High order methods for hyperbolic balance laws : from embedded fronts to structure-preserving schemes

Mirco Ciallella

THÈSE PRÉSENTÉE
POUR OBTENIR LE GRADE DE

DOCTEUR DE

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE MATHÉMATIQUES ET INFORMATIQUE
Mathématiques Appliquées et Calcul Scientifique

Par
Mirco Ciallella

HIGH ORDER METHODS FOR HYPERBOLIC BALANCE LAWS: FROM EMBEDDED FRONTS TO STRUCTURE-PRESERVING SCHEMES

Sous la direction de
Mario Ricchiuto et **Renato Paciorri**

Soutenue le 22 Septembre 2022

Membres du jury :

M.	BONFIGLIOLI, Aldo	Associate Professor	Università della Basilicata	Examinateur
Mme.	CHERTOCK, Alina	Professor	North Carolina State University	Examinatrice
M.	CLAIN, Stéphane	Professor	Universidade do Minho	Rapporteur
M.	IOLLO, Angelo	Professeur	Université de Bordeaux	Président du jury
M.	NASUTI, Francesco	Professor	Università di Roma La Sapienza	Rapporteur
M.	PACIORRI, Renato	Associate Professor	Università di Roma La Sapienza	Co-directeur de thèse
M.	RICCHIUTO, Mario	Directeur de Recherche	INRIA	Directeur de thèse
Mme.	WEYNANS, Lisl	Maitre de Conférence	Université de Bordeaux	Examinatrice

Titre : Méthodes d'ordre élevé pour les systèmes hyperboliques : des frontières immergées aux schémas préservant la structure

Résumé :
Les équations aux derivées partielles peuvent être utilisées pour décrire de nombreux phénomènes physiques qui se produisent en ingénierie et en physique. Parmi eux, les équations hyperboliques sont très importantes lors de la modélisation de processus physiques comme en mécanique des fluides: en particulier, nous nous concentrons ici sur les équations d'Euler pour la dynamique des gaz et les équations de Saint-Venant pour les écoulements à surface libre. En raison de la structure mathématique de ces systèmes, aucune solution analytique n'est généralement pas disponible. Pour cette raison, nous devons recourir à des méthodes numériques pour les approximer. Le but de ces méthodes est d'approcher avec précision les solutions de ces problèmes avec le coût de calcul le plus bas. Afin d'exploiter au mieux la puissance et les architectures des ordinateurs modernes, des méthodes d'ordre élevé ont été introduites afin de fournir un avantage par rapport aux méthodes traditionnelles. Leur principal avantage est la possibilité d'obtenir des erreurs de discrétisation plus faibles sur des grilles plus grossières. Cependant, elles s'accompagnent de certains défis liés aux conditions aux bords et à la limitation des chocs, que nous aborderons dans le manuscrit. De plus, il est difficile de prouver, pour ces méthodes, leur capacité à préserver certaines propriétés physiques. Pour les problèmes de dynamique des fluides, on est principalement intéressé par le maintien, au niveau discret, de la positivité de variables spécifiques (par exemple, la hauteur d'eau et la densité d'un gaz) et à la conservation des états stationnaires (par exemple, le lac au repos). Bien qu'il soit trivial d'y penser, la préservation de ces propriétés physiques au niveau discret n'est pas automatique et les schémas doivent être correctement conçus pour cela. Plusieurs techniques de discrétisation ont été développées et testées sur des problèmes difficiles.

Mots clés :

Équations hyperboliques, méthodes d'ordre élevé, frontières immergées, shock tracking, positivité préservée, well balancing

Title : High order methods for hyperbolic balance laws: from embedded fronts to structure-preserving schemes

Abstract :

Partial Differential Equations can be used to describe many physical phenomena that arise in engineering and physics. Among them, hyperbolic balance laws are very important when modeling physical processes like fluid mechanics: in particular, herein, we focus on the Euler equations for gasdynamics and shallow water equations for free surface flows. Due to the mathematical structure of these systems, in general, no analytical solution is available. For this reason, we have to rely on numerical methods to approximate them. The goal of such methods is to accurately approximate the solutions of these problems with the lowest computational cost. In order to exploit at best modern computer power and architectures, high-order methods have been introduced to provide an advantageous alternative to low order schemes. Their main advantage is the capability of obtaining lower discretization errors on coarser grids. However, they come with some challenges related to boundary conditions and shock limiting, which we will address in the manuscript. In addition to this, it is difficult to prove, for such methods, their capability of preserving some physical properties. For fluid dynamics problems, one is mainly interested in maintaining, at the discrete level, positive levels of specific variables (e.g. water height and density) and in conserving stationary states (e.g. lake at rest). Although trivial to think about it, the preservation of these physical properties at the discrete level is not automatic and schemes must be properly designed for that. Several discretization techniques have been developed and tested on challenging problems.

Keywords :

Hyperbolic equations, high order methods, embedded boundaries, shock tracking, positivity preserving, well balancing

---

INRIA Bordeaux Sud-Ouest, Institute de Mathématiques de Bordeaux (IMB)

Acknowledgements

I would like to begin these acknowledgements by thanking Mario, my guide and mentor who accompanied me on this long and difficult journey. No matter how far I wanted to go, in the end you have always tried to bring out the best in me with your patience (I know it's not been easy), your advices on work *and* life, and your endless amount of ideas. The motivation I have in pursuing this career also comes from the visible passion and love that you put in your everyday life. A final thank you is for reminding me, during the hard times, that there is much more in life than just work... Sometimes we forget about that.

Another special thank goes to Renato Paciorri and Aldo Bonfiglioli for the support you have given me starting from the Master thesis. My passion for CFD and numerical methods grew a lot also thanks to you. Finally, if it hadn't been for you, I would have never ended up in Bordeaux, where I now feel at home for the first time since I left Rome to live abroad.

This PhD defense would not be possible without a jury and reviewers that accepted to read this very long manuscript and attend my presentation. For this reason, I thank Alina Chertock, Stéphane Clain, Angelo Iollo, Francesco Nasuti, and Lisl Weynans.

Although the pandemic hit our lives pretty badly, I managed to spend an amazing experience thanks to all the people I met during these years. Among all, I could never be more grateful to have met, in alphabetical order, Carolina, Davide, Elie, Giulia, Ludovica, Pietro, Sixtine, and Tiffanie. Each one of you has left something in my heart.

Finally, I want to thank my family for their endless support and love. Life is like driving boat in the dark, you have been and will always be my lighthouses along the way.

Résumé étendu en Français

Contexte général

Les équations aux derivées partielles (EDP) peuvent être utilisées pour décrire un nombre considérable de phénomènes physiques qui se produisent dans différents domaines tels que l'ingénierie, la physique, la chimie et la biologie. Parmi elles, les lois d'équilibre hyperboliques sont d'une importance capitale dans la modélisation des processus physiques, notamment en mécanique des fluides, l'électromagnétisme et les problèmes de circulation. Cependant, en raison de la structure mathématique de ces systèmes (souvent non linéaires), aucune solution analytique n'est généralement disponible pour des applications réalistes. Pour cette raison, nous devons nous appuyer sur des méthodes numériques pour discrétiser et approximer ces équations. Mais, ce n'est pas une tâche triviale, car il est bien connu que les systèmes de lois d'équilibre peuvent développer des discontinuités, même pour des données initiales lisses. La capacité de prédire correctement de tels phénomènes a ouvert la voie à un grand nombre de résultats de recherche et de communautés traitant de ce sujet. Dans le passé, de nombreuses techniques ont été utilisées et développées pour mieux comprendre ces processus physiques : différence finie [278, 196, 167, 168, 250], volume finis [157, 263, 198, 126, 289, 124], éléments finis [138, 59, 174, 173, 146, 6, 111], Galerkin discontinu [256, 180, 98, 94, 97], etc.

Au fil des ans, ces techniques ont été utilisées pour résoudre toutes sortes de problèmes. Dans ce manuscrit, nous allons nous concentrer uniquement sur des techniques numériques pour résoudre des problèmes de dynamique des fluides. En particulier, la validation numérique des nouvelles techniques présentées sera réalisée sur les équations d'Euler compressibles pour la dynamique des gaz et les équations de Saint-Venant pour les écoulements à surface libre sous l'action de la gravité.

Motivation

L'objectif des méthodes numériques mentionnées ci-dessous est d'approcher avec précision les solutions de ces problèmes avec le coût de calcul le plus bas possible. Traditionnellement, les méthodes numériques d'ordre inférieur ont été utilisées pour résoudre les équations hyperboliques, notamment pour les applications industrielles. Bien que ces méthodes puissent avoir d'excellentes propriétés de stabilité, pour les simulations à long terme, elles souffrent d'un manque de précision et d'une dissipation numérique excessive.

Afin d'exploiter au mieux la puissance et les architectures informatiques modernes, les méthodes d'ordre élevé [276, 179, 4, 31] ont été introduites pour fournir une alternative avantageuse. En effet, les méthodes d'ordre élevé permettent d'obtenir des résultats plus précis dans la même dimension du système semi-discrétisé. Néanmoins, ces méthodes ne possèdent généralement pas les propriétés de stabilité souhaitées, surtout si des discontinuités apparaissent dans la solution. Il est donc nécessaire d'avoir un bon compromis entre la précision et la stabilité. Ces problèmes de stabilité posent plusieurs défis qui doivent être relevés afin de fournir des résultats précis dans toutes sortes de situations. En particulier, les défis étudiés dans cette thèse sont liés à l'imposition cohérente des conditions aux limites, le traitement des discontinuités, la positivité des variables physiques, et la préservation de certaines propriétés du modèle au niveau discret.

En ce qui concerne les deux derniers sujets, les méthodes qui sont capables de préserver au niveau discret certaines propriétés permettent d'obtenir des résultats très précis en utilisant des grilles plus grossières, donc avec des coûts de calcul plus faibles. Récemment, le développement de méthodes préservant la structure des lois d'équilibre hyperboliques a été un domaine de recherche actif [247, 254, 34, 258, 300, 193, 79, 16]. Bien qu'il soit trivial d'y penser, la préservation des propriétés physiques au niveau numérique n'est pas automatique et les schémas numériques doivent être correctement conçus pour en tenir compte.

Ce travail se concentrera sur trois défis principaux, pertinents dans plusieurs applications:

- *l'interaction entre un maillage linéaire et une frontière courbe* (imposition de conditions aux limites cohérentes) ;
- *méthodes de suivi des chocs* (traitement précis des discontinuités) ;
- *schémas préservant la structure* (préservation de la positivité des variables physiques et techniques de discrétisation pour la préservation des solutions stationnaires).

Contributions scientifiques

Nous décrivons ici brièvement les principales contributions présentées dans chaque chapitre de ce manuscrit.

Dans le chapitre 3, nous proposons une voie alternative pour concevoir des conditions aux limites cohérentes afin de résoudre des problèmes d'écoulement compressible sur des domaines courbes discrétisés avec des mailles polygonales [87]. En particulier, nous présentons une correction du flux aux limites basée sur une reformulation simplifiée

de la méthodes des frontières immergées *Shifted Boundary Method* (SBM) qui améliore la cohérence de tous les types de conditions aux limites pour les équations d'Euler, tant pour les configurations bidimensionnelles que tridimensionnelles. Le SBM, qui nécessite généralement des calculs fastidieux de dérivées partielles d'ordre élevé, a été remplacée par une *correction polynomiale* triviale, qui consiste en une évaluation hors élément du polynôme de la cellule, qui prend en compte tous les termes de dérivées hautes d'ordre arbitraire. En raison de sa généralité, l'extension aux grilles tridimensionnelles a été simple et, avec quelques ajustements, la même approche peut également être mise en œuvre pour d'autres EDP.

Dans le chapitre 4, nous décrivons l'algorithme de shock-fitting pour les maillages non structurés UnDiFi [68] (*Unstructured Discontinuity Fitting*), maintenant disponible sur le dépôt open-source à `https://github.com/UnDiFi/UnDiFi-2D`. En traitant les discontinuités de solution comme des frontières internes réelles du domaine de calcul, UnDiFi est capable de fournir des solutions précises sur des grilles grossières, tout en conservant l'ordre de précision de la discrétisation spatiale.

Dans le chapitre 5, nous présentons l'*Extrapolated Discontinuity Tracking* [89, 90] (eDIT): une nouvelle méthode de *shock-tracking* qui emprunte des idées aux méthodes de domaine approximatif, et en particulier à la SBM. Comme dans cette dernière, nous imposons des conditions modifiées sur des manifolds de chocs de substitution, agissant comme des frontières entre les régions de *upstream* et de *downstream*. Ces frontières de substitution sont composées de deux ensembles de faces de maillage entourant la cavité des éléments traversés par le choc. Les valeurs des variables d'écoulement imposées à ces frontières de substitution sont extrapolées à partir du front de choc suivi, en tenant compte des éléments suivants les conditions non linéaires de saut et de propagation des ondes, comme cela est fait dans l'approche UnDiFi. Comme dans SBM, l'extrapolation est basée sur une expansion en série de Taylor tronquée depuis les limites de substitution jusqu'au front, permettant de préserver la précision globale de la discrétisation, même lorsque des solutions de discontinuité sont considérées. Cette nouvelle méthode constitue un pont entre l'adaptation aux chocs et les méthodes de frontières immergées, qui est virtuellement indépendante de la structure des données du maillage et du solveur gazodynamique [25].

Dans le chapitre 6, nous présentons une nouvelle méthode arbitraire d'ordre supérieur préservant la positivité pour les équations de Saint-Venant à partir d'un schéma WENO

classique [86]. Pour assurer la positivité de la hauteur d'eau, la méthode *Modified Patankar Deffered Correction* [234] (mPDeC) est utilisée pour l'intégration temporelle de cette variable. En utilisant la méthode des lignes, nous pouvons facilement diviser le processus de discrétisation et traiter les dérivées spatiales en premier. La seule contrainte de ces schémas de Patankar est que nous devons refondre notre système d'ODE, résultant de la méthode des lignes, comme un PDS. Lorsque nous effectuons le calcul des dérivées spatiales, nous devons faire attention à éviter l'échec de positivité car les méthodes de Patankar n'influencent que l'évolution temporelle. Dans notre cas, pour disposer d'un outil d'ordre élevé arbitraire, nous discrétisons les dérivées spatiales avec un schéma volumes finis basé sur la reconstruction WENO, couplé avec un limiteur positif classique [306]. Grâce aux propriétés provenant de la méthode Patankar (linéairement implicite), la reconstruction positive n'est plus affectée par les sévères restrictions CFL données par les poids de Lobatto et les simulations peuvent être exécutées avec des pas de temps plus importants.

Dans le chapitre 7, nous visons à maximiser la précision des simulations en combinant la technique de *Flux Globalization* et la précision d'ordre élevé des méthodes WENO à volumes finis [91]. L'aspect le plus délicat de l'algorithme est la définition appropriée de l'intégrale du terme source (flux de sources) et la stratégie de quadrature utilisée pour la faire correspondre avec la reconstruction WENO du flux hyperbolique. Lorsque cette construction est correctement réalisée, on peut montrer que le schéma volumes finis WENO résultant admet des états stationnaires discrets exacts caractérisés par des flux globaux constants et qui est capable de préserver une grande famille d'équilibres mobiles lisses et discontinus en régime permanent. Ces techniques ne garantissent pas seulement la préservation de ces équilibres, mais améliorent également la précision générale de la solution. De plus, pour préserver exactement l'état stationnaire du lac au repos, une modification supplémentaire doit être apportée à la définition de l'intégrale du terme source discrétisée, afin de garantir la propriété $\mathcal{C}$. Il est également montré qu'une fois que la quadrature du flux du terme source est bien implémentée, d'autres termes source peuvent être facilement ajoutés au système sans problème pour étudier des types d'équilibres plus compliqués découlant de nouvelles équations.

Mirco Ciallella

Discover your vision and the rest will follow

Arnold Schwarzenegger

Contents

List of Figures

List of Tables

Introduction

General context

Partial Differential Equations (PDEs) can be used to describe a tremendous amount of physical phenomena that arise in different fields such as engineering, physics, chemistry and biology. Among them, hyperbolic balance laws are of paramount importance when modeling physical processes including fluid mechanics, electromagnetism and traffic problems. However, due to the mathematical structure of these systems (often nonlinear), in general, no analytical solution is available for realistic applications. For this reason, we have to rely on numerical methods to discretize and approximate these equations. However, this is not a trivial task since, it is well-known that, systems of balance laws can develop discontinuities even for smooth initial data, and the capability of correctly predicting such phenomena paved the way to a huge amount of research outputs and communities addressing this topic. In the past, many techniques have been used and developed to have a better understanding of such physical processes: finite difference [278, 196, 167, 168, 250], finite volume [157, 263, 198, 126, 289, 124], finite element [138, 59, 174, 173, 146, 6, 111], discontinuous Galerkin [256, 180, 98, 94, 97] and so on.

Over the years, these techniques have been employed to solve all kind of problems. In this manuscript, we are going to focus only on numerical techniques to solve fluid dynamics problems. In particular, the numerical validation of the new presented techniques will be performed on the compressible Euler equations for gasdynamics and the shallow water equations for free surface flows under the action of gravity.

Motivation

The goal of the aforementioned numerical methods is to accurately approximate the solutions of these problems with the lowest possible computational cost. Traditionally, low order numerical methods have been used to solve hyperbolic equations, especially for industrial applications. Although these methods can have excellent stability properties, for long-time simulations, they suffer from a lack of accuracy and their excessive numerical dissipation. In order to exploit at best modern computer power and architectures,

high-order methods [276, 179, 4, 31] have been introduced to provide an advantageous alternative. Indeed, high order methods allow to obtain more accurate results within the same dimension of the semi-discretized system. Nevertheless, these methods often lack desired stability properties, especially if discontinuities arise in the solution. It is therefore necessary to have the good trade-off between accuracy and stability. These stability issues introduce several challenges that need to be tackled in order to provide accurate results in all kind of situations. In particular, the challenges studied in this thesis are related to the consistent imposition of boundary conditions, the treatment of discontinuities, the positivity of physical variables, and the preservation of some properties of the model at the discrete level.

Regarding the last two topics, methods which are capable of preserving at the discrete level some properties allow to obtain very accurate results using coarser grids, hence with lower computational costs. Lately, the development of structure-preserving methods for hyperbolic balance laws have been an active field of research [247, 254, 34, 258, 300, 193, 79, 16]. Although trivial to think about it, the preservation of physical properties at the numerical level is not automatic and numerical schemes must be properly designed to account for that.

This work will focus on three main challenges, relevant in several applications:

- *the interaction between a linear mesh and a curved boundary* (imposition of consistent boundary conditions);
- *shock-tracking methods* (accurate treatment of discontinuities);
- *structure-preserving schemes* (positivity preservation of physical variables and discretization techniques for the preservation of stationary solutions)

Accurate boundary conditions

In numerous gasdynamic applications, the interaction between a mesh and a boundary is often of interest because of all the troubles introduced by the imposition of consistent boundary conditions. It is a fact that, when working with high order methods, the boundary conditions should be consistent with the discretization of the problem [31], meaning the numerical scheme used to approximate the solution. In general, to simulate problems involving internal boundaries, two paths can be followed: body-fitted grids and immersed/embedded boundary methods [248].

The *body-fitted meshes* allow placing nodes optimally to resolve geometrical features of the problem and easily increase mesh density towards the wall by making anisotropic

near-surface mesh with cell sizes different by orders of magnitude to resolve sharp gradients in the direction normal to the wall. When grid nodes are carefully placed over the considered geometry, the imposition of boundary conditions becomes straightforward. However, working with complicated (curved) geometries might introduce some troubles in the generation of adapted unstructured meshes. Moreover, internal boundaries could be moving within the computational domain making the adaptation process even harder. In the context of keeping a body-fitted mesh while allowing a moving boundary, we mention that several non-trivial algorithms have been developed to perform these tasks: among all of them we refer to body-fitted anisotropic mesh adaptation [29, 19] and overset grid methods [77, 37].

Immersed and *embedded* boundary methods have been developed to allow a flexible management of complex geometries. Although similar, these two approaches rely on a slightly different philosophy. Immersed methods are based on an extension of the flow equations outside the physical domain (typically within solid bodies). This extension is formulated using some smooth approximation of the Dirac delta function to localize the boundary, as well as to impose the boundary conditions. These methods are relatively old, and based on the original idea of Peskin [248]. Finite element and unstructured mesh extensions for elliptic PDEs as well as for incompressible, and compressible flows have been discussed in [40, 12, 231, 108, 266, 183]. Embedded methods, on the other hand, solve the PDEs only in the physical domain, while replacing the exact boundary with some more or less accurate approximation, combined with some weak enforcement of the boundary conditions. There is a certain number of techniques to perform this task, which go from the combination of XFEM-type methods with penalization or Nitsche's type approaches [165], to several types of cut finite element methods with improved stability [63, 64, 213, 35, 147], to approximate domain methods such as the well known ghost-fluid method [129, 128, 284, 130, 202], and the more recent shifted boundary method [206, 207, 279, 232, 200, 26, 72].

In general, when either immersed or embedded boundary methods are considered, the grid points do not lay over the internal boundary which makes trickier the imposition of boundary conditions. In the aforementioned references, we cited several works that exploit different philosophies to properly impose these conditions. However, many efforts have been put to solve this issue, it is not trivial how to treat internal boundaries while retaining high order (at least second order) of accuracy [212, 163, 26]. This reasoning also concerns curved boundaries discretized using body-fitted linear meshes because, even if the mesh is conformal, a curved boundary is approximated with a series of segments that do not exactly match the real boundary everywhere. In this case, obtaining convergence trends higher than two becomes challenging.

Depending on the considered application, the general aforementioned "internal boundaries" may be of different nature and describe different objects. Generally speaking, there are three different kinds of boundary conditions.

- *Dirichlet*: it specifies the value that the solution needs to take along the boundary ($u(x) = u_D$).
- *Neumann*: it specifies the value that the derivative of the solution needs to take along the boundary ($\partial_n u(x) = u_N$).
- *Robin*: it is a weighted combination of Dirichlet and Neumann boundary conditions ($au(x) + b\partial_n u(x) = u_R$).

When dealing with fluid dynamics simulations, at a given boundary, different types of boundary conditions can be used for different variables.

- *Inlet/outlet*: variables are prescribed with a predefined profile on the boundary such that the inflow and outflow conditions are consistent with the physics of the problem.
- *Far-field*: it is a particular case of inlet/outlet, which consists in imposing the whole solution vector with a prescribed set of variables because the boundary is considered "far away" from the studied problem.
- *Periodic*: it consists of pairs of geometrically identical boundaries at which all flow conditions are matched.
- *Slip-wall*: it imposes to zero the normal component of the velocity and keeps the tangential components untouched at the assigned boundary.

In Part II, we address several aspects related to the interaction between a mesh and a boundary when the position of the latter does not coincide with the previously generated mesh. To do so, we are going to exploit the philosophy of the Shifted Boundary Method (SBM). This method falls in the category of surrogate/approximate boundary algorithms and its key feature relies on *shifting* the location where boundary conditions are applied from the true to the surrogate boundary, and to appropriately modify the boundary conditions, in order to preserve optimal convergence rates of the numerical solution. This modification, performed using extrapolations based on Taylor series expansions truncated to achieve the desired accuracy, is extremely important to avoid a reduction in the convergence rates of the overall discretization. This method is simple, efficient and it is not

affected by the small-cell problem, which would introduce severe time-step restrictions and stability problems if no precautions are taken [213, 35, 36, 147].

It is a fact that, when dealing with high order methods, such as discontinuous Galerkin (DG), accuracy may be dramatically affected when the boundary domain is curved. This plays a crucial role for high order (higher than two) methods where errors due to geometrical approximation may even overcome those related to the actual discretization of the scheme. For this reason, a special treatment of boundary conditions should be implemented to preserve the designed order of accuracy: classical boundary conditions are consistent with the discretization technique only when the boundary is straight. When dealing with finite element methods, the most classical approach is to work with an iso-parametric approximation in which the geometry, as well as the flow solution, are approximated by some high order polynomial [308]. Standard approaches range from the use of various maps based on some local interpolated or modal polynomial approximation of the curved geometry, to the more recent use of rational B-spline or NURBS approximations used in the so called iso-geometric analysis (IGA) [175]. However, increased accuracy comes with a price to pay. A significant bottleneck in high order simulations is the generation of unstructured curvilinear meshes [188, 295, 53, 137, 302]: this is usually achieved by increasing the degree of freedom per grid cell. Moreover, curvilinear meshes comes with several troubles mainly related to

- difficulties in mesh generation,
- non-linear mapping of the element in the physical space with respect to that in the reference space,
- non-constant Jacobian,
- implementation of *ad-hoc* quadrature formulas to compute surface and volume integrals over curved elements.

In alternative to high order meshing, one can improve the boundary conditions by accounting, on a straight faced mesh, for the local features of the true geometry. Early work on curvature corrected wall boundary conditions can be found in [299], while the specific case of high order schemes and wall boundaries in two space dimensions has been thoroughly treated in the well known paper by Krivodonova and Berger [187]. In the last reference, the authors discretize the physical domain using polygonal meshes and propose an approach to correct the direction used when prescribing the slip-wall condition, which shows a recovery of third and, for some cases, fourth order of accuracy on 2D geometries.

However, this method can be formulated only for slip-wall conditions, and the work is limited to 2D geometries. More recently, a new method called Reconstruction for Off-site Data (ROD) has been proposed [102] for the imposition of Dirichlet boundary conditions, in a high-order finite volume framework. Afterwords, the work was also extended to other boundary conditions [105, 104, 103] and non-conformal meshes [131] obtaining very promising results. Due to the finite volume framework, the approach is based on a constrained least-squares, which is used to handle several constraints from scattered mean values associated to the elements. Due to the linear system arising from the constrained least-squares problem, a matrix should be inverted locally and this may introduce some issues for ill-conditioned problems.

Treatment of discontinuities

Generally speaking, the numerical techniques used to simulate flows with shock waves are essentially two: the widely used shock-capturing (SC) methods, and the less common front-tracking/shock-fitting methods. Shock-capturing methods relies on the proven mathematical legitimacy of weak solutions: all types of flows, including flows with shocks, can be computed by using the same discretization of the equations [296]. Nevertheless, the shocks always appear smeared in a region whose thickness is of two or three cells rather than actual discontinuities. Since the states of the cells inside this region are unphysical [305], the shock-capturing methods suffer from some numerical problems concerning stability, accuracy and quality of the solutions that sometimes give anomalous results. A catalogue of these *failings* was made by Quirk in the early 90s of the last century [253]. Front-tracking/shock-fitting methods consist in explicitly identifying the shock as a line in 2D (surface in 3D) within the flow-field and computing its motion, and its upstream and downstream states according to the Rankine-Hugoniot equations. These techniques have been a huge topic of research, starting from the 60s, gaining attention from many research groups. Among them, we are going to refer mainly to those works coming from Gino Moretti and collaborators, which became popular under the name of shock-fitting (SF), and those coming from James Glimm and collaborators, which are known under the name of front-tracking (FT).

Two different SF methodologies blossomed between the 60s and 80s: the boundary shock-fitting [223] and floating shock-fitting [268]. In the former approach, the shock is made to coincide with one of the boundaries of the computational domain so that the treatment of the jump relations across the shock is confined to the boundary points. Even though this method greatly simplified the coding, the treatment of shocks appearing

within the computational domain, and of shock interactions, became a major challenge. The floating shock-fitting approach was developed to be capable of dealing with more complex flow configurations. In the floating version, discontinuities can freely move over a background structured mesh: a shock front is described by its intersections with the grid-lines, which give rise to x and y shock points, meaning that they are allowed to move onto grid-lines. Even though floating shock-fitting codes have been used with success in the past to compute steady and un-steady two- and three-dimensional flows involving shock reflections and shock interactions [228, 229, 252], they are very complex to code and require extensive changes in the computational kernel of the gasdynamic solver. Exploiting the flexibility given by unstructured meshes, Paciorri and Bonfiglioli developed a new unstructured shock-fitting technique for unstructured vertex-centered solvers, firstly presented in [235], that alleviated many of the difficulties of the shock-fitting techniques in the structured-grid framework. In recent years, the unstructured shock-fitting technique was improved to deal with interactions among discontinuities in two-dimensional flows [237], three-dimensional flows [48] and un-steady compressible flows [45, 69] opening a new route in simulating flow-fields with shock waves. In particular, not only shocks and contact discontinuities are fitted, but also the interaction points, for example the triple points arising in Mach reflections [177]. A limitation of this technique is that it heavily relies on the flexibility of triangular and tetrahedral grids to locally produce a fitted unstructured grid around the discontinuities. For this reason, this limits its application to unstructured vertex-centered codes.

Although similar to SF approaches, front-tracking methods (FT) [271, 150, 148, 81, 118], as conceived by Glimm and collaborators, differ slightly from those and, for this reason, we recall here the main contributions, whose development started in the 80s. In [184], the authors summarize the FT approach by stating that these methods solve an initial-boundary value problem on both sides of the front and never use states on the opposite side of the front:

- away from the front, this is readily achieved by using any finite difference scheme compatible with the resolution one needs in the interior;
- near the front, whenever a stencil gets cut off by the front, the algorithm use the *ghost* states at the nearest crossing point (obtained through linear interpolation from the front states) and place them at the missing stencil points.

This method happened to be quite effective and it was extended and applied to many applications [152, 150, 151, 149]. However, by following this approach, the method results being affected by the small-cell problem and CFL stability limits. Afterwords, Glimm and

collaborators proposed a *conservative* FT method [155, 153, 154] with the goal of fixing the stability problems related to its unconservative counterpart. The issue was addressed by merging small cells with adjacent cut cells on the same side of the front and applying the interior scheme in the new constructed cells. Similar approaches had also been followed by other researchers [82, 166, 244, 281] with the aim of having a similar tool. With the goal of avoiding merging algorithms and small-cell problems, Mao De-kang introduced a conservative front-tracking method [208, 209, 210, 211] to track discontinuities in a capturing fashion. This approach is based on extrapolations carried out only on the smooth side of the flow, and a modification of the interior scheme to ensure conservation and the right order-of-convergence. Despite the good performances, this FT approach needs extensive modifications in the interior scheme used and is Cartesian-grid-based making it complicated to couple with other mesh data structures.

Structure-preserving

Structure-preserving schemes belongs to a family of methods that consists in improving the numerical solution of hyperbolic balance laws by preserving, at the discrete level, some physical properties and solutions that exist at the continuous level. Although applicable to all kinds of balance laws, here we refer to the shallow water equations to solve free surface problems and the issues associated to such applications.
When solving hyperbolic balance laws, several physical properties can be considered to develop accurate schemes.

- *Positivity of primitive variables* [247, 216, 306, 171, 170, 215, 86]. One important issue of high-order conservative schemes is that non-physical negative water height can lead to an ill-posed system, which may cause the blow-up of numerical solutions; in particular for higher-order conservative schemes, positivity failure can easily occur near wet-dry areas.

- *Preservation of equilibrium and stationary states* [201, 115, 301, 76, 39, 218, 38, 230]. It is known that standard methods can fail in solving systems of balance laws when approaching equilibria (or near equilibria); instead, a scheme is said to satisfy the $\mathcal{C}$-property if it solves correctly the steady state solutions corresponding to the lake-at-rest. Many papers [27, 57, 75, 191, 204] treat the lake-at-rest equilibrium for the shallow water equations; however, it is significantly more difficult to obtain well-balanced schemes for more general steady states.

- *Entropy conservation (or stability)* [249, 251, 257, 60, 282, 283, 134]. Weak solutions are not unique and, in order to select the "physically relevant" solution among all weak solutions, entropy functions are used as the admissibility criterion. In smooth regions, an extra conservation law for the entropy function is obtained; however, at shock waves, we require the entropy to dissipate, which leads to an inequality stating that the total entropy is non-increasing with respect to time. It is then natural to seek numerical schemes which satisfy a discrete version of these conditions.

It has been shown that being able to preserve such physical properties results in schemes characterized by better numerical properties and improved solutions [141], using coarser meshes with respect to classical methods.

As already mentioned, despite of the fact that the work discussed in the manuscript applies to general systems of balance laws, this chapter focuses on the shallow water equations. These equations are ubiquitous in coastal hydrodynamics. The dynamic processes that occur in the near-shore can be generated by several drivers: external forces, fluid motion of the water manifests itself as coastal currents, tides and tidal currents, internal and surface waves, storm surges and others. It is straightforward to think that when these phenomena happen, we may have the interaction of water waves with the so-called "dry areas", which could be inhabited centers near the sea or rivers subject to flooding.
For these reason, we deal with the issue of positivity failure, that may often occur close to dry areas. In this context, modified Patankar schemes are numerical methods for the solution of positive and conservative production-destruction systems (PDS). They already have shown to be easily adaptable to explicit Runge-Kutta schemes [185, 186, 178] in a way to ensure positivity and conservation irrespective of the time step size. Being unconditionally positive, for any time step, is an appealing feature that does not characterize classical positive limiters [306, 247] used in the context of high order WENO [275, 179] schemes for hyperbolic PDEs. Even though these limiters are very effective, they come with some stability issues, which make the CFL condition much stricter than usual. The key idea behind these methods is the *Patankar's trick* [243] which consists in multiplying the destruction terms with weights making the scheme linearly implicit. As this procedure destroys conservation, the production terms must be weighted accordingly. *Modified* Patankar schemes have been successfully employed for a large number of different application [62, 162, 169, 270] and their success is particularly based on the fact that they are able to solve stiff PDS. Even if (modified) Patankar (mP) methods have been already used inside a numerical method for fluid simulations [170, 171, 216], those mP schemes have been based on extensions of classical RK methods and they are of maximum order three. Moreover, they have been only used for the source terms, multicomponent terms or

in a post-processing process. By applying the modified Patankar trick inside the Deferred Correction (DeC) framework, the authors of [234] were able to construct a conservative, arbitrarily high-order and positivity preserving method for PDS of ordinary differential equations (ODEs). However when dealing with PDEs, the situation is of course different because, generally, PDEs are not written as a PDS. Recasting PDEs in PDS is not trivial but is much simpler when we cope with hyperbolic PDEs written in conservative formulations.

As we mentioned above, another way to obtain accurate results with low computational costs is to design methods which are capable of preserving some physical states. The shallow water equations, for example, are verified by some steady state solutions that are very easily described. The simplest equilibrium is the lake-at-rest steady state where the water level is constant and velocity is zero. The preservation of this equilibrium at the numerical level is not automatic and numerical schemes must be specifically designed in order to balance the discretization of the flux and the source in this situation [27, 41, 115, 141, 192, 21, 218, 219, 220]. These methods increase the accuracy of the solutions when close to these equilibria, but not only; it is possible to see great improvement in the discretization error also for more general tests. There are also other types of steady states that solve the shallow water equations. A generalized form of the lake-at-rest is obtained when the discharge (momentum) is not zero, but constant in space. In this situation, plenty of equilibria can occur. Again, an analytical solution is not always available, but some implicit forms of the solution [113] are known. It is then interesting to build schemes that are able to preserve also these equilibria at the discrete level, because it gives us a tool to efficiently predict some flows using very coarse grids. A new class of schemes has been presented in the recent years with the aim of preserving more general equilibria of balance laws [83, 80, 85]. They rely on the idea of defining a *global flux* that incorporates both the hyperbolic flux and the source term thanks to a discretization of the integral of the source term. This method has been already applied in different contexts [85, 83, 70, 71] and proved to be very promising for general systems of balance laws with non-trivial steady states.

Scientific contributions

The thesis has contributed to several projects published, or submitted, in international journals. Here, we briefly describe the main contributions presented in each chapter of this manuscript.

In Chapter 3, we propose an alternative path to design consistent boundary condi-

tions to solve compressible flow problems on curved domains discretized with polygonal meshes [87]. In particular, we present a boundary flux correction based on a simplified reformulation of the SBM that improves the consistency of all kinds of boundary conditions for the Euler equations, for both two-dimensional and three-dimensional configurations. The SBM, which usually would require cumbersome computations of high order partial derivatives, has been replaced by a trivial *polynomial correction*, which consists in an off-element evaluation of the cell polynomial, that takes into account all the high derivative terms of arbitrary order. Due to its generality, the extension to three-dimensional grids has been straightforward and, with few adjustments, the same approach can also be implemented for other PDEs.

In Chapter 4, we describe the shock-fitting algorithm for unstructured meshes [68] (Unstructured Discontinuity Fitting), now available at the *open-source* repository at `https://github.com/UnDiFi/UnDiFi-2D`. By treating solution discontinuities as actual internal boundaries of the computational domain, UnDiFi is able to provide accurate solutions on coarse grids, while also retaining the designed order-of-accuracy of the spatial discretization.

In Chapter 5, we introduce the extrapolated Discontinuity Tracking [89, 90] (eDIT) method: a new *shock-tracking* method that borrow ideas from approximate domain methods, and in particular from the SBM. As in the latter, we impose modified conditions on surrogate shock-manifolds, acting as boundaries between the shock-upstream and shock-downstream regions. These surrogate boundaries are composed of two sets of mesh faces enclosing the cavity of elements crossed by the shock. The values of the flow variables imposed on these surrogate boundaries are extrapolated from the tracked shock front accounting for the non-linear jump and wave propagation conditions, as done in the UnDiFi approach. As in the SBM, the extrapolation is based on a truncated Taylor series expansion from the surrogate boundaries to the front, allowing to preserve the overall accuracy of the discretization, even when discontinuity solutions are considered. This new method constitutes a bridge between shock-fitting and embedded boundary methods, which is virtually independent of the mesh data structure and gasdynamic solver [25].

In Chapter 6, we present a new arbitrary high-order positivity-preserving method for the shallow water equations starting from a classical WENO scheme [86]. To ensure the *positivity* of the water height, the modified Patankar Deferred Correction [234] (mPDeC) method is used for the time-integration of this variable. By using the method of lines,

we can easily split the discretization process and treat the spatial derivatives first. The only constraint of such Patankar schemes is that we need to recast our ODE system, resulting from the method of lines, as a PDS. When performing the computation of spatial derivatives, we have to pay attention to avoid positivity failure because Patankar methods only influence the time evolution. In our case, to have an arbitrary high order tool, we discretize the spatial derivatives with a finite volume scheme based on the WENO reconstruction, coupled with a classical positive limiter [306]. Thanks to the properties coming from the (linearly implicit) Patankar method, the positive reconstruction is no longer affected by the severe CFL restrictions given by Lobatto weights and simulations can be run with larger timesteps.

In Chapter 7, we aim at maximizing the accuracy of simulations by combining the flux globalization technique and the high order accuracy of finite volume WENO methods [91]. The most delicate aspect of the algorithm is the appropriate definition of the *source flux* (integral of the source term) and the quadrature strategy used to match it with the WENO reconstruction of the hyperbolic flux. When this construction is correctly done, one can show that the resulting WENO finite volume scheme admits exact discrete steady states characterized by constant global fluxes and is able to preserve a large family of smooth and discontinuous steady state moving equilibria. These techniques do not only guarantee the preservation of these equilibria, but also improve the general accuracy of the solution. Moreover, to exactly preserve the lake-at-rest steady state, a further modification must be done in the definition of the discretized source integral. It is also shown that, once the quadrature of the source flux is well implemented, more source terms can be easily added to the system with no issues to study more complicated types of equilibria arising from the new equations.

To summarize, this thesis has contributed to seven publications

- **Shifted boundary polynomial corrections for compressible flows: high order on curved domains using linear meshes** (submitted to *Applied Mathematics and Computation* [87]),

- **UnDiFi-2D: an unstructured discontinuity fitting code for 2D grids** (published in *Computer Physics Communications* [68]),

- **Extrapolated shock tracking: bridging shock-fitting and embedded boundary methods** (published in *Journal of Computational Physics* [89]),

- **Extrapolated discontinuity tracking for complex 2D shock interactions** (published in *Computer Methods in Applied Mechanics and Engineering* [90]),
- **Extrapolated shock fitting for two-dimensional flows on structured grids** (published in *AIAA Journal* [25]),
- **An arbitrary high order and positivity preserving method for the shallow water equations** (published in *Computer & Fluids* [86]),
- **Arbitrary high order well-balanced WENO finite volume schemes for moving equilibria preservation** (submitted to *Journal of Scientific Computing* [91]),

and five talks held in international conferences

- **Extrapolated discontinuity tracking: how to reduce the impact of the mesh topology in shock-fitting algorithms** (WCCM ECCOMAS 2020),
- **Higher-order shifted boundary method with Finite Volume and Discontinuous Galerkin schemes for hyperbolic systems: initial results** (COUPLED 2021),
- **Polynomial corrections for high order compressible flows simulations on curved domains with linear meshes** (HONOM 2022)
- **Extrapolated discontinuity tracking for 2D shock interactions: bridging shock-fitting and embedded boundary Methods** (SHARK 2022)
- **Arbitrary high order well-balanced WENO finite volume scheme for moving equilibria preservation** (CEDYA 2022)

Part I

Hyperbolic Problems and Numerical Methods

Chapter 1

Hyperbolic Balance Laws

In order to understand what comes in Parts II and III, we introduce some knowledge about hyperbolic conservation laws. In this chapter, we focus on the equations, and their properties, and repeat important results.

1.1 Basic definitions and notations

The models used in this work can be all recast in the form of *hyperbolic balance laws*. To begin with, let us consider the equation in quasi-lienar form reading

$$\partial_t \mathbf{u}(\mathbf{x},t) + \sum_{i=1}^{d} \mathcal{A}_i(\mathbf{x},t,\mathbf{u}(\mathbf{x},t))\,\partial_{x_i}\mathbf{u}(\mathbf{x},t) = \tilde{\mathcal{S}}(\mathbf{x},t,\mathbf{u}(\mathbf{x},t)), \qquad t\in[0,T], \quad \mathbf{x}\in\mathbf{\Omega}, \quad (1.1)$$

$$\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x}), \qquad \mathbf{x}\in\mathbf{\Omega}, \quad (1.2)$$

where $t\in\mathbb{R}^+$ denotes the time, $\mathbf{x}$ is the variable of the space coordinates, $\mathbf{\Omega}\in\mathbb{R}^d$ is the spatial domain (a connected open set with dimension d) and $\mathbf{u}:\mathbf{\Omega}\times[0,T]\to\mathcal{U}\subseteq\mathbb{R}^N$ is the unknown solution with $N\in\mathbb{N}$ being the number of equations of the system (1.1). Moreover, $\mathcal{A}_i : \mathbf{\Omega}\times[0,T]\times\mathcal{U}\to\mathbb{R}^N\times\mathbb{R}^N$ and $\tilde{\mathcal{S}} : \mathbf{\Omega}\times[0,T]\times\mathcal{U}\to\mathbb{R}^N$ are given functions. We then denote by ∂_t the partial derivative with respect to the time t and by

∂_{x_i} the derivative with respect to the i-th space component x_i. The function $\mathbf{u}_0 : \mathbf{\Omega} \to \mathcal{U}$ describes the initial condition (1.2) of $\mathbf{u}$ and, with (1.1), they make up the *initial value problem* (IVP). When system (1.1) is also equipped with additional constaints on $\partial\mathbf{\Omega}$, namely *boundary conditions*, the problem considered is the *initial-boundary value problem* (IBVP). Herein, we are always going to deal with the latter.
Since we are only going to deal with hyperbolic problems throughout this manuscript, we starts by defining what hyperbolicity means:

Definition 1 (Hyperbolicity)**.** *The system* (1.1) *is called* ***hyperbolic*** *if the matrix* $\mathcal{A}(\mathbf{x}, t, \mathbf{u}, \mathbf{n}) = \sum_{i=1}^{d} n_i \mathcal{A}_i(\mathbf{x}, t, \mathbf{u})$ *is diagonalisable for all* $t \in [0, T]$, $\mathbf{x} \in \mathbf{\Omega}$, $\mathbf{u} \in \mathcal{U}$, *and* $\mathbf{n} = \{n_i\}_{i=1,\dots,d}$.
We can prove hyperbolicity of system (1.1) *following two approaches.*

1. *If* $\mathcal{A}_i(\mathbf{x}, t, \mathbf{u})$ *is symmetric for* $i = 1, \dots, d$, *then* $\mathcal{A}(\mathbf{x}, t, \mathbf{u}, \mathbf{n})$ *is symmetric for all* $\mathbf{n} \in \mathbb{R}^d$. *Recall that if* $\mathcal{A}(\mathbf{x}, t, \mathbf{u}, \mathbf{n})$ *is symmetric, then it is diagonalizable. When the matrices* $\mathcal{A}_i(\mathbf{x}, t, \mathbf{u})$ *are all symmetric, we say the system* (1.1) *is* ***symmetric hyperbolic****.*

2. *If* $\mathcal{A}$ *has* N *real eigenvalues* $\lambda_1 \leq \dots \leq \lambda_N$, *with the corresponding set of eigenvectors* $\{r_i\}_{i=1,\dots,N}$. *Furthermore, the system is called* ***strictly hyperbolic*** *if the eigenvalues are also distinct.*

Rather than working with the quasi-linear form introduced in (1.1), we focus on a different form:

Definition 2 (Balance Law)**.** *The quasi-linear system* (1.1) *is called* ***balance law*** *if there exist a set of functions* $\mathcal{F}_i : \mathbf{\Omega} \times [0, T] \times \mathbf{u} \to \mathbb{R}^N$, *namely the* ***flux functions****, such that* $\mathcal{A}_i(\mathbf{x}, t, \mathbf{u}) = \partial_{\mathbf{u}} \mathcal{F}_i(\mathbf{x}, t, \mathbf{u})$, *for all* $1 \leq i \leq d$, $t \in [0, T]$, $\mathbf{x} \in \mathbf{\Omega}$ *and* $\mathbf{u} \in \mathcal{U}$.

Once the flux functions are defined, we can rewrite the balance law as

$$\partial_t \mathbf{u}(\mathbf{x}, t) + \sum_{i=1}^{d} \partial_{x_i} \mathcal{F}_i(\mathbf{x}, t, \mathbf{u}) = \mathcal{S}(\mathbf{x}, t, \mathbf{u}(\mathbf{x}, t)), \qquad t \in [0, T], \quad \mathbf{x} \in \mathbf{\Omega}, \tag{1.3}$$

where the function $\mathcal{S} : \mathbf{\Omega} \times [0, T] \times \mathcal{U} \to \mathbb{R}^N$ is referred to as *source term*. The hyperbolic problems recast using the flux functions, as described by (1.3), are called to be in *conservative form* whereas those written using the matrices $\mathcal{A}_i$ are in *non-conservative form*.
To be complete, we also define by $\nabla\cdot$ the divergence with respect to x_i in order to recast the flux functions in a more compact way:

$$\nabla \cdot \mathcal{F} = \sum_{i=1}^{d} \partial_{x_i} \mathcal{F}_i. \tag{1.4}$$

The Cauchy problem given by (1.3), with the corresponding initial conditions (1.2), can be finally reformulated by applying the divergence formulation (1.4):

$$\partial_t \mathbf{u}(\mathbf{x},t) + \nabla \cdot \mathcal{F}(\mathbf{x},t,\mathbf{u}) = \mathcal{S}(\mathbf{x},t,\mathbf{u}(\mathbf{x},t)), \qquad t \in [0,T], \quad \mathbf{x} \in \mathbf{\Omega}, \tag{1.5}$$

Now that the concept of Balance law is defined, we can introduce a subset of this problem.

Definition 3 (Conservation law)**.** *If the source term* $\mathcal{S}$ *vanishes in the balance law* (1.5)*, the system is called* ***conservation law****,*

$$\partial_t \mathbf{u}(\mathbf{x},t) + \nabla \cdot \mathcal{F}(\mathbf{x},t,\mathbf{u}) = 0, \qquad t \in [0,T], \quad \mathbf{x} \in \mathbf{\Omega}, \tag{1.6}$$

and the function $\mathbf{u}$ *is called* ***conserved variable****.*

It should be noticed that hyperbolic problems may not have classical solution for all $t > 0$, even for smooth initial data. We define a classical solution as:

Definition 4 (Classical solution)**.** $\mathbf{u} : \mathbf{\Omega} \times [0,T] \to \mathcal{U}$ *is a* ***classical solution*** *of the Cauchy problem* (1.5)*, equipped with* (1.2) *if* $\mathbf{u} \in C^1$ *fulfills* (1.5) *pointwise.*

Indeed, after some time, discontinuities may occur in the solution. When this happens, the function $\mathbf{u}$ is not of class C^1 therefore the Picard-Lindelöf theorem that guarantees the local existence of a unique solution does not apply [99].

We can easily show that an hyperbolic conservation law, equipped with smooth initial condition, may give rise to both continuous and discontinuous solutions. To do so, we introduce a mathematical procedure that allows to get the analytical solution from such problems: the *method of characteristics* (MOC). Here we only introduce this method but a detailed description of the MOC can be found in [125].

Example 5. *For the sake of simplicity, we consider a one-dimensional scalar hyperbolic conservation law*

$$\partial_t u(x,t) + \partial_x f(u(x,t)) = 0, \qquad t > 0, \quad x \in \mathbb{R}, \tag{1.7}$$

$$u(x,0) = u_0(x), \qquad x \in \mathbb{R}, \tag{1.8}$$

$$\tag{1.9}$$

with smooth flux function $f \in C^1$*, and* u *being a classical solution of* (1.7)*. By introducing the jacobian of the flux* $a(u) := \partial_u f(u) = f'(u)$*, we can rewrite* (1.7) *in the following quasi-linear form*

$$\partial_t u(x,t) + a(u(x,t))\partial_x u(x,t) = 0. \tag{1.10}$$

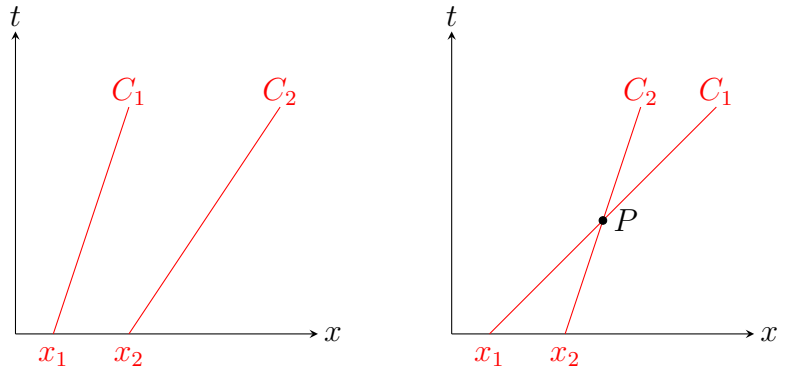


Figure 1.1: Characteristic lines with different initial conditions.

The ***characteristic lines*** *associated with* (1.10) *are defined as the solution* $x : [0, T] \to \mathbb{R}$ *of the ordinary differential equation*

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = a(u(x(t), t)), \qquad x(0) = x_0. \tag{1.11}$$

Therefore, a characteristic is a line of the (x, t) *plane such that, at each point,* $\frac{\mathrm{d}}{\mathrm{d}t}x(t)$ *equals the local value of the jacobian of the flux.*

The main property of such characteristic lines is that a classical solution u *of* (1.10) *is constant moving along them. We can easily prove that by writing down the total derivative along these curves*

$$\frac{\mathrm{d}}{\mathrm{d}t}u(x(t), t) = (\partial_t u)(x(t), t) + (\partial_x u)(x(t), t)\frac{\mathrm{d}}{\mathrm{d}t}x(t) = (\partial_t u + a(u)\partial_x u)(x(t), t) = 0. \tag{1.12}$$

It follows that the slopes $\frac{\mathrm{d}}{\mathrm{d}t}x(t) = a(u(x(t), t)) = a(u_0(x_0))$ *of the characteristics are constant, meaning that the characteristic lines are straight lines defined by the linear equation*

$$x(t) = x_0 + t\, a(u_0(x_0)), \tag{1.13}$$

and the classical solution u *is given by*

$$u(x, t) = u_0(x - t\, a(x, t)). \tag{1.14}$$

Since the slopes of the characteristic lines depends on the initial condition u_0*, we may have both the situations occuring in Figure 1.1. As shown in Figure 1.1, two characteristics starting at different points can intersect if* a *is not constant. If there exist* $x_1 < x_2$ *such*

that

$$m_1 := \frac{1}{a(u_0(x_1))} < \frac{1}{a(u_0(x_2))} := m_2, \tag{1.15}$$

*then, the two characteristics C_1 and C_2 have, respectively, the slopes m_1 and m_2. It is clear that they have to intersect each other, at some point P. At P, the solution u should have both values $u_0(x_1)$ and $u_0(x_2)$, which means that the solution is no longer unique. In this case, the MOC fails because it assumes that the solution is smooth at all times, while the solution is clearly discontinuous. At P, a so-called **jump** occurs.*

1.2 Weak and Entropy solutions

Now that it is clear why the notion of classical solution is not enough: we can have jumps occuring in the solution even if we start from smooth initial conditions; therefore, we need to introduce a different notion that helps us to model such phenomena. In order to deal with jumps we introduce the concept of *weak solutions*, which also allow to have discontinuous solutions.

Remark 6. *A classical solution is also a weak solution, but the opposite is not true.*

The problems that arise from the MOC comes from the assumption that the solution is smooth and writing the hyperbolic conservation law in quasi-linear form. To handle jumps we recast (1.5) in its integral form using the Gauss divergence theorem, which motivates the designations of balance law

$$\begin{aligned}\int_{\mathbf{\Omega}} \frac{\partial \mathbf{u}}{\partial t}(\mathbf{x}, t)\,\mathrm{d}\mathbf{x} &= -\int_{\mathbf{\Omega}} \nabla \cdot \mathcal{F}(\mathbf{x}, t, \mathbf{u})\,\mathrm{d}\mathbf{x} + \int_{\mathbf{\Omega}} \mathcal{S}(\mathbf{x}, t, \mathbf{u})\,\mathrm{d}\mathbf{x} \\ &= -\int_{\partial\mathbf{\Omega}} \mathcal{F}(\mathbf{x}, t, \mathbf{u}) \cdot \mathbf{n}\,\mathrm{d}S + \int_{\mathbf{\Omega}} \mathcal{S}(\mathbf{x}, t, \mathbf{u})\,\mathrm{d}\mathbf{x},\end{aligned} \tag{1.16}$$

and conservation law

$$\begin{aligned}\int_{\mathbf{\Omega}} \frac{\partial \mathbf{u}}{\partial t}(\mathbf{x}, t)\,\mathrm{d}\mathbf{x} &= -\int_{\mathbf{\Omega}} \nabla \cdot \mathcal{F}(\mathbf{x}, t, \mathbf{u})\,\mathrm{d}\mathbf{x} \\ &= -\int_{\partial\mathbf{\Omega}} \mathcal{F}(\mathbf{x}, t, \mathbf{u}) \cdot \mathbf{n}\,\mathrm{d}S.\end{aligned} \tag{1.17}$$

Equation (1.16) points out that the rate of change of the integral of $\mathbf{u}$ is given by the flux over the boundary and additional source terms. Furthermore, if the source terms vanish, as in (1.17), the time variation of $\mathbf{u}$ is equal to the flux loss through the boundary $\partial\mathbf{\Omega}$. We can notice that, if the solution is smooth, the integral and quasi-linear forms are equal but we cannot move from one formulation to the other if jumps occur in the solution.

Since we know that we use the MOC for the smooth solution, we then need a condition that specifies how the jumps are moving in time. To study the behaviour of the solution at discontinuities, we focus on piecewise smooth functions $\mathbf{u}$: there exists a finite number of smooth hypersurfaces Σ in $\mathbb{R}^d \times \mathbb{R}^+$ outside which $\mathbf{u}$ is continuously differentiable, and along which $\mathbf{u}$ has a jump. We denote by $\mathfrak{n} = (\mathfrak{n}_t, \mathfrak{n}_1, \ldots, \mathfrak{n}_d)$ the normals to the hypersurfaces Σ. The theorem that describes how jumps are travelling through the (x, t) plane is the *Rankine-Hugoniot condition.*

Theorem 7 (Rankine-Hugoniot condition)**.** *If $\mathbf{u} : \boldsymbol{\Omega} \times [0, T] \to \mathcal{U}$ is a piecewise smooth function. Then,* $\mathbf{u}$ *is a solution of the hyperbolic conservation law*

$$\partial_t \mathbf{u}(\mathbf{x}, t) + \sum_{i=1}^{d} \partial_{x_i} \mathcal{F}_i(\mathbf{x}, t, \mathbf{u}) = 0, \tag{1.18}$$

if and only if

- $\mathbf{u}$ *is a classical solution of* (1.18) *where* $\mathbf{u}$ *is continuously differentiable, and*
- $\mathbf{u}$ *satisfies the* ***Rankine-Hugoniot condition***

$$\mathfrak{n}_t(\mathbf{u}_+ - \mathbf{u}_-) + \sum_{i=1}^{d} \mathfrak{n}_i \left(\mathcal{F}_i(\mathbf{u}_+) - \mathcal{F}_i(\mathbf{u}_-)\right) = 0, \tag{1.19}$$

  along the smooth hypersurface Σ where $\mathbf{u}$ *is discontinuous.*

We can prove the Rankine-Hugoniot condition with the following example for one-dimensional scalar conservation laws.

Example 8. *Let us consider again the hyperbolic scalar conservation law* (1.7) *integrated over the one-dimensional domain $[a, b]$, which reads*

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \int_a^b u(x, t)\, \mathrm{d}x &= -\left[f(u(x, t))\right]_a^b \\ &= -\left[f(u(b, t)) - f(u(a, t))\right]. \end{aligned} \tag{1.20}$$

Suppose we have a jump travelling through the (x, t) plane along the trajectory $x_s(t)$ (see Figure 1.2). Since a jump is present in the solution we cannot integrate accross it because we have a left-limiting value and a right-limiting value. Therefore we can recast the left-hand side of Equation (1.20) *using the* ***Leibnitz integral rule*** *for differentiation under*

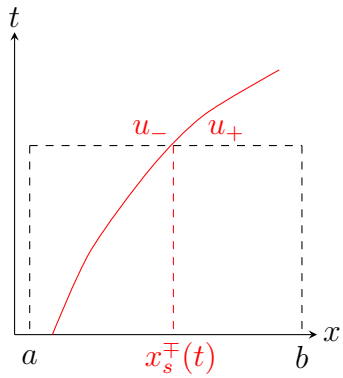


Figure 1.2: Jump moving through the (x,t) plane.

the integral sign:

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}\int_a^{x_s^-(t)} u(x,t)\,\mathrm{d}x + \frac{\mathrm{d}}{\mathrm{d}t}\int_{x_s^+(t)}^b u(x,t)\,\mathrm{d}x\\ &= \int_a^{x_s^-(t)} \frac{\partial}{\partial t}u(x,t)\,\mathrm{d}x + u(x_s^-(t),t)\frac{\mathrm{d}}{\mathrm{d}t}x_s(t) + \int_{x_s^+(t)}^b \frac{\partial}{\partial t}u(x,t)\,\mathrm{d}x - u(x_s^+(t),t)\frac{\mathrm{d}}{\mathrm{d}t}x_s(t),\end{aligned} \tag{1.21}$$

where $u(x_s^-(t),t) = u_-$ *and* $u(x_s^+(t),t) = u_+$*, and* $w = \frac{\mathrm{d}}{\mathrm{d}t}x_s(t)$ *is the* ***speed of the jump****.*

Since in the subdomains $[a, x_s(t))$ *and* $(x_s(t), b]$ *the solution is continuous we can use the quasi-linear form* $\frac{\partial u}{\partial t} = -\frac{\partial f(u)}{\partial x}$ *to derive the Ranking-Hugoniot condition from* (1.20) *and* (1.21)

$$\begin{aligned}&\int_a^{x_s^-(t)} \frac{\partial}{\partial x}f(u)\,\mathrm{d}x + u_-w + \int_{x_s^+(t)}^b \frac{\partial}{\partial x}f(u)\,\mathrm{d}x - u_+w = -\left[f(u(b,t)) - f(u(a,t))\right]\\ &-\left[f(u_-) - f(u(a,t))\right] + wu_- - \left[f(u(b,t)) - f(u_+)\right] - wu_+ = -\left[f(u(b,t)) - f(u(a,t))\right].\end{aligned} \tag{1.22}$$

By cancelling out the equal flux terms, we end up with

$$f(u_+) - f(u_-) - w(u_+ - u_-) = 0 \tag{1.23}$$

which is the Ranking-Hugoniot condition (1.19) *for one-dimensional scalar conservation law.*

For simplifity, we denote the jump of quantities $[\![u]\!] := u_+ - u_-$ *and* $[\![f(u)]\!] := f(u_+) - f(u_-)$*. With these notation, the Ranking-Hugoniot condition can be formulated through*

$$[\![f(u)]\!] - w[\![u]\!] = 0 \quad \Longrightarrow \quad w = \frac{[\![f(u)]\!]}{[\![u]\!]} \tag{1.24}$$

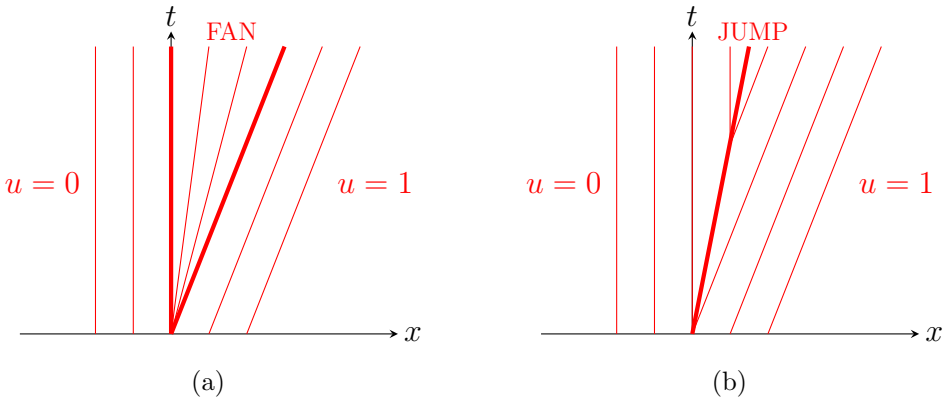


Figure 1.3: Multiple satisfied weak solutions for the Riemann problem (1.26).

When working with piecewise constant initial conditions, several problems can arise.

Definition 9 (Riemann problem)**.** *A Cauchy problem made up by a hyperbolic conservation law* (1.18) *along with piecewise constant initial data characterized by a single discontinuity in the domain is called* ***Riemann problem****.*

Example 10. *For one-dimensional scalar conservation law, Riemann problems*[1] *have the following general form*

$$\partial_t u(x,t) + \partial_x f(u(x,t)) = 0, \quad u(x,0) = u_0(x) = \begin{cases} u_-, & \text{if } x < 0, \\ u_+, & \text{if } x > 0. \end{cases} \tag{1.25}$$

By considering that jumps may occur in the solution we admit much more possible solutions. However, it is straightforward to prove with a simple example that, in general, weak solutions are not unique.

Example 11. *We consider the one-dimensional Riemann problem equipped with the Burger's equation and discontinuous initial conditions*

$$\partial_t u(x,t) + \partial_x \left(\frac{1}{2} u^2(x,t) \right) = 0, \quad u(x,0) = u_0(x) = \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{if } x > 0. \end{cases} \tag{1.26}$$

Figure (1.3) *clearly shows that this Riemann problem may have multiple weak solutions that are always satisfied. Indeed, the solution could be either a* ***rarefaction fan****, which is a particolar solution described by* $u(x,t) = x/t$*, or a* ***jump*** *with shock speed* $w = 0.5$*. Unfortunately, infinite weak solution can be built starting from these initial conditions.*

[1] These problems are named after Bernhard Riemann who used these initial data to study the behaviour of two gases separated by a membrane. Moreover, it should be noticed that Riemann problems naturally appear in finite volume schemes, discussed in Chapter 2.

Therefore, weak solution are not unique and, for this reason, we need more conditions to pick the *right* weak solution. Here is when the concept of entropy becomes fundamental. For this reason, we introduce a new class of solutions, namely the *entropy solutions*.

Since we cannot prove the uniqueness of weak solutions, we have to find additional constraints to select, among the infinite possible weak solutions, the only physically relevant one. In fact, in reality some viscosity is always associated with a given conservation law. Therefore, instead of (1.18), we always have

$$\partial_t \mathbf{u}^\epsilon + \sum_{i=1}^{d} \partial_{x_i} \mathcal{F}_i(\mathbf{u}^\epsilon) = \sum_{i=1}^{d} \epsilon \partial^2_{x_i} \mathbf{u}^\epsilon, \tag{1.27}$$

where $\mathbf{u}^\epsilon$ is the *entropy solution* and the zero viscosity limit, $\epsilon \to 0$, is only a convenient mathematical idealization. An entropy condition states that the physical solution satisfies a general principle of thermodynamics that the entropy always decreases. For the sake of simplicity, the following definitions and theorem are given for a scalar equation.

Definition 12 (Entropy)**.** *Assume that Ψ is a convex domain, a convex function $\varrho : \Psi \to \mathbb{R}$ is called* ***entropy function*** *for the hyperbolic conservation law* (1.18) *if there exist d functions $g_i : \Psi \to \mathbb{R}$, $i = 1, \ldots, d$, such that*

$$\varrho'(u) f_i'(u) = g_i'(u), \qquad \text{for } 1 \leq i \leq d, \tag{1.28}$$

holds. The function g is called ***entropy flux*** *and the pair (ϱ, g) is denoted* ***entropy pair****.*

There are many versions of the entropy condition for a given conservation law that have been developed and applied [189, 194, 195, 199]. Herein, we recall only few of them.

- Given an entropy pair (ϱ, g), a solution u for the conservation law (1.7) is said to be the entropy solution if it satisfies

$$\frac{\partial \varrho(u)}{\partial t} + \frac{\partial g(u)}{\partial x} \leq 0.$$

- The *Lax's entropy condition* states that

  Definition 13. *A solution for the Riemann problem with left and right states u_- and u_+ is said to be the entropy solution if the shock speed, w, satisfies*

$$f'(u_+) < w < f'(u_-).$$

With the introduction of the entropy solution concept, we are able to discard the unphysical weak solutions. Furthermore, Kružkov [189] showed that, for a scalar quasi-linear equation in multiple space dimensions, a unique entropy solution exists. Here, we refer to the theorem provided in [156]:

Theorem 14 (Existence and uniqueness of entropy solutions)**.** *Given the function u_0 locally bounded. Then, the Cauchy problem given by* (1.18) *and* (1.2) *has a unique entropy solution u [33].*

1.3 Physical models used in this thesis

In this section, we give few examples of the models that will be considered in this work. We used them as benchmark problems to asses the new numerical techniques that will be discussed in Parts II and III. We focus on one-dimensional, two-dimensional, and three-dimensional problems.

1.3.1 Euler equations for gasdynamics

The first example of non-linear system of hyperbolic conservation laws considered herein is the multi-dimensional *Euler equations for gasdynamics*. This model arises from the Navier-Stokes equations when viscous forces are neglected. In particular, when the viscosity vanishes, the Reynolds number approaches infinity and the Navier-Stokes system boils down to the Euler equations. This simplified model is used to solve many fluid dynamics problems involving low viscosity, however in general the assumed negligible viscosity is no longer valid in the region of fluid close to a solid boundary. Although in nature there are limited examples of inviscid fluids, e.g. superfluids, this system of equations is often used in the community of researchers studying hyperbolic equations for its mathematical features. Indeed, these equations are one of the most investigated system in computational fluid dynamics.

The Euler equations for gasdynamics are described by the conservation law (1.6), with conserved variables and fluxes that are given by

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho E \end{bmatrix}, \qquad \mathcal{F}(\mathbf{u}) = \begin{bmatrix} \mathbf{F} & \mathbf{G} \end{bmatrix} = \begin{bmatrix} \rho v_x & \rho v_y \\ \rho v_x^2 + p & \rho v_x v_y \\ \rho v_x v_y & \rho v_y^2 + p \\ \rho v_x H & \rho v_y H \end{bmatrix}, \tag{1.29}$$

for two-dimensional problems, and

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ \rho E \end{bmatrix}, \qquad \mathcal{F}(\mathbf{u}) = \begin{bmatrix} \mathbf{F} & \mathbf{G} & \mathbf{M} \end{bmatrix} = \begin{bmatrix} \rho v_x & \rho v_y & \rho v_z \\ \rho v_x^2 + p & \rho v_x v_y & \rho v_x v_z \\ \rho v_x v_y & \rho v_y^2 + p & \rho v_y v_z \\ \rho v_x v_z & \rho v_y v_z & \rho v_z^2 + p \\ \rho v_x H & \rho v_y H & \rho v_z H \end{bmatrix}, \tag{1.30}$$

for three-dimensional problems.

ρ denotes the mass density, $\mathbf{v} = (v_x, \quad v_y, \quad v_z)$ the velocity, p the pressure, and with $E = e + k$ we define the specific total energy, e being the specific internal energy, and $k = \mathbf{v} \cdot \mathbf{v}/2$ being the kinetic energy. Finally, the total specific enthalpy is $H = h + k$, with $h = e + p/\rho$ the specific enthalpy. For simplicity in this thesis we work with the classical perfect gas equation of state:

$$p = (\gamma - 1)\rho e, \tag{1.31}$$

with γ the constant (for a perfect gas) ratio of specific heats.

The Euler equations can be derived from the related conservation law.

- *Conservation of mass*: it is a well-known fact that the total mass of the fluid is conserved.
- *Conservation of momentum*: following the Newton's second law of motion, the rate of change of the momentum is only dictated by the fluid pressure, in the absence of external forces.
- *Conservation of energy*: the total energy is made up by two components, the internal (potential) energy and the kinetic energy.

In order to study the hypebolicity of this system, we consider the one-dimensional Euler equations recast in quasi-linear form.

$$\partial_t \mathbf{u} + \mathcal{A}(\mathbf{u})\, \partial_x \mathbf{u} = 0, \tag{1.32}$$

where $\mathcal{A}(\mathbf{u}) = \partial_{\mathbf{u}} \mathcal{F}(\mathbf{u})$ is the jacobian matrix that reads

$$\mathcal{A}(\mathbf{u}) = \begin{bmatrix} 0 & 0 & 1 \\ v_x[-H + (\gamma - 1)k] & \gamma v_x & H - (\gamma - 1)v_x^2 \\ \frac{\gamma - 3}{2} v_x^2 & \gamma - 1 & (3 - \gamma)v_x \end{bmatrix} \tag{1.33}$$

As already stated, system (1.32) is hyperbolic, meaning that the matrix $\mathcal{A}$ has only real eigenvalues and can be diagonalized by $\mathbf{R}$, the matrix of eigenvectors.
The eigenvalues are represented as a diagonal matrix, in matricial form,

$$\Lambda = \mathrm{diag}\,(\lambda_1, \quad \lambda_2, \quad \lambda_3) = \mathrm{diag}\left(v_x - \sqrt{\gamma p/\rho}, \quad v_x, \quad v_x + \sqrt{\gamma p/\rho}\right), \tag{1.34}$$

where $a = \sqrt{\gamma p/\rho}$ is the *speed of sound*, corresponding to three different waves along which information are propagated:

- One slow running *acoustic wave* travelling along $v_x - a$,
- An *entropy wave* travelling along v_x,
- One fast running *acoustic wave* travelling along $v_x + a$.

Whereas the matrices of the right $\mathbf{R}$ and left $\mathbf{L}$ eigenvectors are:

$$\mathbf{R} = \begin{bmatrix} 1 & \frac{\rho}{a} & \frac{\rho}{a} \\ k & \rho[\frac{H}{a} + v_x] & \rho[\frac{H}{a} - v_x] \\ v_x & \rho(\frac{v_x}{a} + 1) & \rho(\frac{v_x}{a} - 1) \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 1 - \frac{\gamma-1}{2}M^2 & -\frac{\gamma-1}{a^2} & \frac{\gamma-1}{a^2}v_x \\ \frac{1}{2\rho}[\frac{\gamma-1}{a}k - v_x] & \frac{1}{2\rho}\frac{\gamma-1}{a} & \frac{1}{2\rho}[-\frac{\gamma-1}{a}v_x + 1] \\ \frac{1}{2\rho}[\frac{\gamma-1}{a}k + v_x] & \frac{1}{2\rho}\frac{\gamma-1}{a} & \frac{1}{2\rho}[-\frac{\gamma-1}{a}v_x - 1] \end{bmatrix}, \tag{1.35}$$

where $M = v_x/a$ is the *Mach number*. The Mach number is a dimensionless number governing this kind of flow, which allows for a classification in two flow regimes: subsonic ($M < 1$) and supersonic ($M > 1$).

Hence, the jacobian $\mathcal{A}$ can be recast using the eigenvector decomposition:

$$\mathcal{A} = \mathbf{R}\Lambda\mathbf{L} \qquad \text{with} \qquad \mathbf{R} = \mathbf{L}^{-1}. \tag{1.36}$$

Equation (1.36) allows to write the quasi-linear system (1.32) in characteristic form, by multiplying everything by $\mathbf{L}$,

$$\mathbf{L}\partial_t\mathbf{u} + \Lambda\mathbf{L}\partial_x\mathbf{u} = 0 \quad \Rightarrow \quad \partial_t\mathcal{W} + \Lambda\partial_x\mathcal{W} = 0. \tag{1.37}$$

We can notice that (1.37) is a set of three scalar advection equations:

$$\frac{\partial w^{(i)}}{\partial t} + \lambda^{(i)}\frac{\partial w^{(i)}}{\partial x} = 0, \qquad i = 1, 2, 3 \tag{1.38}$$

Each one of them describes how the solution evolves along the three characteristic waves.

It is possible to draw similar considerations by studying the full three-dimensional system [265] but, for the sake of simplicity, we will not consider it here.

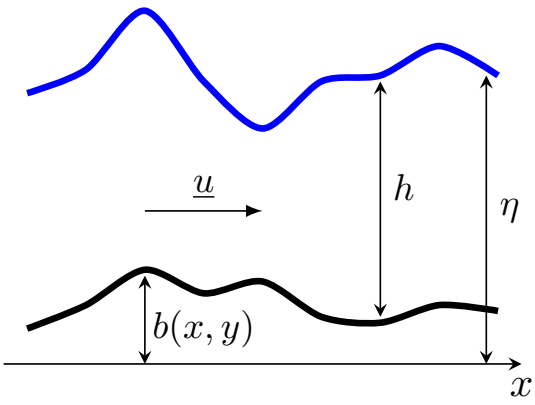


Figure 1.4: Shallow water equations: definition of the variables.

1.3.2 Shallow water equations

A simplification of the Euler equations leads to the *shallow water equations*. To derive the one-dimensional equations, we assume that the vertical velocity of a fluid in a channel is negligible whereas, the horizontal velocity is roughly constant through any vertical cross section. Indeed, this is a very good approximation if we study flows that are shallow with respect to the wave length. Another important assumption to derive the shallow water system is considering the fluid incompressible, meaning that the density ρ is constant.

The shallow water equations can be used to model free surface flows under the action of gravity. This system takes the form of a hyperbolic balance law with source terms (1.5), modeling the effects of bathymetry and friction.

Conserved variables, fluxes and source terms read

$$\mathbf{u} = \begin{bmatrix} h \\ hv_x \\ hv_y \end{bmatrix}, \quad \mathcal{F}(\mathbf{u}) = \begin{bmatrix} \mathbf{F} & \mathbf{G} \end{bmatrix} = \begin{bmatrix} hv_x & hv_y \\ hv_x^2 + g\frac{h^2}{2} & hv_xv_y \\ hv_xv_y & hv_y^2 + g\frac{h^2}{2} \end{bmatrix}, \tag{1.39}$$

$$\mathcal{S}(\mathbf{u}, \mathbf{x}) = \begin{bmatrix} 0 \\ S_x(\mathbf{u}, x) \\ S_y(\mathbf{u}, x) \end{bmatrix} = -gh \begin{bmatrix} 0 \\ \frac{\partial b}{\partial x}(\mathbf{x}) \\ \frac{\partial b}{\partial y}(\mathbf{x}) \end{bmatrix} - g \begin{bmatrix} 0 \\ \frac{n^2}{h^{7/3}}|hv_x|hv_x \\ \frac{n^2}{h^{7/3}}|hv_y|hv_y \end{bmatrix}, \tag{1.40}$$

for two-dimensional problems.

h denotes the relative water height, $\mathbf{v} = (v_x, \quad v_y)$ are the flow speed components, and together they make up the discharge $h\mathbf{v} = (hv_x, \quad hv_y)$. Finally, g is the gravity acceleration, $b(\mathbf{x})$ is the local bathymetry, and n is the Manning friction coefficient. The source term accounts for the effects of variations of the bathymetry and of bottom friction. Finally, it is also convenient to introduce the free surface water level $\eta := h + b$. All the aforementioned variables can be better interpreted by looking at Figure 1.4. In analogy

to Section 1.3.1, here we present a similar eigenvectors decomposition carried out for the one-dimensional shallow water system, which can be rewritten in quasi-linear form as

$$\partial_t \mathbf{u} + \mathcal{A}(\mathbf{u})\, \partial_x \mathbf{u} = \mathcal{S}, \tag{1.41}$$

where the jacobian matrix $\mathcal{A}(\mathbf{u}) = \partial_{\mathbf{u}} \mathcal{F}(\mathbf{u})$ in this case reads

$$\mathcal{A}(\mathbf{u}) = \begin{bmatrix} 0 & 1 \\ gh - v_x^2 & 2v_x \end{bmatrix} \tag{1.42}$$

For this case, the eigenvalues are

$$\Lambda = \mathrm{diag}\,(\lambda_1, \quad \lambda_2) = \mathrm{diag}\left(v_x - \sqrt{gh}, \quad v_x + \sqrt{gh}\right), \tag{1.43}$$

Whereas the matrices of the right $\mathbf{R}$ and left $\mathbf{L}$ eigenvectors are:

$$\mathbf{R} = \begin{bmatrix} 1 & 1 \\ v_x - a & v_x + a \end{bmatrix}, \quad \mathbf{L} = \frac{1}{2a} \begin{bmatrix} v_x + a & -1 \\ -(v_x - a) & 1 \end{bmatrix}, \tag{1.44}$$

where $a = \sqrt{gh}$ is the *celerity*, the corresponding speed of sound for shallow water flows. The celerity allows to define the *Froude number* $Fr = v_x/a$, the corresponding Mach number for shallow water flows. Also in this case, the Froude number allows for a classification of the fluid in two regimes: subcritical ($Fr < 1$) and supercritical ($Fr > 1$).

The same considerations to obtain the hyperbolic system in characteristic form can be applied here. Also in this case, for simplicity, we avoid doing the eigenvector decomposition for the two-dimensional system [58].

Chapter 2

Numerical Schemes for Hyperbolic Problems

In this chapter, we present state-of-the-art numerical methods for hyperbolic conservation and balance laws and theoretical results for several types of schemes. Many schemes and studies can be found in the literature [287]. However, most of them are based on finite difference (FD), finite volume (FV) and finite element approaches (FE), and we herein only focus on FV and FE-type. The books [199, 119] and references therein provide more insights about classical and modern approaches used in this field. Part II is related to FE based schemes such as discontinuous Galerkin (DG) and residual distribution (RD) schemes, whereas in Part III we consider FV schemes. For all the contributions of this work, we are going to focus on the state-of-the-art of these schemes to address other problems that arise for hyperbolic problems. In the last part of this chapter, we also present the different time integration methods used in conjunction with the spatial discretizations mentioned.

2.1 Domain discretization

The spatial domain $\mathbf{\Omega}$ is discretized by means of a tessellation $\mathscr{T}$ composed of $\mathscr{N}$ non-overlapping simplicial (triangles/squares in 2D, tetrahedra/cubes in 3D) elements. We denote by K the generic element, and set $\mathbf{\Omega}_h = \bigcup K$. Note that in general $\mathbf{\Omega}_h \neq \mathbf{\Omega}$. The difference is often due to the approximation of the boundary.

2.2 Finite Volume (FV) schemes

Finite Volume (FV) schemes are commonly used in many engineering applications, and still the object of intense research [157, 247, 190].

For FV schemes, the computational domain $\mathbf{\Omega}_h$ is split in non-overlapping elements K, called cells or control volumes. It is easy then to simplify the generic multi-dimensional formulation to obtain the Cartesian one (and the one-dimensional one).

To derive all FV methods we start from the divergence form of the hyperbolic equation (1.3) and integrate over the control volume K by keeping the time derivative continuous. We obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_K \mathbf{u}(\mathbf{x},t)\,\mathrm{d}\mathbf{x} + \int_K \sum_{i=1}^{d} \mathcal{F}_i(\mathbf{x},t,\mathbf{u})\,\mathrm{d}\mathbf{x} = \int_K \mathcal{S}(\mathbf{x},t,\mathbf{u})\,\mathrm{d}\mathbf{x}. \tag{2.1}$$

For the control volume K we define the so-called *cell-average* at time t as

$$\bar{\mathbf{u}}_K(t) := \frac{1}{|K|}\int_K \mathbf{u}(\mathbf{x},t)\,\mathrm{d}\mathbf{x}, \tag{2.2}$$

where $|K|$ denotes the measure (length, area, volume). On K, the cell-average is an approximated mean value, therefore a FV method is an evolution equation of cell averages. Equation (2.1) can be recast, by applying the Gauss divergence theorem, as

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{u}}_K(t) = -\sum_{i=1}^{d} \frac{1}{|K|}\int_{\partial K} \mathcal{F}_i(\mathbf{x},t,\mathbf{u})\cdot\mathbf{n}_i\,\mathrm{d}S + \frac{1}{|K|}\int_K \mathcal{S}(\mathbf{x},t,\mathbf{u})\,\mathrm{d}\mathbf{x}. \tag{2.3}$$

Since the approximation is not continuous across ∂K , we have to discretize the right hand side of (2.3), called *residual*, which represents the flux across the boundary of the cell. We denote by e the edges (surfaces) of the cell such that $\partial K = \bigcup \Gamma_{K,L}$, where

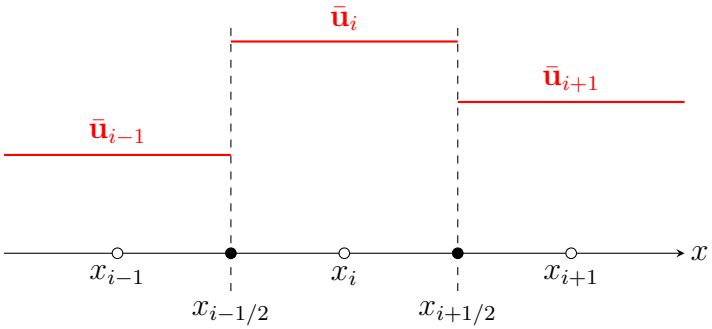


Figure 2.1: Schematic visualization of piece-wise constant reconstruction in 1D.

$\Gamma_{K,L} = \partial K \cap \partial L$ is the face separating K and L. Hence, we can rewrite (2.3) as

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{u}}_K(t) = -\sum_{i=1}^{d} \frac{1}{|K|} \sum_{\substack{e \subset \partial K, \\ e = \Gamma_{K,L}}} \int_{\Gamma_{K,L}} \mathcal{F}_i(\mathbf{x}, t, \mathbf{u}) \cdot \mathbf{n}_i \, \mathrm{d}S + \frac{1}{|K|} \int_K \mathcal{S}(\mathbf{x}, t, \mathbf{u}) \, \mathrm{d}\mathbf{x}. \tag{2.4}$$

It should be noticed that (2.4) is an exact relation. However, in order to compute the flux term, we need the solution at each interface of each cell, starting from the cell-averages $\bar{\mathbf{u}}_K$. For this reason, the introduction of the concept of *reconstruction* is necessary. The most naive, but consistent, reconstruction possible is the piece-wise constant (see Figure 2.1): the reconstructed values correspond to the cell-average of the cell itself. Therefore, we started from a continuous function and ended up with a discontinuous one, with jumps at each interface, $\mathbf{u}_h^-$ and $\mathbf{u}_h^+$. To solve the ambiguity of the flux term at each interface, we introduce the concept of *numerical flux*, which is an approximation of the flux that depends on both the reconstructed values at each interface $\mathbf{u}_h^-$ and $\mathbf{u}_h^+$.

Definition 15 (Numerical flux)**.** *We recall in particular that a* ***numerical flux*** $\hat{\mathcal{F}}$ *is a locally Lipschitz continuous function that is consistent with the flux* $\mathcal{F}$*,* $\forall \mathbf{u} \in \mathcal{U}$*, meaning that*

$$\hat{\mathcal{F}}(\mathbf{u}, \mathbf{u}) \cdot \mathbf{n} = \mathcal{F}(\mathbf{u}) \cdot \mathbf{n}, \tag{2.5}$$

Example 16 (Rusanov flux)**.** *A simple and robust numerical flux is the* ***Rusanov*** *(or* ***Local Lax-Friedrichs****) flux:*

$$\hat{\mathcal{F}}(\mathbf{u}_h^-, \mathbf{u}_h^+) \cdot \mathbf{n} = \frac{1}{2}\left(\mathcal{F}(\mathbf{u}_h^+) + \mathcal{F}(\mathbf{u}_h^-)\right) \cdot \mathbf{n} - \frac{1}{2} s_{max} \left(\mathbf{u}_h^+ + \mathbf{u}_h^-\right), \tag{2.6}$$

where s_{max} *is the maximum eigenvalue of the Jacobians of the flux,* $\mathcal{A}_{\mathbf{n}}\left(\mathbf{u}_h^+\right)$ *and* $\mathcal{A}_{\mathbf{n}}\left(\mathbf{u}_h^-\right)$*.*

Remark 17. *Godunov came up with the idea to solve Riemann problems, exactly, at each interface.*

In the initial scheme proposed by Godunov the numerical flux is defined as the analytical flux evaluated at the interface using the exact solution of the Riemann problem. For this to be feasible in practice an important condition is that the characteristics from two neighboring Riemann problems should not intersect. This condition translates on a time step constraint for the finite volume scheme based on $\lambda := \max_j |\mathcal{A}(\bar{\mathbf{u}}_j)|$.

When applying a numerical flux $\hat{\mathcal{F}}_i$, Equation (2.4) can be rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{u}}_K(t) = -\sum_{i=1}^{d}\frac{1}{|K|}\sum_{\substack{e\subset\partial K,\\ e=\Gamma_{K,L}}}\int_{\Gamma_{K,L}}\hat{\mathcal{F}}_i(\mathbf{u}_h^-,\mathbf{u}_h^+)\cdot\mathbf{n}_i\,\mathrm{d}S + \frac{1}{|K|}\int_K \mathcal{S}(\mathbf{x},t,\mathbf{u})\,\mathrm{d}\mathbf{x}. \tag{2.7}$$

Equation (2.7) is a semi-discrete formulation where the time derivative is kept continuous. In order to obtain a fully discrete scheme, we apply a time integration scheme to the ODE system (2.7).

The formulation (2.7) can be simplified a lot when considering the domain being discretized, with a two-dimensional Cartesian mesh, into $N_x \times N_y$ control volumes:

$$K_{i,j} = [x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}] \tag{2.8}$$

with the dimensions given by $\Delta x = x_{i+1/2} - x_{i-1/2}$ and $\Delta y = y_{j+1/2} - y_{j-1/2}$. For the control volume $K_{i,j}$, we define the so-called *cell-average* at time t as

$$\bar{\mathbf{u}}_{i,j}(t) := \frac{1}{\Delta x\Delta y}\int_{x_{i-1/2}}^{x_{i+1/2}}\int_{y_{j-1/2}}^{y_{j+1/2}}\mathbf{u}(\mathbf{x},t)\,\mathrm{d}\mathbf{x}. \tag{2.9}$$

Following some manipulations, for Cartesian meshes, Equation (2.7) can be recast as

$$\frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{u}}_{i,j}(t) = -\frac{1}{\Delta x}(\mathbf{F}_{i+1/2,j}(t) - \mathbf{F}_{i-1/2,j}(t)) - \frac{1}{\Delta y}(\mathbf{G}_{i,j+1/2}(t) - \mathbf{G}_{i,j-1/2}(t)) + \mathbf{S}_{i,j}(t) \tag{2.10}$$

where $\mathbf{S}_{i,j}$ is the cell average of the source terms over cell $K_{i,j}$

$$\mathbf{S}_{i,j}(t) := \frac{1}{\Delta x\Delta y}\int_{x_{i-1/2}}^{x_{i+1/2}}\int_{y_{j-1/2}}^{y_{j+1/2}}\mathcal{S}(\mathbf{x},t,\mathbf{u})\,\mathrm{d}\mathbf{x} \tag{2.11}$$

and $\mathbf{F}_{i+1/2,j}$ and $\mathbf{G}_{i,j+1/2}$ are the cell-averages of the physical fluxes over cell boundaries

at time t:

$$\mathbf{F}_{i+1/2,j}(t) := \frac{1}{\Delta y}\int_{y_{j-1/2}}^{y_{j+1/2}} \mathbf{F}(\mathbf{u}(x_{i+1/2}, y, t))\,\mathrm{d}y, \tag{2.12}$$

$$\mathbf{G}_{i,j+1/2}(t) := \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{G}(\mathbf{u}(x, y_{j+1/2}, t))\,\mathrm{d}x. \tag{2.13}$$

Equation (2.10), along with all the previous definitions, is so far exact.
$\bar{\mathbf{u}}_{i,j}$, $\mathbf{F}_{i+1/2,j}$, $\mathbf{G}_{i,j+1/2}$ and $\mathbf{S}_{i,j}$ can be then approximated to the desired order of accuracy using appropriate quadrature formulae and reconstruction techniques. Many low order and high order reconstruction techniques have been developed in the last decades. Few of them can be found in [275, 157, 293, 100, 277, 139, 31, 120, 233]. The procedure to evaluate (2.12) and (2.13) consists in discretizing the integrals over the faces using an appropriate numerical quadrature formula. To simplify the notation, We drop the time dependence. By applying a Q-points quadrature rule, one obtains the following expression for the fluxes:

$$\mathbf{F}_{i+1/2,j} \approx \frac{1}{\Delta y}\sum_{q=1}^{Q} \omega_q \mathbf{F}(\mathbf{u}(x_{i+1/2}, y_q)), \tag{2.14}$$

$$\mathbf{G}_{i,j+1/2} \approx \frac{1}{\Delta x}\sum_{q=1}^{Q} \omega_q \mathbf{G}(\mathbf{u}(x_q, y_{j+1/2})), \tag{2.15}$$

where the subscript $q = 1, \ldots, Q$ corresponds to different Gaussian points $x_q \in [x_{i-1/2}, x_{i+1/2}]$ and $y_q \in [y_{j-1/2}, y_{j+1/2}]$, and weights $\omega_q \in [0, 1]$. However, (2.14) involves point-wise values of $\mathbf{u}$ whereas (2.10) evolves the cell averages of $\mathbf{u}$. Therefore, the second step consists in reconstructing the point-wise values $\mathbf{u}(x_{i+1/2}, y_q)$ from the spatial averages. Once the reconstruction has been performed, at each face we have two sets of values of $\mathbf{u}$, corresponding to $x^-_{i+1/2}$ and $x^+_{i+1/2}$, which will be referred to as the left and right extrapolated values:

$$\mathbf{u}^-_{i+1/2,q} = \mathbf{u}(x^-_{i+1/2}, y_q), \qquad \mathbf{u}^+_{i+1/2,q} = \mathbf{u}(x^+_{i+1/2}, y_q). \tag{2.16}$$

The last step in the evaluation of the fluxes replaces $\mathbf{F}(\mathbf{u}(x_{i+1/2}, y_q))$ in (2.14), and $\mathbf{G}(\mathbf{u}(x_q, y_{j+1/2}))$ in (2.15), with a monotone and consistent numerical flux $\hat{\mathbf{F}}$, and $\hat{\mathbf{G}}$,

like the one introduced in (2.6). Finally Equations (2.14) and (2.15) can be recast as

$$\mathbf{F}_{i+1/2,j} \approx \frac{1}{\Delta y} \sum_{q=1}^{Q} \omega_q \hat{\mathbf{F}}(\mathbf{u}^-_{i+1/2,q}, \mathbf{u}^+_{i+1/2,q}), \tag{2.17}$$

$$\mathbf{G}_{i,j+1/2} \approx \frac{1}{\Delta x} \sum_{q=1}^{Q} \omega_q \hat{\mathbf{G}}(\mathbf{u}^-_{q,j+1/2}, \mathbf{u}^+_{q,j+1/2}). \tag{2.18}$$

2.2.1 Second-order MUSCL-Hancock-type scheme

In this section, we briefly recall the standard second order MUSCL-Hancock scheme [287] applied to the one dimensional version of (2.10) evolved in time from t^n to $t^{n+1} = t^n + \Delta t$:

$$\bar{\mathbf{u}}_i^{n+1} = \bar{\mathbf{u}}_i^n - \frac{\Delta t}{\Delta x} \left(\hat{\mathbf{F}}_{i+1/2}(\mathbf{u}^-_{i+1/2}, \mathbf{u}^+_{i+1/2}) - \hat{\mathbf{F}}_{i-1/2}(\mathbf{u}^-_{i-1/2}, \mathbf{u}^+_{i-1/2}) \right) + \Delta t \mathbf{S}_i^n(\mathbf{u}_i^{n+1/2}). \tag{2.19}$$

The values

$$\begin{aligned} \mathbf{u}^+_{i+1/2} &= \mathbf{u}_i^n(x_{i+1/2}, t^{n+1/2}), \\ \mathbf{u}^-_{i+1/2} &= \mathbf{u}_i^n(x_{i-1/2}, t^{n+1/2}), \\ \mathbf{u}_i^{n+1/2} &= \mathbf{u}_i^n(x_i, t^{n+1/2}), \end{aligned} \tag{2.20}$$

represent the evaluation of the reconstruction polynomial $\mathbf{u}_i^n(x,t)$, in the space-time cell $[x_{i-1/2}, x_{i+1/2}] \times [t^n, t^{n+1}]$, respectively at the two boundaries $x_{i\pm1/2}$ and in the barycenter x_i of element K_i at the time-midpoint of $[t^n, t^{n+1}]$ called $t^{n+1/2}$.
The reconstruction polynomial $\mathbf{u}_i^n(x,t)$ is computed following the MUSCL-Hancock approach [292, 293, 287], with the *minmod* slope limiter. Therefore, we can rewrite $\mathbf{u}_i^n(x,t)$ in the form of a linear polynomial in space and time, such that

$$\mathbf{u}_i^n(x,t) = \bar{\mathbf{u}}_i^n + \partial_x \bar{\mathbf{u}}_i^n (x - x_i) + \partial_t \bar{\mathbf{u}}_i^n (t - t^n). \tag{2.21}$$

In Equation (2.21) the spatial gradient is given by

$$\partial_x \bar{\mathbf{u}}_i^n = \text{minmod}\left(\frac{\Delta \bar{\mathbf{u}}^n_{i+1/2}}{\Delta x}, \frac{\Delta \bar{\mathbf{u}}^n_{i-1/2}}{\Delta x} \right), \tag{2.22}$$

with the jumps $\Delta \bar{\mathbf{u}}^n_{i-1/2} = \bar{\mathbf{u}}_i^n - \bar{\mathbf{u}}_{i-1}^n$ and $\Delta \bar{\mathbf{u}}^n_{i+1/2} = \bar{\mathbf{u}}_{i+1}^n - \bar{\mathbf{u}}_i^n$ across the left and right cell boundaries, and the *minmod* function that reads

$$\text{minmod}(a,b) = \begin{cases} 0, & \text{if } ab \le 0, \\ a, & \text{if } |a| < |b|, \\ b, & \text{if } |a| \ge |b|. \end{cases} \tag{2.23}$$

Instead, $\partial_t \bar{\mathbf{u}}_i^n$ has to be defined as an approximation of the time derivative and it is therefore computed using the discrete version of the governing equations

$$\partial_t \bar{\mathbf{u}}_i^n = -\frac{\mathbf{F}(\bar{\mathbf{u}}_i^n + \frac{1}{2}\Delta x \partial_x \bar{\mathbf{u}}_i^n) - \mathbf{F}(\bar{\mathbf{u}}_i^n - \frac{1}{2}\Delta x \partial_x \bar{\mathbf{u}}_i^n)}{\Delta x} + \mathbf{S}(\bar{\mathbf{u}}_i^n).$$

Finally, the time step Δt is chosen according to the Courant-Friedrichs-Levy (CFL) condition

$$\Delta t = \mathrm{CFL} \min_{K_i} \frac{\Delta x}{|\lambda_{max,i}|}, \quad \forall K_i \in \mathbf{\Omega}_h,$$

where $\mathrm{CFL} < 1$, and $|\lambda_{max,i}|$ is the maximum absolute value of the eigenvalues of the flux jacobian in cell K_i.

2.2.2 Weighted Essentially Non-Oscillatory (WENO) method

In this section, we discuss one of the most common reconstruction techniques used to go beyond second order while avoding severe Gibbs oscillations. As it is described in Shu's seminal paper [275] the classical WENO (as well as ENO) approach contains three major steps:

1. select a number of candidate stencils;
2. using each stencil to reconstruct a low order polynomial;
3. use a linear or nonlinear combination of the resulting set of polynomials to evaluate the numerical flux at quadrature points.

Scalar reconstruction for 2D Cartesian grids

The goal of the WENO method is to compute point-wise value of variable of interest $u(x, y)$ at Gaussian quadrature points $(x_{i+1/2}, y_\theta)$, in order to have a conservative and high order accurate procedure. In general, two ways can be followed to obtain the same result: genuine multidimensional reconstruction and dimension-by-dimension reconstruction. The former [274] relies on considering all control volumes in the multidimensional stencil at the same time to build the reconstructed variables; whereas, the latter [285, 73] is a procedure made up by successive one-dimensional reconstruction sweeps.
The dimension-by-dimension reconstruction is much simpler and less computationally expensive than the genuine multidimensional one. For this reason, we shall only focus on the latter.

The high order reconstructed variables we are looking for will be referred to as $u^L_{i+1/2,\theta}$ and $u^R_{i+1/2,\theta}$. For the left values, we need to reconstruct the variable inside the cell $\Omega_{i,j}$,

while, for the right values, similar arguments apply on the cell $\Omega_{i+1,j}$. We aim at reconstructing the variables with an accuracy of order p (p odd). So, we define a stencil of p cells

$$\{\Omega_{l_x,l_y}, \quad l_x = i-r+1,\dots,i+r-1,\ l_y = j-r+1,\dots,j+r-1\}, \tag{2.24}$$

where $2r-1=p$. For instance, WENO5 has accuracy $p=5$, with $r=3$, and uses a 5-cells stencil from $i-2$ to $i+2$.
In the first step of the two-dimensional reconstruction, a one-dimensional WENO reconstruction along the x-direction is performed obtaining the averages at cell interface $x_{i+1/2}$ with respect to the y-direction for $l_y = j-r+1,\dots,j+r-1$

$$v^R_{l_y} = \frac{1}{\Delta y}\int_{y_{l_y-1/2}}^{y_{l_y+1/2}} u(x^R_{i+1/2},y)\,\mathrm{d}y, \qquad v^L_{l_y} = \frac{1}{\Delta y}\int_{y_{l_y-1/2}}^{y_{l_y+1/2}} u(x^L_{i+1/2},y)\,\mathrm{d}y. \tag{2.25}$$

In the second sweep we perform another one-dimensional reconstruction along the y-direction in the Gaussian integration points on the y-axis ($x = x_{i+1/2}, y = y_\theta$), with $y_\theta \in [y_{j-1/2}, y_{j+1/2}]$. The reconstructed values can be, more generally, defined for each WENO sweep as the one-dimensional averages q_i of a function $q(\xi)$

$$q_i = \frac{1}{\Delta\xi}\int_{\xi_{i-1/2}}^{\xi_{i+1/2}} q(\xi)\,\mathrm{d}\xi \tag{2.26}$$

where $\Delta\xi = \xi_{i+1/2} - \xi_{i-1/2}$ is the cell size.
For each one-dimensional step of the procedure, there are r candidate stencils for reconstruction. For each of these stencils, made up by r cells, there is a correspoding polynomial of degree $(r-1)$:

$$p_m(\xi)\ , \quad m = 0,\dots,r-1. \tag{2.27}$$

The goal of the WENO reconstruction is that of using all information coming from the r stencils employed for the reconstruction, in opposition to the ENO procedure that always chooses the best stencil and discards the rest. For this reason, the WENO approach defines the reconstructed value as a convex combination of the r values of all polynomials in each quadrature point, weighted with positive nonlinear weights. The weights are chosen in order to achieve $(2r-1)$th order of accuracy when the solution is smooth and to mimic the ENO idea when discontinuities occur in the field. For a given (quadrature) point $\tilde{\xi}$ the design of weights consists of three steps. Firstly, the optimal linear weights d_m are sought so that the combination of all polynomials with these weights produces the polynomial of degree $(2r-2)$ corresponding to the large stencil. However, these weights are not enough

to control Gibbs oscillations as Godunov stated "Linear numerical schemes for solving PDEs, having the property of not generating new extrema (monotone scheme), can be at most first-order accurate" [157]. Therefore, if all the optimal weights d_m are positive, the nonlinear weights ω_m can be defined as

$$\alpha_m = \frac{d_m}{(\beta_m + \varepsilon)^2} , \quad \omega_m = \frac{\alpha_m}{\sum_{k=0}^{r-1} \alpha_k} , \quad m = 0, \dots, r-1, \tag{2.28}$$

where ε is a small constant introduced to avoid division by zero (generally $\varepsilon = 10^{-6}$) and β_m are the smoothness indicators

$$\beta_m = \sum_{k=1}^{r-1} \int_{\xi_{i-1/2}}^{\xi_{i+1/2}} \left(\frac{\mathrm{d}^k}{\mathrm{d}x^k} p_m(\xi) \right)^2 \Delta\xi^{2k-1} \mathrm{d}\xi , \quad m = 0, \dots, r-1. \tag{2.29}$$

Remark 18. *The optimal linear weights d_m depend on the specific (quadrature) point $\tilde{\xi}$.*

If some of d_m are negative then a special procedure must be used to tackle the reconstruction problem [274].
The final WENO reconstructed quantity is given by

$$q(\tilde{\xi}) = \sum_{k=0}^{r-1} \omega_k p_k(\tilde{\xi}). \tag{2.30}$$

In multiple space dimensions, the one-dimensional WENO procedure is applied during each one-dimensional sweep. For the first sweep, the weights are designed to obtain reconstructed values at $x_{i+1/2}$; the linear weights d_m and smoothness indicators β_m can be found in [179, 28] up to $r = 6$. For the second sweep, which will be different from the first one, the reconstruction is designed to achieve $(2r-1)$th order for Gaussian points y_α. The values of the optimal weigths are computed for a specific Gaussian integration formula used to discretize the spatial integrals. The numerical experiments presented herein have been performed through a piece-wise parabolic WENO recontruction ($r = 3$), which formally correponds to fifth order accurate approximation for smooth solutions. However, in order to actually retain the fifth-order accuracy the quadrature formulae must be consistent with the WENO reconstruction. As Titarev and Toro stated in [285], the best results in terms of accuracy and computational cost for $r = 3$ are obtained if the following two-point Gaussian quadrature rule is used:

$$\int_{-1}^{1} \varphi(\xi) \, \mathrm{d}\xi \approx \varphi\left(-\frac{1}{\sqrt{3}} \right) + \varphi\left(+\frac{1}{\sqrt{3}} \right). \tag{2.31}$$

However, Equation (2.31) leads eventually to a formal fourth order of accuracy. The interested reader can deepen into [285] for optimal weights and smoothness indicators for this specific situation. One may think that using a three-point Gaussian quadrature rule would be enough to obtain the accuracy sought. In this case, we would have:

$$\int_{-1}^{1} \varphi(\xi)\,\mathrm{d}\xi \approx \frac{5}{9}\varphi\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}\varphi(0) + \frac{5}{9}\varphi\left(+\sqrt{\frac{3}{5}}\right). \tag{2.32}$$

Although the reasoning seems consistent, the WENO procedure turned out to be a reconstruction with negative weights, which can be tackled only with special *ad-hoc* procedure [274]. Instead, the four-point Gaussian quadrature rule is provided with positive optimal weights and it is given by:

$$\begin{aligned} &\int_{-1}^{1} \varphi(\xi)\,\mathrm{d}\xi \approx \sum_{\theta=1}^{4} \omega_\theta \varphi(\xi_\theta^q), \quad \text{with} \\ &\xi_{1,4} = \mp\sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}, \quad \xi_{2,3} = \mp\sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}, \\ &\omega_{1,4} = \frac{18+\sqrt{30}}{36}, \quad \omega_{2,3} = \frac{18-\sqrt{30}}{36}. \end{aligned} \tag{2.33}$$

Up to our knowledge, the quadrature described by (2.31) and (2.32) have already been thoroughly discussed in many references cited above. The case with 4 quadrature points in (2.33) has not been fully described in literature, hence, we are going to introduce all the coefficients needed to use such formula in Appendix A.4 and a Matlab script to compute the weights and coefficients for all orders is provided in [88].

Remark 19 (Positivity limiter)**.** *We aim at a positive solution and during the reconstruction procedure, it might happen that $q(x^L_{i+1/2})$ or $q(x^R_{i+1/2})$ become negative. In order to ensure that positive cell averages lead to positive reconstructions at the cell interfaces, we use the positivity limiter introduced by Perthame and Shu [247] and developed for two dimensional problems in [306]. We refer to [300, 306] for details on the implementation. This limiter when used in combination with the forward Euler (FE) method restricts the CFL conditions to $CFL^{FE} := w_1^{Lobatto}$ the weight of the Gauss–Lobatto quadrature rule of the corresponding space accuracy. For instance, with WENO5, $CFL^{FE} = 1/12$. The restriction slightly improves for high order SSPRK methods (see Section 2.5.1 for more information about* ***Runge-Kutta*** *methods), for example we have $CFL^{SSPRK(5,4)} \approx 1.508/12$. Unfortunately, explicit SSPRK methods are at most fourth order accurate, so for fifth order schemes (as DeC5, see Section 2.5.2 for further insights about* ***deferred correction*** *methods), there is no warranty that the solution stays nonnegative under any CFL condi-*

tion.

2.3 Discontinuous Galerkin (DG) schemes

In many practical and industrial applications, FV methods are still the most used schemes. Nevertheless, for long-time simulations and high-order accuracy they are quite costly and requires more efforts to be parallelized. In order to use modern computer power more adequately, there is a great demand for high-order methods. These schemes have the potential to providing accurate solutions with reasonable computational costs. In the finite element framework, one of the most favorable scheme is the Discontinuous Galerkin (DG) introduced by Reed and Hill [256] in 1973 to solve the hyperbolic neutron transport equation in a nuclear reactor. Its capability to go easily high-order and its local formulation, which makes it straightforward to parallelize, are the main features that make these methods so appealing for the hyperbolic community. The mathematical foundations of these methods were grouped in a series of paper published by Shu and collaborators [95, 93, 96, 97] around the 90s. In [93] and [95], it was proven that the resulting class of DG methods is formally high-order accurate in smooth regions, total-variation bounded in one space dimension, and maximum-norm bounded in any number of space dimensions.

In this section, we introduce the *discontinuous Galerkin* (DG) framework [97, 31]. To obtain semi-discrete equations, we replace $\mathbf{u}$ by a discrete approximation $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h^p \times \mathbb{R}^{d+2}$, where $\boldsymbol{\mathcal{V}}_h$ is the broken space of broken polynomials of degree p within each element K, and discontinuous across faces. The DG weak formulation can be derived by multiplying (1.5) with a test function $\mathbb{v}$ and integrate over the domain $\boldsymbol{\Omega}_h$: find $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h^p \times \mathbb{R}^{d+2}$ such that $\forall K$, and $\forall \mathbb{v}_h \in \boldsymbol{\mathcal{V}}_h^p$ we have

$$\int_K \mathbb{v}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t}\,\mathrm{d}\mathbf{x} + \int_K \mathbb{v}_h \mathcal{F}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x} = \int_K \mathbb{v}_h \mathcal{S}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x}, \qquad \forall \mathbb{v}_h. \tag{2.34}$$

To simplify the notation, we dropped the space and time dependence.
Then, when integrating by parts the second integral of (2.34) and applying the Gauss divergence theorem, we end up with

$$\int_K \mathbb{v}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t}\,\mathrm{d}\mathbf{x} + \int_{\partial K} \mathbb{v}_h \mathcal{F}(\mathbf{u}_h)\cdot\mathbf{n}\,\mathrm{d}S - \int_K \nabla\mathbb{v}_h \cdot \mathcal{F}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x} = \int_K \mathbb{v}_h \mathcal{S}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x}, \quad \forall \mathbb{v}_h, \tag{2.35}$$

with $\mathcal{F}(\mathbf{u}_h)$ being approximated, due to the discontinuity of the solution, with a consistent numerical flux (2.6) (as it was done in the FV case) which depends on the face values

of the internal state $\mathbf{u}_h^-$, of the neighbouring element state $\mathbf{u}_h^+$, and $\mathbf{n}$ being the outward unit normal to ∂K.
We now set $\boldsymbol{\mathcal{V}}_h^p = \text{span}\{\psi_i\}_{1\le i\le D}$, so that

$$\mathbf{u}_h(\mathbf{x},t) = \sum_{i=1}^{D} \mathbf{u}_i(t)\psi_i(\mathbf{x}). \tag{2.36}$$

The semi-discrete equations can be written as: find $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h^p \times \mathbb{R}^{d+2}$ such that $\forall K$ we have

$$\int_K \psi_j \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t}\,\mathrm{d}\mathbf{x} + \int_{\partial K} \psi_j \hat{\mathcal{F}}(\mathbf{u}_h^-,\mathbf{u}_h^+)\cdot\mathbf{n}\,\mathrm{d}S - \int_K \nabla\psi_j\cdot\mathcal{F}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x} = \int_K \psi_j \mathcal{S}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x},\ \ 1\le j\le D. \tag{2.37}$$

On simplex elements, the number of *degrees of freedom* D can be shown to be

$$D = \prod_{l=1}^{d}(p+l)/l.$$

To have high-order accuracy, we replace in (2.37) the integrals by consistent quadrature rules as follows

$$\int_{\partial K} \psi_j \hat{\mathcal{F}}(\mathbf{u}_h^-,\mathbf{u}_h^+)\cdot\mathbf{n}\,\mathrm{d}S \approx \sum_{\substack{e\subset\partial K,\\ e=\Gamma}} |e| \sum_{q_e=1}^{Q_e} \omega_{q_e}\,\psi_j(\mathbf{x}_{q_e})\,\hat{\mathcal{F}}(\mathbf{u}_h(\mathbf{x}_{q_e})^-,\mathbf{u}_h(\mathbf{x}_{q_e})^+)\cdot\mathbf{n}(\mathbf{x}_{q_e}), \tag{2.38}$$

$$\int_K \nabla\psi_j\cdot\mathcal{F}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x} \approx |K| \sum_{q_K=1}^{Q_K} \omega_{q_K}\,\nabla\psi_j(\mathbf{x}_{q_K})\cdot\mathcal{F}(\mathbf{u}_h(\mathbf{x}_{q_K})), \tag{2.39}$$

$$\int_K \psi_j \mathcal{S}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x} \approx |K| \sum_{q_K=1}^{Q_K} \omega_{q_K}\,\psi_j(\mathbf{x}_{q_K})\,\mathcal{S}(\mathbf{u}_h(\mathbf{x}_{q_K})). \tag{2.40}$$

Finally, by assembling all the contributions (2.37), we obtain in each element K a system of ODEs reading

$$\frac{\mathrm{d}\mathsf{u}}{\mathrm{d}t} + M^{-1}\mathsf{R}(\mathbf{u}_h) = 0, \tag{2.41}$$

where R includes the second, third and fourth integrals in (2.37), and M denotes the mass matrix

$$[M]_{jk} = \int_K \psi_j\psi_k\,\mathrm{d}\mathbf{x}, \tag{2.42}$$

and the approximation $\mathsf{u} \in \boldsymbol{\mathcal{V}}_h^p$ is evolved in time by a time integration method, e.g. Runke-Kutta methods.

Remark 20. *Instead of working with a broken space of broken polynomials of degree p within each triangle K and discontinuous across faces, we can also consider globally*

continuous piecewise polynomials of degree p and sum over all elements. In this case, there are no discontinuities at the element boundaries and no numerical fluxes are needed. Following the above described step leads the classical ***continuous Galerkin*** *scheme. This shows well the connection between continuous and discontinuous Galerkin methods.*

2.4 Residual Distribution (RD) schemes

Residual distribution (RD) schemes, also known as fluctuation splitting schemes, date back to the seminal work of Roe [263] where the first basic idea of RD schemes has been introduced, before being further extended in [264]. In [263] the author initially suggested to see the integral of the divergence of the flux of a hyperbolic balance law as a measure of the error, i.e., a fluctuation that could be possibly evolved in such way that its decomposition in signals allows to evolve the approximation towards the sought solution.

RD schemes, also known under the name fluctuation splitting, can naturally be seen as a generalization of some numerical methods (like DG or FV) methods since these schemes can be recast in this framework [7, 111]. Here, we present the framework for steady state problems. The analysis of unsteady problems requires more precautions to appropriately account for the coupling between the time derivative and the evolution operator. The interested reader can find more about RD for time-dependent problems in [260, 261]. More recent developments in this field can be found in [9, 8, 10, 11, 172].

As already mentioned, herein, we describe the RD framework for steady-state simulations. So, we consider a steady-state balance law

$$\nabla \cdot \mathcal{F}(\mathbf{u}) = \mathcal{S}(\mathbf{u}), \qquad \mathbf{x} \in \mathbf{\Omega} \subset \mathbb{R}^d \tag{2.43}$$

with boundary conditions enforced on $\partial\mathbf{\Omega}$. As in classical FE methods, for each element K, we denotes by σ a generic degree of freedom (DOF) inside one element. Therefore, we have a set of DOF $\sum_{N_\sigma}$ acting on the set of continuous polynomials $\boldsymbol{\mathcal{V}}_h^p$ of degree p. As shown before, the space $\boldsymbol{\mathcal{V}}_h^p$ is spanned by basis functions $\{\psi_\sigma\}_{\sigma\in\sum_{N_\sigma}}$. Several interpolation options are available: Lagrange and Berstein polynomials are the most commonly used. For any element K, N_σ is the number of DOF. We recall that we approximate the solution in each element using a polynomial of degree p and $\mathbf{u}_h$ denotes our numerical solution. For the continuity requirement (see Figure 2.2), we need an additional condition on the splitting of the domain $\mathbf{\Omega}$, e.g. a conformal triangulation.

A linear combination of basis functions ψ_σ will be used to describe the numerical

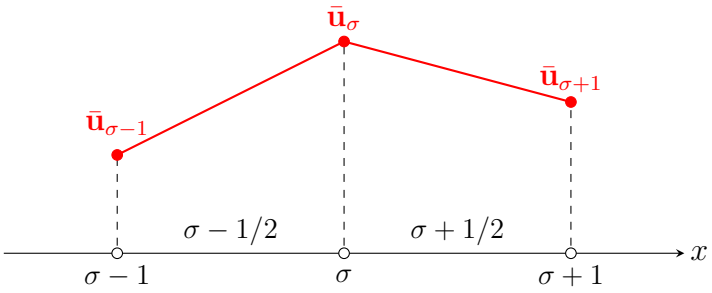


Figure 2.2: Schematic visualization of residual distribution in 1D.

solution

$$\mathbf{u}_h(\mathbf{x}) = \sum_{K \in \mathbf{\Omega}} \sum_{\sigma \in \sum_{N_\sigma}} \mathbf{u}_{h,\sigma} \psi_\sigma|_K(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathbf{\Omega} \tag{2.44}$$

where the coefficients $\mathbf{u}_{h,\sigma}$ has to be found by means of a numerical method.

The residual distribution approach can be summarized in three main steps (see Figure 2.3), described several times in the aforementioned references.

1. For any element K, define the total residual. This term is also referred to as *fluctuation* in literature and reads

$$\begin{aligned} \Phi^K &= \int_K \nabla \cdot \mathcal{F}(\mathbf{u}_h) \cdot \mathbf{n} \, \mathrm{d}\mathbf{x} - \int_K \mathcal{S}(\mathbf{u}_h) \, \mathrm{d}\mathbf{x} \\ &= \int_{\partial K} \mathcal{F}(\mathbf{u}_h) \cdot \mathbf{n} \, \mathrm{d}S - \int_K \mathcal{S}(\mathbf{u}_h) \, \mathrm{d}\mathbf{x} \end{aligned} \tag{2.45}$$

   for a conformal mesh and with globally continuous approximation.

   When performing numerical quadrature for (2.45) we obtain

$$\Phi^K = \sum_{\substack{e \subset \partial K, \\ e = \Gamma}} |e| \sum_{q_e=1}^{Q_e} \omega_{q_e} \mathcal{F}(\mathbf{u}_h(\mathbf{x}_{q_e})) \cdot \mathbf{n}_{q_e} - |K| \sum_{q_K=1}^{Q_K} \omega_{q_K} \mathcal{S}(\mathbf{u}_h(\mathbf{x}_{q_K})) \tag{2.46}$$

2. Split the total residual Φ^K into nodal residuals Φ^K_σ, for each degree of freedom $\sigma \in K$, so that the sum of all the contributions over an element K is the fluctuation term itself, meaning that

$$\Phi^K = \sum_{\sigma \in K} \Phi^K_\sigma, \qquad K \in \mathbf{\Omega}_h. \tag{2.47}$$

3. The resulting scheme is finally obtained by adding up all nodal residuals of one

degree of freedom σ from different elements K, so that

$$\sum_{K|\sigma\in K} \Phi_\sigma^K = 0, \qquad \forall \sigma \in \mathbf{\Omega}. \tag{2.48}$$

Equation (2.48) allows us to compute the coefficients $\mathbf{u}_{h,\sigma}$ in our numerical approximation (2.44).

To describe the complete scheme we have to take the boundary into account. For an element Γ on the boundary $\partial\mathbf{\Omega}_h$, we define *boundary residuals* Φ_σ^Γ the following conservation relation

$$\sum_{\sigma\in\Gamma} \Phi_\sigma^\Gamma = \int_\Gamma \left(\hat{\mathcal{F}}(\mathbf{u}_h, \mathbf{g}) \cdot \mathbf{n} - \mathcal{F}(\mathbf{u}_h) \cdot \mathbf{n}\right) \mathrm{d}S, \tag{2.49}$$

where $\mathbf{g}$ is a *Dirichlet* boundary condition enforced on the boundary $\Gamma \in \partial\mathbf{\Omega}_h$.

Now, We can write the final discretization for any DOF,

$$\sum_{K\in\mathbf{\Omega}_h|\sigma\in K} \Phi_\sigma^K(\mathbf{u}_h) + \sum_{\Gamma\in\partial\mathbf{\Omega}_h|\sigma\in\Gamma} \Phi_\sigma^\Gamma(\mathbf{u}_h) = 0. \tag{2.50}$$

The first term of (2.50) represents the contribution of the internal element whereas the second one exists if $\sigma \in \Gamma$ and describes the contribution of the boundary.

Finally, we have to define the residuals Φ_σ^K in order to specify more precisely the numerical scheme. It was shown that, by properly choosing Φ_σ^K, we can recover several schemes from the RD framework [17].

Example 21 (FE schemes in the RD framework)**.** *The following schemes are usually considered:*

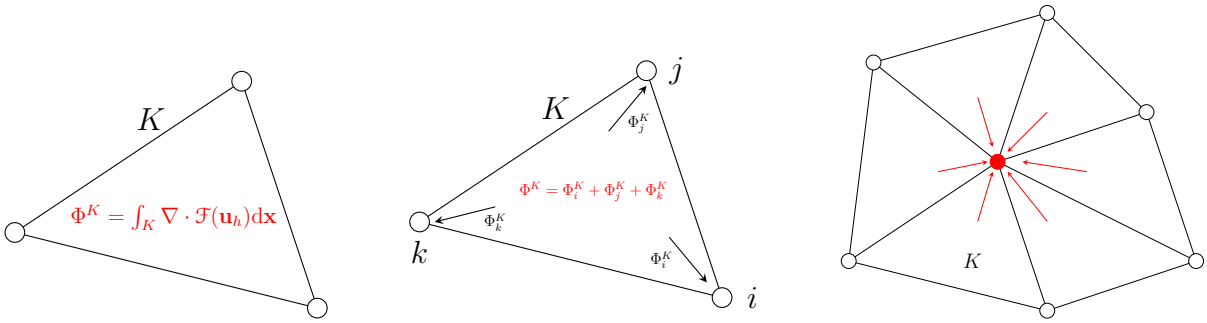
- ***Continuous Galerkin*** *(CG),*

$$\begin{aligned}\Phi_\sigma^K(\mathbf{u}_h) :&= \int_K \psi_\sigma \nabla \cdot \mathcal{F}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x} \\ &= \int_{\partial K} \psi_\sigma \mathcal{F}(\mathbf{u}_h)\cdot\mathbf{n}\,\mathrm{d}S - \int_K \nabla\psi_\sigma \cdot \mathcal{F}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x}.\end{aligned} \tag{2.51}$$

- ***Streamline Upwind Petrov Galerkin*** *(SUPG) [174],*

$$\begin{aligned}\Phi_\sigma^K(\mathbf{u}_h) :&= \int_{\partial K} \psi_\sigma \mathcal{F}(\mathbf{u}_h)\cdot\mathbf{n}\,\mathrm{d}S - \int_K \nabla\psi_\sigma \cdot \mathcal{F}(\mathbf{u}_h)\,\mathrm{d}\mathbf{x} \\ &\quad + h_K \int_K \left(\nabla_{\mathbf{u}}\mathcal{F}(\mathbf{u}_h)\cdot\nabla\psi_\sigma\right)\tau_K\left(\nabla_{\mathbf{u}}\mathcal{F}(\mathbf{u}_h)\cdot\nabla\mathbf{u}_h\right),\end{aligned} \tag{2.52}$$

with $\tau_K > 0$.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 2.3: Residual distribution algorithm for linear triangular elements (compute the total residual, split the total residual, combine the residuals).

- ***Discontinuous Galerkin*** *(DG),*

$$\Phi_\sigma^K(\mathbf{u}_h) := \int_{\partial K} \psi_\sigma \hat{\mathcal{F}}(\mathbf{u}_h^-, \mathbf{u}_h^+) \cdot \mathbf{n} \, \mathrm{d}S - \int_K \nabla \psi_\sigma \cdot \mathcal{F}(\mathbf{u}_h) \, \mathrm{d}\mathbf{x}. \tag{2.53}$$

So far, we have only rewritten classical FE schemes within the RD framework. We can also define new schemes that have no straightforward variational formulation. The schemes are referred to as *Limited Residual Distribution* (LRD) [13], and read

$$\Phi_\sigma^K(\mathbf{u}_h) = \beta_\sigma \int_{\partial K} \psi_\sigma \mathcal{F}(\mathbf{u}_h) \cdot \mathbf{n} \, \mathrm{d}S \tag{2.54}$$

where the distribution coefficients β_σ have to be defined to guarantee conservation, i.e $\sum_{\sigma \in K} \beta_\sigma = 1$, and satisfy a discrete maximum principle. It is then the choice of the coefficients β_σ that determines the property of the discrete solution. Several design criteria have been proposed and used [110, 242, 280].

- *Upwinding*: an upwind scheme distributes the fractions of the cell fluctuation only among its downstream DOF.
- *Positivity*: the positivity criterion ensures that the maximum principle holds also at the discrete level.
- *Linearity preservation*: linearity preservation is an accuracy requirement and refers to the ability of the discrete scheme to reproduce exactly a linear polynomial.

Herein, we only define the schemes employed to run the simulations presented in Chapter 4 and Chapter 5.
In particular we are only going to recall the basic implementation of linear and non-linear schemes for the steady advection equation.

- **Linear schemes**

  We describe the multidimensional-upwind and linearity-preserving LDA scheme. This second-order linear method is defined by the distribution coefficients:

$$\beta_i^{\mathrm{LDA}} = \mathcal{K}_i^+ \left(\sum_{j \in K} \mathcal{K}_j^+ \right)^{-1} \quad \in [0,1], \tag{2.55}$$

  where

$$\mathcal{K}_i^+ = \max\{0, \mathcal{K}_i\}, \quad \mathcal{K}_i^- = \min\{0, \mathcal{K}_i\} \qquad \text{such that} \quad \mathcal{K}_i = \mathcal{K}_i^+ + \mathcal{K}_i^-$$

  and $\mathcal{K}_i$ denotes the scalar $\mathcal{K}_i = \frac{1}{2}\mathcal{A}_i \cdot \mathbf{n}_i$ with $\mathbf{n}_i$ the scaled inward normal poiting at node i. The parameters $\mathcal{K}_i$ can be used as sensors to distinguish between downstream and upstream nodes. In particular, $\mathcal{K}_i > 0$ only if $\mathcal{A}$ and $\mathbf{n}_i$ are oriented towards the same direction, thus if node i is downstream. The local nodal residuals are given by

$$\Phi_i^{K,\mathrm{LDA}} = \beta_i^{\mathrm{LDA}} \Phi^K. \tag{2.56}$$

  From a similar implementation and following the same multidimensional upwinding procedure, the first-order N scheme $\Phi_i^{K,\mathrm{N}}$ can be recovered using a simpler geometrical representation of the advection coefficients.

- **Non-linear schemes**

  Non-linear schemes are needed to compute all those simulations that feature discontinuities. The approach used to run such computations is based on the local blending between a low-order (N) and a high-order (LDA) method, such that

$$\Phi_i^K = (1 - \Theta)\, \Phi_i^{K,\mathrm{LDA}} + \Theta \Phi_i^{K,\mathrm{N}}. \tag{2.57}$$

  Blending the LDA and the N scheme, using a *smoothness indicator* Θ, basically consists in adding a dissipation term to the LDA scheme.

2.5 Time integration schemes

In this section, we briefly discuss the time integration methods that we employed in this work. When working with the method of lines [198], the space and time discretizations are decoupled and treated independently. Therefore, some of the methods presented before (FV, DG, or CG) can be used to discretize the hyperbolic problem in space obtaining an

ordinary differential equation (ODE). Afterwords, in a second step, this will be solved by time integration schemes, e.g. Runge-Kutta. RD is an exception and cannot be treated this way, otherwise unexpected problems would arise (order degradation). In this case, space and time need to be discretized together. Abgrall has recently presented an approach in the RD setting [8] based on the deferred correction (DeC) methods, combined with RD schemes. Due to the very good features of the DeC methods, also in other frameworks, and their ability of reaching arbitrary high order easily, we employed them in this work and applied in the FV context. More information about the DeC and its applications can be found in [14, 217, 86, 234, 18] Finally, we also introduce the high order explicit one-step predictor-corrector ADER methods [121, 203, 65, 55, 56, 139], which recently have been compared to the DeC [294], for DG space-time formulations.

2.5.1 Runge-Kutta methods

There are many numerical techniques to solve numerically an ODE. A first ansatz was given by finite differences, where the derivative in time is replaced by differences of states in different timesteps. Backward (implicit) and forward (explicit) Euler are examples of this strategy. Another approach would be to reformulate the ODE by integrating it in time. With different quadrature formulas and approximation techniques, we can recover various *Runge-Kutta* methods (explicit and implicit ones).

We consider the following time-dependent ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t}\underline{\boldsymbol{y}} = \mathbb{L}(t, \underline{\boldsymbol{y}}), \qquad t \in [t^n, t^{n+1}], \tag{2.58}$$

$$\underline{\boldsymbol{y}}(0) = \underline{\boldsymbol{y}}_0. \tag{2.59}$$

A general (explicit or implicit) Runge-Kutta method with s stages can be represented by its *Butcher tableau* [66]:

$$\begin{array}{c|c} \tilde{c} & A \\ \hline & b \end{array}, \tag{2.60}$$

where $A \in \mathbb{R}^{s\times s}$, and $b, \tilde{c} \in \mathbb{R}^s$.
Considering (2.61), one step to compute $\underline{\boldsymbol{y}}^{n+1} = \underline{\boldsymbol{y}}(t^{n+1})$ from $\underline{\boldsymbol{y}}^n = \underline{\boldsymbol{y}}(t^n)$ with $t^{n+1} =$

$t^n + \Delta t$ can be recast as

$$\begin{aligned}\underline{\boldsymbol{y}}^i &:= \underline{\boldsymbol{y}}^0 + \Delta t \sum_{j=1}^{s} A_{i,j}\, \mathbb{L}(t^n + \tilde{c}_j \Delta t, \underline{\boldsymbol{y}}^j),\\ \underline{\boldsymbol{y}}^{n+1} &:= \underline{\boldsymbol{y}}^0 + \Delta t \sum_{i=1}^{s} b_i\, \mathbb{L}(t^n + \tilde{c}_j \Delta t, \underline{\boldsymbol{y}}^i).\end{aligned} \tag{2.61}$$

Example 22 (RK(4,4))**.** *The most famous RK scheme is the fourth order Runge-Kutta method, with four stages, defined as*

$$\begin{aligned}\underline{\boldsymbol{y}}^{(0)} &= \underline{\boldsymbol{y}}^n,\\ \underline{\boldsymbol{y}}^{(1)} &= \underline{\boldsymbol{y}}^n + \frac{1}{2}\Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(0)}),\\ \underline{\boldsymbol{y}}^{(2)} &= \underline{\boldsymbol{y}}^n + \frac{1}{2}\Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(1)}),\\ \underline{\boldsymbol{y}}^{(3)} &= \underline{\boldsymbol{y}}^n + \Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(2)}),\\ \underline{\boldsymbol{y}}^{n+1} &= \underline{\boldsymbol{y}}^n + \frac{1}{6}\Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(0)}) + \frac{1}{3}\Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(1)}) \frac{1}{3}\Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(2)}) + \frac{1}{3}\Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(3)}),\end{aligned} \tag{2.62}$$

and its corresponding butcher tableau is

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0\\ 1/2 & 1/2 & 0 & 0 & 0\\ 1/2 & 0 & 1/2 & 0 & 0\\ 1 & 0 & 0 & 1 & 0\\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}, \tag{2.63}$$

Rather than using Equation (2.61), Runge-Kutta schemes can also be expressed using the Shu-Osher formulation [277]:

$$\begin{aligned}\underline{\boldsymbol{y}}^{(0)} &:= \underline{\boldsymbol{y}}^n,\\ \underline{\boldsymbol{y}}^{(i)} &:= \sum_{k=0}^{i-1} \left(\alpha_{i,k}\, \underline{\boldsymbol{y}}^{(k)} + \Delta t\, \delta_{i,k}\, \mathbb{L}(\underline{\boldsymbol{y}}^{(k)})\right), \quad \alpha_{i,k}, \delta_{i,k} \geq 0, \quad i = 1, \ldots, s,\\ \underline{\boldsymbol{y}}^{n+1} &:= \underline{\boldsymbol{y}}^{(s)},\end{aligned} \tag{2.64}$$

where we assume that $\mathbb{L}$ depends on t only through $\underline{\boldsymbol{y}}$. Then, for consistency reasons, we have to impose that $\sum_{k=0}^{i-1} \alpha_{i,k} = 1$ and $\delta_{i,k} = 0$, whenever $\alpha_{i,k} = 0$. By recasting (2.61) as a convex combination of simple explicit Euler method (2.64), it was possible to prove the

Strong Stability Preserving (SSP) property of RK methods. The SSP property ensures that every solution of $\underline{\boldsymbol{y}}$ decreases in time, provided that this holds for the simple explicit Euler method with Δt_{euler}. It should be noticed that each intermediate step $\underline{\boldsymbol{y}}^{(i)}$ is a convex combination of forward Euler steps, with a different time-step restriction $\frac{\delta_{i,k}}{\alpha_{i,k}}\Delta t \leq \Delta t_{euler}$. The most commonly used SSPRK scheme for engineering applications is the third order method proposed by Gottlieb and Shu [159].

Example 23 (SSPRK(3,3))**.** *The third order, with three stages, SSP Runge-Kutta is defined as*

$$\begin{aligned}
\underline{\boldsymbol{y}}^{(0)} &= \underline{\boldsymbol{y}}^{n}, \\
\underline{\boldsymbol{y}}^{(1)} &= \underline{\boldsymbol{y}}^{n} + \Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(n)}), \\
\underline{\boldsymbol{y}}^{(2)} &= \frac{3}{4}\underline{\boldsymbol{y}}^{n} + \frac{1}{4}\underline{\boldsymbol{y}}^{(1)} + \frac{1}{4}\Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(1)}), \\
\underline{\boldsymbol{y}}^{n+1} &= \frac{1}{3}\underline{\boldsymbol{y}}^{n} + \frac{2}{3}\underline{\boldsymbol{y}}^{(2)} + \frac{2}{3}\Delta t\, \mathbb{L}(\underline{\boldsymbol{y}}^{(2)}),
\end{aligned} \tag{2.65}$$

and its corresponding butcher tableau is

$$\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
1/2 & 1/4 & 1/4 & 0 \\
\hline
 & 1/6 & 1/6 & 2/3
\end{array}, \tag{2.66}$$

For more information about RK and SSP-RK schemes, several works can be found in the literature [160, 159, 182, 255].

2.5.2 Deferred Correction methods

Another way to solve ordinary differential equations is the *deferred correction* (DeC) method, which was firstly introduced in [123]. Here we introduce the DeC by following the reasoning given by Abgrall in [8]. In this work we only focus on the explicit, arbitrary high order DeC method for ODEs (2.58), although further extensions in the implicit and semi-implicit framework can be found in the literature [221].

The main idea is based on the Picard-Lindelöf Theorem in the continuous setting. The theorem states the existence and uniqueness of solutions of ODEs. The classical proof makes use of the so-called *Picard iterations* to minimize the error and prove convergence.

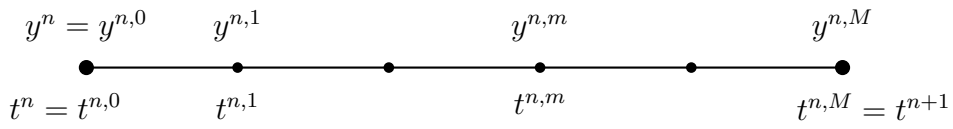


Figure 2.4: Time interval divided into sub-time steps

The foundation of DeC relies on mimicking the Picard iterations at the discrete level. The approximation error decreases with several iteration steps. Abgrall [8] introduced two operators: $\mathcal{L}^1$ and $\mathcal{L}^2$. Here, the $\mathcal{L}^1$ operator represents a low-order easy-to-solve numerical scheme, e.g. the explicit Euler method, and $\mathcal{L}^2$ is a high-order operator that can present difficulties in its practical solution, e.g. an implicit RK scheme. The DeC method can be written as a combination of these two operators. In this setting, we have that the solution of $\mathcal{L}^1(\underline{\boldsymbol{y}}) = 0$ is easy to obtain, whereas $\mathcal{L}^2(\underline{\boldsymbol{y}}) = 0$ might be more difficult to solve. The DeC procedure combines the two operators exploiting the accuracy of $\mathcal{L}^2$ and the simplicity of $\mathcal{L}^1$ obtaining an explicit (and easy-to-solve) high order method.

Consider a time step $[t^n, t^{n+1}]$ and let us subdivide it into M sub-time steps $\{[t^{n,m-1}, t^{n,m}]\}_{m=1}^{M}$ where the boundary points coincide with the extrema of the time step, i.e. $t^n = t^{n,0}$ and $t^{n,M} = t^{n+1}$. We consider a discretization of the variables in the sub-time nodes $t^{n,m}$ denoted as $y^{n,m}$ as explained in Figure 2.4. We define the $\mathcal{L}^2$ operator as

$$\mathcal{L}^2(\underline{\boldsymbol{y}}^{n,0}, \dots, \underline{\boldsymbol{y}}^{n,M}) := \begin{cases} \underline{\boldsymbol{y}}^{n,M} - \underline{\boldsymbol{y}}^{n,0} - \displaystyle\int_{t^{n,0}}^{t^{n,M}} \mathfrak{I}_M(\mathbb{L}(\underline{\boldsymbol{y}}^0), \dots, \mathbb{L}(\underline{\boldsymbol{y}}^M))\mathrm{d}s \\ \vdots \\ \underline{\boldsymbol{y}}^{n,1} - \underline{\boldsymbol{y}}^{n,0} - \displaystyle\int_{t^{n,0}}^{t^{n,1}} \mathfrak{I}_M(\mathbb{L}(\underline{\boldsymbol{y}}^0), \dots, \mathbb{L}(\underline{\boldsymbol{y}}^M))\mathrm{d}s \end{cases} . \tag{2.67}$$

where $\mathfrak{I}_M$ denotes an interpolation polynomial of degree M evaluated at the points $\{t^r\}_{r=0}^{M}$. We apply Lagrange polynomials $\{\varphi_r\}_{r=0}^{M}$. They satisfy $\varphi_r(t^m) = \delta_{r,m}$ and $\sum_{r=0}^{M} \varphi_r(s) = 1$ for any $s \in [t^0, t^M]$. Using these properties and using a high order quadrature rule in the same points, we can actually compute the integral of the interpolants $\{t^m\}_{m=0}^{M}$, with weights

$$\theta_r^m := \frac{1}{\Delta t} \int_{t^0}^{t^m} \varphi_r(s)\mathrm{d}s,$$

resulting in

$$\mathcal{L}^2(\underline{\boldsymbol{y}}^{n,0},\dots,\underline{\boldsymbol{y}}^{n,M}) \approx \begin{cases} \underline{\boldsymbol{y}}^{n,M} - \underline{\boldsymbol{y}}^{n,0} - \Delta t \sum_{r=0}^{M} \theta_r^M \mathbb{L}(\underline{\boldsymbol{y}}^{n,r}) \\ \vdots \\ \underline{\boldsymbol{y}}^{n,1} - \underline{\boldsymbol{y}}^{n,0} - \Delta t \sum_{r=0}^{M} \theta_r^1 \mathbb{L}(\underline{\boldsymbol{y}}^{n,r}) \end{cases} \tag{2.68}$$

The $\mathcal{L}^2$ operator represents a high order numerical scheme if set equal to zero, i.e. $\mathcal{L}^2(\underline{\boldsymbol{y}}^0,\dots,\underline{\boldsymbol{y}}^M) = 0$. The order depends on the distribution of the subtimesteps, for instance, with M equispaced subtimesteps one obtains $(M+1)$th order, while with M Gauss–Lobatto quadrature subtimesteps one has $(2M)$th order. Unfortunately, the resulting scheme is implicit and, further, the terms $\mathbb{L}$ may be nonlinear. As a consequence, the only $\mathcal{L}^2$ formulation is not explicit and more efforts have to be made to solve it. For this purpose, a simplification of the $\mathcal{L}^2$ operator is introduced. The $\mathcal{L}^1$ operator is given by the forward Euler discretization for each state $\underline{\boldsymbol{y}}^m$ in the time interval, i.e.

$$\mathcal{L}^1(\underline{\boldsymbol{y}}^0,\dots,\underline{\boldsymbol{y}}^M) := \begin{cases} \underline{\boldsymbol{y}}^M - \underline{\boldsymbol{y}}^0 - \beta^M \Delta t \mathbb{L}(\underline{\boldsymbol{y}}^0) \\ \vdots \\ \underline{\boldsymbol{y}}^1 - \underline{\boldsymbol{y}}^0 - \beta^1 \Delta t \mathbb{L}(\underline{\boldsymbol{y}}^0) \end{cases} \tag{2.69}$$

with coefficients $\beta^m := \frac{t^m - t^0}{t^M - t^0}$.
To simplify the notation and to describe DeC, we introduce the matrix of states for the variable $\underline{\boldsymbol{y}}$ at all subtimesteps.

$$\bar{\underline{\boldsymbol{y}}} := (\underline{\boldsymbol{y}}^0,\dots,\underline{\boldsymbol{y}}^M) \in \mathbb{R}^{M\times S}, \text{ such that} \tag{2.70}$$

$$\mathcal{L}^1(\bar{\underline{\boldsymbol{y}}}) := \mathcal{L}^1(\underline{\boldsymbol{y}}^0,\dots,\underline{\boldsymbol{y}}^M) \text{ and } \mathcal{L}^2(\bar{\underline{\boldsymbol{y}}}) := \mathcal{L}^2(\underline{\boldsymbol{y}}^0,\dots,\underline{\boldsymbol{y}}^M). \tag{2.71}$$

So far, the DeC algorithm uses a combination of the $\mathcal{L}^1$ and $\mathcal{L}^2$ operators to provide an iterative procedure. The aim is to recursively approximate $\bar{\underline{\boldsymbol{y}}}^*$, the numerical solution of the $\mathcal{L}^2(\bar{\underline{\boldsymbol{y}}}^*) = 0$ scheme, similarly to the Picard iterations in the continuous setting. The successive states of the iteration process will be denoted by the superscript (k), where k is the iteration index, e.g. $\bar{\underline{\boldsymbol{y}}}^{(k)} \in \mathbb{R}^{M\times S}$. The total number of iterations (also called correction steps in the following) is denoted by k^{DeC}. To describe the procedure, we have to refer to both the m-th subtimestep and the k-th iteration of the DeC algorithm. We will indicate the variable by $\underline{\boldsymbol{y}}^{m,(k)} \in \mathbb{R}^S$. Finally, the DeC method can be written as

DeC Algorithm

$$\begin{aligned}
&\underline{\boldsymbol{y}}^{0,(k)} := \underline{\boldsymbol{y}}(t^n), \quad k = 0, \dots, k^{DeC}, \\
&\underline{\boldsymbol{y}}^{m,(0)} := \underline{\boldsymbol{y}}(t^n), \quad m = 1, \dots, M, \\
&\mathcal{L}^1(\bar{\underline{\boldsymbol{y}}}^{(k)}) = \mathcal{L}^1(\bar{\underline{\boldsymbol{y}}}^{(k-1)}) - \mathcal{L}^2(\bar{\underline{\boldsymbol{y}}}^{(k-1)}) \text{ with } k = 1, \dots, k^{DeC},
\end{aligned} \tag{2.72}$$

where k^{DeC} is the number of iterations that we want to compute.

Algorithm 1 DeC step

Require: $\underline{\boldsymbol{y}}^n$, Δt, $\mathbb{L}$ evolution functions

for $k = 0$ **to** k^{DeC} **do**
  $\underline{\boldsymbol{y}}^{0,(k)} \leftarrow \underline{\boldsymbol{y}}^n$
end for
for $m = 1$ **to** M **do**
  $\underline{\boldsymbol{y}}^{m,(0)} \leftarrow \underline{\boldsymbol{y}}^n$
end for
for $k = 1$ **to** k^{DeC} **do**
  for $m = 1$ **to** M **do**
    Compute $\underline{\boldsymbol{y}}^{m,(k)}$ solving $\mathcal{L}^{1,m}(\bar{\underline{\boldsymbol{y}}}^{(k)}) = \mathcal{L}^{1,m}(\bar{\underline{\boldsymbol{y}}}^{(k-1)}) - \mathcal{L}^{2,m}(\bar{\underline{\boldsymbol{y}}}^{(k)})$
  end for
end for
$\underline{\boldsymbol{y}}^{n+1} \leftarrow \underline{\boldsymbol{y}}^{M,(k^{DeC})}$
return $\underline{\boldsymbol{y}}^{n+1}$

Using the procedure (2.72), we need, in particular, as many iterations as the desired order of accuracy p, i.e. $k^{DeC} = p$. This means that we choose the number of subtimesteps in a way that the order of the $\mathcal{L}^2$ operator is itself equal to p. In practice, for each correction and each subtimestep, $\mathcal{L}^{1,m}(\underline{\boldsymbol{y}}^{(k)}) = \mathcal{L}^{1,m}(\underline{\boldsymbol{y}}^{(k-1)}) - \mathcal{L}^{2,m}(\underline{\boldsymbol{y}}^{(k)})$ reduces to solve

$$\underline{\boldsymbol{y}}_\alpha^{m,(k)} - \underline{\boldsymbol{y}}_\alpha^0 - \Delta t \sum_{r=0}^{M} \theta_r^m \mathbb{L}_\alpha(\underline{\boldsymbol{y}}^{r,(k-1)}) = 0, \qquad \forall \alpha = 1, \dots, I. \tag{2.73}$$

Hence, the DeC algorithm can be seen as Algorithm 1 where (2.73) gives the update formula. For more information and properties of the DeC approach, we refer to [14, 294] and references therein.

Notice that, in every step, we solve the equations for the unknown variables $\bar{\underline{\boldsymbol{y}}}^{(k)}$ which appears only in the $\mathcal{L}^1$ formulation. The operator that can then be easily inverted. Conversely, $\mathcal{L}^2$ is only applied to already computed predictions of the solution $\bar{\underline{\boldsymbol{y}}}^{(k-1)}$. Therefore, the scheme (2.73) is completely explicit and of arbitrary high-order as stated in [8]. The following theorem proves the accuracy.

Theorem 24 (Accuracy of DeC)**.** *Let $\mathcal{L}^1_\Delta$ and $\mathcal{L}^2_\Delta$ be two operators, which depend on the*

discretization scale Δ*, such that*

- $\mathcal{L}^1_\Delta$ *is coercive with respect a norm, i.e.* $\exists \alpha_1 > 0$ *independent of* Δ*, such that for any* $\underline{\mathbf{a}}, \underline{\mathbf{b}}$*, we have*

$$\|\mathcal{L}^1_\Delta(\underline{\mathbf{a}}) - \mathcal{L}^1_\Delta(\underline{\mathbf{b}})\| \geq \alpha_1 \|\underline{\mathbf{a}} - \underline{\mathbf{b}}\|, \tag{2.74}$$

- $\mathcal{L}^1_\Delta - \mathcal{L}^2_\Delta$ *is Lipschitz with constant* $\alpha_2 > 0$ *uniformly with respect to* Δ*, i.e. for any* $\underline{\mathbf{a}}, \underline{\mathbf{b}}$*,*

$$\|(\mathcal{L}^1_\Delta(\underline{\mathbf{a}}) - \mathcal{L}^2_\Delta(\underline{\mathbf{a}})) - (\mathcal{L}^1_\Delta(\underline{\mathbf{b}}) - \mathcal{L}^2_\Delta(\underline{\mathbf{b}}))\| \leq \alpha_2 \Delta \|\underline{\mathbf{a}} - \underline{\mathbf{b}}\|, \tag{2.75}$$

We also assume that there exists a unique $\underline{\mathbf{a}}^*_\Delta$ *such that* $\mathcal{L}^2(\underline{\mathbf{a}}^*_\Delta) = 0$*. Then, if* $\tilde{\Delta} := \frac{\alpha_2}{\alpha_1}\Delta < 1$*, the DeC is converging to* $\underline{\mathbf{a}}^*_\Delta$ *and after* k *iterations the error* $\|\underline{\mathbf{a}}^{(k)} - \underline{\mathbf{a}}^*_\Delta\|$ *is smaller than* $\tilde{\Delta}^k \|\underline{\mathbf{a}}^{(0)} - \underline{\mathbf{a}}^*_\Delta\|$*.*

Proof. To simplify the notation, we drop the dependency on Δ. We start proving the accuracy of DeC, $\mathcal{L}^1(\underline{\mathbf{a}}^{(k)}) = \mathcal{L}^1(\underline{\mathbf{a}}^{(k-1)}) - \mathcal{L}^2(\underline{\mathbf{a}}^{(k-1)})$, stating that $\mathcal{L}^1(\underline{\mathbf{a}}^*) = \mathcal{L}^1(\underline{\mathbf{a}}^*) - \mathcal{L}^2(\underline{\mathbf{a}}^*)$, such that

$$\begin{aligned}
\mathcal{L}^1(\underline{\mathbf{a}}^{(k)}) - \mathcal{L}^1(\underline{\mathbf{a}}^*) &= \left(\mathcal{L}^1(\underline{\mathbf{a}}^{(k-1)}) - \mathcal{L}^1(\underline{\mathbf{a}}^*)\right) - \left(\mathcal{L}^2(\underline{\mathbf{a}}^{(k-1)}) - \mathcal{L}^2(\underline{\mathbf{a}}^*)\right) \\
&= \left(\mathcal{L}^1(\underline{\mathbf{a}}^{(k-1)}) - \mathcal{L}^2(\underline{\mathbf{a}}^{(k-1)})\right) - \left(\mathcal{L}^1(\underline{\mathbf{a}}^*) - \mathcal{L}^2(\underline{\mathbf{a}}^*)\right)
\end{aligned}$$

and

$$\begin{aligned}
\alpha_1 \|\underline{\mathbf{a}}^{(k)} - \underline{\mathbf{a}}^*)\| &\leq \|\mathcal{L}^1(\underline{\mathbf{a}}^{(k)}) - \mathcal{L}^1(\underline{\mathbf{a}}^*)\| \\
&= \| \left(\mathcal{L}^1(\underline{\mathbf{a}}^{(k-1)}) - \mathcal{L}^2(\underline{\mathbf{a}}^{(k-1)})\right) - \left(\mathcal{L}^1(\underline{\mathbf{a}}^*) - \mathcal{L}^2(\underline{\mathbf{a}}^*)\right) \| \\
&\leq \alpha_2 \Delta \|\underline{\mathbf{a}}^{(k-1)} - \underline{\mathbf{a}}^*\| \leq (C\Delta)^k \|\underline{\mathbf{a}}^{(0)} - \underline{\mathbf{a}}^*\|
\end{aligned}$$

After each correction, the error decreases of a factor Δ. After k^{DeC} iterations, the scheme behaves like Δ^k. □

2.5.3 Explicit one-step *predictor-corrector* ADER methods

Following the idea of solving a Generalized Riemann Problem [74] (GRP), the ADER (Arbitrary high order DErivative Riemann problem) approach has been proposed for the linear advection equation with constant coefficients by Toro and collaborators [288]. The first step of the methodology involves piecewise polynomial data reconstruction, where a non-linear ENO reconstruction is applied in order to avoid spurious oscillations of the numerical solution. Then, a GRP is defined at each cell interface. The ADER approach obtains the high order time derivatives of the GRP solution at the cell interface via the

Cauchy-Kovalevskaya procedure, which replaces time derivatives by spatial derivatives using repeated differentiation of the differential form of the PDE. The spatial derivatives, which may also jump at the interface, are defined via the solution of linearized Riemann problems for the derivatives, where linearization is carried out about the Godunov state obtained from the classical Riemann problem between the boundary extrapolated values at the interface. An important step forward in the development of more general ADER schemes was achieved in Dumbser and collaborators [121], where a new class of ADER-FV methods was introduced. The main contribution of this paper consists in the introduction of a new element-local space-time DG predictor, which allows at the same time the treatment of stiff source terms [122], as well as the replacement of the cumbersome Cauchy-Kovalevskaya procedure.

In this section, we briefly recall the *explicit one-step predictor-corrector ADER* approach. The temporal domain is as usual discretized in temporal slabs $[t^n, t^{n+1}]$. Since the ADER method cannot be used in the MOL framework, we introduce it here coupled with a high order piecewise polynomial representation, e.g. discontinuous Galerkin. This framework is often referred to as ADER-DG and its two-dimensional version is the one used for the contribution in Chapter 3.

The *predictor* step consists in a local solution of Equation (1.5) inside each space-time element $C_i^n = K \times [t^n, t^{n+1}] \in \mathbb{R}^{d+1}$. Each space-time element C_i^n is defined by the three-dimensional geometry closed by element K at time t^n and the same element at time t^{n+1}, given that the mesh is not moving. This step is called *local* because it is obtained by only considering the space-time element C_i^n with initial data $\mathbf{u}_h^n$, without taking into account any interaction with its neighbours. It provides, for each space-time control volume C_i^n, a polynomial data representation $\mathbf{q}_h^n$ of high order, both in space and time, which is going to be used as a predictor solution to evaluate the numerical fluxes and sources when integrating the PDE in the final corrector step of the ADER scheme.

In the *corrector* step, the weak form of Equation (1.5) are integrated over the space-time element C_i^n by using the predictor solution $\mathbf{q}_h^n$, to return the final solution $\mathbf{u}_h^{n+1}$, by also taking into account of the coupling with neighbours through the numerical flux computations across ∂C_i^n. This ensures high order accuracy both in space and time. The scheme is by construction conservative for it takes into account all the flux contributions over ∂C_i^n.

Predictor step: high order in time

A predictor solution is computed, within element C_i^n, by means of high order piecewise space-time polynomials $\mathbf{q}_h^n(\mathbf{x},t)$ of degree p that read

$$\mathbf{q}_h^n(\mathbf{x},t) := \sum_{\ell=0}^{\mathcal{Q}-1} \theta_\ell(\mathbf{x},t)\mathbf{q}_\ell^n, \qquad (\mathbf{x},t) \in C_i^n, \qquad \mathcal{Q} = \bar{\mathcal{L}}(p,d+1), \tag{2.76}$$

where $\bar{\mathcal{L}}(p,d) = \frac{1}{d!}\prod_{m=1}^{d}(p+m)$ being the total number of expansion coefficients, and $\theta_\ell(\mathbf{x},t)$ being the modal space-time basis of the polynomials of degree p in $d+1$ dimensions (d space dimensions plus time) which read

$$\theta_\ell(x,y,t)_{|C_i^n} = \frac{(x-x_{bi}^n)^{p_\ell}}{p_\ell! h_i^{p_\ell}} \frac{(y-y_{bi}^n)^{q_\ell}}{q_\ell! h_i^{q_\ell}} \frac{(t-t_{bi}^n)^{r_\ell}}{r_\ell! h_i^{r_\ell}}, \tag{2.77}$$

$$\ell = 0,\ldots,\mathcal{L}(N,d+1), \quad 0 \le p_\ell + q_\ell + r_\ell \le N. \tag{2.78}$$

The predictor $\mathbf{q}_h^n$ is computed by means of an iterative procedure that looks for the polynomial satisfying a weak form of Equation (1.5) obtained for any C_i^n. In order to do so, we multiply Equation (1.5) by a test function θ_k, integrate over C_i^n and write it down with respect to $\mathbf{q}_h^n$:

$$\begin{aligned}\int_{t^n}^{t^{n+1}} \int_K \theta_k(\mathbf{x},t)\frac{\partial \mathbf{q}_h^n}{\partial t}\,\mathrm{d}\mathbf{x}\mathrm{d}t + \int_{t^n}^{t^{n+1}} \int_K \theta_k(\mathbf{x},t)\nabla\cdot\mathcal{F}(\mathbf{q}_h^n)\,\mathrm{d}\mathbf{x}\mathrm{d}t = \\ \int_{t^n}^{t^{n+1}} \int_K \theta_k(\mathbf{x},t)\mathcal{S}(\mathbf{q}_h^n)\,\mathrm{d}\mathbf{x}\mathrm{d}t,\end{aligned} \tag{2.79}$$

where the first term is integrated by parts in time

$$\begin{aligned}&\int_K \theta_k(\mathbf{x},t^{n+1})\mathbf{q}_h^n(\mathbf{x},t^{n+1})\mathrm{d}\mathbf{x} - \int_{K_i} \theta_k(\mathbf{x},t^n)\mathbf{u}_h^n(\mathbf{x},t^n)\mathrm{d}\mathbf{x} - \\ &\int_{t^n}^{t^{n+1}} \int_K \frac{\partial \theta_k}{\partial t}(\mathbf{x},t)\mathbf{q}_h^n(\mathbf{x},t)\mathrm{d}\mathbf{x}\mathrm{d}t + \int_{t^n}^{t^{n+1}} \int_K \theta_k(\mathbf{x},t)\nabla\cdot\mathcal{F}(\mathbf{q}_h^n)\mathrm{d}\mathbf{x}\mathrm{d}t = \\ &\int_{t^n}^{t^{n+1}} \int_K \theta_k(\mathbf{x},t)\mathcal{S}(\mathbf{q}_h^n)\,\mathrm{d}\mathbf{x}\mathrm{d}t,\end{aligned} \tag{2.80}$$

where $\mathbf{u}_h^n$ is the known initial condition at time t^n.
Equation (2.80) is made up by volume integrals to be calculated within C_i^n without surface integrals, meaning that there is no need of any communication with neighbouring control volumes. The predictor solution $\mathbf{q}_h^n$ can be finally calculated via a simple discrete Picard iteration for each space-time element.

Corrector step: one step fully discrete ADER-DG method

The last step is recovered starting from the weak formulation of the governing equations (1.5), given that the test function ψ_k coincides with the basis functions ψ_ℓ

$$\int_{t^n}^{t^{n+1}} \int_K \psi_j \left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathcal{F}(\mathbf{u}) \right) \mathrm{d}\mathbf{x}\mathrm{d}t = \int_{t^n}^{t^{n+1}} \int_K \psi_j \mathcal{S}(\mathbf{u}) \, \mathrm{d}\mathbf{x}\mathrm{d}t, \tag{2.81}$$

When substituing $\mathbf{u}$ with Equation (2.36) at time t^n and t^{n+1}, the following formula can be easily obtained with few manipolations

$$\begin{aligned} &\left(\int_K \psi_j \psi_i \mathrm{d}\mathbf{x} \right) \left(\mathbf{u}_i^{n+1} - \mathbf{u}_i^n \right) + \int_{t^n}^{t^{n+1}} \int_{\partial K} \psi_j \hat{\mathcal{F}}(\mathbf{q}_h^+, \mathbf{q}_h^-) \cdot \mathbf{n} \, \mathrm{d}S\mathrm{d}t - \\ &\int_{t^n}^{t^{n+1}} \int_K \nabla \psi_j \cdot \mathcal{F}(\mathbf{q}_h) \, \mathrm{d}\mathbf{x}\mathrm{d}t = \int_{t^n}^{t^{n+1}} \int_K \psi_j \mathcal{S}(\mathbf{q}_h^n) \, \mathrm{d}\mathbf{x}\mathrm{d}t. \end{aligned} \tag{2.82}$$

It can be noticed that the use of the predictor solution $\mathbf{q}_h$ allows to compute the integrals in Equation (2.82) with high order of accuracy in both space and time.

Part II

High Order Treatment of Curved Boundaries and Discontinuities on Unstructured Linear Meshes

Chapter 3

High Order Polynomial Correction for Boundary Conditions

In this chapter, a simple but effective high order polynomial correction has been developed and implemented to run compressible flow simulations characterized by curved domains discretized using unstructured linear meshes. When working with high order methods, such as discontinuous Galerkin (DG), it is well known that accuracy may be dramatically affected when the boundary domain is curved. For this reason, a special treatment of the boundary conditions should be implemented to preserve the designed order of accuracy. The formulation given by the Shifted Boundary Method [206, 207] (SBM) allows it to be fast to implement in any finite element framework and easy to extend to three-dimensional configurations without any problem. The SBM has been introduced to cope with embedded boundary problems and consists in retaining the designed order of accuracy of the discretization method using modified boundary conditions based on truncated Taylor series expansions. Herein we exploit this capability to properly enforce weak boundary conditions when the physical boundary is not fitted by high order meshes. The SBM, which usually would require cumbersome computations of high order partial derivatives, has been replaced with a straightforward correction, which consists in an off-element evaluation of the solution, that takes into account all the high derivative terms in both 2D and 3D for polynomials of arbitrary order. This technique allows high order convergence rates without the need of generating high order curvilinear meshes to approximate the curved boundaries of the domain. We implemented our consistent high order boundary conditions within a classical DG framework (see Section 2.3 for further details), coupled with Runge-Kutta methods (see Section 2.5.1) for steady flows, and ADER schemes (see Section 2.5.3) for unsteady flows. Several validation tests are presented to prove the convergence properties of the method in 2D and 3D.

3.1 Boundary conditions on fully conformal meshes

When the boundary ∂K of element K belongs to $\partial\Omega_h$, the normal flux function $\mathcal{F}(\mathbf{u}_h)\cdot\mathbf{n}$ must account for the appropriate boundary conditions. The flux consistent with such conditions will be denoted by $\mathcal{F}^{bc}_{\mathbf{n}}$. In this work, the boundary flux function $\mathcal{F}^{bc}_{\mathbf{n}}$ is obtained by defining a ghost state $\mathbf{u}^{bc}$, and introducing a *numerical flux* $\mathcal{F}^{bc}_{\mathbf{n}} = \hat{\mathcal{F}}(\mathbf{u}_h^-, \mathbf{u}^{bc})$ defined by some approximate Riemann solver (2.6) based on the internal state $\mathbf{u}_h^-$, and on $\mathbf{u}^{bc}$. Depending on the condition to be enforced, different definitions of the ghost state are used:

- for far-field, which can be seen as a Dirichlet-type BC enforced weakly through fluxes, all the components of $\mathbf{u}^{bc}$ are set to prescribed values;

- at inflow/outflow boundaries $\mathbf{u}^{bc}$ is obtained by imposing the Riemann invariants associated to characteristics entering the domain values obtained from prescribed reference values of density, pressure, and Mach number;

- for slip walls we wish to set

$$\mathbf{v}\cdot\mathbf{n} = w. \tag{3.1}$$

  In this case $\mathbf{u}^{bc}$ has the same density, internal energy, and tangential velocity of $\mathbf{u}_h^-$ and the opposite normal relative velocity component $\mathbf{v}\cdot\mathbf{n}-w$. For this condition, an alternative way consists in defining directly the boundary flux function by setting

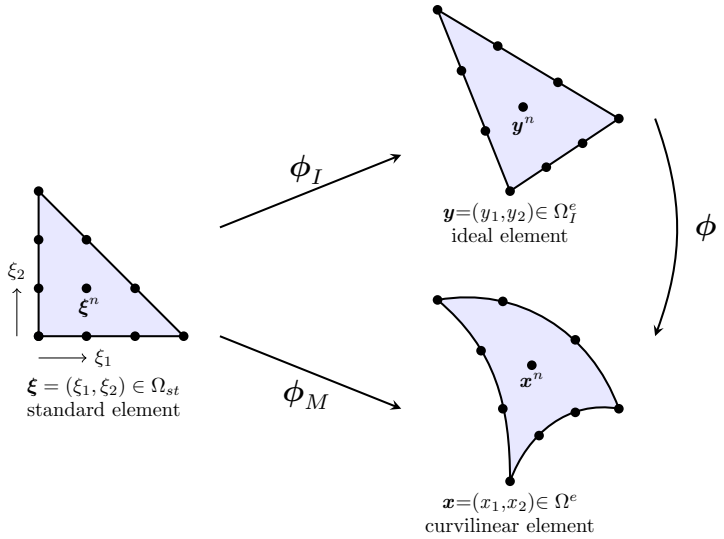


Figure 3.1: Standard reference element, straight-sided ideal element and curvilinear element

$\mathbf{v}\cdot\mathbf{n}=w$, leading to

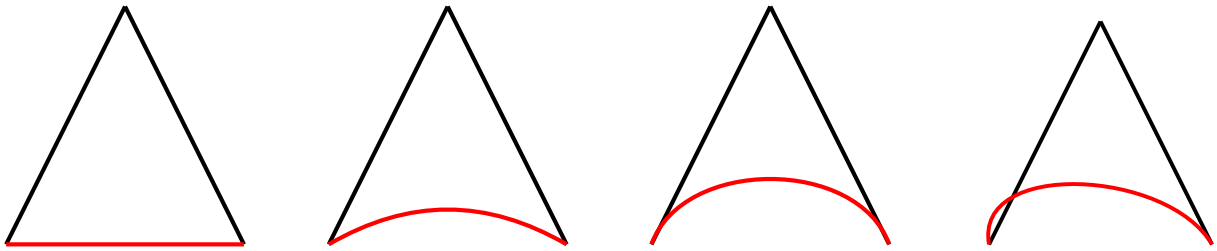
$$\mathcal{F}_{\mathbf{n}}^{bc}=\mathcal{F}_{\mathbf{n}}(\mathbf{u}^{bc})=\begin{pmatrix}\rho w & \rho\mathbf{u}\,w+p\,\mathbf{n} & \rho H\,w\end{pmatrix}. \tag{3.2}$$

When doing so, the value of the total enthalpy should be consistently modified as

$$H=\frac{\gamma}{\gamma-1}\frac{p}{\rho}+\frac{1}{2}\left(w^2+v_t^2\right), \tag{3.3}$$

with $v_t=\mathbf{v}\cdot\mathbf{t}$ the tangent velocity. For static walls, this reduces to $w=0$ and $\mathcal{F}_{\mathbf{n}}^{bc}=\begin{pmatrix}0 & p\,\mathbf{n} & 0\end{pmatrix}$.

For high order methods, one of the key aspects to achieve a genuinely high order of accuracy is to be able to simultaneously control the error on geometry and flow variables. To this end, a curved high order approximation of the boundary domain is required, which entails the use of some iso-parametric approximation of the boundary, and the generation of a valid curved volume mesh [298]. Curvilinear grids represent geometric boundaries with far superior accuracy, and with larger elements, than classical simplicial meshes. We point out in Figure 3.1 the map of a standard reference element Ω_{st} onto the straight-sided element Ω_I^e through the mapping $\boldsymbol{\phi}_I:\Omega_{st}\to\Omega_I^e$ and onto the curvilinear element $\boldsymbol{\phi}_M:\Omega_{st}\to\Omega^e$. The deformation mapping $\boldsymbol{\phi}:\Omega_I^e\to\Omega^e$ is defined through the composition $\boldsymbol{\phi}=\boldsymbol{\phi}_M\circ\boldsymbol{\phi}_I^{-1}$. Several approaches exist to obtain valid high order meshes, either based on curving existing linear meshes [114, 205, 267, 136, 224], or on some optimization or variational method [142, 290, 291]. Despite the recent progress, while generating linear meshes for complex geometries has reached a very high level of



(a) Valid (b) Valid (c) Invalid (d) Invalid

Figure 3.2: Examples of valid and invalid elements

maturity, robust curved mesh generation remains a relatively complex issue. Indeed, the curving process can create elements which self intersect or have near tangent vertices. Figure 3.2 illustrates these problems. Figure 3.2 shows (a) a valid linear element, (b) a valid high order element, (c) an invalid high order element due to near tangent vertices and (d) an invalid element due to self intersection. The black edges of the elements can be considered to be interior mesh entities and are therefore not curved. For simple cases, the best way to fix these problems is that of curving the other two edges of the triangle but usually there is no guidance on how to curve interior entities. The difficult task of high order meshing is therefore to obtain a deformation such that the resulting curvilinear mesh is geometrically conforming with the boundary, its interior is curved to balance the boundary deformation and produce a valid mesh.

In this work we relied to this end on the capabilities of `gmsh`[1] [143] to obtain simplicial and high order meshes for the simple curved geometries studied.

3.2 High order boundary conditions based on the Shifted Boundary Method

Herein, we aim at side stepping the need of generating curved meshes, and work with conformal linear ones. The idea is to use a simplified formulation of the Shifted Boundary Method (SBM) , originally introduced to handle non-conformal meshes within second order of accuracy for elliptic, parabolic and hyperbolic problems [206, 207, 279], to compensate for the geometrical error and retain the high order of accuracy. Let Ω_h be a linear conformal mesh discretizing the physical domain Ω, and $\tilde{\Gamma} := \partial\Omega_h$ the linear approximation of the curved boundary $\Gamma = \partial\Omega$. In the following, we will refer to $\tilde{\Gamma}$ as to the surrogate boundary. For any point on $\tilde{\Gamma}$ we assume to be able to define a map to a unique

[1] `https://gmsh.info/`

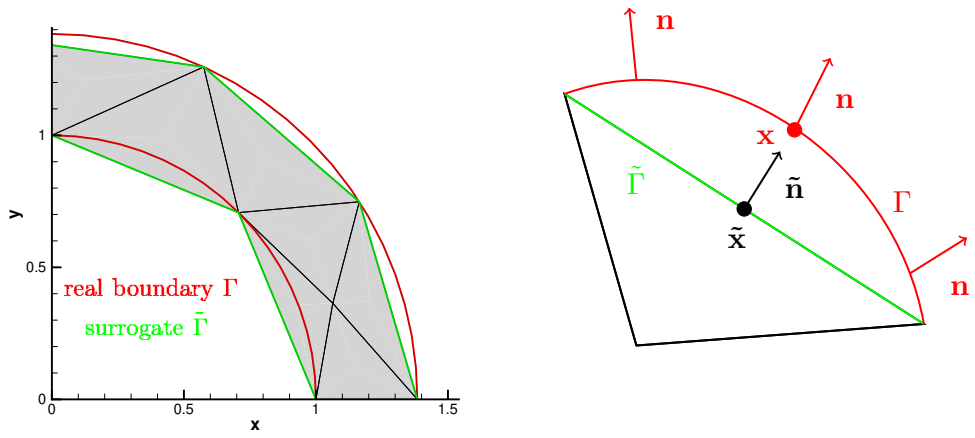


Figure 3.3: The SBM: the surrogate and actual boundaries with correspondent normals.

point of the true boundary Γ:

$$\mathbf{M} : \tilde{\Gamma} \rightarrow \Gamma$$
$$\tilde{\mathbf{x}} \rightarrow \mathbf{x}$$

The map $\mathbf{M}$ can be built in several ways, for example using a closest point projection, or using level sets, or equivalently using distances along directions normals to the true boundary Γ, as shown in Fig. 3.3. Since the gap between $\tilde{\Gamma}$ and Γ is going to be of crucial importance, in terms of accuracy of the solution, the map $\mathbf{M}$ will be characterized by a distance vector function:

$$\mathbf{d}(\tilde{\mathbf{x}}) = \mathbf{x} - \tilde{\mathbf{x}} = [\mathbf{M} - \mathbf{I}](\tilde{\mathbf{x}}). \tag{3.4}$$

If $\mathbf{M}$ is built using distances along normals to Γ, the vector $\mathbf{d}(\tilde{\mathbf{x}})$ is parallel to $\mathbf{n}$, the normal to Γ in $\mathbf{x} = \mathbf{M}(\tilde{\mathbf{x}})$.

Following the shifted boundary approach, we now modify the boundary conditions to retain the appropriate consistency order. The basic method is best described for Dirichlet conditions.

For instance, let ρ_D be the prescribed value of the density (similar expressions can be written for all variables). The main idea is that a smooth exact solution of the problem

will verify the estimate

$$\rho_D(\tilde{\mathbf{x}}+\mathbf{d}) = \rho(\tilde{\mathbf{x}}) + \|\mathbf{d}\| \sum_{j=1}^{d} n_j \, \partial_{x_j}\rho(\tilde{\mathbf{x}}) + \|\mathbf{d}\|^2 \sum_{j=1}^{d}\sum_{k=1}^{d} \frac{1}{2!} \, n_j n_k \, \partial^2_{x_j x_k}\rho(\tilde{\mathbf{x}}) \qquad (3.5)$$
$$+ \|\mathbf{d}\|^3 \sum_{j=1}^{d}\sum_{k=1}^{d}\sum_{\ell=1}^{d} \frac{1}{3!} \, n_j n_k n_\ell \, \partial^3_{x_j x_k x_\ell}\rho(\tilde{\mathbf{x}}) + \dots$$

The idea is thus to modify the boundary condition on $\tilde{\Gamma}$ to account for all the corrective terms, which boils down to use a modified prescribed value which, for different accuracy orders, is given by

$$\rho_{SBM}(\tilde{\mathbf{x}}) = \rho_D(\tilde{\mathbf{x}}+\mathbf{d}) - \|\mathbf{d}\| \sum_{j=1}^{d} n_j \, \partial_{x_j}\rho(\tilde{\mathbf{x}}) \qquad \text{second order} \qquad (3.6)$$
$$- \|\mathbf{d}\|^2 \sum_{j=1}^{d}\sum_{k=1}^{d} \frac{1}{2!} \, n_j n_k \, \partial^2_{x_j x_k}\rho(\tilde{\mathbf{x}}) \qquad \text{third order}$$
$$- \|\mathbf{d}\|^3 \sum_{j=1}^{d}\sum_{k=1}^{d}\sum_{\ell=1}^{d} \frac{1}{3!} \, n_j n_k n_\ell \, \partial^3_{x_j x_k x_\ell}\rho(\tilde{\mathbf{x}}) \qquad \text{fourth order}$$
$$\dots$$

Note that for an explicit method all the terms involved in the right hand side of the last expression are known. For implicit schemes, they will modify the structure of the algebraic equations obtained.

3.3 Derivative free formulation via polynomial corrections

The correction terms in (3.6) become more and more cumbersome and costly as the order of accuracy is increased, especially in three space dimensions. We propose here a different formulation that somehow simplifies the evaluation of these terms, especially on straight-sided simplicial meshes. For both nodal and modal bases defined in physical space, we easily evaluate these terms without the need of a map to the reference space, which would be instead required for curved elements.

Starting from a Taylor series expansion of arbitrary order of accuracy for a generic variable p, we can recast it in terms of increment as:

$$p(\mathbf{x}) - p(\tilde{\mathbf{x}}) = \nabla p(\tilde{\mathbf{x}}) \cdot \mathbf{d} + \frac{1}{2}\mathbf{d}^\intercal \cdot \mathscr{H}(p(\tilde{\mathbf{x}})) \cdot \mathbf{d} + \dots .$$

Therefore, by simply evaluating the polynomial in $\mathbf{x} = \tilde{\mathbf{x}} + \mathbf{d}$ and calculating the difference with that evaluated in $\tilde{\mathbf{x}}$, we get all the necessary correction terms in one polynomial evaluation. For example, the modified Dirichlet condition (3.6) can be written as

$$\rho_{SBM}(\tilde{\mathbf{x}}) = \rho_D(\tilde{\mathbf{x}} + \mathbf{d}) - [\rho(\tilde{\mathbf{x}} + \mathbf{d}) - \rho(\tilde{\mathbf{x}})] = \rho(\tilde{\mathbf{x}}) + [\rho_D(\tilde{\mathbf{x}} + \mathbf{d}) - \rho(\tilde{\mathbf{x}} + \mathbf{d})]\,, \tag{3.7}$$

which shows that the correction of the SBM method can be also seen as a direct shift on the surrogate boundary of the extrapolated polynomial error on the true boundary. This much simpler formulation only requires an extra polynomial evaluation, and is readily implemented on straight sides simplex elements for which the bases are easily expressed in physical space. As we mentioned above, these extrapolated variables are then used to compose a *ghost state* $\mathbf{u}^{bc}$ that will be used, along with the internal state $\mathbf{u}_h^-$, as input for the numerical flux $\hat{\mathcal{F}}(\mathbf{u}_h^-, \mathbf{u}^{bc})$ to obtain the consistent boundary flux $\mathcal{F}_{\mathbf{n}}^{bc}$.

3.4 Treatment of slip-wall boundary conditions

A similar approach can be applied to impose the slip wall condition (3.1). An important issue to take into account in this case is that, besides the position of the surrogate wall boundary $\tilde{\Gamma}_w$, also its normal $\tilde{\mathbf{n}}$ does not coincide with $\mathbf{n}$, the normal to the true wall boundary Γ_w. This difference affects both the magnitude and rate of convergence of the error as shown, e.g. in [187, 31, 30]. To overcome this issue, here we start from the formulation used in [279].

We start by decomposing the unit normal vector $\tilde{\mathbf{n}}$ at $\tilde{\mathbf{x}}$ as

$$\tilde{\mathbf{n}} = (\tilde{\mathbf{n}} \cdot \mathbf{n})\,\mathbf{n} + \sum_{k=1}^{d-1} (\tilde{\mathbf{n}} \cdot \mathbf{t}_k)\,\mathbf{t}_k,$$

where $\mathbf{t}_k$ are the vectors tangent to Γ_w. By doing so, $\mathcal{F}_{\tilde{\mathbf{n}}}^{bc}$ can be recast as

$$\mathcal{F}_{\tilde{\mathbf{n}}}^{bc} = (\tilde{\mathbf{n}} \cdot \mathbf{n})\mathcal{F}_{\mathbf{n}} + \sum_{k=1}^{d-1} (\tilde{\mathbf{n}} \cdot \mathbf{t}_k)\mathcal{F}_{\mathbf{t}_k}. \tag{3.8}$$

Then we can apply the Taylor expansion to the normal velocity appearing in the flux terms,

$$\mathcal{F}_{\mathbf{n}} = \begin{pmatrix} \rho w_{SBM} \\ \rho w_{SBM}\,\mathbf{u} + p\,\mathbf{n} \\ \rho w_{SBM}\,H \end{pmatrix}, \qquad \mathcal{F}_{\mathbf{t}_k} = \begin{pmatrix} \rho\, v_{\mathbf{t}_k} \\ \rho\, v_{\mathbf{t}_k}\,\mathbf{u} + p\,\mathbf{t}_k \\ \rho\, v_{\mathbf{t}_k}\,H \end{pmatrix}, \tag{3.9}$$

such that,

$$
\begin{aligned}
w_{SBM} = w &- \|\mathbf{d}\| \sum_{i=1}^{d} n_i \sum_{j=1}^{d} n_j \, \partial_{x_j} v_i(\tilde{\mathbf{x}}) && \text{second order} \\
&- \|\mathbf{d}\|^2 \sum_{i=1}^{d} n_i \sum_{j=1}^{d} \sum_{k=1}^{d} \frac{1}{2!} \, n_j n_k \, \partial^2_{x_j x_k} v_i(\tilde{\mathbf{x}}) && \text{third order} \\
&- \|\mathbf{d}\|^3 \sum_{i=1}^{d} n_i \sum_{j=1}^{d} \sum_{k=1}^{d} \sum_{\ell=1}^{d} \frac{1}{3!} \, n_j n_k n_\ell \, \partial^3_{x_j x_k x_\ell} v_i(\tilde{\mathbf{x}}) && \text{fourth order}
\end{aligned}
\tag{3.10}
$$

having set $\mathbf{n} = \{n_j\}_{j=1,\dots,d}$ and $\mathbf{v} = \{v_j\}_{j=1,\dots,d}$.

As done before, we use the fact that the Taylor series development on the right hand side of (3.10) is exact when applied to a polynomial of degree low enough to simplify it and recast it as a correction as

$$
w_{SBM} = \mathbf{v}(\tilde{\mathbf{x}}) \cdot \mathbf{n} + \left[w - \mathbf{v}(\tilde{\mathbf{x}} + \mathbf{d}) \cdot \mathbf{n} \right]. \tag{3.11}
$$

It should be noticed that, without the formulation in Equation (3.11), to perform the extrapolation given by Equation (3.10), high order derivatives of the the velocity components are needed. However, since all derivatives are usually computed with respect to the conservative variables, either the *chain rule* or some kind of linearization should be implemented to recover the higher order derivatives. For non moving walls with $w = 0$ we consider a simpler strategy to impose $W := (\rho \mathbf{v} \cdot \mathbf{n})|_{\Gamma_w} = 0$. This allows to work directly with derivatives and variations of the momentum variable. We thus replace (3.10) by

$$
\begin{aligned}
W_{SBM} = &- \|\mathbf{d}\| \sum_{i=1}^{d} n_i \sum_{j=1}^{d} n_j \, \partial_{x_j} (\rho v)_i(\tilde{\mathbf{x}}) && \text{second order} \\
&- \|\mathbf{d}\|^2 \sum_{i=1}^{d} n_i \sum_{j=1}^{d} \sum_{k=1}^{d} \frac{1}{2!} \, n_j n_k \, \partial^2_{x_j x_k} (\rho v)_i(\tilde{\mathbf{x}}) && \text{third order} \\
&- \|\mathbf{d}\|^3 \sum_{i=1}^{d} n_i \sum_{j=1}^{d} \sum_{k=1}^{d} \sum_{\ell=1}^{d} \frac{1}{3!} \, n_j n_k n_\ell \, \partial^3_{x_j x_k x_\ell} (\rho v)_i(\tilde{\mathbf{x}}) && \text{fourth order}
\end{aligned}
\tag{3.12}
$$

and (3.11) by

$$
W_{SBM} = (\rho \mathbf{v})(\tilde{\mathbf{x}}) \cdot \mathbf{n} - (\rho \mathbf{v})(\tilde{\mathbf{x}} + \mathbf{d}) \cdot \mathbf{n} \tag{3.13}
$$

As discussed before a fully consistent definition of the flux is obtained by consistently correcting the value of the total enthalpy as in (3.3), replacing w by w_{SBM} or by W_{SBM}/ρ

depending on the case.

Remark 25 (Boundary flux and penalty term)**.** *When using the numerical flux* (3.2)*, instead of a classical numerical flux* $\hat{\mathcal{F}}(\mathbf{u}_h^-, \mathbf{u}_{\tilde{\Gamma}}^{bc})$*, for higher (third, fourth, etc) order schemes, obtaining the correct convergence rates has required the inclusion of a penalty term similar to the diffusion term of the Rusanov flux. For slip walls this term reads*

$$\mathcal{P}_w := \alpha_w \left(\mathbf{u} - \mathbf{u}_{\tilde{\Gamma}}^{bc}\right) = \alpha_w \, \rho \begin{pmatrix} 0 \\ \mathbf{v}\cdot\mathbf{n} - w_{SBM} \\ \dfrac{(\mathbf{v}\cdot\mathbf{n})^2}{2} - \dfrac{w_{SBM}^2}{2} \end{pmatrix}, \tag{3.14}$$

where $\alpha_w = \|\mathbf{v}\| + \sqrt{\gamma p/\rho}$*.*

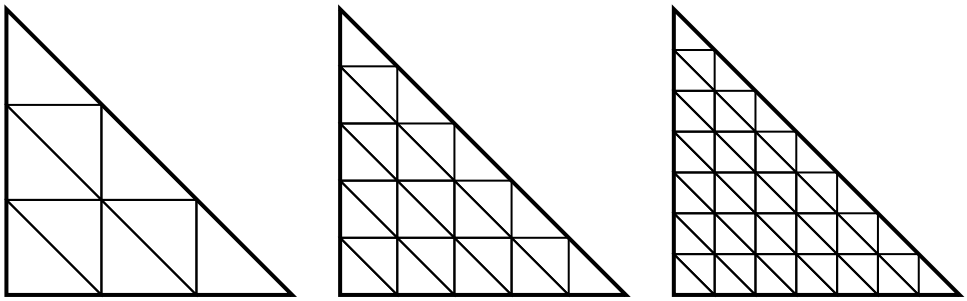
3.5 Other existing approaches

The corrections proposed in the previous paragraph will be compared to the approach proposed by Krivodonova and Berger in [187], referred to as *algorithm I* in the reference. We recall this approach for completeness. We start by defining in each quadrature point a special state of primitive variables $\mathbf{u}^b$, and a corresponding numerical flux:

$$\mathbf{u}^b = \begin{pmatrix} \rho^b \\ \mathbf{v}^b \\ p^b \end{pmatrix} = \begin{pmatrix} \rho \\ v_t\,\mathbf{t} \\ p \end{pmatrix}, \qquad \mathcal{F}_{\tilde{\mathbf{n}}}^{bc}(\mathbf{u}^b) = \begin{pmatrix} \rho^b\,(\mathbf{v}^b\cdot\tilde{\mathbf{n}}) \\ \rho^b\,(\mathbf{v}^b\cdot\tilde{\mathbf{n}})\,\mathbf{v}^b + p\,\tilde{\mathbf{n}} \\ \rho^b\,(\mathbf{v}^b\cdot\tilde{\mathbf{n}})\,H^b \end{pmatrix} \tag{3.15}$$

Note that [187] contains a typo in the flux expression which does not include the pressure term. Even though the modification introduced in (3.15) already allows a fair improvement of the discretization error and convergence rate, we will see that this shows some limitations when increasing the accuracy already beyond third order [102].

Remark 26. *Flux* (3.15) *can be easily recovered from Equations* (3.8) *and* (3.9)*, by*



(a) $N = 1$ (b) $N = 2$ (c) $N = 3$

Figure 3.4: Sub-triangulation of K for polynomials of degree N.

imposing $w = 0$*:*

$$\begin{aligned}
\mathcal{F}^{bc}_{\tilde{\mathbf{n}}} &= \begin{pmatrix} \rho\, w \\ \rho\, w\,(w\,\mathbf{n} + v_t\,\mathbf{t}) + p\,\mathbf{n} \\ \rho\, w\, \bar{H} \end{pmatrix} (\tilde{\mathbf{n}}\cdot\mathbf{n}) + \begin{pmatrix} \rho\, v_t \\ \rho\, v_t\,(w\,\mathbf{n} + v_t\,\mathbf{t}) + p\,\mathbf{t} \\ \rho\, v_t\, \bar{H} \end{pmatrix} (\tilde{\mathbf{n}}\cdot\mathbf{t}) \\
&= \begin{pmatrix} 0 \\ p\,\mathbf{n} \\ 0 \end{pmatrix} (\tilde{\mathbf{n}}\cdot\mathbf{n}) + \begin{pmatrix} \rho\, v_t \\ \rho\, v_t\,(v_t\,\mathbf{t}) + p\,\mathbf{t} \\ \rho\, v_t\, \bar{H} \end{pmatrix} (\tilde{\mathbf{n}}\cdot\mathbf{t}) \\
&= \begin{pmatrix} \rho\, v_t(\mathbf{t}\cdot\tilde{\mathbf{n}}) \\ \rho\, v_t\,(\mathbf{t}\cdot\tilde{\mathbf{n}})\, v_t\,\mathbf{t} + p\,\tilde{\mathbf{n}} \\ \rho\, v_t\,(\mathbf{t}\cdot\tilde{\mathbf{n}})\, \bar{H} \end{pmatrix}
\end{aligned} \tag{3.16}$$

3.6 Shock limiting

To handle discontinuous solutions in the time dependent case we use the MOOD approach [92, 116, 117], which has already been effectively applied in the ADER framework [55, 56, 203]. The algorithm is based on an *a posteriori* technique. The solution is first evolved from t^n to t^{n+1} using a high order ADER-DG (see Section 2.5.3 for further details) method. Then several admissibility criteria are checked, and the solution in all *troubled cells* is recomputed a-posteriori using a MUSCL-Hancock TVD (see Section 2.2.1 for further details) finite volume scheme. Herein we are going to briefly describe the main concept behind this approach. More insights can be found in the aforementioned references.

The first step consists in finding a high order candidate solution at time t^{n+1} that is denoted by $\mathbf{u}_h^{n+1,*}(\mathbf{x}, t^{n+1})$. Then, we define a sub-triangulation of K, made up by a set of

$N_k = (2N+1)^d$ non-overlapping sub-triangles k_α (see Fig. 3.4), whose volume is denoted by $|k_\alpha|$. Consequently, each space-time control volume is split into sub-prisms, called sub-volumes. For any sub-triangle, we can define the corresponding sub-cell average at time t^n:

$$\mathbf{V}_\alpha^n(\mathbf{x}, t^n) = \frac{1}{|k_\alpha|} \int_{k_\alpha} \mathbf{u}_h^n(\mathbf{x}, t^n) \mathrm{d}\mathbf{x} := \mathcal{P}(\mathbf{u}_h^n), \quad \forall \alpha \in [1, N_k], \tag{3.17}$$

and the candidate sub-cell average at time t^{n+1}

$$\mathbf{V}_\alpha^{n+1}(\mathbf{x}, t^n) = \frac{1}{|k_\alpha|} \int_{k_\alpha} \mathbf{u}_h^{n+1}(\mathbf{x}, t^{n+1}) \mathrm{d}\mathbf{x} := \mathcal{P}(\mathbf{u}_h^{n+1,*}), \quad \forall \alpha \in [1, N_k], \tag{3.18}$$

where $\mathcal{P}(\mathbf{u}_h)$ is the L_2 projection operator into the space of piecewise constant cell averages.

Secondly, the troubled cells have to be identified by checking the candidate solution $\mathbf{u}_h^{n+1,*}(\mathbf{x}, t^{n+1})$ with a set of detection criteria. Herein we follow the criteria described in [54]. Thus, it is required that the computed solution is physically acceptable, i.e. that it belongs to the phase space of the conservation law being solved. For instance, when working with the Euler equation of gasdynamics, density and pressure should be positive, meaning that computationally speaking they are greater that a prescribed tolerance $\nu = 10^{-12}$. Then, the solution should verify a relaxed discrete maximum principle (DMP):

$$\min_{\mathbf{m} \in \mathcal{V}(K)} \left(\min_{\beta \in [1, N_k]} \left(\mathbf{V}_{\mathbf{m},\beta}^n \right) \right) - \delta \leq \mathbf{V}_\alpha^{n+1} \leq \max_{\mathbf{m} \in \mathcal{V}(K)} \left(\max_{\beta \in [1, N_k]} \left(\mathbf{V}_{\mathbf{m},\beta}^n \right) \right) + \delta, \quad \forall \alpha \in [1, N_k], \tag{3.19}$$

where $\mathcal{V}(K)$ is the set of elements containing all the neighbors of K, i.e. sharing a common node with K, and δ is a parameter which, according to [140], reads

$$\delta = \max \left(\delta_0,\ \delta_1 \left[\max_{\mathbf{m} \in \mathcal{V}(K)} \left(\max_{\beta \in [1, N_k]} \left(\mathbf{V}_{\mathbf{m},\beta}^n \right) \right) - \min_{\mathbf{m} \in \mathcal{V}(K)} \left(\min_{\beta \in [1, N_k]} \left(\mathbf{V}_{\mathbf{m},\beta}^n \right) \right) \right] \right), \tag{3.20}$$

where $\delta_0 = 10^{-4}$ and $\delta_1 = 10^{-3}$.

If a cell does not fulfill the detection criteria in all its sub-cells, then it is flagged as *troubled*. It is likely that some false positive activations of the limiter occur; however these local effects do not reduce the overall quality of the simulations. Only on these troubles cells we apply a second order accurate MUSCL-Hancock TVD finite volume scheme. In this way, we can re-compute the solution in order to evolve the cell averages $\mathbf{V}_\alpha^n$ in time and obtain $\mathbf{V}_\alpha^{n+1}$. Finally, we can recover from these cell averages a high order polynomial

$\mathbf{u}_h^{n+1}$ of degree N. This is done by applying a reconstruction operator $\mathcal{R}$ such that

$$\int_{k_\alpha} \mathbf{u}_h^{n+1}(\mathbf{x}, t^{n+1}) \mathrm{d}\mathbf{x} = \int_{k_\alpha} \mathbf{V}_\alpha^{n+1}(\mathbf{x}, t^{n+1}) \mathrm{d}\mathbf{x} := \mathcal{R}\left(\mathbf{V}_\alpha^{n+1}(\mathbf{x}, t^{n+1})\right), \quad \forall \alpha \in [1, N_k], \tag{3.21}$$

which is conservative on the main cell K thanks to the additional linear constraint

$$\int_K \mathbf{u}_h^{n+1}(\mathbf{x}, t^{n+1}) \mathrm{d}\mathbf{x} = \int_K \mathbf{V}_\alpha^{n+1}(\mathbf{x}, t^{n+1}) \mathrm{d}\mathbf{x}. \tag{3.22}$$

Remark 27. *The projection operator $\mathcal{P}$ and the reconstruction operator $\mathcal{R}$ satisfy the property $\mathcal{P} \cdot \mathcal{R} = \mathcal{I}$, with $\mathcal{I}$ being the identity operator.*

3.7 Numerical results

In this Section, we test the new modified boundary treatments with several academic test-cases proving that the new method is able to provide high-order convergence for both far-field and wall boundary conditions on 2D and 3D unstructured meshes. We also show the numerical results obtained on a problem that involves shocks, correctly captured in the framework of ADER-DG methods thanks to our *a posteriori* sub-cell FV limiting technique. The results are provided with convergence analysis performed with classical and modified boundary conditions, and, when possible, with curvilinear meshes.

3.7.1 2D tests with smooth solutions

We start our suite of benchmarks with two-dimensional tests involving smooth solution profiles; this easily allows to asses the claimed properties of the proposed SBM formulation.

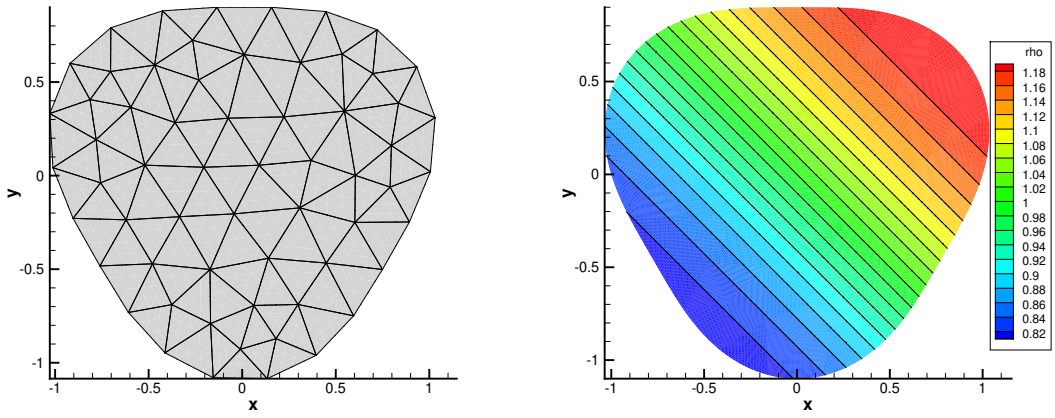
Manufactured solution on 2D curved domains: far-field BC

In order to assess the capability of our new flux correction, we start from a manufactured solution by considering the two-dimensional inhomogeneous Euler equations:

$$\partial_t \mathbf{u} + \nabla \cdot \mathcal{F}(\mathbf{u}) = \mathcal{S}, \qquad \text{with} \qquad \mathcal{S} = \begin{pmatrix} 0.4\,\cos(x+y) \\ 0.6\,\cos(x+y) \\ 0.6\,\cos(x+y) \\ 1.8\,\cos(x+y) \end{pmatrix}. \tag{3.23}$$

This system has the following exact steady state solution, as given in [214],

$$\rho = 1 + 0.2\,\sin(x+y), \quad v_x = 1, \quad v_y = 1, \quad p = 1 + 0.2\,\sin(x+y), \tag{3.24}$$



(a) Coarse mesh

(b) Initial condition

Figure 3.5: Manufactured solution (2D): test-case setup.

which is imposed on the real curved domain as far-field boundary conditions. A very coarse mesh was generated and then refined by splitting. The four nested grids described in Table 3.1 have been used to perform grid-convergence analysis.

The boundary where the far-field condition is applied will be referred to as Γ_D. We tested the new SBM flux correction on a complicated geometry, taken from [102], that can be described with the following equation written in polar coordinates:

$$\Gamma_D : \begin{pmatrix} x \\ y \end{pmatrix} = r(\alpha,\theta) \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \quad \text{where} \quad r(\alpha,\theta) = r_0 \left(1 + \frac{1}{10}\sin(\alpha\theta)\right), \quad r_0, \alpha \in \mathbb{R}, \tag{3.25}$$

where $r_0 = 1$ and $\alpha = 3$. We refer to Figure 3.5 for a visual representation.

Since the standard far-field boundary condition is enforced onto a curved boundary, discretized with a polygonal mesh, we expect to have second order of accuracy at best no matter what the degree of the polynomial is and this is well-observed in Table 3.2. This problem is cured of course when one uses iso-parametric elements and curved meshes. For the sake of completeness we provide throughout this chapter also some results on curved meshes, see for example the central part of Table 3.3, but we underline that our technology for curved meshes allows to obtain third order accurate results with DG-$\mathbb{P}2/\mathbb{Q}2$ and slightly better results when using DG-$\mathbb{P}3/\mathbb{Q}3$, but without achieving the expected fourth order.

Finally, the best results, in terms of error magnitude and convergence rates, are obtained with the shifted boundary correction, as one can see by comparing the previous results with those reported in Table 3.4. Convergence plots for the conservative variable ρ are presented in Figure 3.8a. All convergence trends are recovered until order four.

Table 3.1: Manufactured solution (2D): mesh characteristics.

Grid level	Nodes	Triangles	h
0	61	98	1.5450E-1
1	219	392	7.7252E-2
2	829	1,568	3.8626E-2
3	3,225	6,272	1.9313E-2

Table 3.2: Manufactured solution (2D): convergence tests without the SBM correction.

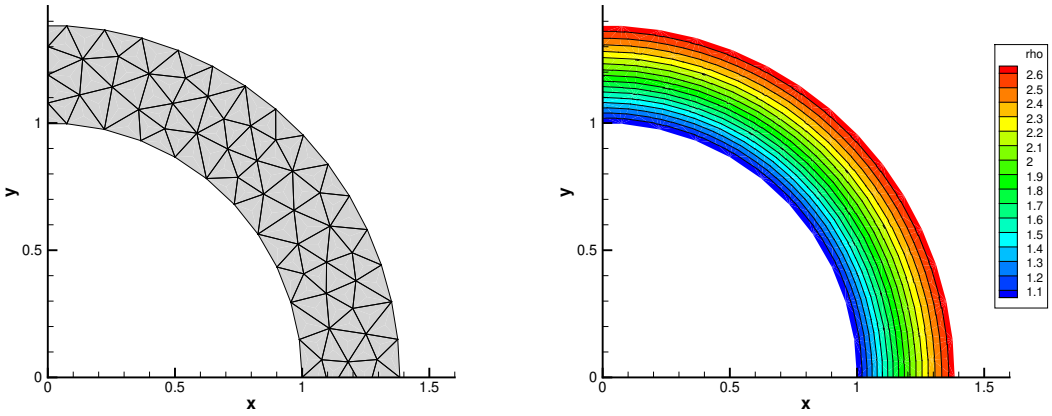
	ρ		ρv_x		ρv_y		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
				DG-$\mathbb{P}1$				
0	1.7293E-3	–	2.3514E-3	–	2.2344E-3	–	6.6466E-3	–
1	4.4790E-4	1.95	6.0925E-4	1.95	5.7869E-4	1.95	1.7138E-3	1.96
2	1.1276E-4	1.99	1.5358E-4	1.99	1.4579E-4	1.99	4.3195E-4	1.99
3	2.8241E-5	2.00	3.8473E-5	2.00	3.6512E-5	2.00	1.0820E-4	2.00
				DG-$\mathbb{P}2$				
0	1.6583E-3	–	2.3344E-3	–	2.2040E-3	–	6.4258E-3	–
1	4.2195E-4	1.97	5.9156E-4	1.98	5.5908E-4	1.98	1.6308E-3	1.98
2	1.0522E-4	2.00	1.4777E-4	2.00	1.3975E-4	2.00	4.0791E-4	2.00
3	2.6187E-5	2.01	3.6855E-5	2.00	3.4867E-5	2.00	1.0170E-4	2.00
				DG-$\mathbb{P}3$				
0	1.6817E-3	–	2.3809E-3	–	2.2562E-3	–	6.5431E-3	–
1	4.3095E-4	1.96	6.0642E-4	1.97	5.7430E-4	1.97	1.6640E-3	1.98
2	1.0768E-4	2.00	1.5092E-4	2.01	1.4280E-4	2.01	4.1580E-4	2.00
3	2.6792E-5	2.01	3.7467E-5	2.01	3.5428E-5	2.01	1.0345E-4	2.01

Table 3.3: Manufactured solution (2D): convergence tests with curved meshes.

	ρ		ρv_x		ρv_y		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
				DG-$\mathbb{P}1/\mathbb{Q}1$				
0	1.7293E-3	–	2.3514E-3	–	2.2344E-3	–	6.6466E-3	–
1	4.4790E-4	1.95	6.0925E-4	1.95	5.7869E-4	1.95	1.7138E-3	1.96
2	1.1276E-4	1.99	1.5358E-4	1.99	1.4579E-4	1.99	4.3195E-4	1.99
3	2.8241E-5	2.00	3.8473E-5	2.00	3.6512E-5	2.00	1.0820E-4	2.00
				DG-$\mathbb{P}2/\mathbb{Q}2$				
0	7.8113E-5	–	4.9287E-5	–	6.0091E-5	–	2.3914E-4	–
1	1.1494E-5	2.76	6.1144E-6	3.01	7.9719E-6	2.91	3.4731E-5	2.78
2	1.5902E-6	2.85	7.8242E-7	2.97	1.0271E-6	2.96	4.7828E-6	2.86
3	2.1096E-7	2.91	1.0184E-7	2.94	1.3078E-7	2.97	6.3447E-7	2.91
				DG-$\mathbb{P}3/\mathbb{Q}3$				
0	5.2248E-6	–	4.7012E-6	–	4.6229E-6	–	1.5742E-5	–
1	5.6390E-7	3.21	4.7322E-7	3.31	4.7014E-7	3.29	1.6226E-6	3.28
2	6.1309E-8	3.20	4.7773E-8	3.31	4.8063E-8	3.29	1.7095E-7	3.25
3	1.0299E-8	2.57	9.2506E-9	2.36	1.0123E-8	2.25	3.6478E-8	2.23

Table 3.4: Manufactured solution (2D): convergence tests with the SBM correction.

	ρ		ρv_x		ρv_y		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
				DG-$\mathbb{P}1$/SBM-$\mathbb{P}1$				
0	1.0784E-3	–	1.2139E-3	–	1.2479E-3	–	3.3758E-3	–
1	2.3747E-4	2.18	2.4029E-4	2.34	2.4644E-4	2.34	7.2503E-4	2.22
2	5.5268E-5	2.10	5.3230E-5	2.17	5.4522E-5	2.18	1.6929E-4	2.10
3	1.3351E-5	2.05	1.2599E-5	2.08	1.2898E-5	2.08	4.1189E-5	2.04
				DG-$\mathbb{P}2$/SBM-$\mathbb{P}2$				
0	7.5258E-5	–	5.0257E-5	–	5.8531E-5	–	2.2711E-4	–
1	1.1105E-5	2.76	5.6058E-6	3.16	7.1750E-6	3.03	3.2987E-5	2.78
2	1.5701E-6	2.82	7.2761E-7	2.95	9.2582E-7	2.95	4.6465E-6	2.83
3	2.1377E-7	2.88	1.0003E-7	2.86	1.1793E-7	2.97	6.3270E-7	2.88
				DG-$\mathbb{P}3$/SBM-$\mathbb{P}3$				
0	1.7331E-6	–	2.8523E-6	–	2.9879E-6	–	6.3787E-6	–
1	7.0445E-8	4.62	9.8935E-8	4.85	1.0298E-7	4.86	2.3519E-7	4.76
2	3.2654E-9	4.43	3.9178E-9	4.66	4.0466E-9	4.67	1.0200E-8	4.53
3	1.7487E-10	4.22	1.8709E-10	4.39	1.9248E-10	4.39	5.3564E-10	4.25



(a) Coarse mesh

(b) Initial condition

Figure 3.6: Supersonic vortex bounded by two circular walls (2D): test case setup.

Supersonic vortex bounded by two circular walls: slip wall BC

In order to test the new flux corrections for wall boundary conditions, we consider an isentropic supersonic flow between two concentric circular arcs of radii $r_i = 1$ and $r_o = 1.384$. The exact density in terms of radius r is given by

$$\rho = \rho_i \left(1 + \frac{\gamma+1}{2} M_i^2 \left(1 - \left(\frac{r_i}{r}\right)^2\right)\right)^{\frac{1}{\gamma-1}}. \tag{3.26}$$

The velocity and pressure are given by

$$\|\mathbf{v}\| = \frac{c_i M_i}{r}, \quad p = \frac{\rho^\gamma}{\gamma}, \tag{3.27}$$

where c_i is the speed of sound on the inner circle. The Mach number on the inner circle M_i is set to 2.25 and the density ρ_i to 1.
The fluid's velocity vector components in (x, y) can be computed as follows:

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \|\mathbf{v}\| \begin{pmatrix} y/r \\ -x/r \end{pmatrix}, \tag{3.28}$$

where $r = \sqrt{x^2 + y^2}$.

A set of refined meshes is obtained by means of conformal refinement of an initial triangulation shown in Fig. 3.6. For this test case, we first performed the convergence test by enforcing the standard weak wall boundary condition on the polygonal boundary. Table 3.6 points out the grid-convergence analysis showing that the method is not even able to converge with a full second order trend. It can also be noticed that, for each mesh, the higher is the degree of the polynomial, the larger the error is. This is due to the fact that the scheme is trying to better approximate a solution that is indeed wrong because the boundary, which should be curved, is approximated by a polygon (see Fig. 3.3).

The same test, with the same boundary condition, has also been run with high order curvilinear meshes showing that when increasing the order of the polynomial used to approximate the boundary we are able to recover the formal order of accuracy of the method, as visible in Table 3.7. Note that this result is only possible when using isoparametric elements, so the degree of the polynomial $\mathbb{Q}$ that approximates the geometry and the mesh has to increase with the polynomial $\mathbb{P}$ of the scheme (e.g. DG-$\mathbb{P}1/\mathbb{Q}1$, DG-$\mathbb{P}2/\mathbb{Q}2$, and DG-$\mathbb{P}3/\mathbb{Q}3$). Convergence plots obtained with curvilinear element are shown in Figure 3.10a for the primitive variable ρ.

The results obtained using the SBM correction on straight sided meshes are reported in Table 3.8 and shown in Figure 3.10c, for the approach using the truncated Taylor series development for the velocity, and in Table 3.9 with the simplified polynomials correction (3.13) based on the momentum variables which requires no complex derivative evaluations. We observe the expected rates and low error levels in both cases, with a slight improvement with the simplified approach which may be related to the fact that we are extrapolating the whole ρv_n term, rather than only v_n, as done for the classical SBM formulation (with ρ taken from the quadrature point).

For the sake of completeness, we present in Table 3.10 and Figure 3.10b the results obtained with the correction (3.15). It should be noticed that even for this simple case, considering quadratic geometries, we have an order-of-accuracy degradation for finer meshes when using polynomials of degree higher than two ($\mathbb{P}3$). Instead, better convergence

Table 3.5: Supersonic vortex bounded by two circular walls (2D): mesh characteristics.

Grid level	Nodes	Triangles	h
0	238	376	1.024E-01
1	852	1,504	5.121E-02
2	3,208	6,016	2.560E-02
3	12,432	24,064	1.280E-02

Table 3.6: Supersonic vortex bounded by two circular walls (2D): convergence tests without SBM correction.

	ρ		ρv_x		ρv_y		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
				DG-$\mathbb{P}1$				
0	4.0507E-2	–	5.9773E-2	–	6.0006E-2	–	1.7033E-1	–
1	1.3141E-2	1.62	1.9227E-2	1.64	1.9330E-2	1.63	5.6565E-2	1.59
2	4.4645E-3	1.56	6.5585E-3	1.55	6.5822E-3	1.55	1.9499E-2	1.54
3	1.5786E-3	1.50	2.3646E-3	1.47	2.3684E-3	1.47	6.9664E-3	1.48
				DG-$\mathbb{P}2$				
0	7.5758E-2	–	1.3956E-1	–	1.4152E-1	–	3.3516E-1	–
1	2.7584E-2	1.46	5.1868E-2	1.43	5.1990E-2	1.44	1.2308E-1	1.45
2	9.6935E-3	1.51	1.8547E-2	1.48	1.8548E-2	1.49	4.3649E-2	1.50
3	3.4687E-3	1.48	6.5731E-3	1.50	6.5656E-3	1.50	1.5548E-2	1.49
				DG-$\mathbb{P}3$				
0	1.9104E-1	–	2.2847E-1	–	2.2992E-1	–	7.2469E-1	–
1	8.3972E-2	1.19	9.9892E-2	1.19	1.0244E-1	1.17	3.2130E-1	1.17
2	3.5508E-2	1.24	4.0370E-2	1.31	4.1040E-2	1.32	1.3453E-1	1.26
3	1.5314E-2	1.21	1.6278E-2	1.31	1.6507E-2	1.31	5.7225E-2	1.23

trends are given also in this situation by the SBM correction in Figure 3.10c.

3.7.2 3D tests with smooth solutions

We repeat the same study of the previous Section in the three-dimensional case.

Manufactured solution on 3D curved domains: far-field BC

We repeat the same study of the previous section in the three-dimensional case. We consider the three-dimensional inhomogeneous Euler equations:

$$\partial_t \mathbf{u} + \nabla \cdot \mathcal{F}(\mathbf{u}) = \mathcal{S}, \qquad \text{with} \qquad \mathcal{S} = \begin{pmatrix} 0.6\,\cos(x+y+z) \\ 0.8\,\cos(x+y+z) \\ 0.8\,\cos(x+y+z) \\ 0.8\,\cos(x+y+z) \\ 3.0\,\cos(x+y+z) \end{pmatrix}. \tag{3.29}$$

This system has the following exact steady state solution, recovered using the *manufactured solution technique*,

$$\rho = 1+0.2\,\sin(x+y+z), \quad v_x = 1, \quad v_y = 1, \quad v_z = 1, \quad p = 1+0.2\,\sin(x+y+z), \tag{3.30}$$

Table 3.7: Supersonic vortex bounded by two circular walls (2D): convergence test with curved meshes.

	ρ		ρv_x		ρv_y		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
				DG-$\mathbb{P}1/\mathbb{Q}1$				
0	4.0507E-2	–	5.9773E-2	–	6.0006E-2	–	1.7033E-1	–
1	1.3141E-2	1.62	1.9227E-2	1.64	1.9330E-2	1.63	5.6565E-2	1.59
2	4.4645E-3	1.56	6.5585E-3	1.55	6.5822E-3	1.55	1.9499E-2	1.54
3	1.5786E-3	1.50	2.3646E-3	1.47	2.3684E-3	1.47	6.9664E-3	1.48
				DG-$\mathbb{P}2/\mathbb{Q}2$				
0	1.2075E-3	–	1.7408E-3	–	1.7527E-3	–	4.8709E-3	–
1	1.9156E-4	2.66	2.6389E-4	2.72	2.6721E-4	2.71	7.7666E-4	2.65
2	2.5831E-5	2.89	3.3546E-5	2.98	3.3662E-5	2.99	1.0323E-4	2.91
3	3.0727E-6	3.07	4.1031E-6	3.03	4.0812E-6	3.04	1.2467E-5	3.05
				DG-$\mathbb{P}3/\mathbb{Q}3$				
0	7.0183E-5	–	8.8438E-5	–	8.9166E-5	–	2.7348E-4	–
1	4.8983E-6	3.84	6.7326E-6	3.72	6.7616E-6	3.72	1.9264E-5	3.83
2	4.0876E-7	3.58	5.9445E-7	3.50	5.9936E-7	3.50	1.6698E-6	3.50
3	3.2822E-8	3.64	5.1193E-8	3.54	5.1593E-8	3.54	1.3795E-7	3.60

Table 3.8: Supersonic vortex bounded by two circular walls (2D): convergence test with SBM correction (with Equation (3.10)).

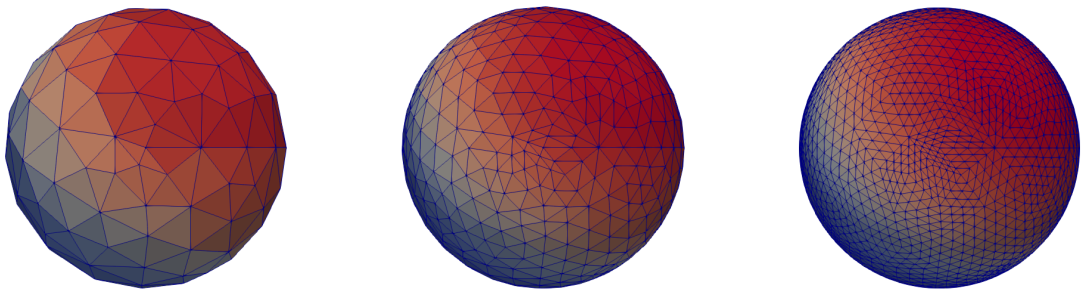
	ρ		ρv_x		ρv_y		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
				DG-$\mathbb{P}1$/SBM-$\mathbb{P}1$				
0	1.6446E-2	–	3.5594E-2	–	3.5415E-2	–	7.7895E-2	–
1	4.5403E-3	1.86	9.9195E-3	1.84	9.9279E-3	1.84	2.1613E-2	1.85
2	9.5300E-4	2.25	2.0713E-3	2.26	2.0770E-3	2.26	4.5047E-3	2.26
3	1.9068E-4	2.32	4.1246E-4	2.33	4.1396E-4	2.33	8.9777E-4	2.33
				DG-$\mathbb{P}2$/SBM-$\mathbb{P}2$				
0	1.2652E-3	–	1.8623E-3	–	1.8779E-3	–	5.2247E-3	–
1	1.9622E-4	2.69	2.6306E-4	2.82	2.6695E-4	2.81	7.9217E-4	2.72
2	2.5770E-5	2.93	3.3068E-5	2.99	3.3222E-5	3.01	1.0284E-4	2.95
3	3.0587E-6	3.08	4.0533E-6	3.03	4.0349E-6	3.04	1.2390E-5	3.05
				DG-$\mathbb{P}3$/SBM-$\mathbb{P}3$				
0	6.2321E-5	–	9.0618E-5	–	8.7148E-5	–	2.4363E-4	–
1	2.9113E-6	4.42	4.9355E-6	4.20	4.7997E-6	4.18	1.1976E-5	4.35
2	1.5293E-7	4.25	2.6814E-7	4.20	2.6463E-7	4.18	6.3695E-7	4.23
3	8.5252E-9	4.17	1.4884E-8	4.17	1.4727E-8	4.17	3.5447E-8	4.17

Table 3.9: Supersonic vortex bounded by two circular walls (2D): convergence test with SBM correction (with Equation (3.13)).

	ρ		ρv_x		ρv_y		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
				DG-$\mathbb{P}1$/SBM-$\mathbb{P}1$				
0	1.6335E-2	–	3.5355E-2	–	3.5183E-2	–	7.7374E-2	–
1	4.5259E-3	1.85	9.8900E-3	1.84	9.8996E-3	1.84	2.1548E-2	1.84
2	9.5188E-4	2.25	2.0689E-3	2.26	2.0748E-3	2.26	4.4993E-3	2.26
3	1.9058E-4	2.32	4.1225E-4	2.33	4.1376E-4	2.33	8.9728E-4	2.33
				DG-$\mathbb{P}2$/SBM-$\mathbb{P}2$				
0	1.2604E-3	–	1.8419E-3	–	1.8568E-3	–	5.1834E-3	–
1	1.9606E-4	2.69	2.6221E-4	2.82	2.6608E-4	2.81	7.9063E-4	2.72
2	2.5765E-5	2.93	3.3039E-5	2.99	3.3192E-5	3.01	1.0279E-4	2.95
3	3.0584E-6	3.08	4.0524E-6	3.03	4.0340E-6	3.04	1.2388E-5	3.05
				DG-$\mathbb{P}3$/SBM-$\mathbb{P}3$				
0	6.2289E-5	–	9.1157E-5	–	8.7853E-5	–	2.4332E-4	–
1	2.8518E-6	4.45	4.9380E-6	4.20	4.7981E-6	4.19	1.1777E-5	4.37
2	1.4569E-7	4.29	2.6571E-7	4.22	2.6206E-7	4.19	6.1275E-7	4.26
3	7.9556E-9	4.20	1.4647E-8	4.18	1.4496E-8	4.18	3.3523E-8	4.19

Table 3.10: Supersonic vortex bounded by two circular walls (2D): convergence test with the modified flux introduced in (3.15).

	ρ		ρv_x		ρv_y		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
DG-$\mathbb{P}1$ with correction (3.15)								
0	1.9580E-2	–	4.1681E-2	–	4.1868E-2	–	8.9637E-2	–
1	5.3355E-3	1.88	1.1620E-2	1.84	1.1676E-2	1.84	2.4676E-2	1.86
2	1.1690E-3	2.19	2.5781E-3	2.17	2.5983E-3	2.17	5.3291E-3	2.21
3	2.4711E-4	2.24	5.5186E-4	2.22	5.5738E-4	2.22	1.0988E-3	2.28
DG-$\mathbb{P}2$ with correction (3.15)								
0	1.2764E-3	–	1.8543E-3	–	1.8656E-3	–	5.2077E-3	–
1	1.9749E-4	2.69	2.6604E-4	2.80	2.6928E-4	2.79	7.9744E-4	2.71
2	2.5994E-5	2.92	3.3359E-5	2.99	3.3458E-5	3.00	1.0360E-4	2.94
3	3.0959E-6	3.06	4.1008E-6	3.02	4.0776E-6	3.03	1.2531E-5	3.04
DG-$\mathbb{P}3$ with correction (3.15)								
0	8.3269E-5	–	1.0748E-4	–	1.0489E-4	–	3.2138E-4	–
1	5.3483E-6	3.96	6.6779E-6	4.00	6.5981E-6	3.99	2.0358E-5	3.98
2	4.6348E-7	3.52	5.0053E-7	3.73	5.0276E-7	3.71	1.7297E-6	3.55
3	4.5383E-8	3.35	4.3567E-8	3.52	4.3975E-8	3.51	1.6679E-7	3.37



(a) Grid level 0 (b) Grid level 1 (c) Grid level 2

Figure 3.7: Manufactured solution (3D): density contours on the three grid levels.

Table 3.11: Manufactured solution (3D): mesh characteristics.

'rid level	Nodes	Tetrahedra	h
0	169	778	2.8274E-01
1	971	5,072	1.4137E-01
2	6,437	35,968	7.0686E-02

Table 3.12: Manufactured solution (3D): convergence tests without SBM correction.

	ρ		ρv_x		ρv_y		ρv_z		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
					DG-$\mathbb{P}1$					
0	8.7027E-3	–	8.8173E-3	–	8.8456E-3	–	8.7977E-3	–	3.3551E-2	–
1	2.4345E-3	1.84	2.4372E-3	1.86	2.4416E-3	1.86	2.4330E-3	1.85	9.3166E-3	1.85
2	6.7942E-4	1.84	6.7824E-4	1.85	6.8014E-4	1.84	6.7829E-4	1.84	2.6093E-3	1.84
					DG-$\mathbb{P}2$					
0	7.6027E-4	–	9.1039E-4	–	8.9306E-4	–	8.4296E-4	–	3.2308E-3	–
1	1.7566E-4	2.11	2.1977E-4	2.05	2.1528E-4	2.05	1.9975E-4	2.08	7.8918E-4	2.03
2	4.2006E-5	2.06	5.4457E-5	2.01	5.3289E-5	2.01	4.8978E-5	2.03	1.9422E-4	2.02
					DG-$\mathbb{P}3$					
0	6.9598E-4	–	8.7766E-4	–	8.5810E-4	–	7.9330E-4	–	3.1103E-3	–
1	1.6687E-4	2.06	2.1777E-4	2.01	2.1300E-4	2.01	1.9540E-4	2.02	7.7173E-4	2.01
2	4.1490E-5	2.01	5.4856E-5	1.99	5.3567E-5	1.99	4.9074E-5	1.99	1.9345E-4	2.00

which is imposed on the domain boundaries as far-field boundary conditions.

A sphere is now considered as the boundary of domain in order to introduce the error given by the curvature. A detail of the domain and meshes is represented in Fig. 3.7. As expected when applying the far-field condition on the real curved boundary, the geometrical error given by the linear mesh overcomes by far that given by the discretization technique with the outcome that no better than second order of accuracy can be achieved (see Table 3.12). As before the best results are obtained when using the SBM correction, for which all high order convergences are well recovered as seen in Table 3.13. Again, convergence plots for the conservative variable ρ are presented in Figure 3.8b.

Table 3.13: Manufactured solution (3D): convergence tests with SBM correction.

	ρ		ρv_x		ρv_y		ρv_z		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
					DG-$\mathbb{P}1$/SBM-$\mathbb{P}1$					
0	9.0767E-3	–	9.1066E-3	–	9.1410E-3	–	9.1148E-3	–	3.5055E-2	–
1	2.4981E-3	1.86	2.4841E-3	1.87	2.4895E-3	1.88	2.4859E-3	1.87	9.5724E-3	1.87
2	6.9067E-4	1.85	6.8601E-4	1.86	6.8798E-4	1.86	6.8738E-4	1.85	2.6541E-3	1.85
					DG-$\mathbb{P}2$/SBM-$\mathbb{P}2$					
0	3.2537E-4	–	3.3772E-4	–	3.4049E-4	–	3.3983E-4	–	1.2000E-3	–
1	4.5765E-5	2.83	4.5417E-5	2.89	4.5165E-5	2.91	4.5562E-5	2.90	1.8400E-4	2.71
2	7.8831E-6	2.54	7.4985E-6	2.60	7.4576E-6	2.60	7.5288E-6	2.60	2.9674E-5	2.63
					DG-$\mathbb{P}3$/SBM-$\mathbb{P}3$					
0	2.0810E-5	–	2.3072E-5	–	2.3226E-5	–	2.2280E-5	–	8.4229E-5	–
1	1.3956E-6	3.90	1.4671E-6	3.98	1.4566E-6	4.00	1.4633E-6	3.93	5.5552E-6	3.92
2	1.3372E-7	3.38	1.3562E-7	3.44	1.3579E-7	3.42	1.3562E-7	3.43	5.1363E-7	3.44

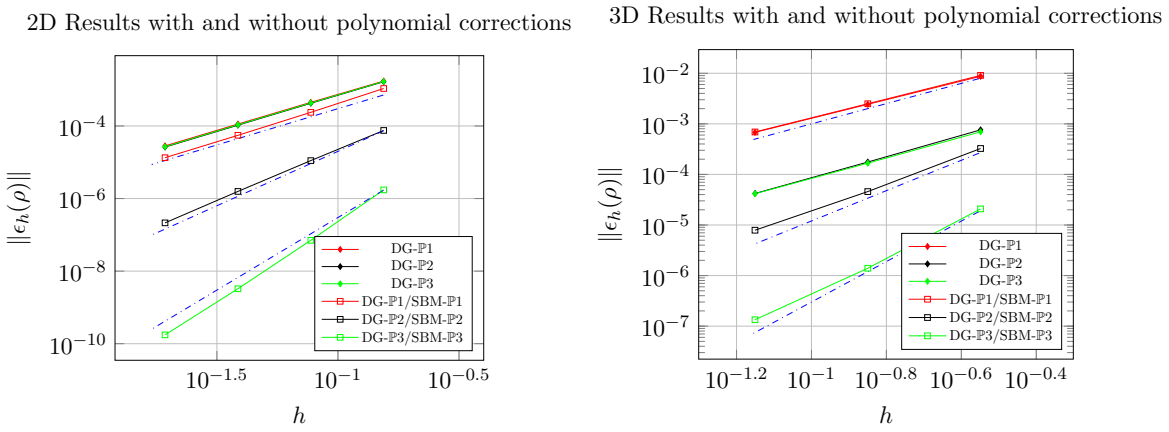


Figure 3.8: Manufactured solution (2D/3D): results obtained with and without the SBM correction.

Supersonic vortex bounded by two cylindrical walls: slip wall BC

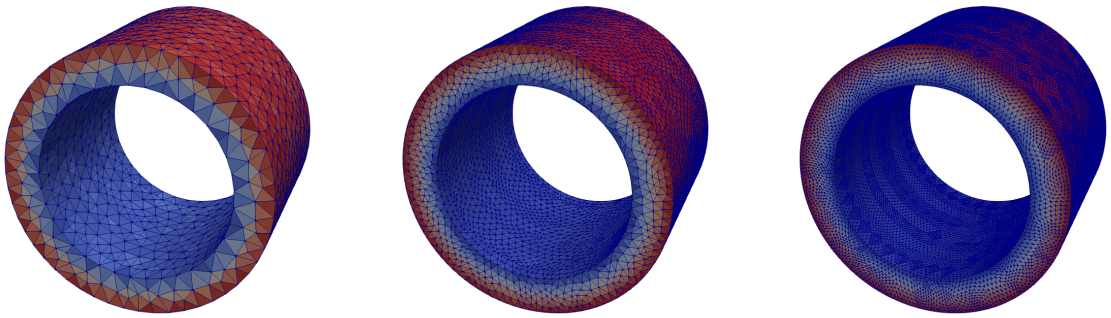
We consider a 3D isentropic supersonic flow between two concentric cylindrical surfaces of radii $r_i = 1$ and $r_o = 1.384$. The exact density, velocity and pressure in terms of radius r are given by Equations (3.26) and (3.27) and the solution in the inner surface is taken as that in Section 3.7.1. The fluid's velocity vector components in (x, y, z) can be computed as follows:

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \|\mathbf{v}\| \begin{pmatrix} y/r \\ -x/r \\ 0 \end{pmatrix}, \tag{3.31}$$

where $r = \sqrt{x^2 + y^2}$ because the cylindrical surfaces are developed along the z–axis. Simulations are first run with classical reflecting wall boundary conditions applied on the approximated boundary. The results presented in Table 3.15 show convergence trends of rates between 1.5 and 1 for all high order polynomials. Finally, Table 3.16 presents the results obtained using the SBM wall flux correction: already a little improvement is shown for $\mathbb{P}1$ and for higher order all convergence trends are properly recovered (see Figure 3.10d for the convergences obtained for the primitive variable ρ).

Table 3.14: Supersonic vortex bounded by two cylindrical walls (3D): mesh characteristics.

Grid level	Nodes	Tetrahedra	h
0	3,071	13,059	1.0382E-01
1	20,929	104,472	5.2221E-02
2	153,242	835,776	2.6112E-02



(a) Grid level 0 (b) Grid level 1 (c) Grid level 2

Figure 3.9: Supersonic vortex bounded by two cylindrical walls (3D): density contours.

Table 3.15: Supersonic vortex bounded by two cylindrical walls (3D): convergence tests without SBM correction.

	ρ		ρv_x		ρv_y		ρv_z		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
					DG-$\mathbb{P}1$					
0	3.7322E-2	–	6.4933E-2	–	6.5061E-2	–	1.1475E-2	–	1.6600E-1	–
1	1.4081E-2	1.41	2.0592E-2	1.65	2.0796E-2	1.65	5.0424E-3	1.18	6.0193E-2	1.46
2	5.0733E-3	1.47	6.6189E-3	1.64	6.7225E-3	1.63	1.9803E-3	1.35	2.1242E-2	1.50
					DG-$\mathbb{P}2$					
0	5.2040E-2	–	5.9910E-2	–	6.1607E-2	–	1.6668E-2	–	2.0989E-1	–
1	2.2376E-2	1.22	2.3548E-2	1.35	2.4191E-2	1.35	8.6415E-3	0.95	8.6540E-2	1.28
2	9.0045E-3	1.31	8.8598E-3	1.41	9.1564E-3	1.40	3.8322E-3	1.17	3.4325E-2	1.33
					DG-$\mathbb{P}3$					
0	8.0027E-2	–	8.787E-2	–	8.9290E-2	–	2.5696E-2	–	3.0165E-1	–
1	4.1512E-2	0.94	3.931E-2	1.16	4.0139E-2	1.15	1.5240E-2	0.75	1.5087E-1	1.00
2	2.0451E-2	1.02	1.729E-2	1.18	1.7731E-2	1.18	8.3006E-3	0.87	7.3682E-2	1.03

Table 3.16: Supersonic vortex bounded by two cylindrical walls (3D): convergence tests with the SBM correction.

	ρ		ρv_x		ρv_y		ρv_z		ρE	
Grid level	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$	L_2	$\tilde{n}$
					DG-$\mathbb{P}1$/SBM-$\mathbb{P}1$					
0	1.9580E-2	–	5.4668E-2	–	5.3611E-2	–	5.8383E-3	–	9.1443E-2	–
1	6.1094E-3	1.68	1.5653E-2	1.80	1.5334E-2	1.80	2.0797E-3	1.49	2.7820E-2	1.72
2	1.8376E-3	1.73	4.4901E-3	1.80	4.4063E-3	1.80	7.0284E-4	1.57	8.2183E-3	1.76
					DG-$\mathbb{P}2$/SBM-$\mathbb{P}2$					
0	1.3668E-3	–	2.4856E-3	–	2.4388E-3	–	6.4125E-4	–	6.3670E-3	–
1	1.9656E-4	2.80	3.6385E-4	2.77	3.5349E-4	2.79	9.7458E-5	2.72	9.2034E-4	2.79
2	3.0200E-5	2.70	5.3739E-5	2.76	5.2281E-5	2.75	1.5670E-5	2.64	1.3566E-4	2.76
					DG-$\mathbb{P}3$/SBM-$\mathbb{P}3$					
0	6.3903E-5	–	1.0992E-4	–	1.0858E-4	–	2.7024E-5	–	2.6291E-4	–
1	5.8075E-6	3.46	1.0098E-5	3.44	9.9331E-6	3.45	2.3153E-6	3.54	2.3666E-5	3.47
2	5.8223E-7	3.32	1.0373E-6	3.28	1.0171E-6	3.29	2.3309E-7	3.31	2.3781E-6	3.31

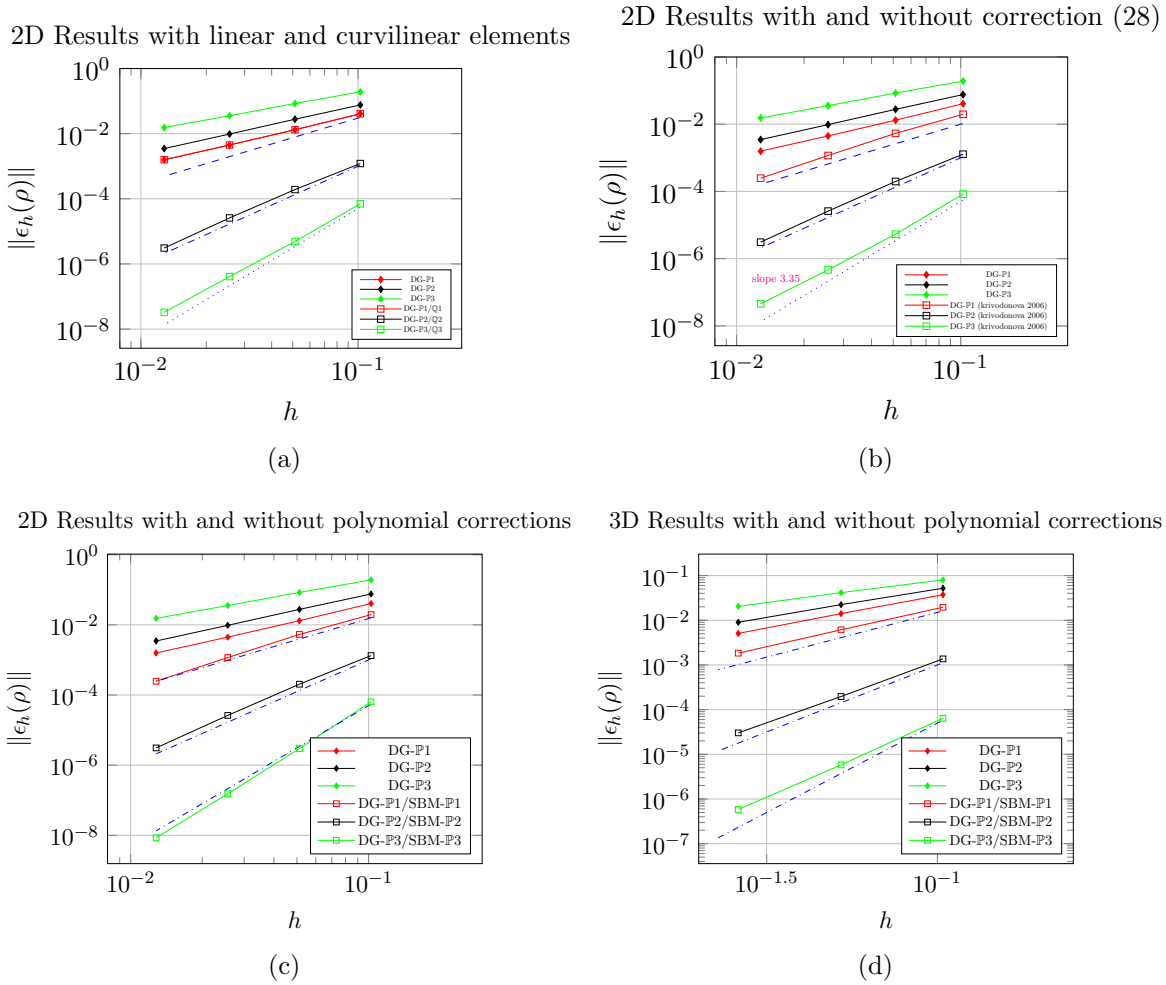
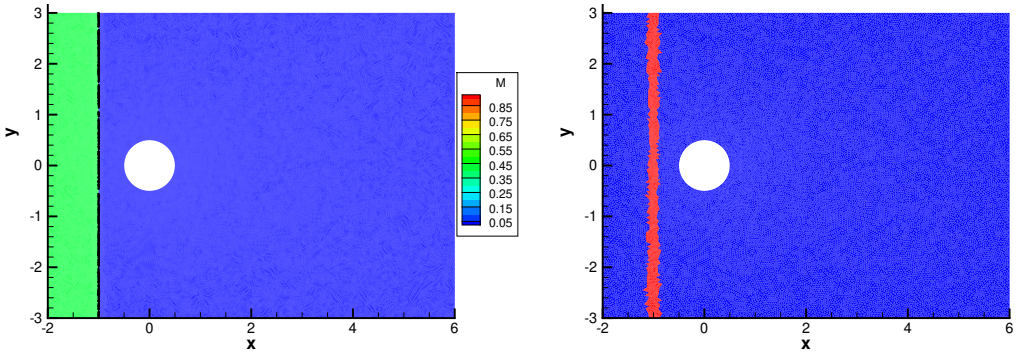


Figure 3.10: Supersonic vortex bounded by two walls (2D/3D): convergence tests.



(a) Initial condition

(b) Troubled cells

Figure 3.11: Shock-cylinder interaction (2D): test case set up.

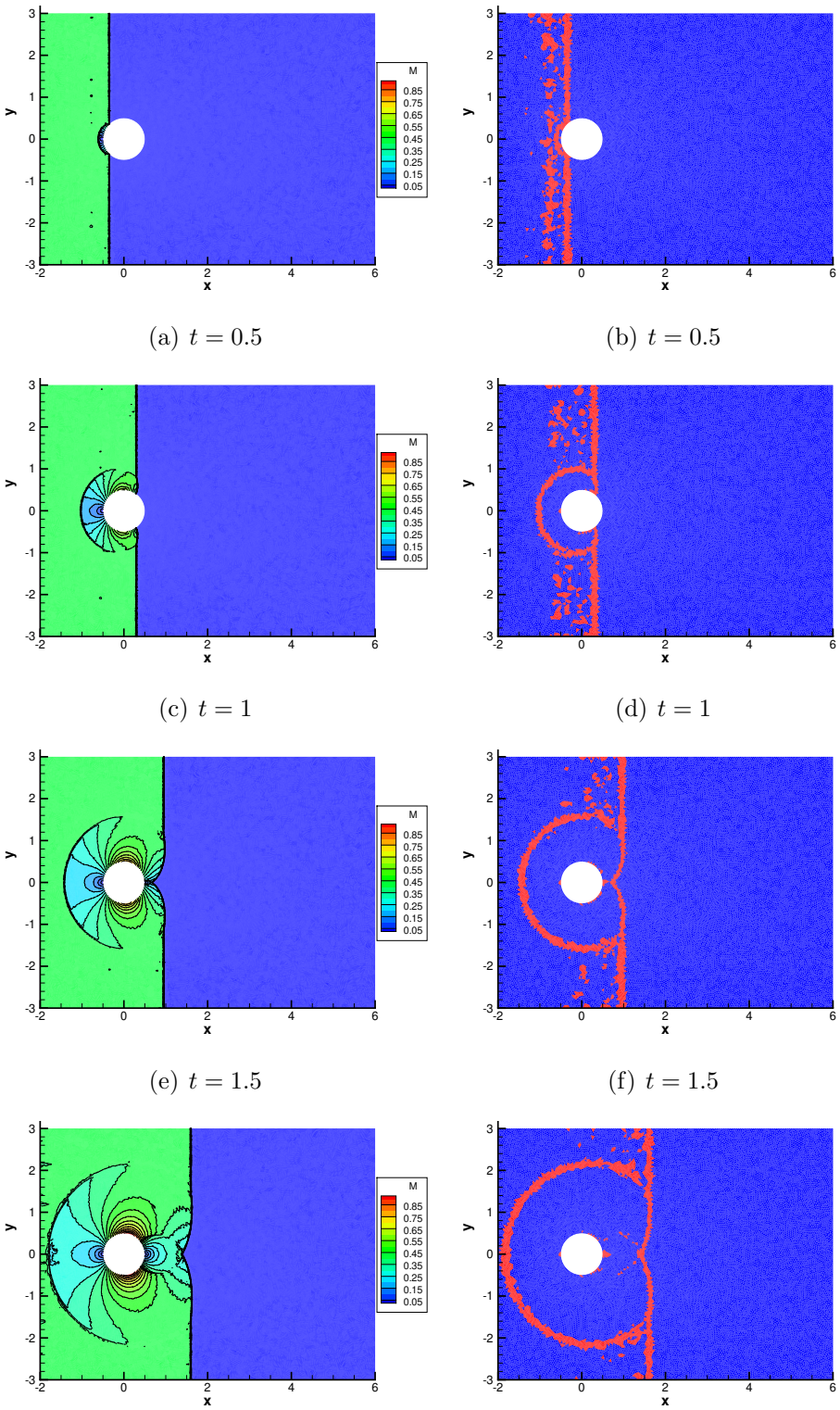
3.7.3 Shock-cylinder interaction

The problem we are looking to solve here is the interaction of a shock wave with a two-dimensional cylinder. The computational domain is $[-2, 6] \times [-3, 3]$ discretized with an unstructured triangulation made up by 7,761 grid points and 15,198 elements. The scheme is supplemented with *a posteriori* sub-cell finite volume limiter. The cylinder is centered in $(0, 0)$ and has radius 0.5. The initial condition of the shock wave of a *shock Mach number* $M_s = 1.3$ is setup via the Rankine-Hugoniot conditions. The flow upstream the shock is at rest and is characterized by density and pressure, respectively being $\rho = 1.4$ and $p = 1$.

The simulation has been run with polynomials $\mathbb{P}1$, $\mathbb{P}2$ and $\mathbb{P}3$ comparing the implemented wall boundary conditions, with and without the SBM flux correction. Figure 3.11 points out the limiter activations at time $t = 0$ of the simulation. In order to show the limiter activations over time, in Fig. 3.12 we plotted the troubled cells and solutions, at different time steps, computed with $\mathbb{P}3$ polynomials. No fundamental difference is observed in the limiter activations for the classical and new wall boundary conditions. Figure 3.13 presents a close up of the solutions around the body. It is again observed that by increasing the order of the polynomials the results obtained with classical wall boundary conditions get worse and the SBM flux correction really introduces a notable improvement.

3.8 Chapter summary

A novel and effective approach to handle boundary conditions with arbitrary high order of accuracy is proposed to solve compressible flow problems with curved domains discretized through DG schemes on simple linear meshes. The proposed strategy relies on the shifted boundary method that allows to overcome the second-order geometrical error due to the



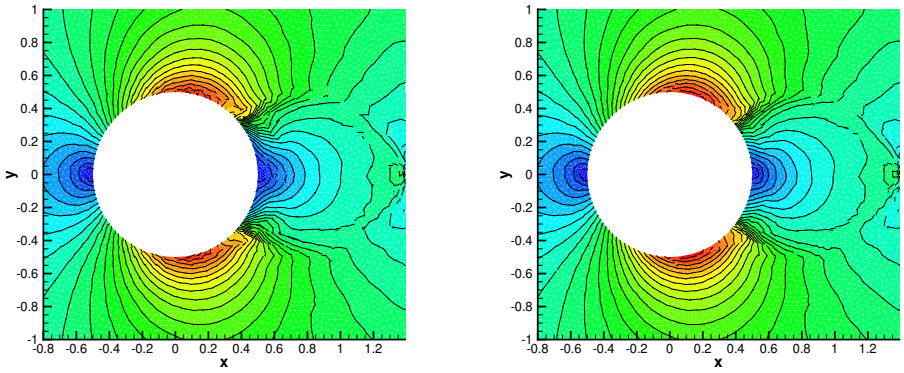
(a) $t = 0.5$ (b) $t = 0.5$

(c) $t = 1$ (d) $t = 1$

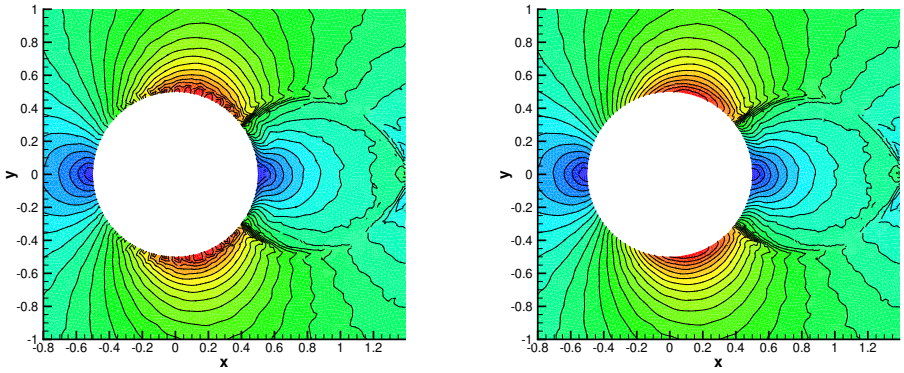
(e) $t = 1.5$ (f) $t = 1.5$

(g) $t = 2$ (h) $t = 2$

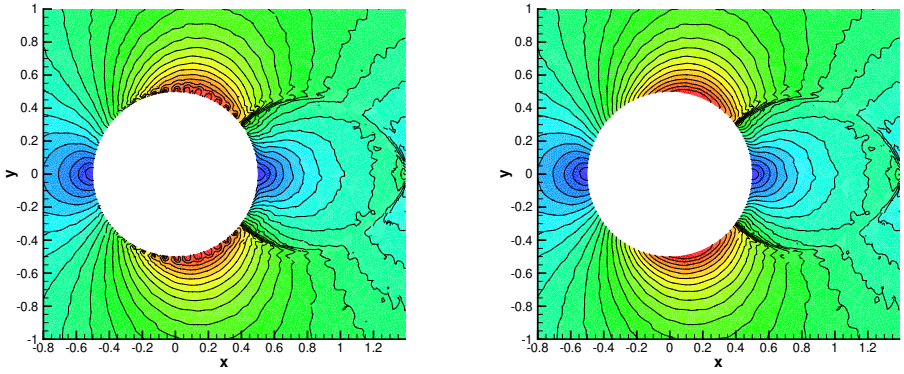
Figure 3.12: Shock-cylinder interaction (2D): Mach number iso-contours and troubled cells at different time steps (simulation run with DG-$\mathbb{P}3$/SBM-$\mathbb{P}3$).



(a) DG-$\mathbb{P}1$ (b) DG-$\mathbb{P}1$/SBM-$\mathbb{P}1$



(c) DG-$\mathbb{P}2$ (d) DG-$\mathbb{P}2$/SBM-$\mathbb{P}2$



(e) DG-$\mathbb{P}3$ (f) DG-$\mathbb{P}3$/SBM-$\mathbb{P}3$

Figure 3.13: Shock-cylinder interaction (2D): Mach number iso-contours at the final time $t = 2$ (simulation run with several schemes).

inconsistent treatment of curved boundaries, thanks to a polynomial correction embedded in the boundary flux. In particular, the only major problem coming with the high order formulation of the SBM is the computation of high order partial derivatives of the basis functions. But, within this work, this cumbersome procedure has been considerably simplified by rewriting the boundary conditions directly as a function of the DG polynomials evaluated on both the real and the surrogate boundary. The new approach has been tested over a large set of benchmarks in 2D and 3D and with both steady and unsteady flows. The formal order of accuracy provided by the employed DG-$\mathbb{P}N$ schemes has been numerically retrieved in all the performed test cases, and furthermore, the boundary corrections have also been coupled with the *a posteriori* sub-cell FV limiter, thus allowing the effective simulation of shocks and discontinuities.

Chapter 4

Unstructured Discontinuity Fitting (UnDiFi)

In this chapter, we present a new open-source CFD tool for shock-fitting simulations on unstructured grids. Shock-fitting techniques consists in explicitly identifying discontinuities of the flow-field as lower dimensional manifolds, whose motion is described by the Rankine-Hugoniot equations. Contrary to classical shock-capturing methods, shock-fitting allows a better modeling of discontinuities resulting in high order of accuracy. However, its limited use is mainly due to the difficulties related to the code development behind these methods. The recent rising interest towards these approaches [310, 107, 303, 181] for simulating compressible flows gave rise to the idea of making our shock-fitting code [68], whose results have been presented for the first time in [236], publicly accessible in order to further promote its collaborative development. The current repository includes the algorithmic features and test-cases published in a series of papers [176, 238, 44, 46, 69]. The sections deepen the algorithmic steps of the code, the characteristics of the repository and few results focusing on the test-cases that better represent all the implemented features. The algorithm has been coupled with two second-order residual distribution (for an introduction to these methods see Section 2.4 and references therein) codes. The code can be downloaded in the open-source repository made available at `https://github.com/UnDiFi/UnDiFi-2D`. At present, only the two-dimensional inviscid version of the algorithm has been released, because it is the one that has been most actively developed over the years and has reached a sufficient level of maturity and generality. However, old and ongoing developments on different extension of UnDiFi [245, 246, 22, 90, 43] may be added in future releases of the code.

4.1 *UnDiFi-2D*: directory tree

The main directory `UnDiFi-2D` contains the following sub-directories:

1. `bin`: where all the executables are installed;
2. `lib`: where various libraries and their source codes are stored;
3. `doc`: which contains the documentation;
4. `source`: where the source files of the *UnDiFi-2D* code are stored;
5. `source_utils`: contains
   - the source files of various I/O format converters;
   - the `Triangle` [273, 272] mesh-generator;
6. `tests`: contains the various test-cases described in Section 4.3;
7. `tools`: contains the source code of the `f77split` and `f90split` programs [1];
8. `EulFS.3.7`: where the source files of the `EulFS` [42, 49] gas-dynamic solver are stored.

9. `NEO`: where the source files of the `NEO` solver [20, 260, 259] are stored;

In addition to these sub-directories, the main directory contains the script (`compile_all.sh`) which compiles all the software packages.

Full description on how to download, compile and run the code can be found at the documentation page `https://github.com/lkampoli/UnDiFi/index.html`.

The directory tree highlights the fact that the software is made up of three key components:

1. the shock-fitting module *UnDiFi-2D* which handles the motion of the discontinuities and their interactions (if any), but also drives the other two components, i.e.

2. the gas-dynamic solver, either `EulFS` or `NEO`, which is used to discretize the governing PDEs in smooth regions of the flow-field;

3. the meshing software `Triangle`, which is used to locally re-mesh while the discontinuities move throughout the computational domain.

Communication among the driver *UnDiFi-2D*, the gas-dynamic solver and the meshing software is handled using format converters (to be found in the `source_utils` folder) that rely on disk I/O. This programming approach is certainly not the best from the standpoint of computational efficiency, one of the reasons being that one has to switch among the different data-structures used by the three different modules. However, this approach is very convenient, since it allows us to use off-the-shelf gas-dynamic solvers and mesh generation tools that are treated as black boxes and can be replaced by similar ones only by changing the format converters, with a modest coding effort.

4.2 Unstructured shock-fitting: algorithmic features

Figure 4.1 shows the algorithmic workflow of the *UnDiFi-2D* code, including the sequence of subroutines and external programs being called during a typical run. The CFD codes (either `NEO` or `EulFS`) and the `Triangle` mesh-generator are invoked as black boxes, communication being handled through disk I/O, see the 1 and 5 circles in Figure 4.1. Figure 4.1 also includes those optional parts that ensure the time-accurate integration (circled points 2, 3, 4).

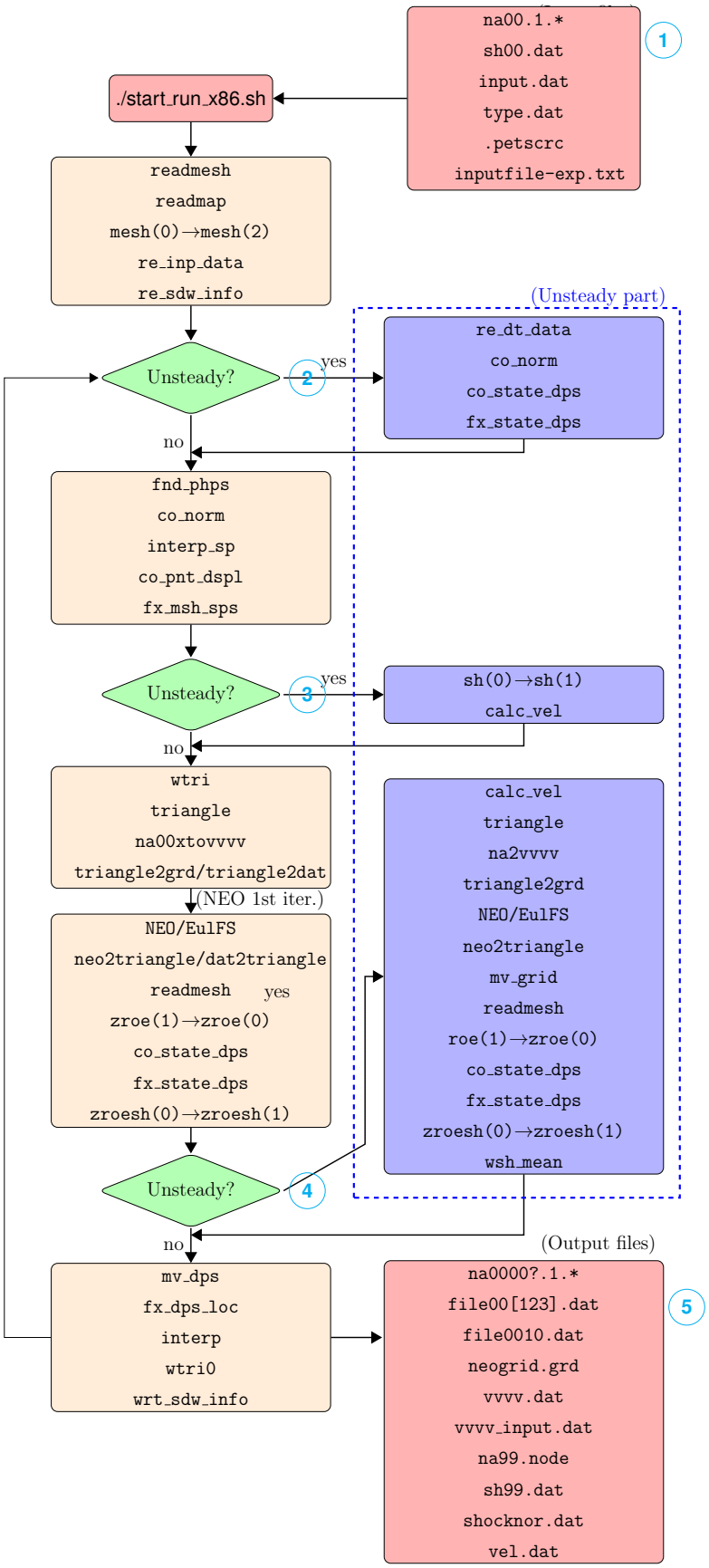


Figure 4.1: Typical *UnDiFi-2D* workflow.

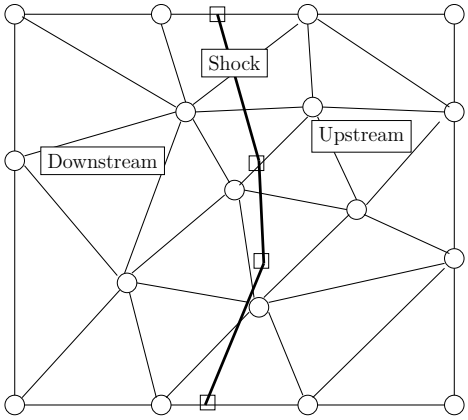
Even though a thorough description of the various procedures listed in Figure 4.1 is available in the *UnDiFi-2D* documentation, it is appropriate to give a brief description of the data storage and key algorithmic ingredients of the shock-fitting algorithm. Regardless of whether steady or time-accurate simulations are performed, the approach is inherently time-dependent, because both the solution and the grid change with time, due to the displacement of the fitted discontinuities. When a steady solution exists, the shock speed will asymptotically vanish and the tessellation of the flow domain will not any longer change.

As far as data storage is concerned, the dependent variables and grid velocity vector are available within all grid-points of a two-dimensional triangulation that covers the entire computational domain; this is what we call the *background* mesh. In addition to the background mesh, the fitted discontinuities (either shocks or slip-lines) are discretized using a collection of grid-points (the shock-points) which are mutually joined to form a connected series of line segments (the shock-edges); shock-points and shock-edges make up what we call the *shock*-mesh. In contrast to the grid-points of the background mesh, where a single set of dependent variables is stored, the shock-points are duplicated items that share the same geometrical location, but store two different sets of dependent variables, corresponding to the two sides of the discontinuity. This is schematically shown in Figure 4.2d. Shock-edges, which connect the shock-points on both sides of the discontinuity (see Figure 4.2d where the width of the discontinuity has been increased to improve readability) also overlap, so that each fitted discontinuity behaves like a double-sided internal boundary of zero thickness. As shown in Figure 4.2a, the spatial location of the fitted discontinuities is independent of the location of the grid-points that make up the background grid.

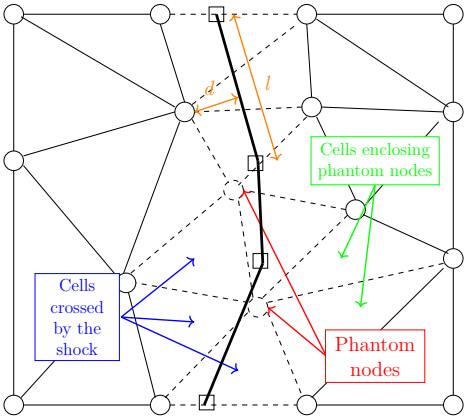
The sequence of operations that leads from the available mesh and solution at time t^n to an updated mesh and solution at time $t^{n+1} = t + \Delta t$ can be split into the seven steps that will be described in Sections 4.2.1 to 4.2.7.

4.2.1 Cell Removal Around the Shock Front

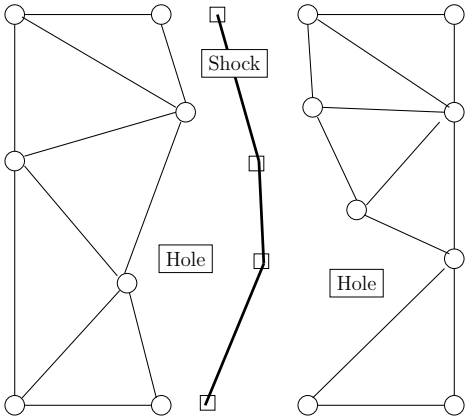
In this first step, the fitted discontinuities are laid on top of the background mesh, as shown in Figure 4.2a. All those cells that are crossed by the fitted discontinuities and those mesh points that are located too close to it are temporarily removed from the background mesh, as shown in Figure 4.2b. This operation is performed in the subroutine `fnd_phps`, whose interface is shown in Listing 4.1. Here, the arguments referring to the background mesh are passed with index zero. We call phantom those grid-points of the background mesh (shown using dashed circles in Figure 4.2b) that have been temporarily removed. All



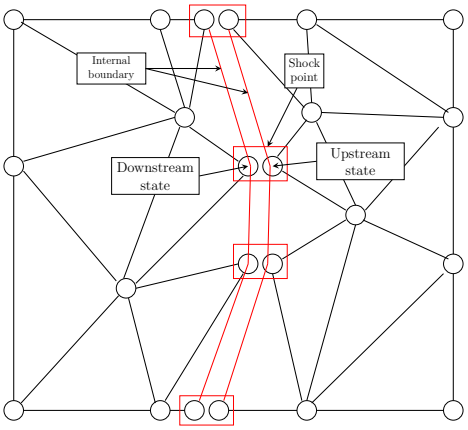
(a) Shock front moving over the background triangular mesh at time t.



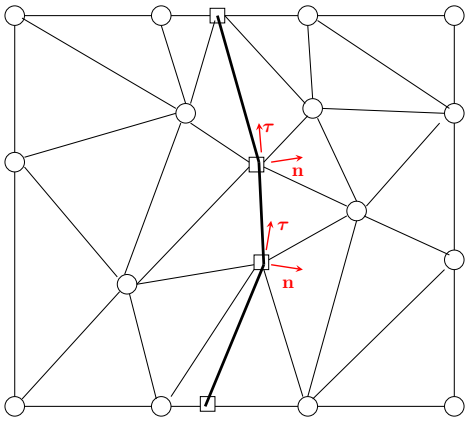
(b) Dashed lines mark the cells to be removed; dashed circles denote the phantom nodes.



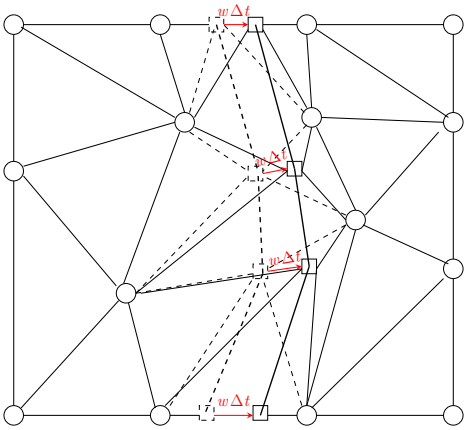
(c) The background mesh is split into disjoint sub-domains by a hole which encloses the shock.



(d) The triangulation around the shock is rebuilt.



(e) Calculation of the shock-tangent and shock-normal unit vectors.



(f) The shock displacement induces mesh deformation.

Figure 4.2: Unstructured shock-fitting: summary of the key algorithmic steps.

cells having at least one phantom node among their vertices are also removed from the background triangulation; these are the cells shown using dashed edges in Figure 4.2b. Further details concerning the criteria used to identify and remove the phantom nodes can be found in [236].

Listing 4.1: Cell removal around the shock front.

```
call fnd_phps(
.    nedge(0),           ! number of faces
.    istak(lbndfac(0)),  ! boundary faces index pointer
.    nbfac(0),           ! number of boundary faces
.    istak(lcelnod(0)),  ! cell to node index pointer
.    nvt,                ! number of vertices of a cell
.    nelem(0),           ! number of elements of the background grid
.    dstak(lcorg(0)),    ! point coordinates
.    xysh,               ! shock point coordinates
.    istak(lnodcod(0)),  ! node code flag
.    npoin(0),           ! number of points of the background grid
.    istak(lnodptr(0)),  ! node index pointer
.    nbpoin(0),          ! number of boundary points
.    nshocks,            ! number of shocks
.    nshockpoints,       ! number of shock points
.    nshocksegs,         ! number of shock segments
.    nphanpoints,        ! number of phantom points
.    istak(lpmap(0)))    ! pointer to integer array
```

It is important to underline that all the quantities that refer to the background or the computational mesh (such as the coordinates of the grid-points) are stored inside a large double precision array (`dstak`) where integers (`istak`) can also be stored by using an `EQUIVALENCE` statement. Therefore, when calling the various subroutines, these quantities are passed providing the corresponding pointers inside the `dstak` array; for example: `dstak(lcorg(0))` represents the pointer to the coordinates of the background mesh. On the contrary, the quantities that refer to the discontinuities (for example the coordinates of the shock-points of the shock-mesh) are stored in specific arrays; for example: `xysh` contains the coordinates of the shock-points.

4.2.2 Local Re-Meshing Around the Shock Front

Following the cell removal step, the background triangulation has been split into two or more disjoint sub-domains, as shown in Figure 4.2c. The hole dug by the fitted front is then re-meshed using a Constrained Delaunay Tessellation (CDT): the edges that make up the fitted discontinuity and the boundary of the hole are both constrained to be part of the final tessellation; this is illustrated in Figure 4.2d. This operation is performed in the subroutines `fx_msh_sps`, `wtri` and by calling `Triangle` whose interfaces are shown

in Listings 4.2, 4.3 and 4.4. Observe that re-meshing should be localized around the discontinuities. The code currently included in the repository is not fully compliant with the algorithm described above, because the CDT is applied to the entire computational domain and not only within the hole carved around the shock. Future releases will fix this open issue. Upon completion of this stage, the computational domain is discretized using what we call the *computational* mesh, which differs from the background mesh only in the neighborhood of the fitted discontinuities. Further details concerning the software used to construct the CDT have been given in Section 4.1.

Listing 4.2: Local fix of special points.

```
call fx_msh_sps(
.    istak(lbndfac(0)), ! boundary faces index pointer
.    istak(lnodcod(0)), ! node code flag
.    nbfac(0),          ! number of boundary faces
.    nbfac_sh,          ! number of shock faces
.    nvt,               ! number of vertices of a cell
.    nelem(0),          ! number of elements of the background grid
.    dstak(lcorg(0)),   ! point coordinates
.    xysh,              ! shock point coordinates
.    dstak(lcorg(0)+npoin(0)*ndim),                      ! upstream
.    dstak(lcorg(0)+npoin(0)*ndim+nshmax*npshmax*ndim), ! downstream
.    npoin(0),          ! number of points of the background grid
.    nshocks,           ! number of shocks
.    nshockpoints,      ! number of shock points
.    nshocksegs,        ! number of shock  segments
.    nspecpoints,       ! number of special points
.    typespecpoints,    ! type of special points
.    shinspps,          ! special points in shock
.    ispclr)            ! special points color
```

Listing 4.3: Local re-meshing around the shock front.

```
call wtri(
.    istak(lbndfac(0)), ! boundary faces index pointer
.    nbfac(0),          ! number of boundary faces
.    nbfac_sh,          ! number of shock faces
.    istak(lcelnod(0)), ! cell to node index pointer
.    nvt,               ! number of vertices of a cell
.    dstak(lcorg(0)),   ! point coordinates
.    xysh,              ! shock point coordinates
.    dstak(lcorg(0)+npoin(0)*ndim),                      ! upstream coor.
.    dstak(lcorg(0)+npoin(0)*ndim+nshmax*npshmax*ndim), ! downstream coor.
.    dstak(lzroe(0)),                                    ! roe variables
.    dstak(lzroe(0)+npoin(0)*ndof),                      ! upstream state
.    dstak(lzroe(0)+npoin(0)*ndof+nshmax*npshmax*ndof), ! downstream state
.    istak(lnodcod(0)),          ! upstream node code flag
.    istak(lnodcod(0)+npoin(0)), ! downstream node code flag
.    npoin(0),          ! number of points of the background grid
.    fname(1:7),        ! mesh input file name to Triangle
.    nshocks,           ! number of shocks
.    nshockpoints,      ! number of shock points
```

```
.     nshocksegs,         ! number of shock segments
.     nphanpoints)        ! number of phantom points
```

Listing 4.4: Mesh generation with Triangle.

```
write(*,1001,advance='no')'triangle --> '
execmd = bindir(1:10)//'triangle_'//hostype(1:6)//
.         ' -nep '//fname(1:7)//' > log/triangle.log'
ifail = system(execmd)
call flush(6)
if (ifail /= 0) then
  write(6,*)'Triangle has returned an error code ifail = ', ifail
  call exit(ifail)
endif
write(*,1002)' ok'
```

4.2.3 Calculation of the Unit Vectors Normal to the Shock Front

In order to apply the jump relations, normal $\mathbf{n}$ and tangent $\boldsymbol{\tau}$ unit vectors are needed within each pair of grid-points located along the discontinuities, see Figure 4.2e. These unit vectors are computed using finite-difference (FD) formulae which involve the coordinates of the shock-point itself and those of its neighboring shock-points. This operation is performed in the subroutine `co_norm` whose interface is shown in Listing 4.5. Depending on the local, shock-downstream flow regime, it may be necessary to use upwind-biased formulae to avoid the appearance of geometrical instabilities along the fitted discontinuity. Full details describing how to select the stencil can be found in [236]. Be aware that the FD formulae reported in [236] contain a typo and should be replaced by those published in [89].

Listing 4.5: Calculation of the unit vectors normal to the shock front.

```
call co_norm(
.     xysh,               ! shock point coordinates
.     dstak(lzroe(0)+npoin(0)*ndof),                        ! upstream
.     dstak(lzroe(0)+npoin(0)*ndof+nshmax*npshmax*ndof), ! downstream
.     norsh,              ! normal vectors
.     nshocks,            ! number of shocks
.     nshockpoints,       ! number of shock points
.     typeshocks,         ! type of shock
.     nspecpoints,        ! number of special points
.     typespecpoints,     ! type of special points
.     shinspps,           ! special points in shock
.     ispclr,             ! special points color
.     istak(lia(0)),      ! pointer for integer arrays in setbndrynodeptr
.     istak(lja(0)),      ! pointer for integer arrays in setbndrynodeptr
.     istak(liclr(0)),    ! pointer for integer arrays in setbndrynodeptr
.     nclr(0),            ! colours of boundary patches
.     dstak(lcorg(0)))    ! point coordinates
```

4.2.4 Solution Update Using the Shock-Capturing Code

Using the computational mesh as input, a single time step calculation is performed using one of the available shock-capturing solvers which returns updated nodal values at time $t+\Delta t$, as shown in Listing 4.6 for the `EulFS` solver. Since the discontinuities are seen by the shock-capturing code as internal boundaries (of zero thickness) moving with the velocity of the discontinuity, there is no need to modify the spatial discretization scheme already implemented in the PDEs solver to account for the presence of the fitted discontinuities. In practice, the shock-capturing solver is used as a black-box: it receives in input the computational grid, the nodal values of the solution and grid velocity at time t and returns the updated solution at time $t + \Delta t$. The solution returned by the shock-capturing solver at time $t + \Delta t$ is however missing some boundary conditions. These missing pieces of information will be determined as described in Section 4.2.5.

Listing 4.6: Call to the gasdynamic solver.

```
execmd = bindir(1:10)//"eulfs_"//hostype(1:6)//"-itmax 1 > log/eulfs.log"
ifail = system(execmd)
call flush(6)
if (ifail /= 0) then
  write(6,*) "Eulfs has returned an error code ifail = ", ifail
  call exit(1)
endif
```

4.2.5 Enforcement of the Jump Relations

The missing pieces of information that are needed to correctly update the solution within all pairs of grid-points located along the discontinuities are obtained by enforcing the Rankine-Hugoniot jump relations; this also provides the local velocity of the discontinuity along its normal. The jump relations are a set of non-linear algebraic equations that can be solved within each pair of grid-points located along the discontinuities by means of Newton-Raphson's algorithm. In order to match the number of unknowns with the available equations, one or more additional pieces of information are required within both or either of the two sides of the fitted discontinuity, depending on whether this is a shock or a contact discontinuity. These additional pieces of information are obtained from the characteristic formulation of the Euler equations and correspond to those characteristic quantities that are convected towards the discontinuity from the sub-domain that is attached to that side of the discontinuity. Using an upwind-biased discretization within the shock-capturing solver, one can reasonably assume that the spatial and temporal evolution of these characteristic quantities has been correctly computed.

The enforcement of the jump relations is performed in the subroutine `co_state_dps`

whose interface is shown in Listing 4.7. Full algorithmic details concerning the practical implementation of the jump relations for shocks and contact discontinuities are reported elsewhere [176, 236, 238] and will not be repeated here. Furthermore, an ad-hoc treatment is required within those special points where different discontinuities interact (triple or quadruple points) or whenever a discontinuity interacts with a solid or free boundary. The algorithmic details are described in [176, 238] and their implementation can be found in the subroutines `co_utp`, `co_uqp`, respectively.

Listing 4.7: Enforcement of the jump relations.

```
call co_state_dps(
.    xysh,           ! shock point coordinates
.    dstak(lzroe(0)+npoin(0)*ndof),                      ! upstream
.    dstak(lzroe(0)+npoin(0)*ndof+nshmax*npshmax*ndof), ! downstream
.    zroeshuold,     ! upstream zroe states at previous iteration
.    zroeshdold,     ! downstream zroe states at previous iteration
.    norsh,          ! normal vectors
.    wsh,            ! shock velocity
.    nshocks,        ! number of shocks
.    nshockpoints,   ! number of shock points
.    nshocksegs,     ! number of shock segments
.    typeshocks,     ! type of shock
.    iter)           ! current iteration
```

4.2.6 Shock displacement

The enforcement of the jump relations provides the speed, w, at which each pair of grid-points located on the discontinuity move along its local normal unit vector, $\mathbf{n}$. The position of the discontinuity at time $t^{n+1} = t^n + \Delta t$ is computed in a Lagrangian manner by displacing all its grid-points, as shown in Figure 4.2f where the dashed and solid lines represent the discontinuity at time t^n, respectively t^{n+1}. When simulating steady flows, this can be accomplished using the following first-order-accurate (in time) integration formula

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + w_i^n \mathbf{n}_i^n \Delta t, \tag{4.1}$$

which returns the spatial coordinates of the i-th shock-point at time t^{n+1}. The low temporal accuracy of Equation (4.1) does not affect the spatial accuracy of the steady state solution which only depends on the spatial accuracy of the gas-dynamic solver and that of the tangent and normal unit vectors.

On the contrary, when dealing with unsteady flows, the temporal accuracy of the shock motion has to be the same as that of the spatial discretization, i.e. second order accurate in our case. This can be accomplished using a predictor-corrector type temporal

integration scheme, or a Runge-Kutta multi-step scheme. In the former case, the predictor step estimates the position of the discontinuity at time level $n + 1/2$ using the explicit Euler scheme:

$$\mathbf{x}_i^{n+\frac{1}{2}} = \mathbf{x}_i^n + w_i^n \mathbf{n}_i^n \frac{\Delta t}{2}. \tag{4.2}$$

The speed of the discontinuity $w_i^{n+\frac{1}{2}}$ and the normal unit vector $\mathbf{n}_i^{n+\frac{1}{2}}$ at time level $n+1/2$ are then computed using the intermediate position of the discontinuity $\mathbf{x}_i^{n+\frac{1}{2}}$ and, finally, the position of each shock-point is updated at time level $n + 1$ in the corrector step:

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + w_i^{n+\frac{1}{2}} \mathbf{n}_i^{n+\frac{1}{2}} \Delta t. \tag{4.3}$$

This operation is performed in the subroutine `mv_dps` whose interface is shown in Listing 4.8.

Listing 4.8: Shock displacement.

```
call mv_dps(
.    xysh,           ! shock point coordinates
.    dstak(lzroe(0)+npoin(0)*ndof+nshmax*npshmax*ndof), ! downstream
.    wsh,            ! shock velocity
.    i,              ! current iteration
.    nshocks,        ! number of shocks
.    nshockpoints,   ! number of shock points
.    nshocksegs,     ! number of shock segments
.    typeshocks)     ! type of shock
```

One further observations is in order concerning the discontinuity displacement step. Figure 4.2f shows that even when the background mesh is fixed in space, the triangular cells that abut on the discontinuity have one of their edges that moves with the discontinuity, thus deforming the cell. This implies that the shock-capturing solver used in Step 4.2.4 must be capable of handling moving meshes, i.e. it must be capable of solving the governing PDEs written using an Arbitrary Eulerian Lagrangian (ALE) formulation.

4.2.7 Interpolation of the Phantom Nodes

Upon completion of the previous steps, all grid-points of the computational mesh have been updated at time $t + \Delta t$. The computational mesh is made up of shock-mesh and all grid-points of the background mesh, except those that have been declared *phantom*. Therefore, the nodal values within the phantom nodes have not been updated to time $t + \Delta t$. However, during the current time step, the discontinuities might have moved sufficiently far away from their previous position, that some of the phantom nodes may re-appear in the computational mesh at the next time step. It follows that also the nodal

values within the phantom nodes need to be updated to time $t + \Delta t$. This is easily accomplished by transferring the available solution at time $t + \Delta t$ from the current computational mesh to the grid-points of the background one, using linear interpolation. This operation is performed in the subroutine `interp` whose interface is shown in Listing 4.9. Once the phantom nodes have been updated, the computational mesh used in the current time interval has completed its task and can be discarded. At this stage the numerical solution has correctly been updated at time $t+\Delta t$ within all grid-points of the background and shock meshes.

The next time interval can be computed re-starting from step 4.2.1 of the algorithm.

Listing 4.9: Interpolation of the phantom nodes.

```
call interp(
.    istak(lbndfac(1)), ! boundary faces index pointer
.    nbfac(1),          ! number of boundary faces
.    istak(lcelnod(1)), ! cell to node  index  pointer
.    nvt,               ! number of vertices of a cell
.    nelem(1),          ! number of elements of the shocked grid
.    dstak(lcorg(1)),   ! point coordinates
.    dstak(lzroe(1)),   ! point roe states
.    xysh,              ! shock point coordinates
.    dstak(lcorg(1)+npoin(0)*ndim),                     ! upstream
.    dstak(lcorg(1)+npoin(0)*ndim+nshmax*npshmax*ndim), ! downstream
.    nphampoints,       ! number of phantom points
.    dstak(lcorg(0)),   ! background point coordinates
.    dstak(lzroe(0)),   ! background points roe states
.    istak(lnodcod(0)), ! background grid node code flag
.    npoin(0),          ! number of points of the background grid
.    nshocks,           ! number of shocks
.    nshockpointsold,   ! old number of shock points
.    nshockpoints,      ! new number of shock points
.    istak(lia(1)),     !pointer for integer arrays in setbndrynodeptr
.    istak(lja(1)),     ! pointer for integer arrays in setbndrynodeptr
.    istak(liclr(0)),   ! pointer for integer arrays in setbndrynodeptr
.    nclr(0))           ! colours  of  boundary  patches
```

4.3 Applications

In order to illustrate the capabilities of *UnDiFi-2D* several test-cases can be run within each of the sub-directories (listed in Figure 4.3) of the folder `tests`. In addition to this, these test-cases are intended to test different parts of the code and, therefore, the successful execution of all test-cases (which can be performed with the script `run_all_x86.sh`) represents a verification of the code. These checks are particularly important before a new version of the code is released to verify that changes and modifications of the code have not produced unwanted side-effects elsewhere in the code.

More specifically, in the shock-shock or shock-wall interaction test-cases (with uniform states), the various discontinuities bound regions of uniform flow, where an analytical solution can be computed using the jump relations. Therefore, these test-cases provide a powerful code validation tool, because the shock-fitting algorithm is expected to return the exact solution everywhere, even if a first-order accurate discretization of the governing PDEs is used.

An analytical solution is also available for the transonic test-case `Q1D` which features spatially variable fields both upstream and downstream of the shock, so that it can be used to measure the order-of-convergence of the spatial discretization.

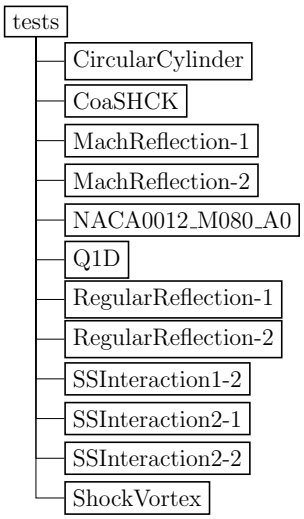


Figure 4.3: Test-cases available in the `tests` directory.

Among the available test-cases, we only show the most significant ones:

- `CircularCylinder-1`,
- `MachReflection-1, MachReflection-2`,
- `ShockVortex`.

4.3.1 Hypersonic flow past a Circular cylinder (`CircularCylinder-1`)

The first test-case deals with the hypersonic flow past a circular cylinder at free-stream Mach number, $M_\infty = 20$. It was one of the first flow configurations to be set-up and tested during the *UnDiFi-2D* code development, because this apparently simple test-case allows to check the basic subroutines of the code, such as those in charge of calculating the shock normal unit vectors and applying the Rankine-Hugoniot relations, under all

Table 4.1: Nodes and cells of the background and computational meshes

Background Mesh		Computational mesh at steady-state		
Triangles	Nodes	Triangles	Nodes	Shock nodes
610	351	624	358	29

possible post-shock conditions. This is because by moving along the bow shock is like sweeping the entire shock polar, starting from a normal shock at the symmetry axis, which then turns into a strong oblique shock and finally becomes a weak wave. As shown in Figure 4.4, the computational domain surrounds the fore half of a circular cylinder having radius $R = 1$. The background mesh has been created using the `delaundo` [225] mesh generator by specifying a evenly spaced distribution of boundary nodes with spacing $h = 0.08\,R$. The numbers of triangles and nodes of the background mesh are reported in Table 4.1. The solution computed on the background mesh using the unstructured code in shock-capturing mode has been used to initialize the flow-field and to determine the parabolic shape of the initial position of the shock front, see Figure 4.4. In the shock-fitting simulation, the initial upstream state in the shock-points has been set equal to the free-stream conditions, while the initial shock-downstream state has been computed from the upstream state and the local shock slope, assuming zero shock speed: $w = 0$. The flow-field and shock position are then integrated in pseudo-time until steady-state is reached. Figure 4.4a shows the shock displacement that occurs between the initial shock-position and the one reached at steady state. During the computation, the number of grid-points and triangles of the computational mesh varies with respect to that of the background mesh, as shown in Table 4.1. Figure 4.4b also displays the background mesh and the computational mesh at steady-state. The smaller frame of Figure 4.4b, which shows an enlargement of the near-shock region, allows comparing the two meshes. It can be seen that these are superimposed everywhere except in the region adjacent to the shock, where the differences between the background (dashed lines) and the computational mesh (solid lines) are due to the addition of the shock-points and shock-edges. Figure 4.4b, as well as Table 4.1, clearly show that the re-meshing technique does not significantly increase the number of grid-points and triangles with respect to those of the background mesh. Figure 4.4c allows to compare the solutions computed by the `EulFS` code working in shock-capturing and shock-fitting mode; similar results can be obtained using `NEO`. Inside this folder (as well as the other folders that will be described below) there is a script (run.sh) that allows to run the shock-capturing or shock-fitting solutions with either `EulFS` or `NEO`. Figure 4.4c also includes a detailed view of the stagnation point region showing the modifications introduced by the shock-fitting algorithm to the background mesh in order to take into account the presence of the shock front. A detailed analysis of the numerical

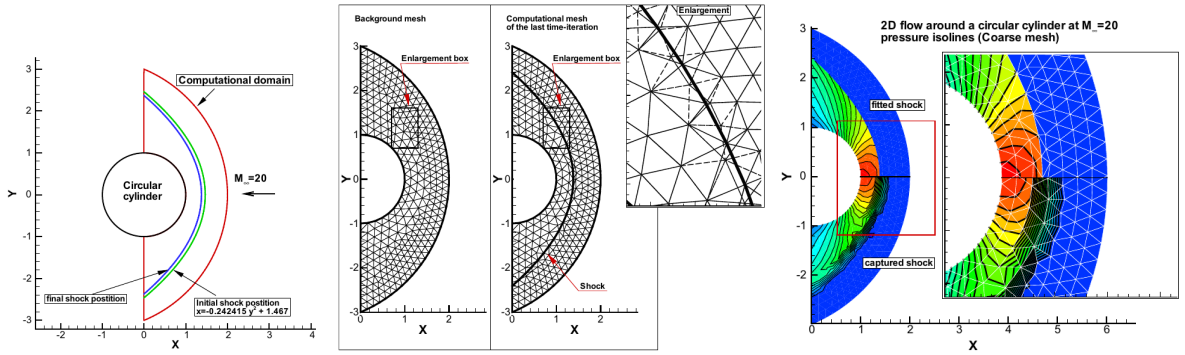


Figure 4.4: Computational domain: initial and final shock position. Comparison between the coarse background and the computational mesh at steady-state. Comparison between the shock-fitting and shock-capturing solutions.

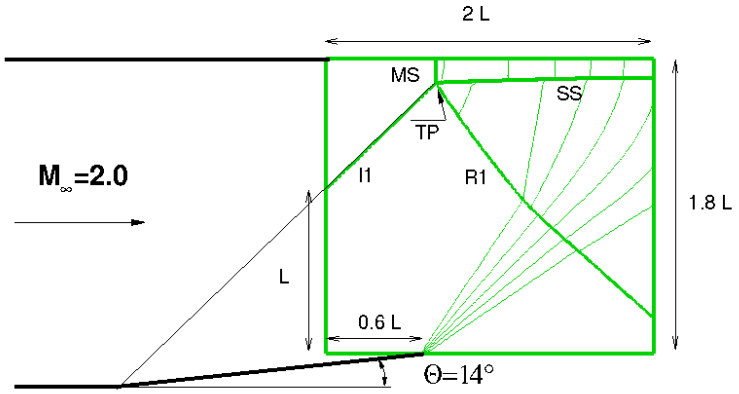


Figure 4.5: Steady Mach reflection: flow configuration.

results of this test-case can be found in [236].

4.3.2 Steady Mach reflection (`MachReflection-1, MachReflection-2`)

Whenever the flow deflection imposed by a solid surface on an impinging weak oblique shock is larger than the maximum allowable deflection for the downstream Mach number, a so-called Mach reflection takes place instead of the regular reflection. This is schematically shown in Figure 4.5: a uniform, supersonic ($M_\infty = 2$) stream of air undergoes a $\Theta = 14°$ deflection through the incident shock, I1. Regular reflection of I1 is however impossible for the chosen pair of M_∞, Θ parameters, so that a steady triple-point (TP) arises which joins I1, the reflected shock (R1), the Mach stem (MS) and the slip-stream (SS).

This test-case has been split into two different simulations: the `MachReflection-1` directory contains the hybrid simulation, whereas a fully-fitted simulation can be run in

the `MachReflection-2` directory. In the former case the incident shock and the slip-stream are captured, whereas the reflected shock and the Mach stem are fitted as a single shock. In the latter case all discontinuities, as well as the triple-point, are fitted. From the viewpoint of code-checking the two simulations are very different. The hybrid simulation only demonstrates the capability of the algorithm to run in hybrid mode, but tests the same functionalities already checked in the circular cylinder simulation, with the only addition of the interaction between a normal shock (the MS) and a flat wall. On the contrary, the fully fitted simulation tests the capability of the `UnDiFi-2D` code to fit the triple point.

The computational domain used for both simulations has been marked in green in Figure 4.5. Inside this area a background grid of almost equilateral triangles has been generated using the `delaundo` [225] mesh generator by specifying a uniform distribution of grid-points along the domain boundaries with spacing $h = 0.0167\,L$, see Figure 4.5. Table 4.2 reports the number of triangles and grid-points of the background mesh along with those of the computational meshes at steady-state for both the hybrid and the fully-fitted simulations; the corresponding number of shock-points is also shown.

Table 4.2: Grid-points and triangles of the background and computational meshes.

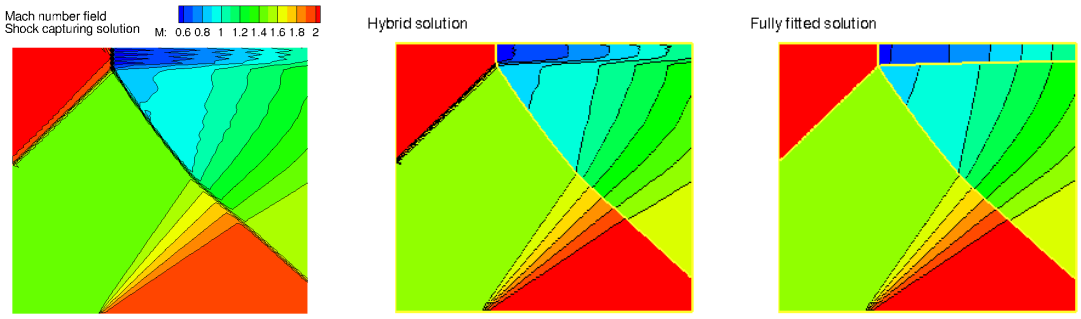
Background Mesh		Computational meshes at steady-state					
		Hybrid simulation			Fully fitted simulation		
Cells	Nodes	Cells	Nodes	Shock-points	Cells	Nodes	Shock-points
29214	14833	29292	17633	142	29365	19633	294

Shock-capturing, hybrid and fully-fitted calculations obtained using the two different Residual Distribution codes, `NEO` and `eulfs`, are compared in Figure 4.6. Even though the two solvers use very similar numerical recipes, it can be seen that the two shock-capturing solutions, Figures 4.6a and 4.6d, exhibit non-negligible differences, in particular downstream of the Mach stem. These differences are significantly reduced in the hybrid simulations, Figures 4.6b and 4.6e, and have almost disappeared in the fully fitted ones, Figs. 4.6c and 4.6f.

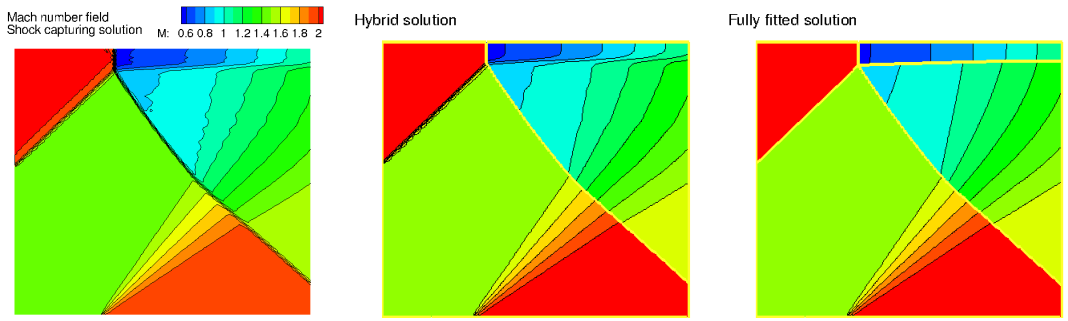
Further analyses about these numerical solutions can be found in [236, 238], whereas the algorithmic details concerning the treatment of the triple point are reported in [176].

4.3.3 Shock-vortex interaction (`ShockVortex`)

This last test-case, which is used to verify that *UnDiFi-2D* works correctly also when dealing with unsteady flows, features the interaction between a moving vortex and a standing shock. Figure 4.7 shows the computational domain along with the boundary



(a) `eulfs` shock-capturing (b) `eulfs` hybrid (c) `eulfs` fully fitted



(d) `NEO` shock-capturing (e) `NEO` hybrid (f) `NEO` fully fitted

Figure 4.6: Mach reflection: Mach number iso-contour lines computed using three different shock-modeling options and the two different shock-capturing codes.

and initial conditions. The flow-field is initialized by adding the perturbation velocity field induced by the vortex to the uniform flow past a steady normal shock. As shown in Figure 4.7, at the initial time $t = 0$, the vortex is located 0.2 unit lengths L ahead of the standing shock.

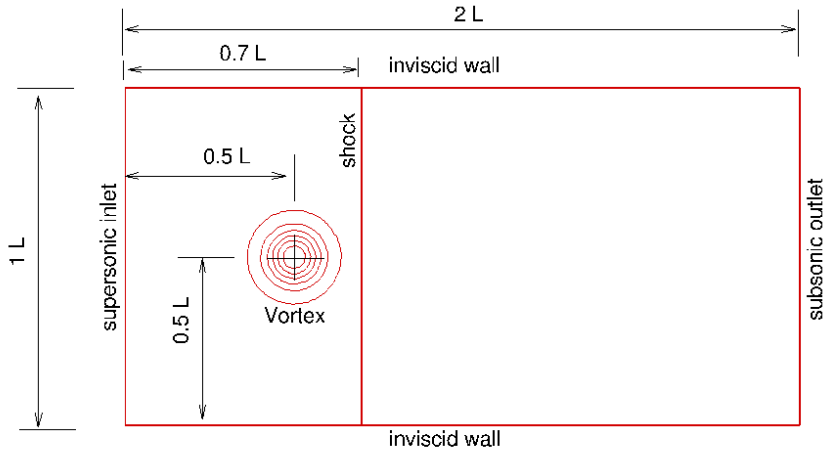


Figure 4.7: Shock–vortex interaction: computational domain and initial condition.

Using a cylindrical reference frame attached to the vortex core, the perturbation velocity field reads,

$$\widetilde{v}_\theta = -\epsilon|\mathbf{v}_\infty|\tau \exp^{\alpha\left(1-\tau^2\right)}, \tag{4.4a}$$

$$\widetilde{v}_r = 0, \tag{4.4b}$$

where the dimensionless radial distance from the vortex core, $\tau{=}\frac{r}{r_c}$, has been used in Equation (4.4a), with $r_c = 0.05\,L$. The two dimensionless parameters α and ϵ, which respectively control the width and magnitude of the velocity perturbation, are mutually related via the shock and vortex Mach numbers:

$$\epsilon = \frac{M_v}{M_s}\frac{\sqrt{2\alpha}}{\exp^{\left(\alpha-\frac{1}{2}\right)}}, \tag{4.5}$$

where

$$M_s = \frac{|\mathbf{v}_\infty|}{a_\infty}, \qquad M_v = \frac{\max|v_\theta|}{a_\infty}. \tag{4.6}$$

For the chosen pair of shock and vortex Mach numbers: $M_s = 2$, $M_v = 0.2$, which gives rise to a *weak* shock-vortex interaction, according to the nomenclature of [161], the vortex strength $\epsilon \approx 8.6\,10^{-2}$ follows from Equation (4.5), having set $\alpha = 0.204$. The Delaunay triangulation of the computational domain was generated using `Triangle`; it features

217569 grid-points and 433664 triangles and a mesh spacing along the boundaries equal to $h/L = 0.00375$. The shock-capturing and shock-fitting simulations were performed using the `NEO` solver. A qualitative comparison between the two shock-modeling options is given in Figure 4.8, where total enthalpy iso-contour lines at three subsequent time instants are shown: shock-capturing on the top row and shock-fitting on the bottom row. In particular, besides the oscillations related to the approximation of the shock, we can see clearly that the contours downstream of the discontinuity are much less smooth in the captured solution. The fitted computations, on the other hand, show very nice and smooth contours. Further details about this test-case and further simulations dealing with shock-vortex interactions can be found in [46, 69].

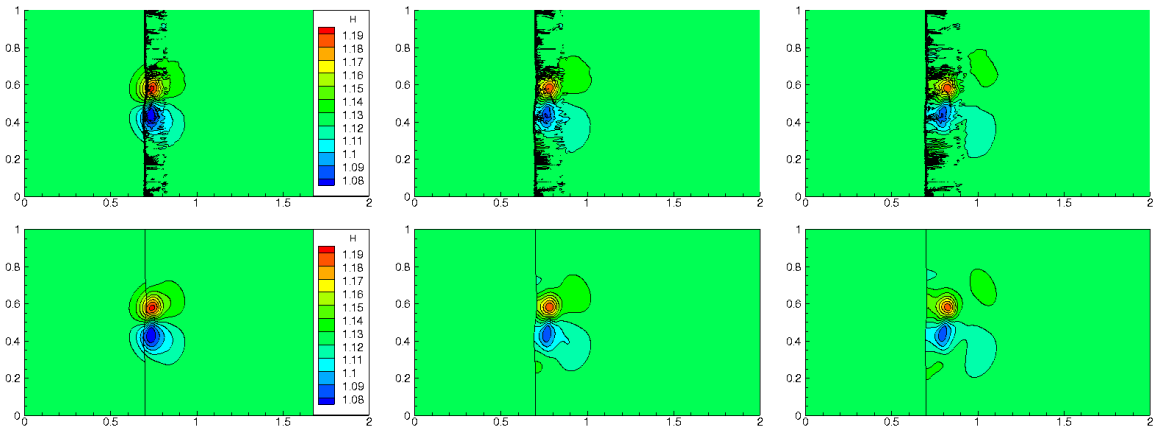


Figure 4.8: Shock-vortex interaction: total enthalpy contours at times $t = 0.3$ (left), $t = 0.4$ (center) and $t = 0.5$ (right) using the LDA scheme; shock-capturing in the upper row of frames and shock-fitting in the lower row.

4.4 Chapter summary

In this section we have presented the state-of-the-art of a 10-year-long development of a shock-fitting technique for unstructured meshes. We have described the key algorithmic features of the technique, which has been implemented in an open-source code, available in a dedicated repository. A fairly extensive selection of ready-to-run test-cases demonstrates the features and current capabilities of the code, which we have shown to be able to deal with compressible flows featuring either isolated or mutually interacting discontinuities. The superior quality of fitted shock-waves over captured ones has also been emphasized. A number of unsolved issues remain to be addressed, as detailed hereafter. When dealing with complex shock-shock and/or shock-boundary interactions in *steady* flows, a preliminary shock-capturing calculation, followed by an automatic shock-detection and shock-pattern identification strategy [240], is capable of supplying a reasonably good

initial guess to the shock-fitting algorithm.

When dealing with *un-steady* flows, however, things get much harder and the currently unsolved issues that remain to be addressed are well summarized in a 1986 paper by Glimm and co-workers [81]:

1. Treating changes of the topology of regions bounded by fronts from simply connected to multiply connected regions.
2. Treating the disappearance of weakening fronts and the appearance of new fronts at boundaries or at collisions of other fronts.

In order to be able to manage all these topological changes, it will be necessary to develop new algorithmic tools capable of detecting the occurrence of a change in the shock-topology and modify accordingly the fitted discontinuities and their mutual interactions. Some tests have been carried out on specific flow configurations, as shown in [52, 239], but the development of a general purpose tool will not be trivial and will probably require the use of advanced, multi-disciplinary techniques such as those used in [240]. The effort which will be necessary to complete the development of the unstructured, shock-fitting technique and bring it to full maturity, is not modest and probably requires the merging of different skills. In order to be successful, it will be necessary to broaden the audience of developers and technical expertise involved. With such a goal in mind, we launch the present project.

Chapter 5

Bridging Shock-Fitting and Embedded Boundary Methods

The goal of this chapter is that of presenting a novel approach for modeling discontinuities: the *extrapolated Discontinuity Tracking* (*eDIT*) method. The initial idea that motivated the present work comes from the similarity between the constraints arising from shock-fitting, as conceived by Paciorri and Bonfiglioli in [235], and those related to the construction of boundary-fitted grids for simulating flows around complex geometries. More precisely, the approach presented in [235] requires that at each time-step the computational grid is locally re-meshed to follow the moving shock-front. This approach turns out to be best suited to vertex-centered CFD solvers for unstructured grids, but it can be problematic if a different type of mesh data structure is used. Likewise UnDiFi [68], presented in Chapter 4, the eDIT method has been coupled with a second-order residual distribution solver (for an introduction to the methods see Section 2.4 and references therein). eDIT, which has been described in [89, 90], allows to overcome this limitation because it avoids any kind of re-meshing or adaptation phase by exploiting some ideas borrowed from immersed/embedded boundary methods, initially introduced to allow a flexible management of complex geometries. In particular, we borrowed the concept of the more recent Shifted Boundary Method (SBM) [206]. As in the latter, we impose modified conditions on surrogate shock-manifolds, acting as boundaries between the shock-upstream and shock-downstream regions. The values of the flow variables imposed on these surrogate boundaries are extrapolated from the tracked shock-front, accounting for the non-linear jump and wave propagation conditions, as done in the unstructured shock-fitting approach [235].

5.1 Extrapolated Discontinuity Tracking (eDIT)

To illustrate the algorithmic features of the eDIT method, let us consider a two-dimensional domain and a discontinuity front crossing the domain at a given time t^n. As shown in Figure 5.1(a), the front is described using a collection of edges whose endpoints are marked by squares. Throughout the chapter these two entities will be referred to as *front-edges* or *discontinuity-edges*, and *front-points* or *discontinuity-points* respectively, and together they constitute what will be referred to as *front-mesh* or *discontinuity-mesh*. The computational domain itself is discretized by means of a background mesh. As already said, we consider here solvers based on a nodal variable arrangement on triangular grids, but the extrapolation proposed readily applies to cell-centered solvers. Figure 5.1(a) clearly shows that the position of the discontinuity-points is completely independent of the location of the grid-points of the *background mesh*. Regardless of the method used to solve the Euler equations on the background mesh, the representation of the flow variables across the discontinuity is discontinuous. In our case, two values of the flow variables are stored in each front-point. The flow solvers considered here are based on a continuous

nodal approximation, which amounts to store one solution value at each grid-point of the background mesh.

Assume that at time t^n the solution is known at all grid- and discontinuity-points. The computation of the subsequent time level $t^{n+1} = t^n + \Delta t$ using the eDIT method can be split into several steps that will be described in detail in the following sub-sections.

5.1.1 Geometrical setting

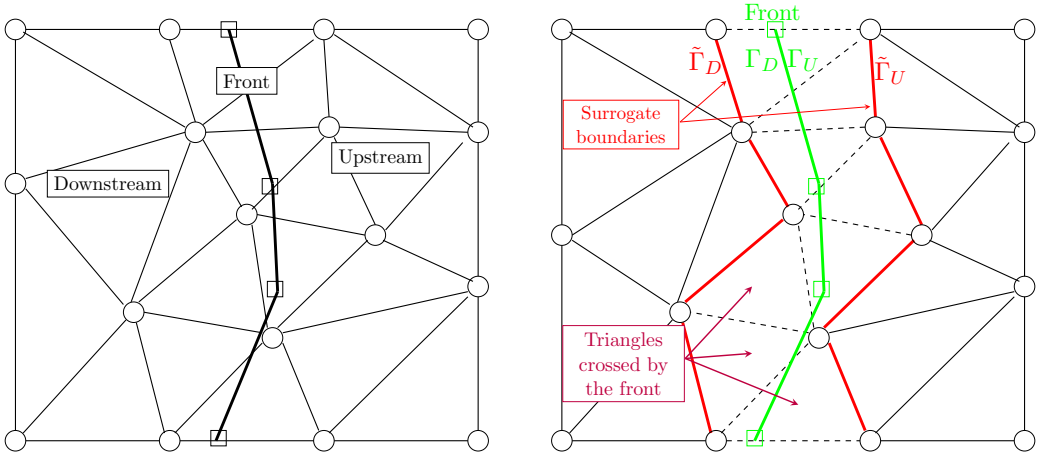
The first step consists in flagging the triangles crossed by the front. As shown on Figure 5.1(b), this leads to the creation of a cavity of flagged elements enclosing the discontinuity. Here, in contrast to the unstructured technique proposed in [235], or to more classical methods such as the boundary shock fitting [269], the mesh within or in proximity of this cavity is not modified to be conformal with the discontinuity. This cavity separates two sub-domains, and allows to define two surrogate boundaries, which are the intersections between the boundaries of the sub-domains and the boundaries of the front cavity. The mesh of the sub-domains separated by the front cavities will be referred to as the *computational mesh*. The computational mesh is identical to the background mesh, except for the removal of the flagged elements enclosing the discontinuity. In the simplest setting of a single discontinuity, the upstream and downstream surrogate boundaries, drawn using red lines in Figure 5.1(b), will be called $\tilde{\Gamma}_U$ and $\tilde{\Gamma}_D$. In the following we also refer to them as surrogate-discontinuities. The discontinuity-boundary, representing the actual front position, will be referred to as Γ and its upstream and downstream sides as Γ_U and Γ_D, respectively. Contrary to $\tilde{\Gamma}_U$ and $\tilde{\Gamma}_D$, Γ_U and Γ_D are superimposed and, of course, coincide with Γ.

Compared to its original version [89], the current algorithm is capable of dealing with several discontinuity-fronts which implies that the computational domain is split into several, disjoint, sub-domains.

5.1.2 Computation of the tangent and normal unit vectors

In order to apply the Rankine-Hugoniot jump relations, the tangent and normal unit vectors along the shock-front have to be calculated within each pair of shock-points. The tangent unit vector $\boldsymbol{\tau}_i$ in shock-point i is obtained from:

$$\boldsymbol{\tau}_i = \frac{\mathbf{V}_{\tau_i}}{\mid \mathbf{V}_{\tau_i} \mid}, \tag{5.1}$$



(a) Front-mesh laid on top of the background-mesh.

(b) Front-mesh, computational-mesh and surrogate boundaries.

Figure 5.1: The computational-mesh is obtained by removing those cells of the background mesh that are crossed by the front-mesh.

where $\mathbf{V}_{\tau_i}$ is the vector tangent to the shock-front in shock-point i. The normal unit vector $\mathbf{n}_i$ is perpendicular to $\boldsymbol{\tau}_i$ and such that it points from the shock-downstream towards the shock-upstream region. The computation of $\mathbf{V}_{\tau_i}$ relies on finite difference formulae which involve the coordinates of the shock-point itself and those of its neighboring shock-points. By reference to Figure 5.2, $\mathbf{x}\left(P_i^t\right)$ denotes the position of shock-point i at time level t. Shock-points $i-1$ and $i+1$ are located on both sides of shock-point i and their position $\mathbf{x}\left(P_{i-1}^t\right)$ and $\mathbf{x}\left(P_{i+1}^t\right)$ at time level t can be used to compute the tangent and normal unit vectors in shock-point i. A preliminary test is required to verify whether these adjacent shock-points belong to the domain of dependence of shock-point i. This is easily checked using the following inequality:

$$\mathbf{v}_{d,i+1}^t \cdot \boldsymbol{\tau}_{i+\frac{1}{2}} - a_{d,i+1}^t < 0, \tag{5.2}$$

where

$$\boldsymbol{\tau}_{i+\frac{1}{2}} = \frac{\mathbf{x}\left(P_{i+1}^t\right) - \mathbf{x}\left(P_i^t\right)}{l_{i+\frac{1}{2}}}, \qquad l_{i+\frac{i}{2}} = |\mathbf{x}\left(P_{i+1}^t\right) - \mathbf{x}\left(P_i^t\right)|, \tag{5.3}$$

and $\mathbf{v}_{d,i+1}^t$ and $a_{d,i+1}^t$ are the shock-downstream flow and acoustic velocity in shock-point $i+1$ at time level t. If Equation (5.2) is verified, shock-point $i+1$ falls within the domain of dependence of shock-point i. Once this test has been repeated in shock-point $i-1$, three different situations may arise:

1. both shock-points $i-1$ and $i+1$ are in the domain of dependence of shock-point i;

2. only shock-point $i-1$ is in the domain of dependence of shock-point i;

3. only shock-point $i+1$ is in the domain of dependence of shock-point i;

When case 1 applies, the computation of $\mathbf{V}_{\tau_i}$ must involve the shock-points on both sides; therefore:

$$\mathbf{V}_{\tau_i} \;=\; \boldsymbol{\tau}_{i+\frac{1}{2}}\, l^2_{i-\frac{1}{2}} \;+\; \boldsymbol{\tau}_{i-\frac{1}{2}}\, l^2_{i+\frac{1}{2}}. \tag{5.4}$$

When case 2 applies, shock-point $i+1$ must not be used in the computation of the tangent vector $\mathbf{V}_{\tau_i}$, and the following upwind-biased formula, which involves shock-point $i-2$, instead of $i+1$, is used:

$$\mathbf{V}_{\tau_i} \;=\; \boldsymbol{\tau}_{i-\frac{1}{2}} \left(l_{i-\frac{1}{2}} + l_{i-\frac{3}{2}}\right)^2 \;+\; \left(\boldsymbol{\tau}_{i-\frac{1}{2}} + \boldsymbol{\tau}_{i-\frac{3}{2}}\right) l^2_{i-\frac{1}{2}}. \tag{5.5}$$

Finally, the third case is specular to the second one, but the corresponding formula involves shock-points i, $i+1$ and $i+2$.

The finite difference approximations (5.4) and (5.5) are both second-order-accurate even if the shock-points are un-evenly spaced along the shock-front.

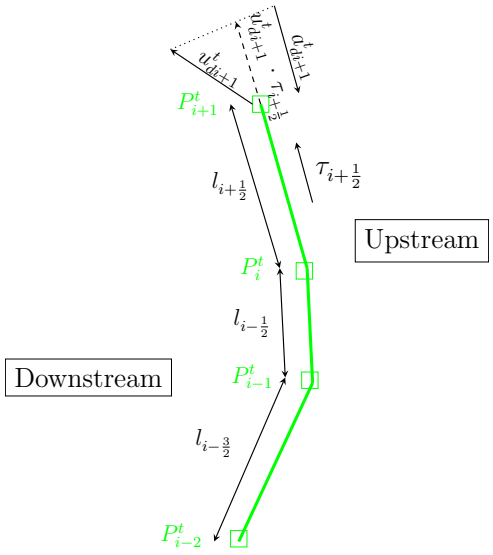


Figure 5.2: Test needed to check whether point P_{i+1} belongs to the domain of dependence of P_i.

It is also convenient to express the components of the velocity vector $\mathbf{v}$ in the $(\mathbf{n}, \boldsymbol{\tau})$ reference frame which is locally attached to the discontinuity:

$$v_n = \mathbf{v} \cdot \mathbf{n} \quad \text{and} \quad v_\tau = \mathbf{v} \cdot \boldsymbol{\tau}. \tag{5.6}$$

5.1.3 Definition of the sub-domains

In principle the flow may evolve n different fronts, which need to be handled numerically. The identification of these discontinuities is in itself a challenging problem (see e.g. [241, 32] and references therein), which is out of the scopes of this work. Here we assume to be given in advance a set $\{\Gamma_j\}_{j=1}^{n}$ of n discontinuities, as well as their topology and nature (shocks and/or contact discontinuities), and the set of numbered sub-domains $\{\Omega_l\}_{l=1}^{m}$ separated by the discontinuities. We also start from a brute set of the ensemble of all points and edges of the surrogate discontinuities, which we denote by $\tilde{\Gamma}$. We process this ensemble as follows:

1. associate to each point in $\tilde{\Gamma}$ the index of the closest discontinuity (distance measured by orthogonal projection);

2. associate to each point in $\tilde{\Gamma}$ the Upstream/Downstream flag based on its position w.r.t. the orientation of the front normals;

3. for each discontinuity assemble the corresponding arrays of upstream and downstream surrogate boundaries (edge collection) as better shown in Figure 5.3;

4. build a pointer providing the explicit mapping between actual and surrogate discontinuities.

The connectivity obtained allows to move easily from one surrogate discontinuity to the corresponding front-mesh, or to the other surrogate corresponding to the same discontinuity (e.g. move between the upstream and downstream surrogates $\tilde{\Gamma}_{Uj}$ and $\tilde{\Gamma}_{Dj}$ of the two shocks of Figure 5.3), as well as between different surrogates bounding the same sub-domain (e.g. from $\tilde{\Gamma}_{D1}$ to $\tilde{\Gamma}_{U2}$ in Figure 5.3).

5.1.4 Solution update using the shock-capturing code

The solution is updated to time level t^{n+1} using a shock-capturing code. The flow computations are performed on the non-communicating sub-domains separated by the front cavity, and including the surrogate discontinuities $\tilde{\Gamma}_U$ and $\tilde{\Gamma}_D$ as boundaries (see Figure 5.4). As we will see in the next section, the boundary conditions imposed on the surrogate discontinuities are defined starting from the values of the flow variables on Γ_U and Γ_D which, as already said, are in general different, because Γ is a discontinuity.
Concerning the solver used in this work, it is based on a Residual Distribution (RD) method evolving in time approximations of the values of the flow variables in grid-points.

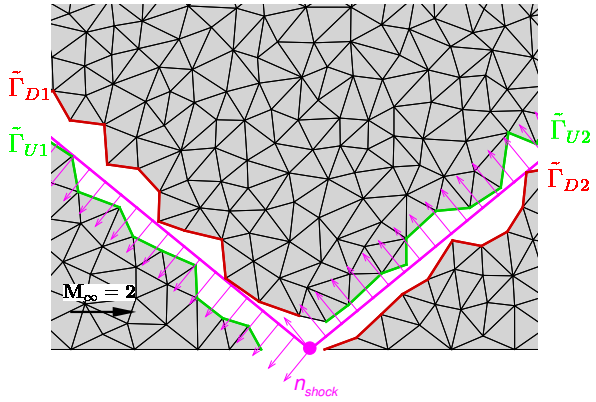


Figure 5.3: Definition of surrogate boundaries when multiple fronts interact; in the regular reflection shown here, the surrogate boundaries marked in green are located upstream w.r.t. the incident and reflected shocks, whereas those marked in red are located downstream.

The method has several appealing characteristics, including the possibility of defining genuine multidimensional upwind strategies for Euler flows, by means of a wave decoupling exploiting appropriately preconditioned forms of the equations [47]. By combining ideas from both the stabilized finite element and finite volume methods, these schemes allow to achieve second order of accuracy and monotonicity preservation with a compact stencil of nearest neighbors. The interested reader can refer to [112, 4] and references therein for an in-depth review of this family of methods, as well as to [47, 50] and references therein for some specific choices of the implementation used here.

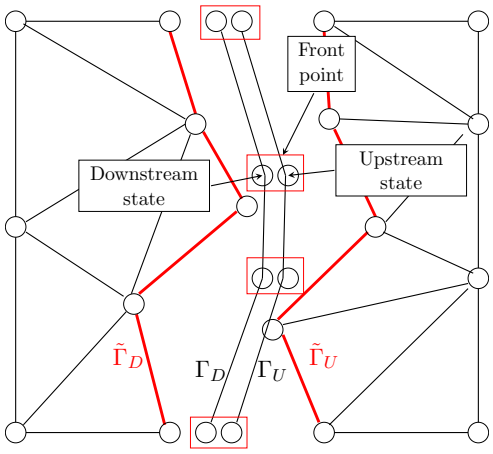


Figure 5.4: The solution update is performed using the computational mesh.

5.1.5 Surrogate discontinuity conditions update

The flow solver provides updated nodal values at all grid-points of the computational mesh at time level t^{n+1}. When dealing with *shock waves*, the shock-upstream surrogate boundary, $\tilde{\Gamma}_U$ behaves like a supersonic outflow and, therefore, no boundary conditions should be applied. However, along the shock-downstream surrogate $\tilde{\Gamma}_D$ the flow is subsonic in the shock-normal direction and in principle only the characteristic variable [1]. conveyed by the slow acoustic wave has been correctly updated, while boundary conditions for the remaining ones (the fast acoustic, entropy and vorticity waves) should be imposed. The situation is similar on both sides of a *contact discontinuity*.

Updating correctly the interface conditions across the embedded discontinuity is very delicate, and it is the key of the method proposed. It involves several steps, which are detailed hereafter.

Computational mesh to discontinuity mesh transfer

The transfer of the flow data from the computational mesh to the front-mesh is necessary before imposing the interface conditions. In this work we use a CFD solver strongly relying on the use of Roe's parameter vector $\mathbf{Z} = \sqrt{\rho}\,(1, H, v_x, v_y)^t$ [263, 109] as the dependent variable, so the transfer is also done in terms of $\mathbf{Z}$. This is evidently not a necessary choice, and any other choice of state vector is acceptable. This first transfer is required to update the flow variables along Γ_U and Γ_D. Following [279, 89], this is achieved by first defining a map M, such that

$$\tilde{\mathbf{x}} = M(\mathbf{x}), \tag{5.7}$$

where, as shown in Figure 5.5, the point $\tilde{\mathbf{x}} = \mathbf{x}(A^i)$ belonging to the computational mesh is the projection in the direction of the front-normal $\mathbf{n}$ of a front-point $\mathbf{x} = \mathbf{x}(P_i)$ of the front-mesh. Then we use a forward Taylor series expansion to express the data along the discontinuity in terms of the values of the flow variables and of their derivatives on the computational mesh. A first-order transfer reads

$$\mathbf{Z}(\mathbf{x}) \;=\; \mathbf{Z}(\tilde{\mathbf{x}}) \;+\; \mathcal{O}(\|\mathbf{x} - \tilde{\mathbf{x}}\|)\,, \tag{5.8}$$

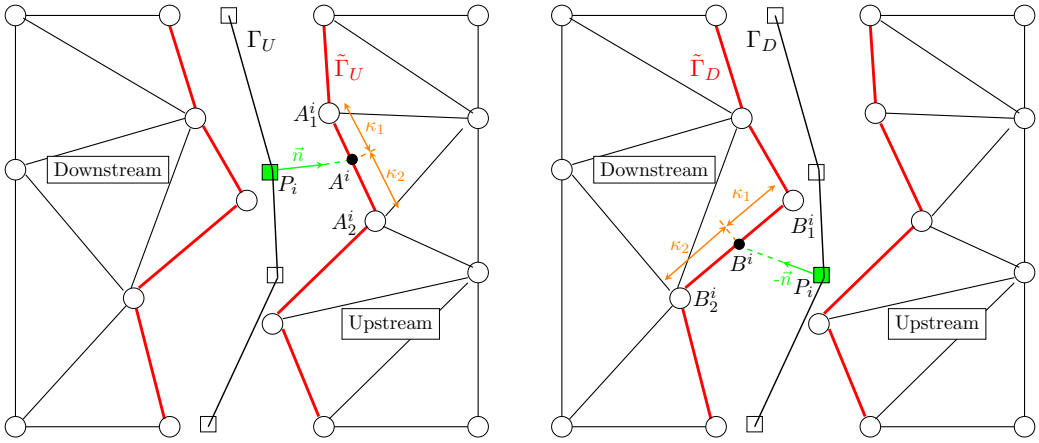
while a second order extrapolation is written as

$$\mathbf{Z}(\mathbf{x}) \;=\; \mathbf{Z}(\tilde{\mathbf{x}}) \;+\; \nabla\mathbf{Z}(\tilde{\mathbf{x}}) \cdot (\mathbf{x} - \tilde{\mathbf{x}}) \;+\; \mathcal{O}(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2). \tag{5.9}$$

[1] Hereafter, characteristic variables shall also be referred to as Riemann variables

With the exception of few methods based on a C^1 approximation (see e.g. [145]), in general, the gradients of the flow variables are undefined at mesh-points. To avoid handling specific singular cases, we have implemented (5.9) using interpolated reconstructed nodal gradients. Note that, in order to achieve an overall second order of accuracy in the calculation of $\mathbf{Z}(\mathbf{x})$, the approximation of the gradient in Equation (5.9) only needs to be consistent, i.e. at least first-order-accurate.

In practice we proceed as shown in Figure 5.5. When moving from the computational



(a) From the surrogate boundary $\tilde{\Gamma}_U$ to the shock-upstream side of the front Γ_U.

(b) From the surrogate boundary $\tilde{\Gamma}_D$ to the shock-downstream side of the front Γ_D.

Figure 5.5: First transfer between the the surrogate boundaries and the front-mesh.

mesh to Γ_U (see Figure 5.5a) a front-point P_i is mapped to a point A^i. The values of the dependent variables and of their gradients in A^i are interpolated from the neighbors A_1^i and A_2^i:

$$\phi(A^i) = \kappa_2\,\phi(A_1^i) + \kappa_1\,\phi(A_2^i), \tag{5.10}$$

where ϕ is either $\mathbf{Z}$ or $\nabla\mathbf{Z}$, and κ_1 and κ_2 are the weights, equal to the normalized distances between A^i and grid-points A_1^i and A_2^i. The evaluation of the gradient in the grid-points of the surrogate boundaries may be performed using different approaches, as reported in Section 5.1.7. Once the value of $\mathbf{Z}$ and $\nabla\mathbf{Z}$ in point A^i has been computed using Equation (5.10), $\mathbf{Z}$ in P_i is computed by means of Equation (5.9), having set $\mathbf{x} = \mathbf{x}(P_i)$ and $\tilde{\mathbf{x}} = \mathbf{x}(A^i)$.

When moving from the computational mesh to Γ_D the procedure is identical: the front-point P_i is updated using its mapped point B^i (see Figure 5.5b). Values of the relevant quantities are again linearly interpolated, and $\mathbf{Z}$ is computed by means of Equation (5.9), having set $\mathbf{x} = \mathbf{x}(P_i)$ and $\tilde{\mathbf{x}} = \mathbf{x}(B^i)$.

Riemann variables and jump conditions enforcement

The solution updated by the flow solver and extrapolated to the discontinuity mesh does not take into account the coupling between the different sub-domains. This coupling is done here based on the Rankine-Hugoniot jump relations. On the upstream side Γ_U of a *shock wave* all the Riemann variables are transported into the shock, so the data extrapolated from the computational mesh on this side is assumed to be correct. On the shock-downstream side, however, this holds true only for the Riemann variable associated with the slow acoustic wave that moves towards the shock (the subscript $_D$ denotes values along Γ_D):

$$R_D = a_D^{n+1} + \frac{\gamma-1}{2}(v_n)_D^{n+1}. \tag{5.11}$$

We need to provide relations to compute along Γ_D the values of the Riemann variables corresponding to the fast acoustic wave, as well as to the entropy and vorticity waves. These relations are supplied by the Ranking-Hugoniot jump conditions which, for a perfect gas, can be recast as:

$$\begin{aligned}
\rho_D^{n+1}(v_n)_D^{n+1} - w_s^{n+1}\rho_D^{n+1} &= \rho_U^{n+1}(v_n)_U^{n+1} - w_s^{n+1}\rho_U^{n+1} \\
\rho_D^{n+1}((v_n)_D^{n+1} - w_s^{n+1})^2 + p_D^{n+1} &= \rho_U^{n+1}((v_n)_U^{n+1} - w_s^{n+1})^2 + p_U^{n+1} \\
\frac{\gamma}{\gamma-1}\frac{p_D^{n+1}}{\rho_D^{n+1}} + \frac{1}{2}((v_n)_D^{n+1} - w_s^{n+1})^2 &= \frac{\gamma}{\gamma-1}\frac{p_U^{n+1}}{\rho_U^{n+1}} + \frac{1}{2}((v_n)_U^{n+1} - w_s^{n+1})^2 \\
(v_\tau)_D^{n+1} &= (v_\tau)_U^{n+1}
\end{aligned} \tag{5.12}$$

with w_s^{n+1} the shock speed. The algebraic non-linear system made up of Equations (5.11) and (5.12) allows to compute the five unknowns (ρ_D, $\mathbf{v}_D$, p_D, w_s) given the known upstream state (ρ_U, $\mathbf{v}_U$, p_U) and the slow acoustic Riemann variable R_D.

For a *contact discontinuity*, we use as input the following set of Riemann variables, which we assume to be correctly updated in the computational mesh:

$$\begin{gathered}
R_D = a_D^{n+1} + \frac{\gamma-1}{2}(v_n)_D^{n+1}\ ,\ R_U = a_U^{n+1} - \frac{\gamma-1}{2}(v_n)_U^{n+1} \\
s_D = \frac{p_D^{n+1}}{(\rho_D^{n+1})^\gamma}\ ,\ s_U = \frac{p_U^{n+1}}{(\rho_U^{n+1})^\gamma} \\
V_D = (v_\tau)_D^{n+1}\ ,\ V_U = (v_\tau)_U^{n+1}
\end{gathered} \tag{5.13}$$

Note that when dealing with a contact discontinuity the situation is essentially symmetric and the jump relations simplify somewhat, and can be shown to reduce to:

$$\begin{aligned} p_U^{n+1} &= p_D^{n+1} \\ (v_n)_U^{n+1} &= w_{cd}^{n+1} \\ (v_n)_D^{n+1} &= w_{cd}^{n+1} \end{aligned} \tag{5.14}$$

The algebraic non-linear system made up of Equations (5.13) and (5.14) allows to compute the nine unknowns (ρ_U, $\mathbf{v}_U$, p_U, ρ_D, $\mathbf{v}_D$, p_D, w_{cd}), given the six Riemann variables (R, s, V) on the two sides of the discontinuity and the jump relations (5.14).

For both shocks and contact discontinuities, a non-linear system of algebraic equations needs to be solved in each discontinuity-point. This is done using the Newton-Raphson root-finding algorithm, thus providing the correct states and w at time level t^{n+1} .
Whenever two or more discontinuities interact special care is required for the front-points belonging to different discontinuities. The numerical treatment of these points is done in a case by case manner and detailed in the sections related to the specific tests in Section 5.3.

Discontinuity mesh to computational mesh transfer

Once the value of the unknowns have been corrected along the discontinuity mesh to account for the jump conditions, the corrected values need to be transferred back to the surrogate discontinuities. To do this, we proceed in a similar manner as before by writing,

$$\mathbf{Z}(\tilde{\mathbf{x}}) = \mathbf{Z}(\mathbf{x}) - \nabla\mathbf{Z}(\tilde{\mathbf{x}}) \cdot (\mathbf{x} - \tilde{\mathbf{x}}) + \mathcal{O}(\|\mathbf{x} - \tilde{\mathbf{x}}\|^2). \tag{5.15}$$

Note that there is a substantial difference in the direction in which the Taylor expansion is performed which is now a backward expansion from the arrival point (the surrogate discontinuity) to the point where the data is available (the discontinuity mesh). As in the previous case, several choices are possible to evaluate the gradient involved in this transfer. We shall focus on this in Section 5.1.7. In the first-order accurate case, Equation (5.15) simplifies to:

$$\mathbf{Z}(\tilde{\mathbf{x}}) = \mathbf{Z}(\mathbf{x}) + \mathcal{O}(\|\mathbf{x} - \tilde{\mathbf{x}}\|) \tag{5.16}$$

Finally note that for shock-waves only grid-points on $\tilde{\Gamma}_D$ need to be updated.

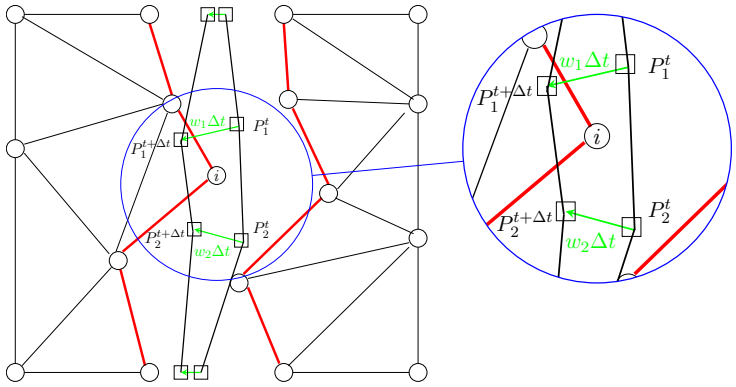


Figure 5.6: The front overtakes a grid-point of the background mesh during its motion.

5.1.6 Front displacement and nodal re-initialization

The new position of the front at time level t^{n+1} is computed by displacing all discontinuity-points using the following first-order-accurate (in time) formula:

$$\mathbf{x}\left(P^{n+1}\right) \;=\; \mathbf{x}\left(P^{n}\right) \;+\; w^{n+1}\,\mathbf{n}\,\Delta t \tag{5.17}$$

where $\mathbf{x}\left(P\right)$ denotes the geometrical location of the shock-points, and w^{n+1} may represent either w_s^{n+1} or w_{cd}^{n+1}. The use of a first-order-accurate temporal integration formula has no impact as long as steady flows are of interest. For unsteady flows, second-order-accurate time integration formulae should be used, as done for example in [45, 69].

Since the discontinuity can freely float over the background triangulation, it may happen that it crosses the surrogate boundaries. This situation has been sketched in Figure 5.6, where grid-point i has been overtaken by the moving front. Whenever this happens, the flow state within grid-point i has to be changed accordingly. In particular, the flow variables in these points are re-computed through an interpolation. To evaluate whether a grid-point i has been overtaken or not by the discontinuity a simple approach has been implemented. Since $\mathbf{n}$ points always upstream, the sign of the following scalar product allows us to distinguish the grid-points located upstream from those located downstream.

$$\alpha_i \;=\; \left(\mathbf{x}_p - \mathbf{x}_i\right)\cdot\mathbf{n} \;=\; \begin{cases} < 0\,, i \text{ is upstream} \\ > 0\,, i \text{ is downstream} \end{cases} \tag{5.18}$$

where $\mathbf{x}_i$ and $\mathbf{x}_p$ are, respectively, the coordinates of a grid-point i and its projection over the closest front-edge and $\mathbf{n}$ is the normal vector computed in the closest front-point to $\mathbf{x}_i$. If α_i changes sign when the front moves, this means that grid-point i has been overtaken

and its state has to be updated.

5.1.7 Evaluation of the nodal gradients

The technique used to extrapolate the flow variables back and forth between the surrogate discontinuities and the discontinuity mesh involves the knowledge of nodal gradients which need to be reconstructed. The solvers used in this work are based on collocated nodal methods, so we will discuss the recovery of the nodal gradients in this specific case. The generalization to other cases is of course possible, but left out of this work.

As for the enforcement of the jump conditions we distinguish two main cases. For a shock wave, in the upstream domain we already said that all characteristic information runs into the shock. For this situation, it seems natural to evaluate the nodal gradients along the surrogate discontinuity based on the data pre-computed by the flow solver within the upstream domain. To this end we have tested two gradient recovery strategies:

- a Green-Gauss (GG) gradient reconstruction, essentially boiling down to an area weighted average of the gradients in the cells surrounding a grid-point;
- an area-weighted version of Zienkiewicz-Zhu's patch super-convergent method (ZZ) [309].

Explicit formulas for both methods are provided in Appendix A.1.

Within the region downstream of a shock, as well as for contact discontinuities, as discussed in the previous sections, the data computed by the flow solver must be corrected to account for the jump conditions. This fact has led us in the past to include the corrected data in the evaluation of the nodal gradients in these cases [89]. In the aforementioned reference, which did not account for discontinuity interactions, the gradient term of Equation (5.15) is replaced with the cell-wise gradient of an auxiliary triangle (marked using a dashed line in Figure 5.7) containing the surrogate grid-point to be updated (grid-point i in Figure 5.7), and defined by a point on the discontinuity-mesh (point P^i in Figure 5.7), and two grid-points of the computational mesh, not belonging to the surrogate discontinuity. However, when interactions are present, singular topological situations may arise in which this approach cannot be used. For example, for region 1 in Figure 5.7, a long strip of one row of elements is trapped within two discontinuities and the interaction point. For the grid-points along the boundary of this strip there are no neighboring inner grid-points to construct the auxiliary triangle. Several other gradient-reconstruction formulations have also been tested. The simplest involves using a one-sided recovered gradient not accounting for the Rankine-Hugoniot correction. The latter not only turned out to be very

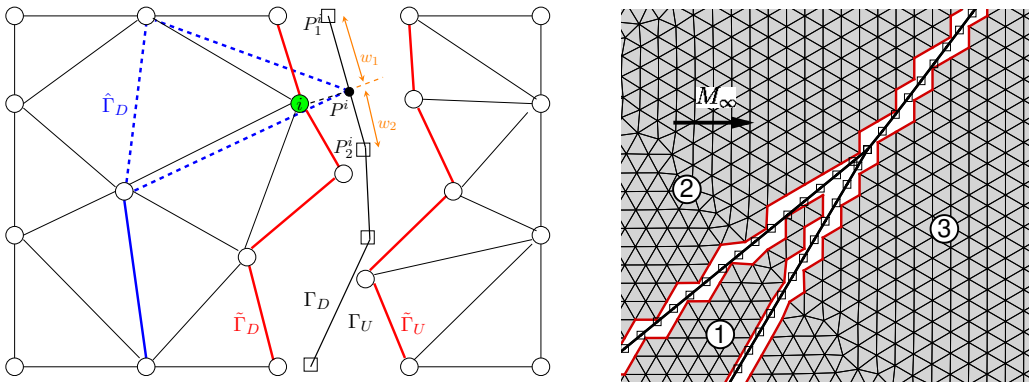


Figure 5.7: Left: auxiliary triangle defining the nodal gradient with corrected data on the front. Right: a problematic example of an interaction for which no auxiliary triangle can be built close to the interaction point in domain 1.

useful for computing interacting discontinuities, but much simpler, especially in view of possible extensions to three space dimensions. In the following, we will refer to the method of [89] as eST, while $eDIT_{GG}$ (GG reconstruction), and $eDIT_{ZZ}$ (ZZ reconstruction) will denote the extrapolated discontinuity tracking obtained using one sided reconstructions. Finally, $eDIT_{FO}$ (first order) refers to simulations run with a second-order-accurate discretization of the governing PDEs, but only first-order accurate data transfer between the surrogate and actual discontinuities, i.e. by using Equations (5.8) and (5.16).

5.2 Remarks on conservation

We make a small detour to discuss the issue of conservation. A first important remark is that the notion of conservation is essential when considering the approximation of shocks. Failing to properly account for conservation of mass, momentum, and energy would lead to a wrong approximation of these features, both in terms of position and strength. In smooth regions, as well as across contact discontinuities, the use of non-conservative approaches is less critical, and in some cases even advantageous (e.g. [5, 106, 112, 2]).

It is also important to realize that the definition of discrete conservation is associated to the identification of both a geometrical cell over which conservation is expressed, and a numerical flux expressing local conservation. The interested reader can refer to [4, 3]

for a discussion. Typically, global conservation over the domain is thus expressed as

$$\int_{\partial\Omega} \hat{\mathcal{F}}_n(\mathbf{Z})\mathrm{d}\Gamma = 0 \tag{5.19}$$

with $\hat{\mathcal{F}}_n$ the numerical flux associated to the boundary conditions. Now note that, even for exact quadrature, the numerical flux is not equal to the physical one $\mathcal{F}_n(\mathbf{Z})$, and the difference between the two is in general of the order of the truncation error of the discretization. So conservation is still verified within some numerical approximation.

The situation is similar for the embedded approach proposed here, for which, despite the exact imposition of the jump conditions across discontinuities, conservation can only be measured within the truncation of the extrapolation method. To be more precise, let us focus on the configuration of Figure 5.8. In the figure, the domains on the right of $\tilde{\Gamma}_U$ is the upstream domain, while the domain on the left of $\tilde{\Gamma}_D$ is the downstream one. The flow is assumed to go from the right to the left. A discontinuity is placed in the middle of the domain. We have denoted by Γ_U and Γ_D the upstream and downstream sides of the discontinuity. The discretization of all the domains, including the upstream/downstream boundaries and the discontinuity, is shown in the figure. In particular, as already discussed, unlike in the unstructured shock-fitting method [235], for the eDIT method the shock-edges are no longer part of the computational mesh. In the region between the shock mesh and the computational mesh, the conservation law is then replaced by the extrapolation procedure, which is the main source of loss of conservation. Note however, that this loss only occurs in the smooth parts of the flow. In particular, denoting by Ω_e^U the region enclosed between Γ^U and $\tilde{\Gamma}^U$, given a smooth exact solution of the Euler equations we can write easily the upstream consistency estimate

$$\begin{aligned}\mathcal{I}_h^U(\mathbf{Z}_h) &:= \oint_{\partial\Omega_e^U} \mathcal{F}_n(\mathbf{Z}_h)\mathrm{d}\Gamma = \oint_{\partial\Omega_e^U} [\mathcal{F}_n(\mathbf{Z}_h) - \mathcal{F}_n(\mathbf{Z}_h^{\mathsf{ex}})]\,\mathrm{d}\Gamma \\ &= \oint_{\partial\Omega_e^U} [\mathcal{F}_n(\mathbf{Z}_h(\tilde{\mathbf{x}}) + \nabla\mathbf{Z}_h(\tilde{\mathbf{x}})\cdot(\mathbf{x}-\tilde{\mathbf{x}})) - \mathcal{F}_n(\mathbf{Z}_h^{\mathsf{ex}}(\tilde{\mathbf{x}}) + \nabla\mathbf{Z}_h^{\mathsf{ex}}(\tilde{\mathbf{x}})\cdot(\mathbf{x}-\tilde{\mathbf{x}}))]\,\mathrm{d}\Gamma + \mathcal{O}(\mathbf{d}^2).\end{aligned}$$

Where $\mathbf{Z}^{\mathsf{ex}}$ is the Roe's parameter vector evaluated for the exact solution. Formally replacing the nodal values of the solution with those of the exact one, we obtain the consistency estimate $\mathcal{I}_h^U(\mathbf{Z}_h^{\mathsf{ex}}) = \mathcal{O}(\mathbf{d}^2)$. Using the exact same arguments, we can write the estimate $\mathcal{I}_h^D(\mathbf{Z}_h^{\mathsf{ex}}) = \mathcal{O}(\mathbf{d}^2)$ for the loss of conservation in the downstream region Ω_e^D enclosed between Γ^D and $\tilde{\Gamma}^D$.

Let us now set $\tilde{\Gamma}^{\mathcal{T}} = \tilde{\Gamma}_U^{\mathcal{T}} + \tilde{\Gamma}_D^{\mathcal{T}}$ and $\tilde{\Gamma}^{\mathcal{B}} = \tilde{\Gamma}_U^{\mathcal{B}} + \tilde{\Gamma}_D^{\mathcal{B}}$. By construction of the method

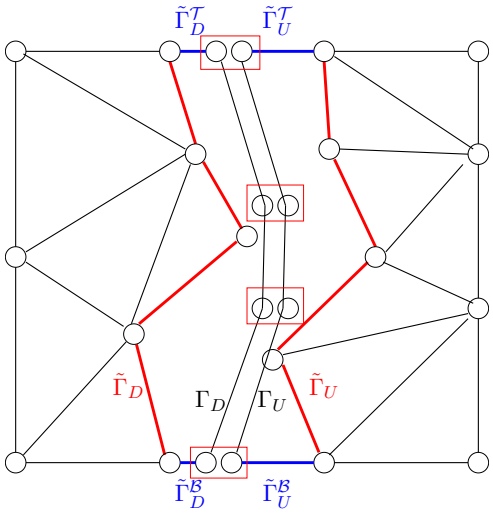


Figure 5.8: Area of the cavity Ω_e when open shock geometries are considered.

proposed here, the jump condition $[\![\mathcal{F}_n(\mathbf{Z}_h)]\!]_\Gamma = 0$ is exactly satisfied. This allows readily to write the global consistency estimate on conservation (cf. 5.8 for the notation)

$$\int_{\tilde{\Gamma}_U} \mathcal{F}_n(\mathbf{Z}_h^{\mathsf{ex}})\mathrm{d}\Gamma + \int_{\tilde{\Gamma}_D} \mathcal{F}_n(\mathbf{Z}_h^{\mathsf{ex}})\mathrm{d}\Gamma + \int_{\tilde{\Gamma}^{\mathcal{T}}} \mathcal{F}_n(\mathbf{Z}_h^{\mathsf{ex}})\mathrm{d}\Gamma + \int_{\tilde{\Gamma}^{\mathcal{B}}} \mathcal{F}_n(\mathbf{Z}_h^{\mathsf{ex}})\mathrm{d}\Gamma = \mathcal{I}_h(\mathbf{Z}_h^{\mathsf{ex}}) = \mathcal{O}(\mathbf{d}^2), \quad (5.20)$$

reducing to

$$\int_{\tilde{\Gamma}_U} \mathcal{F}_n(\mathbf{Z}_h^{\mathsf{ex}})\mathrm{d}\Gamma + \int_{\tilde{\Gamma}_D} \mathcal{F}_n(\mathbf{Z}_h^{\mathsf{ex}})\mathrm{d}\Gamma = \mathcal{I}_h(\mathbf{Z}_h^{\mathsf{ex}}) = \mathcal{O}(\mathbf{d}^2), \quad (5.21)$$

whenever the out/inflow on the top/bottom boundaries is zero. So the conservation error is solely controlled from the accuracy of the extrapolation formula. This means that first or second order is obtained in terms of conservation, and thus in terms of position and magnitude of the discontinuities, depending on whether the gradient correction is included or not. This will be verified in practice in the numerical results section. Note that this behavior is better than what any fully conservative capturing method can provide, and can of course be further improved by enhancing the extrapolation accuracy.

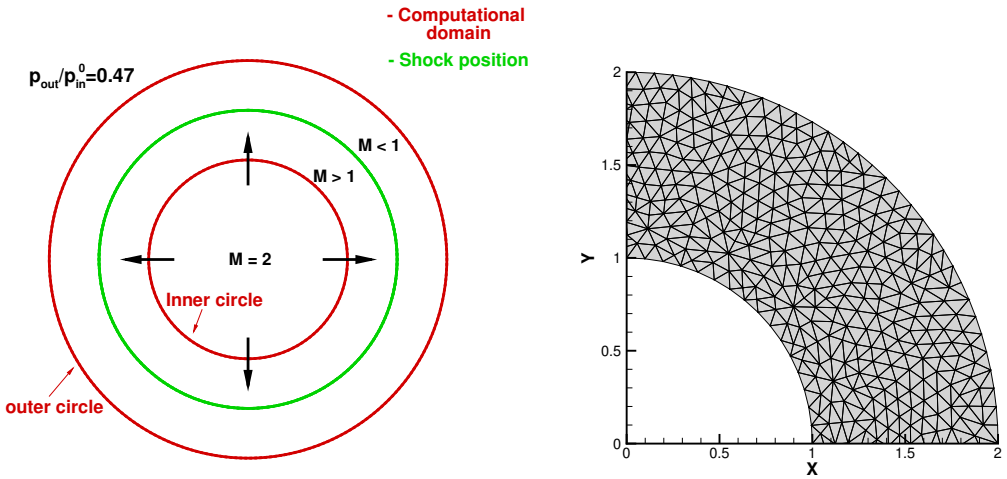
5.3 Validation

To illustrate the capabilities of our method, we provide here examples representative of several types of interactions which can occur in gas-dynamics. The solutions obtained with the extrapolated tracking method will be compared to *i*) full-fledged computations, *ii*) hybrid simulations in which only some of the discontinuities are explicitly tracked while

the others are captured, and *iii*) solutions obtained with the standard shock-capturing method. To evaluate the gradient recovery strategies, a grid-convergence analysis is presented for two test-cases (transonic source flow and blunt body problem) to asses quantitative differences between the approach described in [89] and the more flexible ones proposed here to deal with interactions.

5.3.1 Transonic source flow

This test-case is very useful because of the availability of the analytical solution, which allows to perform grid-convergence studies [89, 51, 67]. Assuming that the analytical velocity field has a purely radial velocity component, it may be easily verified that the two-dimensional, compressible Euler equations, written in a polar coordinate system, become identical to those governing a compressible quasi-one-dimensional flow with a nozzle area variation linear w.r.t. the radial distance from the pole of the reference frame. The computational domain consists in the annulus sketched in Figure 5.9(a): the ratio



(a) Sketch of the computational domain.

(b) Detail of the level 0 unstructured mesh inside the first gradient.

Figure 5.9: Transonic source flow.

between the radii of the outer and inner circles ($L = r_{in}$) has been set equal to $r_{out}/r_{in} = 2$. A transonic (shocked) flow has been simulated by imposing a supersonic inlet flow at $M = 2$ on the inner circle and a ratio between the outlet static and inlet total pressures $p_{out}/p_{in}^0 = 0.47$ such that the shock forms at $r_{sh}/r_{in} = 1.5$. The Delaunay mesh shown in Figure 5.9(b), which contains 6,916 grid-points and 13,456 triangles, has been generated using the `gmsh` mesh generator [144] in such a way that no systematic alignment occurs between the edges of the triangulation and the circular iso-contour lines of the analytical

solution. By doing so, the discrete problem is made truly two-dimensional. The sequence of nested triangulations that have been employed for SC and all $eDIT$ computations are summarized in Table 5.1 for both the background and the computational grids. The

Table 5.1: Transonic source flow: characteristics of the background and computational meshes used to perform the grid-convergence tests.

	Background grid		*Both grids*	*Computational grid*	
Grid level	Grid-points	Triangles	h	Grid-points	Triangles
0	1,369	2,548	0.5286E-01	1,369	2,328
1	5,286	10,192	0.2641E-01	5,286	9,788
2	20,764	40,768	0.1320E-01	20,764	39,968
3	82,296	163,072	0.6602E-02	82,296	161,454

discretization error for a mesh of size h, ϵ_h, is defined as the difference between the numerical solution, u_h, and the analytical one, u_0:

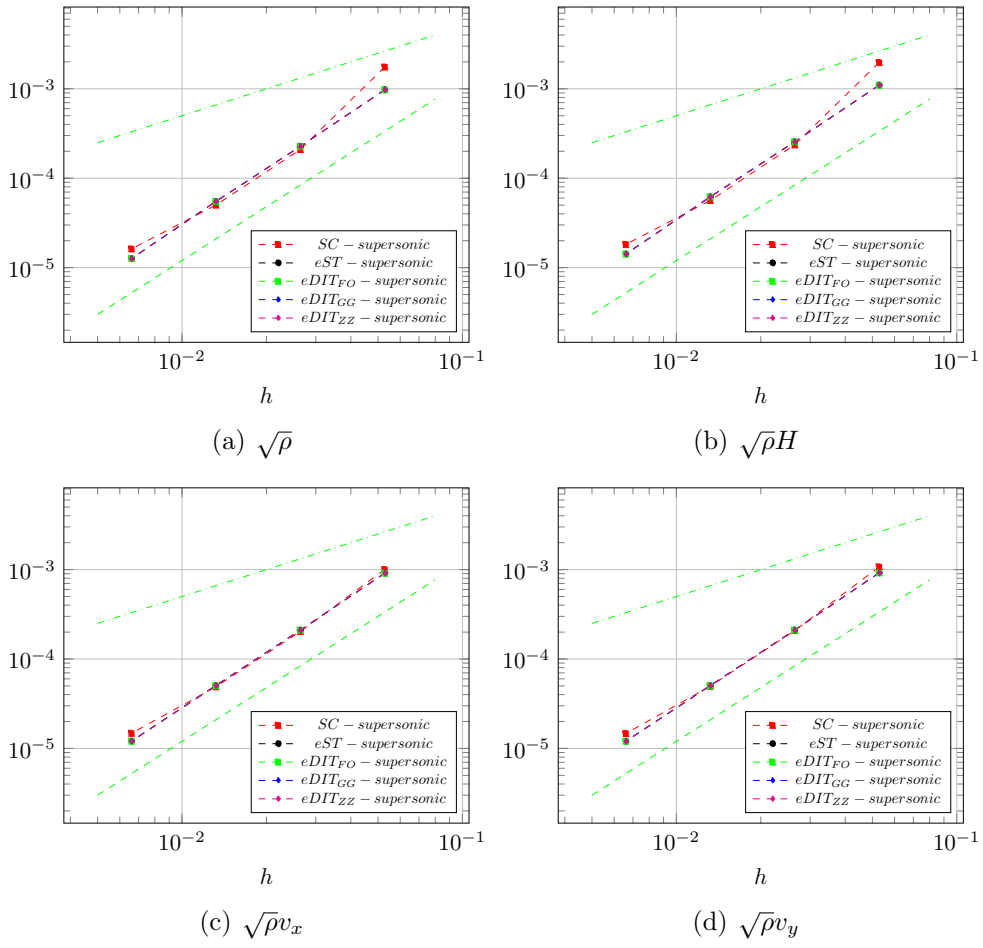
$$\epsilon_h(\mathbf{x}) \;=\; u_h(\mathbf{x}) \;-\; u_0(\mathbf{x}). \tag{5.22}$$

The availability of the exact solution allows to compute the discretization error *locally* using (5.22) or, *globally*, by computing the L_q norms of the discretization error over the entire computational domain Ω using

$$L_q(\epsilon_h) \;=\; \left(\frac{1}{|\Omega|} \int_{\Omega} |\epsilon_h(\mathbf{x})|^q \, d\Omega \right)^{1/q}, \qquad q \;=\; 1,\, 2,\, \infty. \tag{5.23}$$

Herein, the L_1 norm has been employed and computed using Gaussian quadrature rules. Different versions of the extrapolated shock/discontinuity tracking method will be compared to SC computations by doing a thorough grid-convergence analysis. Results obtained from the aforementioned approaches have been separately displayed in the grid-convergence plots for the shock-upstream (supersonic) and shock-downstream (subsonic) regions, i.e. $r < r_{sh}$ and $r > r_{sh}$. In all convergence plots the errors are computed in terms of the parameter vector $\mathbf{Z}$.

Firstly, the global errors obtained on the shock-upstream side of the domain have been displayed to corroborate the fact that, for the supersonic region, the results obtained with all methods are almost identical. Indeed, Figure 5.10 shows that, on the shock-upstream side, all approaches are able to retain the formal order of accuracy of the method. Instead, when looking at the results shown in Figure 5.11, which shows the global errors within the subsonic region, the situation is considerably different. An almost asymptotic convergence is observed for all second order versions of the extrapolated tracking method (eST, $eDIT_{GG}$ and $eDIT_{ZZ}$). Instead, for SC, the convergence rate drops to one, and



(a) $\sqrt{\rho}$ (b) $\sqrt{\rho}H$

(c) $\sqrt{\rho}v_x$ (d) $\sqrt{\rho}v_y$

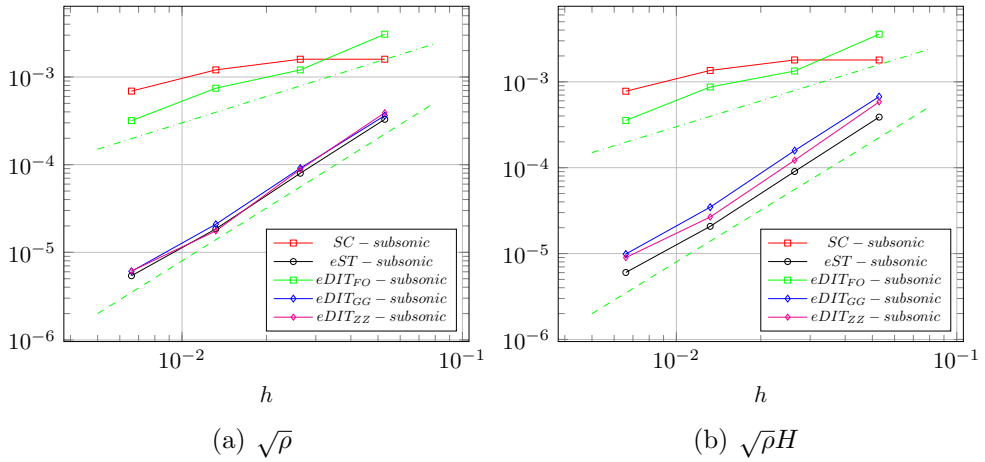
Figure 5.10: Transonic source flow: grid-convergence analysis within the shock-upstream (supersonic) region.

even less for the coarsest meshes. eST and $eDIT_{GG}$ solutions are almost indistinguishable, with global errors that favor more the first approach, probably due to the use of flow data explicitly accounting for the jump conditions also in the extrapolation. The $eDIT_{ZZ}$ is still comparable having a trend in the middle between the two. Although the differences are not remarkable, using a more accurate reconstruction seems to have some effect on the magnitude of the error. The improvement is however so small that the eST can be safely replaced by either $eDIT_{GG}$ or $eDIT_{ZZ}$. This also shows that our basic idea can be coupled to different extrapolation methods, and others, possibly improved ones, could be suggested in the future. As expected, $eDIT_{FO}$ performs as the others in the supersonic side of the domain showing a second-order trend. However in the subsonic side, due to the first order extrapolations carried out at the shock, its trend follows the first order slope curve, because it does not include the nodal gradients in the extrapolation.
Figure 5.13 points out the flux balance computed on the two surrogate boundaries $\tilde{\Gamma}_U$ and $\tilde{\Gamma}_D$ showing that convergence trends strictly depends on the order of the extrapolation. It is indeed shown that first (second) order behavior is displayed for first (second) order extrapolations, confirming the arguments of Section 5.2.

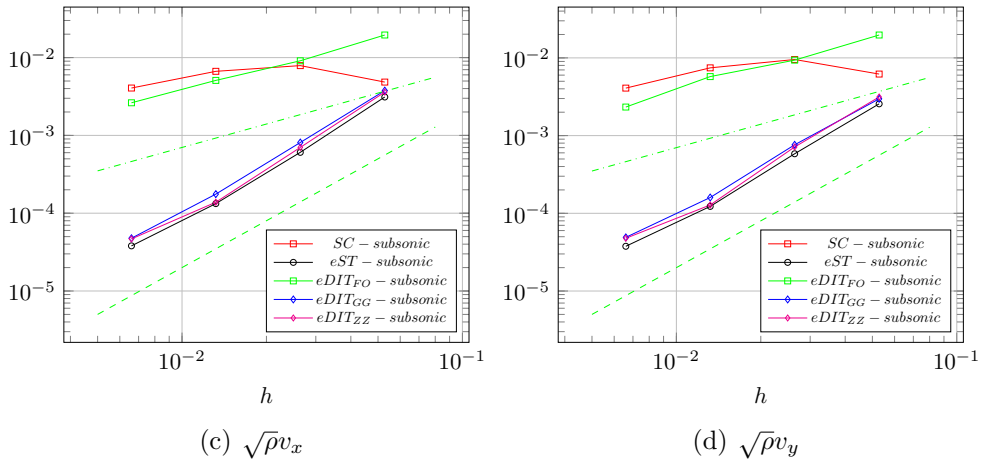
Finally, Figure 5.12, which plots the local discretization error against the radial distance r, clearly reveals the huge reduction of the numerical errors that the various $eDIT$ approaches provide w.r.t. SC. The green line in Figure 5.12 represents the actual position where the shock occurs ($r = 1.5$). As already verified in Figure 5.10, the error computed with either SC or any of the $eDIT$ versions within the shock-upstream region is almost the same. Iso-contour lines of the fourth component of $\mathbf{Z}$ computed on *grid level 2* are displayed in Figure 5.14: the SC solution is shown in Figure 5.14(a), whereas Figure 5.14(b) shows the results of two different $eDIT$ calculations. More precisely, the second-order-accurate $eDIT_{GG}$ result is displayed in the upper half of Figure 5.14(b) and $eDIT_{FO}$ is shown in the lower half of the same frame. When mutually comparing the three different calculations, it can be seen that both the SC and $eDIT_{FO}$ calculations are affected by severe oscillations downstream of the shock, which completely disappear when using $eDIT_{GG}$. Even though the $eDIT_{FO}$ solution looks slightly better than the SC one, Figure 5.11 confirms that $eDIT_{FO}$ cannot be better than first-order accurate behind the shock.

5.3.2 Hypersonic flow past a blunt body

We consider now a hypersonic ($M_\infty = 20$) flow past the fore-body of a circular cylinder. Compared to the previous test-case, this flow is a comprehensive test-bed for the algorithm, because the entire shock-polar is swept whilst moving along the bow shock which

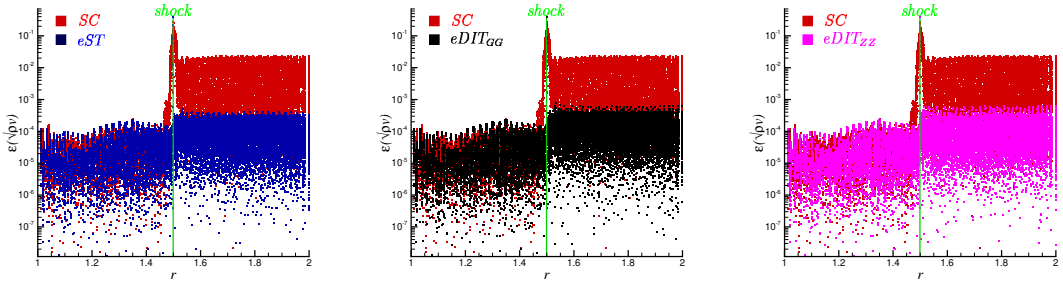


(a) $\sqrt{\rho}$ (b) $\sqrt{\rho}H$



(c) $\sqrt{\rho}v_x$ (d) $\sqrt{\rho}v_y$

Figure 5.11: Transonic source flow: grid-convergence analysis within the shock-downstream (subsonic) region.



(a) eST (b) $eDIT_{GG}$ (c) $eDIT_{ZZ}$

Figure 5.12: Transonic source flow: local discretization error analysis carried out on *grid level 2* to compare the SC computations to the $eDIT$ ones w.r.t. $Z_4 = \sqrt{\rho}v_y$.

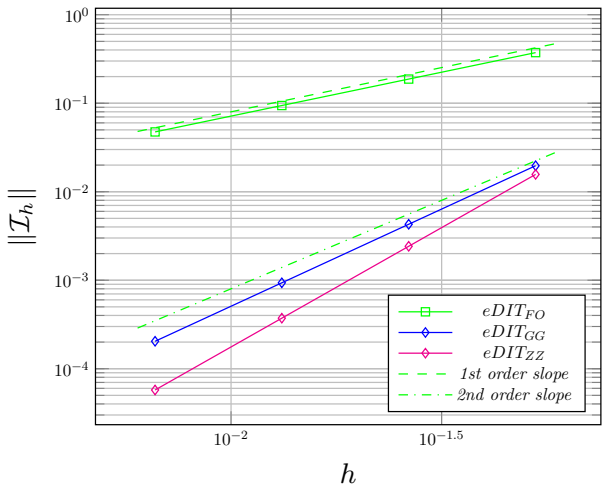
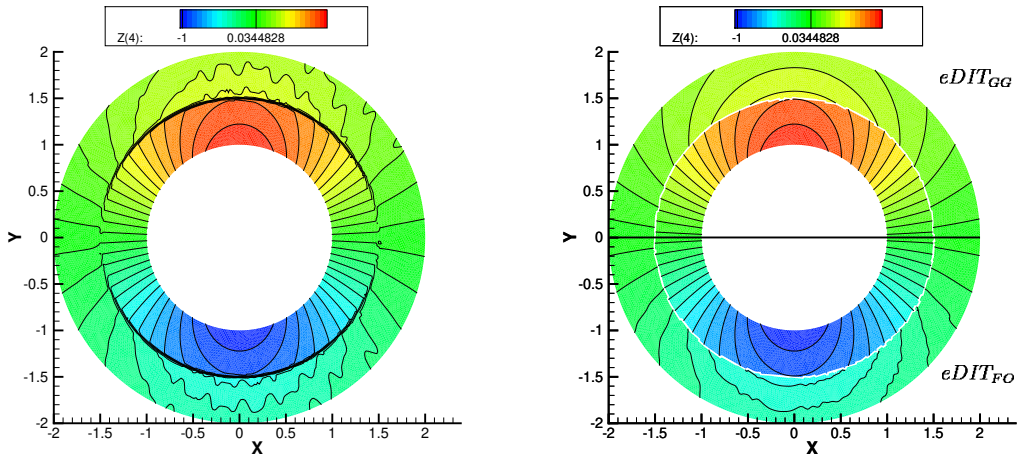
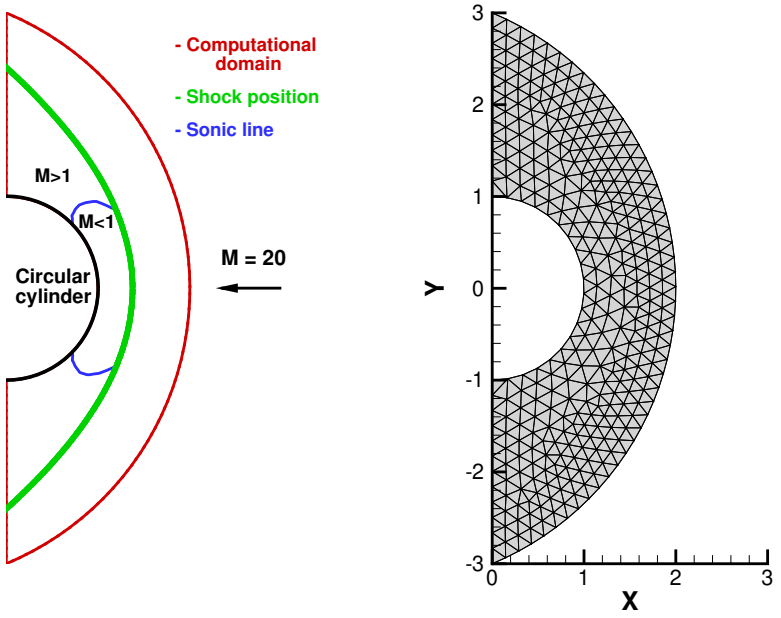


Figure 5.13: Transonic source flow: grid-convergence analysis on the flux balance between $\tilde{\Gamma}_U$ and $\tilde{\Gamma}_D$.



(a) *SC* (b) $eDIT_{GG}$ vs. $eDIT_{FO}$

Figure 5.14: Transonic source flow: iso-contours comparison on *grid level 2* w.r.t. $Z_4 = \sqrt{\rho}v_y$.



(a) Sketch of the computational domain.

(b) Detail of the level 0 unstructured mesh.

Figure 5.15: Hypersonic flow past a blunt body.

forms ahead of the blunt body. Moreover, as shown in Figure 5.15(a), the flow-field is much more complex than that examined in Section 5.3.1, due to the presence of a stagnation region within a subsonic pocket, as well as a smooth re-acceleration of the flow to supersonic conditions along the body. Even though no analytical solution is available for the entire flow-field, we know that total enthalpy, H, is preserved along streamlines at steady state. For a constant profile of $H = H_\infty$ ahead of the shock, this leads to an exact solution featuring a homogeneous total enthalpy field, thus the condition $H = H_\infty$ can be used to study the convergence behavior of the various combinations of numerical schemes and shock-modeling options. Starting from a coarse mesh (shown in Figure 5.15(b)) obtained with the `delaundo` frontal/Delaunay mesh generator [226, 227], we created three levels of nested refined meshes with characteristics summarized in Table 5.2 for both the background and computational grids.

Table 5.2: Hypersonic flow past a blunt body: characteristics of the background and computational meshes used to perform the grid-convergence tests.

	Background grid		*Both grids*	*Computational grid*	
Grid level	Grid-points	Triangles	h	Grid-points	Triangles
0	351	610	0.16	351	535
1	1,311	2,440	0.08	1,311	2,292
2	5,061	9,760	0.04	5,061	9,460
3	19,881	39,040	0.02	19,881	38,439

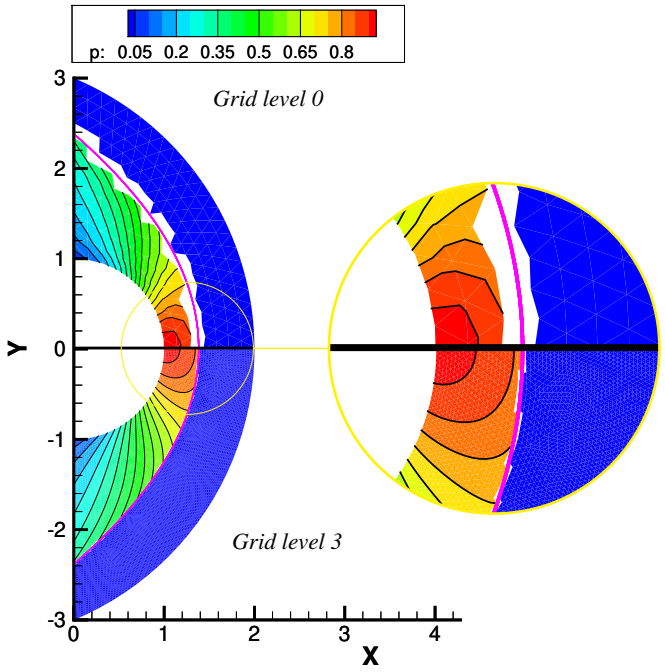
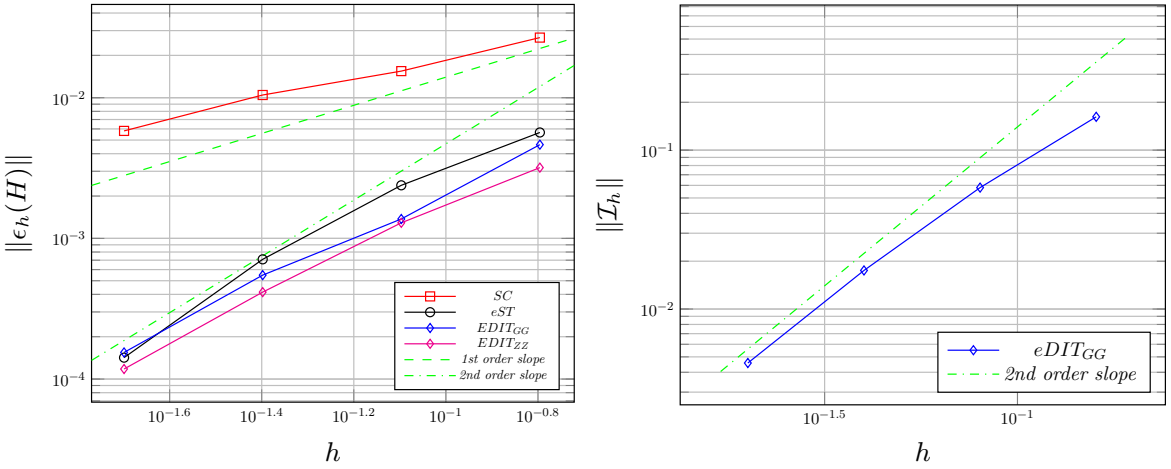


Figure 5.16: Hypersonic flow past a blunt body: comparison between the solutions obtained with $eDIT_{GG}$ on *Grid level 0* and *Grid level 3* in terms of pressure iso-lines.

The different extrapolated tracking methods discussed in the previous sections are expected to provide orders of convergence similar to those observed for the source flow of Section 5.3.1, as well as smooth and clean results with very low numerical perturbations, even on the coarsest grids; this latter aspect is illustrated in Figure 5.16, where the pressure iso-contour lines computed using $eDIT_{GG}$ on the level 0 (coarsest) and level 3 (finest) grids are mutually compared. Figure 5.25a shows the grid-convergence analysis pointing out again that the different extrapolation techniques provide error levels relatively close, and a second order slope, while the captured result converges with a less than first order rate, and errors of one or two orders of magnitude larger. The patch super-convergent gradient recovery (ZZ) provides a better result for this case. Figure 5.25b shows the effect of a second order extrapolation, $eDIT_{GG}$, on the flux balance within the cavity for this test-case. As discussed in Section 5.2 we recover the error associated to the second order extrapolation. Similar results for other second order extrapolations are straightforward to obtain and therefore not included in Figure 5.25b.

Figure 5.18 shows the total-enthalpy discretization error over the entire computational domain for the two solutions obtained with SC and $eDIT_{GG}$ and two different grid-levels: the coarsest and the finest. The maximum value of the error is also explicitly given in



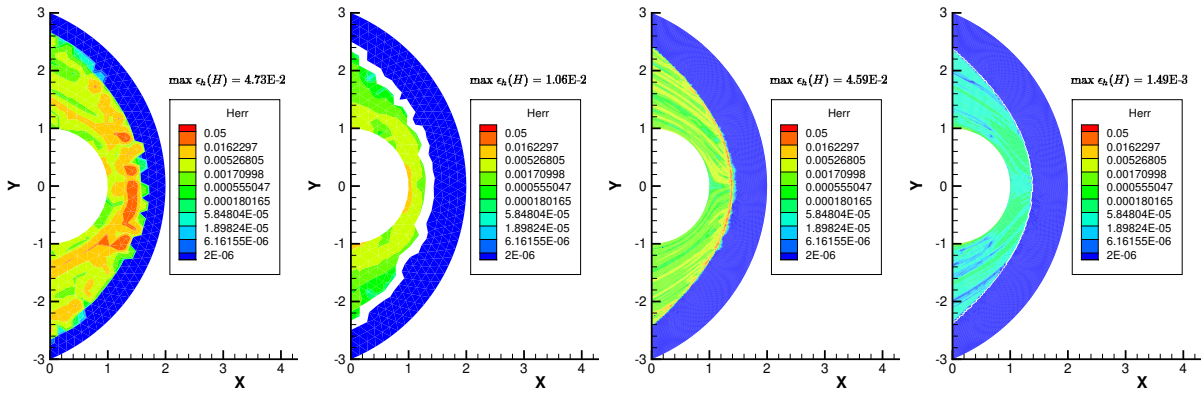
(a) Total enthalpy error

(b) Flux balance

Figure 5.17: Hypersonic flow past a blunt body: grid-convergence analysis.

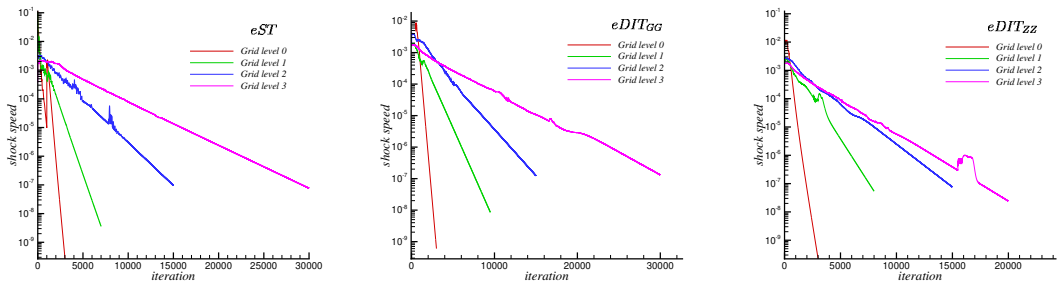
each of the four frames. While confirming the lower error levels on the coarse mesh, and the rapid error convergence obtained with second-order shock-tracking, the four frames of Figure 5.18 allow to visually highlight the substantial difference in error generation. Indeed, in both captured solutions (Figure 5.18(a) and 5.18(c)), the largest error is generated along the shock. Conversely, for the two $eDIT_{GG}$ calculations the numerical error is essentially connected with the wall boundary condition, and undoubtedly related to the entropy generated at the stagnation point and advected downstream. The $eDIT_{ZZ}$ solution is virtually identical and not discussed for brevity.

As a final note, we remark that total enthalpy is a conserved variable if and only if the time derivative in Equation (1.30) vanishes. This means that the convergence to steady state plays a major role in allowing to correctly measure the decay rate of the discretization error. Convergence to steady-state is an important aspect of this type of methods, which is why we also report on Figure 5.19 the iterative convergence of the $eDIT$ algorithm when starting from a captured solution. In all our calculations we have set as a stopping criterion a threshold on the norm of the shock speed of $\sim 10^{-7}$. As Figure 5.19 shows, this level is reached quite monotonically in all the computations. This despite the shock crossing some nodes during the iterations, which produces a change in topology of the surrogate discontinuities and of the computational domain which can be seen in some of the peaks appearing locally in the iterative convergence plots. Similar convergence curves are often hard to obtain with non-linear shock capturing methods.



(a) *SC*, *Grid level 0* (b) $eDIT_{GG}$, *Grid level 0* (c) *SC*, *Grid level 3* (d) $eDIT_{GG}$, *Grid level 3*

Figure 5.18: Hypersonic flow past a blunt body: discretization error of the total enthalpy, $\epsilon_h(H)$, obtained with *SC* and $eDIT_{GG}$ over the entire computational domain for *Grid level 0* and *Grid level 3*.



(a) *eST* (b) $eDIT_{GG}$ (c) $eDIT_{ZZ}$

Figure 5.19: Hypersonic flow past a blunt body: Pseudo-time shock convergence.

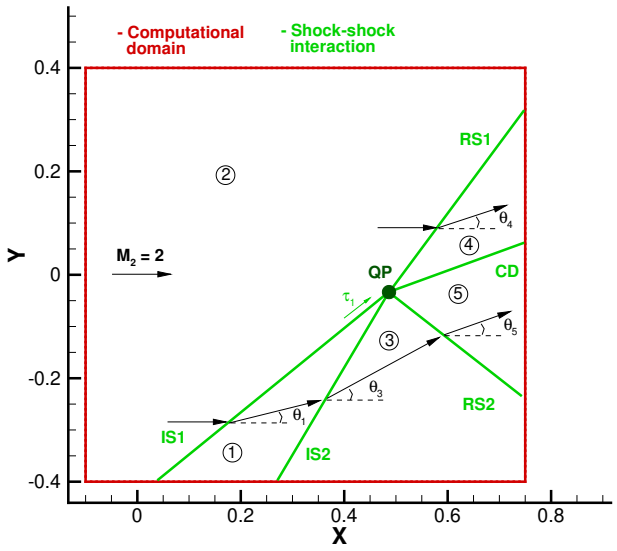


Figure 5.20: Interaction between two shocks of the same family: sketch of the flow.

5.3.3 Interaction between two shocks of the same family

The present and subsequent test-cases address the interaction among different discontinuities, a kind of flow-topology which had not been addressed in the first journal appearance [89] of the *eST* algorithm and thus represent one of the key novelties of [90]. We consider here the interaction between two oblique shocks colliding and giving rise to a five-waves interaction. As shown in the sketch of Figure 5.20, the two incident shocks (IS1 and IS2) of the same family interact in a quadruple point, referred to as QP in the figure, generating three reflected discontinuities: a strong reflected shock (RS1), with jumps of magnitude larger than the two incident ones; a contact discontinuity (CD); a weak reflected shock (RS2), with jumps so small that it could be considered as a Mach wave. More in general, depending on the free-stream Mach number and the flow deflection angles caused by the incident shocks, RS2 can be either a weak expansion or compression wave. With reference to Figure 5.20, in this case the free-stream Mach number and the flow deflection angles are: $M_2 = 2$, $\theta_1 = 10°$ and $\theta_2 = 20°$ which can be shown to lead to states 4 and 5 characterized by $p_4 = p_5 = 2.822$, $M_4 = 1.218$, $M_5 = 1.28$ and $\theta_4 = \theta_5 = 19.87°$. For this choice of parameters, RS2 is so weak that we decided not to track this wave since its resolution has negligible overall effect on the flow.

Simulations of the interaction have been performed using a Delaunay triangulation containing 7,921 nodes and 15,514 triangles. As before, the same mesh is used to perform

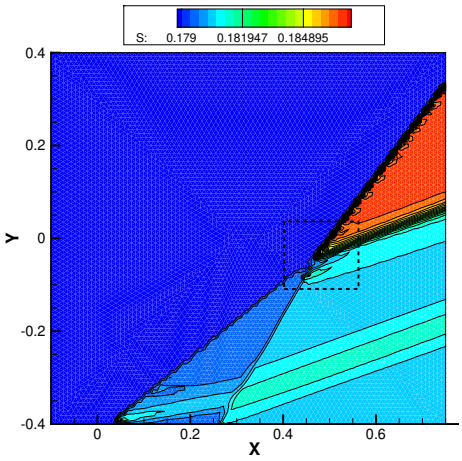
the SC simulation and as a background grid for the $eDIT_{GG}$ computations. Results with the $eDIT_{ZZ}$ are virtually identical to the latter, and not discussed. As mentioned in Section 5.1.5, the interaction points need to be modeled explicitly on a case by case basis. If only some of the discontinuities meeting at the interaction point are tracked, the only possibility is to solve independent algebraic problems arising from the jump conditions for each discontinuity, and use some approximate formula for QP. For the type of interaction considered here, we have used the following relation to compute the velocity of the interaction point QP:

$$\mathbf{w}_{QP} = \mathbf{w}_{IS1} + (\mathbf{w}_{IS2} \cdot \boldsymbol{\tau_1})\, \boldsymbol{\tau_1} \tag{5.24}$$

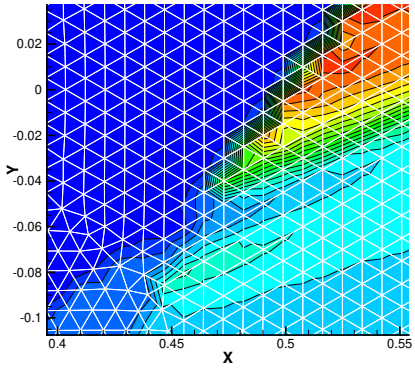
where $\boldsymbol{\tau_1}$ is the vector tangential to IS1 (see Figure 5.20a). Equation (5.24) is similar, but not identical, to the formula proposed in [78] for the same purpose. We compare in Figure 5.21 three sets of results: *i*) a fully captured solution (top row of Figure 5.21); *ii*) a hybrid solution in which CD is captured, and all the shocks (IS1, IS2 and RS1) are tracked (central row of Figure 5.21); *iii*) a fully tracked result (bottom row of Figure 5.21) where also CD is fitted. First of all, this result shows that it is possible to track several discontinuities and capture others, all involved in the same interaction. Compared to the fully captured solution, this hybrid simulation with captured CD provides a much cleaner entropy field, as seen comparing the top and central rows of frames in Figure 5.21. However, we can also see a small anomaly due to the capturing of the CD, which is still observable in part of the flow downstream of the reflected shock RS1: see the close up of Figure 5.21(d). These spurious disturbances are removed by adding the contact discontinuity to the set of discontinuities to be tracked by the algorithm, as visible on the bottom row of results of Figure 5.21. Note that despite its apparent simplicity, the results obtained are surprisingly good as this is a difficult simulation. Indeed, on a mesh as coarse as the one used here, the elements crossed by the discontinuity generate a relatively large cavity around the interaction point. This is clearly visible for example in Figure 5.21(f). The extrapolation strategy proposed allows to handle this delicate geometrical situation.

5.3.4 Shock-shock interaction: interaction between two shocks of different families

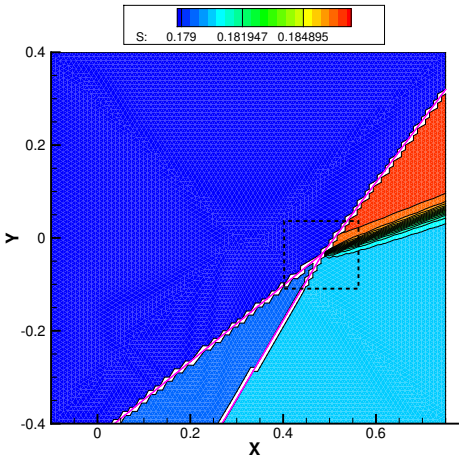
The first shock-shock interaction considered herein is the interaction between two shock of different families (see sketch in Figure 5.22a). In this case, a uniform supersonic stream of air, $M_1 = 3$, is crossed by two incident shocks (IS1 and IS2) of different intensities, which respectively deflect the undisturbed flow of state 1 of $\theta_2 = -15°$ and $\theta_3 = 20°$.



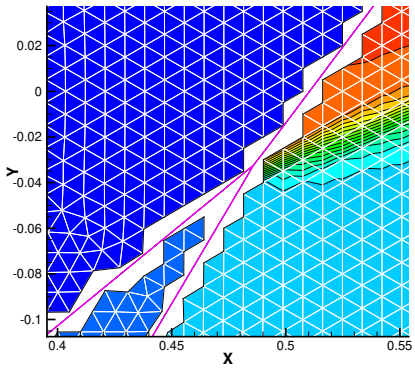
(a) SC solution



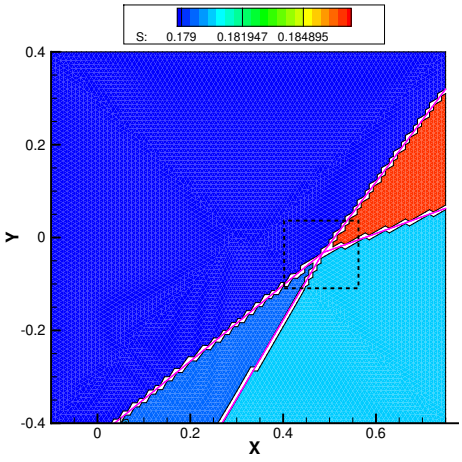
(b) close-up around the QP



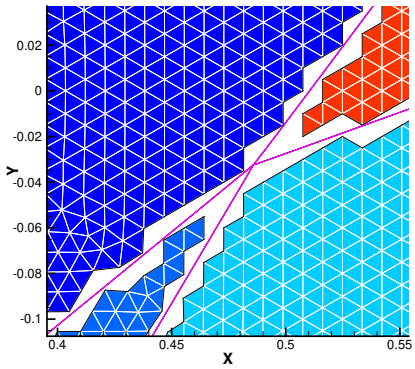
(c) Hybrid $eDIT_{GG}$: the CD is captured.



(d) close-up around the QP

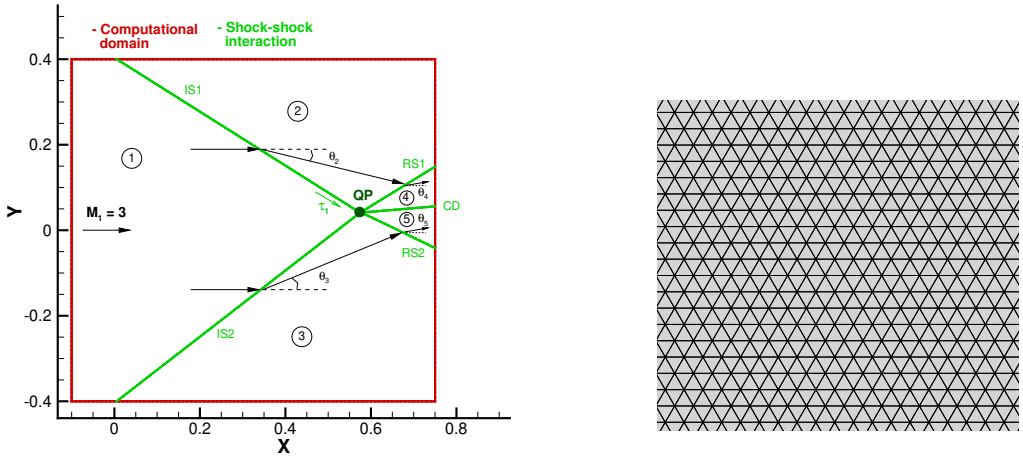


(e) Hybrid $eDIT_{GG}$: the CD is tracked.



(f) close-up around the QP

Figure 5.21: Interaction between two shocks of the same family: numerical solutions (comparisons in terms of $S = p\rho^{-\gamma}$).

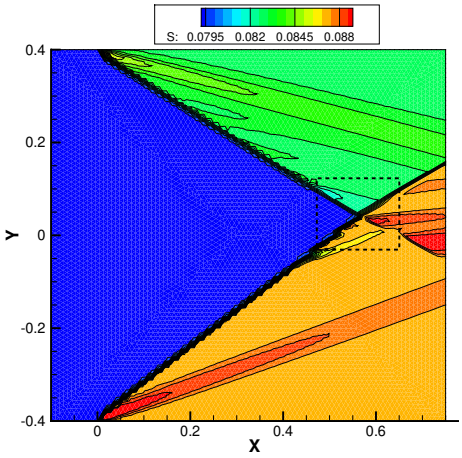


(a) Sketch of the computational domain

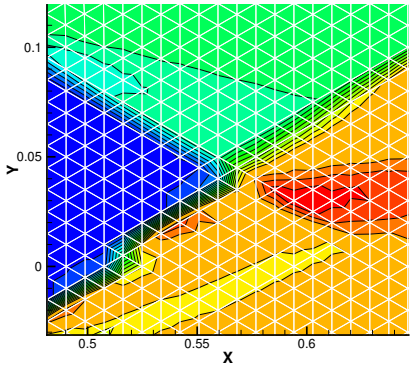
(b) Background mesh portion

Figure 5.22: Interaction between two shocks of different families.

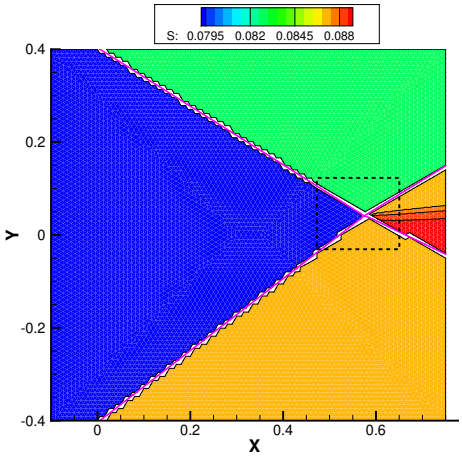
IS1 and IS2 interact in the point QP, named quadruple point, giving rise to two reflected shocks (RS1 and RS2) and a contact discontinuity (CD). RS1 and RS2 deflect again that air-stream of state 2 and 3 obtaining state 4 and 5, characterized by the same flow deflection angle $\theta_4 = \theta_5 = 4.796°$. The mesh used as a background grid for the *eDIT* algorithm and to compute the *SC* solution, shown in Figure 5.22b, is made up by 7,921 nodes and 15,514 triangles and it was generated through the `delaundo` mesh generator (see Ref. [226, 227] for further information). Firstly, it is fundamental to describe how to model the quadruple point and compute its motion. This can be done either by following an analytical procedure, that computes all the state of the quadruple point and its speed (see [237]), or by using an empirical formula to drive the interaction point (see Ref. [78] for more information). The shortcoming of the former is that computing an interaction requires the solution in all the states making up the interaction point (usually it concerns that a big non-linear systems have to be solved by means of a numerical procedure like a Newton-Raphson algorithm) whereas the availability of such formulae would allow to avoid to spend this computational time. When working with the unstructured discontinuity fitting *SF*, it is highly advised to update each state of the interaction point with the right one coming from the computation of all states because the mesh is adapted to the point. Instead, in this case, since *eDIT* does not re-mesh around discontinuities and interaction points, the triangles that should occupy the space around the interaction point are removed. Therefore, simple approximations could be used to predict the evolution of the special point that can be then adjusted depending on the number of discontinuity that we decide to track.



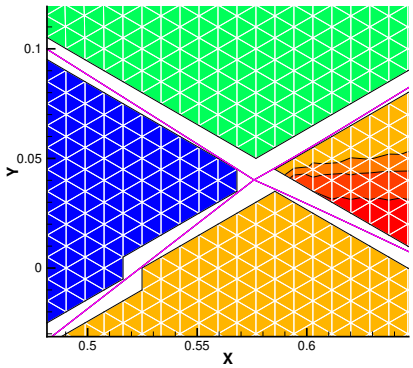
(a) SC solution



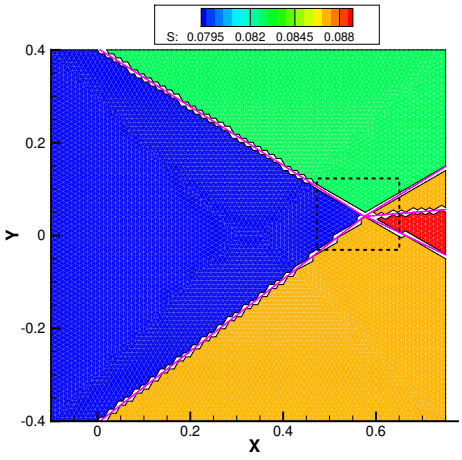
(b) close-up around QP



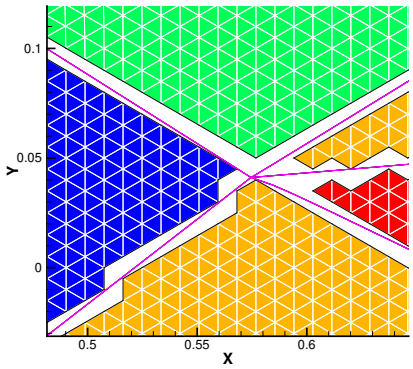
(c) Hybrid $eDIT$ solution



(d) close-up around QP

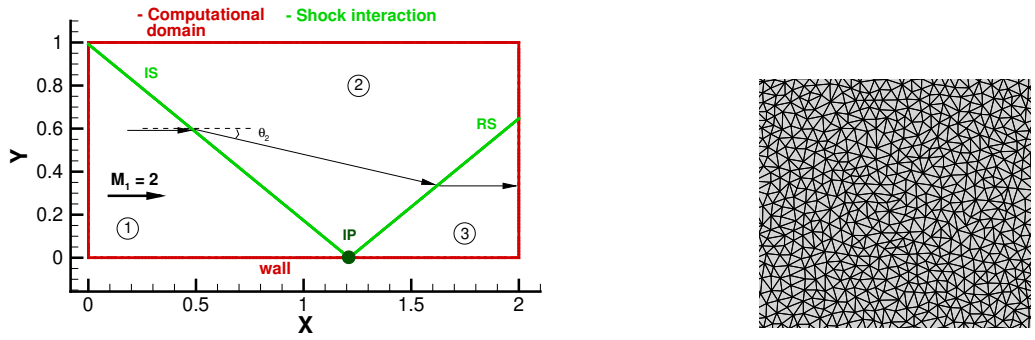


(e) Fully-fitted $eDIT$ solution



(f) close-up around QP

Figure 5.23: Interaction between two shocks of different families: numerical solutions (comparisons in terms of $S = p\rho^{-\gamma}$)



(a) (b)

Figure 5.24: Regular reflection: a) sketch of the shock-wall interaction; b) background mesh portion

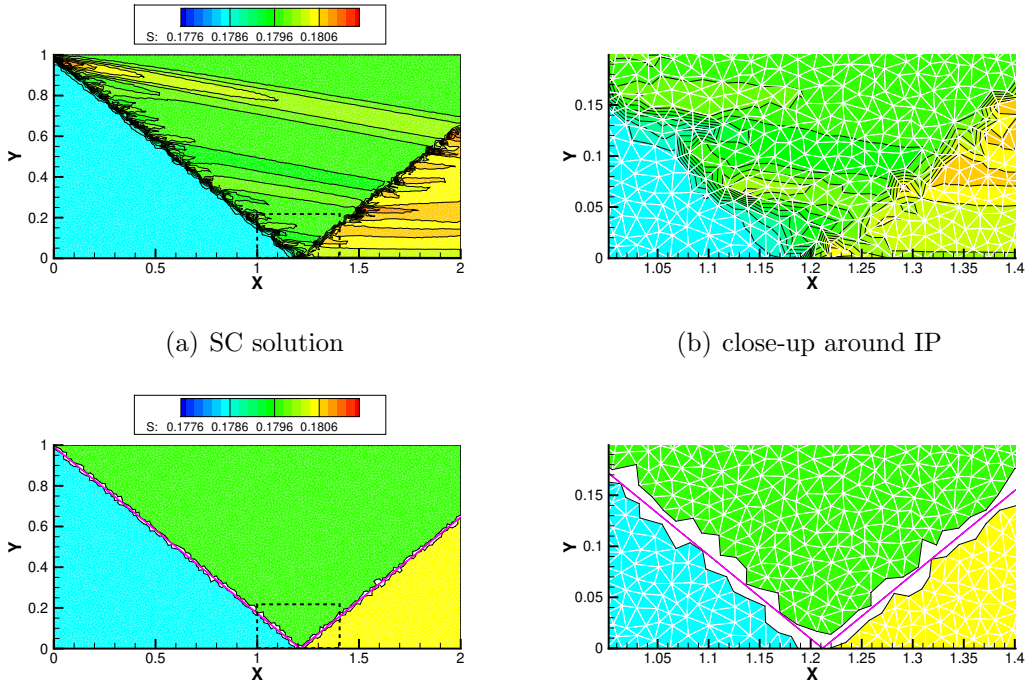
when the fully-fitted simulation is performed the motion of the quadruple point could be still being computed through the solution of the analytical problem, which permits a better prediction of the states downstream QP. Figure 5.23 shows the numerical solutions obtained with three different approaches: SC, hybrid $eDIT$, fully-fitted $eDIT$. On the left hand side of Figure 5.23 the entropy field is shown whereas, on the right hand side, a close-up of the dashed square is displayed along with the shock poly-line and the computational grid. For this study, the QP's speed has been computed by means of the relation displayed below, which mainly depends on the motion of IS1 and its tangential direction:

$$\mathbf{w}_{QP} = \mathbf{w}_{IS1} + ((\mathbf{w}_{IS2} + \mathbf{w}_{RS1} + \mathbf{w}_{RS2}) \cdot \boldsymbol{\tau}_{\mathbf{1}}) \cdot \boldsymbol{\tau}_{\mathbf{1}} \tag{5.25}$$

where $\boldsymbol{\tau}_{\mathbf{1}}$ is the tangential vector to IS1 (see Figure 5.22).
For the first time, in Figure 5.23e, we show a fitted solution with a contact discontinuity. It should be also noticed that, in Figure 5.23c, capturing CD does not imply a notable degradation of the entropy field but being able to track that too allows an even better prediction of the overall solution.

5.3.5 Shock-wall interaction: regular reflection

The first test-case considered concerns the regular reflection of a straight oblique shock onto an horizontal wall. Figure 5.24a shows a uniform flow, $M_1 = 2$, being deflected by a weak oblique shock (IS) of an angle $\theta_2 = -10°$. In particular, when IS impinges, in point IP, on a straight wall, it gives rise to a reflected shock (RS) of the opposite family, which makes again parallel to the wall direction. Two different approaches, SC and fully-fitted $eDIT$, have been used to simulate the present test-case in order to point



(a) SC solution

(b) close-up around IP

(c) Fully-fitted *eDIT* solution

(d) close-up around IP

Figure 5.25: Regular reflection: numerical solutions (comparisons in terms of $S = p\rho^{-\gamma}$)

out the advantages and shortcomings of both *SC* and *eDIT*. The mesh used in the *SC* simulation, which represents the background mesh in the *eDIT* simulations, consists in 10,281 nodes and 16,016 triangles and has been built using the `triangle` mesh generator. It should be noted that any interaction point, such as point IP, has to be modeled properly. For this case, IP's normal component of the velocity is projected along the straight wall to make it move parallel to the wall (see Figure 5.26). Figure 5.25 shows the numerical results obtained with the two approaches mentioned before. For any solution displayed on the left side, a close-up of the interaction point, within the dashed square, is shown on the right side with the correspondent computational mesh and shock-mesh represented with, respectively, white triangles and purple poly-lines. Several considerations can be inferred by looking at Figure 5.25 regarding the results obtained. The first thing that stands out above everything is the presence of notable spurious disturbances, coming from the captured shocks, that influence the whole entropy field. It should be noticed from Figure 5.24b that the mesh generated to perform the SC simulation is highly irregular making impossible for it to align with the shock patterns. Although better meshes can be created or anisotropic mesh adapters can be employed to obtain better results, the use of such grids allows to better point out the capabilities of the presented algorithm. Carrying out a fully-fitted simulation permits to get rid of the spurious errors shown in

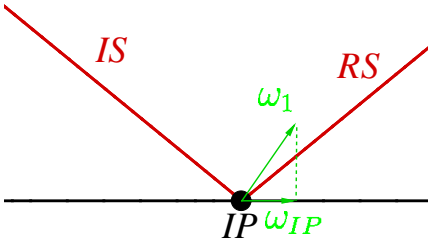


Figure 5.26: Regular reflection: speed vector computation for the interaction point where IS impinges the wall.

Figure 5.25a and sweep away all the problems introduced by the capture of the shocks finally obtaining a spotless solution in the whole computational domain. Indeed, since only uniform states are present, the fully-fitted simulation is able to recover the exact solution.

5.3.6 Shock-wall interaction: Mach reflection

An oblique shock that impinges on a straight wall gives rise to either a regular or a Mach reflection, depending on the combination of free-stream Mach number, M_1, and flow deflection angle, θ_2, that the free-stream flow undergoes while passing through the oblique shock (hereafter also referred to as the incident shock). Whenever θ_2 is larger than the maximum deflection that the supersonic stream behind the incident shock can sustain, a Mach reflection takes place. As sketched in Figure 5.27, a Mach reflection consists in a fairly complex three-shocks system (the incident shock, IS, the reflected shock, RS, and the Mach stem, MS) interacting in the triple point, TP, from which a contact discontinuity, CD, also arises. Note that differently from the test-case addressed in Section 5.3.3, here not all the regions surrounding TP involve uniform flow. Moreover, CD presents a slight angle, which makes it in general not mesh-aligned.

For the chosen setting, $M_1 = 2$ and $\theta_2 = 14°$, we consider four different types of simulations:

1. fully captured;

2. an extrapolated shock-tracking setting in which the Mach stem (MS) and the reflected shock (RS) are tracked as a single discontinuity;

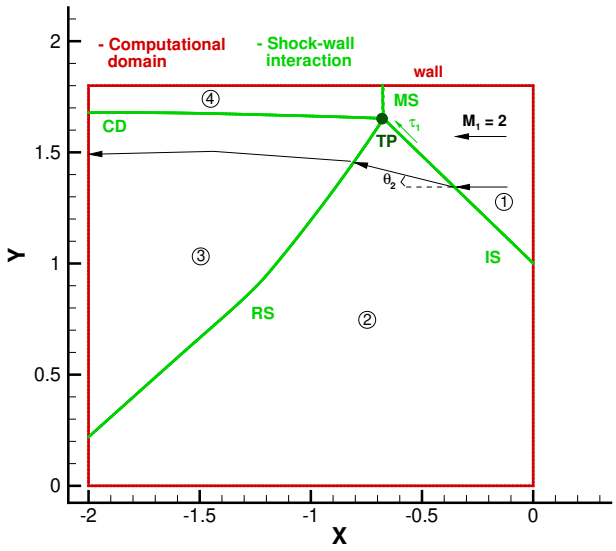


Figure 5.27: Mach reflection: sketch of the flow.

3. a hybrid setting in which only the shocks are tracked, but CD is captured;

4. full-fledged discontinuity-tracking for all shocks plus the CD.

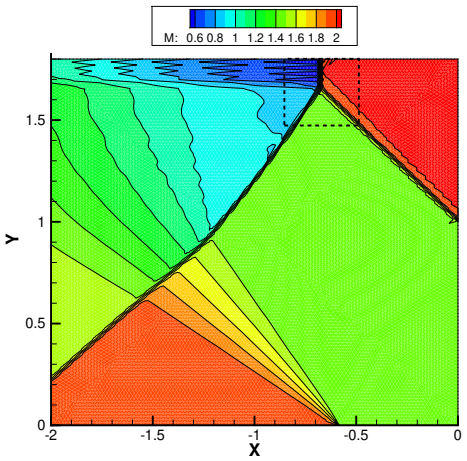
Note that no modeling of the triple point TP is necessary in configuration 2 which involves a unique shock-mesh. In case 3 we need to provide an explicit model for the triple-point. Not having enough equations to compute the TP, because the CD is captured, rather than being tracked, we need to resort to a heuristic approach. As in the case of Section 5.3.3, a nonlinear algebraic problem is solved independently for each discontinuity, and for the TP velocity we use the following simplified relation:

$$\mathbf{w}_{TP} = \mathbf{w}_{IS} + (\mathbf{w}_{MS} \cdot \boldsymbol{\tau_1})\, \boldsymbol{\tau_1} \tag{5.26}$$

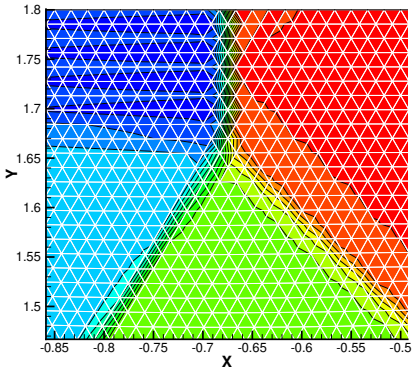
where $\boldsymbol{\tau_1}$ is the unit vector tangential to the IS (see Figure 5.27a).
Equation (5.26) is similar, but not identical, to the formula used in [78] for the same purpose.

Finally, in configuration 4, all four states surrounding the TP and its unknown velocity, $\mathbf{w}_{TP}$, can be coupled into a single non-linear system of algebraic equations, whose solution provides updated values downstream of the RS and MS as well as $\mathbf{w}_{TP}$. Full details are given in [177].

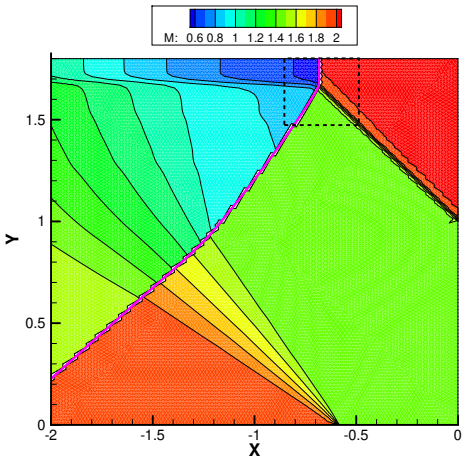
The results are arranged in four rows in Figures 5.28 and 5.29, showing on the left an overview of the Mach contours, and on the right a zoom of the TP. Compared to the SC calculation (top row in Figure 5.28), tracking MS and RS (second row in Figure 5.28)



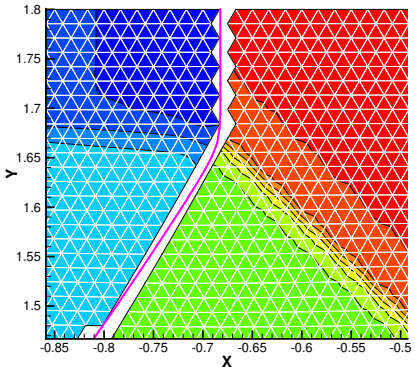
(a) SC solution



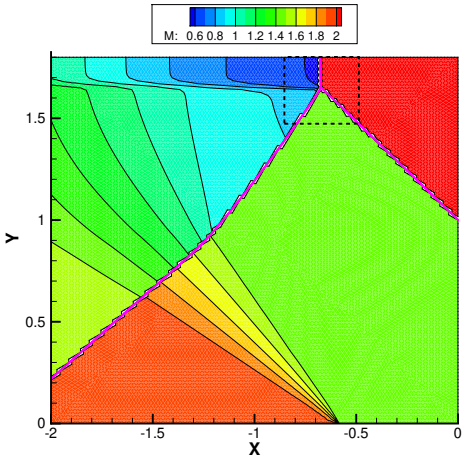
(b) close-up around the TP



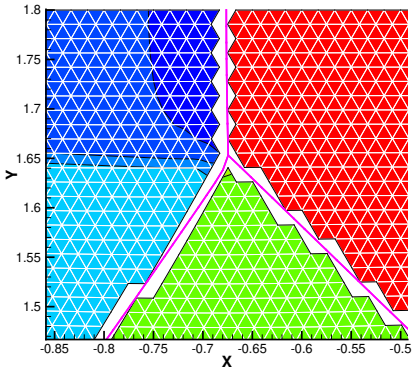
(c) Hybrid $eDIT_{GG}$: configuration 2



(d) close-up around the TP

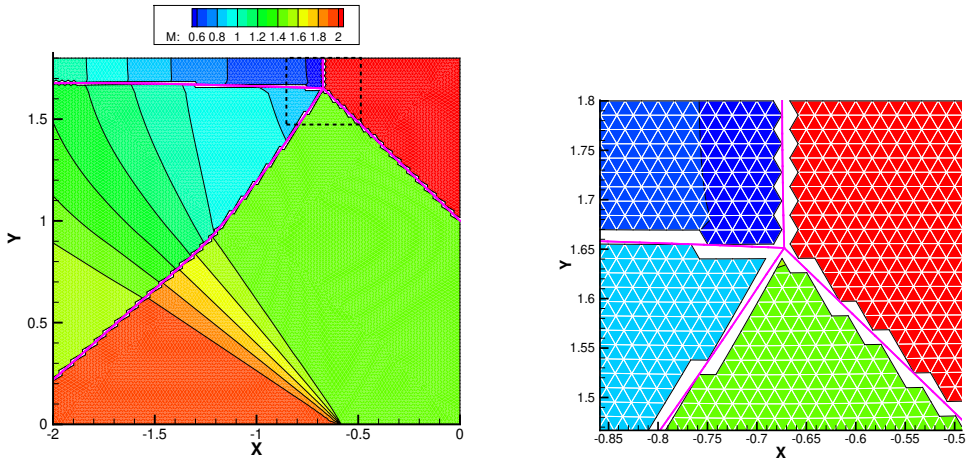


(e) Hybrid $eDIT_{GG}$: configuration 3



(f) close-up around the TP

Figure 5.28: Mach reflection: the Mach number iso-lines of the numerical solutions.



(a) Full-fledged $eDIT_{GG}$ solution

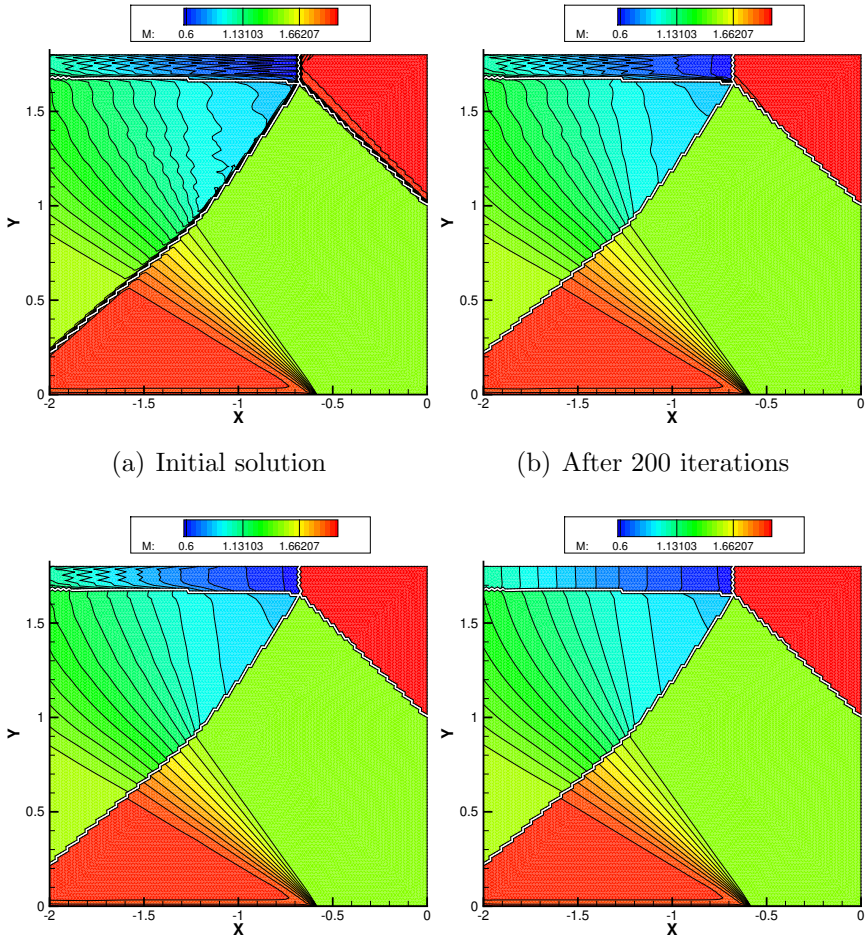
(b) close-up around the TP

Figure 5.29: Mach reflection: the Mach number iso-lines of the full-fledged (configuration 4) numerical solutions.

already provides an enormous improvement in the quality of the solution. The CD remains however poorly captured on this coarse mesh. Adding the IS to the tracking set (third row in Figure 5.28) leads to a cleaner flow, which is visible in the nicer straight contour lines downstream of RS. However, CD is still poorly resolved. Finally, in full-fledged *eDIT* mode of Figure 5.29 we are able to resolve the CD in a single row of elements. A small kink in the contour lines in the non-constant region around CD is still visible. We assume these to be due to perturbations arising in correspondence of the steps in the surrogate CD and propagating downstream. A qualitative view of the error cleaning and pseudo-time convergence of the full-fledged simulation is displayed in Figure 5.30. More pseudo-time convergence results are available in Appendix A.2.

5.3.7 Supersonic channel flow

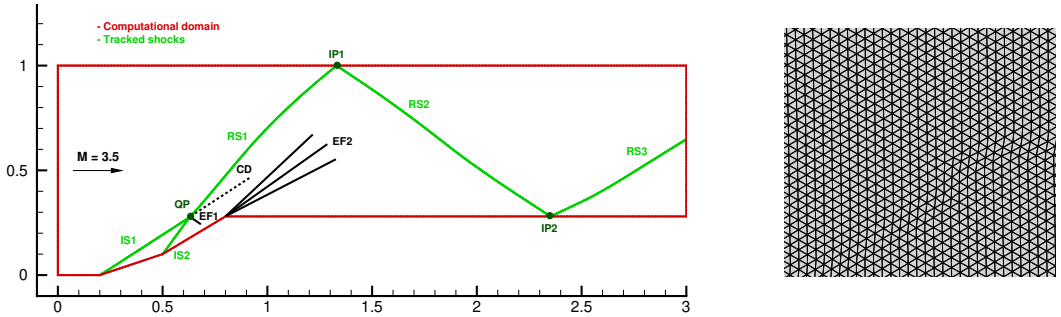
The last test-case considered consists in the supersonic, $M_\infty = 3.5$, flow in a planar channel, whose variable-area geometry is shown in Figure 5.31(a) and reported in [310]: the two constant-area portions of the duct are joined through a double ramp. The flow pattern is as follows: two oblique, straight shocks of the same family, labeled IS1 and IS2, originate at the two convex corners of the ramp and their interaction gives rise to a wave configuration already explored in Section 5.3.3 where IS1, IS2, a new shock RS1, a slip-stream CD and an expansion fan EF1 meet at the interaction point QP. A second, stronger expansion fan EF2 takes place at the concave corner of the lower wall and interacts with RS1, which bends before being reflected from the upper wall. The reflected shock RS2 is again reflected by the lower wall and leaves the duct as shock RS3.



(a) Initial solution (b) After 200 iterations

(c) After 400 iterations (d) Converged solution

Figure 5.30: Mach reflection: pseudo-time convergence of the full-fledged $eDIT_{GG}$ simulation (Mach number iso-contour lines)



(a) Sketch of the computational domain and flow-pattern. (b) Detail of the background triangulation.

Figure 5.31: Supersonic channel flow.

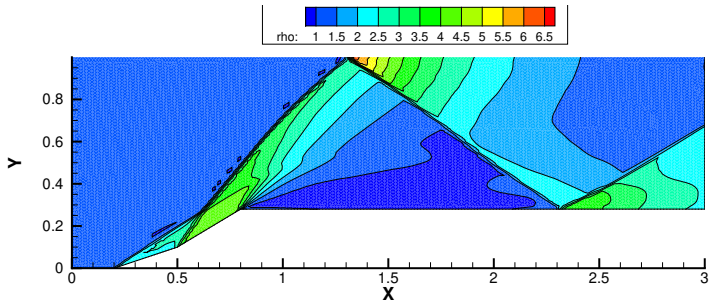
The goal of the present test-case consists in assessing that the different features that have been implemented within the $eDIT$ algorithm, already described in the previous sections, work well also when combined used to simulate a fairly complex shock-pattern, such as the one illustrated in Figure 5.31(a). To improve simulation fidelity, we track as many discontinuities as we can, i.e. the three shock waves IS1, IS2 and RS1, which meet at QP, and the two regular reflections made up by RS1, RS2 and RS3 taking place on the upper and lower walls. Only the CD has been captured.
When simulating this test-case with the $eDIT$ algorithm, two different models have been employed: the one already described in Section 5.3.3 for QP, where the interaction between shocks of the same family takes place, and a different one for points IP1 and IP2, where a regular reflection takes place on the upper, resp. lower wall. As shown in Figure 5.26, the velocity of points IP1 and IP2 is set equal to the component tangential to the wall of the corresponding incident shock velocity.

Simulations have been performed using both SC and $eDIT_{GG}$. The unstructured grid used in the SC calculation and as the background triangulation in the $eDIT_{GG}$ calculation, which is made up of 12,417 grid-points and 24,310 nearly equilateral triangles, has been generated using the frontal mesh generator of the `gmsh` software [144]. A detail of the mesh is shown in Figure 5.31(b).

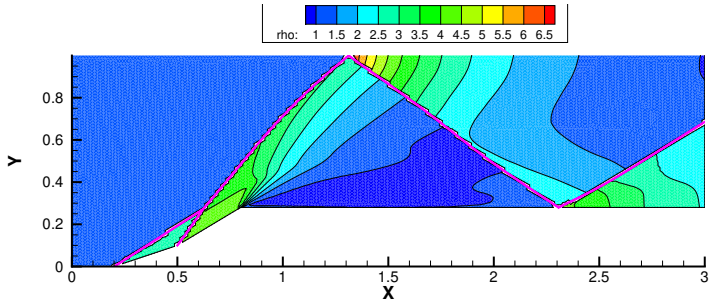
The comparison between the two sets of calculations is reported in Figures 5.32 and 5.33, were density and Mach number iso-contours are respectively displayed. It can be seen that the capture of the discontinuities, see Figures 5.32(a) and 5.33(a), gives rise to spurious disturbances (in particular downstream of RS2) which are not present in the $eDIT_{GG}$ calculation (Figures 5.32(b) and 5.33(b)) pointing out a globally cleaner flow-field featuring smoother iso-contours. The oscillations that occur downstream of RS2 are more clearly visible in Figure 19, which shows the dimensionless pressure profiles along the upper and lower walls. It is evident that by resorting to a shock-tracking approach, not only we are able to get rid of the major oscillation introduced by SC and exactly represent the discontinuities as having zero-thickness, but we also get a better prediction of wall pressure peaks, which are smoothed out in the SC calculation.

5.4 Chapter summary

A new technique recently proposed in [89] to simulate flows with shock waves has been further improved [90] to make it capable of dealing with different kinds of discontinuities (both shock waves and contact discontinuities) as well as shock-shock and shock-wall interactions, thus opening the possibility to compute complex flows. Moreover, since the

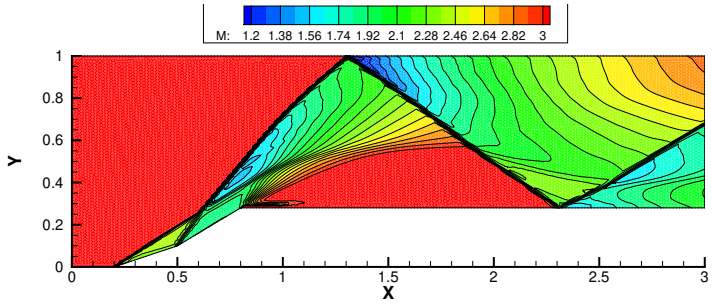


(a) *SC* solution

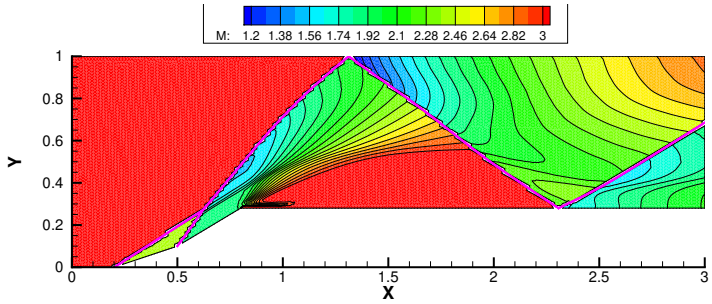


(b) $eDIT_{GG}$ solution

Figure 5.32: Supersonic channel flow: dimensionless density ρ/ρ_∞ iso-contours.

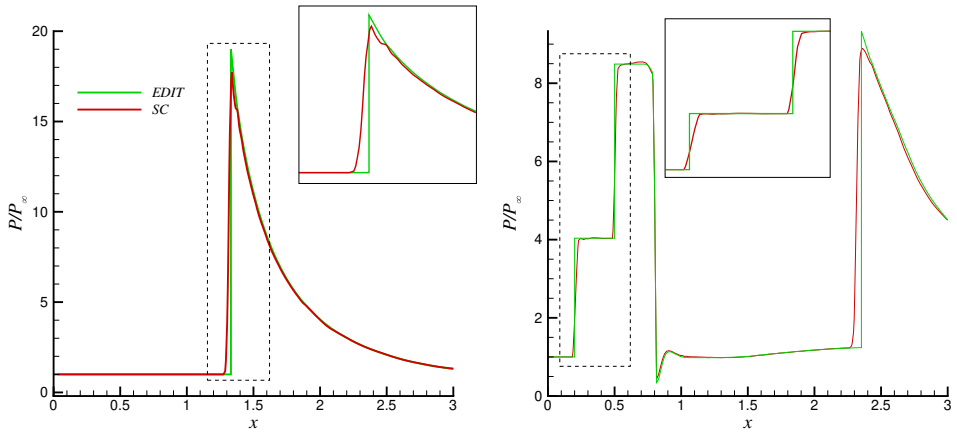


(a) *SC* solution



(b) $eDIT_{GG}$ solution

Figure 5.33: Supersonic channel flow: Mach iso-contours.



(a) Upper wall. (b) Lower wall.

Figure 5.34: Supersonic channel flow: dimensionless pressure distribution along the walls of the channel computed by means of SC and $eDIT$.

algorithm can be run in hybrid mode, it is also possible to study complicated flows by tracking some of the discontinuities and leaving others to be captured. The proposed technique provides genuinely second-order-accurate results even for flows featuring very strong shocks, without the complexity of the re-meshing/adaptation phase of previous fitting approaches [235, 304, 101]. For this reason, the present algorithm is substantially independent from the data structure of the gasdynamic solver and, therefore, it can be applied with small modifications to both cell-centered and vertex-centered solvers on both unstructured and structured grids [24].

Part III

Arbitrary High-Order Structure-Preserving Finite Volume Schemes for Hyperbolic Problems with Source Terms

Chapter 6

Well-Balanced Positivity-Preserving Schemes without Restrictions on the CFL

In this chapter, we develop and present an arbitrary high order well-balanced [34] finite volume (explained in Section 2.2) WENO (see Section 2.2.2 for further details) method combined with the modified Patankar Deferred Correction [234] (mPDeC) time integration method, for the shallow water equations. To obtain a positive WENO spatial reconstruction, a positive limiter must be used [306, 247]. Unfortunately, such limiters have very strict CFL constraints which become worse as the order of the scheme goes up and can only be used together with SSP Runge-Kutta schemes (see Section 2.5.1 for further details). Due to the positivity-preserving property of mPDeC (which is a modification of the DeC method explained in Section 2.5.2), the resulting scheme is unconditionally positive for the water height, meaning that the aforementioned CFL restriction no longer influences the stability region of the fully discrete method. To apply the mPDeC approach, we have to interpret the spatial semi-discretization of the equation for the water height as a production-destruction system [86]. Only small modifications inside the classical WENO implementation are necessary to do so. We herein focus on a fifth order method, to demonstrate the good performances of the new method and verify the theoretical properties; but the method is of arbitrary order. In Appendices A.3 and A.4, we describe in details the WENO reconstruction and apply it for WENO5 with 4-points Gaussian quadrature rule, which, up to our knowledge, is not available in literature.

6.1 Well-balanced modification of the standard Finite Volume method

In order to achieve Well-Balancing with respect to the (eventually dry) lake at rest steady state, in this work we coupled the WENO formulation with a simple modification firstly introduced in [34]. The modification consists in recasting the original problem into an equivalent one in terms of the deviation of the sought solution $\mathbf{u}$ from the reference solution $\tilde{\mathbf{u}}$ which must be preserved. In the particular case in which a steady solution ($\frac{\partial \tilde{\mathbf{u}}}{\partial t} = 0$) must be preserved, the modification leads to the new problem

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}_{i,j}+&\frac{1}{\Delta x}(\mathbf{F}_{i+1/2,j}(\mathbf{u})-\mathbf{F}_{i-1/2,j}(\mathbf{u}))-\frac{1}{\Delta x}(\mathbf{F}_{i+1/2,j}(\tilde{\mathbf{u}})-\mathbf{F}_{i-1/2,j}(\tilde{\mathbf{u}}))+\\ &\frac{1}{\Delta y}(\mathbf{G}_{i,j+1/2}(\mathbf{u})-\mathbf{G}_{i,j-1/2}(\mathbf{u}))-\frac{1}{\Delta y}(\mathbf{G}_{i,j+1/2}(\tilde{\mathbf{u}})-\mathbf{G}_{i,j-1/2}(\tilde{\mathbf{u}}))=\\ &\mathbf{S}_{i,j}(\mathbf{u})-\mathbf{S}_{i,j}(\tilde{\mathbf{u}}),\end{aligned} \tag{6.1}$$

which can be interpreted as a classical finite volume formulation with modified fluxes and source:

$$\begin{aligned}\overline{\mathbf{F}}_{i+1/2,j}(\mathbf{u}) &= \mathbf{F}_{i+1/2,j}(\mathbf{u})-\mathbf{F}_{i+1/2,j}(\tilde{\mathbf{u}}),\\ \overline{\mathbf{G}}_{i,j+1/2}(\mathbf{u}) &= \mathbf{G}_{i,j+1/2}(\mathbf{u})-\mathbf{G}_{i,j+1/2}(\tilde{\mathbf{u}}),\\ \overline{\mathbf{S}}_{i,j}(\mathbf{u}) &= \mathbf{S}_{i,j}(\mathbf{u})-\mathbf{S}_{i,j}(\tilde{\mathbf{u}}).\end{aligned} \tag{6.2}$$

This approach is very easy to code and, further, the structures related to the steady reference solution can be computed in advance once and then used for every timestep without affecting the computational time. It must be underlined that, with this technique, all cell average computations, WENO reconstruction and source terms of the reference solution are performed following the same procedures and quadrature rules carried out for solving the balance law. Hence, all the terms always match when at the equilibrium.

6.2 Patankar method for production-destruction systems

Many problems (2.58) in nature can be written as a production destruction system (PDS) for the unknown $y \in \mathbb{R}^S$

$$f_\alpha(y) = \sum_{\beta=1}^{S} (p_{\alpha,\beta}(y) - d_{\alpha,\beta}(y)), \tag{6.3}$$

where $p_{\alpha,\beta}, d_{\alpha,\beta} \geq 0$ are the production and destruction terms, respectively. The production and destruction terms are conveniently written as matrices. Applications for PDS are for example the biological and/or chemical reactions such as algae bloom [61]. Also parts (or all) of the semi discretization of hyperbolic conservation/balance laws can be interpreted in such PDS system as described in [170, 171, 215] and also later in this work. The calculated solutions are often describing physical quantities that enjoy some properties, for instance concentrations of chemicals or water height in the context of SWE should be nonnegative. The following definition may be introduced for ODE systems:

Definition 28. *An ODE* (2.58) *is called* positive*, if positive initial data* $y_0 > 0$ *result in positive solutions* $y(t) > 0, \forall t$*. Here, inequalities for vectors are interpreted componentwise, i.e.,* $y(t) > 0$ *means* $\forall \alpha \colon y_\alpha(t) > 0$*. A PDS* (6.3) *is* conservative*, if* $p_{\alpha,\beta}(y) = d_{\beta,\alpha}(y), \forall\, \alpha,\, \beta,\, y$*.*

These properties should be preserved by the numerical scheme as well. Thus, we introduce the following discrete counterpart.

Definition 29. *A numerical method computing* $y^{n+1} \approx y(t_{n+1})$ *given* $y^n \approx y(t_n)$ *is called* conservative*, if* $\sum_\alpha y_\alpha^{n+1} = \sum_\alpha y_\alpha^n$*. It is called* unconditionally positive*, if* $y^n > 0$ *implies* $y^{n+1} > 0$*.*

From literature [61], it is well-known that the implicit Euler method is conservative and *unconditionally* positive preserving whereas the explicit Euler method is only conservative (it might be positive under time step restrictions). To avoid solving a fully nonlinear

system of equations, the so–called Patankar modifications have been applied to the explicit Euler method. To build an unconditionally positive numerical scheme, Patankar had the idea [243] of firstly weighting the destruction term in the original explicit Euler method with a coefficient as follows

$$y_\alpha^{n+1} = y_\alpha^n + \Delta t \left(\sum_{\beta=1}^{S} p_{\alpha,\beta}(y^n) - \sum_{\beta=1}^{S} d_{\alpha,\beta}(y^n) \frac{y_\alpha^{n+1}}{y_\alpha^n} \right), \quad \alpha = 1, \dots, S. \tag{6.4}$$

Indeed, the resulting scheme (6.4) is unconditionally positive and the implicit terms can be collected on the left hand side, but the conservation relation is violated. Burchard et al. had the idea [61] not only to weight the destruction term but also the production term:

$$y_\alpha^{n+1} = y_\alpha^n + \Delta t \left(\sum_{\beta=1}^{S} p_{\alpha,\beta}(y^n) \frac{y_\beta^{n+1}}{y_\beta^n} - \sum_{\beta=1}^{S} d_{\alpha,\beta}(y^n) \frac{y_\alpha^{n+1}}{y_\alpha^n} \right), \quad \alpha = 1, \dots, S. \tag{6.5}$$

They called their constructed scheme (6.5) **modified Patankar scheme** and proved that it is is unconditionally positive and conservative. The resulting scheme is linearly implicit, meaning that collecting all the implicit terms on the left hand side, we obtain a linear system at each time iteration. Based on this technique, extensions to second and third order modified Patankar Runge–Kutta (MPRK) methods have been made by several researchers in such context, cf. [170, 171, 185, 186]. Also the semi implicit RK methods proposed in [84] can be interpreted as Patankar methods as they weight only the destruction terms [286]. Finally, in [234] an arbitrarily high-order, conservative and positivity preserving scheme based on the DeC framework has been constructed. Herein we describe the main idea. For the details of the properties and proofs, we refer again to [234].

6.3 Modified Patankar Deferred Correction method

The modified Patankar Deferred Correction (mPDeC) is based on the DeC algorithm (2.73) and it consists in a modification of the $\mathcal{L}^2$ operator through the modified Patankar trick. This amounts to weight the production-destruction terms with respect to the intermediate approximations.

Using the fact that initial states $y_\alpha^{0,(k)}$ are identical for any correction k, the mPDeC correction steps can be rewritten [234] for $k = 1, \dots, K$, $m = 1, \dots, M$ and $\forall \alpha = 1, \dots, S$

as

$$y_\alpha^{m,(k)} - y_\alpha^0 - \sum_{r=0}^{M} \theta_r^m \Delta t \sum_{\beta=1}^{S} \left(p_{\alpha,\beta}(y^{r,(k-1)}) \frac{y^{m,(k)}_{\gamma(\beta,\alpha,\theta_r^m)}}{y^{m,(k-1)}_{\gamma(\beta,\alpha,\theta_r^m)}} - d_{\alpha,\beta}(y^{r,(k-1)}) \frac{y^{m,(k)}_{\gamma(\alpha,\beta,\theta_r^m)}}{y^{m,(k-1)}_{\gamma(\alpha,\beta,\theta_r^m)}} \right) = 0, \quad (6.6)$$

where θ_r^m are the DeC quadrature weights in time and

$$\gamma(\alpha,\beta,\theta) := \begin{cases} \alpha & \text{if } \theta \geq 0, \\ \beta & \text{if } \theta < 0. \end{cases}$$

Finally, the new numerical solution is $y^{n+1} = y^{M,(K)}$. As in the classical DeC framework, the choice of the distribution, the number of subtimesteps M and the number of iterations K determines the order of accuracy of the scheme. To reach order p, classically $M = p-1$ equispaced subintervals and $K = p$ corrections should be used. As proven in [234], the scheme is conservative, positivity preserving and can reach arbitrary high order.
A description of the assembly of the mass matrix of the system (6.6) is described in Algorithm 2 and one timestep of the mPDeC algorithm is sketched in Algorithm 1 where the evolution formula is given by Algorithm 3.

Remark 30 (Subtimestep distribution)**.** *In our numerical simulations, we apply Gauss-Lobatto nodes in every timestep. They have the advantage of requiring less subtimesteps to reach pth order of accuracy. In the following we will use $M = 3$ Gauss-Lobatto subtimesteps, which guarantee 6th order of accuracy for the operator $\mathcal{L}^2$ and $K = 5$ iterations aiming at a 5th order scheme to match the spatial discretization accuracy of WENO5.*

Remark 31 (Solution of the linear system)**.** *At each subtimestep m and iteration (k) we need to solve the linear system given by* (6.6)*. The mass matrix obtained has the following form:*

$$\mathbb{M}(y^{m,(k-1)})_{\alpha,\beta} = \begin{cases} 1 + \Delta t \sum_{r=0}^{M} \sum_{\beta=1}^{S} \frac{\theta_r^m}{y_\alpha^{m,(k-1)}} \left(d_{\alpha,\beta}(y^{r,(k-1)}) \mathbb{1}_{\{\theta_r^m > 0\}} - p_{\alpha,\beta}(y^{r,(k-1)}) \mathbb{1}_{\{\theta_r^m < 0\}} \right), & \text{for } \alpha = \beta, \\ -\Delta t \sum_{r=0}^{M} \frac{\theta_r^m}{y_\beta^{m,(k-1)}} \left(p_{\alpha,\beta}(y^{r,(k-1)}) \mathbb{1}_{\{\theta_r^m > 0\}} - d_{\alpha,\beta}(y^{r,(k-1)}) \mathbb{1}_{\{\theta_r^m < 0\}} \right), & \text{for } \alpha \neq \beta, \end{cases} \quad (6.7)$$

where $\mathbb{1}$ is the indicator function. The mass matrix assembly algorithm is described in Algorithm 2. The linear system will then read

$$\mathbb{M}(y^{m,(k-1)}) y^{m,(k)} = y(t^n). \quad (6.8)$$

Remark 32 (Division on almost wet areas)**.** *When the water height is low, we might encounter troubles in computing the divisions in* (6.7) *as the denominator might be very small. They hypothesis behind the production and destruction system that says that as* $h_\alpha \to 0$ *also* $d_\alpha \to 0$*, can be difficult to be obtain at a numerical level. Hence, to be sure that those divisions do not lead to extremely high values when they should go to 0, we slightly modify the way we implement the division as suggested in [216]. Given any numerator* n *and denominator* d *of* (6.7)*, we approximate the division by*

$$\frac{n}{d} \approx \begin{cases} 0 & d < \varepsilon, \\ \frac{2d\cdot n}{d^2+\max\{d^2,\varepsilon\}} & d \geq \varepsilon, \end{cases} \tag{6.9}$$

with ε *a small tolerance value. Along the computations, if not specified, we will use* $\varepsilon := 10^{-6}$*. This formulation allow to smoothly pass from* $\frac{n}{d}$ *to* 0 *as* $d \to 0$*. Moreover, when* $d^2 \geq \varepsilon$ *the division will be exact.*

In the following, we apply the mPDeC time marching algorithm (6.6) to the WENO finite volume semi-discretization to solve the shallow water equations. Below we describe the actual implementation procedure.

Algorithm 2 Mass

Require: Production-destruction functions $p_{\alpha,\beta}(\cdot)$, $d_{\alpha,\beta}(\cdot)$, Δt, previous correction variables $\underline{\boldsymbol{y}}^{(k-1)}$, current subtimestep m.

$\mathbb{M} \leftarrow \mathbb{I}$
for $\alpha = 1$ **to** S **do**
  for $\beta = 1$ **to** S **do**
    for $r = 0$ **to** M **do**
      if $\theta_r^m \geq 0$ **then**
        $\mathbb{M}_{\alpha,\beta} \leftarrow \mathbb{M}_{\alpha,\beta} - \Delta t \theta_r^m \frac{p_{\alpha,\beta}(\boldsymbol{y}^{r,(k-1)})}{y_\beta^{m,(k-1)}}$
        $\mathbb{M}_{\alpha,\alpha} \leftarrow \mathbb{M}_{\alpha,\alpha} + \Delta t \theta_r^m \frac{d_{\alpha,\beta}(\boldsymbol{y}^{r,(k-1)})}{y_\alpha^{m,(k-1)}}$
      else
        $\mathbb{M}_{\alpha,\beta} \leftarrow \mathbb{M}_{\alpha,\beta} + \Delta t \theta_r^m \frac{d_{\alpha,\beta}(\boldsymbol{y}^{r,(k-1)})}{y_\beta^{m,(k-1)}}$
        $\mathbb{M}_{\alpha,\alpha} \leftarrow \mathbb{M}_{\alpha,\alpha} - \Delta t \theta_r^m \frac{p_{\alpha,\beta}(\boldsymbol{y}^{r,(k-1)})}{y_\alpha^{m,(k-1)}}$
      end if
    end for
  end for
end for
return $\mathbb{M}$

Algorithm 3 mPDeC Update formula

Require: $\underline{\boldsymbol{y}}^{(k-1)}$, Δt, production-destruction functions $p_{\alpha,\beta}(\cdot)$, $d_{\alpha,\beta}(\cdot)$, m.

Compute the mass matrix $\mathbb{M}(\boldsymbol{y}^{m,(k-1)}) \leftarrow$Mass$(p_{\alpha,\beta}(\cdot), d_{\alpha,\beta}(\cdot), \Delta t, \underline{\boldsymbol{y}}^{(k-1)}, m)$ using Algorithm 2
Compute $\boldsymbol{y}^{m,(k)}$ solving the linear system $\mathbb{M}(\boldsymbol{y}^{m,(k-1)})\boldsymbol{y}^{m,(k)} = \boldsymbol{y}^n$ given by (6.6) with Jacobi Algorithm 4
return $\boldsymbol{y}^{m,(k)}$

Algorithm 4 Jacobi iterative method

Require: $\mathbb{D}$ diagonal of the matrix, $\mathbb{L}$ off-diagonal terms of the matrix, $\mathtt{r}$ right hand side of the system, $\mathtt{tol}$ tolerance.

$\mathtt{err} \leftarrow 2 \cdot \mathtt{tol}, \quad k \leftarrow 0, \quad \mathtt{x}^k \leftarrow \mathtt{r}$
while $\mathtt{err} > \mathtt{tol}$ **do**
  $k \leftarrow k + 1$
  $\mathtt{x}^{k+1} \leftarrow \mathbb{D}^{-1}(\mathtt{r} - \mathbb{L}\mathtt{x}^k)$
  $\mathtt{err} \leftarrow ||\mathtt{x}^k - \mathtt{x}^{k-1}||$
end while
return $\mathtt{x}^{k+1}$

6.4 Finite Volume schemes as production-destruction systems

In the following part, we will describe how the semi-discretization, using the WENO approach, can be written and interpreted as a PDS in order to apply the mPDeC scheme.

First of all we must underline the fact that in order to preserve the positivity of the water height h, the mPDeC scheme is going to be applied only to the first equation of system (1.39) as h is the only variable that must stay nonnegative. Thus, a simple DeC approach is going to be used to evolve the momentum equations. So from now on, we shall only talk about the modifications introduced for the first equation to turn it into a production-destruction system. Given the foundations of finite volume schemes, each control volume has fluxes entering and exiting its boundary and, for each boundary face, the flux going from element $\alpha = [i, j]$ to element $\beta = [l, r]$ is going to be equal in module and opposite in sign to the flux from element β to element α. This is the key feature for turning a finite volume schemes into a PDS. Note that the source is zero on h component. Therefore, we can define the production and destruction terms for a general $\alpha = [i, j]$ and the neighboring $\beta = [l, r]$ as

$$\begin{aligned}
p_{[i,j],[i-1,j]}(u) &= +\frac{1}{\Delta x}\overline{\mathbf{F}}_{i-1/2,j}(\mathbf{u})^+, \quad d_{[i,j],[i-1,j]}(u) = -\frac{1}{\Delta x}\overline{\mathbf{F}}_{i-1/2,j}(\mathbf{u})^-, \\
p_{[i,j],[i+1,j]}(u) &= -\frac{1}{\Delta x}\overline{\mathbf{F}}_{i+1/2,j}(\mathbf{u})^-, \quad d_{[i,j],[i+1,j]}(u) = +\frac{1}{\Delta x}\overline{\mathbf{F}}_{i+1/2,j}(\mathbf{u})^+, \\
p_{[i,j],[i,j-1]}(u) &= +\frac{1}{\Delta y}\overline{\mathbf{G}}_{i,j-1/2}(\mathbf{u})^+, \quad d_{[i,j],[i,j-1]}(u) = -\frac{1}{\Delta y}\overline{\mathbf{G}}_{i,j-1/2}(\mathbf{u})^-, \\
p_{[i,j],[i,j+1]}(u) &= -\frac{1}{\Delta y}\overline{\mathbf{G}}_{i,j+1/2}(\mathbf{u})^-, \quad d_{[i,j],[i,j+1]}(u) = +\frac{1}{\Delta y}\overline{\mathbf{G}}_{i,j+1/2}(\mathbf{u})^+,
\end{aligned} \tag{6.10}$$

where with the superscript $^+$ and $^-$ we denote the positive and the negative part respectively. All the other $p_{\alpha,\beta}$ and $d_{\alpha,\beta}$ not defined here are set to 0. Clearly, this define a conservative and positive PDS, as the properties in Definition 28 are verified. The visualization of the production and destruction terms in Figure 6.1 may help the reader. We clearly observe that the matrices $(p_{\alpha,\beta})$ and $(d_{\alpha,\beta})$ are $S \times S$ sparse matrices, with, at

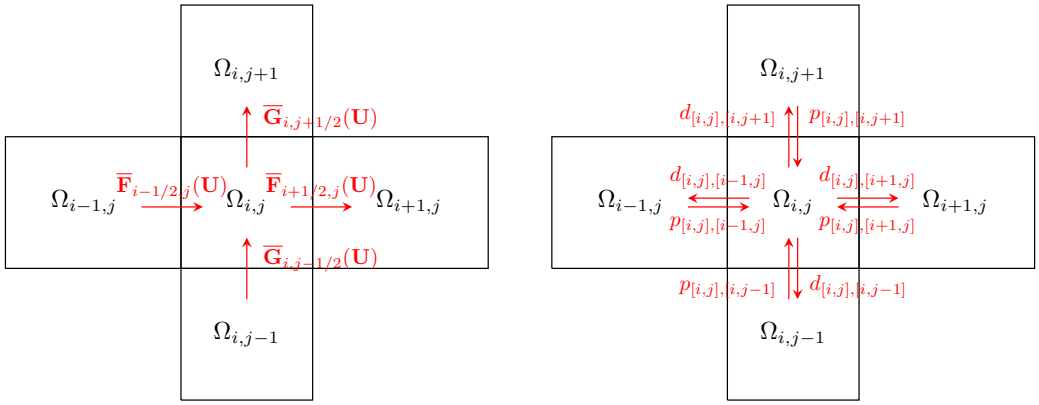


Figure 6.1: Cell $\Omega_{i,j}$, its four neighbors and its production and destruction terms.

most, 4 nonzero entries per row and $S = N_x \cdot N_y$. Once the production and destruction matrices have been assembled, the next step consist in running the mPDeC algorithm in Equation (6.6). Note that the matrix built in (6.7) is sparse as well with at most 5 nonzero entries for each row (4 nonzero entries of the production/destruction terms and the diagonal term). Hence, in the numerical computations we will use the classical Jacobi iterative method, see Algorithm 4, to obtain the solution of system (6.6) at each iteration. Indeed, it is provable [234, 61] that the Jacobi iteration algorithm converge on the matrix defined in (6.7).

In all calculations, we set the tolerance to machine precision and the algorithm converge towards the solution in few iterations. Experimentally, we have seen that usually 10-20 iterations suffice, in the worst cases 40 iterations are needed and the number of iterations do not depend on the mesh size. Overall, considering the assembly the mass matrix, the inversion of the system with Jacobi iterations, the whole mPDeC procedure increases the computational costs of $\sim 10\%$ with respect to the original DeC algorithm using the same CFL. The code has not been construct with the goal of optimizing all the procedures, so it might well be that this extra computational cost can be decreased with better implementations. The advantage is that no CFL restrictions are required in order to guarantee the positivity of the solution.

Remark 33 (Efficiency and properties of Jacobi iterative method)**.** *Though being a very simple algorithm, Jacobi iterative method is particularly suited for this application. Indeed, the mass matrix in* $\mathbb{R}^{S\times S}$ *is very sparse, i.e., only* $5S$ *non-zero elements, hence, at each iterations, only* $5S$ *multiplications are computed. This is in contrast with other methods that might be optimal for more general structures of matrices, but that do not exploit the sparsity of the matrix. As an example, Krylov subspace preconditioner provides an useful framework in many situations. Nevertheless, the solution of the minimization problem*

that pops up cannot be performed with computational costs linear in S, for example using a GMRES method. On the other side, a conjugate gradient method (CGM) would have a complexity of $10S$ for each iteration. The costs are quite similar. The convergence might be slower with Jacobi, but we will show in the test section that the amount of iterations is never too large. Moreover, we are sure that each iteration of the Jacobi method will return a positive approximation of the solution, while this cannot be guaranteed with the CGM. For these reasons, we believe that the Jacobi method is a reasonable choice.

6.5 Full Algorithm

After describing how the WENO (FV) procedure can be interpreted as a PDS, we give a more precise description of the full algorithm which is used to calculate the numerical solution. In our version of this approach, we have to adapt the steps taking into account the re-interpretation of the WENO approach as a production destruction system and the well balancing approach and this is described in Algorithm 5.

Finally, we can combine all the ingredients described above in a full algorithm as in Algorithm 6. There, we simply use the mPDeC to evolve in time and the production and destruction functions are given by the WENO description from above.

Algorithm 5 WENO FV with PDS structure

Require: $\mathbf{u}_{i,j}$, well balanced fluxes

1: Reconstruct on the quadrature points on cell interfaces the variable $\mathbf{u}$ in a high order fashion
2: Compute the numerical fluxes at quadrature points on cell interfaces $\hat{\mathbf{F}}(\mathbf{u}^L, \mathbf{u}^R)$
3: Subtract the correction for well balanced problems and obtain $\overline{\mathbf{F}}$ and $\overline{\mathbf{G}}$
4: Integrate over the cell interface to obtain the numerical fluxes $\mathbf{F}_{i+1/2,j}, \mathbf{F}_{i-1/2,j}$ and $\mathbf{G}_{i,j-1/2}, \mathbf{G}_{i,j+1/2}$
5: Compute $p_{\alpha,\beta}(\mathbf{u})$, $d_{\alpha,\beta}(\mathbf{u})$ as in (6.10)
6: **return** $p_{\alpha,\beta}(\mathbf{u})$, $d_{\alpha,\beta}(\mathbf{u})$

Algorithm 6 Full algorithm

Require: $\mathbf{u}_{i,j}^0$, T

1: $t = 0$
2: **while** $t < T$ **do**
3: Compute Δt by CFL restrictions
4: $\mathbf{u}^{n+1}$=DeC($\mathbf{u}^n, \Delta t$, WENOPDS) with DeC Algorithm 1 where update formula (6.6) is used for h and (2.73) is used for hu and hv and the WENO PDS function are given by Algorithm 5
5: $t = t + \Delta t$
6: **end while**

Remark 34 (Difference with respect to classical WENO)**.** *We want to highlight the differences between a classical WENO and the proposed algorithm are minimal. Indeed, once the spatial discretization is performed with a simple WENO step we need to apply two easy modifications. The first one consists in subtracting the flux related to the steady state variables. The second one consists in the definition of the production and destruction terms. Then the mPDeC can be applied as a simple time integration scheme. The code and these modifications are available at the reproducibility repository [88].*

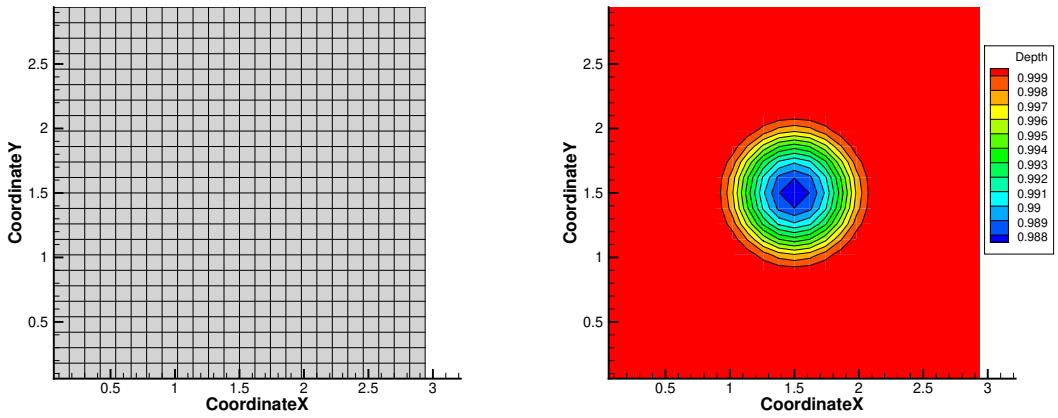
Remark 35 (Advantages of mPDeC)**.** *We shall remark that the presented method does not require any CFL constraint to obtain positive solutions for h, while the classical positivity limiter for WENO5 requires a CFL number of $1/12 \approx 0.083$. Clearly, a CFL number for a classical explicit method must be anyway used (between 1 and 1.5), but this allow to run the simulation with much less time steps than a classical explicit WENO scheme, with the extra computational cost of the Jacobi iterative method, which is negligible with respect the cost of decreasing the CFL number of factor of 12. Moreover, the procedure allows to have a provably positive method with arbitrarily high order of accuracy, while with positive limiters applied to WENO schemes only SSPRK methods guarantee the positivity of the solutions and they exist only up to order 4 [158].*

6.6 Validation

The goal of this section is to present the results obtained with the fifth order positivity-preserving mPDeC scheme, compared to that given by the classical fifth order DeC time integration method. The first test case consists in assessing the convergence properties of the spatial and temporal discretization on an unsteady vortex-type solution [262]. Afterwards, we focus on testing the well-balanced implementation for the lake at rest solution by showing its impact on perturbation analysis. Finally, three challenging simulations are performed to prove its capabilities to cope with wet-dry fronts. For all the simulations carried out herein periodic boundary conditions have been considered together with the local Lax-Friedrich (Rusanov) numerical flux.

6.6.1 Unsteady vortex

In order to verify the order of accuracy we consider a moving smooth vortex. The computational domain is the square $[0, 3] \times [0, 3]$. The initial condition is given by some perturbations δ applied on a homogeneous background field $(h_0, v_{x0}, v_{y0}) = (1, 2, 3)$. Hence,



(a) Computational mesh (b) Exact solution

Figure 6.2: Unsteady vortex: test case setting.

the perturbation for the depth variable h is

$$h(r) = h_0 - \delta h(r) = h_0 - \gamma \begin{cases} e^{-\frac{1}{\arctan^3(1-r^2)}}, & \text{if } r < 1, \\ 0, & \text{else,} \end{cases} \qquad (6.11)$$
$$\text{with } r = \sqrt{(x-1.5)^2 + (y-1.5)^2},$$

and the vortex amplitude is $\gamma = 0.1$. The velocity field is affected by the following perturbations

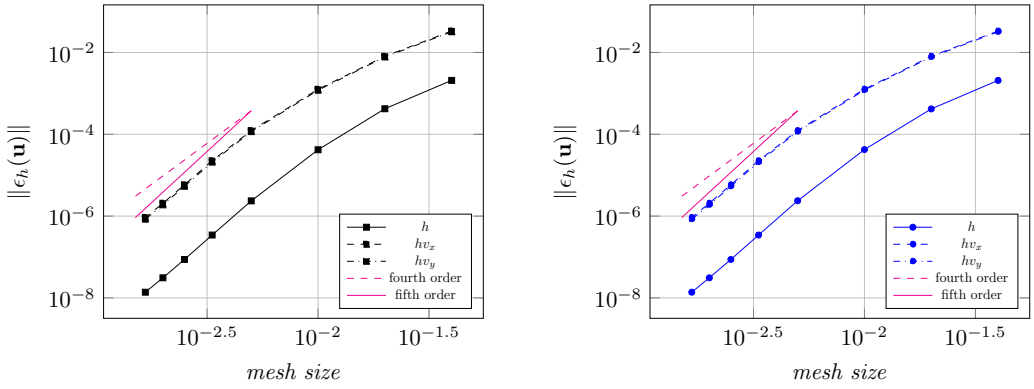
$$\begin{pmatrix} \delta v_x \\ \delta v_y \end{pmatrix} = \sqrt{2\, g\, \partial_r h} \begin{pmatrix} (y-1.5) \\ -(x-1.5) \end{pmatrix}, \qquad (6.12)$$

where $\partial_r h = \partial_r h(r)$ is a function of the radial distance from the center of the vortex

$$\partial_r h(r) = \frac{3\,\gamma\, e^{-\frac{1}{\arctan^3(1-r^2)}}}{\arctan^4(r^2-1)((r^2-1)^2+1)}. \qquad (6.13)$$

It is important to highlight the fact that this solution is $\mathcal{C}^\infty$, which is a fundamental property for testing arbitrarily high order schemes. Many vortex-type solutions can be found available online but most of them can only be used to test lower order schemes. The exact solution of this problem is given by

$$\begin{aligned} h(x,y,t) &= h(x - v_{x0}t, y - v_{y0}t, 0), \\ v_x(x,y,t) &= v_x(x - v_{x0}t, y - v_{y0}t, 0), \\ v_y(x,y,t) &= v_y(x - v_{x0}t, y - v_{y0}t, 0). \end{aligned} \qquad (6.14)$$

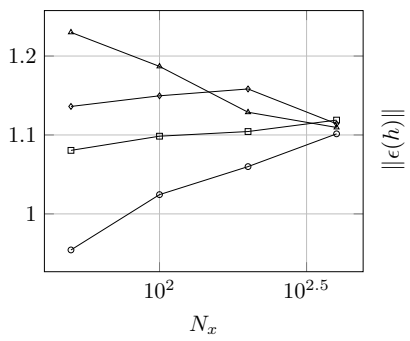


(a) WENO5−DeC5 (b) WENO5−mPDeC5

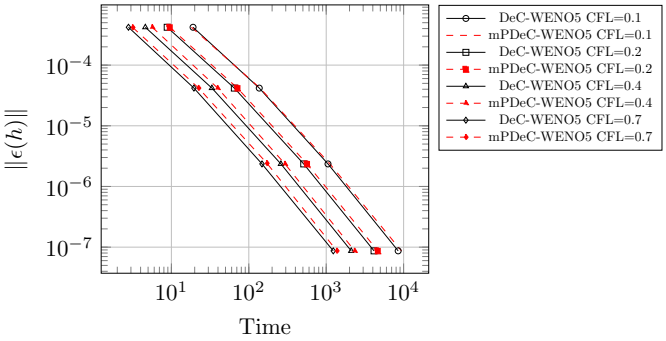
Figure 6.3: Unsteady vortex: convergence tests.

For the unsteady vortex, two convergence tests are run for WENO5 coupled with both the time integration schemes DeC5 and mPDeC to corroborate the fact that, for smooth flows, the results should be almost identical. We used CFL=0.7. The convergence tests are run on Cartesian meshes of size 25×25, 50×50, 100×100, 200×200, 300×300, 400×400, 500×500 and 600×600. The computational mesh of the coarsest grid and initial condition for this test case are shown in Figure 6.2. For these convergence tests we used a tolerance $\varepsilon = 10^{-30}$ both for the positivity limiter and the mPDeC divisions, since the errors that we obtain are of the order of 10^{-8}. The error $||\epsilon_h(\mathbf{u})||$ is the $\mathbb{L}^1$ norm of the difference between the exact solution and the approximated one. Figure 6.3 points out the predicted fifth order behavior for both time integration schemes.

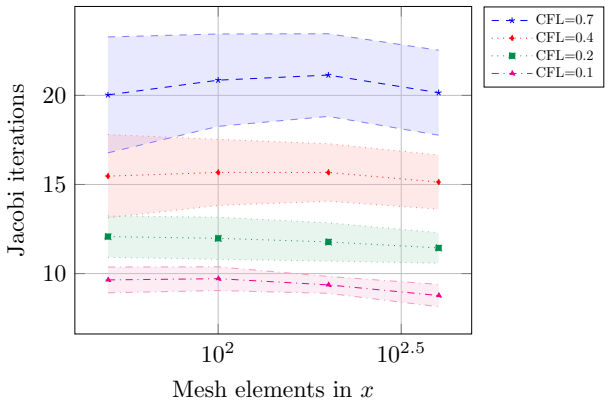
In Figure 6.4 we study the computational costs of the two methods (mPDeC and DeC) with respect to different CFL numbers and mesh refinements. In Figure 6.4(a) we plot the ratio of computational time of mPDeC over the computational time of DeC needed to finish the simulation for the same CFL and mesh. We see that the for fine mesh, when the computational time is more reliable, all ratios are close to 1.1. This means that the overhead that mPDeC requires, with respect to an explicit method, is of about 10%. In Figure 6.4(b) we plot error with respect to computational time and we see a very small difference between mPDeC and DeC, while there is a huge difference in computational costs when changing the CFL. The mPDeC is guaranteed to run at any CFL < 1, while the positivity for other SSPRK methods is guaranteed only for CFL $< 1/12$ with WENO5. Hence, there is huge advantage with mPDeC. Finally, in Figure 6.4(c) we can observe on average how many iterations are needed to solve the linear system and a confidence interval defined by the average $\pm \frac{1}{2}$ standard deviation. It is clear that Jacobi iterations are mainly driven by the CFL number, which explicitly appear as a coefficient of the mass matrix minus the identity. Indeed, all factors $\frac{\mathrm{d}}{\mathrm{d}t}\frac{d}{h}$ or $\frac{\mathrm{d}}{\mathrm{d}t}\frac{p}{h}$ are linearly dependent on the



(a) Ratio of computational time of mPDeC over DeC



(b) Computational time and error



(c) Average of Jacobi iterations and confidence interval

Figure 6.4: Unsteady vortex test: computational time and Jacobi iterations

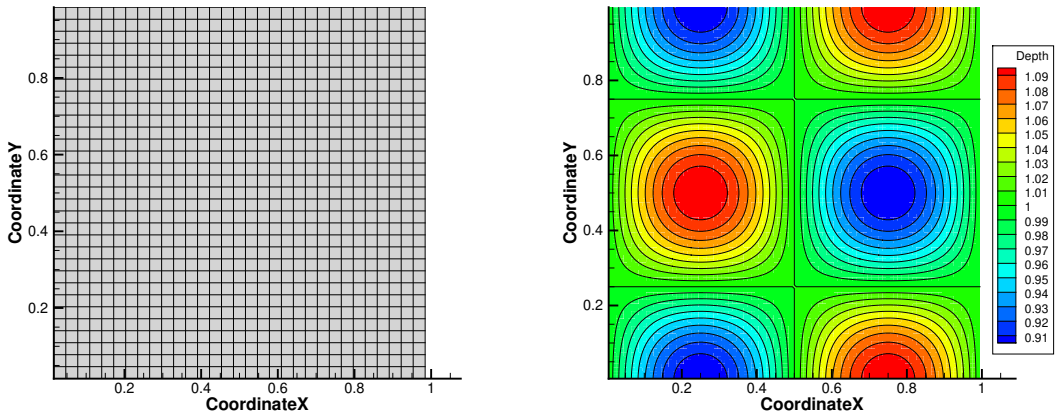
CFL as production and destruction terms are proportional to $\frac{1}{\Delta x}$.

6.6.2 Lake at rest

As already introduced in the theoretical part, this test case is needed to prove the presented scheme is well-balanced. The computational domain is the square $[0,1] \times [0,1]$ and the steady solution of this problem and bathymetry are briefly summarized below

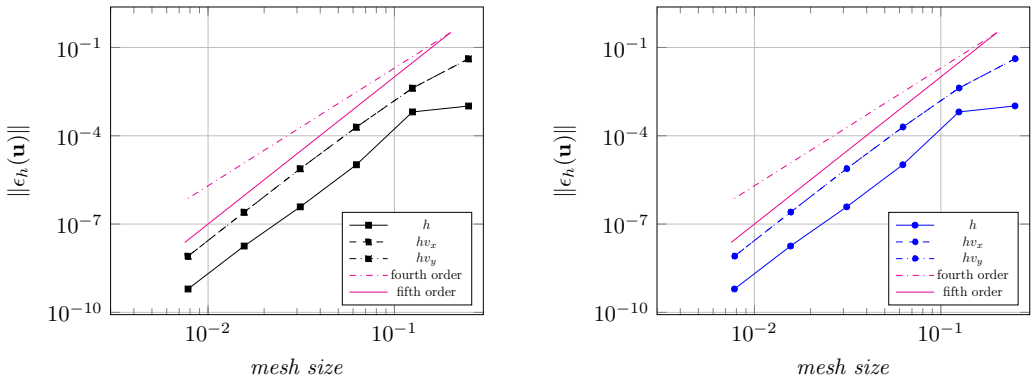
$$b(x,y) = 0.1 \sin(2\pi x) \cos(2\pi y), \qquad h(x,y) = 1 - b(x,y), \qquad v_x = v_y = 0. \quad (6.15)$$

This benchmark can also be used to test once again the order of accuracy of our discretization. Indeed, we expect the method to converge with a fifth order slope when not well-balanced and we expect machine precision errors for all the well-balanced tests. This simulation has been performed with four different settings: fifth order DeC and mPDeC, well-balanced and not well-balanced. For all cases, we employed a fifth order WENO discretization for the spatial derivatives and CFL=0.9. Also for this test, we chose $\varepsilon = 10^{-30}$



(a) Computational mesh (b) Exact solution of h

Figure 6.5: Lake at rest: test case setting.



(a) WENO5−DeC5 (b) WENO5−mPDeC5

Figure 6.6: Lake at rest: convergence tests without preserving the exact solution.

to check the accuracy of the scheme with errors of the order of 10^{-10}. As expected, the error computed for the well-balanced simulations is exactly zero therefore we did not plot them along with the other results. The interested reader can run the simulations and test the properties of our method by downloading the code available at the reproducibility repository [88]. The Cartesian mesh employed for this convergence test are 4×4, 8×8, 16×16, 32×32, 64×64 and 128×128. The exact solution is presented in Figure 6.5, along with the 32×32 mesh. As can be noticed from Figure 6.6, mPDeC5 allows a fifth order convergence rate as theoretically proved with results almost identical to those of DeC5.

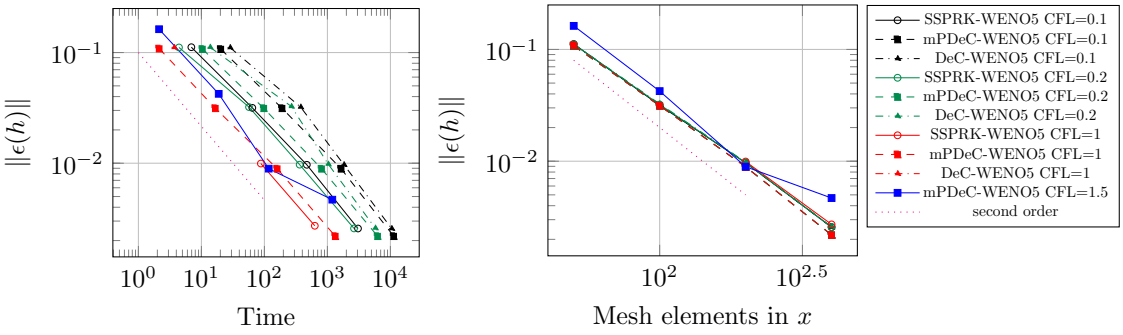


Figure 6.7: Wet and dry lake at rest test: error and computational time.

6.6.3 Wet-dry lake at rest

Now we test the capability of dealing with wet and dry regions of the scheme in a very simple context. We consider a bathymetry given by a bump

$$b(x,y) = \begin{cases} e^{1-\frac{1}{1-r^2}}, & \text{if } r^2 < 1, \\ 0, & \text{else }, \end{cases} \qquad \text{where } r^2 = x^2 + y^2, \tag{6.16}$$

on the domain $[-5,5] \times [-2,2]$. The lake at rest solution is

$$h(x,y) = \max\{0.7 - b(x,y), 0\}, \qquad v_x = v_y = 0. \tag{6.17}$$

The maximum of the bathymetry is 1, hence, there is a dry island at the center of the domain. For practical purposed, we set the initial conditions to be

$$h_0(x,y) = \max\{0.7 - b(x,y), \varepsilon\}, \qquad v_x = v_y = 0. \tag{6.18}$$

with $\varepsilon = 10^{-6}$. We consider final time $T = 1$.

First, we test the non-well-balanced schemes, to asses the capability of preserving the water height positivity and the accuracy of such methods. The positivity of the classical schemes is not preserved, even with the positivity limiter. In the dry region, the shallow water model does not hold and for the time integration schemes it is hard to verify hypotheses that guarantee the positivity of the solution. Hence, for the classical schemes, we force the positivity of the water height every time we need to compute the flux or to convert the variables from conservative to primitive and *vice versa*. On the other hand, the mPDeC scheme always preserve the positivity of the solution and none of these trick is required.

In figure 6.7 we observe that for similar computational times, the mPDeC-WENO5

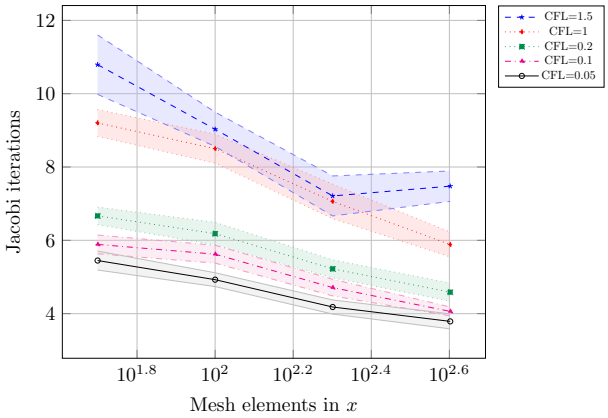


Figure 6.8: Wet and dry lake at rest test: average of Jacobi iterations and confidence interval for mPDeC-WENO5.

gives the same accuracy of all classical schemes. For all schemes the accuracy is 2, as the solution is not $\mathcal{C}^1$ everywhere. The difference is that mPDeC-WENO5 can be run up to CFL=1, without any problem, while for CFL=1 the SSPRK(6,4)-WENO5 scheme, even with the checks on the positivity, can have problems and might have exploding velocities and water heights. As the reconstruction does not guarantee the positivity for such high CFLs. Hence, it is not safe to run such simulations.

Another property we want to assess within this test is the number of Jacobi iterations required in the simulation. In figure 6.8 we plot the number of Jacobi iterations needed to converge at different CFLs and at different mesh refinement levels. The plot shows the average along the simulation with a confidence interval given by $\pm\frac{1}{2}$ standard deviation. It is clear that the CFL plays a very important role in the Jacobi algorithm. Indeed, the mass matrices are always composed by an identity matrix plus some terms that are of the type $\Delta t\frac{p}{h}$ or $\Delta t\frac{d}{h}$, where the production or destruction terms are proportional to $\frac{1}{\Delta x}$. Hence, the CFL coefficient pops out from this factor. It is clear that decreasing the CFL shortens the distance between the mass matrix and the identity matrix and, hence, speeds up the convergence of the algorithm. It is also clear from figure 6.8 that refining the mesh implies a lower number of Jacobi iterations. This can be maybe explained by the sparsity of the matrix which has a fraction of nonzero elements equal to $\frac{5N}{N^2} = \frac{5}{N}$ with $S = N_x \times N_y$, and, hence, it increases when refining the mesh. Even though this phenomenon is observable only for this wet and dry test, but not for unsteady vortex. So it might be that other reasons are responsible for this. This interesting effect is still under investigation.

Adding the well-balanced technique we obtain machine precision errors for all schemes.

N_x	Error h	Order h	Error v_x	Order v_x	Error v_y	Order v_y
600	8.346e-05	3.899	7.759e-04	3.057	6.911e-04	3.091
700	4.166e-05	4.508	4.526e-04	3.498	3.648e-04	4.144
800	2.134e-05	5.007	2.533e-04	4.346	1.886e-04	4.941
1000	6.185e-06	5.551	7.406e-05	5.511	5.225e-05	5.753
1200	2.219e-06	5.621	2.478e-05	6.006	1.774e-05	5.924
1400	1.120e-06	4.438	1.034e-05	5.669	7.561e-06	5.532
1600	5.896e-07	4.804	5.200e-06	5.148	3.799e-06	5.155

Table 6.1: Almost dry lake at rest: mPDeC-WENO5

N_x	Error h	Order h	Error v_x	Order v_x	Error v_y	Order v_y
600	8.378e-05	3.891	7.765e-04	3.057	6.907e-04	3.091
700	4.191e-05	4.493	4.532e-04	3.493	3.644e-04	4.147
800	2.160e-05	4.963	2.538e-04	4.342	1.884e-04	4.942
1000	6.426e-06	5.434	7.432e-05	5.504	5.228e-05	5.745
1200	2.569e-06	5.029	2.502e-05	5.972	1.797e-05	5.857
1400	1.376e-06	4.051	1.058e-05	5.586	7.805e-06	5.410
1600	9.062e-07	3.127	5.496e-06	4.901	4.096e-06	4.829

Table 6.2: Almost dry lake at rest: SSPRK-WENO5

6.6.4 Almost dry lake at rest

Now we modify the previous test, in order to have a smooth solution and to be able to obtain a fifth order accuracy in the schemes. We consider again the bathymetry (6.20) on the domain $[-5,5]\times[-2,2]$. The lake at rest solution is defined, this time, as

$$h(x,y)=\max\{0.999-b(x,y),0\},\qquad v_x=v_y=0. \tag{6.19}$$

Notice that the bathymetry has a peak with value 1, but, depending on the mesh refinement, the peak will be lower. In most of the simulations, this test will be completely wet and $\mathcal{C}^\infty$. If we discretize the mesh with more than 600×180 elements, the water level will be below the threshold $\varepsilon=10^{-6}$. As before, we initialize the water height at least equal to $\varepsilon=10^{-6}$ and we let the schemes evolve up to final time $T=1$.
We compare the mPDeC-WENO5 and the SSPRK-WENO5 schemes. For the SSPRK-WENO5 to be run, we need to introduce extra checks on the water height along the computations of the flux, so that it does not become negative, while for the mPDeC time integration we do not need this type of extra corrections. In both cases we expect to have a small perturbation of the solution while computing the L_1 error, indeed, the initial condition set with minimum level at 10^{-6} should introduce an error, in very few cells, of this order. In table 6.1 there is the error analysis for the mPDeC-WENO5 method, while in table 6.2 there is the one referred to the SSPRK-WENO5 method. Despite expecting the error of the initialization to become evident around error of 10^{-6}, for the mPDeC

simulation, this small perturbation confined to very few cells does not propagate much and lead to very accurate results also for errors $\approx 5 \cdot 10^{-7}$. Even the correction to avoid division by zero does not seem to affect the accuracy of the solution, probably because the water height never reaches values much lower than 10^{-6}. So, the order of accuracy stays very close to five, the expected one, even for almost dry solutions.
On the other side, in the SSPRK simulation the need of extra corrections in the flux every time the solution is below 10^{-6} adds further errors that are visible at level of L_1 error around 10^{-6}. Indeed, it seems that the error of the water height starts plateauing close to that value. And there it loses the expected fifth order of accuracy.

6.6.5 Perturbation of the lake at rest

In order to better highlight the improvements one gets with the well-balanced implementation, a perturbation analysis is run on a problem with both wet and dry areas. This test case is run on the rectangular domain $[-5,-2] \times [5,2]$. The bathymetry is given by

$$b(x,y) = \begin{cases} e^{1-\frac{1}{1-r^2}}, & \text{if } r^2 < 1, \\ 0, & \text{else }, \end{cases} \qquad \text{where } r^2 = x^2 + y^2. \tag{6.20}$$

The lake at rest solution that we are going to conserve with the well-balanced implementation is

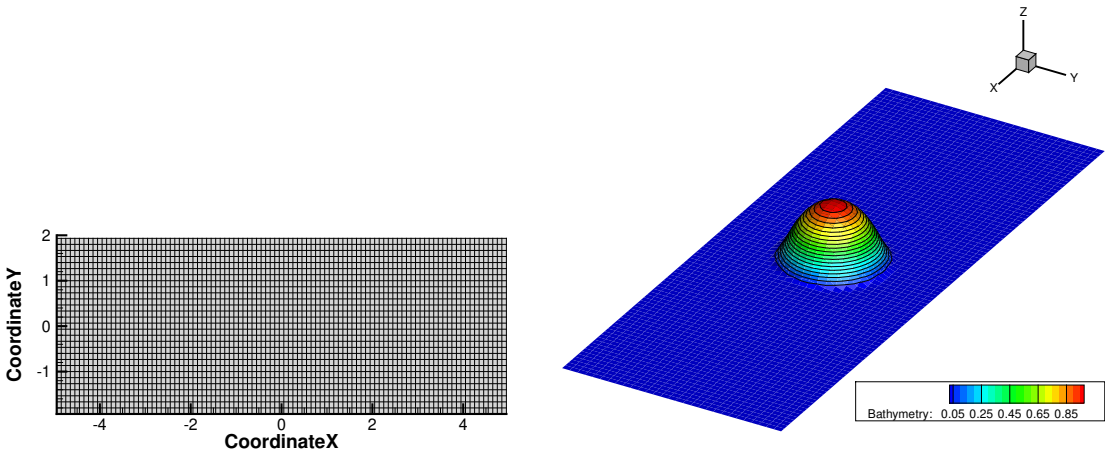
$$h(x,y) = \max\{0.7 - b(x,y), \varepsilon\}, \qquad v_x = v_y = 0. \tag{6.21}$$

with $\varepsilon = 10^{-6}$, whereas the perturbation shape that we want to study is

$$\tilde{h}(x,y) = h(x,y) + \begin{cases} 0.05\, e^{1-\frac{1}{(1-\rho^2)^2}}, & \text{if } \rho^2 < 1, \\ 0, & \text{else }, \end{cases} \qquad \text{where } \rho^2 = 9((x+2)^2 + (x-0.5)^2). \tag{6.22}$$

Two simulations have been performed: one with the well-balanced correction and one without. Both simulations use WENO5 as spatial discretization and mPDeC5 for integrating the ODEs coming from the semi-discrete system. From this test on the tolerances of the positivity limiter and of the mPDeC divisions is set to $\varepsilon = 10^{-6}$. The computational mesh and bathymetry plot are shown in Figure 6.9.

The results obtained for the two implementations are displayed in Figure 6.10 where only the iso-lines of the water height h are shown. Four snapshots are presented at different times of the simulation, $t = 0, 0.25, 0.5, 1$. The results on the right-hand side of Figure 6.10 are those computed without the well-balancing correction. As can be



(a) Computational mesh

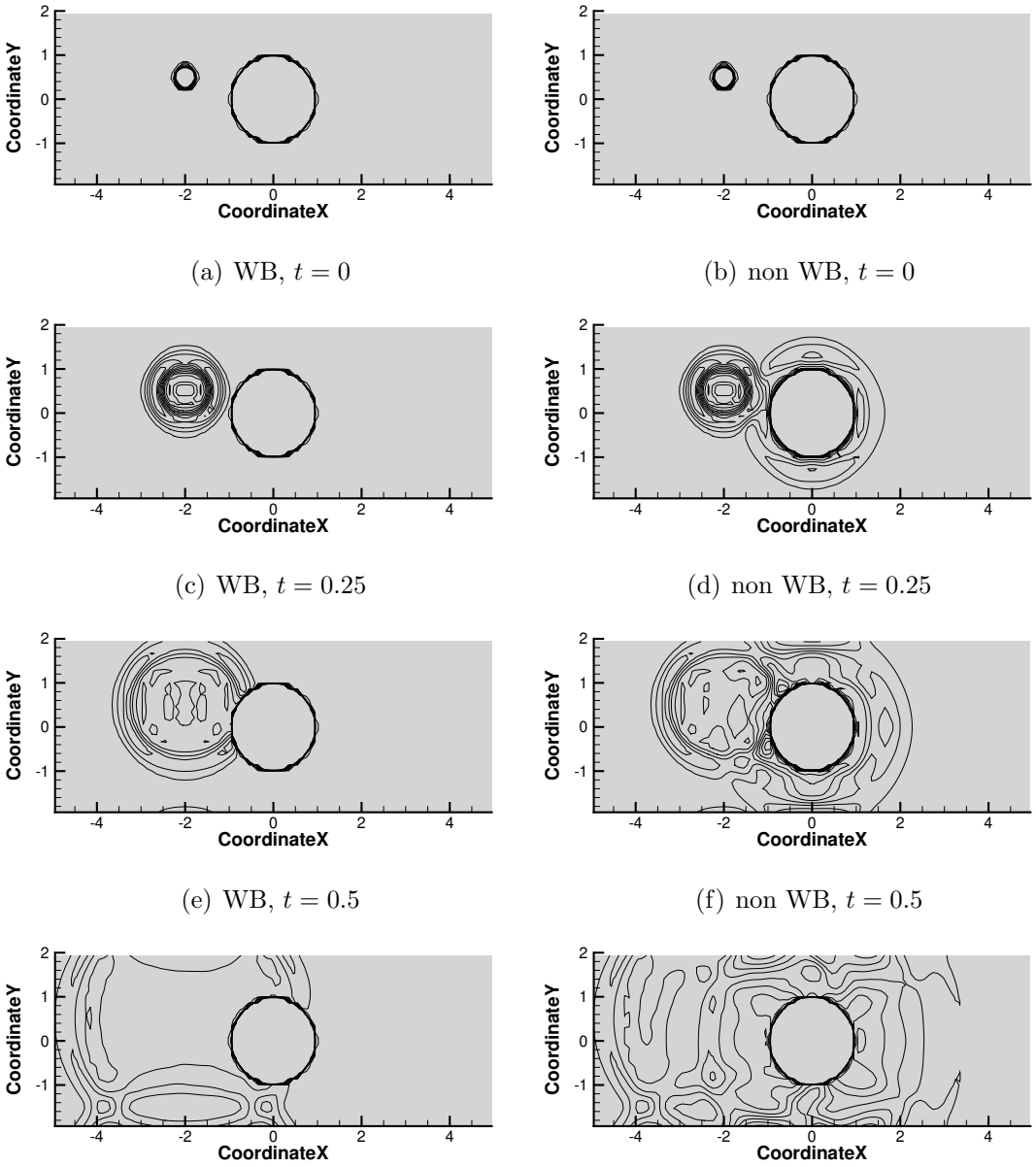
(b) Bathymetry

Figure 6.9: Perturbation analysis over a steady solution: test case setting

noticed, in this case, the numerical error propagates from the non-constant bathymetry area around the island placed in the middle of the domain. This error propagates and interacts with the perturbation, making the perturbation waves indistinguishable from the noise. This test case allows to better assess the well-balanced implementation already tested in the previous test case. Indeed, for this case, we have a dry area which involves a jump in the derivative of the water height causing a reduction of the order of accuracy given by the WENO method, whose limiters work with the high order derivatives of the solution. On the other side, the simulation runs with the well-balanced correction allows to exactly preserve the lake at rest solution over which the perturbation analysis is carried out. This leads to a much better capturing of the perturbation, whose evolution is not influenced by the spurious disturbances coming from the wet-dry area.

6.6.6 Circular dry dam break problem

We simulate the break of a circular dam separating two basins with water heights $h_1 = 2.5$ and $h_2 = \varepsilon = 10^{-6}$, meaning that the water in the first basin is falling over a dry area which is all around it. The radius of the discontinuity is $r = 7$. A sketch of the initial condition, along with the computational mesh, is given in Figure 6.11. The computational domain is the square $[0, 40] \times [0, 40]$ discretized with 100×100 cells and the simulation is run until a final time $t_{end} = 0.9$. This is a somewhat challenging test for the WENO5−mPDeC5 method that has to face both the capture of a sharp discontinuity and the progressing wetting of a dry area while always maintaining its appealing properties. The results are printed for different times, $t = 0, 0.3, 0.6, 0.9$, in Figure 6.12 showing the evolution of the



(a) WB, $t = 0$

(b) non WB, $t = 0$

(c) WB, $t = 0.25$

(d) non WB, $t = 0.25$

(e) WB, $t = 0.5$

(f) non WB, $t = 0.5$

(g) WB, $t = 1$

(h) non WB, $t = 1$

Figure 6.10: Perturbation analysis over a steady solution: water height h iso-lines for well-balanced (left-hand side) and non well-balanced (right-hand side) results.

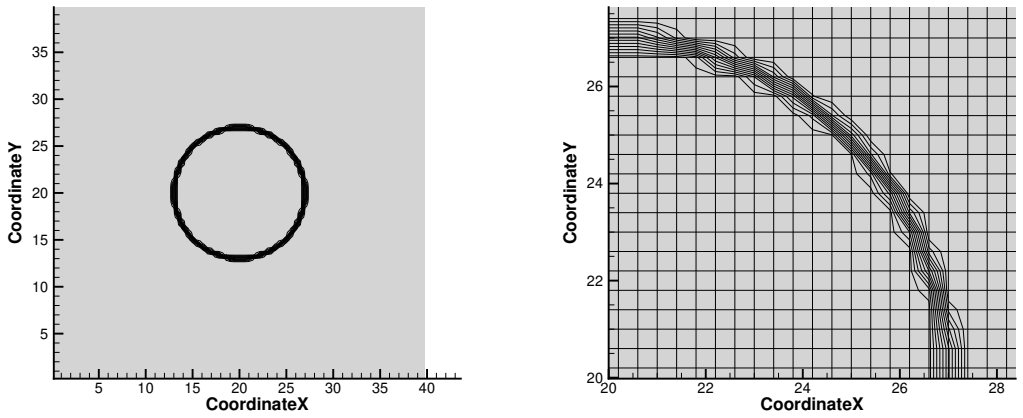
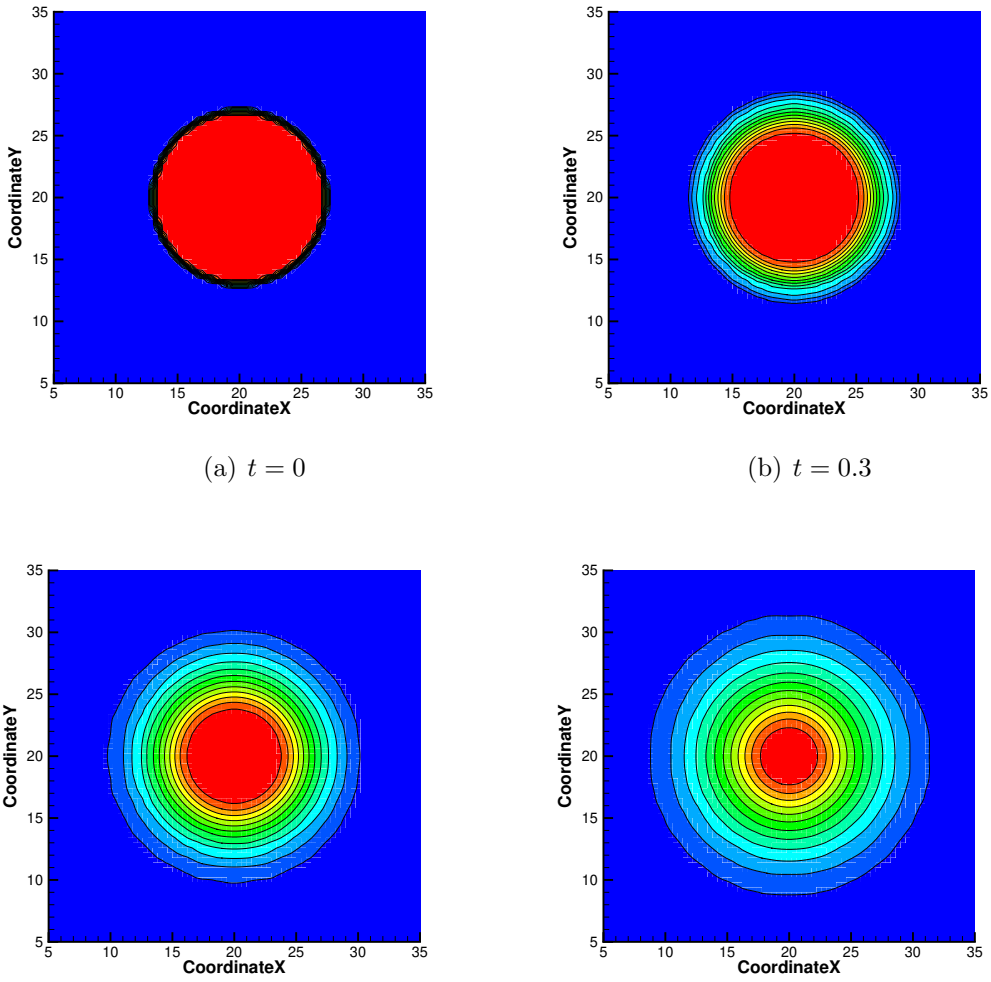


Figure 6.11: Circular dry dam break: computational domain and initial solution.

water height. The advantages of the mPDeC5 have been clearly proven by running this challenging test case with different CFL conditions. As a matter of fact, we managed to run this test case with a CFL up until 1.5 while ensuring positivity. In particular, the water height equation does not cause any CFL restriction due to the nature of the method used to solve it, which is unconditionally positive. However, since the momentum equations are solved by means of an explicit DeC scheme, the CFL must be bounded. It should be noticed that, in order to retain positivity in the spatial reconstruction, a positivity limiter has been implemented. For explicit SSPRK methods, this limiter has been proven to cause a huge restriction in the CFL condition, which now has to be less than $\frac{1}{12}\text{CFL}^{\text{SSPRK}}$. Nevertheless, since arbitrary high order DeC cannot be recast as a convex combination of explicit Euler method, there is no proof that the solution would stay positive also under that strict condition. On the other side, in the modified Patankar DeC framework, the limiter is only imposed on the water height equation which is unconditionally positive for any CFL by definition. This means that even when the limiter plays a role in the simulation, like in this case, the scheme stays positive with much higher CFL numbers. Furthermore, it should be underlined the fact that the system is linear implicit, which only requires a simple linear solver, e.g. Jacobi method.

6.6.7 Circular wet dam break problem

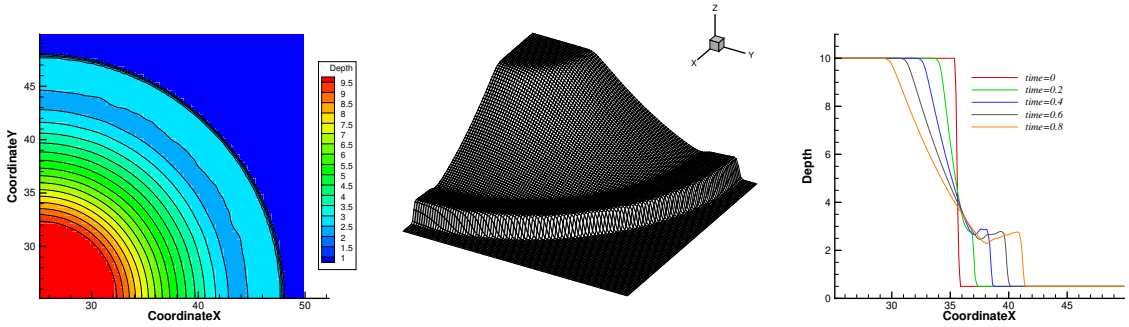
Next, we consider a wet dam break problem with a setup similar to the previous one. In this case the water height of the two basins are $h_1 = 10$ and $h_2 = 0.5$, like it was done in [261]. In this case, the computational domain is the square $[0, 50] \times [0, 50]$ discretized by a 200×200 mesh. The simulation is run until $t_{end} = 0.8$ with CFL=1 and the solutions



(a) $t = 0$ (b) $t = 0.3$

(c) $t = 0.6$ (d) $t = 0.9$

Figure 6.12: Circular dry dam break: water height h iso-contours.



(a) Iso-contours of h at time $t_{end} = 0.8$ (b) Plot of h at time $t_{end} = 0.8$ (c) h time evolution along the diagonal of the domain

Figure 6.13: Circular wet dam break

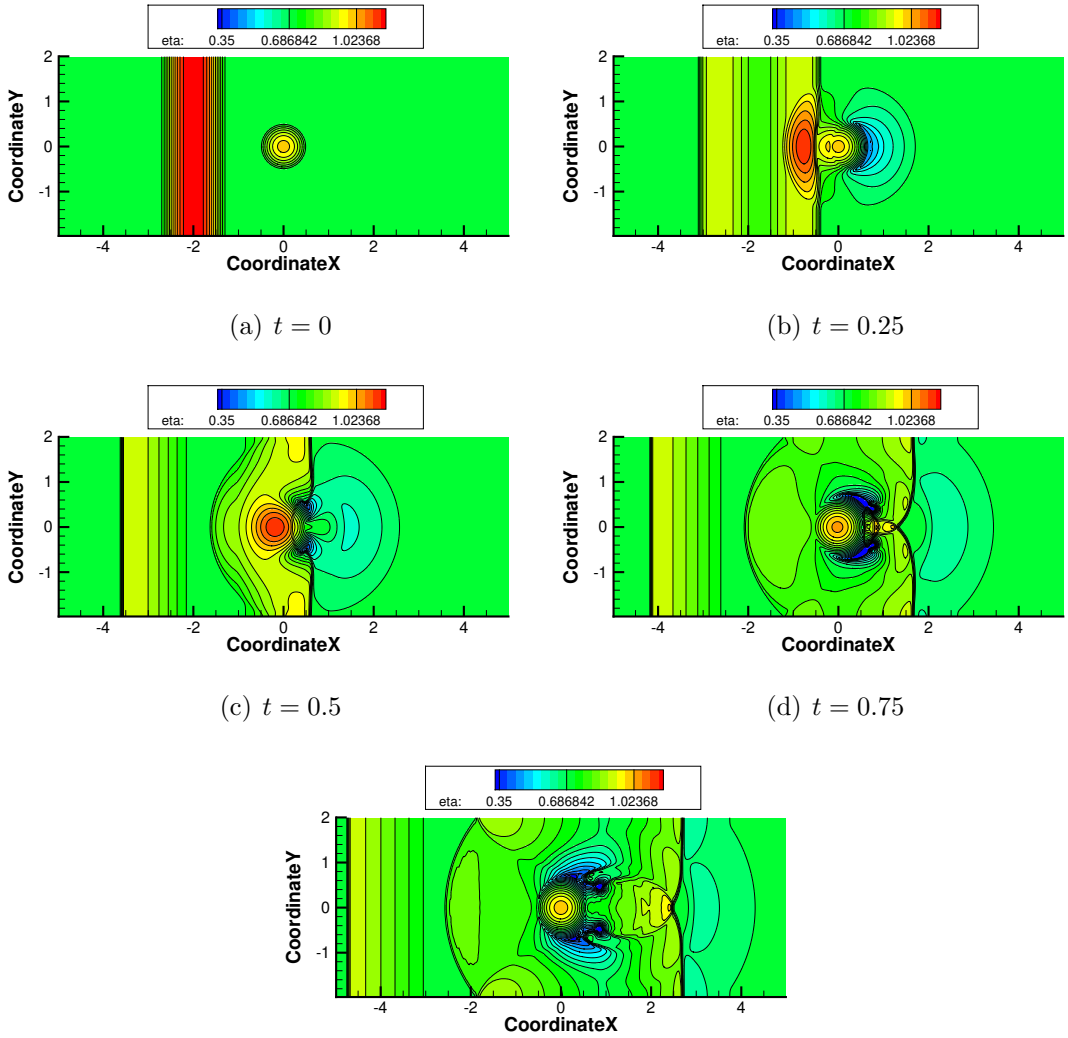
is displayed only for one quarter of the domain. Figure 6.13 displays the final snapshot of the solution at time $t_{end} = 0.8$ plotted both in 2D and 3D displaying the correct evolution of the water height and the time evolution of the water depth extracted along the diagonal of the portion of the domain studied. The method shows good properties such as discontinuities sharply captured, no oscillations and high order approximation of smooth features.

6.6.8 Wave over dry island

For the last test case, we simulate a wave crashing over a dry island showing the robustness of our method when facing more realistic simulations. The computational domain is the rectangle $[-5, -2] \times [5, 2]$ discretized by a 400×120 mesh. The simulation can be reproduced by taking Equation (6.20) for the bathymetry $b(x, y)$ and the following initial conditions

$$h(x,y) = 0.7 - b(x,y) + \begin{cases} 0.5\, e^{1-\frac{1}{(1-\rho^2)^2}}, & \text{if } \rho^2 < 1, \\ 0, & \text{else,} \end{cases} \qquad \text{where } \rho^2 = (x+2)^2, \quad (v_x, v_y) = (1, 0), \tag{6.23}$$

in case $h(x, y) \leq \varepsilon = 10^{-6}$, we set $h(x, y) = \varepsilon$ and $u = 0$. The simulation has been run with the WENO5−mPDeC5 scheme until a final time $t = 1$ with CFL=0.9 and the results are displayed in Figure 6.14 for different times $t = 0, 0.25, 0.5, 0.75, 1$. In this case, the variable $\eta = h + b$ has been chosen to be plotted since it better represents the underlying physics. It should be noticed that the top of the island, which is dry at the beginning, gets wet and dry several times during the simulation while never giving rise to problems due to negative water height. This is instead something that we cannot ensure for the DeC5 case. The simulation starts with a background state moving with speed $v_x = 1$



(a) $t = 0$

(b) $t = 0.25$

(c) $t = 0.5$

(d) $t = 0.75$

(e) $t = 1$

Figure 6.14: Wave over dry island: $\eta = h + b$ iso-contours at different times.

which helps the wave traveling towards the island and immediately starts wetting the island from the left and drying it on the right. The wave breaks into two smaller waves respectively approaching and moving away from the island following the eigenvalues of the flux Jacobian. At time $t = 0.25$ a run up is happening where the traveling wave is trying to submerge the top of the island while, on the other side, a section of the island is drying. In consecutive times, the wave overtakes the island causing different interacting shock fronts and two symmetrical minimum points highlighted in dark blue at time $t = 1$. Several structures could been observed in this simulations and the repeated wetting/drying procedures never results in troubles for the mPDeC method.

6.7 Chapter summary

We have presented a new well-balanced, positivity preserving high order numerical method for solving the shallow water equations. By re-writing the WENO semi-discretization in terms of productions-destructions terms, we were able to use the modified Patankar Deferred Correction methods of arbitrarily high-order to ensure the unconditionally positivity of water height. The restriction on the CFL number comes only from the explicit DeC-solver for the momentum equations. The used CFL numbers are of the order of 1 and are much larger than the ones used in explicit SSPRK WENO methods in combination with positivity preserving limiters [300], where the CFL must be lowered to 1/6 or 1/12. One can relax further the CFL constraint by using implicit DeC or RK methods, though introducing more difficulties. With classical explicit approaches, one must use SSPRK to guarantee positivity and, as known from literature [158], explicit (implicit) SSPRK methods exist only up to order four (sixth) for general cases.

By applying mPDeC, we avoid those issues and the price to pay is that of solving a (very sparse) linear system for the water height. However, as mentioned before, this increase in computational costs is around 10% in our numerical simulations, but the procedure can still be optimized.

Chapter 7

Well-Balanced Schemes based on Flux Globalization for Moving Equilibria Preservation

In the context of preserving stationary states, e.g. lake at rest and moving equilibria, a new formulation of the shallow water system, called Flux Globalization has been introduced by Cheng *et al.* [80]. This approach consists in including the integral of the source term in the *global flux* and reconstructing the new *global flux* rather than the conservative variables. The resulting scheme is able to preserve a large family of smooth and discontinuous steady state moving equilibria. Given that high order methods allow to obtain more accurate results within the same dimension of the semi-discretized system, herein we focus on an arbitrary high order WENO [275, 277] Finite Volume (FV) generalization of the global flux approach [91]. More information about the WENO reconstruction can be found in Section 2.2.2. As mentioned in the introduction, developing high order schemes that are able to preserve exactly such properties is much more challenging with respect to classical low order schemes for several technical reasons that will be better deepen below. The most delicate aspect of the algorithm is the appropriate definition of the *source flux* (integral of the source term) and the quadrature strategy used to match it with the WENO reconstruction of the hyperbolic flux. When this construction is correctly done, one can show that the resulting WENO FV scheme admits exact discrete steady states characterized by constant global fluxes.

7.1 Shallow water system with Flux Globalization

The idea based on global fluxes consists in incorporating the source term in the flux term and rewriting the one-dimensional version of the shallow water equations (1.39) in the equivalent conservation form:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathcal{G}(\mathbf{u}, x)}{\partial x} = 0 \quad \text{such that} \quad \mathcal{G}(\mathbf{u}, x) = \begin{bmatrix} q \\ K \end{bmatrix} = \begin{bmatrix} q \\ \frac{q^2}{h} + g\frac{h^2}{2} + \mathcal{R} \end{bmatrix}, \tag{7.1}$$

so that K becomes a global equilibrium variable with

$$\mathcal{R}(x,t) := -\int^x S(\mathbf{u}, \xi)\, \mathrm{d}\xi = g \int^x \left[h(\xi,t)\frac{\partial b(\xi)}{\partial \xi} + \frac{n^2}{h^{7/3}(\xi,t)} |q(\xi,t)| q(\xi,t) \right] \mathrm{d}\xi. \tag{7.2}$$

System (7.1) is therefore a hyperbolic system with a global flux, which can be solved with classical techniques for conservation laws without special treatments of the source term. The advantage of the global flux is to have a natural formulation of a quantity $\mathcal{R}$ that can be balanced for steady state solutions.

7.2 Global Flux Finite Volume method

The hyperbolic system considered herein is solved by means of the Method Of Lines (MOL). Hence, in this setting, space and time can be treated independently. This section has the goal of presenting a modified arbitrary high order finite volume framework for balance laws with global fluxes. A second order global flux approach is available in [80], but it is limited to linear reconstructions. In this section we aim at building an arbitrary

high order global flux method based on WENO reconstructions.
The computational domain $\mathbf{\Omega}$ is discretized into N_x equispaced control volumes $\Omega_i = [x_{i-1/2}, x_{i+1/2}]$ of size Δx centered at $x_i = i\Delta x$ with $i = i_l, \ldots, i_r$.
Considering the system of hyperbolic balance laws described by (1.3) and (1.39), for the control volume Ω_i we can define the cell average at time t:

$$\bar{\mathbf{u}}_i(t) := \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{u}(x,t)\, \mathrm{d}x. \tag{7.3}$$

The semi-discrete finite volume scheme for the system (7.1) reads

$$\frac{\mathrm{d}\bar{\mathbf{u}}_i}{\mathrm{d}t} + \frac{1}{\Delta x}(\mathbf{H}_{i+1/2} - \mathbf{H}_{i-1/2}) = 0, \tag{7.4}$$

where $\mathbf{H}_{i+1/2}$ is a numerical flux consistent with the global flux $\mathcal{G}$. The global flux differs from the original flux and this makes tricky the development of an upwind scheme based on the solution of Riemann problems by an approximate solver. Thus, we are going to employ an upwind scheme that can be easily applied to problems with global fluxes. In order to have a global flux formulation, all the update terms must depend only on the global flux $\mathcal{G}$ itself. In this way, when we reach a numerical steady state, we are sure that $\mathcal{G}$ is constant all over the domain. We start from $\bar{\mathcal{G}}_i$ cell average values and we reconstruct the global flux at interfaces with a high order WENO procedure. Then, we choose $\mathbf{H}_{i+1/2}$ to be an upwind numerical flux defined only on the global flux:

$$\mathbf{H}_{i+1/2} = L^{-1}\Lambda^+ L\, \mathcal{G}^L_{i+1/2} + L^{-1}\Lambda^- L\, \mathcal{G}^R_{i+1/2}. \tag{7.5}$$

Here, $\mathcal{G}^{L,R}_{i+1/2}$ are the discontinuous reconstructed point values of the global flux $\mathcal{G}(\mathbf{u})$ respectively at the left and right side of the cell interface $x_{i+1/2}$. L is the matrix of the left eigenvectors computed from the flux Jacobian of the hyperbolic problem (1.3) in the Roe averaged state, i.e.,

$$J(\mathbf{u}^*) = \begin{pmatrix} 0 & 1 \\ -u_*^2 + gh_* & 2u_* \end{pmatrix}, \qquad \text{with } \begin{cases} h_* = \frac{h^L + h^R}{2}, \\ u_* = \frac{\sqrt{h^L}u^L + \sqrt{h^R}u^R}{\sqrt{h^L}+\sqrt{h^R}}. \end{cases} \tag{7.6}$$

$\Lambda^\pm$ correspond to the upwinding weights

$$\Lambda_i^+ = \begin{cases} 1, & \text{if } \lambda_i > 0, \\ 0, & \text{if } \lambda_i < 0, \end{cases} \qquad \Lambda_i^- = \begin{cases} 1, & \text{if } \lambda_i < 0, \\ 0, & \text{if } \lambda_i > 0. \end{cases} \tag{7.7}$$

It is important to notice that the Jacobian of the flux cannot be directly computed as not all the quantities are available at the interface (we must reconstruct only the global flux, not the conserved quantities). Hence, one has to recover the value of h from the global flux itself. To do so in a positive manner, we use the technique proposed in [80] and reported in A.5.

In order to obtain the reconstructed values at the left and right side of interfaces, we will use a high order WENO reconstruction technique (presented in 2.2.2), where starting from cell averages, one obtains a nonlinear reconstruction inside a cell. In particular, we will consider piece–wise continuous reconstructions for all the quantities of interests, i.e., h, q, b, $\mathcal{R}$, $\mathcal{G}$ and K. On the other side, S, being the derivative of $\mathcal{R}$, must be defined in a distributional sense. This means that a particular form of S must be chosen when integrating it to obtain $\mathcal{R}$ at the interfaces. The details of this procedure will be specified in Section 7.2.2.

7.2.1 Global flux definitions

Now, to obtain the reconstruction of the global flux at the interfaces, we need, first of all, the values of the cell averages, defined as

$$\bar{\mathcal{G}}_i(\mathbf{u}, x) = \bar{\mathcal{F}}_i(\mathbf{u}) + \begin{bmatrix} 0 \\ \bar{\mathcal{R}}_i \end{bmatrix}. \tag{7.8}$$

We can compute the cell averaged flux evaluating the flux in some high order quadrature points of the cell, through the WENO reconstruction of the conserved variables in the quadrature points, i.e.,

$$\bar{\mathcal{F}}_i(\mathbf{u}) = \sum_q w_q \mathcal{F}(\tilde{\mathbf{u}}(x_{i,q})). \tag{7.9}$$

For the cell average of the global integral source $\mathcal{R}$, we have to perform a reconstruction based on the source terms evaluated, as well, in the quadrature points. Moreover, it will be performed iteratively, from the left boundary of the domain to the right. So that, the cell averages of the integral source can be defined as

$$\bar{\mathcal{R}}_i \approx \sum_q w_q \mathcal{R}_{i,q}. \tag{7.10}$$

To obtain $\mathcal{R}$ in the quadrature points, we have to keep in mind its definition as integral of S (7.2). So, we can define them using the right interface values of $\mathcal{R}$, i.e.,

$$\mathcal{R}_{i,q} = \mathcal{R}^R_{i-1/2} + \int_{x^R_{i-1/2}}^{x_{i,q}} \tilde{S}(x)\,\mathrm{d}x = \mathcal{R}^R_{i-1/2} + \sum_\theta \underbrace{\int_{x^R_{i-1/2}}^{x_{i,q}} \ell_\theta(x)\,\mathrm{d}x}_{r^q_\theta}\; S(x_{i,\theta}), \quad i > i_l, \tag{7.11}$$

where $\tilde{S}$ is a high order reconstruction of S done in quadrature points $x_{i,q}$ for each cell and ℓ_θ are the Lagrangian polynomials in the quadrature points:

$$\tilde{S}_i(x) = \sum_\theta \ell_\theta(x) S(x_{i,\theta}). \tag{7.12}$$

This definition necessitates the values of $\mathcal{R}$ at the right interface. In order to define a global $\mathcal{R}$, we need to define it on both sides of the interfaces. In particular, we can define $\mathcal{R}$ up to a constant, so we impose that $\mathcal{R}$ at the left extrema is equal to 0, i.e.,

$$\mathcal{R}^R_{i_\ell-1/2} = 0. \tag{7.13}$$

Then, we proceed with the integration from left to right, obtaining the values of $\mathcal{R}$ at the left interfaces from the right interface ones:

$$\mathcal{R}^L_{i+1/2} = \mathcal{R}^R_{i-1/2} + \int_{x^R_{i-1/2}}^{x^L_{i+1/2}} S(\mathbf{u}(x), x)\,\mathrm{d}x = \mathcal{R}^R_{i-1/2} + \Delta x \bar{S}_i, \quad i > i_l, \tag{7.14}$$

with $\bar{S}_i$ being the cell average of S computed as

$$\bar{S}_i := \frac{1}{\Delta x} \int_{x^R_{i-1/2}}^{x^L_{i+1/2}} S(\mathbf{u}, \xi)\,\mathrm{d}\xi \approx \sum_q w_q S(x_{i,q}). \tag{7.15}$$

Finally, we need to give a recursive definition of the right interface values as

$$\mathcal{R}^R_{i+1/2} = \mathcal{R}^L_{i+1/2} + [\![\mathcal{R}_{i+1/2}]\!], \tag{7.16}$$

where $[\![\mathcal{R}_{i+1/2}]\!]$ denotes the jump of $\mathcal{R}$ on $x_i + 1/2$. At this level, we still need to define how we compute the source in the quadrature points $S(x_{i,\theta})$ and the jump of the integral source $[\![\mathcal{R}_{i+1/2}]\!]$.

Since S is defined only in a distribution sense, the definition of the jump needs a careful choice. Essentially, S on the interfaces is a Dirac delta function, as we need to compute the derivative of the bathymetry which is discontinuous. The definition of such term, in

order to have a high order scheme, is not uniquely determined. Hence, it will be chosen so that the scheme is consistent and balanced with the flux in the lake at rest equilibrium. In the next section, we describe how to obtain the discretization of the source and jump of the integral of the source.

7.2.2 Global Flux formulation for lake at rest preservation

By construction, the scheme presented so far is able to preserve a constant global flux for the family of moving equilibria solutions, i.e., $q \equiv q_0$ and $K \equiv K_0$. However, in the lake at rest simulation, the preservation of the constant water free surface does not come for free, in particular we cannot preserve *a priori* $\eta = h + b \equiv \eta_0$. Therefore, a special treatment of the source term is required to assure so. The friction term is neglected here, because it vanishes when the speed is zero as in the lake at rest equilibrium.
We start by defining a high order reconstruction of h, η and b in cell Ω_i using the WENO reconstruction with the same weights for h, η and b. In particular, the weights will be chosen according to η in order to preserve the lake at rest balance. This means that we obtain a reconstructions for the three variables in the quadrature points $x_{i,q}$ that we denote with $\tilde{h}_{i,q}$, $\tilde{\eta}_{i,q}$ and $\tilde{b}_{i,q}$, and they are such that

$$\tilde{h}_{i,q} = \tilde{\eta}_{i,q} - \tilde{b}_{i,q}. \tag{7.17}$$

We then recast the source term, given that $h(x) = \eta(x) - b(x)$, as

$$S(\mathbf{u}, x) = gh(x)\partial_x b(x) = g\eta(x)\partial_x b(x) - g\partial_x \left(\frac{b^2(x)}{2} \right). \tag{7.18}$$

Equation (7.18) allows us to give proper shape to (7.14), which now reads

$$\mathcal{R}^L_{i+1/2} = \mathcal{R}^R_{i-1/2} - \int_{x^R_{i-1/2}}^{x^L_{i+1/2}} S(\mathbf{u}(x), x)\, \mathrm{d}x \tag{7.19}$$

$$= \mathcal{R}^R_{i-1/2} + g \int_{x^R_{i-1/2}}^{x^L_{i+1/2}} \eta(x)\partial_x b(x)\, \mathrm{d}x - g \left(\frac{(b^L_{i+1/2})^2}{2} - \frac{(b^R_{i-1/2})^2}{2} \right) \tag{7.20}$$

$$(\text{if } \eta \equiv \eta_0) = \mathcal{R}^R_{i-1/2} + g\eta_0 \left(b^L_{i+1/2} - b^R_{i-1/2} \right) - g \left(\frac{(b^L_{i+1/2})^2}{2} - \frac{(b^R_{i-1/2})^2}{2} \right). \tag{7.21}$$

As we see, the values of the bathymetry at the interfaces of the cells are necessary to compute (7.19). It is crucial to maintain the well balanced structure in the quadrature points $x_{i,q}$, hence, we reconstruct $b^L_{i+1/2}$ and $b^R_{i-1/2}$ using a Lagrangian polynomial interpolating

the values at the quadrature points, i.e.,

$$\tilde{b}_i(x) := \sum_q \ell_q(x)\tilde{b}_{i,q}, \qquad \text{and} \qquad b^L_{i+1/2} = \tilde{b}_i(x_{i+1/2}), \qquad b^R_{i-1/2} = \tilde{b}_i(x_{i-1/2}). \tag{7.22}$$

Equation (7.21), corresponding to the case $\eta \equiv \eta_0$, is the value that $\mathcal{R}^L_{i+1/2}$ assumes at the interface when the lake at rest is preserved. However, to get there, a proper discretization of the integral $\int_{x^R_{i-1/2}}^{x^L_{i+1/2}} \eta(x)\partial_x b(x)$ should be performed. To do so, let us introduce $Q(x) := g\eta(x)\partial_x b(x)$ and reconstruct it as the source term in (7.12). We can finally integrate it using ad hoc quadrature formulae.

$$\int_{x^R_{i-1/2}}^{x^L_{i+1/2}} \tilde{Q}(x)\mathrm{d}x = \int_{x^R_{i-1/2}}^{x^L_{i+1/2}} \sum_\theta \ell_\theta(x)\mathrm{d}x\; Q_{i,\theta}, \tag{7.23}$$

where

$$Q_{i,\theta} = \tilde{\eta}_{i,\theta} \sum_q \ell'_q(x_{i,\theta})\tilde{b}_{i,q}. \tag{7.24}$$

In particular, we applied the same approach to treat the derivative of the bathymetry. The fully integrated relation for $R^L_{i+1/2}$ now becomes

$$\mathcal{R}^L_{i+1/2} = \mathcal{R}^R_{i-1/2} + \int_{x^R_{i-1/2}}^{x^L_{i+1/2}} \sum_\theta \ell_\theta(x)\mathrm{d}x\; Q_{i,\theta} - g\left(\frac{(b^L_{i+1/2})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right) \tag{7.25}$$

$$= \mathcal{R}^R_{i-1/2} + g\sum_\theta \int_{x^R_{i-1/2}}^{x^L_{i+1/2}} \ell_\theta(x)\;\mathrm{d}x\;\tilde{\eta}_{i,\theta} \sum_s \ell'_s(x_\theta)\tilde{b}_{i,s} - g\left(\frac{(b^L_{i+1/2})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right). \tag{7.26}$$

Notice that the value of the bathymetry at the interfaces and the derivative of the bathymetry in the quadrature points is given by the same reconstruction (7.22). So, when $\eta \equiv \eta_0$, we have that

$$\begin{aligned}\mathcal{R}^L_{i+1/2} =& \mathcal{R}^R_{i-1/2} + g\eta_0 \int_{x^R_{i-1/2}}^{x^L_{i+1/2}} \sum_s \tilde{b}_{i,s}\ell'_s(x)\;\mathrm{d}x\; - g\left(\frac{(b^L_{i+1/2})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right)\\ =& \mathcal{R}^R_{i-1/2} + g\eta_0(b^L_{i+1/2} - b^R_{i-1/2}) - g\left(\frac{(b^L_{i+1/2})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right).\end{aligned} \tag{7.27}$$

Following the same reasoning $\mathcal{R}_{i,q}$ introduced in (7.11) can be recast as

$$\mathcal{R}_{i,q} = \mathcal{R}^R_{i-1/2} - \int_{x^R_{i-1/2}}^{x_{i,q}} S(\mathbf{u}(x), x)\,\mathrm{d}x \tag{7.28}$$

$$= \mathcal{R}^R_{i-1/2} + g\int_{x^R_{i-1/2}}^{x_{i,q}} \eta(x)\partial_x b(x)\,\mathrm{d}x - g\left(\frac{(b_{i,q})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right) \tag{7.29}$$

$$(\text{if } \eta \equiv \eta_0) = \mathcal{R}^R_{i-1/2} + g\eta_0\left(b_{i,q} - b^R_{i-1/2}\right) - g\left(\frac{(b_{i,q})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right). \tag{7.30}$$

Once again, this last identity corresponds to the solution at the lake at rest condition but the integral $\int_{x^R_{i-1/2}}^{x_{i,q}} \eta(x)\partial_x b(x)\,\mathrm{d}x$ has to be discretized to be consistent also for other solutions. Following what we did in (7.19), (7.28) becomes

$$\mathcal{R}_{i,q} = \mathcal{R}^R_{i-1/2} - \int_{x^R_{i-1/2}}^{x_{i,q}} \tilde{S}(x)\,\mathrm{d}x \tag{7.31}$$

$$= \mathcal{R}^R_{i-1/2} + \sum_\theta \int_{x^R_{i-1/2}}^{x_{i,q}} \ell_\theta(x)\,\mathrm{d}x\,\tilde{Q}_{i,\theta} - g\left(\frac{(b_{i,q})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right) \tag{7.32}$$

$$= \mathcal{R}^R_{i-1/2} + g\sum_\theta \int_{x^R_{i-1/2}}^{x_{i,q}} \ell_\theta(x)\,\mathrm{d}x\,\tilde{\eta}_{i,\theta}\sum_s \ell'_s(x_{i,\theta})\tilde{b}_{i,s} - g\left(\frac{(b_{i,q})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right). \tag{7.33}$$

So that, when $\eta \equiv \eta_0$, we have

$$\mathcal{R}_{i,q} = \mathcal{R}^R_{i-1/2} + g\eta_0\int_{x^R_{i-1/2}}^{x_{i,q}} \sum_s \ell'_s(x_{i,\theta})\tilde{b}_{i,s}\,\mathrm{d}x\,\tilde{\eta}_{i,\theta} - g\left(\frac{(b_{i,q})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right) \tag{7.34}$$

$$= \mathcal{R}^R_{i-1/2} + g\eta_0\left(\tilde{b}_{i,q} - b^R_{i-1/2}\right) - g\left(\frac{(b_{i,q})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right). \tag{7.35}$$

We still have not defined the jump of the integral source term $[\![\mathcal{R}_{i+1/2}]\!]$. Equipped with (7.19) and (7.28), considering $\eta \equiv \eta_0$ and $q \equiv 0$, we can build it such that the

cell averages $\bar{K}_i$ are constant in all the cells. We have that

$$K_{i,q} = \mathcal{F}_{i,q} + \mathcal{R}_{i,q} = \mathcal{R}^R_{i-1/2} + g\frac{(\eta_0 - b_{i,q})^2}{2} + g\eta_0\left(b_{i,q} - b^R_{i-1/2}\right) - g\left(\frac{(b_{i,q})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right) \tag{7.36}$$

$$= \mathcal{R}^R_{i-1/2} + g\frac{\eta_0^2}{2} - \cancel{g\eta_0 b_{i,q}} + \cancel{g\frac{b_{i,q}^2}{2}} + g\eta_0\left(\cancel{b_{i,q}} - b^R_{i-1/2}\right) - g\left(\cancel{\frac{(b_{i,q})^2}{2}} - \frac{(b^R_{i-1/2})^2}{2}\right) \tag{7.37}$$

$$= \mathcal{R}^R_{i-1/2} + g\frac{\eta_0^2}{2} - g\eta_0 b^R_{i-1/2} + g\frac{(b^R_{i-1/2})^2}{2} \tag{7.38}$$

and, clearly, $\bar{K}_i = K_{i,q}$ for all q. Hence, to prove that the scheme is indeed well-balanced for the lake at rest solution, we set $\bar{K}_{i+1} - \bar{K}_i = 0$:

$$\bar{K}_{i+1} - \bar{K}_i = \mathcal{R}^R_{i+1/2} - \mathcal{R}^R_{i-1/2} - g\eta_0 b^R_{i+1/2} + g\frac{(b^R_{i+1/2})^2}{2} + g\eta_0 b^R_{i-1/2} - g\frac{(b^R_{i-1/2})^2}{2} \overset{!}{=} 0. \tag{7.39}$$

Now, we compute the difference $\mathcal{R}^R_{i+1/2} - \mathcal{R}^R_{i-1/2}$ using (7.27) and we get

$$\mathcal{R}^R_{i+1/2} - \mathcal{R}^R_{i-1/2} = \mathcal{R}^L_{i+1/2} - \mathcal{R}^R_{i-1/2} + [\![\mathcal{R}_{i+1/2}]\!] \tag{7.40}$$

$$= g\eta_0\left(b_i + 1/2^L - b_i - 1/2^R\right) - g\left(\frac{(b_i + 1/2^L)^2}{2} - \frac{(b_i - 1/2^R)^2}{2}\right) + [\![\mathcal{R}_{i+1/2}]\!]. \tag{7.41}$$

Inserting (7.40) in (7.39), we get

$$\bar{K}_{i+1} - \bar{K}_i = g\eta_0\left(b_i + 1/2^L - \cancel{b_i - 1/2^R}\right) - g\left(\frac{(b_i + 1/2^L)^2}{2} - \cancel{\frac{(b_i - 1/2^R)^2}{2}}\right) + [\![\mathcal{R}_{i+1/2}]\!] \tag{7.42}$$

$$-g\eta_0 b^R_{i+1/2} + g\frac{(b^R_{i+1/2})^2}{2} + \cancel{g\eta_0 b^R_{i-1/2}} - \cancel{g\frac{(b^R_{i-1/2})^2}{2}} \overset{!}{=} 0, \tag{7.43}$$

which is indeed equal to zero if

$$[\![\mathcal{R}_{i+1/2}]\!] \overset{!}{=} g\eta_0\left(b_i + 1/2^R - b_i + 1/2^L\right) - g\left(\frac{(b_i + 1/2^R)^2}{2} - \frac{(b_i + 1/2^L)^2}{2}\right).$$

In order to have a consistent approximation of the jump of the integral source $[\![\mathcal{R}_{i+1/2}]\!]$,

we define it as

$$[\![\mathcal{R}_{i+1/2}]\!] := g\frac{\eta_i + 1/2^R + \eta_i + 1/2^L}{2}\left(b_i + 1/2^R - b_i + 1/2^L\right) - g\left(\frac{(b_i + 1/2^R)^2}{2} - \frac{(b_i + 1/2^L)^2}{2}\right). \tag{7.44}$$

Remark 36 (Friction)**.** *When adding the friction term to the global flux, we do not need to be so careful. Indeed, the friction term is defined as rational of polynomials of conservative variables that can be reconstructed piecewise continuously on the mesh. Hence, they do not contribute to the jump of* $\mathcal{R}$ *at the interfaces. So, it is enough to define*

$$\mathcal{R}_{i,q} := \mathcal{R}^R_{i-1/2} + g\sum_\theta \int_{x^R_{i-1/2}}^{x_{i,q}} \ell_\theta(x)\,\mathrm{d}x \left(\tilde{\eta}_{i,\theta}\sum_s \ell'_s(x_{i,\theta})\tilde{b}_{i,s} + g\frac{\tilde{q}_{i,\theta}|\tilde{q}_{i,\theta}|n^2}{\tilde{h}^{7/3}_{i,\theta}}\right) - g\left(\frac{(b_{i,\theta})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right) \tag{7.45}$$

and

$$\mathcal{R}^L_{i+1/2} := \mathcal{R}^R_{i-1/2} + g\sum_\theta \int_{x^R_{i-1/2}}^{x_{i+1/2}} \ell_\theta(x)\,\mathrm{d}x \left(\tilde{\eta}_{i,\theta}\sum_s \ell'_s(x_{i,\theta})\tilde{b}_{i,s} + g\frac{\tilde{q}_{i,\theta}|\tilde{q}_{i,\theta}|n^2}{\tilde{h}^{7/3}_{i,\theta}}\right) - g\left(\frac{(b^L_{i+1/2})^2}{2} - \frac{(b^R_{i-1/2})^2}{2}\right). \tag{7.46}$$

In Algorithm 7 we summarize the steps of the reconstruction of the source integral.

Algorithm 7 Source integral reconstruction

$\mathcal{R}_{i_l-1/2} := 0$
for $i = i_l, \dots, i_r$ **do**
  Reconstruct the variables h, η and b in each quadrature point θ of the cell, obtaining $\tilde{h}_{i,\theta}$, $\tilde{\eta}_{i,\theta}$ and $\tilde{b}_{i,\theta}$ using the same WENO weights (computed with respect to η)
  Reconstruct q in the quadrature points obtaining $\tilde{q}_{i,\theta}$
  Define $\mathcal{R}_{i,q}$ as in (7.45)
  Define $\mathcal{R}^L_{i+1/2}$ as in (7.46)
  Define $[\![\mathcal{R}_{i+1/2}]\!]$ as in (7.44)
  Define $\mathcal{R}^R_{i+1/2} := \mathcal{R}^L_{i+1/2} + [\![\mathcal{R}_{i+1/2}]\!]$
end for

7.3 Validation

The arbitrary high order well-balanced WENO finite volume scheme based on Flux Globalization has been tested and validated on several test cases to assess convergence properties and performances. In Section 7.3.1 we study the lake-at-rest steady state to verify the well-balancedness of the scheme by comparing the global flux methods with and without the formulation introduced in Section 7.2.2. In Section 7.3.2 we add a perturbation to the previous tests to compare the quality of the computed solutions by using the new

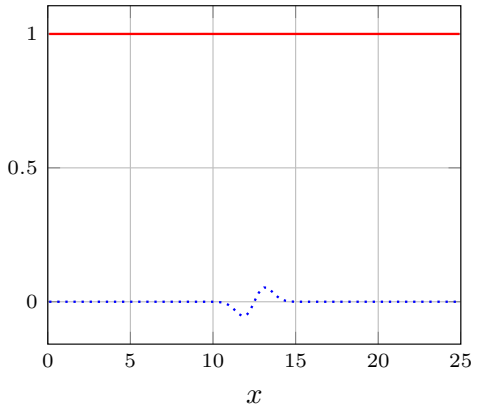


Figure 7.1: Lake at rest solution: η (red) and b (blue).

well-balanced global flux schemes and the classical WENO method. In Section 7.3.3 we focus on more general steady state with nonzero global fluxes and we test the method on subcritical, supercritical and transcritical flows. Finally, in Section 7.3.6 we add also the friction to the previous tests.

For smooth tests we will perform a convergence analysis to assess the order of accuracy of our methods. In particular, we will validate the algorithm with WENO3 and WENO5 reconstructions.

7.3.1 Lake at rest

Firstly, we consider the lake at rest solution characterized by the initial data

$$h(x,0) = 1 - b(x), \qquad q(x,0) \equiv 0, \tag{7.47}$$

over the computational domain $[0, 25]$ with subcritical inlet/outlet at the two boundaries. The bathymetry employed contains a sinusoidal bump damped at the boundaries (see Figure 7.1 for better visualization of solution and bathymetry) and it reads

$$b(x) = 0.05 \sin(x - 12.5) \exp\left(1 - (x - 12.5)^2\right). \tag{7.48}$$

Let us remark that the classical bathymetry provided in [113] is only piecewise polynomial, but globally only $\mathcal{C}^0$. Hence, it is not suited to test the accuracy of very high order methods. On the other side, the bathymetry (7.48) is $\mathcal{C}^\infty$ and it has values smaller than machine precision at the boundaries. The gravitational constant is considered to be $g = 1$ and the simulation is run until the final time $T = 1$ with $N_e = \{25, 50, 100, 150, 200, 400, 800\}$ uniform cells using both the non-well-balanced (non-WB) and well-balanced (WB) version of the algorithm to assess the convergence and well-balancing properties. The convergence

Table 7.1: Lake at rest: errors and estimated order of accuracy (EOA) with WB and non-WB schemes, using WENO3 and WENO5 reconstructions.

	Non-WB				WB			
	h		q		h		q	
N_e	L_2 error	EOA	L_2 error	EOA	L_2 error	EOA	L_2 error	EOA
	GF-WENO3				GF-WENO3			
25	1.0384E-4	–	4.7943E-5	–	9.8858E-14	–	1.2228E-15	–
50	1.5496E-5	2.67	9.2488E-6	2.31	9.8667E-14	–	1.4249E-15	–
100	1.2117E-6	3.62	3.6777E-7	4.59	9.8276E-14	–	1.6041E-15	–
150	2.6776E-7	3.69	1.5898E-7	2.05	1.9644E-13	–	3.3908E-15	–
200	9.6323E-8	3.53	7.6469E-8	2.53	1.9619E-13	–	3.6713E-15	–
400	8.2671E-9	3.53	6.0441E-9	3.65	2.9360E-13	–	6.1689E-15	–
800	6.8811E-10	3.58	4.7122E-10	3.67	5.8655E-13	–	1.3035E-14	–
	GF-WENO5				GF-WENO5			
25	5.1800E-5	–	6.1657E-5	–	9.8947E-14	–	1.3247E-15	–
50	4.4066E-6	3.45	1.5244E-6	5.18	9.8661E-14	–	1.4060E-15	–
100	6.7998E-7	2.66	3.5908E-7	2.06	9.8289E-14	–	1.5992E-15	–
150	1.5437E-7	3.63	8.8535E-8	3.42	1.9639E-13	–	3.4157E-15	–
200	4.1973E-8	4.50	2.3725E-8	4.55	1.9611E-13	–	3.7034E-15	–
400	1.3952E-9	4.89	7.5991E-10	4.95	2.9357E-13	–	6.2007E-15	–
800	4.3120E-11	5.01	2.2633E-11	5.06	5.8648E-13	–	1.3039E-14	–

tests performed with WENO3 and WENO5 are listed in Table 7.1. It can be noticed that the error decay for the non-WB simulations matches the order of the reconstruction for both WENO3 and WENO5, while, for the WB cases, the scheme is able to preserve the exact solution up to machine precision.

7.3.2 Small perturbation of the lake-at-rest solution

For this test case, we analyze the perturbation of the lake at rest solution characterized by

$$h(x,0) = 1 - b(x) + \begin{cases} \alpha\psi(x), & \text{if } 9 < x < 10 \\ 0, & \text{otherwise} \end{cases}, \qquad q(x,0) \equiv 0 \tag{7.49}$$

with $\psi(x)$ a perturbation function defined by

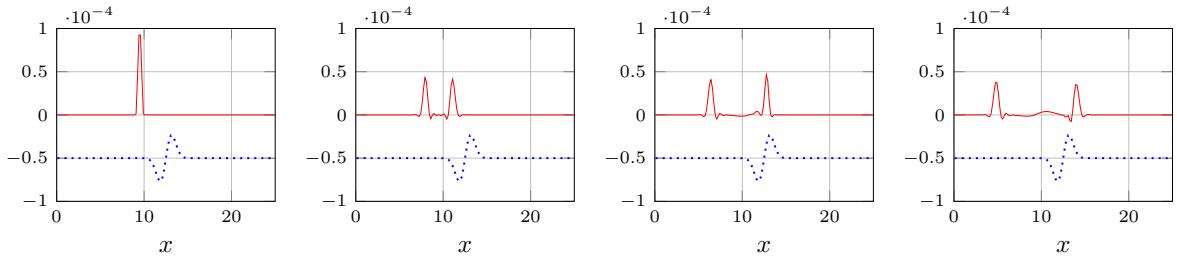
$$\psi(x) := \exp\left(1 - \frac{1}{(1 - r(x))^2}\right), \qquad \text{with } r(x) := 4(x - 9.5)^2 \tag{7.50}$$

and $\alpha = 10^{-4}$ over the computational domain $[0, 25]$ with subcritical inlet/outlet at the two boundaries. The bathymetry is a rescaling of (7.48) and it is defined as

$$b(x) = 0.5 \sin(x - 12.5) \exp\left(1 - (x - 12.5)^2\right). \tag{7.51}$$

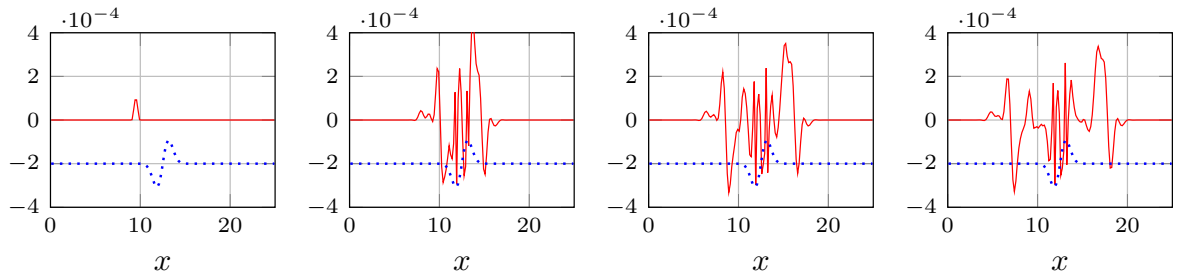
A slightly different bathymetry, with respect to Section 7.3.1, has been chosen in order to introduce more noise in the non-WB simulation and appreciate more the method capabilities. The simulation was run using a mesh with 150 cells. Figure 7.2 shows the

evolution over time of the perturbation added over the lake at rest solution computed with the GF-WENO5 WB scheme. To better present the results, the plots show the relative variable $h - h_{eq}$ where h_{eq} represents the lake at rest solution without perturbation provided in Equation (7.47). We then compare the results in Figure 7.2 with those computed using the classical WENO5 approach, pointed out in Figure 7.3. It should be noticed that the classical approach fails at correctly reproducing the perturbation. Indeed, it generates a discretization error of several orders of magnitude higher than the perturbation itself, eventually spoiling the final result. Contrary to that, the GF-WENO5 WB scheme correctly reproduce the perturbation which splits into two waves traveling at opposite directions and interacting with the bathymetry. The obtained result is fairly accurate, also considering the coarse mesh used for this study. To reach results similar to the GF one with $N_e = 150$, we need around $N_e = 800$ cells for the classical WENO5 method, see Figure 7.4.



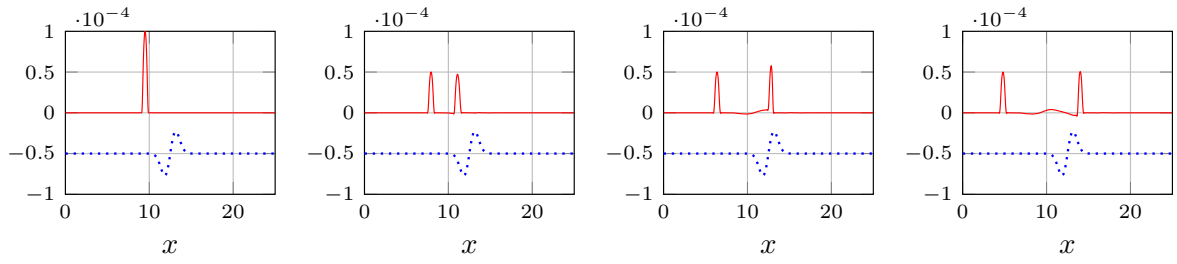
(a) $t = 0$ (b) $t = 0.5$ (c) $t = 1$ (d) $t = 1.5$

Figure 7.2: Small perturbation of the lake at rest solution computed with the GF-WENO5 WB scheme: $h - h_{eq}$ (red) and rescaled b (blue) with $N_e = 150$.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 1$ (d) $t = 1.5$

Figure 7.3: Small perturbation of the lake at rest solution computed with the WENO5 scheme: $h - h_{eq}$ (red) and rescaled b (blue) with $N_e = 150$.



(a) $t = 0$ (b) $t = 0.5$ (c) $t = 1$ (d) $t = 1.5$

Figure 7.4: Small perturbation of the lake at rest solution computed with the WENO5 scheme: $h - h_{eq}$ (red) and rescaled b (blue) with $N_e = 800$.

7.3.3 Steady states with smooth bathymetry without friction ($n = 0$)

Here, we test the method for some moving equilibria steady state problems. We run the tests up to convergence towards steady state in different situations. In the subcritical and supercritical tests the bathymetry is smooth and equal to (7.48), as we want to assess the high order accuracy of the schemes. For the transcritical tests we use a modification of the bathymetry used in [113], to study a very similar discontinuous problem. Depending on the initial and boundary conditions set at the borders of the domain, the flow may be supercritical, subcritical or transcritical. The meshes used are made up by $N_e \in \{25, 50, 75, 100, 125, 150, 250, 500\}$ uniform cells. We consider the following three sets of initial and boundary conditions:

- Supercritical flow

$$\begin{aligned} &h(x,0) = 2 - b(x), \qquad q(x,0) \equiv 0, \\ &h(0,t) = 2, \qquad q(0,t) = 24, \end{aligned} \tag{7.52}$$

- Subcritical flow

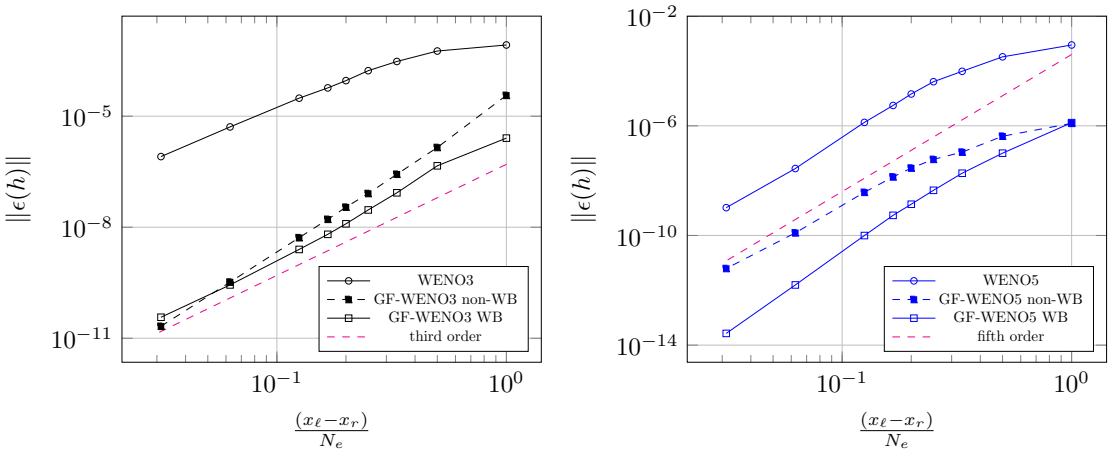
$$\begin{aligned} &h(x,0) = 2 - b(x), \qquad q(x,0) \equiv 0, \\ &q(0,t) = 4.42, \qquad h(25,t) = 2, \end{aligned} \tag{7.53}$$

- Transcritical flow

$$
\begin{aligned}
b(x) &= \begin{cases} 0.2\exp\left(1-\frac{1}{1-\left(\frac{|x-10|}{5}\right)^2}\right), & \text{if } |x-10|<5, \\ 0, & \text{else,} \end{cases} \\
h(x,0) &= 0.33 - b(x), \qquad q(x,0) \equiv 0, \\
q(0,t) &= 0.18, \qquad h(25,t) = 0.33.
\end{aligned}
\tag{7.54}
$$

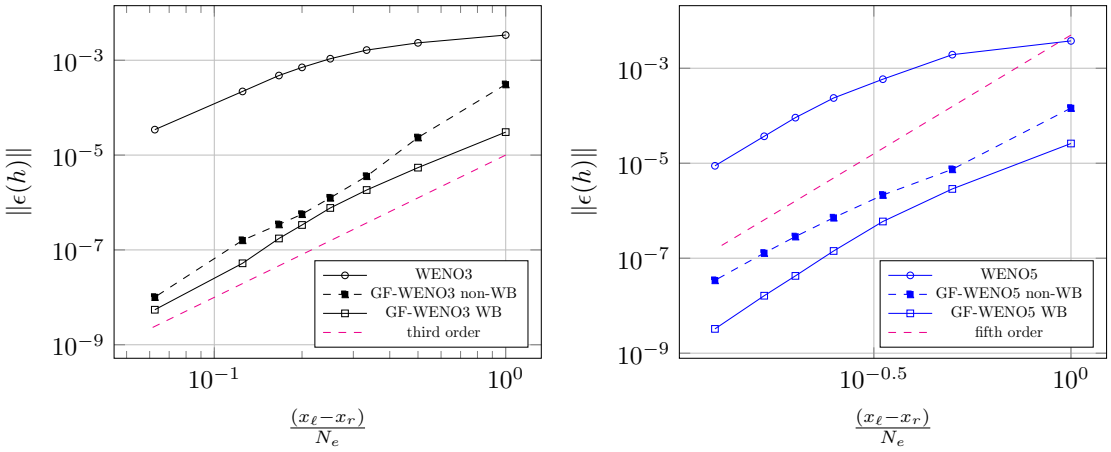
The gravitational constant is set to $g = 9.812$ for all these tests. For these three cases, we compare the results obtained using a classical WENO finite volume scheme and the new approach based on flux globalization. As already mentioned, for supercritical and subcritical cases, the bathymetry (7.48) allows us to perform convergence tests for very high order methods, when also the flow is smooth. For the transcritical case with shock, only a qualitative analysis of the test is performed. Hence, when supercritical and subcritical flows are of interest, we can study the convergence properties of the new scheme by finding the exact solution given by the non-linear equations taken from [113]. Both the WB and non-WB versions of the scheme have been run to compare the influence of the formulation on the ability of preserving the balanced steady state solution. Finally, we also run the same test cases with the classical WENO3 and WENO5 schemes.

Convergence curves for supercritical and subcritical flows are depicted in Figures 7.5 and 7.6, respectively. All curves, both for WENO3 and WENO5, show the correct third and fifth order accuracy. However, it should be noticed that the GF formulation allows a much better prediction of the solution characterized by a discretization error that is by far lower compared to the classical method. Introducing then the WB formulation allows even better convergence trends with slightly lower errors with the advantage of preserving also the lake-at-rest solution. In some cases, e.g. supercritical flow with GF-WENO5 WB and GF-WENO3 non-WB, we observe a superconvergence behavior, which might be caused by the extra constraints that the flux globalization and WB techniques impose. The behavior is still under investigation. After the convergence analysis, we focus on a more qualitative analysis of the results computed with the WENO5 and GF-WENO5 methods. To do so we introduce a set of four representative variables that we are going to use to compare the classical WENO5 with the new GF-WENO5 WB. The three main variables that we used to assess the new method are η, q and K (K is only defined for GF formulation). In particular, it should be noticed that, by construction, the methods based on flux globalization preserve moving equilibria, i.e., $q_x \equiv 0$, $K_x \equiv 0$. This means that the GF-WENO5 will approximate these quantities up to the order of the residual of the time derivative at the end of the simulation. In our case, we are able to preserve the constant q



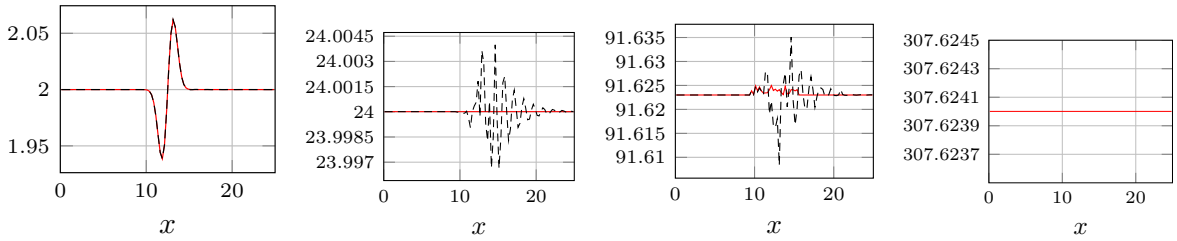
(a) WENO3 (b) WENO5

Figure 7.5: Supercritical flow: convergence tests with WENO3 and WENO5.



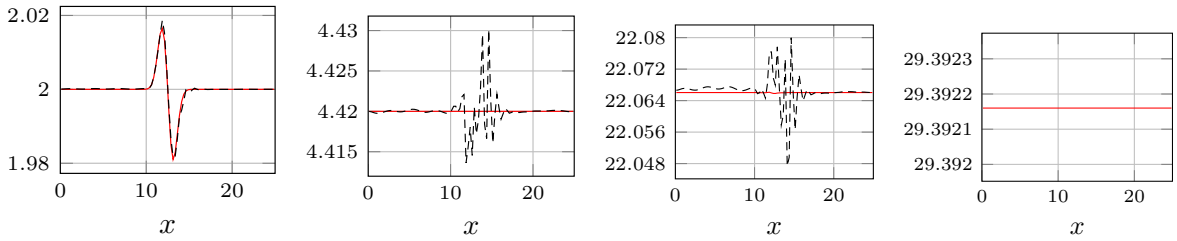
(a) WENO3 (b) WENO5

Figure 7.6: Subcritical flow: convergence tests with WENO3 and WENO5.



(a) η (b) q (c) Υ (d) K

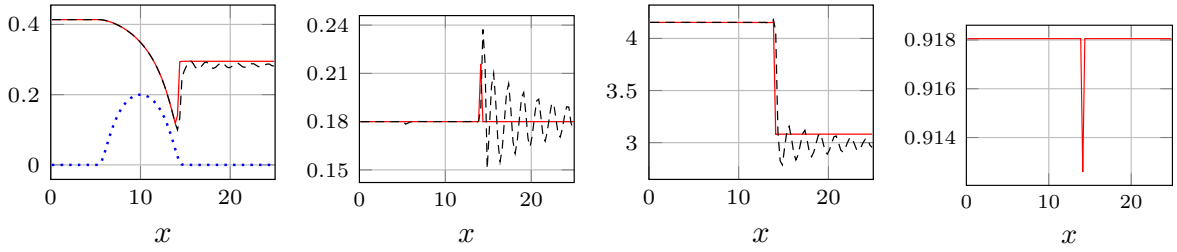
Figure 7.7: Supercritical flow: relevant variables computed with GF-WENO5 (red continuous line) and WENO5 (black dashed line) schemes with $N_e = 100$.



(a) η (b) q (c) Υ (d) K

Figure 7.8: Subcritical flow: relevant variables computed with GF-WENO5 (red continuous line) and WENO5 (black dashed line) schemes with $N_e = 100$.

and K up to $\sim 10^{-9}$. The last variable we decided to study is the aforementioned Υ, which corresponds to the smooth formulation of the more general global flux introduced herein. Although our approach is developed to preserve other equilibria, this variable allows to get more insights about the capability of the new algorithm since, for smooth flows, the analytical Υ should be constant at equilibrium. Solutions for supercritical and subcritical flows are shown in Figures 7.7 and 7.8. For both cases, it is clear that q and K are well-preserved, and Υ is much better predicted with respect to the one computed through the classical WENO5 method. The test case that stands out more among the three situations considered here is the transcritical flow with shock in Figure 7.9. The WENO5 method introduces spurious oscillations where the shock occurs that are then propagated in the rest of the computational domain making the overall solution spoiled. By using the new GF-WENO5 approach, oscillations are not present and the correct solution is recovered before and after the shock. It can be noticed that, when discontinuous flows are of interest, Υ is not globally constant but features a jump where the shock occurs, see Figure 7.9.



(a) η (b) q (c) Υ (d) K

Figure 7.9: Transcritical flow: relevant variables computed with GF-WENO5 (red continuous line), WENO5 (black dashed line) schemes and b (blue dotted line) with $N_e = 100$.

7.3.4 Small perturbation of steady states with smooth bathymetry without friction ($n = 0$)

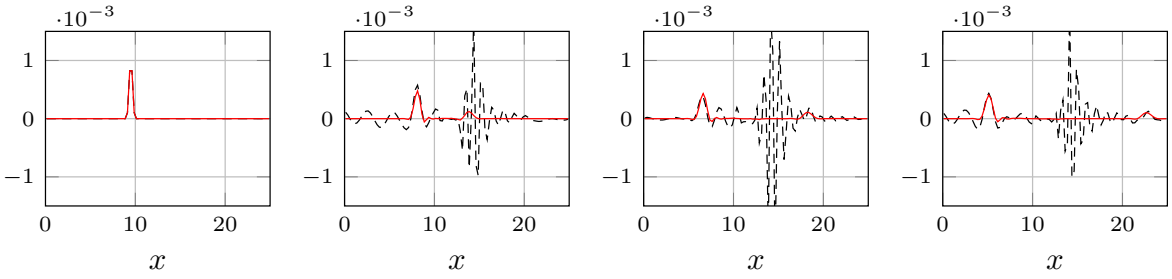
In this section, we add a perturbation to the tests of Section 7.3.3 to compare the GF-WENO5 and the classical WENO5 methods for supercritical and subcritical flows. For the subcritical case we use the perturbation (7.50) with $\alpha = 10^{-3}$. In Figure 7.10 we plot the solution for both methods at different timesteps with $N_e = 100$, while in Figure 7.11 we use $N_e = 800$. It can be noticed that both methods converge towards the exact solution, but with the GF-WENO5 method, even with a coarse mesh we obtain a very accurate approximation of the perturbation, while the WENO5 method performs poorly. A much more refined mesh is needed for the classical methods to nicely approximate this kind of solutions.

For the supercritical case we use the perturbation (7.50) with $\alpha = 10^{-4}$. In Figure 7.12 we plot the solution for both methods at different timesteps with $N_e = 100$, while in Figure 7.13 we use $N_e = 800$. Similar conclusions can be drawn from these simulations: the GF-WENO5 is accurate in representing the perturbation even for very coarse meshes, while the WENO5 needs more discretization cells.

7.3.5 Steady state with discontinuous bathymetry without friction ($n = 0$)

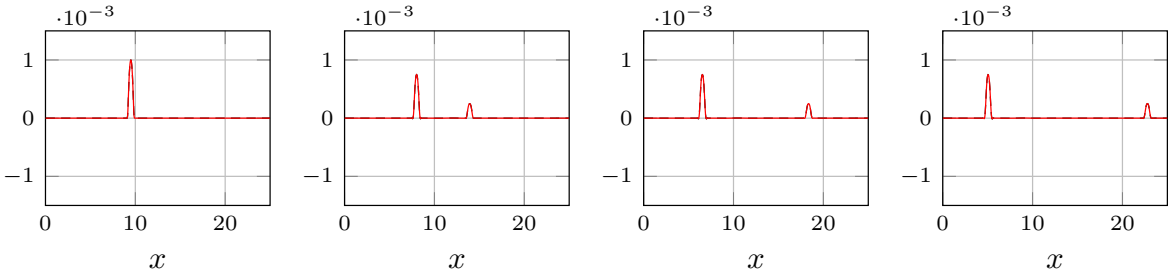
Here, we change the bathymetry and test the subcritical and supercritical flows with a discontinuous step. The bathymetry is defined by

$$b = \begin{cases} 0.2 & \text{if } 8 < x < 12, \\ 0. & \text{else.} \end{cases} \tag{7.55}$$



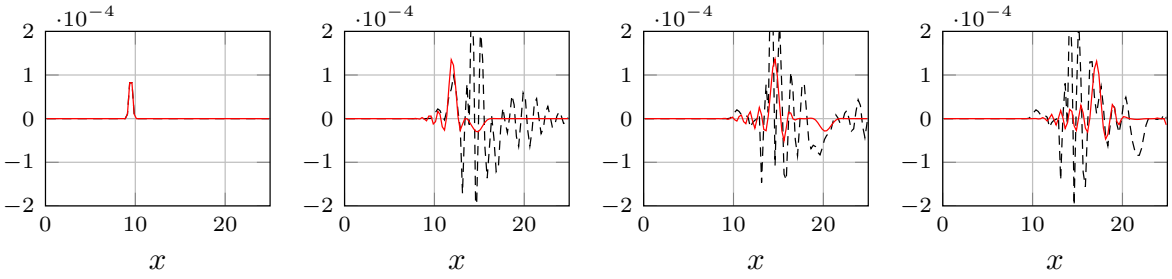
(a) $t = 0$ (b) $t = 0.66$ (c) $t = 1.33$ (d) $t = 2$

Figure 7.10: Small perturbation of the subcritical solution computed with the WENO5 scheme (black dashed) and GF WENO5 (red continuous): $h - h_{eq}$ with $N_e = 100$.



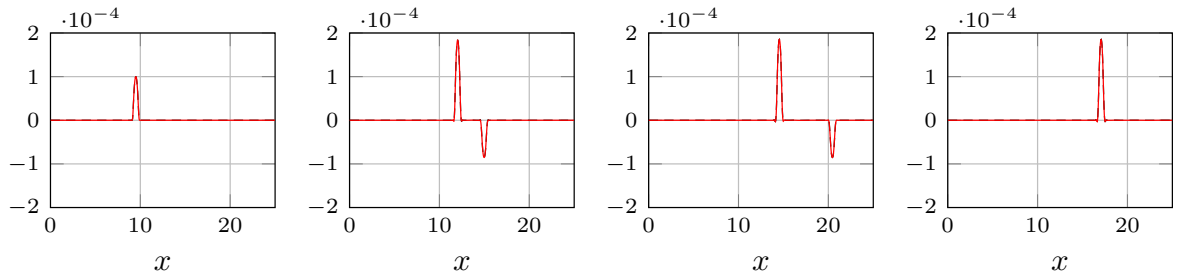
(a) $t = 0$ (b) $t = 0.66$ (c) $t = 1.33$ (d) $t = 2$

Figure 7.11: Small perturbation of the subcritical solution computed with the WENO5 scheme (black dashed) and GF WENO5 (red continuous): $h - h_{eq}$ with $N_e = 800$.



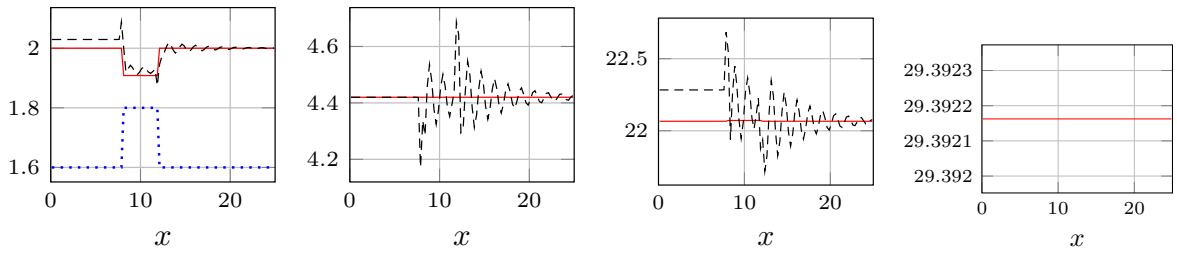
(a) $t = 0$ (b) $t = 0.66$ (c) $t = 1.33$ (d) $t = 2$

Figure 7.12: Small perturbation of the supercritical solution computed with the WENO5 scheme (black dashed) and GF WENO5 (red continuous): $h - h_{eq}$ with $N_e = 100$.



(a) $t = 0$ (b) $t = 0.66$ (c) $t = 1.33$ (d) $t = 2$

Figure 7.13: Small perturbation of the supercritical solution computed with the WENO5 scheme (black dashed) and GF WENO5 (red continuous): $h - h_{eq}$ with $N_e = 800$.



(a) η (b) q (c) Υ (d) K

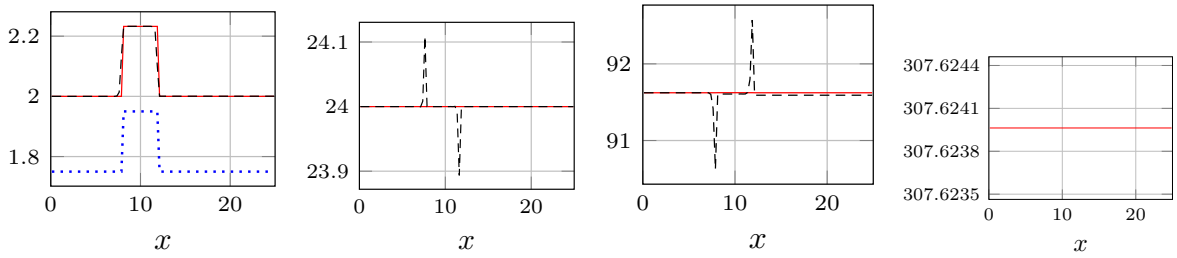
Figure 7.14: Subcritical flow: relevant variables computed with GF-WENO5 (red continuous line) and WENO5 (black dashed line) schemes and rescaled b (blue dotted line) with $N_e = 100$.

In Figure 7.14 we test the subcritical flow with the same initial conditions of Section 7.3.3 with $T = 500$ and $N_e = 100$. We can observe that the GF-WENO5 performs amazingly without producing any oscillations and obtaining errors of the order of 10^{-10}, while the WENO5 scheme wildly oscillates after the discontinuities of the bathymetry.

For the supercritical case we have used the initial conditions of Section 7.3.3 till final time $T = 50$ with $N_e = 100$. In Figure 7.15 we observe that WENO5 performs better than before, still producing spurious oscillations and not exactly catching the outflow solution. On the other side, GF-WENO5 obtains constant global fluxes with a machine precision accuracy.

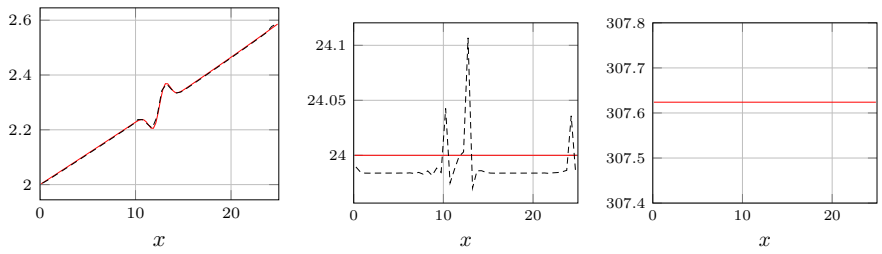
7.3.6 Steady states with smooth bathymetry with friction ($n = 0.05$)

In this section we focus on the supercritical and subcritical flows studied in Section 7.3.3 including the friction term in the source term. As for the previous cases, also when friction with constant Manning coefficient n is present, we can obtain moving equilibria. Again



(a) η (b) q (c) Υ (d) K

Figure 7.15: Supercritical flow: relevant variables computed with GF-WENO5 (red continuous line) and WENO5 (black dashed line) schemes and rescaled b (blue dotted line) with $N_e = 100$.



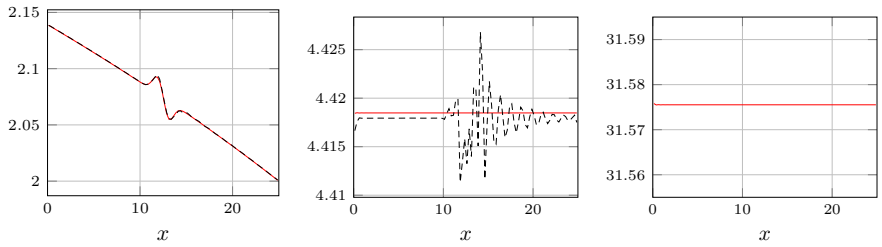
(a) η (b) q (c) K

Figure 7.16: Supercritical flow with friction: relevant variables computed with GF-WENO5 (red continuous line) and WENO5 (black dashed line) schemes.

the quantity that are preserved at equilibrium are q and K. That is why it is interesting to perform simulations similar to the previous ones comparing standard methods with GF ones.

We consider the subcritical case defined in (7.52) and the supercritical case defined in (7.53) with the same bathymetry, on which we add the friction term with Manning coefficient $n = 0.05$. In the supercritical case, the friction term implies a slow down of the physical speed from left to right and a consequent increasing of η from left to right. In the subcritical case, conversely, we expect h to decrease from left to right and the speed to increase. The variable Υ is not conserved and, for these tests, there is not another constant variable that can be easily computed analytically, hence, we do not plot it.

We display in Figures 7.16 and 7.17 both the solutions computed with the new GF-WENO5 WB scheme and the classical WENO5. It should be noticed that both schemes obtain valid and consistent results with the expected solution. The difference between the schemes is remarkable and the global flux variables clearly highlights it. The WENO5 case, without the global flux, is characterized by strong spurious oscillations around the area where the effect of the bathymetry is stronger. On the other side, the GF-WENO5 results are very precise and are able to preserve the global flux variables up to $\sim 10^{-9}$.



(a) η (b) q (c) K

Figure 7.17: Subcritical flow with friction: relevant variables computed with GF-WENO5 (red continuous line) and WENO5 (black dashed line) schemes.

7.4 Chapter summary

In this chapter, we presented a novel arbitrary high-order well-balanced FV method based on flux globalization for the shallow water equations with source terms. The high order accuracy is obtained through a WENO reconstruction performed on the variables of interest, i.e. free surface level and global fluxes, that allows to exactly preserve the moving water equilibria characterized by constant global fluxes. Afterwords, by introducing a particular quadrature procedure for the source flux and considering a jump of the global fluxes at each interface, it was also possible to embed the preservation of other equilibria, e.g. "lake-at-rest". Several simulations have been performed to assess the structure-preserving and convergence properties of the new family of schemes. The preservation of the constant flux has been verified on academic moving water test cases (subcritical, transcritical and supercritical), while the well-balancedness, with respect to the lake-at-rest equilibrium, has been tested on the classical "lake-at-rest" and "perturbed lake-at-rest" cases. Once the space discretization is well implemented, the introduction of additional source terms is straightforward, e.g. Manning friction, as demonstrated in Section 7.3. With the presented scheme, we were able to outperform classical methods in many situations, obtaining much more accurate solutions and useful properties also at the discrete level. Moreover, the high order accuracy of the scheme allows to obtain precise solutions also when the equilibria are not reached.

Conclusions and Future Perspectives

In this thesis we have studied several approaches to improve the high order numerical simulation of systems of hyperbolic conservation and balance laws. The objective has always been that of obtaining the best result possible while saving computational time. For this reason, we developed several numerical techniques with the goal of tackling some of the challenges imposed by high order methods:

- imposition of boundary conditions;
- treatment of discontinuities;
- structure-preservation.

Accurate boundary conditions

Regarding the imposition of boundary conditions (BCs), it is known that high-order methods have to be coupled with high-order meshes in order to have a consistent enforcement of BCs. In this context, in Chapter 3 we presented a simple correction of arbitrary order [87] to solve compressible flow problems with high order finite element methods using polygonal meshes to discretized the curved boundaries of the domain. The paradigm of the shifted boundary method [206, 207] has been used to overcome the (second-order) *geometrical error* [222] due to the classical treatment of curved boundaries. This allows to save the computational time and coding efforts that one would need to couple a finite element code with curvilinear grids: mesh generation process, iso-parametric transformation and special quadrature formulas. The new approach has been validated over a large set of test-cases including two- and three-dimensional geometries, and steady and unsteady flows. The system of Euler equations for gasdynamics has been used for validation but this approach can be easily extended to other models. The boundary correction can easily be adapted to shock limiters (sub-cell MOOD [92, 116] in our case) in order to run simulations with shock waves and other discontinuities.

Further extensions of the present work will concern first of all its application in the context of moving meshes and moving interfaces [139], and then the additional development necessary for its usage in fully embedded computations [232, 197], which do not

require any conformal meshing of internal boundaries. Finally, we also plan to use a similar approach in the context of more complex models as Navier-Stokes equations, for which the extension should be straightforward, and the MHD equations [127] where also the magnetic field should be correctly handled.

Treatment of discontinuities

About the special treatment reserved to shocks, in Part II, we discussed two approaches that take into account the presence of discontinuities by considering them as internal boundaries of the computational domain, characterized by special boundary conditions dictated by the Rankine-Hugoniot equations. Both these approaches allow to overcome the accuracy degradation common of all shock-capturing methods following two different strategies.
Firstly, in Chapter 4, we have described the key algorithmic features of the 10-year-long shock-fitting technique [235] for unstructured meshes, called Unstructured Discontinuity Fitting [68] (UnDiFi), available in an open-source dedicated repository. UnDiFi exploits the capabilities of unstructured grids to easily adapt to complex geometries to simplify the fitting/tracking techniques previously introduced and allows to have a flexible framework by which it is possible to couple different vertex-centered solvers and mesh generators. An extensive set of ready-to-run test-cases demonstrates the described features and current capabilities of the code, which we have shown to be able to deal with compressible flows featuring either isolated or mutually interacting discontinuities.

Following this work, in Chapter 5 we presented a novel technique, called extrapolated Discontinuity Tracking [89, 90] (eDIT), to simulate flows with different kind of (interacting) discontinuities. As already mentioned, both techniques provides genuinely second-order-accurate results even for flows featuring very strong shocks. However, the eDIT method is based on a different philosophy. The majority of shock-fitting/front-tracking methods are characterized by either a re-meshing/adaptation step or a modification in the update formula of the scheme to take into account the presence of these new internal boundaries. The philosophy behind eDIT is built around the approximate domain methods, in particular the shifted boundary method [206, 207]. Hence, the tracked discontinuities are now considered as *immersed* boundaries and the information coming from the Ranking-Hugoniot equations are transferred from the real shock manifold to its approximated position. This procedure avoids re-meshing/adaptation steps and the small-cut-cell problem, common of many shock-tracking/fitting methods. Moreover, due

to its generic formulation, the new approach can be easily coupled with different solvers and mesh data structure [23, 25, 24].

Although many improvements of the fitting/tracking technique have been made so far, further developments are in progress, including its coupling with a structured-grid, finite volume solver for two-dimensional [23] and three-dimensional flows [24]. Simple extensions of this work may concern more complex models such as Navier-Stokes equations and two-phase flows. Moreover, by increasing the order of the polynomial used to approximate the solution, and the order of the Taylor expansions, the method could be easily extended to higher order (more than two) formulations. Finally, its applications to sensitivity analysis [132] in presence of shocks will be also topics of future works.

Structure-preserving

Part III of this thesis concerns the development of new high order structure-preserving schemes. Structure-preserving schemes are particular numerical discretization techniques that are able to preserve at the discrete level some physical properties that characterize the problem at the continuous level.
In Chapter 6, we have presented a new well-balanced, positivity-preserving, arbitrary high-order numerical method for the solution of the shallow water equations. In particular, by recasting the WENO semi-discretization in terms of productions-destructions terms, we were able to use the modified Patankar Deferred Correction (mPDeC) methods to ensure the unconditionally positivity of the water height. Thanks to the implicit structure of the first equation, the restriction on the CFL number comes only from the explicit DeC-solver used for the momentum equations (CFL$\sim$1). By doing so, the CFLs used are much larger than those used when working with classical explicit SSPRK methods in combination with positivity-preserving limiters [300], for which the CFL must be lowered to 1/12 (for WENO5). By applying mPDeC, we avoid those issues and the price to pay is that of solving a (very sparse) linear system for the water height. However, as mentioned before, this increase in computational costs is around 10% in our numerical simulations.

In this work we have only considered positivity preservation and well-balanced properties. In the future, we would like to extend our investigation to more general well-balancing techniques [218, 219] and entropy conservative/stable methods. Regarding the former, various approaches exist as described *inter alia* in [9, 15, 79, 133, 135, 254], but from our perspective the convex limiting strategies seems the most promising one [164, 193]. In order to do so, the basic stability properties of Patankar methods as ODE solvers have to be first fully understood [178, 286]. Finally, herein we have focused on the shallow water

system. However, the method can be also adapted to more complex models, e.g. the shallow water equations together with biochemical processes like algae bloom in oceans, seas and open water cancels or the Euler equations of gasdynamics, where special treatments for the pressure positivity are necessary [307, 297]. Another issue that we would like to address is making this approach as cheap as possible in terms of computational costs; to do so an optimized version of the code should be implemented.

In the context of preserving stationary solutions produced by balance laws, in Chapter 7, we also presented a novel arbitrary high-order well-balanced finite-volume method based on flux globalization for the shallow water equations. The high order accuracy is obtained through a WENO reconstruction performed on the variables of interest, i.e. free surface level and global fluxes, permitting to exactly preserve the constant fluxes in moving water equilibria. By designing a particular strategy, it was also possible to embed into the new scheme a particular solution: the "lake-at-rest". This was possible by introducing a particular quadrature procedure for the source flux and allowing it to jump at each interface. It should be also mentioned that, once the space discretization is well implemented, the introduction of additional source terms (e.g. Manning friction) is straightforward compared to similar techniques such as those based on the hydrostatic reconstruction [218, 219, 38]. With the presented scheme, we were able to outperform classical methods in many situations, obtaining much more accurate solutions and useful properties also at the discrete level.

There are several extensions to this work that could be carried out. The authors are already studying a DGSEM-like formulation to simplify the global flux procedure. Another interesting topic could be that of studying different balance laws for which complicated equilibria could be object of interest; in this context, we would like to extend our method for balance laws with non-conservative products treated with path-conservative strategies. Finally, a multi-dimensional version of the global flux must be carefully defined to work with more realistic problems.

Appendix

A.1 Nodal gradient reconstruction

As explained in Section 5.1, in the present version of the *eDIT* algorithm, the computational domain is tesselated into triangles and bi-linear shape functions with unknowns stored at the vertices of the triangulation that provide the functional representation of the dependent variable, which is Roe's parameter vector $\mathbf{Z} = \sqrt{\rho}\,(1, H, u, v)^t$. Using the aforementioned setting, the gradient of $\mathbf{Z}$ is constant within each triangular cell T and can be readily computed as follows:

$$\nabla \mathbf{Z}_T \;=\; \frac{\sum_{k=1}^{3} (\mathbf{Z}_k\, \mathbf{n}_k)}{2\, A_T}, \tag{56}$$

where $\mathbf{n}_k$ denotes the normal to the face opposite vertex k, scaled by its length and pointing inside triangle T and A_T denotes the area of triangle T.

As explained in Section 5.1.5, however, in order to perform the data trasfer between the surrogate boundaries and the discontinuity-front, it is also necessary to compute the gradient of $\mathbf{Z}$ within every grid-point of the surrogate boundaries; this is accomplished using either the Green-Gauss (GG) or Zienkiewicz-Zhu (ZZ) approaches to be detailed hereafter.

Green-Gauss reconstruction The GG reconstruction consists in camputing the nodal gradient in grid-point i as the area-weighted average of the cell-wise constant gradients of all triangles that surround grid-point i, meaning that

$$\nabla \mathbf{Z}_i \;=\; \sum_{T} \theta_T \nabla \mathbf{Z}_T, \tag{57}$$

where

$$\theta_T \;=\; \frac{A_T}{\sum_T A_T} \quad \text{such that} \quad \sum_{T} \theta_T \;=\; 1. \tag{58}$$

The summations in Equations (57) and (58) range over all triangles surrounding the

grid-point i.

Zienkiewicz-Zhu reconstruction Similarly to the GG reconstruction, all steps involved in the ZZ gradient-reconstruction have to be repeated within all grid-points of the surrogate boundaries. In order to alleviate the notation, however, we shall hereafter drop the index that refers to the grid-point and use subscript k to refer to one of the four components of the parameter vector and subscript j to refer the cartesian coordinates, i.e. $(x_1, x_2) = (x, y)$. The starting point in the ZZ reconstruction consists in using a linear polynomial expansion of each component of the gradient, i.e.

$$\left(\frac{\partial Z_k}{\partial x_j}\right) \;=\; \mathbf{P} \cdot \mathbf{a}_{k,j}, \tag{59}$$

where

$$\mathbf{P} \;=\; [\,1\,,\,x_1\,,\,x_2\,] \quad \text{and} \quad \mathbf{a}_{k,j} \;=\; [\,a_0\,,\,a_1\,,\,a_2\,]_{k,j}. \tag{60}$$

Following the Zienkiewicz-Zhu (ZZ) patch recovery procedure, the unknown parameters $\mathbf{a}_{k,j}$ are computed via a least-squares fit which amounts to minimizing the following test function:

$$F(\mathbf{a}_{k,j}) \;=\; \sum_T \theta_T \left[\left(\frac{\partial Z_k}{\partial x_j}\right)_T \;-\; \mathbf{P}(x_1^T, x_2^T) \cdot \mathbf{a}_{k,j}\right]^2 \tag{61}$$

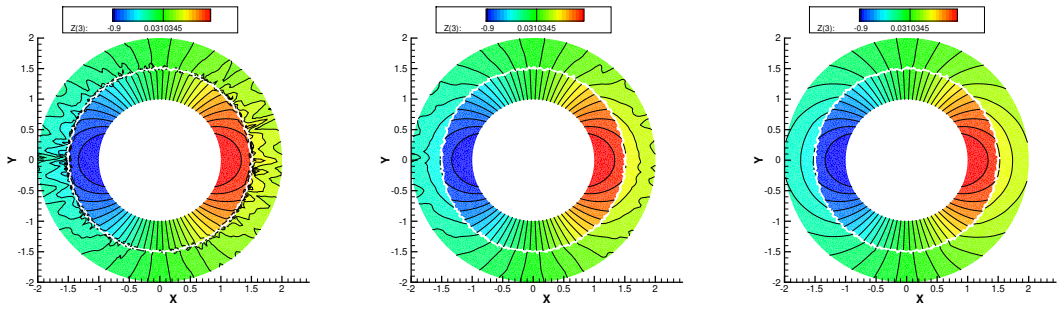
The summation in Equation (61) ranges over all triangles surrounding the given grid-point, the gradient in (61) is computed element-wise according to (56) and vector $\mathbf{P}$ is computed in the barycentric coordinates of triangle $T : \left(x_1^T, x_2^T\right)$. Given that three unknowns, $\mathbf{a}_{k,j}$, must be computed for each component of $\mathbf{Z}$ and each cartesian coordinate, a stencil of at least three triangles is required. The minimization problem defined by (61) can be solved in matrix form:

$$\mathbf{a}_{k,j} \;=\; \underline{\underline{A}}^{-1}_{k,j} \cdot \mathbf{b}_{k,j}, \tag{62}$$

where,

$$\underline{\underline{A}}_{k,j} \;=\; \sum_T \theta_T\, \mathbf{P}^t(x_1^T, x_2^T)\, \mathbf{P}(x_1^T, x_2^T) \quad \text{and} \quad \mathbf{b}_{k,j} \;=\; \sum_T \theta_T\, \mathbf{P}^t(x_1^T, x_2^T) \left(\frac{\partial Z_k}{\partial x_j}\right)_T. \tag{63}$$

The *GG* reconstruction, Equation (57), can be recovered from the *ZZ* reconstruction (62) by choosing a constant polynomial expansion: $\mathbf{P} \;=\; [\,1\,,\,0\,,\,0\,]$.



(a) Iteration 1 (b) Iteration 201 (c) Converged solution

Figure 18: Transonic source flow: pseudo-temporal evolution of the $eDIT$ simulation starting from a SC solution (in terms of $\sqrt{\rho}u$).

A.2 Pseudo-temporal evolution and iterative convergence of the $eDIT$ method

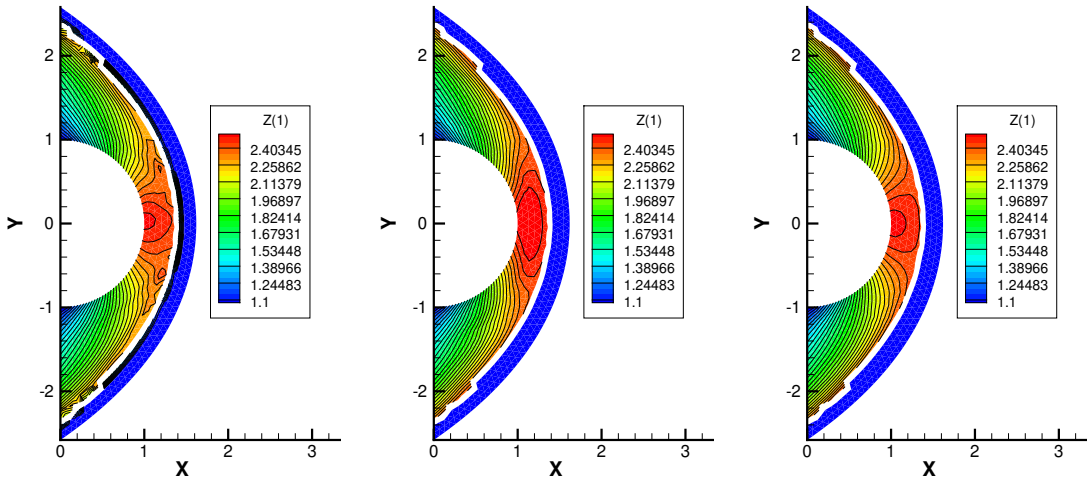
In this Appendix we give further insight into the pseudo-temporal evolution of the flow-field to show how the $eDIT$ algorithm, starting from a converged SC solution used as initial condition, leads to a steady, oscillation-free, shock-fitted result.

Figures 18, 19 and 20 show a sequence of three frames that refer to different instances of the pseudo-temporal evolution of the solution for the three test-cases already described in Sections 5.3.1, 5.3.2 and 5.3.6. In order to improve readability, the shock-mesh has not been plotted. It can be seen that the $eDIT$ method requires a few hundred pseudo-time steps to get rid of the severe oscillations inherited by the SC calculation used to initialize the flow-field. Further iterations are required while the shock slows down, and it settles to its steady location.

A.3 Reconstruction of the primitive variable

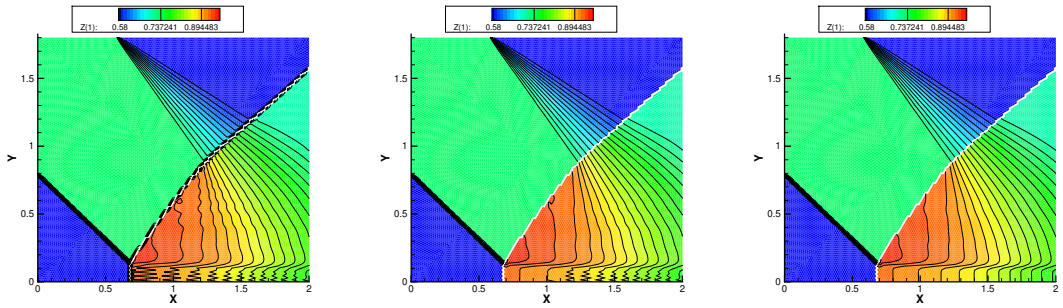
The goal of this section is that of deepening the procedure to compute the coeffiecients of the polynomials and linear weights needed to compute the WENO procedure in Section 2.2.2. The results and the actual coefficients for WENO5 with 4 Gaussian quadrature points are written in Section A.4.

In the following assemble the system that allow to find the coefficients for the WENO polynomial at any quadrature point. The symbolic Matlab script coded for this purpose is also available at [88]. With few adjustments, the script can be used to compute all the ingredients needed for a WENO reconstruction of arbitrary high order with arbitrary high order quadrature formulae.



(a) Iteration 1 (b) Iteration 101 (c) Converged solution

Figure 19: Hypersonic flow past a blunt body: pseudo-temporal evolution of the *eDIT* simulation starting from a *SC* solution (in terms of $\sqrt{\rho}$).



(a) Iteration 1 (b) Iteration 201 (c) Converged solution

Figure 20: Steady Mach reflection: pseudo-temporal evolution of the *eDIT* simulation starting from a *SC* solution (in terms of $\sqrt{\rho}$).

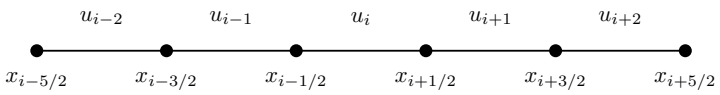


Figure 21: Stencil of five cell averages for WENO5 reconstruction.

The reconstruction of the primitive variables is needed in a high order finite volume method, as only cell averages are available from the previous time step. For the sake of simplicity we are going to work in a simpler one-dimensional scalar framework since the reconstruction is done dimension-by-dimension. We consider a scalar function $u(x)$ whose cell averages u_i are known. We aim at reconstructing this variable as a polynomial $v(x)$, where the polynomial may vary in different points. In particular, it will be useful to use a primitive of $v(x)$ which we denote by $\mathcal{P}(x) = \int_{x_0}^{x} v(s)\mathrm{d}s$. Indeed, we can impose that for every cell average, we have

$$u_i = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x)\ \mathrm{d}x \stackrel{!}{=} \int_{x_{i-1/2}}^{x_{i+1/2}} v(x)\mathrm{d}x = \frac{\mathcal{P}(x_{i+1/2}) - \mathcal{P}(x_{i-1/2})}{\Delta x}. \tag{64}$$

The stencil considered for this example is the one used for the WENO5 reconstruction, which is made up by five cell averages as denoted in Figure 21.

The next step consists in using interpolating polynomials $\varphi_{j-1/2}$, e.g. Lagrange polynomials at cell interfaces, to approximate the primitive $\mathcal{P}(x)$, i.e.,

$$\mathcal{P}(x) = \sum_{j=k_1}^{k_2+1} a_{j-1/2}\,\varphi_{j-1/2}(x) \qquad \text{where} \qquad \varphi_{j-1/2}(x) = \prod_{\ell=k_1, i\neq j}^{k_2+1} \frac{x - x_{\ell-1/2}}{x_{j-1/2} - x_{\ell-1/2}} \tag{65}$$

where k_1 and k_2 are the extreme indexes of the considered stencil. So, given then k_2-k_1+1 cell averages, the degree of the polynomial $\mathcal{P}$ will be k_2-k_1+2. For instance, when working with WENO5, the reconstruction is composed by a linear combination of three lower order polynomials of degree 3, so with $\mathcal{P} \in \mathbb{P}_4$, obtained through three cell averages. Then, the 3 polynomials, which then depend on the whole 5 cells stencil, will be be combined with weights which depends on the quadrature points into a fifth order accurate reconstruction. In order to have this result, we need to compute the aforementioned three lower order polynomials and a high order polynomial made up by information coming from the whole stencil.

Let us begin with the procedure to compute the high order polynomial with all available cell averages, i.e., $k_1 = -2$ and $k_2 = 2$. The lower order polynomials can be easily computed following the same approach explained hereafter. In this case $\mathcal{P}(x)$ is a sixth order polynomial and $v(x) = \mathcal{P}'(x)$ is a fifth order polynomial which gives the right

accuracy order. Using (64) for all cell averages in the stencil with the definition of $\mathcal{P}$ given in (65) and using the property of Lagrangian polynomials $\varphi_{j-1/2}(x_{\ell-1/2}) = \delta_{j,\ell}$, we obtain a system of equations for the coefficients $a_{j-1/2}$ with solution

$$a_{j-1/2} = \begin{cases} 0, & \text{if } j = k_1, \\ \sum_{\ell=-k_1}^{-k_1+j-1} u_{i+\ell}, & j = k_1+1, \ldots, k_2+1. \end{cases} \tag{66}$$

Finally, the expression for the high order, *ho*, approximation polynomial $v^{ho}(x)$ can be written as

$$v^{ho}(x) = \sum_{j=k_1}^{k_2+1} a_{j-1/2}\varphi'_{j-1/2}(x) = \sum_{\ell=-k_1}^{k_2} c_\ell^{ho}(x)\, u_{i+\ell} = \sum_{\ell=-2}^{2} c_\ell^{ho}(x)\, u_{i+\ell}, \tag{67}$$

where c_ℓ^{ho} are obtained collecting all the coefficients and basis functions related to $u_{i+\ell}$. In this way, for any quadrature point $\tilde{\xi} \in [x_{i-1/2}, x_{i+1/2}]$ we can evaluate this high order polynomial $v^{ho}(\tilde{\xi})$. Following the same procedure for three lower order polynomials, *lo*, associated to the 3-cells stencils $S^0 = \{u_i, u_{i+1}, u_{i+2}\}$, $S^1 = \{u_{i-1}, u_i, u_{i+1}\}$ and $S^2 = \{u_{i-2}, u_{i-1}, u_i\}$, we obtain an expression for these low order polynomials

$$v_j^{lo}(x) = \sum_{\ell=-j}^{2-j} c_{j\ell}^{lo}(x)\, u_{i+\ell}. \tag{68}$$

The last step concerns the computation of the ideal linear weights. These are the weights that allow to recover the high order reconstruction from a linear combination of the lower order ones for a given quadrature point $\tilde{\xi}$. Therefore, we need to find the linear weights d_j such that

$$v^{ho}(\tilde{\xi}) = \sum_{j=0}^{2} d_j\, v_j^{lo}(\tilde{\xi}) \quad \Longleftrightarrow \quad \sum_{\ell=-2}^{2} c_\ell^{ho}(\tilde{\xi})\, u_{i+\ell} = \sum_{j=0}^{2} \sum_{\ell=-j}^{2-j} d_j\, c_{j\ell}^{lo}(\tilde{\xi})\, u_{i+\ell}, \quad \forall u_{i+\ell}. \tag{69}$$

Since (69) must hold for any quintuplet $\{u_{i+\ell}\}_{\ell=-2}^{2}$, we can write a system of five equations in the 3 linear weights d_j for each quadrature point $\tilde{\xi}$, i.e.,

$$\sum_{j=0}^{2} d_j\, c_{j\ell}^{lo}(\tilde{\xi}) = c_\ell^{ho}(\tilde{\xi}), \quad \forall \ell = -2, \ldots, 2. \tag{70}$$

This is an overdetermined system with five equations and only three unknowns d_j that can be easily solved by means of a least squares method. Moreover, the solutions found verify exactly all the equations, implying that some of the equations are linearly dependent.

A.4 WENO recontruction ($r = 3$) with four-point Gaussian quadrature rule

The goal of this section is to present the fifth order WENO reconstruction with four-point Gaussian quadrature rule. Up to our knowledge, there is no reference in literature that explicitly define linear weights and polynomial coefficients needed for this WENO reconstruction. Reference [285] well described the fifth-order WENO5 with two-point Gaussian quadrature rule, which unfortunately does not allow to go beyond fourth order. Instead, with the four-point Gaussian quadrature rule, one could reach even eighth order. Let us consider a one-dimensional cell $[\xi_{i-1/2}, \xi_{i+1/2}]$, we hereafter provide the expressions for

$$q(\xi^-_{i+1/2})\ ,\ \ q(\xi^+_{i-1/2})\ ,\ \ q\left(\xi_i \pm \frac{\Delta\xi}{2}\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}\right)\ ,\ \ q\left(\xi_i \pm \frac{\Delta\xi}{2}\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}\right), \quad (71)$$

which are used for the first sweep (first two terms corresponding to the two boundaries) and the second sweep (the last two terms corresponding to the 4 quadrature points). For $r = 3$ we have only three candidate stencil for the reconstruction

$$S_0 = (i, i+1, i+2)\ , \quad S_1 = (i-1, i, i+1)\ , \quad S_2 = (i-2, i-1, i). \quad (72)$$

The corresponding smoothness indicators are given by:

$$\begin{aligned}
\beta_0 &= \frac{13}{12}\left(q_i - 2q_{i+1} + q_{i+2}\right)^2 + \frac{1}{4}\left(3q_i - 4q_{i+1} + q_{i+2}\right)^2, \\
\beta_1 &= \frac{13}{12}\left(q_{i-1} - 2q_i + q_{i+1}\right)^2 + \frac{1}{4}\left(q_{i-1} - q_{i+1}\right)^2, \\
\beta_2 &= \frac{13}{12}\left(q_{i-2} - 2q_{i-1} + q_i\right)^2 + \frac{1}{4}\left(q_{i-2} - 4q_{i-1} + 3q_i\right)^2.
\end{aligned}$$

The optimal weights d_m for the left boundary extrapolated value $q^-_{i+1/2}$ at $x_{i+1/2}$ are

$$d_0 = \frac{3}{10}\ , \quad d_1 = \frac{3}{5}\ , \quad d_2 = \frac{1}{10} \quad (73)$$

and $q^-_{i+1/2}$ is given by

$$q^-_{i+1/2} = \frac{1}{6}\omega_0\left(-q_{i+2} + 5q_{i+1} + 2q_i\right) + \frac{1}{6}\omega_1\left(-q_{i-1} + 5q_i + 2q_{i+1}\right) + \frac{1}{6}\omega_2\left(2q_{i-2} - 7q_{i-1} + 11q_i\right). \quad (74)$$

The optimal weights d_m for the right boundary extrapolated value $q^+_{i-1/2}$ at $x_{i-1/2}$ are

$$d_0 = \frac{1}{10}\,, \quad d_1 = \frac{3}{5}\,, \quad d_2 = \frac{3}{10} \tag{75}$$

and $q^+_{i-1/2}$ is given by

$$q^+_{i-1/2} = \frac{1}{6}\omega_0\left(2q_{i+2} - 7q_{i+1} + 11q_i\right) + \frac{1}{6}\omega_1\left(-q_{i+1} + 5q_i + 2q_{i-1}\right) + \frac{1}{6}\omega_2\left(-q_{i-2} + 5q_{i-1} + 2q_i\right). \tag{76}$$

For the first Gaussian quadrature point $\xi^q_1 = \xi_i - \frac{\Delta\xi}{2}\sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}$, the optimal weights are:

$$\begin{aligned}
d_0 &= \frac{269\sqrt{42}\sqrt{2\sqrt{30}+15}}{50428} - \frac{1751\sqrt{35}\sqrt{2\sqrt{30}+15}}{504280} - \frac{411\sqrt{30}}{100856} + \frac{21855}{100856},\\
d_1 &= \frac{411\sqrt{30}}{50428} + \frac{28573}{50428},\\
d_2 &= \frac{1751\sqrt{35}\sqrt{2\sqrt{30}+15}}{504280} - \frac{269\sqrt{42}\sqrt{2\sqrt{30}+15}}{50428} - \frac{411\sqrt{30}}{100856} + \frac{21855}{100856}.
\end{aligned} \tag{77}$$

The reconstructed value can be computed from the three polynomials associated to each stencil:

$$p_0(\xi^q_1) = \left(\frac{\sqrt{5}\sqrt{6}}{140} + \frac{3\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{85}{84}\right) q_i + \left(-\frac{\sqrt{5}\sqrt{6}}{70} - \sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}} - \frac{1}{42}\right) q_{i+1} + \left(\frac{\sqrt{5}\sqrt{6}}{140} + \frac{\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{1}{84}\right) q_{i+2},$$

$$p_1(\xi^q_1) = \left(\frac{\sqrt{5}\sqrt{6}}{140} + \frac{\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{1}{84}\right) q_{i-1} + \left(\frac{41}{42} - \frac{\sqrt{5}\sqrt{6}}{70}\right) q_i + \left(\frac{\sqrt{5}\sqrt{6}}{140} - \frac{\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{1}{84}\right) q_{i+1},$$

$$p_2(\xi^q_1) = \left(\frac{\sqrt{5}\sqrt{6}}{140} - \frac{\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{1}{84}\right) q_{i-2} + \left(\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}} - \frac{\sqrt{5}\sqrt{6}}{70} - \frac{1}{42}\right) q_{i-1} + \left(\frac{\sqrt{5}\sqrt{6}}{140} - \frac{3\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{85}{84}\right) q_i.$$

For the second Gaussian quadrature point $\xi^q_2 = \xi_i - \frac{\Delta\xi}{2}\sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}$, the optimal weights are:

$$\begin{aligned}
d_0 &= \frac{411\sqrt{30}}{100856} - \frac{269\sqrt{42}\sqrt{15-2\sqrt{30}}}{50428} - \frac{1751\sqrt{35}\sqrt{15-2\sqrt{30}}}{504280} + \frac{21855}{100856},\\
d_1 &= \frac{28573}{50428} - \frac{411\sqrt{30}}{50428},\\
d_2 &= \frac{1751\sqrt{35}\sqrt{15-2\sqrt{30}}}{504280} + \frac{269\sqrt{42}\sqrt{15-2\sqrt{30}}}{50428} + \frac{411\sqrt{30}}{100856} + \frac{21855}{100856}.
\end{aligned} \tag{78}$$

The reconstructed value can be then computed from the three polynomials associated to the three stencil:

$$p_0(\xi_2^q) = \left(\frac{3\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140} + \frac{85}{84}\right) q_i + \left(\frac{\sqrt{5}\sqrt{6}}{70} - \sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}} - \frac{1}{42}\right) q_{i+1} + \left(\frac{\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140} + \frac{1}{84}\right) q_{i+2},$$

$$p_1(\xi_2^q) = \left(\frac{\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140} + \frac{1}{84}\right) q_{i-1} + \left(\frac{\sqrt{5}\sqrt{6}}{70} + \frac{41}{42}\right) q_i + \left(\frac{1}{84} - \frac{\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140}\right) q_{i+1},$$

$$p_2(\xi_2^q) = \left(\frac{1}{84} - \frac{\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140}\right) q_{i-2} + \left(\frac{\sqrt{5}\sqrt{6}}{70} + \sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}} - \frac{1}{42}\right) q_{i-1} + \left(\frac{85}{84} - \frac{3\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140}\right) q_i.$$

For the third Gaussian quadrature point $\xi_3^q = \xi_i + \frac{\Delta\xi}{2}\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$, the optimal weights are:

$$\begin{aligned} d_0 &= \frac{1751\sqrt{35}\sqrt{15-2\sqrt{30}}}{504280} + \frac{269\sqrt{42}\sqrt{15-2\sqrt{30}}}{50428} + \frac{411\sqrt{30}}{100856} + \frac{21855}{100856}, \\ d_1 &= \frac{28573}{50428} - \frac{411\sqrt{30}}{50428}, \\ d_2 &= \frac{411\sqrt{30}}{100856} - \frac{269\sqrt{42}\sqrt{15-2\sqrt{30}}}{50428} - \frac{1751\sqrt{35}\sqrt{15-2\sqrt{30}}}{504280} + \frac{21855}{100856}. \end{aligned} \tag{79}$$

The reconstructed value can be then computed from the three polynomials associated to the three stencil:

$$p_0(\xi_3^q) = \left(\frac{85}{84} - \frac{3\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140}\right) q_i + \left(\frac{\sqrt{5}\sqrt{6}}{70} + \sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}} - \frac{1}{42}\right) q_{i+1} + \left(\frac{1}{84} - \frac{\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140}\right) q_{i+2},$$

$$p_1(\xi_3^q) = \left(\frac{1}{84} - \frac{\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140}\right) q_{i-1} + \left(\frac{\sqrt{5}\sqrt{6}}{70} + \frac{41}{42}\right) q_i + \left(\frac{\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140} + \frac{1}{84}\right) q_{i+1},$$

$$p_2(\xi_3^q) = \left(\frac{\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140} + \frac{1}{84}\right) q_{i-2} + \left(\frac{\sqrt{5}\sqrt{6}}{70} - \sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}} - \frac{1}{42}\right) q_{i-1} + \left(\frac{3\sqrt{\frac{3}{7}-\frac{2\sqrt{5}\sqrt{6}}{35}}}{4} - \frac{\sqrt{5}\sqrt{6}}{140} + \frac{85}{84}\right) q_i.$$

For the fourth Gaussian quadrature point $\xi_4^q = \xi_i + \frac{\Delta\xi}{2}\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$, the optimal weights are:

$$\begin{aligned} d_0 &= \frac{1751\sqrt{35}\sqrt{2\sqrt{30}+15}}{504280} - \frac{269\sqrt{42}\sqrt{2\sqrt{30}+15}}{50428} - \frac{411\sqrt{30}}{100856} + \frac{21855}{100856}, \\ d_1 &= \frac{411\sqrt{30}}{50428} + \frac{28573}{50428}, \\ d_2 &= \frac{269\sqrt{42}\sqrt{2\sqrt{30}+15}}{50428} - \frac{1751\sqrt{35}\sqrt{2\sqrt{30}+15}}{504280} - \frac{411\sqrt{30}}{100856} + \frac{21855}{100856}. \end{aligned} \tag{80}$$

The reconstructed value can be then computed from the three polynomials associated to the three stencil:

$$p_0(\xi_4^q) = \left(\frac{\sqrt{5}\sqrt{6}}{140} - \frac{3\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{85}{84}\right) q_i + \left(\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}} - \frac{\sqrt{5}\sqrt{6}}{70} - \frac{1}{42}\right) q_{i+1} + \left(\frac{\sqrt{5}\sqrt{6}}{140} - \frac{\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{1}{84}\right) q_{i+2},$$

$$p_1(\xi_4^q) = \left(\frac{\sqrt{5}\sqrt{6}}{140} - \frac{\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{1}{84}\right) q_{i-1} + \left(\frac{41}{42} - \frac{\sqrt{5}\sqrt{6}}{70}\right) q_i + \left(\frac{\sqrt{5}\sqrt{6}}{140} + \frac{\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{1}{84}\right) q_{i+1},$$

$$p_2(\xi_4^q) = \left(\frac{\sqrt{5}\sqrt{6}}{140} + \frac{\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{1}{84}\right) q_{i-2} + \left(-\frac{\sqrt{5}\sqrt{6}}{70} - \sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}} - \frac{1}{42}\right) q_{i-1} + \left(\frac{\sqrt{5}\sqrt{6}}{140} + \frac{3\sqrt{\frac{2\sqrt{5}\sqrt{6}}{35}+\frac{3}{7}}}{4} + \frac{85}{84}\right) q_i.$$

For all Gaussian quadrature points the solution in ξ can be easily built by assembling the three polynomials with Eq. (2.30).

A.5 Positive reconstruction of water height at interfaces

From the cell averages a high order WENO reconstruction is performed on the fluxes to have $q_{i+1/2}^{L,R}$ and $K_{i+1/2}^{L,R}$. Equipped with $q_{i+1/2}^{L,R}, K_{i+1/2}^{L,R}, \mathcal{R}_{i+1/2}^{L,R}$, the point values $h_{i+1/2}^{L,R}$ can be obtained by solving the nonlinear equation coming from the definition of the global variable K in (7.1):

$$K_{i+1/2}^L = \frac{\left(q_{i+1/2}^L\right)^2}{h_{i+1/2}^L} + \frac{g}{2}\left(h_{i+1/2}^L\right)^2 + \mathcal{R}_{i+1/2}^L, \quad K_{i+1/2}^R = \frac{\left(q_{i+1/2}^R\right)^2}{h_{i+1/2}^R} + \frac{g}{2}\left(h_{i+1/2}^R\right)^2 + \mathcal{R}_{i+1/2}^R \tag{81}$$

Let us solve the depressed cubic equation (81) for $h_{i+1/2}^L$ (the solution for $h_{i+1/2}^R$ is obtained with the same procedure). First of all, it can be noticed that (81) does not have any positive solution unless the determinant is greater than zero, meaning that

$$\left(q_{i+1/2}^L\right)^4 < \frac{8\left(K_{i+1/2}^L - \mathcal{R}_{i+1/2}^L\right)^3}{27\,g}. \tag{82}$$

If (82) is not satisfied we reconstruct $\eta_{i+1/2}^L$ and then compute $h_{i+1/2}^L$ given that

$$h_{i+1/2}^L = \eta_{i+1/2}^L - b_{i+1/2}^L. \tag{83}$$

If (82) is satisfied, then we have to deal with two possibilities. First, if $q^L_{i+1/2} = 0$, we obtain the unique positive solution

$$h^L_{i+1/2} = \sqrt{\frac{2\left(K^L_{i+1/2} - \mathcal{R}^L_{i+1/2}\right)}{g}},$$

while if $q^L_{i+1/2} \neq 0$, we solve Eq. (81) for $h^L_{i+1/2}$ and obtain the following three solutions:

$$h^L_{i+1/2} = 2\sqrt{P}\cos\left(\frac{1}{3}\left[\Theta + 2\pi k\right]\right), \quad k = 0, 1, 2, \tag{84}$$

where

$$P := \frac{2\left(K^L_{i+1/2} - R^L_{i+1/2}\right)}{3g} \quad \text{and} \quad \Theta := \arccos\left(-\frac{\left(q^L_{i+1/2}\right)^2}{g\,P^{3/2}}\right). \tag{85}$$

It can be shown that one of these roots is negative, whilst the other two roots, corresponding to the subcritical and supercritical cases, are positive. We choose the one closer to the corresponding value of $h^L_{i+1/2}$ given in (83).

Bibliography

[1] John Burkardt's home page. Last accessed: January 19, 2021. 90

[2] R. Abgrall, P. Bacigaluppi, and S. Tokareva. A high-order nonconservative approach for hyperbolic equations in fluid dynamics. *Computers & Fluids*, 169:10–22, 2018. Recent progress in nonlinear numerical methods for time-dependent flow & transport problems. 124

[3] R. Abgrall and M. Ricchiuto. Hyperbolic balance laws: residual distribution, local and global fluxes. In *Current Trends in Fluid Dynamics: Modelling and Simulation*. Springer, 2022. to appear, [pdf]. 124

[4] R. Abgrall and Mario Ricchiuto. *High-Order Methods for CFD*, pages 1–54. John Wiley & Sons, Ltd, 2017. viii, 2, 117, 124

[5] Remi Abgrall. How to prevent pressure oscillations in multicomponent flow calculations: A quasi conservative approach. *Journal of Computational Physics*, 125(1):150–160, 1996. 124

[6] Rémi Abgrall. Residual distribution schemes: current status and future trends. *Computers & Fluids*, 35(7):641–669, 2006. vii, 1

[7] Rémi Abgrall. Residual distribution schemes: current status and future trends. *Computers & Fluids*, 35(7):641–669, 2006. 43

[8] Rémi Abgrall. High order schemes for hyperbolic problems using globally continuous approximation and avoiding mass matrices. *Journal of Scientific Computing*, 73(2):461–494, 2017. 43, 48, 50, 51, 53

[9] Rémi Abgrall. A general framework to construct schemes satisfying additional conservation relations. application to entropy conservative and entropy dissipative schemes. *Journal of Computational Physics*, 372:640–666, 2018. 43, 205

[10] Rémi Abgrall. Some remarks about conservation for residual distribution schemes. *Computational Methods in Applied Mathematics*, 18(3):327–351, 2018. 43

[11] Rémi Abgrall, Paola Bacigaluppi, and Svetlana Tokareva. High-order residual distribution scheme for the time-dependent euler equations of fluid dynamics. *Computers & Mathematics with Applications*, 78(2):274–297, 2019. 43

[12] Remi Abgrall, Héloïse Beaugendre, and Cécile Dobrzynski. An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques. *Journal of Computational Physics*, 257:83–101, 2014. 3

[13] Remi Abgrall, Adam Larat, and Mario Ricchiuto. Construction of very high order residual distribution schemes for steady inviscid flow problems on hybrid unstructured meshes. *Journal of Computational Physics*, 230(11):4103–4136, 2011. 46

[14] Rémi Abgrall, Elise Le Mélédo, Philipp Öffner, and Davide Torlo. Relaxation deferred correction methods and their applications to residual distribution schemes. *arXiv preprint arXiv:2106.05005*, 2021. 48, 53

[15] Rémi Abgrall, Jan Nordström, Philipp Öffner, and Svetlana Tokareva. Analysis of the SBP-SAT stabilization for finite element methods part II: entropy stability. *Communications on Applied Mathematics and Computation*, pages 1–23, 2021. 205

[16] Rémi Abgrall, Philipp Öffner, and Hendrik Ranocha. Reinterpretation and extension of entropy correction terms for residual distribution and discontinuous galerkin schemes: Application to structure preserving discretization. *Journal of Computational Physics*, page 110955, 2022. viii, 2

[17] Remi Abgrall and Mario Ricchiuto. High order methods for cfd, 2017. 45

[18] Rémi Abgrall and Davide Torlo. High order asymptotic preserving deferred correction implicit-explicit schemes for kinetic models. *SIAM Journal on Scientific Computing*, 42(3):B816–B845, 2020. 48

[19] Frédéric Alauzet. A changing-topology moving mesh technique for large displacements. *Engineering with Computers*, 30(2):175–200, 2014. 3

[20] L. Arpaia, M. Ricchiuto, and R. Abgrall. An ALE formulation for explicit Runge-Kutta residual distribution. *Journal of Scientific Computing*, pages 1–46, 2014. 91

[21] Luca Arpaia and Mario Ricchiuto. r- adaptation for shallow water flows: conservation, well balancedness, efficiency. *Computers & Fluids*, 160:175–203, 2018. 10

[22] A. Assonitis, R. Paciorri, and A. Bonfiglioli. Numerical simulation of shock boundary layer interaction using shock fitting technique. *Lecture Notes in Mechanical Engineering*, pages 124–134, 2020. 89

[23] Alessia Assonitis, Mirco Ciallella, Renato Paciorri, Mario Ricchiuto, and Aldo Bonfiglioli. A new shock-fitting technique for 2-d structured grids. In *AIAA Scitech 2022 Forum*, page 2008, 2022. 205

[24] Alessia Assonitis, Mirco Ciallella, Renato Paciorri, Mario Ricchiuto, and Aldo Bonfiglioli. A shock-fitting technique for 2d/3d flows with interactions using structured grids. In *AIAA AVIATION 2022 Forum*, page 4123, 2022. 151, 205

[25] Alessia Assonitis, Renato Paciorri, Mirco Ciallella, Mario Ricchiuto, and Aldo Bonfiglioli. Extrapolated shock fitting for two-dimensional flows on structured grids. *AIAA Journal*, 0(0):1–12, 0. ix, 11, 13, 205

[26] Nabil M Atallah, Claudio Canuto, and Guglielmo Scovazzi. The high-order shifted boundary method and its analysis. *Computer Methods in Applied Mechanics and Engineering*, 394:114885, 2022. 3

[27] Emmanuel Audusse, François Bouchut, Marie-Odile Bristeau, Rupert Klein, and Benoıt Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM Journal on Scientific Computing*, 25(6):2050–2065, 2004. 8, 10

[28] Dinshaw S Balsara and Chi-Wang Shu. Monotonicity preserving weighted essentially non-oscillatory schemes with increasingly high order of accuracy. *Journal of Computational Physics*, 160(2):405–452, 2000. 39

[29] Nicolas Barral, Géraldine Olivier, and Frédéric Alauzet. Time-accurate anisotropic mesh adaptation for three-dimensional time-dependent problems with body-fitted moving geometries. *Journal of Computational Physics*, 331:157–187, 2017. 3

[30] Francesco Bassi and S Rebay. Accurate 2d euler computations by means of a high order discontinuous finite element method. In *Fourteenth International Conference on Numerical Methods in Fluid Dynamics*, pages 234–240. Springer, 1995. 67

[31] Francesco Bassi and Stefano Rebay. High-order accurate discontinuous finite element solution of the 2d euler equations. *Journal of computational physics*, 138(2):251–285, 1997. viii, 2, 35, 41, 67

[32] Andrea D Beck, Jonas Zeifang, Anna Schwarz, and David G Flad. A neural network based shock detection and localization approach for discontinuous galerkin methods. *Journal of Computational Physics*, 423:109824, 2020. 116

[33] Sylvie Benzoni-Gavage and Denis Serre. *Multi-dimensional hyperbolic partial differential equations: First-order Systems and Applications*. OUP Oxford, 2006. 26

[34] Jonas P Berberich, Praveen Chandrashekar, and Christian Klingenberg. High order well-balanced finite volume methods for multi-dimensional systems of hyperbolic balance laws. *Computers & Fluids*, 219:104858, 2021. viii, 2, 155, 156

[35] Marsha Berger and Andrew Giuliani. A state redistribution algorithm for finite volume schemes on cut cell meshes. *Journal of Computational Physics*, 428:109820, 2021. 3, 5

[36] Marsha Berger and Christiane Helzel. A simplified h-box method for embedded boundary grids. *SIAM Journal on Scientific Computing*, 34(2):A861–A888, 2012. 5

[37] Michel Bergmann, Michele Giuliano Carlino, and Angelo Iollo. Second order ader scheme for unsteady advection-diffusion on moving overset grids with a compact transmission condition. *SIAM Journal on Scientific Computing*, 44(1):A524–A553, 2022. 3

[38] Christophe Berthon, Solène Bulteau, Françoise Foucher, Meissa M'Baye, and Victor Michel-Dansac. A very easy high-order well-balanced reconstruction for hyperbolic systems with source terms. 2021. 8, 206

[39] Christophe Berthon and Françoise Foucher. Efficient well-balanced hydrostatic upwind schemes for shallow-water equations. *Journal of Computational Physics*, 231(15):4993–5015, 2012. 8

[40] Daniele Boffi and Lucia Gastaldi. A finite element approach for the immersed boundary method. *Computers & Structures*, 81(8-11):491–501, 2003. 3

[41] Andreas Bollermann, Guoxian Chen, Alexander Kurganov, and Sebastian Noelle. A well-balanced reconstruction of wet/dry fronts for the shallow water equations. *Journal of Scientific Computing*, 56(2):267–290, 2013. 10

[42] A. Bonfiglioli. Fluctuation splitting schemes for the compressible and incompressible Euler and Navier-Stokes equations. *Int. J. Comput. Fluid Dyn.*, 14(1):21–39, 2000. 90

[43] A. Bonfiglioli, M. Grottadaurea, R. Paciorri, and F. Sabetta. An unstructured, three-dimensional, shock-fitting solver for hypersonic flows. *Computers & Fluids*, 73(0):162–174, 2013. 89

[44] A. Bonfiglioli and R. Paciorri. Convergence analysis of shock-capturing and shock-fitting solutions on unstructured grids. *AIAA J.*, 52(7):1404–1416, 2014. 89

[45] A Bonfiglioli, R Paciorri, and L Campoli. Unsteady shock-fitting for unstructured grids. *International Journal for Numerical Methods in Fluids*, 81(4):245–261, 2016. 7, 122

[46] A. Bonfiglioli, R. Paciorri, and L. Campoli. Unsteady shock-fitting for unstructured grids. *International Journal for Numerical Methods in Fluids*, 81(4):245–261, 2016. 89, 108

[47] Aldo Bonfiglioli. Fluctuation splitting schemes for the compressible and incompressible Euler and Navier-Stokes equations. *International Journal of Computational Fluid Dynamics*, 14(1):21–39, 2000. 117

[48] Aldo Bonfiglioli, Marco Grottadaurea, Renato Paciorri, and Filippo Sabetta. An unstructured, three-dimensional, shock-fitting solver for hypersonic flows. *Computers & Fluids*, 73:162–174, 2013. 7

[49] Aldo Bonfiglioli and Renato Paciorri. A mass-matrix formulation of unsteady fluctuation splitting schemes consistent with Roe's parameter vector. *International Journal of Computational Fluid Dynamics*, 27(4-5):210–227, 2013. 90

[50] Aldo Bonfiglioli and Renato Paciorri. A mass-matrix formulation of unsteady fluctuation splitting schemes consistent with Roe's parameter vector. *International Journal of Computational Fluid Dynamics*, 27(4-5):210–227, 2013. 117

[51] Aldo Bonfiglioli and Renato Paciorri. Convergence analysis of shock-capturing and shock-fitting solutions on unstructured grids. *AIAA journal*, 52(7):1404–1416, 2014. 127

[52] Aldo Bonfiglioli, Renato Paciorri, Lorenzo Campoli, Valentina De Amicis, and Marcello Onofri. Development of an unsteady shock-fitting technique for unstructured grids. In Gabi Ben-Dor, Oren Sadot, and Ozer Igra, editors, *30th International Symposium on Shock Waves 2*, pages 1501–1504, Cham, 2017. Springer International Publishing. 109

[53] Walter Boscheri and Michael Dumbser. High order accurate direct arbitrary-lagrangian-eulerian ader-weno finite volume schemes on moving curvilinear unstructured meshes. *Computers & Fluids*, 136:48–66, 2016. 5

[54] Walter Boscheri and Michael Dumbser. Arbitrary-lagrangian–eulerian discontinuous galerkin schemes with a posteriori subcell finite volume limiting on moving unstructured meshes. *Journal of Computational Physics*, 346:449–479, 2017. 71

[55] Walter Boscheri and Raphaël Loubère. High order accurate direct arbitrary-lagrangian-eulerian ader-mood finite volume schemes for non-conservative hyperbolic systems with stiff source terms. *Communications in Computational Physics*, 21(1):271–312, 2017. 48, 70

[56] Walter Boscheri, Raphaël Loubere, and Michael Dumbser. Direct arbitrary-lagrangian–eulerian ader-mood finite volume schemes for multidimensional hyperbolic conservation laws. *Journal of Computational Physics*, 292:56–87, 2015. 48, 70

[57] N Botta, R Klein, S Langenberg, and S Lützenkirchen. Well balanced finite volume methods for nearly hydrostatic flows. *Journal of Computational Physics*, 196(2):539–565, 2004. 8

[58] SW Bova and GF Carey. An entropy variable formulation and applications for the two-dimensional shallow water equations. *International journal for numerical methods in fluids*, 23(1):29–46, 1996. 30

[59] Alexander N Brooks and Thomas JR Hughes. Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer methods in applied mechanics and engineering*, 32(1-3):199–259, 1982. vii, 1

[60] Jens Brouwer, Julius Reiss, and Jörn Sesterhenn. Conservative time integrators of arbitrary order for skew-symmetric finite-difference discretizations of compressible flow. *Computers & Fluids*, 100:1–12, 2014. 9

[61] Hans Burchard, Eric Deleersnijder, and Andreas Meister. A high-order conservative Patankar-type discretisation for stiff systems of production–destruction equations. *Applied Numerical Mathematics*, 47(1):1–30, 2003. 157, 158, 162

[62] Hans Burchard, Eric Deleersnijder, and Andreas Meister. Application of modified patankar schemes to stiff biogeochemical models for the water column. *Ocean Dynamics*, 55(3):326–337, 2005. 9

[63] Erik Burman. Ghost penalty. *Comptes Rendus Mathematique*, 348(21-22):1217–1220, 2010. 3

[64] Erik Burman and Peter Hansbo. Fictitious domain methods using cut elements: Iii. a stabilized Nitsche method for Stokes' problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(3):859–874, 2014. 3

[65] Saray Busto, Simone Chiocchetti, Michael Dumbser, Elena Gaburro, and Ilya Peshkov. High order ader schemes for continuum mechanics. *Frontiers in Physics*, 8:32, 2020. 48

[66] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016. 48

[67] M. Sergio Campobasso and Mohammad H. Baba-Ahmadi. Ad-hoc boundary conditions for cfd analyses of turbomachinery problems with strong flow gradients at farfield boundaries. *Journal of Turbomachinery*, 133(4), 04 2011. 127

[68] Lorenzo Campoli, Alessia Assonitis, Mirco Ciallella, Renato Paciorri, Aldo Bonfiglioli, and Mario Ricchiuto. Undifi-2d: an unstructured discontinuity fitting code for 2d grids. *Computer Physics Communications*, 271:108202, 2022. ix, 11, 12, 89, 111, 204

[69] Lorenzo Campoli, Pierrick Quemar, Aldo Bonfiglioli, and Mario Ricchiuto. Shock-fitting and predictor-corrector explicit ale residual distribution. In *Shock Fitting: Classical Tech-*

niques, Recent Developments, and Memoirs of Gino Moretti, pages 113–129. Springer, 2017. 7, 89, 108, 122

[70] Yangyang Cao, Alexander Kurganov, Yongle Liu, and Ruixiao Xin. Flux globalization based well-balanced path-conservative central-upwind schemes for shallow water models. *Journal of Scientific Computing*, 92(2):1–31, 2022. 10

[71] Yangyang Cao, Alexander Kurganov, Yongle Liu, and Vladimir Zeitlin. Flux globalization based well-balanced path-conservative central-upwind scheme for two-layer thermal rotating shallow water equations. *Available at SSRN 4089079*. 10

[72] Tiffanie Carlier, Léo Nouveau, Heloise Beaugendre, Mathieu Colin, and Mario Ricchiuto. An enriched shifted boundary method to account for moving fronts. 2022. 3

[73] Jay Casper and HL Atkins. A finite-volume high-order ENO scheme for two-dimensional hyperbolic systems. *Journal of Computational Physics*, 106(1):62–76, 1993. 37

[74] Cristóbal E Castro and Eleuterio F Toro. Solvers for the high-order riemann problem for hyperbolic balance laws. *Journal of Computational Physics*, 227(4):2481–2513, 2008. 54

[75] Manuel Castro, José Gallardo, and Carlos Parés. High order finite volume schemes based on reconstruction of states for solving hyperbolic systems with nonconservative products. applications to shallow-water systems. *Mathematics of computation*, 75(255):1103–1134, 2006. 8

[76] Manuel J Castro, Alberto Pardo Milanés, and Carlos Parés. Well-balanced numerical schemes based on a generalized hydrostatic reconstruction technique. *Mathematical Models and Methods in Applied Sciences*, 17(12):2055–2113, 2007. 8

[77] William M Chan. Overset grid technology development at nasa ames research center. *Computers & Fluids*, 38(3):496–503, 2009. 3

[78] S Chang, X Bai, D Zou, Z Chen, and J Liu. An adaptive discontinuity fitting technique on unstructured dynamic grids. *Shock Waves*, pages 1–13, 2019. 138, 140, 145

[79] Tianheng Chen and Chi-Wang Shu. Review of entropy stable discontinuous Galerkin methods for systems of conservation laws on unstructured simplex meshes. *CSIAM Transactions on Applied Mathematics*, 1:1–52, 2020. viii, 2, 205

[80] Yuanzhen Cheng, Alina Chertock, Michael Herty, Alexander Kurganov, and Tong Wu. A new approach for designing moving-water equilibria preserving schemes for the shallow water equations. *Journal of Scientific Computing*, 80(1):538–554, 2019. 10, 181, 182, 184

[81] I.-L Chern, J Glimm, O Mcbryan, B Plohr, and S Yaniv. Front tracking for gas dynamics. *Journal of Computational Physics*, 62(1):83 – 110, 1986. 7, 109

[82] I-Liang Chern and Phillip Colella. A conservative front tracking method for hyperbolic conservation laws. *LLNL Rep. No. UCRL-97200, Lawrence Livermore National Laboratory*, 2:83–110, 1987. 8

[83] Alina Chertock, Shumo Cui, Alexander Kurganov, Şeyma Nur Özcan, and Eitan Tadmor. Well-balanced schemes for the euler equations with gravitation: Conservative formulation using global fluxes. *Journal of Computational Physics*, 358:36–52, 2018. 10

[84] Alina Chertock, Shumo Cui, Alexander Kurganov, and Tong Wu. Steady state and sign preserving semi-implicit Runge–Kutta methods for ODEs with stiff damping term. *SIAM Journal on Numerical Analysis*, 53(4):2008–2029, 2015. 158

[85] Alina Chertock, Alexander Kurganov, Xin Liu, Yongle Liu, and Tong Wu. Well-balancing via flux globalization: Applications to shallow water equations with wet/dry fronts. *Journal of Scientific Computing*, 90(1):1–21, 2022. 10

[86] M. Ciallella, L. Micalizzi, P. Öffner, and D. Torlo. An arbitrary high order and positivity preserving method for the shallow water equations. *Computers and Fluids*, page 105630, 2022. x, 8, 11, 13, 48, 155

[87] Mirco Ciallella, Elena Gaburro, Marco Lorini, and Mario Ricchiuto. Shifted boundary polynomial corrections for compressible flows: high order on curved domains using linear meshes. *Submitted to Applied Mathematics and Computation*, 2022. viii, 11, 12, 203

[88] Mirco Ciallella, Lorenzo Micalizzi, Philipp Öffner, and Davide Torlo. Modified Patankar Deferred Correction WENO Code for Shallow Water Equations. `https://github.com/accdavlo/sw-mpdec.git`, October 2021. 40, 164, 168, 211

[89] Mirco Ciallella, Mario Ricchiuto, Renato Paciorri, and Aldo Bonfiglioli. Extrapolated shock tracking: bridging shock-fitting and embedded boundary methods. *Journal of Computational Physics*, 412:109440, 2020. ix, 11, 12, 97, 111, 113, 118, 123, 124, 127, 137, 149, 204

[90] Mirco Ciallella, Mario Ricchiuto, Renato Paciorri, and Aldo Bonfiglioli. Extrapolated discontinuity tracking for complex 2d shock interactions. *Computer Methods in Applied Mechanics and Engineering*, 391:114543, 2022. ix, 11, 13, 89, 111, 137, 149, 204

[91] Mirco Ciallella, Davide Torlo, and Mario Ricchiuto. Arbitrary high order weno finite volume scheme with flux globalization for moving equilibria preservation. *arXiv preprint arXiv:2205.13315*, 2022. x, 12, 13, 181

[92] Stéphane Clain, Steven Diot, and Raphaël Loubère. 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. 70, 203

[93] Bernardo Cockburn, Suchung Hou, and Chi-Wang Shu. 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. 41

[94] Bernardo Cockburn, George E Karniadakis, and Chi-Wang Shu. *Discontinuous Galerkin methods: theory, computation and applications*, volume 11. Springer Science & Business Media, 2012. vii, 1

[95] Bernardo Cockburn and Chi-Wang Shu. Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws. ii. general framework. *Mathematics of computation*, 52(186):411–435, 1989. 41

[96] Bernardo Cockburn and Chi-Wang Shu. The runge-kutta local projection-discontinuous-galerkin finite element method for scalar conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis*, 25(3):337–361, 1991. 41

[97] Bernardo Cockburn and Chi-Wang Shu. The runge–kutta discontinuous galerkin method for conservation laws v: multidimensional systems. *Journal of Computational Physics*, 141(2):199–224, 1998. vii, 1, 41

[98] Bernardo Cockburn and Chi-Wang Shu. Runge–kutta discontinuous galerkin methods for convection-dominated problems. *Journal of scientific computing*, 16(3):173–261, 2001. vii, 1

[99] Earl A Coddington and Norman Levinson. *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955. 19

[100] Phillip Colella and Paul R Woodward. The piecewise parabolic method (ppm) for gas-dynamical simulations. *Journal of Computational Physics*, 54(1):174–201, 1984. 35

[101] Andrew Corrigan, Andrew D. Kercher, and David A. Kessler. A moving discontinuous Galerkin finite element method for flows with interfaces. *International Journal for Numerical Methods in Fluids*, 89(9):362–406, 2019. 151

[102] Ricardo Costa, Stéphane Clain, Raphaël Loubère, and Gaspar J Machado. Very high-order accurate finite volume scheme on curved boundaries for the two-dimensional steady-state convection–diffusion equation with dirichlet condition. *Applied Mathematical Modelling*, 54:752–767, 2018. 6, 69, 73

[103] Ricardo Costa, Stéphane Clain, Gaspar J Machado, and João M Nóbrega. Very high-order accurate finite volume scheme for the steady-state incompressible navier–stokes equations with polygonal meshes on arbitrary curved boundaries. *Computer Methods in Applied Mechanics and Engineering*, 396:115064, 2022. 6

[104] Ricardo Costa, Joao M Nobrega, Stephane Clain, and Gaspar J Machado. Very high-order accurate polygonal mesh finite volume scheme for conjugate heat transfer problems with curved interfaces and imperfect contacts. *Computer Methods in Applied Mechanics and Engineering*, 357:112560, 2019. 6

[105] Ricardo Costa, João M Nóbrega, Stéphane Clain, Gaspar J Machado, and Raphaël Loubère. Very high-order accurate finite volume scheme for the convection-diffusion equation with general boundary conditions on arbitrary curved boundaries. *International Journal for Numerical Methods in Engineering*, 117(2):188–220, 2019. 6

[106] Á. Csík, M. Ricchiuto, and H. Deconinck. A conservative formulation of the multidimensional upwind residual distribution schemes for general nonlinear conservation laws. *Journal of Computational Physics*, 179(1):286–312, 2002. 124

[107] Luke M. D'Aquila, Brian T. Helenbrook, and Alireza Mazaheri. A novel stabilization method for high-order shock fitting with finite element methods. *Journal of Computational Physics*, page 110096, 2021. 89

[108] Marco D de Tullio, Pietro De Palma, Gianluca Iaccarino, Giuseppe Pascazio, and Michele Napolitano. An immersed boundary method for compressible flows using local grid refinement. *Journal of Computational Physics*, 225(2):2098–2117, 2007. 3

[109] H. Deconinck, P.L. Roe, and R. Struijs. A multidimensional generalization of Roe's flux difference splitter for the euler equations. *Computers & Fluids*, 22(2-3):215–222, mar 1993. 118

[110] H Deconinck, R Struijs, G Bourgois, H Paillere, and PL Roe. Multidimensional upwind methods for unstructured grids. *In AGARD*, 1992. 46

[111] Herman Deconinck and Mario Ricchiuto. Residual distribution schemes: foundation and analysis. *Encyclopedia of Computational Mechanics. John Wiley & Sons, Ltd*, 2007. vii, 1, 43

[112] Herman Deconinck and Mario Ricchiuto. *Residual Distribution Schemes: Foundations and Analysis*, pages 1–53. John Wiley & Sons, Ltd, 2017. 117, 124

[113] Olivier Delestre, Carine Lucas, Pierre-Antoine Ksinant, Frédéric Darboux, Christian Laguerre, T-N-Tuoi Vo, Francois James, and Stéphane Cordier. Swashes: a compilation of shallow water analytic solutions for hydraulic and environmental studies. *International Journal for Numerical Methods in Fluids*, 72(3):269–300, 2013. 10, 191, 194, 195

[114] Saikat Dey, Robert M O'Bara, and Mark S Shephard. Towards curvilinear meshing in 3d: the case of quadratic simplices. *Computer-Aided Design*, 33(3):199–209, 2001. 63

[115] MJ Castro Díaz, Juan A López-García, and Carlos Parés. High order exactly well-balanced numerical methods for shallow water systems. *Journal of Computational Physics*, 246:242–264, 2013. 8, 10

[116] Steven Diot, Stéphane Clain, and Raphaël Loubère. Improved detection criteria for the multi-dimensional optimal order detection (mood) on unstructured meshes with very high-order polynomials. *Computers & Fluids*, 64:43–63, 2012. 70, 203

[117] Steven Diot, Raphaël Loubère, and Stephane Clain. The multidimensional optimal order detection method in the three-dimensional case: very high-order finite volume method for hyperbolic systems. *International Journal for Numerical Methods in Fluids*, 73(4):362–392, 2013. 70

[118] Jian Du, Brian Fix, James Glimm, Xicheng Jia, Xiaolin Li, Yuanhua Li, and Lingling Wu. A simple package for front tracking. *Journal of Computational Physics*, 213(2):613–628, 2006. 7

[119] Qiang Du, Roland Glowinski, Michael Hintermüller, and Endre Suli. *Handbook of numerical methods for hyperbolic problems: basic and fundamental issues*. Elsevier, 2016. 31

[120] Michael Dumbser. Arbitrary high order pnpm schemes on unstructured meshes for the compressible Navier–Stokes equations. *Computers & Fluids*, 39(1):60–76, 2010. 35

[121] Michael Dumbser, Dinshaw S Balsara, Eleuterio F Toro, and Claus-Dieter Munz. A unified framework for the construction of one-step finite volume and discontinuous galerkin schemes on unstructured meshes. *Journal of Computational Physics*, 227(18):8209–8253, 2008. 48, 55

[122] Michael Dumbser, Cedric Enaux, and Eleuterio F Toro. Finite volume schemes of very high order of accuracy for stiff hyperbolic balance laws. *Journal of Computational Physics*, 227(8):3971–4001, 2008. 55

[123] Alok Dutt, Leslie Greengard, and Vladimir Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT Numerical Mathematics*, 40(2):241–266, 2000. 50

[124] Bernd Einfeldt, Claus-Dieter Munz, Philip L Roe, and Björn Sjögreen. On godunov-type methods near low densities. *Journal of computational physics*, 92(2):273–295, 1991. vii, 1

[125] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Soc., 2010. 19

[126] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000. vii, 1

[127] Francesco Fambri, Michael Dumbser, and Olindo Zanotti. Space–time adaptive ader-dg schemes for dissipative flows: Compressible navier–stokes and resistive mhd equations. *Computer Physics Communications*, 220:297–318, 2017. 204

[128] Charbel Farhat, Arthur Rallu, and Sriram Shankaran. A higher-order generalized ghost fluid method for the poor for the three-dimensional two-phase flow computation of underwater implosions. *Journal of Computational Physics*, 227(16):7674–7700, 2008. 3

[129] Ronald P Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of computational physics*, 152(2):457–492, 1999. 3

[130] Ronald P Fedkiw, Tariq Aslam, and Shaojie Xu. The ghost fluid method for deflagration and detonation discontinuities. *Journal of Computational Physics*, 154(2):393–427, 1999. 3

[131] Javier Fernández-Fidalgo, Stéphane Clain, Luis Ramírez, Ignasi Colominas, and Xesús Nogueira. Very high-order method on immersed curved domains for finite difference schemes with regular cartesian grids. *Computer Methods in Applied Mechanics and Engineering*, 360:112782, 2020. 6

[132] Camilla Fiorini, Christophe Chalons, and Régis Duvigneau. A modified sensitivity equation method for the euler equations in presence of shocks. *Numerical methods for partial differential equations*, 36(4):839–867, 2020. 205

[133] Travis C Fisher, Mark H Carpenter, Jan Nordström, Nail K Yamaleev, and Charles Swanson. Discretely conservative finite-difference formulations for nonlinear conservation laws in split form: Theory and boundary conditions. *Journal of Computational Physics*, 234:353–375, 2013. 205

[134] Ulrik S Fjordholm, Siddhartha Mishra, and Eitan Tadmor. Well-balanced and energy stable schemes for the shallow water equations with discontinuous topography. *Journal of Computational Physics*, 230(14):5587–5609, 2011. 9

[135] Ulrik Skre Fjordholm, Siddhartha Mishra, and Eitan Tadmor. Arbitrarily high-order accurate entropy stable essentially nonoscillatory schemes for systems of conservation laws. *SIAM Journal on Numerical Analysis*, 50(2):544–573, 2012. 205

[136] M. Fortunato and P.-O. Persson. High-order unstructured curved mesh generation using the winslow equations. *Journal of Computational Physics*, 307:1–14, 2016. 63

[137] Meire Fortunato and Per-Olof Persson. High-order unstructured curved mesh generation using the winslow equations. *Journal of Computational Physics*, 307:1–14, 2016. 5

[138] Leopoldo P Franca, Sergio L Frey, and Thomas JR Hughes. Stabilized finite element methods: I. application to the advective-diffusive model. *Computer Methods in Applied Mechanics and Engineering*, 95(2):253–276, 1992. vii, 1

[139] Elena Gaburro, Walter Boscheri, Simone Chiocchetti, Christian Klingenberg, Volker Springel, and Michael Dumbser. High order direct arbitrary-lagrangian-eulerian schemes on moving voronoi meshes with topology changes. *Journal of Computational Physics*, 407:109167, 2020. 35, 48, 203

[140] Elena Gaburro and Michael Dumbser. A posteriori subcell finite volume limiter for general PNPM schemes: Applications from gasdynamics to relativistic magnetohydrodynamics. *Journal of Scientific Computing*, 86(3):1–41, 2021. 71

[141] José M Gallardo, Carlos Parés, and Manuel Castro. On a well-balanced high-order finite volume scheme for shallow water equations with topography and dry areas. *Journal of Computational Physics*, 227(1):574–601, 2007. 9, 10

[142] A. Gargallo-Peiró, X. Roca, J. Peraire, and J. Sarrate. Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 103(5):342–363, 2015. 63

[143] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. 64

[144] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009. 127, 149

[145] Giorgio Giorgiani, Hervé Guillard, Boniface Nkonga, and Eric Serre. A stabilized powell–sabin finite-element method for the 2d euler equations in supersonic regime. *Computer Methods in Applied Mechanics and Engineering*, 340:216–235, 2018. 119

[146] Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms*, volume 5. Springer Science & Business Media, 2012. vii, 1

[147] Andrew Giuliani. A two-dimensional stabilized discontinuous galerkin method on curvilinear embedded boundary grids. *SIAM Journal on Scientific Computing*, 44(1):A389–A415, 2022. 3, 5

[148] J. Glimm, E. Isaacson, D. Marchesin, and O. McBryan. Front tracking for hyperbolic systems. *Advances in Applied Mathematics*, 2(1):91–119, 1981. 7

[149] James Glimm, John W. Grove, X. L. Li, and N. Zhao. Simple front tracking. In *Nonlinear partial differential equations (Evanston, IL, 1998)*, volume 238 of *Contemp. Math.*, pages 133–149. Amer. Math. Soc., Providence, RI, 1999. 7

[150] James Glimm, John W. Grove, Xiao Lin Li, Keh-ming Shyue, Yanni Zeng, and Qiang Zhang. Three-dimensional front tracking. *SIAM Journal on Scientific Computing*, 19(3):703–727, 1998. 7

[151] James Glimm, John W Grove, Xiaolin L Li, and De Chun Tan. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM Journal on Scientific Computing*, 21(6):2240–2256, 2000. 7

[152] James Glimm, JW Grove, XL Li, W Oh, and DH Sharp. A critical analysis of rayleigh–taylor growth rates. *Journal of Computational Physics*, 169(2):652–677, 2001. 7

[153] James Glimm, Xiao Lin Li, and Yingjie Liu. Conservative front tracking in one space dimension. *Contemporary Mathematics*, 295:253–264, 2002. 8

[154] James Glimm, Xiao Lin Li, Yingjie Liu, and Ning Zhao. Conservative front tracking and level set algorithms. *Proceedings of the National Academy of Sciences*, 98(25):14198–14201, 2001. 8

[155] James Glimm, Xiao Lin Li, and YJ Liu. Conservative front tracking in higher space dimensions. *Transactions of Nanjing University of Aeronautics and Astronautics*, 18:1–15, 2001. 8

[156] Edwige Godlewski and Pierre-Arnaud Raviart. *Hyperbolic systems of conservation laws*. Ellipses, 1991. 26

[157] Sergei Konstantinovich Godunov. A difference scheme for numerical solution of discontinuous solution of hydrodynamic equations. *Math. Sbornik*, 47:271–306, 1959. vii, 1, 32, 35, 39

[158] Sigal Gottlieb, David I Ketcheson, and Chi-Wang Shu. *Strong stability preserving Runge-Kutta and multistep time discretizations*. World Scientific, 2011. 164, 179

[159] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing runge-kutta schemes. *Mathematics of computation*, 67(221):73–85, 1998. 50

[160] Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM review*, 43(1):89–112, 2001. 50

[161] F. Grasso and S. Pirozzoli. Shock-wave-vortex interactions: Shock and vortex deformations, and sound production. *Theoretical and Computational Fluid Dynamics*, 13(6):421–456, 2000. 107

[162] Oliver Gressel. Toward realistic simulations of magneto-thermal winds from weakly-ionized protoplanetary disks. In *Journal of Physics: Conference Series*, volume 837, page 012008. IOP Publishing, 2017. 9

[163] Boyce E Griffith and Charles S Peskin. On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems. *Journal of Computational Physics*, 208(1):75–105, 2005. 3

[164] Hennes Hajduk. Monolithic convex limiting in discontinuous Galerkin discretizations of hyperbolic conservation laws. *Computers & Mathematics with Applications*, 87:120–138, 2021. 205

[165] Anita Hansbo and Peter Hansbo. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Computer methods in applied mechanics and engineering*, 191(47-48):5537–5552, 2002. 3

[166] Ami Harten and James M Hyman. Self adjusting grid methods for one-dimensional hyperbolic conservation laws. *Journal of computational physics*, 50(2):235–269, 1983. 8

[167] Amiram Harten, James M Hyman, Peter D Lax, and Barbara Keyfitz. On finite-difference approximations and entropy conditions for shocks. *Communications on pure and applied mathematics*, 29(3):297–322, 1976. vii, 1

[168] Amiram Harten and Peter D Lax. A random choice finite difference scheme for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 18(2):289–315, 1981. vii, 1

[169] Inga Hense and Aike Beckmann. The representation of cyanobacteria life cycle processes in aquatic ecosystem models. *Ecological Modelling*, 221(19):2330–2338, 2010. 9

[170] Juntao Huang and Chi-Wang Shu. Positivity-preserving time discretizations for production–destruction equations with applications to non-equilibrium flows. *Journal of Scientific Computing*, 78(3):1811–1839, 2019. 8, 9, 157, 158

[171] Juntao Huang, Weifeng Zhao, and Chi-Wang Shu. A third-order unconditionally positivity-preserving scheme for production–destruction equations with applications to non-equilibrium flows. *Journal of Scientific Computing*, pages 1–42, 2018. 8, 9, 157, 158

[172] Matthew E Hubbard, Mario Ricchiuto, and Domokos Sarmany. Space–time residual distribution on moving meshes. *Computers & Mathematics with Applications*, 79(5):1561–1589, 2020. 43

[173] T. J. R. Hughes, G. Scovazzi, and T. E. Tezduyar. Stabilized methods for compressible flows. *Journal of Scientific Computing*, 43(3):343–368, 2010. vii, 1

[174] Thomas JR Hughes, Leopaldo P Franca, and Michel Mallet. A new finite element formulation for computational fluid dynamics: I. symmetric forms of the compressible euler and navier-stokes equations and the second law of thermodynamics. *Computer methods in applied mechanics and engineering*, 54(2):223–234, 1986. vii, 1, 45

[175] T.J.R. Hughes, J. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry, and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:33–40, 2005. 5

[176] M. S. Ivanov, A. Bonfiglioli, R. Paciorri, and F. Sabetta. Computation of weak steady shock reflections by means of an unstructured shock-fitting solver. *Shock Waves*, 20(4):271–284, 2010. 89, 99, 105

[177] Mikhail S Ivanov, Aldo Bonfiglioli, Renato Paciorri, and Filippo Sabetta. Computation of weak steady shock reflections by means of an unstructured shock-fitting solver. *Shock Waves*, 20(4):271–284, 2010. 7, 145

[178] Thomas Izgin, Stefan Kopecz, and Andreas Meister. On Lyapunov stability of positive and conservative time integrators and application to second order modified Patankar–Runge–Kutta schemes. *arXiv preprint arXiv:2202.01099*, 2022. 9, 205

[179] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, 1996. viii, 2, 9, 39

[180] Claes Johnson and Juhani Pitkäranta. An analysis of the discontinuous galerkin method for a scalar hyperbolic equation. *Mathematics of computation*, 46(173):1–26, 1986. vii, 1

[181] Andrew D. Kercher, Andrew Corrigan, and David A. Kessler. The moving discontinuous galerkin finite element method with interface condition enforcement for compressible viscous flows. *International Journal for Numerical Methods in Fluids*, n/a(n/a). 89

[182] David I Ketcheson. Highly efficient strong stability-preserving runge–kutta methods with low-storage implementations. *SIAM Journal on Scientific Computing*, 30(4):2113–2136, 2008. 50

[183] M Ehsan Khalili, Martin Larsson, and Bernhard Müller. Immersed boundary method for viscous compressible flows around moving bodies. *Computers & Fluids*, 170:77–92, 2018. 3

[184] Christian Klingenberg and Bradley Plohr. An introduction to front tracking. In *Multidimensional Hyperbolic Problems and Computations*, pages 203–216. Springer, 1991. 7

[185] Stefan Kopecz and Andreas Meister. Unconditionally positive and conservative third order modified Patankar–Runge–Kutta discretizations of production–destruction systems. *BIT Numerical Mathematics*, pages 1–38, 2018. 9, 158

[186] Stefan Kopecz and Andreas Meister. On the existence of three-stage third-order modified Patankar–Runge–Kutta schemes. *Numerical Algorithms*, pages 1–12, 2019. 9, 158

[187] Lilia Krivodonova and Marsha Berger. High-order accurate implementation of solid wall boundary conditions in curved geometries. *Journal of computational physics*, 211(2):492–512, 2006. 5, 67, 69

[188] Norbert Kroll, Tobias Leicht, Charles Hirsch, Francesco Bassi, Craig Johnston, Kaare Sorensen, and Koen Hillewaert. Results and conclusions of the european project idihom on high-order methods for industrial aerodynamic applications. In *53rd AIAA Aerospace Sciences Meeting*, page 0293, 2015. 5

[189] Stanislav N Kružkov. First order quasilinear equations in several independent variables. *Mathematics of the USSR-Sbornik*, 10(2):217, 1970. 25, 26

[190] Alexander Kurganov. Finite-volume schemes for shallow-water equations. *Acta Numerica*, 27:289–351, 2018. 32

[191] Alexander Kurganov and Doron Levy. Central-upwind schemes for the saint-venant system. *ESAIM: Mathematical Modelling and Numerical Analysis*, 36(3):397–425, 2002. 8

[192] Alexander Kurganov and Guergana Petrova. A second-order well-balanced positivity preserving central-upwind scheme for the saint-venant system. *Communications in Mathematical Sciences*, 5(1):133–160, 2007. 10

[193] Dmitri Kuzmin and Manuel Quezada de Luna. Entropy conservation property and entropy stabilization of high-order continuous Galerkin approximations to scalar conservation laws. *Computers & Fluids*, 213:104742, 2020. viii, 2, 205

[194] Peter Lax. Shock waves and entropy. In *Contributions to nonlinear functional analysis*, pages 603–634. Elsevier, 1971. 25

[195] Peter D Lax. *Hyperbolic systems of conservation laws and the mathematical theory of shock waves*. SIAM, 1973. 25

[196] Peter D Lax and Robert D Richtmyer. Survey of the stability of linear finite difference equations. *Communications on pure and applied mathematics*, 9(2):267–293, 1956. vii, 1

[197] Martin Lesueur, Hadrien Rattez, and Oriol Colomés. μct scans permeability computation with an unfitted boundary method to improve coarsening accuracy. *Computers & Geosciences*, page 105118, 2022. 203

[198] Randall J LeVeque et al. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002. vii, 1, 47

[199] Randall J LeVeque and Randall J Leveque. *Numerical methods for conservation laws*, volume 214. Springer, 1992. 25, 31

[200] Kangan Li, Nabil M. Atallah, G. Alex Main, and Guglielmo Scovazzi. The shifted interface method: A flexible approach to embedded interface computations. *International Journal for Numerical Methods in Engineering*, 121(3):492–518, 2020. 3

[201] Qiuhua Liang and Fabien Marche. Numerical resolution of well-balanced shallow water equations with complex source terms. *Advances in water resources*, 32(6):873–884, 2009. 8

[202] TG Liu, BC Khoo, and KS Yeo. Ghost fluid method for strong shock impacting on material interface. *Journal of computational physics*, 190(2):651–681, 2003. 3

[203] Raphaël Loubere, Michael Dumbser, and Steven Diot. A new family of high order unstructured mood and ader finite volume schemes for multidimensional systems of hyperbolic conservation laws. *Communications in Computational Physics*, 16(3):718–763, 2014. 48, 70

[204] Maria Lukáčová-Medvid'ová, Sebastian Noelle, and Marcus Kraft. Well-balanced finite volume evolution galerkin methods for the shallow water equations. *Journal of computational physics*, 221(1):122–147, 2007. 8

[205] Xiao-Juan Luo, Mark S Shephard, Robert M O'bara, Rocco Nastasia, and Mark W Beall. Automatic p-version mesh generation for curved domains. *Engineering with Computers*, 20(3):273–285, 2004. 63

[206] A Main and G Scovazzi. The shifted boundary method for embedded domain computations. part I: Poisson and Stokes problems. *Journal of Computational Physics*, 372:972–995, 2018. 3, 61, 64, 111, 203, 204

[207] Alex Main and Guglielmo Scovazzi. The shifted boundary method for embedded domain computations. part II: Linear advection–diffusion and incompressible Navier–Stokes equations. *Journal of Computational Physics*, 372:996–1026, 2018. 3, 61, 64, 203, 204

[208] De-kang Mao. A treatment of discontinuities in shock-capturing finite difference methods. *Journal of computational physics*, 92(2):422–455, 1991. 8

[209] De-Kang Mao. A treatment of discontinuities for finite difference methods. *Journal of Computational Physics*, 103(2):359–369, 1992. 8

[210] De-kang Mao. A treatment of discontinuities for finite difference methods in the two-dimensional case. *Journal of Computational Physics*, 104(2):377–397, 1993. 8

[211] De-kang Mao. Towards front-tracking based on conservation in two space dimensions ii, tracking discontinuities in capturing fashion. *Journal of Computational Physics*, 226(2):1550–1588, 2007. 8

[212] Diogo MC Martins, Duarte MS Albuquerque, and José CF Pereira. On the use of polyhedral unstructured grids with a moving immersed boundary method. *Computers & Fluids*, 174:78–88, 2018. 3

[213] Sandra May and Marsha Berger. An explicit implicit scheme for cut cells in embedded boundary meshes. *Journal of Scientific Computing*, 71(3):919–943, 2017. 3, 5

[214] Alireza Mazaheri, Chi-Wang Shu, and Vincent Perrier. Bounded and compact weighted essentially nonoscillatory limiters for discontinuous galerkin schemes: Triangular elements. *Journal of Computational Physics*, 395:461–488, 2019. 72

[215] Andreas Meister and Sigrun Ortleb. On unconditionally positive implicit time integration for the DG scheme applied to shallow water flows. *International Journal for Numerical Methods in Fluids*, 76(2):69–94, 2014. 8, 157

[216] Andreas Meister and Sigrun Ortleb. A positivity preserving and well-balanced DG scheme using finite volume subcells in almost dry regions. *Applied Mathematics and Computation*, 272:259–273, 2016. 8, 9, 160

[217] Sixtine Michel, Davide Torlo, Mario Ricchiuto, and Rémi Abgrall. Spectral analysis of continuous fem for hyperbolic pdes: influence of approximation, stabilization, and time-stepping. *Journal of Scientific Computing*, 89(2):1–41, 2021. 48

[218] Victor Michel-Dansac, Christophe Berthon, Stéphane Clain, and Françoise Foucher. A well-balanced scheme for the shallow-water equations with topography. *Computers & Mathematics with Applications*, 72(3):568–593, 2016. 8, 10, 205, 206

[219] Victor Michel-Dansac, Christophe Berthon, Stéphane Clain, and Françoise Foucher. A well-balanced scheme for the shallow-water equations with topography or manning friction. *Journal of Computational Physics*, 335:115–154, 2017. 10, 205, 206

[220] Victor Michel-Dansac, Christophe Berthon, Stéphane Clain, and Françoise Foucher. A two-dimensional high-order well-balanced scheme for the shallow water equations with topography and manning friction. *Computers & Fluids*, 230:105152, 2021. 10

[221] Michael L Minion. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Communications in Mathematical Sciences*, 1(3):471–500, 2003. 50

[222] Gino Moretti. Importance of boundary conditions in the numerical treatment of hyperbolic equations. *The physics of fluids*, 12(12):II–13, 1969. 203

[223] GlNO MORETTI and MlCHAEL ABBETT. A time-dependent computational method for blunt body flows. *AIAA Journal*, 4(12):2136–2141, 1966. 6

[224] D. Moxey, D. Ekelschot, Ü. Keskin, S.J. Sherwin, and J. Peiró. High-order curvilinear meshing using a thermo-elastic analogy. *Computer-Aided Design*, 72:130–139, 2016. 63

[225] J. D. Müller. Grid generation tools: Delaunay triangulation with delaundo. 103, 105

[226] J-D Müller, Philip L Roe, and H Deconinck. A frontal approach for internal node generation in delaunay triangulations. *International Journal for Numerical Methods in Fluids*, 17(3):241–255, 1993. 133, 140

[227] J.D. Müller. Delaundo mesh generator. Available at `http://www.ae.metu.edu.tr/tuncer/ae546/prj/delaundo/`. 133, 140

[228] F Nasuti and M Onofri. Analysis of unsteady supersonic viscous flows by a shock-fitting technique. *AIAA journal*, 34(7):1428–1434, 1996. 7

[229] Francesco Nasuti. A multi-block shock-fitting technique to solve steady and unsteady compressible flows. In *Computational Fluid Dynamics 2002*, pages 217–222. Springer, 2003. 7

[230] Sebastian Noelle, Yulong Xing, and Chi-Wang Shu. High-order well-balanced finite volume WENO schemes for shallow water equation with moving water. *Journal of Computational Physics*, 226(1):29–58, 2007. 8

[231] L Nouveau, Heloise Beaugendre, C Dobrzynski, R Abgrall, and Mario Ricchiuto. An adaptive, residual based, splitting approach for the penalized Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 303:208–230, 2016. 3

[232] Leo Nouveau, Mario Ricchiuto, and Guglielmo Scovazzi. High-order gradients with the shifted boundary method: An embedded enriched mixed formulation for elliptic pdes. *Journal of Computational Physics*, 398:108898, 2019. 3, 203

[233] Jonatan Núnez-De La Rosa and Claus-Dieter Munz. Hybrid DG/FV schemes for magnetohydrodynamics and relativistic hydrodynamics. *Computer Physics Communications*, 222:113–135, 2018. 35

[234] Philipp Öffner and Davide Torlo. Arbitrary high-order, conservative and positivity preserving patankar-type deferred correction schemes. *Applied Numerical Mathematics*, 153:15–34, 2020. x, 10, 11, 48, 155, 158, 159, 162

[235] Renato Paciorri and Aldo Bonfiglioli. A shock-fitting technique for 2D unstructured grids. *Computers & Fluids*, 38(3):715–726, 2009. 7, 111, 113, 125, 151, 204

[236] Renato Paciorri and Aldo Bonfiglioli. A shock-fitting technique for 2d unstructured grids. *Computers & Fluids*, 38(3):715 – 726, 2009. 89, 95, 97, 99, 104, 105

[237] Renato Paciorri and Aldo Bonfiglioli. Shock interaction computations on unstructured, two-dimensional grids using a shock-fitting technique. *Journal of Computational Physics*, 230(8):3155–3177, 2011. 7, 140

[238] Renato Paciorri and Aldo Bonfiglioli. Shock interaction computations on unstructured, two-dimensional grids using a shock-fitting technique. *Journal of Computational Physics*, 230(8):3155 – 3177, 2011. 89, 99, 105

[239] Renato Paciorri and Aldo Bonfiglioli. Basic elements of unstructured shock-fitting: Results achieved and future developments. In Marcello Onofri and Renato Paciorri, editors, *Shock Fitting: Classical Techniques, Recent Developments, and Memoirs of Gino Moretti*, pages 59–84. Springer International Publishing, Cham, 2017. 109

[240] Renato Paciorri and Aldo Bonfiglioli. Accurate detection of shock waves and shock interactions in two-dimensional shock-capturing solutions. *Journal of Computational Physics*, 406:109196, 2020. 108, 109

[241] Renato Paciorri and Aldo Bonfiglioli. Accurate detection of shock waves and shock interactions in two-dimensional shock-capturing solutions. *Journal of Computational Physics*, 406:109196, 2020. 116

[242] H Paillere and H Deconinck. Compact cell vertex convection schemes on unstructured meshes. In *Problèmes non linéaires appliqués. Ecoles CEA-EDF-INRIA*, pages 91–139, 2000. 46

[243] Suhas Patankar. *Numerical heat transfer and fluid flow*. CRC press, 1980. 9, 158

[244] Richard B Pember, John B Bell, Phillip Colella, William Y Curtchfield, and Michael L Welcome. An adaptive cartesian grid method for unsteady compressible flow in irregular regions. *Journal of computational Physics*, 120(2):278–304, 1995. 8

[245] R. Pepe, A. Bonfiglioli, A. D'Angola, G. Colonna, and R. Paciorri. Shock-fitting versus shock-capturing modeling of strong shocks in nonequilibrium plasmas. *Plasma Science, IEEE Transactions on*, 42(10):2526–2527, Oct 2014. 89

[246] Raffaele Pepe, Aldo Bonfiglioli, Renato Paciorri, Andrea Lani, Jesus Garicano-Mena, and Carl F. Ollivier-Gooch. *Towards a Modular Approach for Unstructured Shock-Fitting*. International Center for Numerical Methods in Engineering, Barcelona, Spain, 2014. 89

[247] Benoit Perthame and Chi-Wang Shu. On positivity preserving finite volume schemes for euler equations. *Numerische Mathematik*, 73(1):119–130, 1996. viii, 2, 8, 9, 32, 40, 155

[248] Charles S Peskin. Flow patterns around heart valves: a numerical method. *Journal of computational physics*, 10(2):252–271, 1972. 2, 3

[249] Sergio Pirozzoli. Generalized conservative approximations of split convective derivative operators. *Journal of Computational Physics*, 229(19):7180–7190, 2010. 9

[250] Sergio Pirozzoli. Numerical methods for high-speed flows. *Annual Review of Fluid Mechanics*, 43(1):163–194, jan 2011. vii, 1

[251] Sergio Pirozzoli. Stabilized non-dissipative approximations of euler equations in generalized curvilinear coordinates. *Journal of Computational Physics*, 230(8):2997–3014, 2011. 9

[252] A Prakash, N Parsons, X Wang, and X Zhong. High-order shock-fitting methods for direct numerical simulation of hypersonic flow with chemical and thermal nonequilibrium. *Journal of Computational Physics*, 230(23):8474–8507, 2011. 7

[253] J.J. Quirk. A contribution to the great Riemann solver debate. *International Journal for Numerical Methods in Fluids*, 18(6):555–574, 1994. 6

[254] Hendrik Ranocha. Shallow water equations: Split-form, entropy stable, well-balanced, and positivity preserving numerical methods. *GEM – International Journal on Geomathematics*, 8(1):85–133, 04 2017. viii, 2, 205

[255] Hendrik Ranocha and Philipp Öffner. L2 stability of explicit runge–kutta schemes. *Journal of Scientific Computing*, 75(2):1040–1056, 2018. 50

[256] William H Reed and Thomas R Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Lab., N. Mex.(USA), 1973. vii, 1, 41

[257] Julius Reiss and Jörn Sesterhenn. A conservative, skew-symmetric finite difference scheme for the compressible navier–stokes equations. *Computers & Fluids*, 101:208–219, 2014. 9

[258] Mario Ricchiuto. On the C-property and generalized C-property of residual distribution for the shallow water equations. *Journal of Scientific Computing*, 48(1):304–318, 2011. viii, 2

[259] Mario Ricchiuto. An explicit residual based approach for shallow water flows. *Journal of Computational Physics*, 280:306 – 344, 2015. 91

[260] Mario Ricchiuto and Remi Abgrall. Explicit runge–kutta residual distribution schemes for time dependent problems: second order case. *Journal of Computational Physics*, 229(16):5653–5691, 2010. 43, 91

[261] Mario Ricchiuto and Andreas Bollermann. Stabilized residual distribution for shallow water simulations. *Journal of Computational Physics*, 228(4):1071–1115, 2009. 43, 175

[262] Mario Ricchiuto and Davide Torlo. Analytical travelling vortex solutions of hyperbolic equations for validating very high order schemes. *arXiv preprint arXiv:2109.10183*, 2021. 164

[263] Philip L Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981. vii, 1, 43, 118

[264] Philip L Roe. Characteristic-based schemes for the euler equations. *Annual review of fluid mechanics*, 18(1):337–365, 1986. 43

[265] Axel Rohde. Eigenvalues and eigenvectors of the euler equations in general geometries. In *15th AIAA Computational Fluid Dynamics Conference*, page 2609, 2001. 28

[266] Alexandre M Roma, Charles S Peskin, and Marsha J Berger. An adaptive version of the immersed boundary method. *Journal of computational physics*, 153(2):509–534, 1999. 3

[267] O Sahni, XJ Luo, KE Jansen, and MS Shephard. Curved boundary layer meshing for adaptive viscous flow simulations. *Finite Elements in Analysis and Design*, 46(1-2):132–139, 2010. 63

[268] M. D. Salas. Shock fitting method for complicated two-dimensional supersonic flows. *Aiaa Journal*, 14(5):583 – 588, 1976. 6

[269] Manuel D Salas. *A shock-fitting primer*. CRC Press, 2009. 113

[270] Kirill Semeniuk and Ashu Dastoor. Development of a global ocean mercury model with a methylation cycle: Outstanding issues. *Global Biogeochemical Cycles*, 31(2):400–433, 2017. 9

[271] D. She, R. Kaufman, H. Lim, J. Melvin, A. Hsu, and J. Glimm. Front tracking methods. In *Handbook of Numerical Methods for Hyperbolic Problems*, volume 17, pages 383—402. Elsevier, 2016. 7

[272] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry. 90

[273] Jonathan Richard Shewchuk. Triangle: A two-dimensional quality mesh generator and delaunay triangulator., 2005. 90

[274] Jing Shi, Changqing Hu, and Chi-Wang Shu. A technique of treating negative weights in WENO schemes. *Journal of Computational Physics*, 175(1):108–127, 2002. 37, 39, 40

[275] Chi-Wang Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced numerical approximation of nonlinear hyperbolic equations*, pages 325–432. Springer, 1998. 9, 35, 37, 181

[276] Chi-Wang Shu. High-order finite difference and finite volume weno schemes and discontinuous galerkin methods for cfd. *International Journal of Computational Fluid Dynamics*, 17(2):107–118, 2003. viii, 2

[277] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988. 35, 49, 181

[278] Gary A Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of computational physics*, 27(1):1–31, 1978. vii, 1

[279] Ting Song, Alex Main, Guglielmo Scovazzi, and Mario Ricchiuto. The shifted boundary method for hyperbolic systems: Embedded domain computations of linear waves and shallow water flows. *Journal of Computational Physics*, 369:45–79, 2018. 3, 64, 67, 118

[280] R Struijs, Herman Deconinck, and PL Roe. Fluctuation splitting schemes for the 2d euler equations. *In its Computational Fluid Dynamics 94 p (SEE N91-32426 24-34)*, 1991. 46

[281] Blair K Swartz and Burton Wendroff. Aztec: A front tracking code based on godunov's method. *Applied numerical mathematics*, 2(3-5):385–397, 1986. 8

[282] Eitan Tadmor. Numerical viscosity and the entropy condition for conservative difference schemes. *Mathematics of Computation*, 43(168):369–381, 1984. 9

[283] Eitan Tadmor. Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems. *Acta Numerica*, 12:451–512, 2003. 9

[284] Hiroshi Terashima and Grétar Tryggvason. A front-tracking/ghost-fluid method for fluid interfaces in compressible flows. *Journal of Computational Physics*, 228(11):4012–4037, 2009. 3

[285] Vladimir A Titarev and Eleuterio F Toro. Finite-volume WENO schemes for three-dimensional conservation laws. *Journal of Computational Physics*, 201(1):238–260, 2004. 37, 39, 40, 215

[286] Davide Torlo, Philipp Öffner, and Hendrik Ranocha. A new Stability Approach for Positivity-Preserving Patankar-type Schemes. *arXiv:2108.07347*, 2021. 158, 205

[287] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013. 31, 36

[288] Eleuterio F Toro, RC Millington, and LAM Nejad. Towards very high order godunov schemes. In *Godunov methods*, pages 907–940. Springer, 2001. 54

[289] Elewterio F Toro. *Godunov methods: Theory and applications*. Springer Science & Business Media, 2012. vii, 1

[290] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts. Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8–26, 2013. 63

[291] M. Turner, J. Peiró, and D. Moxey. Curvilinear mesh generation using a variational framework. *Computer-Aided Design*, 103:73–91, 2018. 63

[292] Bram Van Leer. Towards the ultimate conservative difference scheme. ii. monotonicity and conservation combined in a second-order scheme. *Journal of computational physics*, 14(4):361–370, 1974. 36

[293] Bram Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to Godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979. 35, 36

[294] Maria Han Veiga, Philipp Öffner, and Davide Torlo. Dec and ader: Similarities, differences and a unified framework. *Journal of Scientific Computing*, 87(1):1–35, 2021. 48, 53

[295] François Vilar, Pierre-Henri Maire, and Remi Abgrall. A discontinuous galerkin discretization for solving the two-dimensional gas dynamics equations written under total lagrangian formulation on general unstructured grids. *Journal of Computational Physics*, 276:188–234, 2014. 5

[296] John VonNeumann and Robert D Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of applied physics*, 21(3):232–237, 1950. 6

[297] Cheng Wang, Xiangxiong Zhang, Chi-Wang Shu, and Jianguo Ning. Robust high order discontinuous Galerkin schemes for two-dimensional gaseous detonations. *Journal of Computational Physics*, 231(2):653–665, 2012. 206

[298] Z.J. Wang, Krzysztof Fidkowski, Rémi Abgrall, Francesco Bassi, Doru Caraeni, Andrew Cary, Herman Deconinck, Ralf Hartmann, Koen Hillewaert, H.T. Huynh, Norbert Kroll, Georg May, Per-Olof Persson, Bram van Leer, and Miguel Visbal. High-order cfd methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013. 63

[299] ZJ Wang and Yuzhi Sun. Curvature-based wall boundary condition for the euler equations on unstructured grids. *AIAA journal*, 41(1):27–33, 2003. 5

[300] Yulong Xing and Chi-Wang Shu. A survey of high order schemes for the shallow water equations. *J. Math. Study*, 47(3):221–249, 2014. viii, 2, 40, 179, 205

[301] Yulong Xing, Chi-Wang Shu, and Sebastian Noelle. On the advantage of well-balanced schemes for moving-water equilibria of the shallow water equations. *Journal of scientific computing*, 48(1):339–349, 2011. 8

[302] Junhui Yin, Li Xu, Peng Xie, Lan Zhu, Shucheng Huang, Hangxin Liu, Zhonghai Yang, and Bin Li. A curved boundary treatment for discontinuous galerkin method applied to euler equations on triangular and tetrahedral grids. *Computer Physics Communications*, 258:107549, 2021. 5

[303] M.J. Zahr and P.-O. Persson. An optimization-based approach for high-order accurate discretization of conservation laws with discontinuous solutions. *Journal of Computational Physics*, 365:105 – 134, 2018. 89

[304] M.J. Zahr, A. Shi, and P.-O. Persson. Implicit shock tracking using an optimization-based high-order discontinuous galerkin method. *Journal of Computational Physics*, 410:109385, 2020. 151

[305] Daniel Zaide and Philip Roe. Shock capturing anomalies and the jump conditions in one dimension. In *20th AIAA Computational Fluid Dynamics Conference*, 2011. 6

[306] Xiangxiong Zhang and Chi-Wang Shu. On positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes. *Journal of Computational Physics*, 229(23):8918–8934, 2010. x, 8, 9, 12, 40, 155

[307] Xiangxiong Zhang and Chi-Wang Shu. Positivity-preserving high order finite difference WENO schemes for compressible euler equations. *Journal of Computational Physics*, 231(5):2245–2258, 2012. 206

[308] O.C. Zienkiewicz and P. Morice. *The finite element method in engineering*. McGraw-Hill, 1971. 5

[309] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique. *International Journal for Numerical Methods in Engineering*, 33(7):1331–1364, 1992. 123

[310] Dongyang Zou, Aldo Bonfiglioli, Renato Paciorri, and Jun Liu. An embedded shock-fitting technique on unstructured dynamic grids. *Computers & Fluids*, 218:104847, 2021. 89, 147