymbol{U}^e \cdot (x_k - x_k^e), \tag{10}$$

where $\boldsymbol{x}^e \equiv \{x_k^e\}$ is the centroid of $\mathscr{D}^e$, $\boldsymbol{U}^e$ is the vector of the conserved variable averages and $\partial_k \boldsymbol{U}^e$ are the slopes to be determined. It is easy to see that the expression given in Eq.(10) preserves conservation, since $\int_{\mathscr{D}^e} \widetilde{\boldsymbol{U}}^e \mathrm{d}V = m^e \boldsymbol{U}^e$, and that the conserved variable averages $\boldsymbol{U}^e$ coincide with the values of the conserved variables evaluated at the centroid of the element. As discussed next, $\partial_k \boldsymbol{U}^e$ are determined by reconstructing and limiting the slopes of the primitive variables.

Let $\boldsymbol{Q}^e \equiv \{\rho, \boldsymbol{v}, p\}$ be the $(d+2)$-dimensional vector of the primitive variables corresponding to $\boldsymbol{U}^e$ and Q^e be a generic component of $\boldsymbol{Q}^e$. The initial slopes of Q^e are obtained by the solution of the normal equations arising from the least-square interpolation problem given by

$$\boldsymbol{S}\boldsymbol{\Delta} = \boldsymbol{b}, \tag{11}$$

where

$$S_{kl} \equiv \sum_{q \in \mathcal{Q}} (x_k^q - x_k^e)(x_l^q - x_l^e), \quad \Delta_k \equiv \partial_k Q^e, \quad \text{and} \quad b_k \equiv \sum_{q \in \mathcal{Q}} (x_k^q - x_k^e)(Q^q - Q^e). \tag{12}$$

In (12), $\mathcal{Q}$ denotes the set of neighboring query elements, whose centroid $\boldsymbol{x}^q$ and primitive variables $\boldsymbol{Q}^q$ are selected to perform the least-square interpolation. Here, for an implicitly-defined element associated with the cell $\mathscr{C}^{\boldsymbol{i}}$, the set $\mathcal{Q}$ consists of the elements associated with the valid cells in the standard 4-cell, in 2D, or 6-cell, in 3D, neighborhood of $\mathscr{C}^{\boldsymbol{i}}$ plus the elements associated with the valid cells that extend into the small cells of the same neighborhood of $\mathscr{C}^{\boldsymbol{i}}$. Figure (5a) shows the case of a mesh implicitly defined on a 3×3 grid and Fig.(5b) shows the corresponding cloud of query points (denoted by the open circles) that are selected to perform the least-square reconstruction over the element associated to the central cell. It is worth noting that it is possible that the number of query points is not sufficient for the definition of the least-square problem given in Eq.(11); in these cases, the slopes are set to zero.

Once the reconstruction is obtained a slope limiter is required to avoid non-physical oscillations and ensure monotonicity in the presence of solution discontinuities. The limiter employed in this work consists of a two-step process. The first step is based on a suitably modified version of the dimensionally-split characteristics-based Van Leer limiter [57] combined with the recentering strategy suggested by Berger et al.[58] to handle nonaligned centroids of adjacent elements. More specifically, let $e(\boldsymbol{i})$ be the superscript denoting a quantity defined on the element $\mathscr{D}^e$ associated with the cell $\mathscr{C}^{\boldsymbol{i}}$, and let $\partial_k \boldsymbol{Q}^{e(\boldsymbol{i})}$ be the unlimited slopes in k-th direction obtained via Eq.(11). In the k-th direction, let $e(\boldsymbol{i} + \boldsymbol{\delta}_k)$ be the index of the *right* neighbor of the element $e(\boldsymbol{i})$, and $e(\boldsymbol{i} - \boldsymbol{\delta}_k)$ be the index of the *left* neighbor of element $e(\boldsymbol{i})$. The values of the primitive variables in the right and left neighbors are then used to compute the corresponding right slopes $\partial_k^+ \boldsymbol{Q}^{e(\boldsymbol{i})}$ and the left slopes $\partial_k^- \boldsymbol{Q}^{e(\boldsymbol{i})}$. Consider for example $\partial_k^+ \boldsymbol{Q}^{e(\boldsymbol{i})}$ and an auxiliary recentered location $\widetilde{\boldsymbol{x}}$ of the centroid $\boldsymbol{x}^{e(\boldsymbol{i})}$ obtained by translating $\boldsymbol{x}^{e(\boldsymbol{i})}$ in the plane perpendicular to the k-th direction and aligning it with the centroid of the right neighbor $e(\boldsymbol{i} + \boldsymbol{\delta}_k)$. To illustrate, Fig.(5c) sketches the location of $\widetilde{\boldsymbol{x}}$ when the right slopes are to be computed along the x_2 direction for the element associated to the central cell. Using the unlimited slopes, the primitive variables $\widetilde{\boldsymbol{Q}}$ evaluated at $\widetilde{\boldsymbol{x}}$ are given by

$$\widetilde{\boldsymbol{Q}} = \boldsymbol{Q}^{e(\boldsymbol{i})} + \partial_l \boldsymbol{Q}^{e(\boldsymbol{i})} \cdot (\widetilde{x}_l - x_l^{e(\boldsymbol{i})}) = \boldsymbol{Q}^{e(\boldsymbol{i})} + \sum_{l \neq k} \partial_l \boldsymbol{Q}^{e(\boldsymbol{i})} \cdot (x_l^{e(\boldsymbol{i}+\boldsymbol{\delta}_k)} - x_l^{e(\boldsymbol{i})}) \tag{13}$$

where the second equality follows by noting that $\widetilde{x}_k = x_k^{e(\boldsymbol{i})}$ and $\widetilde{x}_l = x_l^{e(\boldsymbol{i}+\boldsymbol{\delta}_k)}$, for $l \neq k$. Then, the right slopes $\partial_k^+ \boldsymbol{Q}^{e(\boldsymbol{i})}$ are defined as

$$\partial_k^+ \boldsymbol{Q}^{e(\boldsymbol{i})} \equiv \frac{\boldsymbol{Q}^{e(\boldsymbol{i}+\boldsymbol{\delta}_k)} - \widetilde{\boldsymbol{Q}}}{x_k^{e(\boldsymbol{i}+\boldsymbol{\delta}_k)} - x_k^{e(\boldsymbol{i})}}. \tag{14}$$

Substituting Eq.(13) into Eq.(14), and following the same derivation for the left slopes, one obtains

$$\partial_k^{\pm} \boldsymbol{Q}^{e(\boldsymbol{i})} \equiv \frac{\boldsymbol{Q}^{e(\boldsymbol{i}\pm\boldsymbol{\delta}_k)} - \boldsymbol{Q}^{e(\boldsymbol{i})}}{x_k^{e(\boldsymbol{i}\pm\boldsymbol{\delta}_k)} - x_k^{e(\boldsymbol{i})}} - \frac{\sum_{l \neq k} \partial_l \boldsymbol{Q}^{e(\boldsymbol{i})} \cdot (x_l^{e(\boldsymbol{i}\pm\boldsymbol{\delta}_k)} - x_l^{e(\boldsymbol{i})})}{x_k^{e(\boldsymbol{i}\pm\boldsymbol{\delta}_k)} - x_k^{e(\boldsymbol{i})}}. \tag{15}$$

In Eqs.(14) and (15), no summation is implied over repeated subscripts. Moreover, it is worth noting that, far from the embedded boundaries, where the elements and their neighbors are regular entire cells, the slopes given in Eq.(15) coincide with the slopes of standard structured grid methods since, along the k-th direction, one has $x_l^{e(\boldsymbol{i}\pm\boldsymbol{\delta}_k)} = x_l^{e(\boldsymbol{i})}$, for $l \neq k$. This is not the case near the embedded boundary where the term $x_l^{e(\boldsymbol{i}\pm\boldsymbol{\delta}_k)} - x_l^{e(\boldsymbol{i})}$ accounts for the difference in location of the elements' centroids. However, as suggested by Fig.(5c), it is possible that some of the neighboring elements might not be available because the neighboring cells are small or empty

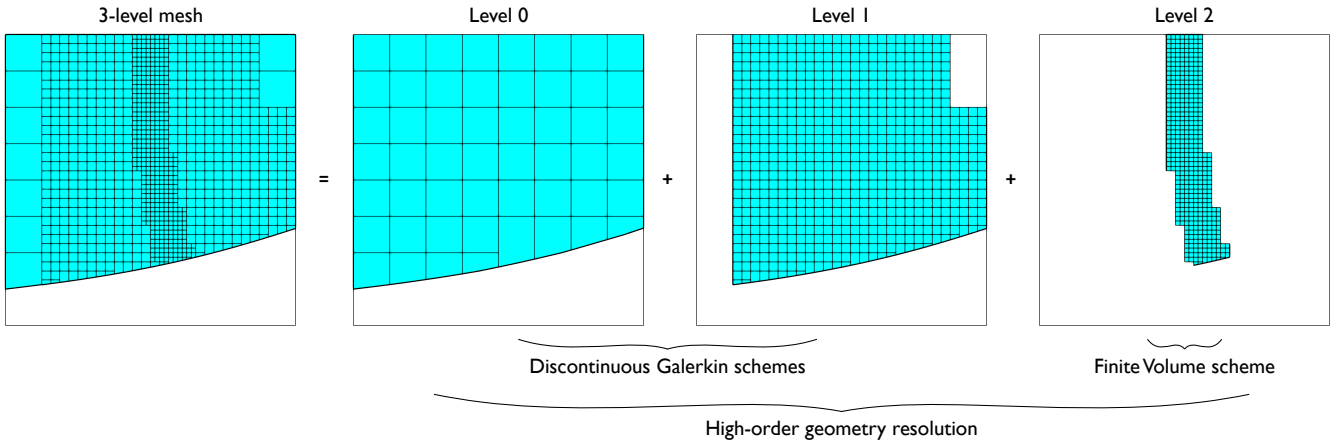


Figure 6: Example of a three-level implicitly-defined mesh in which the levels 0 and 1 use discontinuous Galerkin schemes and the level 2 uses a finite-volume scheme. At all levels, the geometry is resolved with high-order accuracy.

according to the classification introduced in Sec.(3.1). In these cases, the left or the right derivative cannot be computed and will not be considered in the evaluation of the limited slopes.

The slopes of the primitive variables are used to compute the corresponding slopes of the characteristics variables as

$$\partial_k \boldsymbol{C}^{e(\boldsymbol{i})} \equiv \boldsymbol{L}^k \partial_k \boldsymbol{Q}^{e(\boldsymbol{i})} \quad \text{and} \quad \partial_k^{\pm} \boldsymbol{C}^{e(\boldsymbol{i})} \equiv \boldsymbol{L}^k \partial_k^{\pm} \boldsymbol{Q}^{e(\boldsymbol{i})}, \tag{16}$$

where $\boldsymbol{L}^k$ is the matrix of left eigenvectors of $(\partial \boldsymbol{Q}/\partial \boldsymbol{U})(\partial \boldsymbol{F}_k/\partial \boldsymbol{U})(\partial \boldsymbol{U}/\partial \boldsymbol{Q})$ evaluated at $\boldsymbol{U}^{e(\boldsymbol{i})}$. Then, the vector $\overline{\partial}_k \boldsymbol{C}^{e(\boldsymbol{i})}$ of limited slopes of the characteristic variables is obtained by applying the Var Leer limiter

$$\overline{\partial}_k C^{e(\boldsymbol{i})} \equiv \begin{cases} s \min\{\theta^- |\delta^-|, |\delta|, \theta^+ |\delta^+|\}, & \text{if } \mathscr{C}^{\boldsymbol{i} \pm \boldsymbol{\delta}_k} \text{ are valid and } \delta^-, \delta, \delta^+ \text{ have the same sign;} \\ s \min\{\theta^- |\delta^-|, |\delta|\}, & \text{if } \mathscr{C}^{\boldsymbol{i} - \boldsymbol{\delta}_k} \text{ is valid, } \mathscr{C}^{\boldsymbol{i} + \boldsymbol{\delta}_k} \text{ is not valid and } \delta^-, \delta \text{ have the same sign;} \\ s \min\{|\delta|, \theta^+ |\delta^+|\}, & \text{if } \mathscr{C}^{\boldsymbol{i} + \boldsymbol{\delta}_k} \text{ is valid, } \mathscr{C}^{\boldsymbol{i} - \boldsymbol{\delta}_k} \text{ is not valid and } \delta, \delta^+ \text{ have the same sign;} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

component-wise. In Eq.(17), $\overline{\partial}_k C^{e(\boldsymbol{i})}$ denotes a generic component of $\overline{\partial}_k \boldsymbol{C}^{e(\boldsymbol{i})}$, $\delta \equiv \partial_k C^{e(\boldsymbol{i})}$, $\delta^{\pm} \equiv \partial_k^{\pm} C^{e(\boldsymbol{i})}$, $s \equiv \text{sign}(\delta)$, $\theta^- \equiv (x_k^{e(\boldsymbol{i})} - x_k^{e(\boldsymbol{i}-\boldsymbol{\delta}_k)})/(x_k^{e(\boldsymbol{i})} - x_k^{\boldsymbol{i}})$, $\theta^+ \equiv (x_k^{e(\boldsymbol{i}+\boldsymbol{\delta}_k)} - x_k^{e(\boldsymbol{i})})/(x_k^{\boldsymbol{i}} + h_k - x_k^{e(\boldsymbol{i})})$, with no summation implied over the repeated index k. Finally, the limited slopes of the primitive variables and the corresponding slopes of the conserved variables are obtained as

$$\overline{\partial}_k \boldsymbol{Q}^e = (\boldsymbol{L}^k)^{-1} \overline{\partial}_k \boldsymbol{C}^e \quad \text{and} \quad \overline{\partial}_k \boldsymbol{U}^e = \left(\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{Q}} \right)_{\boldsymbol{U}^e} \overline{\partial}_k \boldsymbol{Q}^e \tag{18}$$

respectively.

The slopes computed using Eq.(18) do not ensure that over- or under-shoots of the solution are avoided at the boundaries of the mesh elements, especially in the case of extended elements where left or right derivatives are not available. For this reason, the second step of the limiter applies the Barth-Jespersen limiter [59], which is a scalar limiter that reduces the slopes along all directions by the same multiplicative factor α. Thus, the slopes entering Eq.(10) are defined as $\partial_k \boldsymbol{U}^e \equiv \alpha^e \overline{\partial}_k \boldsymbol{U}^e$, where α^e is computed via a Barth-Jespersen limiter procedure that uses the reconstructed solution at quadrature point on the boundaries $\mathscr{I}^{e,e'} \; \forall e' \in \mathcal{N}^e$.

3.4 Block-structured adaptive mesh refinement

The dG schemes and FV schemes are integrated into a block-structured adaptive mesh refinement algorithm (AMR) [60, 61, 62] that represents the solution as a hierarchy of levels of refinement ranging from coarsest ($\ell = 0$) to finest ($\ell = \ell_{\text{finest}}$). Each level is represented by the union of non-overlapping logically-rectangular grids of a given resolution.

Here, the grids at level $\ell = 0$ are generated at the beginning of the simulation and cover the entire domain; the corresponding implicitly-defined mesh is also generated at startup and does not change during the simulation.

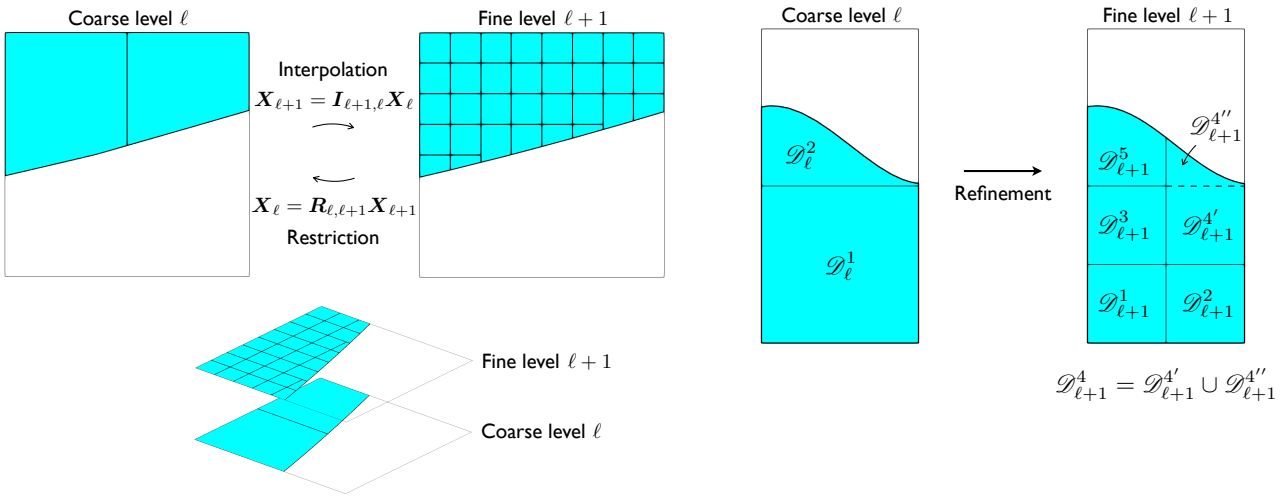


Figure 7: (a) Illustration of the operations performed by the interpolation operator $\boldsymbol{I}_{\ell+1,\ell}$ and the restriction operator $\boldsymbol{R}_{\ell+1,\ell}$. (b) Example of an AMR configuration where the fine element $\mathscr{D}_{\ell+1}^4$ covers portions of the two distinct coarse elements $\mathscr{D}_\ell^1$ and $\mathscr{D}_\ell^2$.

Grids at levels $\ell > 0$ are created and destroyed dynamically during the simulation, and the corresponding implicitly-defined meshes follow the same dynamic evolution. We consider AMR configurations where dG schemes are used at the levels $0 \leq \ell < \ell_{\text{finest}}$ while the FV scheme is always used at the finest level $\ell = \ell_{\text{finest}}$. This includes the case where AMR uses dG schemes only or the case where AMR consists of a single-level FV scheme. The present AMR strategy also supports the use of different order basis functions at different dG levels and, as such, is referred to as *hp*-AMR. Figure (6) shows an example of a 3-level *hp*-AMR configuration with a refinement ratio of 4 between level 0 and level 1 and a refinement ratio of 2 between level 1 and level 2. The figure also shows that dG schemes are used at levels 0 and 1 and a FV scheme is used at the level 2. This configuration is typical of what one would expect for a discontinuity attached to a curved boundary.

The dynamic update of the AMR levels is governed by tagging and un-tagging operations, whereby a cell at a given level is assigned a two-value variable, or tag, that determines whether a finer resolution is required at its location or whether the existing set of finer grid cells can be replaced by a coarse cell. These operations require coarsening and refinement operations that transfer the solution between levels. For instance, when a coarse cell is tagged for refinement and a finer grid is overlaid on it, the solution from the coarse level needs to be interpolated to the newly created fine grid. Conversely, the solution at a fine level must be restricted to the corresponding coarser level to determine whether or not the fine grid is still required at that location. These operations must preserve the high-order accuracy of the method and must be available for all the possible combinations of the numerical schemes used between consecutive levels. Nevertheless, these are linear operations local to elements and therefore can be efficiently implemented using matrix-vector products taking advantage of parallelization offered by dG and FV schemes.

To illustrate how interpolation and restriction are performed here, we introduce the vector $\boldsymbol{X}_\ell$ of the solution at the level ℓ. For a level that uses a dG scheme, $\boldsymbol{X}_\ell$ contains the coefficients of the basis functions representing the conserved variables, whereas, for a level that uses the FV scheme, $\boldsymbol{X}_\ell$ collects the element averages of the conserved variables. For the interpolation and restriction operations, we consider the FV scheme as a dG scheme of discontinuous constant basis functions. This means that interpolation to the finest level returns the element averages, which are subsequently used to perform the reconstruction and limiting process presented in Sec.(3.3.1). Note that interpolation is never performed from the finest level and therefore the reconstructed slopes are not used during the interpolation and restriction operations.

As sketched in Fig.(7a), the interpolation operator $\boldsymbol{I}_{\ell+1,\ell}$ computes the solution $\boldsymbol{X}_{\ell+1}$ at the fine level $\ell+1$ from the solution $\boldsymbol{X}_\ell$ at the coarse level ℓ, and is formally written as

$$\boldsymbol{X}_{\ell+1} = \boldsymbol{I}_{\ell+1,\ell}\boldsymbol{X}_\ell. \tag{19}$$

In Eq.(19), $\boldsymbol{I}_{\ell+1,\ell}$ is a block-structured matrix computed using a Galerkin projection, so that, given an element $\mathscr{D}_{\ell+1}^{e'}$

of the fine level $\ell+1$ and an element $\mathscr{D}_\ell^e$ of the coarse level ℓ for which the interpolation operation is meaningful, the block $\boldsymbol{I}_{\ell+1,\ell}^{e',e}$ is given by

$$\boldsymbol{I}_{\ell+1,\ell}^{e',e} \equiv (\boldsymbol{M}_{\ell+1}^{e'})^{-1} \int_{\mathscr{D}_{\ell+1}^{e'} \cap \mathscr{D}_\ell^e} \boldsymbol{B}_{\ell+1}^{e'\intercal} \boldsymbol{B}_\ell^e \, \mathrm{d}V, \tag{20}$$

where, for an element e on level ℓ, $\boldsymbol{B}_\ell^e$ contains the basis functions and $\boldsymbol{M}_\ell^e$ is the mass matrix as defined in Eq.(8).

It is worth giving a practical example of how the interpolation operator $\boldsymbol{I}_{\ell+1,\ell}$ acts on $\boldsymbol{X}_\ell$. Consider Fig.(7b), which shows a two-element coarse mesh $\mathcal{M}_{h_\ell} \equiv \{\mathscr{D}_\ell^1, \mathscr{D}_\ell^2\}$ associated with a 1×2 coarse grid, and a five-element fine mesh $\mathcal{M}_{h_{\ell+1}} \equiv \{\mathscr{D}_{\ell+1}^1, \mathscr{D}_{\ell+1}^2, \ldots, \mathscr{D}_{\ell+1}^5\}$ associated with a 2×4 grid that is obtained by refining the coarse grid with a refinement ratio of 2. Then, $\boldsymbol{X}_\ell$ consists of the basis functions coefficients of the elements $\mathscr{D}_\ell^1$ and $\mathscr{D}_\ell^2$, $\boldsymbol{X}_{\ell+1}$ consists of the basis functions coefficients of the elements $\mathscr{D}_{\ell+1}^1$ to $\mathscr{D}_{\ell+1}^5$. The interpolation operation given in Eq.(19) becomes

$$\left\{ \begin{array}{c} \boldsymbol{X}_{\ell+1}^1 \\ \boldsymbol{X}_{\ell+1}^2 \\ \boldsymbol{X}_{\ell+1}^3 \\ \boldsymbol{X}_{\ell+1}^4 \\ \boldsymbol{X}_{\ell+1}^5 \end{array} \right\} = \left[\begin{array}{cc} \boldsymbol{I}_{\ell+1,\ell}^{1,1} & \boldsymbol{0} \\ \boldsymbol{I}_{\ell+1,\ell}^{2,1} & \boldsymbol{0} \\ \boldsymbol{I}_{\ell+1,\ell}^{3,1} & \boldsymbol{0} \\ \boldsymbol{I}_{\ell+1,\ell}^{4,1} & \boldsymbol{I}_{\ell+1,\ell}^{4,2} \\ \boldsymbol{0} & \boldsymbol{I}_{\ell+1,\ell}^{5,2} \end{array} \right] \left\{ \begin{array}{c} \boldsymbol{X}_\ell^1 \\ \boldsymbol{X}_\ell^2 \end{array} \right\}, \tag{21}$$

where each block $\boldsymbol{I}_{\ell+1,\ell}^{e,e'}$, with $e = 1, \ldots, 5$ and $e' = 1, 2$, is computed via Eq.(20). While the first, second, third and fifth rows of Eq.(21) are the result of a trivial application of the Galerkin projection, it is interesting to derive the interpolation operation given in the fourth row of Eq.(21). First, with reference to Fig.(7b), note that the fine element $\mathscr{D}_{\ell+1}^4$ is the result of a merging operation and covers portions of the two distinct coarse elements $\mathscr{D}_\ell^1$ and $\mathscr{D}_\ell^2$. Then, assuming for simplicity that $\boldsymbol{X}_\ell$ contains the basis functions coefficients of a discontinuous scalar field $u : \mathcal{M}_{h_\ell} \to \mathbb{R}$ at the coarse level ℓ, the Galerkin projection of u onto the fine element $\mathscr{D}_{\ell+1}^4$ is given by

$$\begin{aligned} \boldsymbol{X}_{\ell+1}^4 &= (\boldsymbol{M}_{\ell+1}^4)^{-1} \int_{\mathscr{D}_{\ell+1}^4} \boldsymbol{B}_{\ell+1}^{4\intercal} u \mathrm{d}V = \\ &= (\boldsymbol{M}_{\ell+1}^4)^{-1} \left(\int_{\mathscr{D}_{\ell+1}^{4'}} \boldsymbol{B}_{\ell+1}^{4\intercal} \boldsymbol{B}_\ell^1 \mathrm{d}V \right) \boldsymbol{X}_\ell^1 + (\boldsymbol{M}_{\ell+1}^4)^{-1} \left(\int_{\mathscr{D}_{\ell+1}^{4''}} \boldsymbol{B}_{\ell+1}^{4\intercal} \boldsymbol{B}_\ell^2 \mathrm{d}V \right) \boldsymbol{X}_\ell^2 = \boldsymbol{I}_{\ell+1,\ell}^{4,1} \boldsymbol{X}_\ell^1 + \boldsymbol{I}_{\ell+1,\ell}^{4,2} \boldsymbol{X}_\ell^2 \end{aligned} \tag{22}$$

where the last equality is obtained by noting that $\mathscr{D}_{\ell+1}^{4'} \equiv \mathscr{D}_{\ell+1}^4 \cap \mathscr{D}_\ell^1$ and $\mathscr{D}_{\ell+1}^{4''} \equiv \mathscr{D}_{\ell+1}^4 \cap \mathscr{D}_\ell^2$, and by using Eq.(20). The structure of $\boldsymbol{I}_{\ell+1,\ell}$ given in Eq.(21) reflects the configurations of the coarse and the fine meshes displayed in Fig.(7b), including the merging between cells; moreover, if $\mathscr{D}_{\ell+1}^{e'} \cap \mathscr{D}_\ell^e = \emptyset$, then the corresponding block $\boldsymbol{I}_{\ell+1,\ell}^{e,e'}$ is a zero matrix.

Figure (7b) also sketches the inverse of the interpolation operation, namely the restriction operation, where a solution $\boldsymbol{X}_\ell$ at a coarse level ℓ is computed from the solution $\boldsymbol{X}_{\ell+1}$ at a fine level $\ell+1$ using the restriction operator $\boldsymbol{R}_{\ell,\ell+1}$

$$\boldsymbol{X}_\ell = \boldsymbol{R}_{\ell,\ell+1} \boldsymbol{X}_{\ell+1}. \tag{23}$$

Following Fortunato et al.[63], the restriction operator can be defined as the adjoint of the interpolation operator and can evaluated as follows

$$\boldsymbol{R}_{\ell,\ell+1} = \boldsymbol{M}_\ell^{-1} \boldsymbol{I}_{\ell+1,\ell}^\intercal \boldsymbol{M}_{\ell+1}, \tag{24}$$

where $\boldsymbol{M}_\ell$ denotes the block-diagonal mass matrix associated to the entire level ℓ, i.e. $\boldsymbol{M}_\ell \equiv \mathrm{diag}(\{\boldsymbol{M}_\ell^e\})$. Note that Eq.(24) can also be obtained by applying the Galerkin projection from the fine level $\ell+1$ onto the coarse level ℓ and using Eq.(20). For the case shown in Fig.(7b), Eq.(24) becomes

$$\left\{ \begin{array}{c} \boldsymbol{X}_\ell^1 \\ \boldsymbol{X}_\ell^2 \end{array} \right\} = \boldsymbol{M}_\ell^{-1} \left[\begin{array}{ccccc} \boldsymbol{I}_{\ell+1,\ell}^{1,1\intercal} & \boldsymbol{I}_{\ell+1,\ell}^{2,1\intercal} & \boldsymbol{I}_{\ell+1,\ell}^{3,1\intercal} & \boldsymbol{I}_{\ell+1,\ell}^{4,1\intercal} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{\ell+1,\ell}^{4,2\intercal} & \boldsymbol{I}_{\ell+1,\ell}^{5,2\intercal} \end{array} \right] \boldsymbol{M}_{\ell+1} \left\{ \begin{array}{c} \boldsymbol{X}_{\ell+1}^1 \\ \boldsymbol{X}_{\ell+1}^2 \\ \boldsymbol{X}_{\ell+1}^3 \\ \boldsymbol{X}_{\ell+1}^4 \\ \boldsymbol{X}_{\ell+1}^5 \end{array} \right\}. \tag{25}$$

A few comments are worth pointing out:

- The interpolation and restriction operators defined via Eqs.(20) and (23), respectively, are valid regardless of the choice of basis functions and can accommodate AMR levels using different mesh sizes, polynomial orders or both. In particular, they are valid even if the same mesh uses the same space of basis functions with different polynomial orders; this feature will be used to define the shock sensor discussed in Sec.(3.4.1).

- From an implementation perspective, given an *hp*-AMR configuration, one only needs to evaluate the interpolation operators and the Cholesky decomposition of the mass matrices; then the restriction operator can be applied on-the-fly using Eq.(24). Moreover, all the standard rectangular elements (which represent most of the mesh elements) share the same mass matrix, which can precomputed and stored at the beginning of the simulation; the same applies to the interpolation operator between two standard rectangular elements of two different AMR levels. For the remaining implicitly-defined mesh elements, the mass matrices and the interpolation operators are in general unique and are computed using high-order quadrature rules [54].

- The mass matrix and interpolation operator are formally given by $\boldsymbol{M}_\ell$ and $\boldsymbol{I}_{\ell+1,\ell}$ that dynamically change as the *hp*-AMR levels evolve. However, these matrices are composed of localized blocks that can be computed and stored after each regridding.

- The interpolation and restriction operations are well-defined in all configurations of the implicitly-defined mesh obtained from the cell-merging algorithm as long as tagged cells are not merged with non-tagged cells. This situation can always be avoided by making sure that if an extended cell is tagged for refinement, so are all the small cells merged with it.

As the last comment on the present *hp*-AMR strategy, recall that when the right-hand side of Eq.(7) or (9) is to be evaluated, the coupling between neighboring elements e and e' occurs solely through the numerical flux $\widehat{\boldsymbol{F}}_n$ appearing in the boundary integrals over $\mathscr{I}^{e,e'}$. This applies equally to the cases when e and e' belong to the same *hp*-AMR level or when e and e' belong to different *hp*-AMR levels, including the level using the FV scheme. In fact, at a generic quadrature point $\boldsymbol{x}$ of $\mathscr{I}^{e,e'}$, $\widehat{\boldsymbol{F}}_n$ is computed as $\widehat{\boldsymbol{F}}_n = \widehat{\boldsymbol{F}}_n(\widetilde{\boldsymbol{U}}^e, \widetilde{\boldsymbol{U}}^{e'})$, where $\widetilde{\boldsymbol{U}}^e$ is the solution evaluated at $\boldsymbol{x}$ using either its representation in terms of basis functions, if the element e belongs to a level using a dG scheme, or the reconstruction provided by Eq.(10), if the element e belongs to a level using the FV scheme. However, is it pointed out that at the boundary between two levels using different schemes, the order of the quadrature rules must be high enough to provide an accurate integration of the contributions from the higher-order scheme.

3.4.1 Cell tagging and shock sensor

Two main criteria are employed to tag the elements for refinement and update the AMR levels during the simulations. The first one is a simple criterion that uses the magnitude of the density gradient. In particular, an element e of the AMR level ℓ is tagged for refinement if the following condition is satisfied

$$\frac{1}{m^e}\int_{\mathscr{D}^e} ||\nabla\rho||\ \mathrm{d}V > \kappa_\ell^\rho \tag{26}$$

where m^e is the volume of $\mathscr{D}^e$, $||\nabla\rho||$ denotes the magnitude of the density gradient and κ_ℓ^ρ is a threshold parameter that determines which elements of the ℓ-th level are tagged for refinement. In general, a more advanced tagging criterion, involving for example the entire set of conserved variables and/or higher order derivatives, is recommended to capture a wider variety of flow configurations, such as discontinuous shear flows. However, for the simulations considered here, the criterion given in Eq.(26) was tuned to use a low-order dG scheme in regions of constant flow and high-order dG schemes in regions of smooth flows. Then, cells at the dG level $\ell = \ell_{\text{finest}} - 1$ are tagged for refinement and replaced by the cells at the FV level $\ell = \ell_{\text{finest}}$ using the second criterion discussed next.

The second criterion is the so-called shock-sensor, which is used to determine whether the solution inside an element is behaving like a discontinuous field. Here, we employ the shock sensor introduced by Persson and Peraire [50] suitably modified to account for the implicitly-defined mesh. In Ref.[50], the Authors defined the following smoothness indicator s^e for the e-th mesh element

$$s^e \equiv \frac{\int_{\mathscr{D}^e}(U-\widetilde{U})^2\ \mathrm{d}V}{\int_{\mathscr{D}^e} U^2\ \mathrm{d}V} \tag{27}$$

where U is a field of the conserved variables, e.g. the density, and $\widetilde{U}$ is the same field expressed using a lower-order set of basis functions. For standard rectangular elements, $\widetilde{U}$ can be computed by considering a truncated series of the basis functions containing the terms up to order $p-1$, where p is the order of the basis functions used to

represent U. However, for implicitly-defined elements, the truncated series would not preserve the average of the field U in $\widetilde{U}$ and would degrade the efficacy of Eq.(27). Therefore, at the level ℓ using the space $\mathcal{V}_{hp}$, $\widetilde{U}$ is evaluated using the Galerkin projection of the field U onto the space $\mathcal{V}_{h(p-1)}$. A discontinuity is then assumed to be present if an element e if the following condition is satisfied

$$\log_{10}(s^e) > -4\log_{10}(p) + \kappa^s, \tag{28}$$

where p is the order of the polynomial basis functions employed at the level containing the element e, and κ^s is a threshold parameter that determines which elements are tagged for refinement. The effect of the parameter κ^s on the evolution of the *hp*-AMR levels will be discussed in the numerical examples. It is worth noting that in Ref.[50] the Authors employed the smoothness sensor given in Eq.(27) to inject a suitably-chosen artificial viscosity to the governing equations. Here, it is employed only to activate the AMR level that uses the FV scheme. We note that the refinement conditions given in (26) and (28) are used not only to tag the elements for refinement but also to remove the finer grids on regions where they are no longer needed.

4 Numerical tests

In this section, the capabilities of the proposed method are assessed for two- and three-dimensional test cases. In all computations, an approximate two-shock Riemann solver [55] is used to compute the numerical flux $\widehat{\boldsymbol{F}}_n$ appearing in Eqs.(7) and (9). Tensor-product Legendre polynomials of degree p are used to define the spaces $\mathcal{P}^e_{hp}$ and $\mathcal{V}_{hp}$ and the corresponding dG scheme is labelled as dG_p.

Whether a single level or an *hp*-AMR is used, the time evolution of the solution unknowns associated to the generic mesh element can be written as

$$\boldsymbol{M}^e \dot{\boldsymbol{X}}^e = \boldsymbol{A}^e(t, \boldsymbol{X}), \tag{29}$$

where $\boldsymbol{A}^e(t, \boldsymbol{X})$ is the result of the evaluation of the right-hand side of Eq.(7) (for dG schemes) or Eq.(9) (for FV schemes), $\boldsymbol{X}$ contains the solution unknowns of the whole problem, and, if the element e belongs to a level implementing the FV scheme, $\boldsymbol{M}^e \equiv \mathrm{diag}(m^e)$ and $\dot{\boldsymbol{X}}^e \equiv \dot{\boldsymbol{U}}^e$. $\boldsymbol{X}$ formally contains the solution unknowns of all the mesh elements but only the neighbors of the e-th element are needed to compute $\boldsymbol{A}^e$ in Eq.(29).

The integration of Eq.(29) over all mesh elements of the *hp*-AMR hierarchy is performed explicitly in time using a high-order TVD Runge-Kutta (RK) algorithms [30]. The order of the RK algorithm is chosen to match the highest spatial discretization order among the levels; for example, if a two-level mesh uses a dG_1 scheme and a dG_2 scheme, Eq.(29) is integrated in time at both levels via a third-order RK algorithm. The time step of the RK algorithm is chosen to be the smallest time step among the levels. Specifically, at level ℓ covering a subregion $\mathscr{D}_\ell$ of the domain $\mathscr{D}$ with grids of characteristic mesh size h_ℓ, the time step τ_ℓ is subject to the CFL condition

$$\frac{\tau_\ell}{h_\ell} < \frac{\mathrm{C}_\ell \overline{\nu}_\ell}{\lambda_\ell}, \tag{30}$$

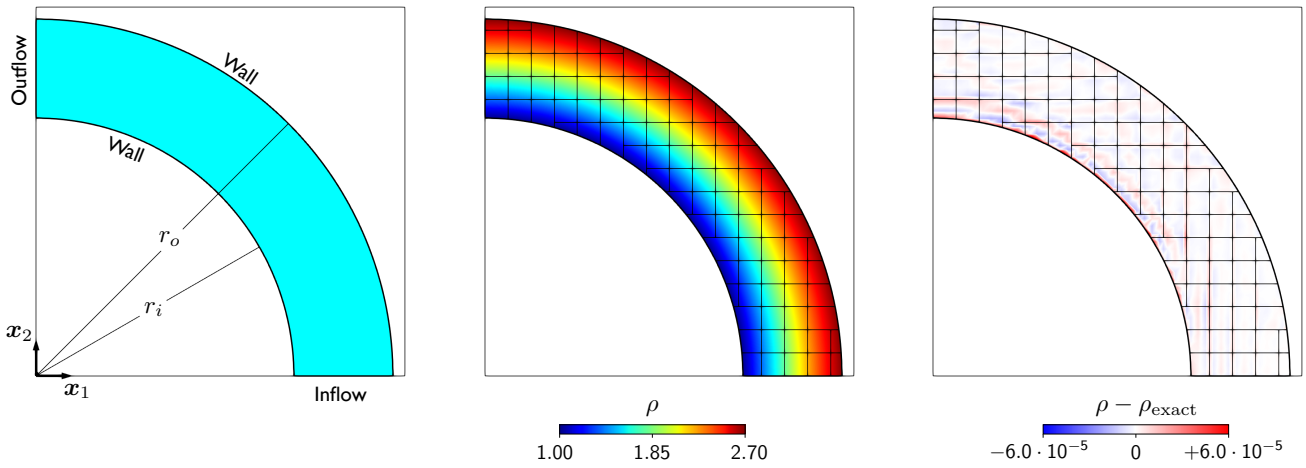
where $\lambda_\ell \equiv \max_{\mathscr{D}_\ell}(\sqrt{v_k v_k} + a)$ is the maximum wave speed on $\mathscr{D}_\ell$ given by the sum of the velocity magnitude $\sqrt{v_k v_k}$ and the speed of sound a, $\overline{\nu}_\ell$ is the volume fraction threshold that triggers the merging process for the small cells, and $\mathrm{C}_\ell = 0.3$ if the level ℓ uses a FV scheme or $\mathrm{C}_\ell = 1/(2p+1)$ if the level ℓ uses a dG_p scheme. Then, the time step of the Runge-Kutta algorithm is $\tau \equiv \min_\ell \tau_\ell$. All the computations use $\gamma = 1.4$ and all reported quantities are non-dimensional.

4.1 Supersonic vortex

In this test, we consider a two-dimensional supersonic vortex problem with an exact smooth solution that has been employed in several studies to investigate the accuracy of numerical methods for inviscid gas dynamics, see e.g. [64, 28]. The problem consists of isentropic flow in a circular annulus of inner radius $r_i = 1$ and outer radius $r_o = 1.384$. The exact solution for density ρ_{exact}, pressure p_{exact}, and velocity components $v_{1\mathrm{exact}}$ and $v_{2\mathrm{exact}}$ are given by the following expressions

$$\rho_{\mathrm{exact}} = \rho_i \left[1 + \frac{\gamma-1}{2} M_i^2 \left(1 - \frac{r_i^2}{r^2}\right)\right]^{\frac{1}{\gamma-1}}, \quad p_{\mathrm{exact}} = \frac{\rho_{\mathrm{exact}}^\gamma}{\gamma}, \quad v_{1\mathrm{exact}} = -v_\theta \sin\theta \quad \text{and} \quad v_{2\mathrm{exact}} = v_\theta \cos\theta, \tag{31}$$

where $v_\theta = a_i M_i \frac{r_i}{r}$, r is the distance between a point inside the annulus and the center of the annulus, and $\rho_i = 1$, $a_i = 1$ and $M_i = 2.25$ are the density, sound speed and Mach number, respectively, at $r = r_i$.



(a) (b) (c)

Figure 8: (a) Geometry of supersonic vortex problem defined in the background square $[0, 1.43]^2$. (b) Density distribution and (c) density error obtained using a 16×16 background mesh and a dG_3 scheme.

The problem is defined in the background square $\mathscr{R} = [0, 1.43]^2$ where the domain $\mathscr{D}$ and its boundaries are represented by the level set function

$$\Phi(\boldsymbol{x}) \equiv \begin{cases} r_i^2 - r^2 & \text{if } r \leq (r_i + r_o)/2 \\ r^2 - r_o^2 & \text{otherwise} \end{cases}, \quad \text{with} \quad r^2 = x_k x_k. \tag{32}$$

Figure (8a) shows the geometry of the problem and indicates the boundary conditions. The exact solution given in Eq.(31) is used to define the initial conditions for Eq.(29). The system is integrated in time until a steady state is reached. We consider two error measures

$$e_{L_2}(\rho, \rho_{\text{exact}}) \equiv \frac{||\rho - \rho_{\text{exact}}||_2}{||\rho_{\text{exact}}||_2} \quad \text{and} \quad e_{L_\infty}(\rho, \rho_{\text{exact}}) \equiv \frac{||\rho - \rho_{\text{exact}}||_\infty}{||\rho_{\text{exact}}||_\infty}, \tag{33}$$

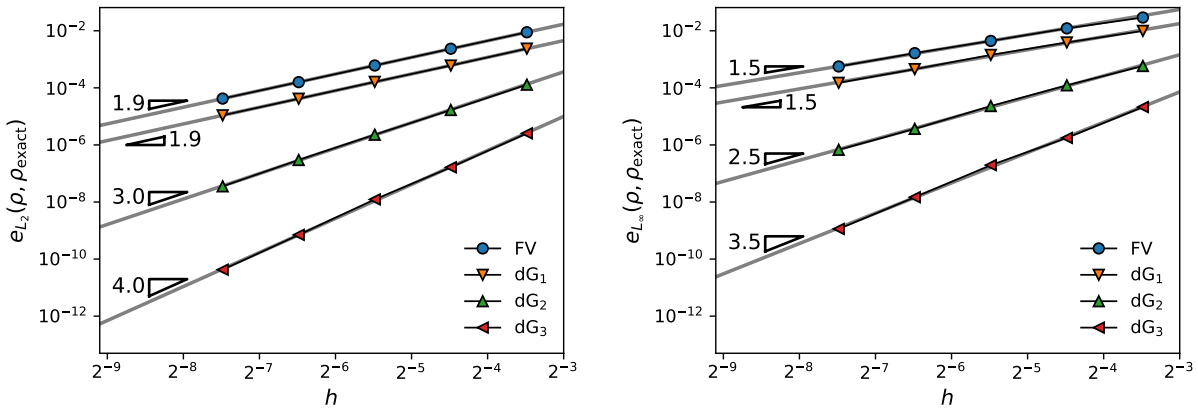
where $||\cdot||_2$ and $||\cdot||_\infty$ are the standard $L_2(\mathscr{D})$ norm and $L_\infty(\mathscr{D})$ norm, respectively, defined on the domain $\mathscr{D}$, ρ is the value computed using the present approach and ρ_{exact} is given by Eq.(31). The steady-state solution is assumed to be reached when the relative error between the evaluation of the norms given in Eq.(33) between two consecutive time steps is less than 10^{-5}. For all considered simulations, this occurred at $t > 5.0$.

The simulations are performed using a single-level mesh, with $h \equiv 1.43/n$, where n is the number of elements per side of the background square, and where the volume fraction threshold $\overline{\nu} = 0.3$. Figure (8b) shows the density distribution obtained using a 16×16 background mesh and a dG_3 scheme; the corresponding density error distribution is reported in fig.(8c), which also shows that the maximum error is observed near the inner boundary of the annulus. The error measures defined in Eq.(33) are shown in Fig.(9) as functions of the numerical scheme and the mesh size h. As seen in the figures, the dG schemes show an $\mathcal{O}(h^{p+1})$ convergence rate in the L_2 norm and an $\mathcal{O}(h^{p+1/2})$ convergence rate in the L_∞ norm, while the FV scheme using the reconstruction (without limiting) presented in Sec.(3.3.1) obtains the same convergence rate as the dG_1 scheme, albeit with a slightly larger error.

4.2 Embedded Sod's shock tube

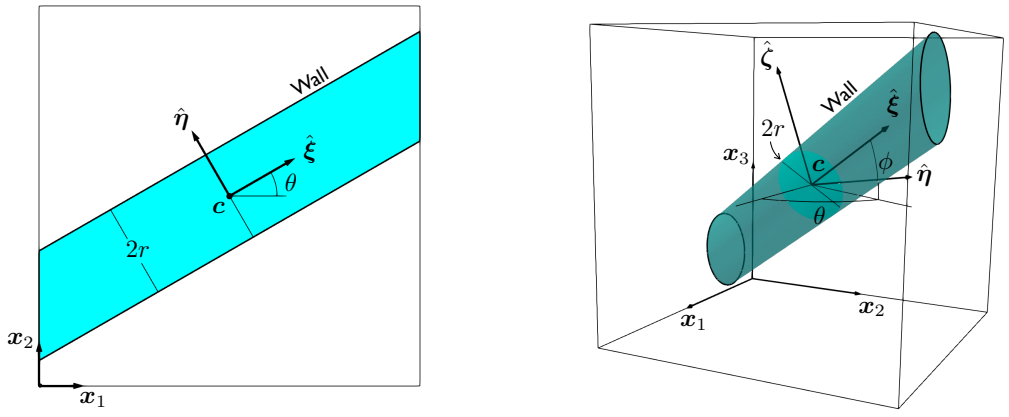
In the second test, we consider Sod's shock tube problem in embedded geometries. These simulations assess the ability of the implicitly-defined mesh and the reconstruction and limiting technique presented in Sec.(3.3.1) to reproduce a one-dimensional flow within a geometry that is not aligned with the background grid.

The problem is defined in the background unit cube $\mathscr{R} = [0, 1]^d$ as shown in Fig.(10). In 2D, the tube is a strip of width $2r$ centered at $\boldsymbol{c} \equiv \{0.5, 0.5\}$ whose orientation is defined by the parameter θ, which denotes the inclination of the tube's centerline. In 3D, the tube is a cylinder with a circular cross section of radius r centered at $\boldsymbol{c} \equiv \{0.5, 0.5, 0.5\}$ whose orientation is defined by the two parameters θ and ϕ, which denote the azimuth and the elevation, respectively, of the tube's centerline. For convenience, we introduce a rotated coordinate system $\hat{\boldsymbol{\xi}}\hat{\boldsymbol{\eta}}$



(a) (b)

Figure 9: Convergence in (a) the L_2 norm and (b) the L_∞ norm of the density error for the supersonic vortex problem.



(a) (b)

Figure 10: (a) Two-dimensional and (b) three-dimensional geometry for the tilted Sod's tube problem defined in the background square $[0, 1]^2$ and in the background cube $[0, 1]^3$, respectively.

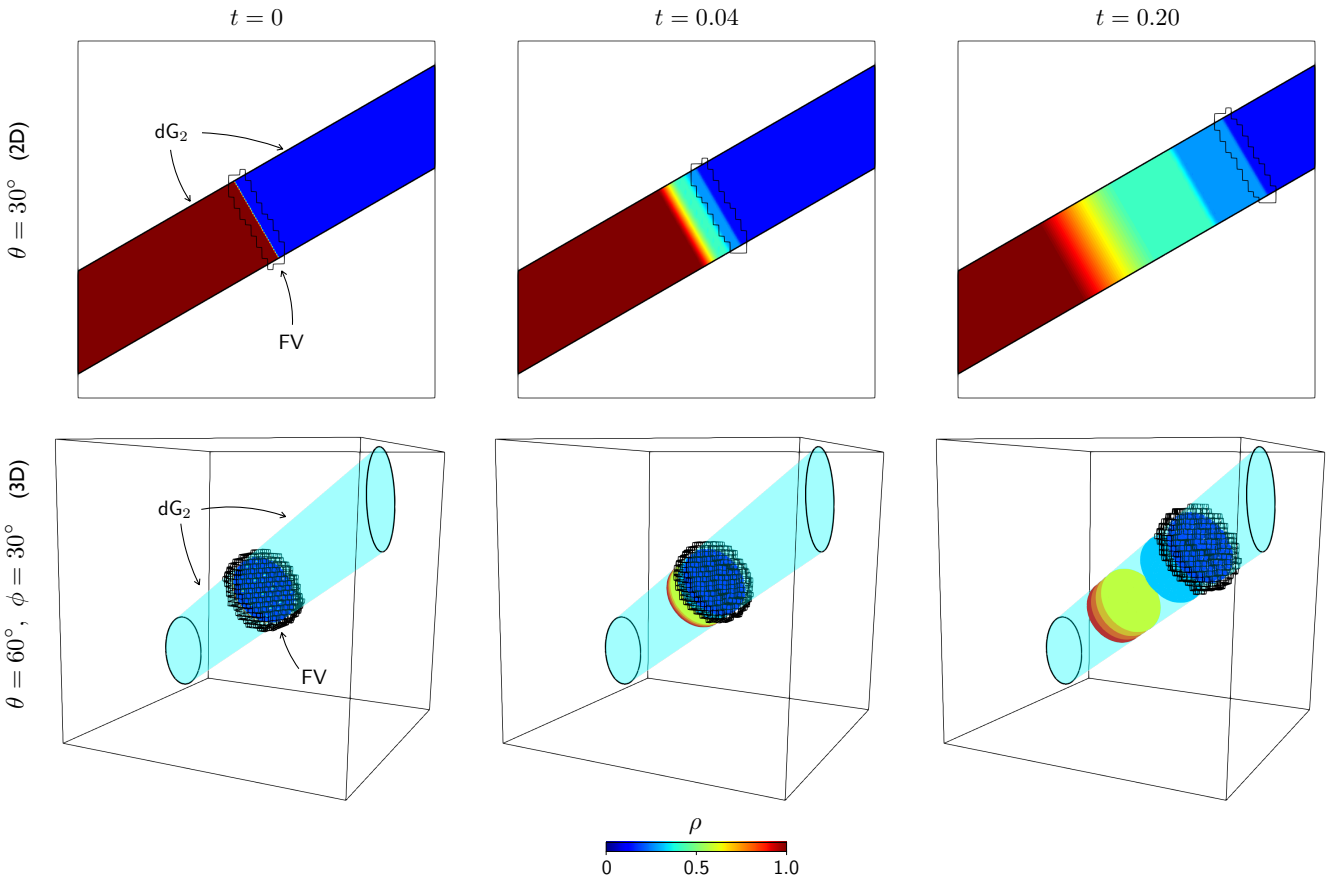


Figure 11: Density distribution at times $t = 0$, $t = 0.04$ and $t = 0.20$ for the 2D embedded Sod's shock tube problem (top row) and the 3D embedded shock tube problem (bottom row). The orientation angles $\theta = 30°$, in 2D, and $\theta = 60°$ and $\phi = 30°$, in 3D, are those defined in Fig.(10). For the 3D case, the density distribution is represented by isosurfaces of $\rho = \{0.15, 0.3, 0.6, 0.7, 0.8, 0.9\}$.

(in 2D) or $\hat{\boldsymbol{\xi}}\hat{\boldsymbol{\eta}}\hat{\boldsymbol{\zeta}}$ (in 3D) centered in $\boldsymbol{c}$, where $\hat{\boldsymbol{\xi}}$ is aligned with the tube's centerline and the unit vectors have the following components

$$\{\hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\eta}}\} = \left\{ \begin{array}{cc} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{array} \right\} \quad \text{or} \quad \{\hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\zeta}}\} = \left\{ \begin{array}{ccc} \cos\theta\cos\phi & -\sin\theta & -\cos\theta\sin\phi \\ \sin\theta\cos\phi & \cos\theta & -\sin\theta\sin\phi \\ \sin\phi & 0 & \cos\phi \end{array} \right\} \tag{34}$$

in 2D or 3D, respectively. Then, a convenient way to represent the domain and its boundaries is through the level set function $\boldsymbol{\Phi}(\boldsymbol{x}) \equiv \eta^2 - r^2$ in 2D, or $\boldsymbol{\Phi}(\boldsymbol{x}) \equiv \eta^2 + \zeta^2 - r^2$ in 3D, where ξ, η and ζ are the components of $\boldsymbol{x}$ in the rotated coordinate system.

The initial conditions are given by

$$\boldsymbol{U}(t=0,\boldsymbol{x}) = \left\{ \begin{array}{ll} \boldsymbol{U}_0^L & \text{if } (x_l - c_l)\hat{\xi}_l \leq 0 \\ \boldsymbol{U}_0^R & \text{if } (x_l - c_l)\hat{\xi}_l > 0 \end{array} \right. , \quad \text{where} \quad \boldsymbol{U}_0^L = \left\{ \begin{array}{c} 1 \\ \boldsymbol{0} \\ \frac{1}{\gamma-1} \end{array} \right\} \quad \text{and} \quad \boldsymbol{U}_0^R = \left\{ \begin{array}{c} 0.125 \\ \boldsymbol{0} \\ \frac{0.1}{\gamma-1} \end{array} \right\}, \tag{35}$$

and the final time is $T = 0.2$. Wall boundary conditions are prescribed along the embedded boundaries. At the intersection between the domain and the background cube's boundary, ghost states with the initial conditions are prescribed at the quadrature points.

The solution to Sod's shock tube problem consists of three waves: a rarefaction wave, a contact discontinuity and a shock wave. To account for the presence of the discontinuities, a two-level mesh is employed. Level $\ell = 0$ is obtained by subdividing the background domain into 64^d cells and uses a dG$_2$ scheme. Level $\ell = 1$ is constructed using a refinement ratio of 4 and is dynamically updated from level 0 using the shock sensor described in Sec.(3.4.1). Specifically, at time t, the smoothness indicator s^e given in Eq.(27) is computed for each element of the implicitly-defined mesh at level 0; then, the elements satisfying the condition (28) are tagged for refinement to level 1. In

Eq.(28), $\kappa_s = -2.00$ for the 2D problem and $\kappa_s = -2.45$ for the 3D problem. Finally, the implicitly-defined mesh is generated using volume fraction thresholds $\overline{\nu}_0 = \overline{\nu}_1 = 0.3$ in 2D and $\overline{\nu}_0 = \overline{\nu}_1 = 0.15$ in 3D. It is worth pointing out that the implicitly-defined elements are simple polygons in 2D, whereas they have curved surfaces in 3D. However, as shown in the results, this will have very little effect on the computed numerical solutions.

The top row of images in Fig.(11) shows the density distribution at times $t = 0$, $t = 0.04$ and $t = 0.20$ for the 2D tube corresponding to the orientation $\theta = 30°$. The bottom row of images in Fig.(11) shows the density distribution at the same times for the 3D tube corresponding to the orientation $\theta = 60°$ and $\phi = 30°$. In 3D, the density distribution is represented by isosurfaces of $\rho = \{0.15, 0.3, 0.6, 0.7, 0.8, 0.9\}$, which allow us to see the rarefaction wave, the contact discontinuity and the shock along the tube. Figure (11) also displays the evolution of the mesh level using the FV scheme, whose location is identified by the stepped black lines. The figure shows that the FV level tracks the shock discontinuity but not the contact discontinuity. Although it is possible to adjust the constant κ_s in (28) in such a way as to make the FV level track the contact discontinuities, numerical experiments show that these discontinuities are only temporarily tracked by the FV scheme and, once they are sufficiently smoothed, they stop satisfying the condition (28) and start being tracked by the unlimited dG scheme. This is an expected behavior since shocks steepen while contact discontinuities do not.

The robustness of the proposed approach is tested by considering four orientations each in 2D and 3D while keeping the same settings for the implicitly-defined mesh and the shock sensor. Figure (12) shows the results at the final time $t = 0.2$ for orientation angles $\theta = 0°$, $30°$, $45°$ and $60°$, in 2D, and for orientation angles $\{\theta, \phi\} = \{0°, 0°\}$, $\{0°, 30°\}$, $\{45°, 60°\}$ and $\{60°, 30°\}$, in 3D. The first row shows the density distribution for the four 2D configurations, the second row shows the corresponding pressure distribution, the third row shows the density distribution for the four 3D configurations and the fourth row shows the corresponding pressure. For the 3D cases, the density distribution is represented by isosurfaces of $\rho = \{0.15, 0.3, 0.6, 0.7, 0.8, 0.9\}$ while the pressure distribution is represented by isosurfaces of $p = \{0.15, 0.6, 0.7, 0.8, 0.9\}$. For all configurations, the coupled dG-FV scheme is able to reproduce the one-dimensional flow in embedded curved geometries that are not aligned with the background grid. This can be seen more clearly by looking at line plots of the density and pressure distribution along the centerline and the wall for the embedded shock tubes at the final time $t = 0.2$. Figures (13a), (13b), (13c) and (13d) show the density along the centerline, the pressure along the centerline, the density along the wall and the pressure along the wall, respectively, as functions of the local reference system coordinate ξ and the 2D tube's orientation. Similarly, Figs.(14a), (14b), (14c) and (14d) show the same quantities as functions of the local reference system coordinate ξ and the 3D tube's orientation. As seen in the figures, the rarefaction wave, the contact discontinuity and the shock wave are all well captured by the proposed approach and match the one-dimensional solution well, regardless of the orientation of the embedded geometry. Moreover, the numerical results in the tilted geometries are nearly indistinguishable from the numerical results in the aligned geometry, which can be viewed as a reference solution where the implicitly-defined mesh has little effect.

Finally, we note in Fig.(10), the wall values in 2D are those computed at $\eta = r$, i.e. at the upper embedded boundary of the tube, whereas the wall values in 3D are those computed at $\eta = \zeta = \tilde{r} \equiv r/\sqrt{2}$, i.e. at one line on the embedded boundary of the tube. Similar results are observed if the wall values are evaluated at other boundary locations.

4.3 Shock reflection

In the third test, we consider a shock reflection from a cylinder in 2D and a shock reflection from a sphere in 3D. These problems have been investigated experimentally [65] and allow us to assess the capabilities of numerical methods to capture the shock structure as well as the details of the smooth flow patterns.

Shock reflection from a cylinder in 2D is defined in the background unit square $\mathscr{R} = [0, 1]^2$. The geometry is represented by the level set function $\Phi(\boldsymbol{x}) \equiv r^2 - (x_k - c_k)(x_k - c_k)$ where $\boldsymbol{c} = \{0.5, 0.5\}$ and $r = 0.1$ are the center and the radius of the cylinder, respectively as shown in Fig.(15a). The shock is initially located at $x_1 = d_s = 0.2$ and travels towards the cylinder at a mach number $M = 2.81$. The initial conditions are given by

$$\boldsymbol{U}(t = 0, \boldsymbol{x}) = \begin{cases} \boldsymbol{U}_0^L & \text{if } x_1 \leq d_s \\ \boldsymbol{U}_0^R & \text{if } x_1 > d_s \end{cases}, \quad \text{where} \quad \boldsymbol{U}_0^R = \begin{Bmatrix} 1 \\ \boldsymbol{0} \\ \frac{1}{\gamma - 1} \end{Bmatrix} \tag{36}$$

where $\boldsymbol{U}_0^L$ is derived from the shock jump relations [56]. Wall boundary conditions are prescribed on the boundary of the cylinder and at $x_2 = 0$ and $x_2 = 1$; inflow boundary conditions are prescribed at $x_1 = 0$ and outflow boundary conditions are prescribed at $x_2 = 1$. The final time of the simulation is $T = 0.2$.

We use a four-level mesh with a refinement ratio of four at each level. Level $\ell = 0$ uses a dG_1 scheme and a mesh size $h_0 = 1/64$; level $\ell = 1$ uses a dG_3 scheme and is generated from level 0 using $\kappa_0^\rho = 0.5$ in (26); level $\ell = 2$

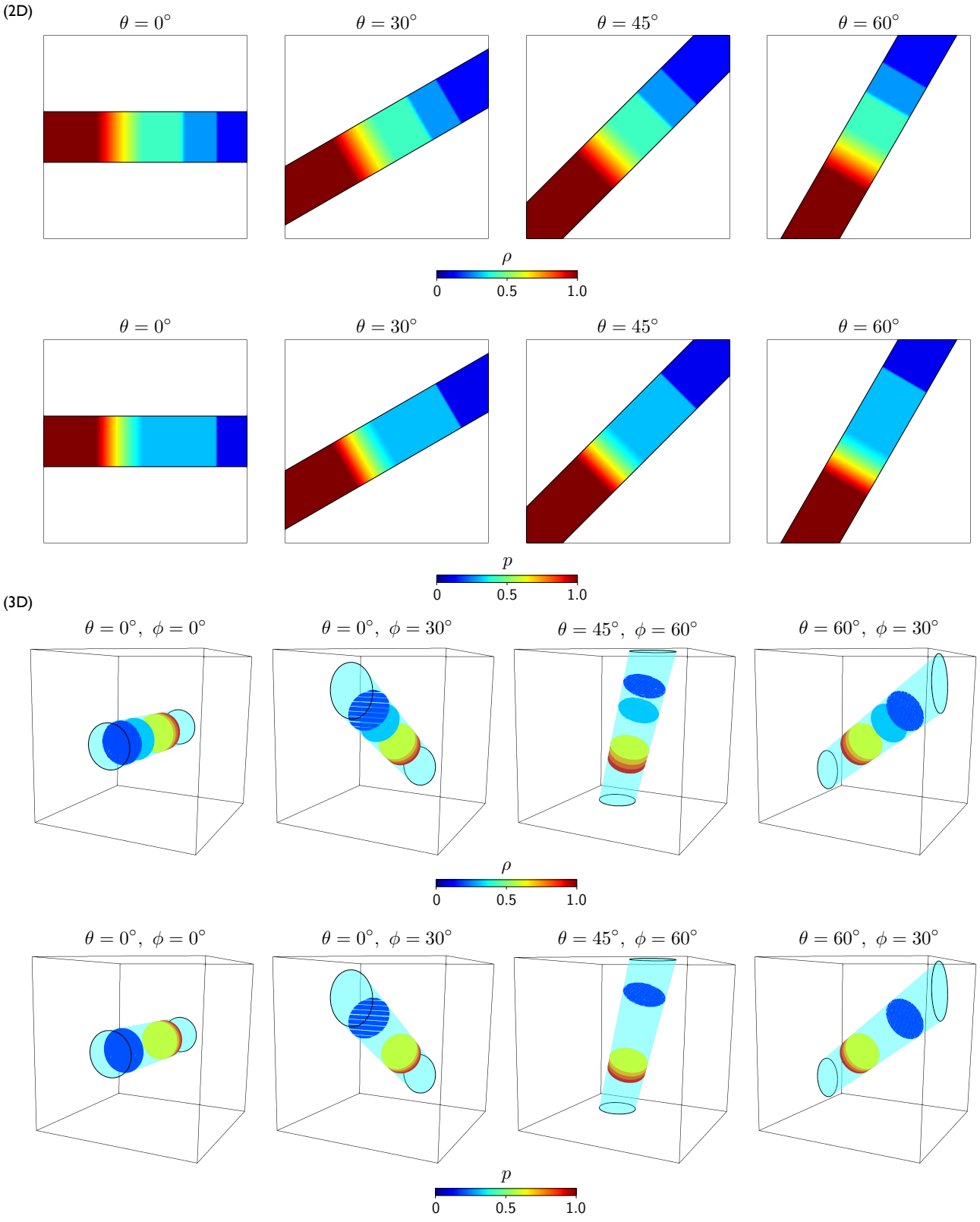
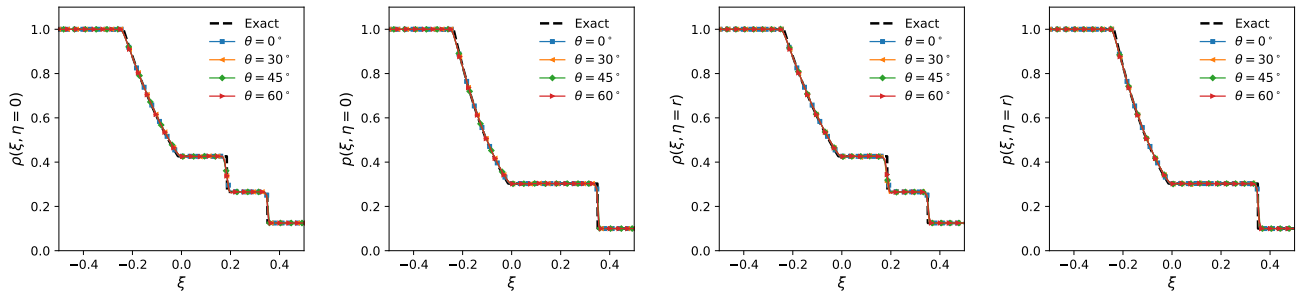
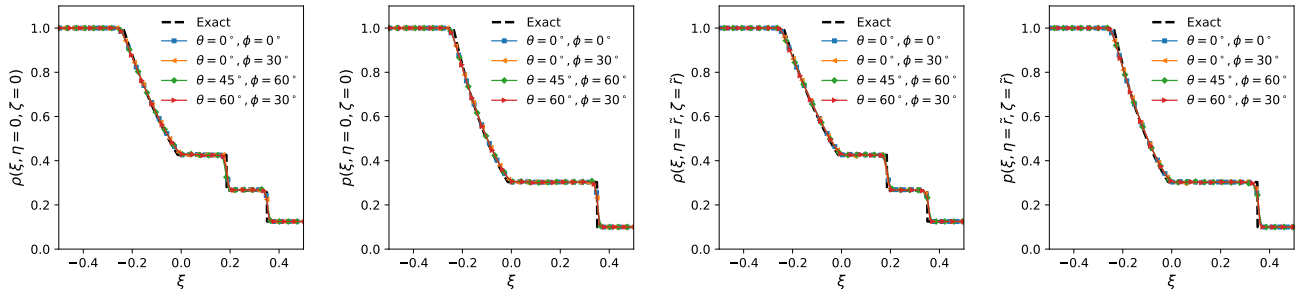


Figure 12: Density and pressure distribution for the eight configurations of the embedded shock tube problem at the final time $t = 0.2$. From the top: (first row) density distribution in 2D, (second row) pressure distribution in 2D, (third row) density distribution in 3D and (fourth row) pressure distribution in 3D. For the 3D cases, the density distribution is represented by isosurfaces of $\rho = \{0.15, 0.3, 0.6, 0.7, 0.8, 0.9\}$ and the pressure distribution is represented by isosurfaces of $p = \{0.15, 0.6, 0.7, 0.8, 0.9\}$.



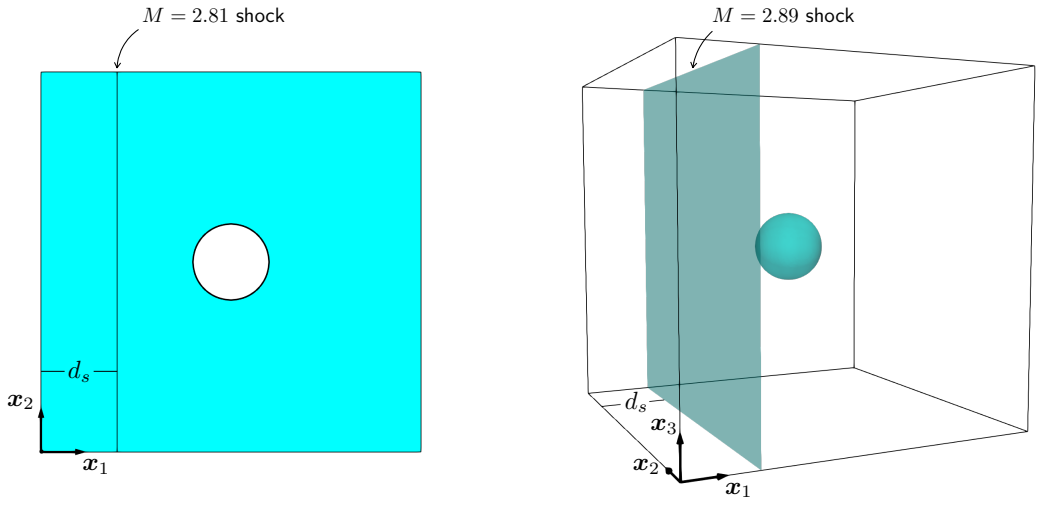
(a) (b) (c) (d)

Figure 13: Distribution of density and pressure at the final time $t = 0.2$ as functions of the orientation of the 2D Sod's shock tube; (a) density along the centerline, (b) pressure along the centerline, (c) density along the wall and (d) pressure along the wall.



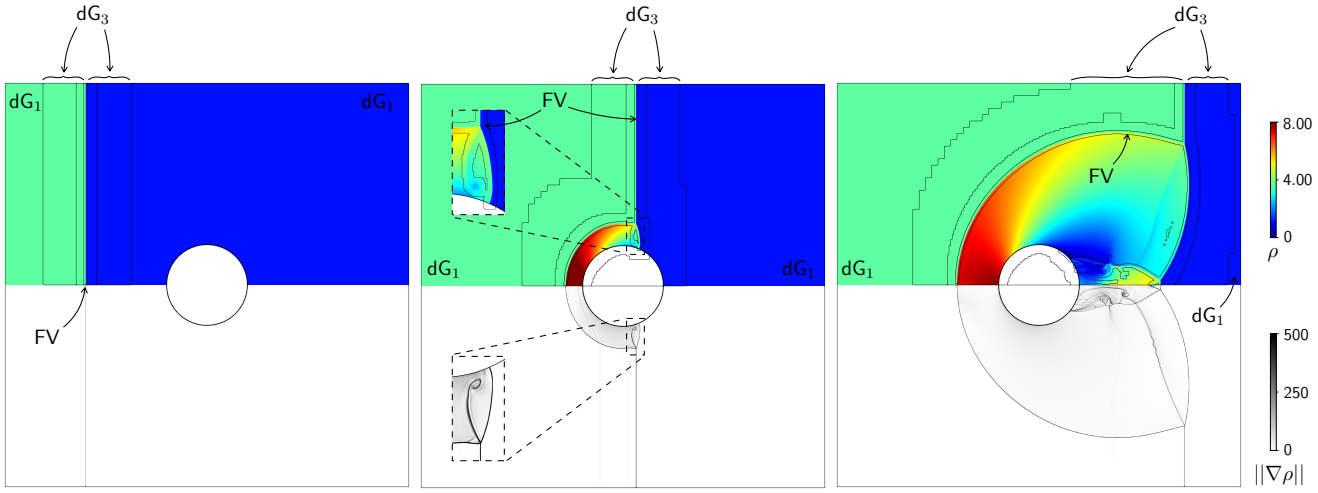
(a) (b) (c) (d)

Figure 14: Distribution of density and pressure at the final time $t = 0.2$ as functions of the orientation of the 3D Sod's shock tube; (a) density along the centerline, (b) pressure along the centerline, (c) density along the wall and (d) pressure along the wall.



(a) (b)

Figure 15: Illustration of (a) the problem of a shock reflecting from a cylinder and (b) the problem of a shock reflecting from a sphere.



$t = 0$ (a) $t = 0.10$ (b) $t = 0.20$ (c)

Figure 16: Distribution of the density ρ and the density gradient magnitude $||\nabla\rho||$ at times (a) $t = 0$, (b) $t = 0.1$ and (c) $t = 0.2$ for a shock reflecting from a cylinder. The inset in figure (b) shows a close-up of the contact discontinuity and the Mach stem. In the images, the stepped black lines denote the outline of the AMR levels.

uses a dG_3 scheme and is generated from level 1 using $\kappa_1^\rho = 2.0$ in (26); level $\ell = 3$ uses the FV scheme and is generated from level 2 using $\kappa^s = -3.25$ in (28). Finally, cell merging is triggered by the volume fraction thresholds $\overline{\nu}_0 = \overline{\nu}_1 = \overline{\nu}_2 = \overline{\nu}_3 = 0.3$.

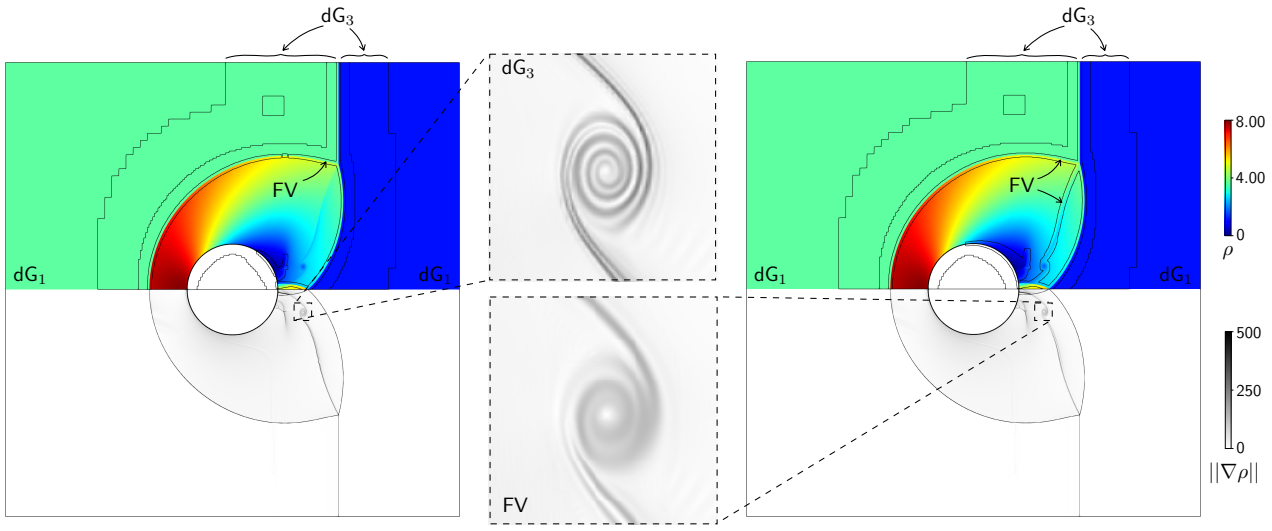
Figure (16) shows contour plots of the density ρ and the density gradient magnitude $||\nabla\rho||$ at $t = 0$, $t = 0.1$ and $t = 0.2$. The figure also show the configurations of the AMR levels and how they adapt to the location of the travelling discontinuities. By looking at Fig.(16b), one can see how the FV level tracks the shock reflecting from the cylinder as well as the shock attached to the surface. Similarly, once the incident shock has moved past the cylinder (Fig.(16c)), the FV level remains attached to the cylinder's surface to track the shock in the wake.

Figure (16) also illustrates the difference in how shocks and contact discontinuities are resolved during the simulation. In particular, by comparing Fig.(16b) and Fig.(16c), one can see that shocks are always resolved by the FV scheme, whereas the contact discontinuity is initially tracked by the FV scheme but, once smoothed, it starts being tracked by the dG schemes. This is consistent to the results of the embedded shock tube test presented in Sec.(4.2). To illustrate the benefits of using high-order schemes versus low-order schemes, Fig.(17a) shows the density and the density gradient magnitude at time $t = 0.16$ for the simulation described above, whereas Fig.(17b) shows the density and the density gradient magnitude at time $t = 0.16$ but with $\kappa^s = -4.00$. A lower value of κ^s increase the number elements where a discontinuity is sensed and, for this test case, causes the contact discontinuity to be tracked by the FV scheme. By comparing the insets of Fig.(17a) and Fig.(17b), it is clear that the dG_3 scheme provides better resolution of the rollup of the contact discontinuity than the solution obtained with the FV scheme.

Finally, Fig.(18) shows the comparison between the Schlieren photograph from Ref.[65] of a shock reflection from a cylinder and the results computed at $t = 0.14$. As seen in the figure, the *hp*-AMR strategy allows us to capture all the details of the flow structures observed in the experiments.

The case of a shock reflection from a sphere is modelled in 3D as shown in Fig.(15b). The problem is defined in the background unit cube $\mathscr{R} = [0, 1]^3$. The geometry is represented by the level set function $\Phi(\boldsymbol{x}) \equiv r^2 - (x_k - c_k)(x_k - c_k)$ where $\boldsymbol{c} = \{0.5, 0.5, 0.5\}$ and $r = 0.1$ are the center and the radius of the sphere, respectively. The shock is initially located at $x_1 = d_s = 0.2$ and travels towards the sphere at a mach number $M = 2.89$. The initial conditions are given by Eq.(36). Wall boundary conditions are prescribed on the boundary of the sphere and at $x_2 = 0$, $x_2 = 1$, $x_3 = 0$ and $x_3 = 1$; inflow boundary conditions are prescribed at $x_1 = 0$ and outflow boundary conditions are prescribed at $x_2 = 1$. The final simulation time is $T = 0.2$.

We use a four-level mesh with a refinement ratio of four between the levels. Level $\ell = 0$ uses a dG_1 scheme and and a mesh size $h_0 = 1/8$; level $\ell = 1$ uses a dG_3 scheme and is generated from level 0 using $\kappa_0^\rho = 0.5$ in (26); level $\ell = 2$ uses a dG_3 scheme and is generated from level 1 using $\kappa_1^\rho = 2.0$ in (26); level $\ell = 3$ uses the FV scheme and is generated from level 2 using $\kappa^s = -3.5$ in (28). Cell merging is triggered by the volume fraction thresholds $\overline{\nu}_0 = \overline{\nu}_1 = \overline{\nu}_2 = \overline{\nu}_3 = 0.15$.



(a) (b)

Figure 17: Distribution of the density ρ and the density gradient magnitude $||\nabla\rho||$ computed at time $t = 0.16$ using different values of the threshold parameter κ^s entering the shock-sensor condition (28): (a) $\kappa^s = -3.25$ and (b) $\kappa^s = -4.00$. The insets show a close-up on the vortex of the contact discontinuity.

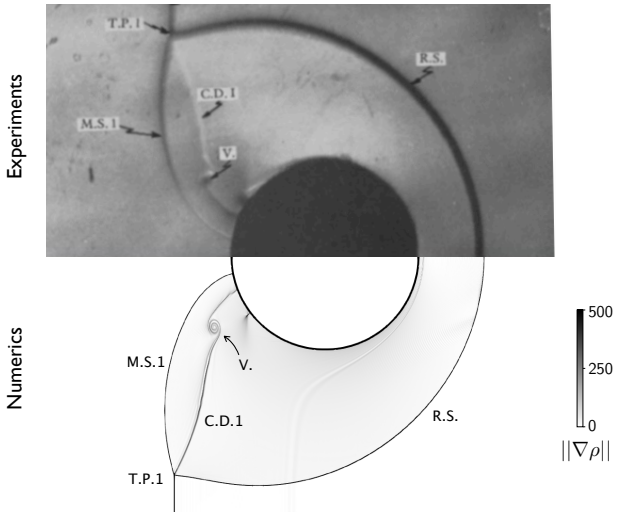


Figure 18: Comparison between the experimental Schlieren photograph of Ref.[65] and the numerical solution at $t = 0.14$. All flow features observed experimentally, such as the reflected shock (R.S.), the primary Mach stem (M.S.1), the primary contact discontinuity (C.D.1), the primary triple point (T.P.1) and the vortex (V.), are well captured.

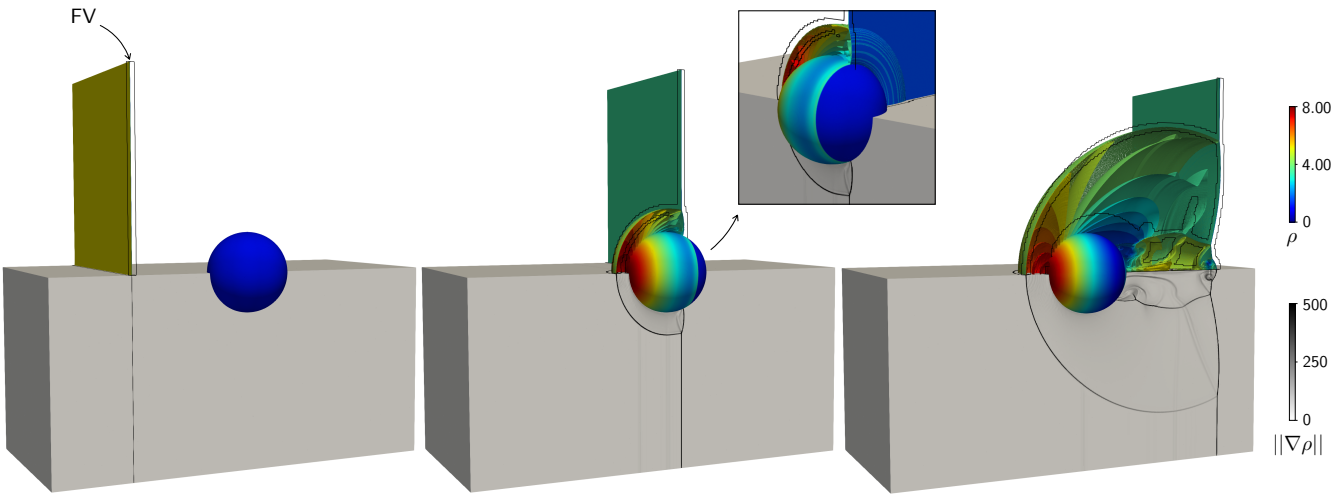


Figure 19: Distribution of the density ρ and the density gradient magnitude $||\nabla\rho||$ at the times $t = 0$, $t = 0.1$ and $t = 0.2$ for a shock reflecting from a sphere. The inset shows a closeup on the shock propagating on the sphere surface at $t = 0.1$.

Figure (19) shows the density ρ and the density gradient magnitude $||\nabla\rho||$ at $t = 0$, $t = 0.1$ and $t = 0.2$. In the figure, the density on the surface of the sphere is represented by a continuous contour plot and, within the fluid domain, by isosurfaces evaluated at $\rho = \{1.5, 2.0, \ldots, 7.5\}$. The figure also shows the trace of the FV level on the planes $x_2 = 0.5$ and $x_3 = 0.5$; for the sake of visualization the AMR levels $\ell = 0$, 1 and 2 are not displayed. Similar to 2D, one can see how the FV level tracks the reflected shock.

The comparison between experimental observations and the numerical results is shown in Figs.(20a) and (20b), which show the Schlieren photograph fom Ref.[65] of shock reflection from a sphere and the results at $t = 0.16$. Although small differences likely due to viscous effects can be seen inside the wake behind the sphere, the *hp*-AMR strategy captures all the details of the flow structures observed in the experiments.

4.4 Shock reflection from a convex-concave surface

The final example is another shock reflection problem consisting of a shock reflecting from a convex-concave wall as illustrated in Fig.(21). Shocks reflecting from curved surfaces have been recently investigated experimentally and numerically [66, 67, 68] because of their importance in shock-focusing applications. Here, we model the experimental test of Ram et al.[66], who provided highly resolved measurements of the transition from regular to Mach reflection for the problem depicted in Fig.(21).

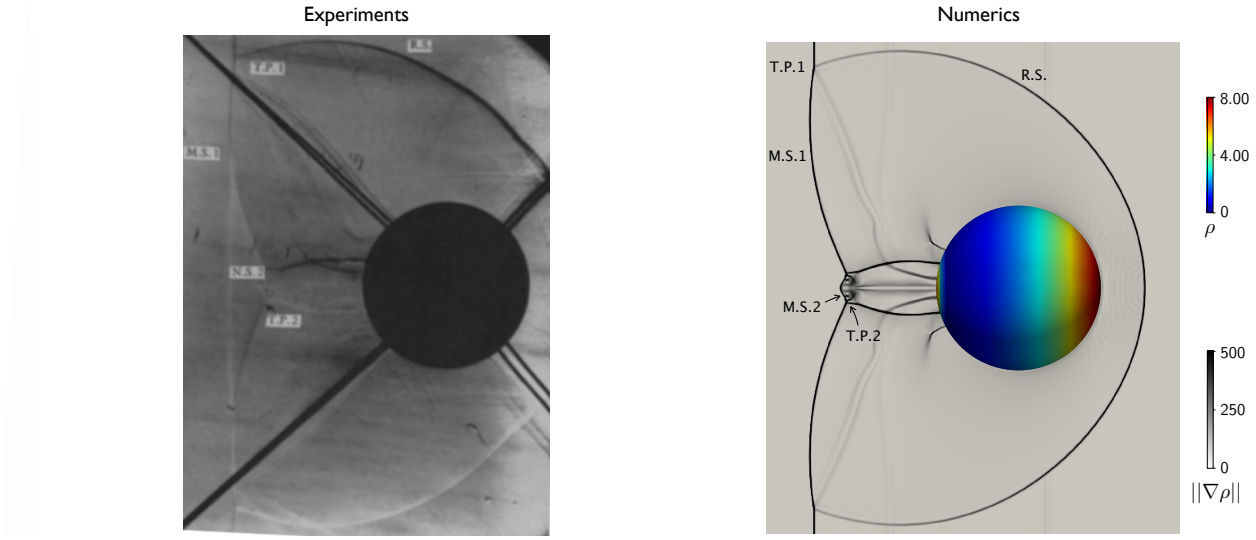
The problem is defined on the background rectangle $\mathscr{R} = [-80, 122.5] \times [0, 120]$. The height of the numerical domain is larger than the height of the observation region in the experiments to avoid any reflection from the top boundary. The convex-concave wall consists of two surfaces generated by two circular arcs and a flat surface. With reference to Fig.(21), the radius of the cylinders is $r = 60.6$, while the location of their centers is given by $\boldsymbol{c}_a \equiv \{r\sin\theta, -r\cos\theta\}$ and $\boldsymbol{c}_b \equiv \{r\sin\theta, r(2-\cos\theta)\}$ with $\theta = 65°$. Then, the geometry is defined by the level set function

$$\boldsymbol{\Phi}(\boldsymbol{x}) \equiv \begin{cases} d_2(\boldsymbol{x}) & \text{if } x_1 > c_{b1} \wedge x_2 < c_{b2} \\ \max\{d_1(\boldsymbol{x}), d_3(\boldsymbol{x})\} & \text{otherwise} \end{cases}, \tag{37}$$

where $d_1(\boldsymbol{x}) \equiv r - \sqrt{(x_k - c_{ak})(x_k - c_{ak})}$, $d_2(\boldsymbol{x}) \equiv \sqrt{(x_k - c_{bk})(x_k - c_{bk})} - r$ and $d_3(\boldsymbol{x}) \equiv x_1 - c_{b1} - r$.

The shock is initially located at $x_1 = 0$. Following the experiments [66], we consider three shock Mach numbers, $M = 1.19$, $M = 1.30$ and $M = 1.41$. The initial conditions are given by Eq.(36) where $d_s = 0$. Wall boundary conditions are prescribed on the convex-concave surface and at the top boundary of the background rectangle, whereas inflow boundary conditions are prescribed at $x_1 = -80$. The final time of the simulation is chosen to make sure the shock does not reach the flat wall.

We use a four-level mesh with a refinement ratio of four between levels. Level $\ell = 0$ uses a dG_1 scheme with mesh size $h = 3.75$; level $\ell = 1$ uses a dG_3 scheme and is generated from level 0 using $\kappa_0^\rho = 0.005$ in (26); level



(a) (b)

Figure 20: Comparison between the experimental Schlieren photograph from Ref.[65] and the numerical solution at $t = 0.16$. The flow features observed experimentally, such as the reflected shock (R.S.), the primary Mach stem (M.S.1), the secondary Mach stem (M.S.2), the primary triple point (T.P.1) and the secondary triple point (T.P.2), are well captured.

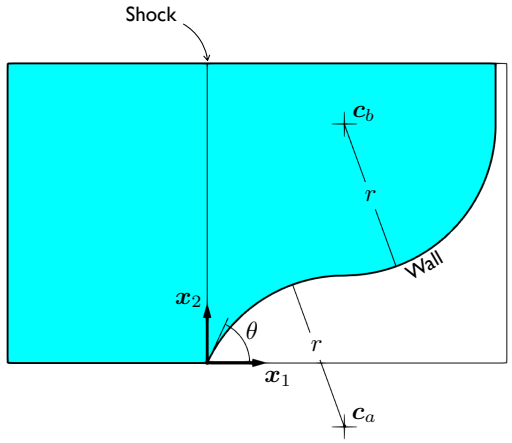


Figure 21: Illustration of the problem of a shock reflecting from a convex-concave wall.

$\ell = 2$ uses a dG_3 scheme and is generated from level 1 using $\kappa_1^\rho = 0.02$; level $\ell = 3$ uses the FV scheme and is generated from level 2 using $\kappa^s = -4.00$ in (28). Finally, cell merging is triggered by the volume fraction thresholds $\overline{\nu}_0 = \overline{\nu}_1 = \overline{\nu}_2 = \overline{\nu}_3 = 0.3$.

Figure (22) show the distribution of the density ρ at selected times for the different Mach numbers. In the figure, the inset of each image shows a close-up of the region where the shock is attached to the curved surface and allows us to distinguish the different shock structures. For instance, for the $M = 1.19$ case reported in the left column of Fig.(22), the reflection structure transitions from regular reflection at $t = 12$ to Mach reflection at $t = 30$. Similarly, the last row of Fig.(22) highlights the formation of the secondary triple point. The images also show the outline of the FV level, which is able to track the shock in all its difference configurations.

The different shock structures can be seen more clearly in Fig.(23), which shows the density gradient magnitude $||\nabla\rho||$ at the same times as shown in Fig.(22), allowing us to see the effect of the Mach number on the flow patterns. For instance, at $t = 12$, the flow with $M = 1.19$ is characterized by regular reflection, the flow with $M = 1.41$ is characterized by Mach reflection and the flow with $M = 1.30$ is transitioning between the two configurations.

Finally, using a sequence of snapshots such as those presented in Fig.(23), one can quantitatively compare the numerical results with the experimental measurements. Specifically, in Ref.[66], the Authors measured the distance d_w between from both the primary triple point and the secondary triple points to the reflecting surface and plotted d_w versus the angle of the reflecting surface at the foot of the Mach stem, θ_w. The comparison between the experiments and the numerics is shown in Figs.(24a), (24b) and (24c), which corresponds to the cases $M = 1.19$, $M = 1.30$, $M = 1.41$, respectively. As shown by the figures, the embedded-boundary *hp*-AMR framework is able to reproduce the experimental measurements with remarkable accuracy.

5 Conclusions and future work

We have presented a methodology for solving the equations of inviscid gas dynamics on structured grids with an embedded boundary representation of complex geometry. The novel features of the approach are the coupling of a discontinuous Galerkin method that provides high-order accuracy in regions of smooth flow with a Finite Volume method that provides robust shock-tracking capabilities of solution discontinuities through an *hp* adaptive mesh refinement strategy and an implicit mesh approach that enables high-order accurate resolution of the geometry. The methodology uses a compact-stencil AMR discretization strategy, is fully explicit and does not require the introduction of additional variables (e.g. the auxiliary flux in artificial-viscosity methods).

The methodology has been tested on embedded-geometry problems with and without solution discontinuities. The supersonic vortex problem of Sec.(4.1) demonstrates the high-order accuracy of dG schemes for smooth flows, while the remaining examples illustrate the performance of the method on problems with discontinuities in several configurations.

The method offers several avenues of improvements and further research. First, the current AMR strategy relies on the definition of two parameters, namely κ_ℓ^ρ and κ^s, entering Eqs.(26) and (27). Here, κ_ℓ^ρ is chosen to ensure that regions of constant fields are resolved with low-order dG schemes while the remaining parts of the domain where the flow is smooth are resolved with high-order dG schemes. In addition, κ^s is chosen so that contact discontinuities transition as soon as possible from being in treated the FV scheme to being treated by dG, while making sure shocks are always contained within the FV level. In the simulations, this criterion allowed high-order accuracy in the regions of smooth flow and robustly captured the travelling shocks but led to a problem-dependent choice of the threshold parameters. Therefore, more advanced conditions for the dynamic evolution of the *hp*-AMR levels, including more sophisticated shock sensors [69, 70, 71, 72], or a more robust treatment of the coupling between the FV and the dG schemes, using e.g. *a posteriori* limiters [73, 74, 75], might be beneficial.

Second, Eq.(29) is integrated in time using high-order TVD Runge-Kutta schemes, which are very efficient for many practical applications but are limited to fourth order accuracy. Methods such as the spectral deferred correction methods [76, 77, 78] or ADER schemes [79, 80, 74, 75] could be implemented within the present framework to obtain space-time orders of accuracy that are higher than the ones considered here.

The numerical test cases considered in Sec.(4) used geometries that could be described by relatively simple level set functions. This does not represent a fundamental limitation of the proposed framework, which could be employed to model more complex geometries, e.g. including sharp edges or corners, provided that the corresponding high-order quadrature rules are available. In fact, it is worth pointing out that even the simple three-dimensional embedded geometries considered in Sec.(4) induce very complex configurations of cut and merged elements, which are robustly handled by present approach in both static and dynamic AMR configurations. However, implicitly-defined meshes where cells are split by embedded boundaries that are smaller than the cell size would required doubly-value cells, which are currently not supported.

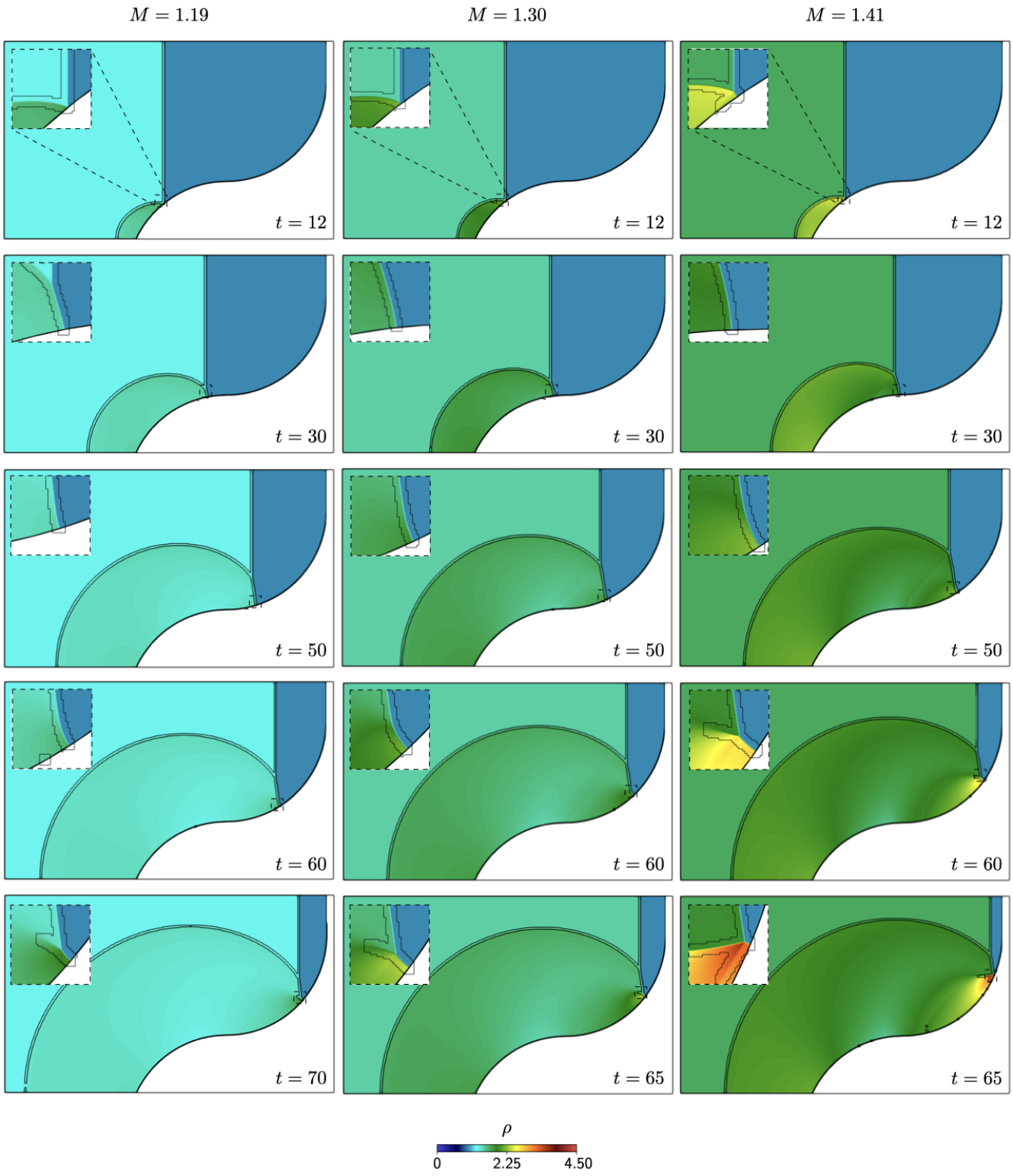


Figure 22: Density distribution at selected times for shock reflection from a convex-concave wall at different Mach numbers: (left column) $M = 1.19$, (center column) $M = 1.30$ and (right column) $M = 1.41$. In each image, the inset shows a close-up of the region where the shock is attached to the curved surface. The time is reported in the bottom right corner of each image.

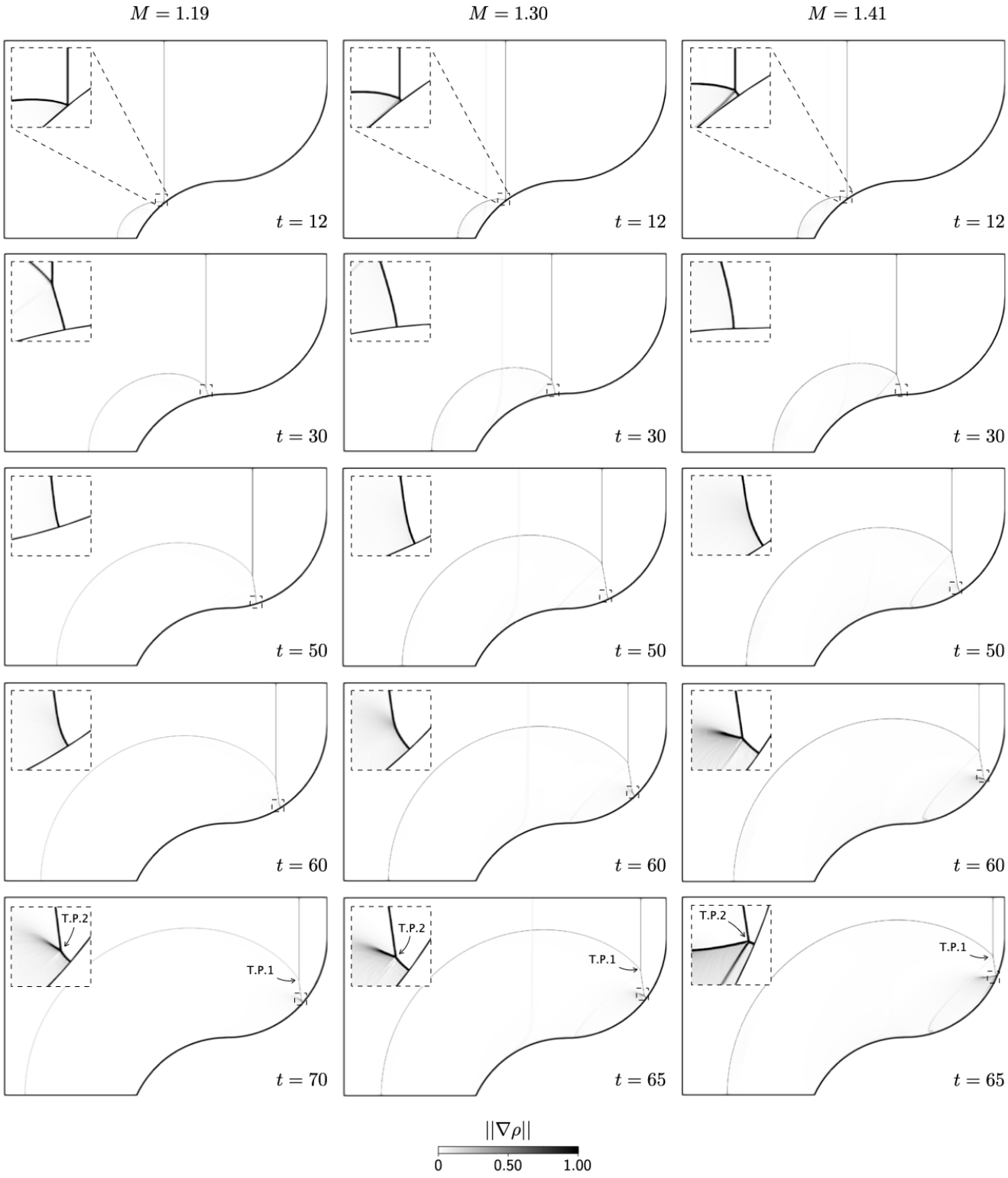
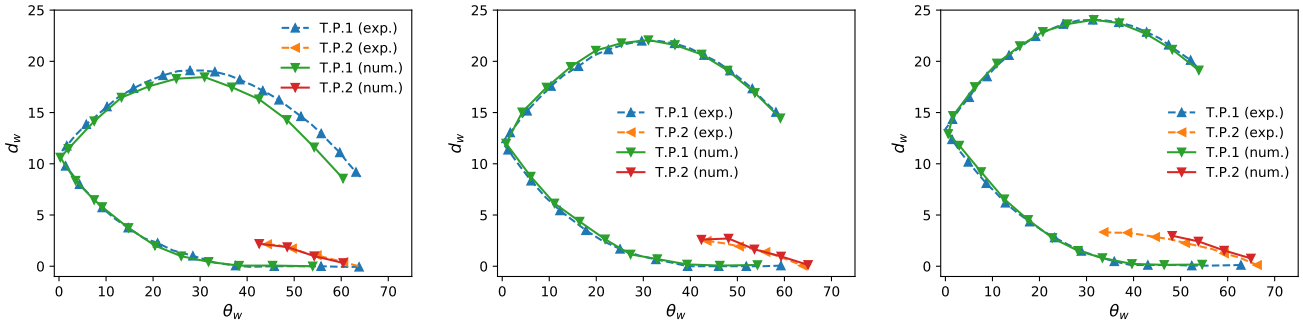


Figure 23: Density gradient distribution at selected times for shock reflection from a convex-concave wall at different Mach numbers: (left column) $M = 1.19$, (center column) $M = 1.30$ and (right column) $M = 1.41$. In each image, the inset shows a close-up of the region where the shock is attached to the curved surface. The time is reported in the bottom right corner of each image.



(a) (b) (c)

Figure 24: Plot of the distance d_w of the triple points from the wall versus the inclination θ_w of wall surface at the foot of the Mach stem: (a) $M = 1.19$, (b) $M = 1.30$, and (c) $M = 1.41$. In the legend of the figures, exp. denotes the experimental measurements and num. denotes the numerical results, while T.P.1 and T.P.2 refer to the results of the primary and secondary triple points, respectively, as indicated in the last row of Fig.(23). Experimental results are taken from Ref.[66].

The methodology was implemented using `AMReX` [62], an exascale-ready software framework for massively parallel, block-structured adaptive mesh refinement applications. The computations were performed using classic MPI parallelization and future research will provide a comprehensive scalability analysis involving also the use of modern accelerators, such as general-purpose graphical processing units.

Future lines of research will also include the addition of viscous and reaction terms to the governing equations (2) and the implementation of the corresponding dG and/or FV schemes. This will provide high-order numerical schemes for more complex phenomena, such as combustion, where high-order resolution of curved geometries would be desirable. Finally, since the methodology already features dynamic adaptivity and dynamic generation of the implicit mesh, the extension to moving geometry would be relatively straightforward. The moving geometry could either be a prescribed motion or it could be computed as part of the evolution of the system by evolving different dynamics on either side of the interface.

Acknowledgements

We thank Dr. Robert Saye for the support in the use of `Algoim` (`https://github.com/algoim/algoim`) and Dr. Weiqun Zhang for the support in the use of `AMReX` (`https://amrex-codes.github.io/amrex/`). The work was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, through U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231; and resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Aftosmis, M., Berger, M., and Adomavicius, G. A parallel multilevel method for adaptively refined cartesian grids with embedded boundaries. In *38th Aerospace Sciences Meeting and Exhibit*, page 808, 2000.

[2] Peskin, C. S. The immersed boundary method. *Acta numerica*, 11:479–517, 2002.

[3] Mokbel, D., Abels, H., and Aland, S. A phase-field model for fluid–structure interaction. *Journal of Computational Physics*, 372:823–840, 2018.

[4] Kemm, F., Gaburro, E., Thein, F., and Dumbser, M. A simple diffuse interface approach for compressible

flows around moving solids of arbitrary shape based on a reduced baer–nunziato model. *Computers & Fluids*, 204:104536, 2020.

[5] Moës, N., Dolbow, J., and Belytschko, T. A finite element method for crack growth without remeshing. *International journal for numerical methods in engineering*, 46(1):131–150, 1999.

[6] Qin, R. and Krivodonova, L. A discontinuous galerkin method for solutions of the euler equations on cartesian grids with embedded geometries. *Journal of Computational Science*, 4(1-2):24–35, 2013.

[7] Alauzet, F., Fabrèges, B., Fernández, M. A., and Landajuela, M. Nitsche-xfem for the coupling of an incompressible fluid with immersed thin-walled structures. *Computer Methods in Applied Mechanics and Engineering*, 301:300–335, 2016.

[8] Saye, R. Implicit mesh discontinuous galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part i. *Journal of Computational Physics*, 344:647–682, 2017.

[9] Saye, R. Implicit mesh discontinuous galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part ii. *Journal of Computational Physics*, 344:683–723, 2017.

[10] Milazzo, A., Benedetti, I., and Gulizzi, V. An extended ritz formulation for buckling and post-buckling analysis of cracked multilayered plates. *Composite Structures*, 201:980–994, 2018.

[11] Berger, M. Cut cells: Meshes and solvers. In *Handbook of Numerical Analysis*, volume 18, pages 1–22. Elsevier, 2017.

[12] LeVeque, R. J. et al. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[13] Versteeg, H. K. and Malalasekera, W. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.

[14] Clarke, D. K., Salas, M., and Hassan, H. Euler calculations for multielement airfoils using cartesian grids. *AIAA journal*, 24(3):353–358, 1986.

[15] Gaffney, JR, R. and Hassan, H. Euler calculations for wings using cartesian grids. In *25th AIAA Aerospace Sciences Meeting*, page 356, 1987.

[16] Hartmann, D., Meinke, M., and Schröder, W. An adaptive multilevel multigrid formulation for cartesian hierarchical grid methods. *Computers & Fluids*, 37(9):1103–1125, 2008.

[17] Ji, H., Lien, F.-S., and Yee, E. Numerical simulation of detonation using an adaptive cartesian cut-cell method combined with a cell-merging technique. *Computers & fluids*, 39(6):1041–1057, 2010.

[18] Hartmann, D., Meinke, M., and Schröder, W. A strictly conservative cartesian cut-cell method for compressible viscous flows on adaptive grids. *Computer Methods in Applied Mechanics and Engineering*, 200(9-12):1038–1052, 2011.

[19] Cecere, D. and Giacomazzi, E. An immersed volume method for large eddy simulation of compressible flows using a staggered-grid approach. *Computer Methods in Applied Mechanics and Engineering*, 280:1–27, 2014.

[20] Muralidharan, B. and Menon, S. A high-order adaptive cartesian cut-cell method for simulation of compressible viscous flow over immersed bodies. *Journal of Computational Physics*, 321:342–368, 2016.

[21] Pember, R. B., Bell, J. B., Colella, P., Curtchfield, W. Y., and Welcome, M. L. An adaptive cartesian grid method for unsteady compressible flow in irregular regions. *Journal of computational Physics*, 120(2):278–304, 1995.

[22] Almgren, A. S., Bell, J. B., Colella, P., and Marthaler, T. A cartesian grid projection method for the incompressible euler equations in complex geometries. *SIAM Journal on Scientific Computing*, 18(5):1289–1309, 1997.

[23] Colella, P., Graves, D. T., Keen, B. J., and Modiano, D. A cartesian grid embedded boundary method for hyperbolic conservation laws. *Journal of Computational Physics*, 211(1):347–366, 2006.

[24] Graves, D., Colella, P., Modiano, D., Johnson, J., Sjogreen, B., and Gao, X. A cartesian grid embedded boundary method for the compressible navier–stokes equations. *Communications in Applied Mathematics and Computational Science*, 8(1):99–122, 2013.

[25] Helzel, C., Berger, M. J., and LeVeque, R. J. A high-resolution rotated grid method for conservation laws with embedded geometries. *SIAM Journal on Scientific Computing*, 26(3):785–809, 2005.

[26] Berger, M. and Helzel, C. A simplified h-box method for embedded boundary grids. *SIAM Journal on Scientific Computing*, 34(2):A861–A888, 2012.

[27] Gokhale, N., Nikiforakis, N., and Klein, R. A dimensionally split cartesian cut cell method for hyperbolic conservation laws. *Journal of Computational Physics*, 364:186–208, 2018.

[28] Berger, M. and Giuliani, A. A state redistribution algorithm for finite volume schemes on cut cell meshes. *Journal of Computational Physics*, 428:109820, 2021.

[29] Cockburn, B., Hou, S., and Shu, C.-W. The runge-kutta local projection discontinuous galerkin finite element method for conservation laws. iv. the multidimensional case. *Mathematics of Computation*, 54(190):545–581, 1990.

[30] Cockburn, B. and Shu, C.-W. The runge–kutta discontinuous galerkin method for conservation laws v: multidimensional systems. *Journal of Computational Physics*, 141(2):199–224, 1998.

[31] Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[32] Cockburn, B. Discontinuous galerkin methods for computational fluid dynamics. *Encyclopedia of Computational Mechanics Second Edition*, pages 1–63, 2018.

[33] Cangiani, A., Dong, Z., and Georgoulis, E. H. hp-version space-time discontinuous galerkin methods for parabolic problems on prismatic meshes. *SIAM Journal on Scientific Computing*, 39(4):A1251–A1279, 2017.

[34] Antonietti, P. F. and Pennesi, G. V-cycle multigrid algorithms for discontinuous galerkin methods on non-nested polytopic meshes. *Journal of Scientific Computing*, 78(1):625–652, 2019.

[35] Bastian, P. and Engwer, C. An unfitted finite element method using discontinuous galerkin. *International journal for numerical methods in engineering*, 79(12):1557–1576, 2009.

[36] Johansson, A. and Larson, M. G. A high order discontinuous galerkin nitsche method for elliptic problems with fictitious boundary. *Numerische Mathematik*, 123(4):607–628, 2013.

[37] Brandstetter, G. and Govindjee, S. A high-order immersed boundary discontinuous-galerkin method for poisson's equation with discontinuous coefficients and singular sources. *International Journal for Numerical Methods in Engineering*, 101(11):847–869, 2015.

[38] Gürkan, C. and Massing, A. A stabilized cut discontinuous galerkin framework for elliptic boundary value and interface problems. *Computer Methods in Applied Mechanics and Engineering*, 348:466–499, 2019.

[39] Lew, A. J. and Buscaglia, G. C. A discontinuous-galerkin-based immersed boundary method. *International Journal for Numerical Methods in Engineering*, 76(4):427–454, 2008.

[40] Heimann, F., Engwer, C., Ippisch, O., and Bastian, P. An unfitted interior penalty discontinuous galerkin method for incompressible navier–stokes two-phase flow. *International Journal for Numerical Methods in Fluids*, 71(3):269–293, 2013.

[41] Kummer, F. Extended discontinuous galerkin methods for two-phase flows: the spatial discretization. *International Journal for Numerical Methods in Engineering*, 109(2):259–289, 2017.

[42] Krause, D. and Kummer, F. An incompressible immersed boundary solver for moving body flows using a cut cell discontinuous galerkin method. *Computers & Fluids*, 153:118–129, 2017.

[43] Saye, R. Fast multigrid solution of high-order accurate multiphase stokes problems. *Communications in Applied Mathematics and Computational Science*, 15(2):147–196, 2020.

[44] Gulizzi, V., Benedetti, I., and Milazzo, A. An implicit mesh discontinuous galerkin formulation for higher-order plate theories. *Mechanics of Advanced Materials and Structures*, 27(17):1494–1508, 2020.

[45] Gulizzi, V., Benedetti, I., and Milazzo, A. A high-resolution layer-wise discontinuous galerkin formulation for multilayered composite plates. *Composite Structures*, 242:112137, 2020.

[46] Müller, B., Krämer-Eis, S., Kummer, F., and Oberlack, M. A high-order discontinuous galerkin method for compressible flows with immersed boundaries. *International Journal for Numerical Methods in Engineering*, 110(1):3–30, 2017.

[47] Geisenhofer, M., Kummer, F., and Müller, B. A discontinuous galerkin immersed boundary solver for compressible flows: Adaptive local time stepping for artificial viscosity–based shock-capturing on cut cells. *International Journal for Numerical Methods in Fluids*, 91(9):448–472, 2019.

[48] Fidkowski, K. J. and Darmofal, D. L. A triangular cut-cell adaptive method for high-order discretizations of the compressible navier–stokes equations. *Journal of Computational Physics*, 225(2):1653–1672, 2007.

[49] Xiao, H., Febrianto, E., Zhang, Q., and Cirak, F. An immersed discontinuous galerkin method for compressible navier-stokes equations on unstructured meshes. *International Journal for Numerical Methods in Fluids*, 91 (10):487–508, 2019.

[50] Persson, P.-O. and Peraire, J. Sub-cell shock capturing for discontinuous galerkin methods. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 112, 2006.

[51] Chaudhuri, A., Jacobs, G. B., Don, W.-S., Abbassi, H., and Mashayek, F. Explicit discontinuous spectral element method with entropy generation based artificial viscosity for shocked viscous flows. *Journal of Computational Physics*, 332:99–117, 2017.

[52] Krivodonova, L. Limiters for high-order discontinuous galerkin methods. *Journal of Computational Physics*, 226(1):879–896, 2007.

[53] Zahr, M. J., Shi, A., and Persson, P.-O. Implicit shock tracking using an optimization-based high-order discontinuous galerkin method. *Journal of Computational Physics*, 410:109385, 2020.

[54] Saye, R. High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles. *SIAM Journal on Scientific Computing*, 37(2):A993–A1019, 2015.

[55] Colella, P. and Glaz, H. M. Efficient solution algorithms for the riemann problem for real gases. *Journal of Computational Physics*, 59(2):264–289, 1985.

[56] Toro, E. F. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[57] Van Leer, B. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979.

[58] Berger, M., Aftosmis, M., and Muman, S. Analysis of slope limiters on irregular grids. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, page 490, 2005.

[59] Barth, T. and Jespersen, D. The design and application of upwind schemes on unstructured meshes. In *27th Aerospace sciences meeting*, page 366, 1989.

[60] Berger, M. J. and Colella, P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[61] Bell, J., Berger, M., Saltzman, J., and Welcome, M. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM Journal on Scientific Computing*, 15(1):127–138, 1994.

[62] Zhang, W., Almgren, A., Beckner, V., Bell, J., Blaschke, J., Chan, C., Day, M., Friesen, B., Gott, K., Graves, D., et al. Amrex: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37):1370–1370, 2019.

[63] Fortunato, D., Rycroft, C. H., and Saye, R. Efficient operator-coarsening multigrid schemes for local discontinuous galerkin methods. *SIAM Journal on Scientific Computing*, 41(6):A3913–A3937, 2019.

[64] Krivodonova, L. and Berger, M. High-order accurate implementation of solid wall boundary conditions in curved geometries. *Journal of computational physics*, 211(2):492–512, 2006.

[65] Bryson, A. and Gross, R. Diffraction of strong shocks by cones, cylinders, and spheres. *Journal of Fluid Mechanics*, 10(1):1–16, 1961.

[66] Ram, O., Geva, M., and Sadot, O. High spatial and temporal resolution study of shock wave reflection over a coupled convex–concave cylindrical surface. *Journal of Fluid Mechanics*, 768:219–239, 2015.

[67] Soni, V., Hadjadj, A., Chaudhuri, A., and Ben-Dor, G. Shock-wave reflections over double-concave cylindrical reflectors. *Journal of Fluid Mechanics*, 813:70, 2017.

[68] Koronio, E., Ben-Dor, G., Sadot, O., and Geva, M. Similarity in mach stem evolution and termination in unsteady shock-wave reflection. *Journal of Fluid Mechanics*, 902, 2020.

[69] Krivodonova, L., Xin, J., Remacle, J.-F., Chevaugeon, N., and Flaherty, J. E. Shock detection and limiting with discontinuous galerkin methods for hyperbolic conservation laws. *Applied Numerical Mathematics*, 48 (3-4):323–338, 2004.

[70] Klöckner, A., Warburton, T., and Hesthaven, J. S. Viscous shock capturing in a time-explicit discontinuous galerkin method. *Mathematical Modelling of Natural Phenomena*, 6(3):57–83, 2011.

[71] Lv, Y., See, Y. C., and Ihme, M. An entropy-residual shock detector for solving conservation laws using high-order discontinuous galerkin methods. *Journal of Computational Physics*, 322:448–472, 2016.

[72] Fernandez, P., Nguyen, C., and Peraire, J. A physics-based shock capturing method for unsteady laminar and turbulent flows. In *2018 AIAA Aerospace Sciences Meeting*, page 0062, 2018.

[73] Clain, S., Diot, S., and Loubère, R. A high-order finite volume method for systems of conservation laws—multi-dimensional optimal order detection (mood). *Journal of computational Physics*, 230(10):4028–4050, 2011.

[74] Dumbser, M., Zanotti, O., Loubère, R., and Diot, S. A posteriori subcell limiting of the discontinuous galerkin finite element method for hyperbolic conservation laws. *Journal of Computational Physics*, 278:47–75, 2014.

[75] Zanotti, O., Fambri, F., Dumbser, M., and Hidalgo, A. Space–time adaptive ader discontinuous galerkin finite element schemes with a posteriori sub-cell finite volume limiting. *Computers & Fluids*, 118:204–224, 2015.

[76] Minion, M. L. Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. *Applied numerical mathematics*, 48(3-4):369–387, 2004.

[77] Almgren, A. S., Aspden, A., Bell, J. B., and Minion, M. L. On the use of higher-order projection methods for incompressible turbulent flow. *SIAM Journal on Scientific Computing*, 35(1):B25–B42, 2013.

[78] Minion, M. L. and Saye, R. Higher-order temporal integration for the incompressible navier–stokes equations in bounded domains. *Journal of Computational Physics*, 375:797–822, 2018.

[79] Titarev, V. A. and Toro, E. F. Ader: Arbitrary high order godunov approach. *Journal of Scientific Computing*, 17(1):609–618, 2002.

[80] Balsara, D. S., Meyer, C., Dumbser, M., Du, H., and Xu, Z. Efficient implementation of ader schemes for euler and magnetohydrodynamical flows on structured meshes–speed comparisons with runge–kutta methods. *Journal of Computational Physics*, 235:934–969, 2013.