POLITECNICO DI MILANO

Mathematical Engineering - Computation Science and Engineering

ADVANCED PROGRAMMING FOR SCIENTIFIC COMPUTING

The nonlinear hyperbolic Euler system of compressible gas dynamics

Advisor: Prof. Paolo Barbante

Sonia Felotti
ID: 928319

Academic Year 2022-2023

Contents

Abstract

In this project, it's addressed the problem of solving systems of hyperbolic conservation laws that arises in many fields, such as in fluid dynamics where gas dynamics of compressible (especially for high mach numbers) and incompressible flows are studied. The major difficulty that one has to face in order to deal with this type of problems is that such systems have the property that discontinuous solutions can appear even if initial data are smooth. For this reason, numerical methods should balance the conflicting requirements of accuracy of the solution in regions where high gradients are present and high order accuracy where the solution is smooth. An explicit Runge-Kutta discontinuous Galerkin (RKDG) method is used to work out numerical schemes for compressible Euler equations of gas dynamics. Different techniques are reviewed in order to control spurious oscillations, in particular a shock detection technique is shown to be useful in order to determinate the regions where the spurious oscillations appear such that a "filter" can be used to eliminate these numeric artifacts. Furthermore, different numerical fluxes are used, with the aim of verifying which ones are more optimal at the different limiter techniques.

Introduction

Mathematical models describe physical systems responding to changes in themselves over space and/or time. These mathematical systems, called partial differential equations (PDEs), are used to model a considerable amount of physical phenomena, such as in fluid dynamics where gas dynamics of compressible (especially for high mach numbers) and incompressible flows are studied. They are modeled by functions, which typically depend on a coordinate variable x, or in multiple dimensions $\mathbf{x}$, and/or a time component t. This project is exclusively concerned with solving a specific subset of PDEs called hyperbolic systems of conservation laws, studied in [23]. Our aim is to numerically approximate solutions to systems of hyperbolic conservation laws. Computing high-order accurate numerical approximations is incredibly expensive. Furthermore, complex hyperbolic conservation laws produce difficult to model physical phenomena such as discontinuities. Discontinuous Galerkin (DG) methods combine features of finite element methods and finite volume method. They are robust and high-order accurate, able to model the difficult to capture physical phenomena common to hyperbolic conservation laws. They use arbitrarily high-order approximations without increasing the stencil size. This feature is entirely due to the discontinuous element-local nature of the method, a key for parallelization techniques. They are able to run on unstructured meshes which can be adaptively refined during computation to assist in capturing moving physical

phenomenon such as shock waves. Furthermore, each element can use a different order approximation, allowing the solution over problematic areas to be cheaply computed by low-order approximations while approximations over non-troublesome areas can be more accurately captured using very high-order approximations. DG methods are especially open to parallel implementations for the following reasons. First, due to their discontinuous nature, they are element local, requiring only information from their immediate neighboring elements to advance the solution to the next time level, even for arbitrarily high-order approximations. Each element, therefore, may be thought of as an independent approximation. Furthermore, they can be paired with an explicit time stepping method, able to step forward in time using only previous information. The computed solution is discontinuous along the edges of each element of the triangulation. For this reason, flux reconstruction methods are necessary to numerically evaluate the flux along the boundaries of the elements. Since a number of flux reconstruction formulas are successfully used in the frame of FV, DG methods simply borrow them.
The numerical schemes are implemented using `deal.II` libraries in the `Eulero` code, therefore the language of choice for the development of this project was `C++`. The `Eulero` code uses the general structure of dealii's tutorial 67 [26] in which the basic functions for the DG are implemented. The tutorial has been modified to implement other numerical fluxes, limiters and indicator, adaptivity of the mesh, supersonic boundary conditions and explicit SSP integration. The solution of common test cases show the capability of the method.

The report is structured as follows:

- firstly basic concepts of conservation laws are introduced
- the model problem and its DG discretization, focusing on numerical fluxes and limiting techniques
- the code structure and, finally, the correctness of the code is tested comparing numerical and theoretical results

Chapter 1

Basic Concepts

Introduction

In this chapter, the basic concepts [23] concerning conservation laws are introduced for a better understanding of the later chapters.

1.1 Conservation laws

Consider the open set $D \subset \mathbb{R}^d$, $0 < d < 3$ as a control volume. If the temporal change of variable's density $\mathbf{w}$ is determined by its variation $\mathbf{F}(\mathbf{w})$ through the boundaries ∂D of the control volume as

$$\partial_t \int_D \mathbf{w} dV = -\int_{\partial D} \mathbf{F}(\mathbf{w}) \cdot \mathbf{n} dS, \tag{1.1}$$

where $\mathbf{n}$ is an outer normal unit vector, then $\mathbf{w}$ is said to be conserved, or to satisfy a **conservation law** and equation (1.1) is called the **integral form**. The variable $\mathbf{w} : (t, \mathbf{x}) \in \mathbb{R}^+ \times D \to \Omega \subset \mathbb{R}^m, \mathbf{w} < \infty$ is called the vector of conserved variables and $\mathbf{F} : \mathbb{R}^m \to \mathbb{R}^{m \times d}$ are called flux functions. Assuming that the control volume D does not change with time and $\mathbf{F}$ is differentiable, applying the divergence theorem equation (1.1) can be written as

$$\int_D (\partial_t \mathbf{w} + \nabla_{\mathbf{x}} \cdot \mathbf{F}(\mathbf{w})) dV = 0. \tag{1.2}$$

Then, dividing (1.2) by the volume of the set D and shrinking it to a point, the **differential form** of the conservation law is obtained as

$$\partial_t \mathbf{w} + \nabla_{\mathbf{x}} \cdot \mathbf{F}(\mathbf{w}) = 0. \tag{1.3}$$

At this point, some important things can be noticed. First of all, even if the spatial domain of the problem is one-dimensional ($d = 1$), $\mathbf{w}$ is not restricted to be scalar.

Furthermore, (1.3) is a first order partial differential equation (PDE), which will allow solutions in the classical sense only when $\mathbf{w} \in C^1(\mathbb{R}^+ \times D)$, however, as it will be shown later, the solutions allowed by a conservation laws are not restricted to be classical. Equation (1.3) can be also be written in quasi linear form as

$$\partial_t \mathbf{w} + \sum_{k=1}^{d} \mathbf{A}_k \partial_{x_k} \mathbf{w} = 0, \tag{1.4}$$

where $\mathbf{A}_k$ is defined as

$$\mathbf{A}_k = \left[\frac{\partial F_i}{\partial w_j}\right]_k \tag{1.5}$$

the Jacobian matrix resulting from partially derive each function of $F_{i,k}$ with respect to w_j, where $1 \leq i,\ j \leq m$ and $1 \leq k \leq d$. If all the eigenvalues of the matrix $\mathbf{A}_k$, $k = 1, .., d$ are real, the system of (1.3) is said to be **hyperbolic**. Moreover, depending on the type of eigenvalues that the system has, it can be characterized as either **strictly**, **strogly** or **weakly** hyperbolic.

1.1.1 Non-linear conservation laws

Consider the scalar conservation laws IVP given by

$$\begin{cases} \partial_t w + \partial_x f(w) = 0, \\ w(0, x) = w_0(x), \end{cases} \tag{1.6}$$

where $f \in C^1(\Omega), \Omega \subset \mathbb{R}$ is a non-linear function. The quasi linear form is given by

$$\begin{cases} \partial_t w + f'(w)\partial_x w = 0, \\ w(0, x) = w_0(x). \end{cases} \tag{1.7}$$

Let $\psi(t)$ be a parametrization of the space component x in terms of the time t, along which the solution $w(t, x)$ of the IVP (1.7) remains constant in time. The following expression is then satisfied

$$\frac{d}{dt} w(t, \psi(t)) = \partial_t w(t, \psi(t)) + \psi'(t)\partial_x w(t, \psi(t)) = 0, \tag{1.8}$$

only when $\psi(t) = f'(w(t, x(t)))$. Then, the expression $\psi(t)$ is called **characteristic curve**, and it is described by the ODE

$$\begin{cases} \psi'(t) = f'(w(t, \psi(t))), \\ \psi(0) = x_0. \end{cases} \tag{1.9}$$

Now, probably one of the most interesting things about hyperbolic conservation laws, is that they can develop discontinuous solutions starting from smooth initial conditions, and this can occur when the characteristic intersect. In order to

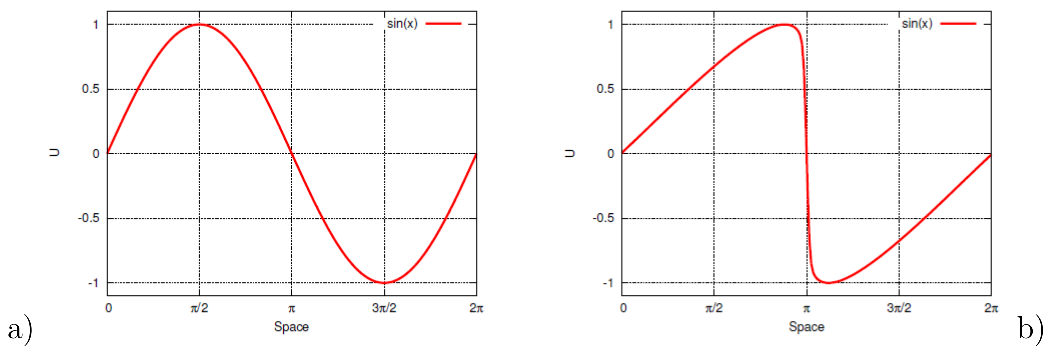


Figure 1.1: Time evolution of the IVP (1.7). a) initial condition, b) shock formation

illustrate this issue, consider the simplest non-linear scalar conservation law, called **inviscid Burgers' equation**

$$\partial_t w(t,x) + \partial_x \left(\frac{w^2(t,x)}{2} \right) = 0, \tag{1.10}$$

where $(t, x) \in \mathbb{R}_+ \times \mathbb{R}$. The quasi linear form for expression (1.10) is obtained

$$\begin{cases} \partial_t w(t,x) + w(t,x)\partial_x w(t,x) = 0, \\ w(0,x) = w_0(x). \end{cases} \tag{1.11}$$

The characteristic lines could cross, in which case a discontinuity will appear in the solution. As an example, take the IVP (1.10) with $w_0(x) = sin(x)$, $x \in [0, 2\pi]$ as illustrated in Figure (1.1) According to (1.9), there are three different possible range of values for the characteristic slope

$$\psi'(t) = \begin{cases} w_0(x) > 0, & x \in (0,\pi), \\ w_0(x) = 0, & x = 0, x = \pi, x = 2\pi, \\ w_0(x) < 0, & x \in (\pi, 2\pi). \end{cases} \tag{1.12}$$

This means that for $x \in (0, \pi)$ the solution will move towards the right, and similarly, for $x \in (\pi, 2\pi)$ the solution will move towards the left. After a brief period of time the characteristic curve around $x = \pi$ will cross generating a discontinuity (Figure 1.1b), where the conserved variable w will jump from having a positive value to a negative value.

The fact that the solution is not continuous anymore generates a contradiction, as a discontinuous function cannot be differentiated. Therefore, the set of solutions in which $w(t, x)$ lays should include also discontinuous functions that allow the so called weak derivatives, these solutions are then called **weak solutions**. Then, multiplying equation (1.3) by a test function $\phi(x)$ and integrating over space and

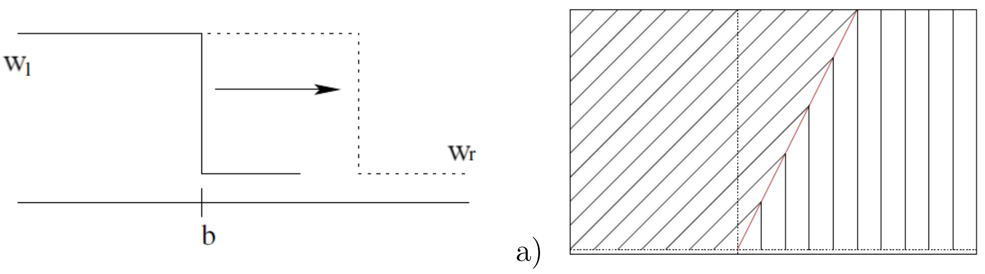


Figure 1.2: Solution for Case 1. a) Movement of the shock; b) Characteristics

time follows

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}_+}[\partial_t \mathbf{w} + (\nabla_x \cdot \mathbf{f}(\mathbf{w}))]\phi dt d\mathbf{x} = 0. \tag{1.13}$$

Then, integrating by parts the following condition should be satisfied by weak solutions

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}_+}[\phi\partial_t \mathbf{w} - \mathbf{f}(\mathbf{w})\nabla_x\phi] dt d\mathbf{x} + \int_{\mathbb{R}^d}\phi(0,x)\mathbf{w}_0(x)d\mathbf{x} = 0, \tag{1.14}$$

for all $\phi \in C_0^\infty(\mathbb{R}^d)$[23]. This solution is also called a generalized solution for PDE (1.3). Since not all discontinuous functions are included in the set of admissible solutions, a generalized solution must satisfy the so called **Rankine-Hugoniot jump conditions** explained as follows [23]. Let $S(t)$ be a smooth surface moving with t, and let a generalized solution $\mathbf{w}$ of (1.3) be discontinuous along $S(t)$, but be continuous differentiable in either side of $S(t)$. Then

$$s(t)[w^j] = [f^j(\mathbf{w})] \cdot \mathbf{n} \tag{1.15}$$

must be satisfied at each point of $S(t)$. Here $[w^j]$ and $[f^j(\mathbf{w})]$ are respectively the differences between the values of $\mathbf{w}$ and $\mathbf{F}$ on the two sides of $S(t)$, $\mathbf{n}$ is a vector normal to $S(t)$ and $s(t)$ is the speed at which $S(t)$ propagates in the direction $\mathbf{n}$.

1.1.2 The Riemann problem

Consider the following initial condition

$$w_0(x) = \begin{cases} w_l & if \;\; x < b, b \in \mathbb{R} \\ w_r & if \;\; x > b, b \in \mathbb{R} \end{cases} \tag{1.16}$$

and assume for now that $w_l > 0$, $w_r > 0$. A conservation law (1.6) with this initial condition is called a **Riemann Problem**. Taking once again as illustration example the scalar Burgers equation (1.10), two different kinds of solutions arise from this problem depending on the values of w_l and w_r.

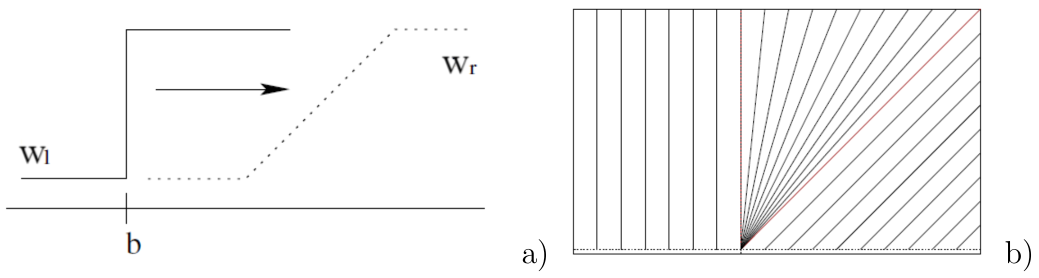


Figure 1.3: Solution for Case 2. a) Rarefaction; b) Characteristics

Case 1: $w_l > w_r$. The characteristic curves at the left of the jump "crash" against the characteristic curves at the right, merging into what is called a **shock wave** (Figure 1.2b). The speed s at which the discontinuity moves, can be computed using the Rankine-Hugoniot jump conditions. In the particular case of scalar Burgers' equation it is be given by

$$s = \frac{w_l + w_r}{2}. \tag{1.17}$$

Case 2: $w_l < w_r$. The characteristic curves at the left side of the discontinuity will have a slower speed than the one at the right side, which produces a "gap" in the domain with no defined solution. A different approach would be to create a transition between the left and right solutions as time evolves, as it is shown in Figure 1.3b. This transition is called a **rarefaction wave**, which is in fact a vanishing viscosity generalized solution. This solution is defined as

$$\begin{cases} w_l & if \quad x < b + w_l t, \\ x/l & if \quad b + w_l t < x < b + w_r t, \\ w_r & if \quad x > b + w_r t. \end{cases} \tag{1.18}$$

Unfortunately, weak solutions are often not unique and it is still considered an open problem to prove uniqueness of solutions to general systems of conservation laws in more than one space dimension. For scalar problems, the criteria to find the correct weak solution can be done by adding restrictions in the way weak solutions are found. These kind of restrictions are called **entropy conditions**, by analogy to physical systems where solutions that are "physically meaningful" are selected.

1.2 Mach Number

The subject of compressible flow is also called gas dynamics, and it has wide applications in high-speed flows around objects of engineering interest. Several startling and fascinating phenomena arise in compressible flows (especially in the supersonic range) that defy expectations developed from incompressible flows. Discontinuities appear within the flow, and a rather strange circumstance arises in which an increase of flow area accelerates a supersonic stream. And, in subsonic compressible duct flow, friction can increase the flow's speed and heat addition can lower the flow's temperature. These phenomena, which have no counterparts in low-speed flows, are therefore worthy of attention. The importance of compressibility in the equations of motion can be assessed by considering the Mach number M. By definition, it is a dimensionless and it is defined as

$$Mach\ number = M = \frac{speed\ of\ object}{speed\ of\ sound} = \frac{v}{c} \tag{1.19}$$

where v is a representative flow speed, and c is the speed of sound, a thermodynamic quantity defined by:

$$c^2 = (\partial p/\partial \rho)_s \tag{1.20}$$

Here the subscript s signifies that the partial derivative is taken at constant entropy. In particular, the dimensionless scaling of the compressible-flow continuity equation for isentropic conditions leads to:

$$\nabla \cdot u = -M^2 \left(\frac{\rho_0}{\rho}\right) \frac{D}{Dt} \left(\frac{p - p_0}{\rho_0 U^2}\right) \tag{1.21}$$

where ρ_0 and p_0 are appropriately chosen reference values for density and pressure. In (1.21), the pressure is scaled by fluid inertia parameters as is appropriate for primarily frictionless high-speed flow. In engineering practice, the incompressible flow assumption is presumed valid if $M < 0.3$, but not at higher Mach numbers. Equation (1.21) suggests that $M = 0.3$ corresponds to $\approx 10\%$ departure from perfectly incompressible flow behavior when the remainder of the right side of (1.21) is of order unity. Using the Mach number, compressible flows can be nominally classified as follows:

- *Incompressible flow*: $M = 0$. Fluid density does not vary with pressure in the flow field. The flowing fluid may be a compressible gas but its density may be regarded as constant.

- *Subsonic flow*: $0 < M < 1$. The Mach number does not exceed unity anywhere in the flow field. Shock waves do not appear in the flow. In engineering practice, subsonic flows for which $M < 0.3$ are often treated as being incompressible.

- *Transonic flow*: The Mach number in the flow lies in the range 0.8–1.2. Shock waves may appear. Analysis of transonic flows is difficult because the governing equations are inherently nonlinear, and also because a separation of the inviscid and viscous aspects of the flow is often impossible.

- *Supersonic flow*: $M > 1$. Shock waves are generally present. In many ways analysis of a flow that is supersonic everywhere is easier than an analysis of a subsonic or incompressible flow. This is because information propagates along certain directions, called characteristics, and a determination of these directions greatly facilitates the computation of the flow field.

- *Hypersonic flow*: $M > 3$. Very high flow speeds combined with friction or shock waves may lead to sufficiently large increases in a fluid's temperature that molecular dissociation and other chemical effects occur.

Chapter 2

Model problem

Introduction

A particular system of conservation law, the Euler equation, are introduced in this chapter. Then, the basic concepts is introduced and characteristics of the strategy which will be implemented for the numerical solution of systems of conservation laws.

2.1 Euler equations

The Euler equations are a conservation law, describing the motion of a compressible inviscid gas,

$$\frac{\partial \mathbf{w}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{w}) = \mathbf{G}(\mathbf{w}), \tag{2.1}$$

coupled with suitable initial conditions $\mathbf{w}(x, 0) = \mathbf{w}_0(x)$, where the $d+2$ components of the solution vector are $\mathbf{w} = (\rho, \rho u_1, \dots, \rho u_d, E)^{\mathrm{T}}$. Here, ρ denotes the fluid density, $\mathbf{u} = (u_1, \dots, u_d)^{\mathrm{T}}$ the fluid velocity, and E the energy density of the gas. The velocity is not directly solved for, but rather the variable $\rho\mathbf{u}$, the linear momentum (since this is the conserved quantity). The Euler flux function, a $(d+2) \times d$ matrix, is defined as

$$\mathbf{F}(\mathbf{w}) = \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u} \otimes \mathbf{u} + \mathbb{I}p \\ (E+p)\mathbf{u} \end{pmatrix} \tag{2.2}$$

with $\mathbb{I}$ the $d \times d$ identity matrix and $\otimes$ the outer product; its components denote the mass, momentum, and energy fluxes, respectively. The right hand side forcing is given by

$$\mathbf{G}(\mathbf{w}) = \begin{pmatrix} 0 \\ \rho\mathbf{g} \\ \rho\mathbf{u}\cdot\mathbf{g} \end{pmatrix}, \tag{2.3}$$

where the vector $\mathbf{g}$ denotes the direction and magnitude of gravity.

This system of conservation laws, also known as the compressible model of gas dynamics, gathers the conservation of **mass**, **momentum** and **energy** respectively. This is a system of $d+2$ equations with $d+3$ variables, which means that an additional equation is needed to close the system. This closure is achieved with an extra expression for the total energy, which is the sum of the internal pressure and the kinetic energy due to the gas momentum, given by

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho\mathbf{u}\cdot\mathbf{u}\right) \tag{2.4}$$

where $\gamma = 1.4$ is the adiabatic constant dependent on the type of gas.

These equations neglect the effect of viscosity and thermal diffusion. Then, for this problem the admissible set of solutions $\mathbb{U}_{ad}$ is defined as $\mathbb{U}_{ad} = \{\mathbf{w} > \mathbb{R}^4 : \rho > 0; p(\mathbf{w}) > 0\}$. Writing equation (2.1) in quasi linear form as in (1.3), the Jacobian matrices $\mathbf{A}_1 = \mathbf{f}_1'(\mathbf{w})$ and $\mathbf{A}_2 = \mathbf{f}_2'(\mathbf{w})$ can be computed in terms of the conserved variables. The eigenvalues of these matrices are respectively

$$\begin{aligned} \lambda_1 &= u - c, \quad & \lambda_2 &= u, \quad & \lambda_3 &= u + c \\ \lambda_1 &= v - c, \quad & \lambda_2 &= v, \quad & \lambda_3 &= v + c \end{aligned} \tag{2.5}$$

Here, c is the so called speed of sound which is given by $c = \sqrt{\frac{\gamma p}{\rho}}$ and the values $u = \frac{\rho u_1}{\rho}$ and $v = \frac{\rho u_2}{\rho}$ are respectively the speeds of the fluid in each space dimension.

2.2 High-order discontinuous Galerkin discretization

In order to compute approximate solutions, it's first necessary to generate a grid, i.e. a discrete representation of the geometric domain D over which the problem has to be solved. The grid divides the domain into a finite number of subdomains (tessellation T over the set $\bar{D}$). Grids can be constructed in two different ways: *structured* and *unstructured* grids. In the following, only structured grids will be considered. This triangulation typically has the following characteristics

- $\bar{D} = \cup_{l=1}^{N} \tau_l$,

- each $\tau \in T$ is a closed set and the interior of $\mathring{\tau}$ is non-empty,
- $\mathring{\tau}_i \cap \mathring{\tau}_j = \emptyset, \ \forall i \neq j, \ \tau_i, \tau_j \in T.$
- for each $\tau \in T$ the boundary $\partial\tau$ is Lipschitz-continuous.

The solution $\mathbf{w}(t, \mathbf{x})$ to the PDE in the whole domain D can be written as the sum of the local solution from each cell $\tau_l \subset D$ of the triangulation T. That is

$$\mathbf{w}(t, \mathbf{x}) = \sum_{l=0}^{N} \mathbf{w}_l(t, \mathbf{x})\chi_l(\mathbf{x}), \tag{2.6}$$

where $\mathbf{w}_l$ is the solution in the cell τ_l and $\chi_l(\mathbf{x})$ is the indicator function of the form

$$\chi_l(\mathbf{x}) = \begin{cases} 1 & if \ \ \mathbf{x} \in \tau_l, \\ 0 & otherwise \end{cases}$$

For spatial discretization, a high-order discontinuous Galerkin (DG) discretization is used, using a solution expansion of the form

$$\mathbf{w}_h(\mathbf{x}, t) = \sum_{j=1}^{n_{\mathbf{dofs}}} \boldsymbol{\varphi}_j(\mathbf{x}) w_j(t). \tag{2.7}$$

Here, $\boldsymbol{\varphi}_j$ denotes the jth basis function, written in vector form with separate shape functions for the different components and letting $w_j(t)$ go through the density, momentum, and energy variables, respectively. In this form, the space dependence is contained in the shape functions and the time dependence in the unknown coefficients w_j. As opposed to the continuous finite element method where some shape functions span across element boundaries, the shape functions are local to a single element in DG methods, with a discontinuity from one element to the next. The connection of the solution from one cell to its neighbors is instead imposed by the numerical fluxes specified below.

For the derivation of the DG formulation, the Euler equations must be multiplied by a test functions v and integrated over an individual cell K, which gives

$$\left(\mathbf{v}, \frac{\partial \mathbf{w}}{\partial t}\right)_K + (\mathbf{v}, \nabla \cdot \mathbf{F}(\mathbf{w}))_K = (\mathbf{v}, \mathbf{G}(\mathbf{w}))_K. \tag{2.8}$$

Then, the second term is integrated by parts, moving the divergence from the solution slot to the test function slot, and produced an integral over the element boundary:

$$\left(\mathbf{v}, \frac{\partial \mathbf{w}}{\partial t}\right)_K - (\nabla \mathbf{v}, \mathbf{F}(\mathbf{w}))_K + \left\langle \mathbf{v}, \mathbf{n} \cdot \widehat{\mathbf{F}}(\mathbf{w}) \right\rangle_{\partial K} = (\mathbf{v}, \mathbf{G}(\mathbf{w}))_K. \tag{2.9}$$

In the surface integral, the term $\mathbf{F}(\mathbf{w})$ is replaced by the term $\widehat{\mathbf{F}}(\mathbf{w})$, the **numerical flux**. The role of the numerical flux is to connect the solution on neighboring elements and weakly impose continuity of the solution. This ensures that the global coupling of the PDE is reflected in the discretization, despite independent basis functions on the cells. The connectivity to the neighbor is included by defining the numerical flux as a function $\widehat{\mathbf{F}}(\mathbf{w}^-, \mathbf{w}^+)$ of the solution from both sides of an interior face, $\mathbf{w}^-$ and $\mathbf{w}^-$. A basic property is required, namely that the numerical flux needs to be conservative. That is, all information (i.e., mass, momentum, and energy) must leave a cell over a face to enter the neighboring cell in its entirety and vice versa. This can be expressed as $\widehat{\mathbf{F}}(\mathbf{w}^-, \mathbf{w}^+) = \widehat{\mathbf{F}}(\mathbf{w}^+, \mathbf{w}^-)$, meaning that the numerical flux evaluates to the same result from either side. Combined with the fact that the numerical flux is multiplied by the unit outer normal vector on the face under consideration, which points in opposite direction from the two sides, the conservation is fulfilled. In the next chapter, different numerical fluxes will be introduced.

2.2.1 Boundary Conditions

In every computations, it has to ensure that the boundary conditions are provided so as to obtain accurate solutions. The boundary conditions are implemented in the program using ghost cells. This means that it assumed that there are cells just outside the actual computational domain and the boundary conditions are provided for these cells. This is done by using the boundary conditions for finding the cell interface fluxes. The edges at the boundary can be determined by checking the values for its left cell and right cell. The interface flux at the given edge is determined by using the flux terms and conserved variables at the left and right cells of that edge. If a given edge has got its left cell or right cell as a boundary, then the arguments fed for calculating the interface flux corresponding to is left or right cell, whichever is at the boundary, will be the specified boundary conditions. The discontinuous Galerkin method imposes boundary conditions not as constraints, but only weakly. Thus, the various conditions are imposed by finding an appropriate exterior quantity $\mathbf{w}^+$ that is then handed to the numerical flux function also used for the interior faces. Depending on the boundary condition, the outer trace $\mathbf{w}^+$ must be specified differently:

- Inflow boundary: all components are prescribed. $\mathbf{w}^+ = \begin{pmatrix} \rho_\mathrm{D}(t) \\ (\rho\mathbf{u})_\mathrm{D}(t) \\ E_\mathrm{D}(t) \end{pmatrix}$ (Dirichlet)
- Subsonic outflow boundary: $\mathbf{w}^+ = \mathbf{w}^-$ except that the energy variable is mod-

ified to support a prescribed pressure p_o, i.e. $\mathbf{w}^- = (\rho^+, \rho u_1^+, \ldots, \rho u_d^+, \frac{p_o}{(\gamma-1)} + \frac{1}{2}\rho|\mathbf{u}^+|^2)$

- Supersonic outflow boundary: $\mathbf{w}^+ = \mathbf{w}^- = \begin{pmatrix} \rho^- \\ (\rho\mathbf{u})^- \\ E^- \end{pmatrix}$ (Neumann)
- Wall boundary : set a no-normal-flux condition on the momentum variable, whereas a Neumann condition for the density and energy with $\rho^+ = \rho^-$ and $E^+ = E^-$. To achieve the no-normal flux condition, set

$$\mathbf{w}^+ = \begin{pmatrix} \rho^- \\ (\rho\mathbf{u})^- - 2[(\rho\mathbf{u})^- \cdot \mathbf{n}]\mathbf{n} \\ E^- \end{pmatrix}. \tag{2.10}$$

2.3 Strong Stability-Preserving High-Order Time Discretization Methods

To discretize in time, the weak form is rearranged and sum over all cells:

$$\sum_{K\in\mathcal{T}_h} \left(\boldsymbol{\varphi}_i, \frac{\partial \mathbf{w}}{\partial t}\right)_K = \sum_{K\in\mathcal{T}_h} \left[(\nabla\boldsymbol{\varphi}_i, \mathbf{F}(\mathbf{w}))_K - \left\langle \boldsymbol{\varphi}_i, \mathbf{n}\cdot\widehat{\mathbf{F}}(\mathbf{w})\right\rangle_{\partial K} + (\boldsymbol{\varphi}_i, \mathbf{G}(\mathbf{w}))_K\right], \tag{2.11}$$

through all basis functions with from 1 to n_{dofs}. Following the Galerkin approach, the test functions of $\mathbf{w}$ are taken from the same space of basis functions, thus equation 2.11 can be re-written, introducing **mass matrix** $\mathcal{M}$ with entries $\mathcal{M}_{ij} = \sum_K (\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_j)_K$, and

$$\mathcal{L}_h(t, \mathbf{w}_h) = [\sum_{K\in\mathcal{T}_h} [(\nabla\boldsymbol{\varphi}_i, \mathbf{F}(\mathbf{w}_h))_K - \left\langle \boldsymbol{\varphi}_i, \mathbf{n}\cdot\widehat{\mathbf{F}}(\mathbf{w}_h)\right\rangle_{\partial K} + (\boldsymbol{\varphi}_i, \mathbf{G}(\mathbf{w}_h)t)_K]]_{i=1,\ldots,n_{dofs}}. \tag{2.12}$$

the operator evaluating the right-hand side of the Euler operator, given a function $\mathbf{w}_h$ associated with a global vector of unknowns and the finite element in use. This function $\mathcal{L}_h$ is explicitly time-dependent as the numerical flux evaluated at the boundary will involve time-dependent data ρ_{D}, $(\rho\mathbf{u})_{\mathrm{D}}$, and $E_{\mathbf{D}}$ on some parts of the boundary, depending on the assignment of boundary conditions. With this notation, the discrete in space is written as

$$\mathcal{M}\frac{\partial \mathbf{w}_h}{\partial t} = \mathcal{L}_h(t, \mathbf{w}_h). \tag{2.13}$$

Equivalently, the system above has the form

$$\frac{\partial \mathbf{w}_h}{\partial t} = \mathcal{M}^{-1}\mathcal{L}_h(t, \mathbf{w}_h). \tag{2.14}$$

For hyperbolic systems discretized by high-order discontinuous Galerkin methods, explicit time integration is due to the fact that the mass matrix $\mathcal{M}$ is block-diagonal and thus easily inverted. Furthermore, since there is no continuity condition, the mass matrices are local operators for every cell, which makes them independent from the neighboring cells. Moreover, a single mass matrix can be defined in the reference domain. This means that the mass matrix is then unique when the polynomials basis have the same degree over the whole domain, like in our code..

Concerning the time discretization, a class of high-order strong stability time discretization methods is introduced. A relevant question concerns stability, because for problems with smooth solutions, usually a linear stability analysis is adequate, but, for problems with discontinuous solutions, however, a stronger measure of stability is usually required. For hyperbolic PDEs, the relevant nonlinear stability property typically takes the form of total variation diminishing (TVD). The idea is to assume that the first-order forward Euler time discretization of the method of lines ODE is strongly stable under a certain norm when the time step Δt is suitable restricted, and then to try to find a higher order time discretization (Runge-Kutta) that maintains strong stability for the same norm, perhaps under a different time step restriction. In other words, given a convex functional $\|\cdot\|$ it is assumed that there exists a value Δt_{FE} such that

$$\|w^n + \Delta t \mathcal{L}(w^n)\| \leq \|w^n\| \quad for\ 0 \leq \Delta t \leq \Delta t_{FE}. \tag{2.15}$$

A s-step numerical method computes the next solution value w^{n+1} from previous values $w^{n-s+1}, ..., w^n$. The method is **Strong Stability Preserving (SSP)** with SSP coefficient c, if it holds that

$$\|w^{n+1}\| \leq \max\{\|w^n\|, \|w^{n-1}\|, ..., \|w^{n-s+1}\|\}, \tag{2.16}$$

whenever (2.15) holds for timestep satisfies

$$\Delta t \leq c\Delta t_{FE} \quad for\ some\ c < 0. \tag{2.17}$$

Thus, the objective of the high-order SSP Runge-Kutta time discretization of maintaining strong stability TVD property $TV(w^{n+1}) \leq TV(w^n)$, $TV(u^n) = \sum_j (|u^n_{j+1} - u^n_j|$ is satisfied.

2.3.1 SSP Runge-Kutta methods

A general m-stage Runge-Kutta method is written in the form

1. Set $w_h^{(0)} = w_h^n$,

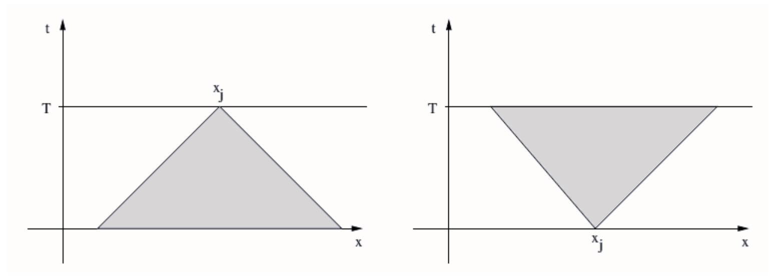
a)  b)

Figure 2.1: Illustration (1D) of the finite speed of propagation of information in conservation laws. a) Domain of dependence; b) Range of influence.

2. for $i = 1, ..., m$,compute the intermediate functions

$$w_h^{(i)} = \sum_{k=0}^{i-1} \left(\alpha_{i,k} w_h^{(k)} + \Delta t \beta_{i,k} \mathcal{L}(\mathbf{w}_h^{(k)}) \right), \tag{2.18}$$

3. set $w_h^{(n+1)} = u_h^m$.

Clearly, if $\beta_{i,k} \neq 0$'s and $\alpha_{i,k} \geq 0$, then since by consistency $\sum_{k=0}^{i-1} \alpha_{i,k} = 1$, it follow that the intermediate stages in 2.18, amount to convex combinations of the forward Euler operators, with Δt replaced by $\frac{\beta_{i,k}}{\alpha_{i,k}} \Delta t$.

Lemma 2.3.1 *If the forward Euler method $w^{n+1} = w^n + \Delta t \mathcal{L}(w^n)$ is strongly stable under the CFL restriction $\Delta t \leq \Delta t_{FE}$, $\|w^n + \Delta t \mathcal{L}(w^n)\| \leq \|w^n\|$, then the Runge-Kutta method* (2.18) *with $\beta_{i,k} \geq 0$ is SPP, $\|w^{n+1}\| \leq \|w^n\|$, provided the following CFL restriction* (2.17) *is fulfilled:*

$$\Delta t \leq c \Delta t_{FE}, \ c = \min_{i,k} \frac{\alpha_{i,k}}{\beta_{i,k}}. \tag{2.19}$$

2.3.2 Courant-Friedrichs-Lewy (CFL) condition

One of the properties of the conservation laws PDE is its finite speed of propagation of information, and this speed is given by the eigenvalues. Thus, the solution in every point of the domain will have a domain of dependence (Figure 2.1a) at previous times and a range of influence in the future (Figure 2.1b). A condition for the stability of the numerical scheme emerges, since it has to work faster than the speed of propagation of information. This translates to a restriction that has to be imposed to the time step size, which is called the **Courant-Friedrichs-Levy condition**. For discontinuous Galerkin methods, the time step is set as follows

$$\Delta t = \frac{\mathrm{Cr}}{p^{1.5}} \left(\frac{1}{\max \left[\frac{\|\mathbf{u}\|}{h_u} + \frac{c}{h_c} \right]} \right) \tag{2.20}$$

The dimensionless number Cr denotes the Courant number and can be chosen up to a maximally stable number $\mathrm{Cr}_{\max}$, whose value depends on the selected time stepping method and its stability properties. The power $p^{1.5}$ used for the polynomial scaling is heuristic and represents the closest fit for polynomial degrees between 1 and 8. Regarding the effective mesh sizes h_u and h_c used in the formula, since the convective transport is directional, thus an appropriate scaling is to use the element length in the direction of the velocity u. The code below derives this scaling from the inverse of the Jacobian from the reference to real cell, i.e., approximating $\frac{\|u\|}{\|h_u\|} \approx \|J^{-1}u\|_\infty$. The acoustic waves, instead, are isotropic in character, which is why the smallest feature size is used, represented by the smallest singular value of J, for the acoustic scaling h_c Finally, it is need to add the convective and acoustic limits, as the Euler equations can transport information with speed $\|u\| + c$.

2.3.3 SSP Runge-Kutta methods with Optimal CFL condition

A class of optimal SSP Runge-Kutta methods of any order for the ODE is introduced.

Lemma 2.3.2 *Consider the family m-stage, mth-order SSP Runge-Kutta methods* (2.18) *with nonnegative coefficients $\alpha_{i,k}$ and $\beta_{i,k}$. The maximum CFL restriction attainable for such methods is the one dictated by the forward Euler scheme,*

$$\Delta t \leq \Delta t_{FE}$$

i.e, (2.17) *holds with maximal CFL coefficient $c = 1$.*

Second order and third order optimal SSP Runge-Kutta methods that have $c = 1$:
SSPRK(2, 2) :

$$\begin{aligned} \mathbf{w}^1 &= \mathbf{w}^n + \Delta t \mathcal{L} \mathbf{w}^n), \\ \mathbf{w}^{n+1} &= \frac{1}{2}\mathbf{w}^n + \frac{1}{2}\mathbf{w}^1 + \frac{1}{2}\Delta t \mathcal{L}(\mathbf{w}^1). \end{aligned} \tag{2.21}$$

SSPRK(3, 3) :

$$\begin{aligned} \mathbf{w}^1 &= \mathbf{w}^n + \Delta t \mathcal{L}\left(\mathbf{w}^n\right), \\ \mathbf{w}^2 &= \frac{3}{4}\mathbf{w}^n + \frac{1}{4}\mathbf{w}^1 + \frac{1}{4}\Delta t \mathcal{L}\left(\mathbf{w}^1\right), \\ \mathbf{w}^{n+1} &= \frac{1}{3}\mathbf{w}^n + \frac{2}{3}\mathbf{w}^2 + \frac{2}{3}\Delta t \mathcal{L}\left(\mathbf{w}^2\right). \end{aligned} \tag{2.22}$$

Chapter 3

Numerical Fluxes

Introduction

In this chapter, some of the most important schemes for the evaluation of the numerical flux are introduced. Computing the exact solution for the Riemann problem at discontinuities is extremely costly for the method because iterative schemes are necessary since it does not exist an analytical solution. For this reason, during the years have been proposed a number of methods with the aim of approximate solutions of the Riemann problem.
A first important classification distinguishes upwind fluxes from centered ones. Examples of upwind schemes are the Rusanov flux, the HLL flux of Harten *et al.* [25] and the HLLC flux of Toro *et al.* [25]. These types of schemes uses information derived from wave propagation and in general perform much better in comparison with centered fluxes. On the other hand, centered schemes like the Lax-Friedrichs flux [25] don't exploit propagation information, resulting more simple and less costly. The objective of each of these schemes is to retrieve a numerical flux $\hat{\mathbf{F}}(\mathbf{w}(x_{j\pm1/2}, t))$ at the boundary points $x_{j\pm1/2}$, using the computed solutions and fluxes on the two adjacent elements. The subscript L means that the quantity is evaluated on the element at the left side of the discontinuity, while subscript R indicates the right element.
Seeking an accurate numerical scheme for capturing shock and contact discontinuities, with minimal numerica dissipation and oscillations, has been a lasting challenge to the computational fluid dynamics. Today, upwind schemes undoubtedly gave become the main spatial discretization techniques adopted in nearly all major codes. Thus, a new breed of upwind schemes have been produced since the beginning of the 90s. The research is motivated by the desire to achieve both the efficiency of the flux-vector splitting and the accuracy of the flux-difference splitting. Shock-capturing schemes were originally developed for high Mach number regimes (supersonic and

hypersonic flows) and are generally upwind biased schemes. Among the many shock-capturing schemes, the advection upstream splitting method (AUSM) developed by Liou and Steffen [17], and its variant AUSM-family schemes [18] are very simple, but are accurate and robust for high Mach number flows. Shima and Kitamura also made an alternative modification of the AUSM-family schemes and developed a new all-speed scheme, named the simple low-dissipation AUSM (SLAU) scheme [19]. SLAU is free from problem-dependent parameters and its algorithm is kept simple. SLAU has demonstrated these advantageous characteristics in many test problems from low to supersonic and hypersonic flow speeds. Thus, the all-speed AUSM-family schemes are expected to be extended to high-order spatial schemes, and of course they have the capability to solve flows at low to high Mach number regimes.

3.1 Lax-Friedrichs numerical flux:

$$\boldsymbol{F}^{lf}_{j+1/2} = \frac{1}{2}\left[\boldsymbol{F}(\mathbf{w}_L) + \boldsymbol{F}(\mathbf{w}_R) - C(\mathbf{w}_R - \mathbf{w}_L)\right], \qquad C = max\{|\lambda_j|\} \tag{3.1}$$

where the numerical viscosity constant C is an estimate of the biggest eigenvalues of the Jacobian matrix defined in (1.13)

3.2 HLL Riemann solver:

The HLL Riemann solver, proposed by Harten Lax and Van Leer in 1983 [22], is based on the approximation of the inter-element numerical flux using estimates for the slowest and fastest signal velocities. The main idea is to assume, for the solution, a wave configuration that consists of two waves separating three constant states.

$$\tilde{\mathbf{w}}(x,t) = \begin{cases} \mathbf{w}_L, & if \;\; x/t < S_L, \\ \mathbf{w}^{hll} & if \;\; S_L \le x/t \le S_R, \\ \mathbf{w}_R, & if \;\; x/t \ge S_R, \end{cases} \quad \Rightarrow \boldsymbol{F}^{hll}_{j+\frac{1}{2}} = \begin{cases} \boldsymbol{F}_L, & if \;\; 0 < S_L, \\ \boldsymbol{F}^{hll} & if \;\; S_L \le 0 \le S_R, \\ \boldsymbol{F}_R, & if \;\; 0 \ge S_R, \end{cases} \tag{3.2}$$

where S_L and S_R are the signal speed and $\mathbf{w}^{hll}$ is the constant state vector given by

$$\mathbf{w}^{hll} = \frac{S_R\mathbf{w}_R - S_L\mathbf{w}_L + F_L - F_R}{S_R - S_L} \tag{3.3}$$

while the corresponding flux $\boldsymbol{F}^{hll}$ is given by

$$\boldsymbol{F}^{hll} = \frac{S_R\boldsymbol{F}_L - S_L\boldsymbol{F}_R + S_L S_R(\mathbf{w}_R - \mathbf{w}_L)}{S_R - S_L} \tag{3.4}$$

Note that, in this case the *star region* consists of a single constant state, thus all intermediate states separated by intermediate waves are lumped into the single state $\mathbf{w}^{hll}$. To close the method one should choose how to approximate the speeds S_L and S_R.

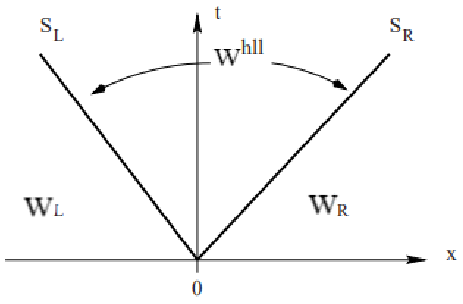


Figure 3.1: Approximate HLL Riemann solver. Solution in the Star Region consists w^{hll} of a single state separated from data states by two waves of speeds S_L and S_R.

3.3 HLLC Riemann Solver:

The HLLC is a modification of the HLL Riemann solver, proposed by Toro, Spruce and Speares in 1992 [12]. It's a three-wave model, resulting in two-star states for the intermediate region of the Riemann-problem solution. In addition to the slowest and fastest signal speeds S_L and S_R it includes a middle wave of speed S^*.
The proposed HLLC approximate Riemann solver is the following:

$$\tilde{\mathbf{w}}(x,t) = \begin{cases} \mathbf{w}_L, & if \ \ x/t < S_L, \\ \mathbf{w}_L^* & if \ \ S_L \le x/t \le S^*, \\ \mathbf{w}_R^* & if \ \ S^* \le x/t \le S_R, \\ \mathbf{w}_R, & if \ \ x/t \ge S_R, \end{cases} \quad \Rightarrow \boldsymbol{F}_{j+\frac{1}{2}}^{hllc} = \begin{cases} \boldsymbol{F}_L, & if \ \ 0 < S_L, \\ \boldsymbol{F}_L^* & if \ \ S_L \le 0 \le S^*, \\ \boldsymbol{F}_R^* & if \ \ S^* \le 0 \le S_R, \\ \boldsymbol{F}_R, & if \ \ 0 \ge S_R, \end{cases} \tag{3.5}$$

where

$$\mathbf{w}_L^* = \frac{1}{T(S^* - S_L)} \int_{TS_L}^{TS^*} \mathbf{w}(x,T)dx$$

$$\mathbf{w}_R^* = \frac{1}{T(S_R - S^*)} \int_{TS^*}^{TS_R} \mathbf{w}(x,T)dx$$

and the intermediate fluxes $\boldsymbol{F}_L^*$ and $\boldsymbol{F}_R^*$ are

$$\boldsymbol{F}_K^* = \boldsymbol{F}_K - S_K(\mathbf{w}_K^* - \mathbf{w}_K), \qquad K = L, R \tag{3.6}$$

In the case of Euler equations, there are more unknowns than equations and some

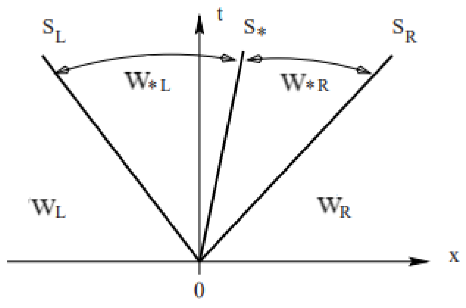


Figure 3.2: HLLC approximate Riemann solver. Solution in the Star Region consists of two constant states separated from each other by a middle wave of speed S^*.

extra conditions need to be imposed. Obvious conditions to impose are those satisfied by the exact solution; for pressure, normal and tangential component of velocity it imposes

$$\begin{aligned} p_L^* = p_R^* = p^*, \\ u_{1,L}^* = u_{1,L}^* = u^* \\ u_{2,L}^* = u_{2,L} \quad u_{2,R}^* = u_{2,R} \end{aligned} \tag{3.7}$$

Thus, the speed S^* is

$$S^* = \frac{p_R - p_L + \rho_L u_{1,L}(S_L - u_{1,L}) - \rho_R u_{1,R}(S_R - u_{1,R})}{\rho_L(S_L - u_{1,L}) - \rho_R(S_R - u_{1,R})} \tag{3.8}$$

and the intermediate fluxes $\boldsymbol{F}_L^*$ and $\boldsymbol{F}_R^*$ are given by (3.6), with the intermediate states given as

$$w_K^* = \rho_K \left(\frac{S_K - u_{1,K}}{S_K - S^*} \right) \begin{bmatrix} 1 \\ S^* \\ u_{2,K} \\ \frac{E_K}{\rho_K} + (S^* - u_{1,K}) \left[S^* + \frac{p_K}{\rho_K(S_K - u_{1,K})} \right] \end{bmatrix} \tag{3.9}$$

3.3.1 Direct wave speed estimates:

Davis and Einfeldt [25] proposed an original idea to approximate S_L and S_R, in particular, they proposed to use the Roe average eigenvalues for the left and right non-linear waves, that is

$$S_L = \tilde{u} - \tilde{a}, \qquad S_R = \tilde{u} + \tilde{a}, \tag{3.10}$$

where $\tilde{u}$ and $\tilde{a}$ are the Roe-average particle and sound speeds respectively, given as follow

$$\tilde{u} = \frac{\sqrt{\rho_L} u_{1,L} + \sqrt{\rho_R} u_{1,R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \qquad \tilde{a} = \sqrt{(\gamma - 1)(\tilde{H} - \frac{1}{2}\tilde{u}^2)},$$

with the enthalpy $H = (E + p)/\rho$ approximated as

$$\tilde{H} = \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

3.4 The Riemann solver of Roe

The Riemann Solver of Roe is the most well-know of all approximate Riemann solvers, which was first presented in the literature in 1981 [22]. The purpose is find direct approximations to the flux function $\mathbf{F}_{i+\frac{1}{2}}$, solving the Riemann problem approximately. From the definition of the Jacobian matrix in (1.13) and using the chain rule, the Riemann problem may be written as

$$\mathbf{w}_t + \mathbf{A}(\mathbf{w})\mathbf{w}_x = \mathbf{0}. \tag{3.11}$$

Roe's approach replaces the Jacobian matrix $\mathbf{A}(\mathbf{w})$ by a constant Jacobian matrix

$$\tilde{\mathbf{A}} = \tilde{\mathbf{A}}(\mathbf{w_L}, \mathbf{w_R}), \tag{3.12}$$

which is a function of the data states $\mathbf{w}_L$, $\mathbf{w}_R$. In this way the original PDEs are replaced by

$$\mathbf{w}_t + \tilde{\mathbf{A}}\mathbf{w}_x = \mathbf{0} \tag{3.13}$$

This is a linear system with constant coefficients. The approximate problem results from replacing the original non-linear conservation laws by a linearised system with constant coefficient but the initial data of the exact problem is conserved. The Roe Jacobian matrix $\tilde{\mathbf{A}}$ is required to satisfy the following properties:

- Hyperbolicity of the system. $\tilde{\mathbf{A}}$ is required to have real eigenvalues $\tilde{\lambda}_i = \tilde{\lambda}_i(\mathbf{w}_L, \mathbf{w}_R)$, which are chosen to order as $\tilde{\lambda}_1 \leq \tilde{\lambda}_2 \leq .... \leq \tilde{\lambda}_m$, and a complete set of linearly independent right eigenvectors $\tilde{\mathbf{K}}^{(1)}$, $\tilde{\mathbf{K}}^{(2)}$, $\tilde{\mathbf{K}}^{(3)}$

- Consistency with the exact Jacobian $\tilde{\mathbf{A}}(\mathbf{w}, \mathbf{w}) = \mathbf{A}(\mathbf{w})$.

- Conservation across discontinuities $\mathbf{F}(\mathbf{w}_R) - \mathbf{F}(\mathbf{w})_L = \tilde{\mathbf{A}}(\mathbf{w}_R - \mathbf{w}_L)$.

Once the matrix $\tilde{\mathbf{A}}$, its eigenvalues and right eigenvectors are available, once solves the Riemann problem. The definition of the flux is

$$\mathbf{F}^{ROE}_{i+\frac{1}{2}} = \frac{1}{2}(\mathbf{F}_L + \mathbf{F}_R) + \frac{1}{2}\sum_{i=1}^{m} \tilde{\alpha}_i |\tilde{\lambda}_i| \tilde{\mathbf{K}}^{(i)} \tag{3.14}$$

where $\tilde{\alpha}_i = \tilde{\alpha}_i(\mathbf{w}_L, \mathbf{w}_R)$ are the wave strengths, such that $\Delta\mathbf{w} = \mathbf{w}_R - \mathbf{w}_L = \sum_{i=1}^{m} \tilde{\alpha}_i \tilde{\mathbf{K}}^{(i)}$. Notice that it is made up of a central average plus the influence of the upwind waves, which adds dissipation. The Roe Riemann solvers applied to Euler equation, using (3.3.1) and (3.3.1), are

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ (\gamma-1)\tilde{H} - \tilde{u}^2 - \tilde{a}^2 & (3-\gamma)\tilde{u} - (\gamma-1)\tilde{v} & -(\gamma-1)\tilde{v} & \gamma - 1 \\ -\tilde{u}\tilde{v} & \tilde{v} & \tilde{u} & 0 \\ \frac{1}{2}\tilde{u}[(\gamma-3)\tilde{H} - \tilde{a}^2] & \tilde{H} - (\gamma-1)\tilde{u}^2 & -(\gamma-1)\tilde{u}\tilde{v} & \gamma\tilde{u} \end{bmatrix} \tag{3.15}$$

$$\tilde{\lambda_1} = \tilde{u} - \tilde{a}, \quad \tilde{\lambda_2} = \tilde{\lambda_3} = \tilde{u}, \quad \tilde{\lambda_4} = \tilde{u} + \tilde{a} \tag{3.16}$$

$$\mathbf{K}^{(1)} = \begin{bmatrix} 1 \\ \tilde{u} - \tilde{a} \\ \tilde{v} \\ \tilde{H} - \tilde{u}\tilde{a}; \end{bmatrix} \quad \mathbf{K}^{(2)} = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \frac{1}{2}\tilde{V}^2; \end{bmatrix} \quad \mathbf{K}^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \tilde{v}; \end{bmatrix} \quad \mathbf{K}^{(4)} = \begin{bmatrix} 1 \\ \tilde{u} + \tilde{a} \\ \tilde{v} \\ \tilde{H} + \tilde{u}\tilde{a}; \end{bmatrix} \tag{3.17}$$

$$\begin{aligned} \tilde{\alpha_1} &= \frac{1}{2\tilde{a}}(\Delta p - \tilde{a}\tilde{\rho}\Delta u) \\ \tilde{\alpha_2} = \tilde{\alpha_3} &= \Delta\rho - \frac{1}{\tilde{a}^2}\Delta p \\ \tilde{\alpha_4} &= \frac{1}{2\tilde{a}^2}(\Delta p + \tilde{a}\tilde{\rho}\Delta u) \end{aligned} \tag{3.18}$$

where $\tilde{\rho} = \sqrt{\rho_L \rho_R}$, $\Delta p = p_R - p_L$ and $\Delta u = u_R - u_L$.

Entropy fix When $|\tilde{u}| \approx \tilde{a}$, the Roe solver would give the wrong answer. This is a result of linearizing the problem and it can be shown that the cause is an expansion solution that violates the second law of thermodynamics. To fix this, it is common to fudge the eigenvalues $\tilde{\lambda}_1$ or $\tilde{\lambda}_3$ by

$$\tilde{\lambda_p} \to \frac{1}{2}\left(\frac{\tilde{\lambda_p}^2}{\epsilon} + \epsilon\right), \quad \epsilon << 1 \tag{3.19}$$

according to whether $|\lambda_1| < \epsilon$ or $|\lambda_3| < \epsilon$. This is called an "entropy fix." It is necessary to implement it for the Roe scheme to be "well behaved." The value of ϵ is arbitrary so long as it is small relative to unity.

3.5 AUSM-Type scheme

Another schemes, proposed by Liou and Steffen in [18], is introduced. It is called Advection Upstream Splitting Method (AUSM). The main idea is to split the numerical flux $\boldsymbol{F}^{AUSM}$ into a convective term $\boldsymbol{F}^{(c)}$ and a pressure term $\boldsymbol{F}^{(p)}$ so that at a continuum level it's found:

$$\boldsymbol{F}^{AUSM} = \boldsymbol{F}^{(c)} + \boldsymbol{F}^{(p)} = \dot{m}\boldsymbol{\Psi} + \boldsymbol{F}^{(p)} = \dot{m}\boldsymbol{\Psi} + \begin{bmatrix} 0 \\ p\mathbf{n_x} \\ p\mathbf{n_y} \\ 0 \end{bmatrix} \tag{3.20}$$

where $\mathbf{\Psi}$ is a vector quantity that represents the problem variables, $\dot{m}$ is the mass flux expressed as:

$$\dot{m} = \rho \mathbf{u} \cdot \mathbf{n} \qquad \mathbf{\Psi} = \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ H \end{bmatrix} \tag{3.21}$$

The discretization of the component of the numerical flux normal depends on the left and right state vectors $\mathbf{w}_L$ and $\mathbf{w}_R$ and it can be defined as:

$$\mathbf{F}_{1/2}(\mathbf{w}_L, \mathbf{w}_R, \mathbf{n}) = \dot{m}_{1/2}\mathbf{\Psi}_{L/R} + \mathbf{p}_{1/2} \tag{3.22}$$

where $\mathbf{\Psi}_{L/R}$ will be determined as

$$\mathbf{\Psi}_{L/R} = \begin{cases} \mathbf{\Psi}_L, & if \dot{m}_{1/2} > 0 \\ \mathbf{\Psi}_R, & otherwise \end{cases} \tag{3.23}$$

Different choice for $\dot{m}_{1/2}$ and $\mathbf{p}_{1/2}$ determine different schemes.

The mass flux scalar term $\dot{m}_{1/2}$ is a function of the interface Mach number M, the left and right neighboring cells density ρ_L, ρ_R and the interface sound speed $a_{1/2}$:

$$\dot{m}_{1/2} = M_{1/2} a_{1/2} \begin{cases} \rho_L, & if M_{1/2} > 0 \\ \rho_R, & otherwise \end{cases} \tag{3.24}$$

where $M_{1/2}$ is a polynomial function of the left and right neighboring cells Mach numbers $\mathcal{M}_L$ and $\mathcal{M}_R$. The AUSM method defines:

$$M_{1/2}^{AUSM} = \mathcal{M}^+(M_L) + \mathcal{M}^-(M_R) \tag{3.25}$$

where the split Mach number polynomial $\mathcal{M}^\pm$ reads:

$$\mathcal{M}^\pm(M) = \begin{cases} \mathcal{M}^\pm_{(1)}(M), & if |M| > 1 \\ \mathcal{M}^\pm_{(2)}(M), & otherwise \end{cases} \tag{3.26}$$

with

$$\begin{aligned} \mathcal{M}^\pm_{(1)}(M) &= \frac{1}{2}(M \pm |M|) \\ \mathcal{M}^\pm_{(2)}(M) &= \pm\frac{1}{4}(M \pm 1)^2 \end{aligned} \tag{3.27}$$

and M_L and M_R are the normal left and right Mach numbers defined as:

$$\begin{aligned} M_L &= \frac{\mathbf{u_L} \cdot \mathbf{n}}{a_{1/2}} \\ M_R &= \frac{\mathbf{u_R} \cdot \mathbf{n}}{a_{1/2}} \end{aligned} \tag{3.28}$$

Let a_j^* be the critical speed of sound evaluated at $w_j(\mathbf{w}_L)$ and calculated via the isoenergetic condition for ideal gas $h_t = \frac{(\gamma+1)a^{*2}}{2(\gamma-1)}$, the the speed of sound used in the $AUSM$,

$$a_{1/2} = a(\mathbf{w}_L, \mathbf{w}_R) = a_j^*/w_j \tag{3.29}$$

allows an exact resolution of the stationary shock. The formula (3.29) is valid for $w_L > a_L^*$. It must be extended it to other conditions; it's suggested the following formula:

$$a_{j+1/2} = \min(\tilde{a}_L, \tilde{a}_R), \quad with\ \tilde{a} = a^{*2}/\max(a^*, |u|).$$

$$a_{j+1/2} = \frac{1}{2}(a_j + a_{j+1}) \tag{3.30}$$

$$a_{j+1/2} = \sqrt{a_j a_{j+1}}$$

The $AUMS$ method has been found [18] to have deficiencies in accuracy, efficiency and robustness. The pressure flux term for the $AUSM$ scheme it can been defined as:

$$\mathbf{p}_{1/2}^{AUSM} = \mathcal{P}^+(M_L)p_L\mathbf{n} + \mathcal{P}^-(M_R)p_R\mathbf{n} \tag{3.31}$$

where the split pressure polynomials $\mathcal{P}^\pm$ and $\mathcal{P}^\pm_{(5)}$ are given by:

$$\mathcal{P}^\pm(M) = \begin{cases} \frac{1}{M}\mathcal{M}^\pm_{(1)}(M), & if|M| > 1 \\ \pm\mathcal{M}^\pm_{(2)}(M)(2 \mp M), & otherwise \end{cases} \tag{3.32}$$

with $\frac{3}{16} \le \alpha \le \frac{1}{8}$.

With the aim of not having the mach number as a parameter and not depending on it, a simple numerical flux of AUSM-family for all speed with no tunable parameters, named $SLAU$ ($simple\ Low-dissipation\ AUSM$), is introduced. This scheme features low dissipation in low Mach number regime, while keeping robustness and non-oscillating nature at high Mach numbers, i.e., stabilities against shock-induced anomalies such as the carbuncle phenomenon. The one of the authors proposed SHUS (simple high-resolution upwind scheme) in which mass flux of the original AUSM was replaced by that of Roe scheme. They showed that SHUS cured a spurious overshoot at a shock front, so called "1D carbucle", appeared in the case of the original AUSM. The mass flux of SHUS is written as follows

$$\dot{m} = \frac{1}{2}\left\{(\rho U)^+ + (\rho U)^- - |\overline{U}|\Delta\rho - \frac{|\overline{M}+1| - |\overline{M}-1|}{2}\overline{\rho}\Delta V_n - \frac{|\overline{M}+1| + |\overline{M}-1| - 2|\overline{M}|}{2\overline{c}}\Delta p\right\} \tag{3.33}$$

This mass flux is appropriate as a starting point for development of new schemes. The mass flux given by (3.33) has been improved when combined with the pressure term given by (3.36). However, a mass flux contains pressure difference terms that tends to exhibit the carbuncle. Thus, this defect was expected to be cured by elimination of the pressure difference term Δp in (3.33). However, this modification turned out to be failure: the modified flux showed the overshoot at a shock front, again. It is probably due to cancellation of the dissipation effects of the third term ($\delta\rho$ term) by the forth term (ΔU term), considering the fact that these terms have opposite signs across the shock where the flow is compressed ($\delta\rho > 0$) and decelerated ($\delta U < 0$). In other words, the original form of the Eq. (3.33) created proper amount of dissipation in a direction normal to the shock, but not in its parallel direction. Having confirmed that the dissipation term of multi-dimensional character is preferred, it proceed to further modification of the Eq. (3.33) in preparation for the final form of the new scheme. If the density difference term is modified to be more dissipative, and the velocity difference term is also eliminated, an obviously very simple expression for mass flux is obtained as in the following Eqs. (3.34).

$$\dot{m}_{1/2} = \frac{1}{2}[(\rho U)_L + (\rho U)_R - |\overline{U}|(\rho_R - \rho_L)](1-g) - \frac{\chi}{2a_{1/2}}(p_R - p_L) \tag{3.34}$$

where

$$\begin{aligned} \chi &= (1-\hat{M})^2 \\ \hat{M} &= \min\left[1.0, \frac{1}{a_{1/2}}\sqrt{u_{1,L}^2 + u_{2,L}^2 + u_{1,R}^2 + u_{2,R}^2}\right] \\ |\overline{U}| &= \frac{\rho_L |U_L| + \rho_R |U_R|}{\rho_L + \rho_R} \\ g = -\max[\min(M_L, 0), -1]&\min[\max(M_R, 0), 1] \;\; \in [0,1] \end{aligned} \tag{3.35}$$

This mass flux does not have dependence on Mach numbers that most of all-speed schemes have. The pressure functions is given by

$$\begin{aligned} p_{1/2} = \frac{p_L + p_R}{2} &+ \frac{\mathcal{P}^+_{(5),\alpha=0}(M_L) + \mathcal{P}^-_{(5),\alpha=0}(M_R)}{2}(p_L - p_R) + \\ &+ (1-\chi)[\mathcal{P}^+_{(5),\alpha=0}(M_L) + \mathcal{P}^-_{(5),\alpha=0}(M_R) - 1]\frac{p_L + p_R}{2} \end{aligned} \tag{3.36}$$

where

$$\mathcal{P}^{\pm}_{(5)}(M) = \begin{cases} \frac{1}{M}\mathcal{M}^{\pm}_{(1)}(M), & if|M| > 1 \\ \pm\mathcal{M}^{\pm}_{(2)}(M)(2 \mp M) \mp \alpha M(M^2-1)^2, & otherwise \end{cases} \tag{3.37}$$

Chapter 4

Dealing with discontinuities

Introduction

When the solution develops discontinuities, high-order DG methods tend to oscillatory solutions, like all high-order methods when not using flux- or slope-limiters. In this chapter, first of all, two different limiting techniques (TVB limiter, which it uses of the indicator KXRCF and filtering monotonization approach) are introduced. Limiters reduce the spurious oscillations that arise in presence of discontinuities when high order spatial discretizations. After, how the DG method computes more accurate approximations of the solution through a process called h-refinement are introduced.

4.1 First procedure: High-Order and TVD methods for non-linear equation

Central to this section is the resolution of two contradictory requirements on numerical methods, namely high-order of accuracy and absence of spurious (unphysical) oscillations in the vicinity of large gradients. It is well-known that high order non-linear schemes produce unphysical oscillations in the vicinity of large gradients [25]. On the other hand, the class of monotone methods do not produce unphysical oscillations. However, monotone methods are at most first order accurate and are therefore of limited use. These difficulties are embodied in the statement of Godunov's theorem [15]. One way of resolving the contradiction is by constructing Total Variation Diminishing Method. An important class of methods for solving systems of non-linear convective dominated PDEs are the so called monotone schemes.

Definition 4.1.1 ***(Monotone Schemes.)*** *A scheme written in the form*

$$\mathbf{w}_i^{n+1} = \boldsymbol{T}(\mathbf{w}_{i-k_L+1}^n, ..., \mathbf{w}_{i+k_R}^N)$$

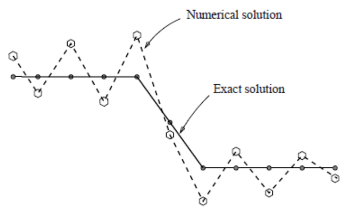


Figure 4.1: Illustration of the numerical phenomenon of spurious oscillations near high gradients.

where k_L and k_R are non negative integers, is said to be monotone if

$$\frac{\partial \boldsymbol{T}}{\partial \mathbf{w}_j^n} \geq 0, \quad \forall j$$

Monotonicity represents the discrete version of an important property of conservation laws: given two initial conditions $v_0(x)$ and $u_0(x)$, then

$$v_0(x) \geq u_0(x) \quad \forall x \quad \Rightarrow \quad v(x,t) \geq u(x,t) \quad \forall x, t > 0$$

For this reason, the usage of a monotone scheme for the discretization of the equations guarantees that no new extrema are created, so that spurious oscillations does not appear.
Unfortunately, the following well known result, due to Godunov [15], states that there exists a trade off between monotonicity and accuracy.

Theorem 4.1.1 ***(Godunov)*** *There are no monotone, linear schemes of second or higher order of accuracy for hyperbolic equations.*

Because of this impassable barrier, then, in order to develop high order monotone methods one has to construct numerical methods that have the following properties

- The schemes have second or higher order of accuracy in smooth parts of the solution
- The schemes produce numerical solutions free from spurious oscillations
- The schemes produce high–resolution of discontinuities, that is the number of mesh points in the transition zone containing the numerical wave is narrow in comparison with that of first–order monotone methods.

The real issue concerning numerical methods is convergence. For non–linear systems convergence proofs rely on non–linear stability, the theory of which relies on functional analysis concepts, such as compactness. Sets of functions whose total variation is bounded lead to compact sets. Total Variation Stable methods are then defined as those whose mesh–dependent approximations lie in compact function sets. It can then be proved that Total Variation Stable methods are convergent. See Harten [14] and LeVeque [16] for details. A subclass of Total Variation Stable methods are those whose total variation does not increase in time.

To overcoming Godunov's barrier, the total variation of a function $u(x,t)$ is introduced

$$TV(u(t)) = \lim_{h\to 0} \sup \frac{1}{h} \int_{-\infty}^{\infty} |u(x+h,t) - u(x,t)| dx \tag{4.1}$$

while the corresponding numerical version, if $u^n = \{u_i^n\}$ is a mesh function, is

$$TV(u^n) = \sum_{i=-\infty}^{\infty} |u_{i+1}^n - u_i^n| \tag{4.2}$$

Obviously, in order for $TV(u^n)$ to be finite, one must assume $u_i^n = 0$ or $u_i^n =$ *constant* as $i \Rightarrow \pm\infty$. Recalling that for the exact solution no new extrema can be created and that values of local minima (maxima) do not decrease (increase), one can observe that following definition.

Definition 4.1.2 ***(TVD Schemes)*** *A scheme characterized by a non-increasing total variation*

$$TV(u(t^m)) \leq TV(u(t^n)) \quad \forall t^m \geq t^n \tag{4.3}$$

is called Total Variation Diminishing (TVD).

Note that, since monotonicity implies (4.3), monotone schemes are a subset of the Total Variation Diminishing schemes. Generally, TVD schemes can be classified as either *flux limiters* or *slope limiters*. The strategy of both of these methods is to use limiters that enforce the TVD property modifying the computed numerical solution at each temporal step. In this project it investigated the concept of slope limiter. This approach does not modify the averages solution in the cells of the triangulation (so conservation property is respected), but it acts on the high order coefficients reconstructing them through the values of the neighboring cells in order to enforce the TVD property maintaining an high order of accuracy. An important weakness of TVD schemes is the loss of accuracy near local extrema. For this reason researchers started to look for a less restrictive version of (4.3). To overcome this problem, it was developed a new class of problem: the Total Variation Bounded (TVB) schemes.

Definition 4.1.3 ***(Total Variation Bounded schemes)*** *A scheme is a Total Variation Bounded (TVB) method if*

$$TV(u(x,t)) \leq M \quad \forall t \leq T_{max} \tag{4.4}$$

4.1.1 TVB type limiter

This type of limiting was proposed throughout the work of Cockburn and Shu in the series of paper [10], [6],[5]. This limiter revises the approximated solution such that its cell average values become TVB stable, and it reduces the high order polynomials to linear polynomials in "trouble" cells where artificial oscillations appear. A basic criteria is implemented such that the high order polynomial is not modified in case the cell is not flagged as a "trouble" cell, thus the numerical scheme does not lose its high order accuracy by the limiter. As it was pointed out before, the states of the approximated solution $\mathbf{w}_h(t, \mathbf{x})$ in a cell are constructed as a linear combination of polynomial basis functions. Therefore, they can also be written as

$$w_{h,j}(t,x) = \sum_{l=0}^{k} w_j^{(l)}(t)\varphi_l^j(x), \tag{4.5}$$

in τ_j, where k is the numbers of basis functions. Here it is assume that $\mathcal{Q}^k$ polynomial basis are being used.

The average gradients of the states, calculated as

$$\overline{\nabla w_j} = \frac{1}{|\tau_l|}\int_{\tau_l} \nabla w_j dx \tag{4.6}$$

are compared component wise with the average values of the state variables $\overline{\mathbf{w}}$ in the element τ_l. The modified average gradient $\nabla w_j^{(m)} = [\partial_x w_j^{(m)}]^T$ is computed as

$$\partial_x w_j^{(m)} = func\left(\overline{\partial_x w_j}, \beta\frac{\Delta_-\overline{w}_j}{\Delta x}, \beta\frac{\Delta_+\overline{w}_j}{\Delta x}\right) \tag{4.7}$$

where $\beta \in [1,2]$ $func(a_1, ..., a_d)$ is a function described later in the following section. If $\partial_x w_j^{(m)} \neq \overline{\partial_x w_j}$, then the solution w_h must be reduced to the following linear polynomial $\Lambda\Pi_h(\mathbf{w}_h)(x)$

$$\Lambda\Pi_h(\mathbf{w}_h)(x) = \overline{w_j} + (x - x_j)\partial_x w_j^{(m)}. \tag{4.8}$$

The limiting criteria relies entirely on the mean values of the states $w^{(0)}$, both inside the cell τ_j and the neighboring cells τ_{j+1} and τ_{j-1}, for the specific direction. To explain this procedure, let the forward and backward differences of the mean values be respectively defined as

$$\Delta_+ w_j^{(0)} = w_{j+1}^{(0)} - w_j^{(0)}, \quad \Delta_- w_j^{(0)} = w_j^{(0)} - w_{j-1}^{(0)}. \tag{4.9}$$

A canonical example for function $func$ is given by the so-called minmod limiter, introduced by Cockburn and Shu [9]. The boundary values are then changed as

$$\tilde{w}^{mod} = \tilde{m}\{\tilde{w}, \Delta_+ w_j^{(0)}, \Delta_- w_j^{(0)}\} \quad \tilde{\tilde{w}}^{mod} = \tilde{m}\{\tilde{\tilde{w}}, \Delta_+ w_j^{(0)}, \Delta_- w_j^{(0)}\}. \tag{4.10}$$

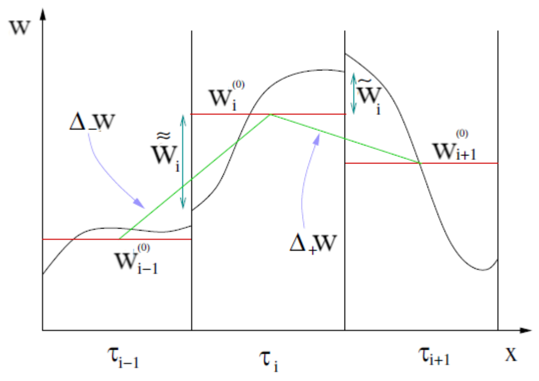


Figure 4.2: Illustration of some quantities used for the TVB limiting procedures

The function $\tilde{m}$ is the modified minmod function given by

$$\tilde{m}(a_1, ..., a_n) = \begin{cases} a_1, & if\ |a_1| \leq Mh^2 \\ m(a_1, ..., a_n), & otherwise \end{cases} \tag{4.11}$$

where the parameter $M \geq 0$ has to be determined, and $m(a_1, ..., a_n)$ is the minmod function defined as

$$m(a_1, ..., a_n) = \begin{cases} s \min_{1\leq j\leq n} |a_j|, & if\ sign(a_1) = ... = sign(a_n) = s, \\ 0, & otherwise, \end{cases} \tag{4.12}$$

The indicator criteria of the modified minmod function is given by the comparison $|a_1| \leq Mh^2$, therefore the selection of the value for the only parameter M has a huge impact on the performance of this limiter. The lower the value of M, the more restrictive the limiter becomes. In the context of scalar conservation laws, some selection options of the parameter M where presented in [10]. This selection options use the fact that the solution of a scalar conservation law satisfies a maximum principle. But this is not the case for arbitrary systems of conservation laws, in which situation M has to be chosen carefully by intuition. One could simply select the parameter M as zero, but this will certainly destroy the accuracy of the numerical solution in regions where the solution is smooth but a local extrema is located.
As a second example, van Albada (vA) limiter are mentioned. The function $func$ are

$$\phi^{\epsilon}_{vA}(a, b) = \frac{(a^2 + \epsilon^2)b + (b^2 + \epsilon^2)a}{a^2 + b^2 + 2\epsilon^2} \qquad \epsilon^2 = \mathcal{O}(\Delta x^3) \tag{4.13}$$

This limiter was first proposed by van Albada, van Leer and Robert [13] in 1982 and has been successfully applied in computations, although a rigorous study of its TVB stability was overshadowed by widespread applications of minmod limiter. In the smooth regions where $a \approx b$, the limiter tends to recover the second order central finite differencing $\frac{a+b}{2}$. Across the discontinuities, however, the average slope is biased to the smallest value among the two one-sided slopes. These mechanisms are expected to ensure the second order accuracy and prevent the undesirable numerical oscillations. From literature, it is expected that van Albada limiter introduces less dissipation than the minmod limiter at discontinuities and smooth extrema.

4.1.2 Shock indicator

Krivodonova et al. [3] introduced a way to detect trouble cells where discontinuities may occur. Is is known as the KXRCF indicator and it is based on the fact that the DG approximation shows strong super convergence at the outflow boundary of each element only when smooth solutions are approximated.
Let the boundaries of each cell τ_l be divided into inflow $(\partial\tau_l^-)$ and outflow $(\partial\tau_l^+)$ portions. The super-convergence results

$$\frac{1}{|\partial\tau_l^+|}\int_{\partial\tau_l^+}(y_h - y)d\sigma = \mathcal{O}(h^{2k+1}), \tag{4.14}$$

where the variable y is a component of the state $\mathbf{w}$. Equation (4.14) means that, at the outflow boundary points, the rate of approximation y_h to a variable y is almost twice the optimal rate of approximation. S.Adjrid and T.C. Massay in [2] and then Y.Cheng and C-.W. Shu in [8] and B. Cockburn et al. in [11] had demonstrated the super-convergence for DG method. At inflow boundaries the rate of approximation would be given by

$$\frac{1}{|\partial\tau_l^-|}\int_{\partial\tau_l^-}(y_h - y)d\sigma = \mathcal{O}(h^{k+1}). \tag{4.15}$$

Hence, analyzing the jump in the inflow edge $\alpha \subseteq \partial\tau_l^-$

$$I_\alpha = \int_\alpha (y_h^- - y_h^+)d\sigma = \int_\alpha (y_h^- - y)d\sigma + \int_{\alpha_{nb}} (y - y_h^+)d\sigma. \tag{4.16}$$

Here y^- and y^+ represent the values at the boundary from inside and outside the cell τ_l respectively, and $\alpha_{nb} = |\alpha|\mathbf{n}_{nb} = -|\alpha|\mathbf{n}$ is the corresponding outflow boundary of the neighboring cell. The first integral is then of order $\mathcal{O}(h^{k+2})$, while the second one is of order $\mathcal{O}(h^{2(k+1)})$. Thus means that in the case of smooth solutions, I_α is of order $\mathcal{O}(h^{k+2})$ across the interface α. If the variable y is discontinuous in a cell, then at least one of the integrals in (4.16) will be of order $\mathbf{O}(h)$. Therefore,

$$I_\alpha = \begin{cases} \mathcal{O}(h^{k+2}), & if\ y|_{\partial\tau_l}\ is\ smooth, \\ \mathcal{O}(h), & if\ y|_{\partial\tau_l}\ is\ discontinuous. \end{cases} \tag{4.17}$$

Normalizing the quantity I_α with a kind of average convergence of order $\mathcal{O}(h^{(k+1)/2})$, the discontinuity detector is then constructed as follows

$$\mathcal{I}_\alpha = \frac{|\int_\alpha (y_h^- - y_h^+) d\sigma|}{h^{\frac{k+1}{2}} |\alpha| \|y_h\|_{\tau_l}} \tag{4.18}$$

for $\alpha \subseteq \partial\tau_l^-$. Since the indicator function will go to zero as the grid size reduced or the polynomial order is increased ($\mathcal{I}_\alpha \to 0$ as $h \to 0$ or $k \to \infty$), and it wil go to infinity ($\mathcal{I}_\alpha \to \infty$) near a discontinuity, the quantity can be interpreted as follows

- if $\mathcal{I}_\alpha < 1$, y is continuous
- if $\mathcal{I}_\alpha > 1$, y is discontinuous

4.2 Second procedure: a filtering monotonization approach

A filtering technique [21] for Discontinuous Galerkin approximations of hyperbolic problems is introduced. Following an approach already proposed for the Hamilton-Jacobi equations by other authors [20], it's aimed at reducing the spurious oscillations that arise in presence of discontinuities when high order spatial discretizations are employed. This goal is achieved using a filter function that keeps the high order scheme when the solution is regular and switches to a monotone low order approximation if it is not. The main novelty of the proposed method is that it is did not rely on a regularity indicator and that a monotonic solution is retrieved automatically.
Since each stage of the TVD method can be represented as $\mathbf{u} = \mathbf{S}(\mathbf{v})$; where $\mathbf{u}$; $\mathbf{v}$ denote the new and old values, respectively, of the vector containing the discrete degrees of freedom which identify the spatial approximation to the solutions of (2.1). $\mathbf{S}$ denotes formally the solution operator associated to a specific time and space discretization. The transition from $\mathbf{u}$ to $\mathbf{v}$ can be interpreted as an advancement in time of $\alpha\Delta t$ time units, where α depends on the details of the TVD method and on the specific stage considered. It is denoted by $\mathbf{S}^M$ the discrete operator associated to the monotonic, low order spatial discretization and by $\mathbf{S}^H$ that associated to a high order, not monotonic spatial discretization.
A filter function F is introduced. This can be defined in several ways, for example

$$F_1(x) =, x\mathbf{I}_{|x|\leq 1}, \qquad F_2(x) = sign(x)\max(1 - ||x| - 1|, 0), \tag{4.19}$$

In the simplest possible filtering approach, the filtered version of $\mathbf{u}$ can be defined as

$$\mathbf{u}^F = \mathbf{S}^M(\mathbf{v}) + \epsilon\alpha\Delta t F\bigg(\frac{\mathbf{S}^H(\mathbf{v}) - \mathbf{S}^M(\mathbf{v})}{\epsilon\alpha\Delta t}\bigg) \tag{4.20}$$

where the low order solution $\mathbf{S}^M$ is computed on the nodes of the high order solution $\mathbf{S}^H$ and ϵ is a suitable parameter, depending on the time and space discretization parameters, such that

$$lim_{(\Delta t,h)\to 0}\ \ \epsilon(\Delta t, h) = 0, \tag{4.21}$$

with $h = \max\{diam(\tau_l) : \tau_l \in T\}$. The filter function is applied component-wise, in this way, the high order method is only applied to the components i for which

$$\frac{|\mathbf{S}^H(\mathbf{v})_i - \mathbf{S}^M(\mathbf{v})_i|}{\epsilon\alpha\Delta t} \leq 1, \qquad i = 1, ..., dim(Q_h). \tag{4.22}$$

As explained in [21], ϵ has to be chosen in such a way that $\epsilon \geq c_0 h$, where c_0 is a sufficiently large constant.
Since this approach is very dissipative and, unless a very large value of c_0 is adopted, it provides solutions that coincide with the low order one. Therefore, the alternative filtering strategy is proposed

$$\mathbf{u}_i^F = \mathbf{S}^M(\mathbf{v})_i + \beta\mathbf{S}^M(\mathbf{v})_i F\left(\frac{\mathbf{S}^H(\mathbf{v})_i - \mathbf{S}^M(\mathbf{v})_i}{\beta\mathbf{S}^M(\mathbf{v})_i}\right), \qquad i = 1, ..., dim(Q_h), \tag{4.23}$$

where $\beta \geq 0$ is suitable parameter that represents a tolerance for the "componentwise relative difference" $\frac{\mathbf{S}^H(\mathbf{v})_i - \mathbf{S}^M(\mathbf{v})_i}{\mathbf{S}^M(\mathbf{v})_i}$, so that $\left|\frac{\mathbf{S}^H(\mathbf{v})_i - \mathbf{S}^M(\mathbf{v})_i}{\mathbf{S}^M(\mathbf{v})_i}\right| \leq \beta$, it's resorted to the high order solution. A too small value of β provides results that are in practically coincident with the low order solution.

The application of the filtering approach to the Euler equation reads as follows:

$$\begin{aligned}
\boldsymbol{\rho}_i^F &= \mathbf{S}^M(\boldsymbol{\rho})_i + \beta_\rho\mathbf{S}^M(\boldsymbol{\rho})_i F\left(\frac{\mathbf{S}^H(\boldsymbol{\rho})_i - \mathbf{S}^M(\boldsymbol{\rho})_i}{\beta_\rho\mathbf{S}^M(\boldsymbol{\rho})_i}\right), \quad i = 1, ..., dim(Q_h),\\
\boldsymbol{\rho}\mathbf{u}_i^F &= \mathbf{S}^M(\boldsymbol{\rho}\mathbf{u})_i + \beta_{\rho\mathbf{u}}\mathbf{S}^M(\boldsymbol{\rho}\mathbf{u})_i F\left(\frac{\mathbf{S}^H(\boldsymbol{\rho}\mathbf{u})_i - \mathbf{S}^M(\boldsymbol{\rho}\mathbf{u})_i}{\beta_{\rho\mathbf{u}}\mathbf{S}^M(\boldsymbol{\rho}\mathbf{u})_i}\right), \quad i = 1, ..., dim(Q_h),\\
\boldsymbol{\rho}\mathbf{E}_i^F &= \mathbf{S}^M(\boldsymbol{\rho}\mathbf{E})_i + \beta_{\rho\mathbf{E}}\mathbf{S}^M(\boldsymbol{\rho}\mathbf{E})_i F\left(\frac{\mathbf{S}^H(\boldsymbol{\rho}\mathbf{E})_i - \mathbf{S}^M(\boldsymbol{\rho}\mathbf{E})_i}{\beta_{\rho\mathbf{E}}\mathbf{S}^M(\boldsymbol{\rho}\mathbf{E})_i}\right), \quad i = 1, ..., dim(Q_h),
\end{aligned} \tag{4.24}$$

where $\boldsymbol{\rho}$, $\rho\mathbf{u}$, $\rho\mathbf{E}$ and β_ρ, $\beta_{\rho\mathbf{u}}$, $\beta_{\rho\mathbf{E}}$ are the vectors of the degrees of freedom and the tolerance parameters for all the conservative variables.

4.3 Mesh refinement

There are a number of ways how one can adaptively refine meshes. The basic structure of the overall algorithm consists of a loop over the following steps:

- Solve the PDE on the current mesh;
- Estimate the error on each cell using some criterion that is indicative of the error;

- Mark those cells that have large errors for refinement, mark those that have particularly small errors for coarsening, and leave the rest alone;
- Refine and coarsen the cells so marked to obtain a new mesh;
- Repeat the steps above on the new mesh until the overall error is sufficiently small.

The situation can be further improved using mesh adaptivity so as to start with a coarse mesh and perform refinement only in the zones where discontinuity is detected. Two different indicators are introduced. The first is is based on the gradient of the variable ρ; more specifically, for each element is defined

$$\eta_K = \log(1 + \sqrt{\nabla\rho^2}); \tag{4.25}$$

as local refinement indicator. The second indicator is base on the gradient of the variable *pressure*.

$$\eta_K = |\nabla p|; \tag{4.26}$$

These two different indicators are chosen to calculate the h-refinement in order to verify the accuracy of the solutions when the indicator is applied on the characteristic or conserved variable. By adapting the mesh, significant savings can be achieved, but, compared to the stationary case, there are considerable difficulties. Let's go through: *time step size and minimal mesh size*. For stationary problems, the general approach is "make the mesh as fine as it is necessary". For problems with singularities, this often leads to situations where many levels of refinement into corners or along interfaces are gotten. However, for time dependent problems, typically it's needed to choose the time step related to the mesh size. For explicit time discretizations, this is obvious, since it's needed to respect a CFL condition that ties the time step size to the smallest mesh size. Refining the mesh further in one place implies not only the moderate additional effort of increasing the number of degrees of freedom slightly, but also the much larger effort of having the solve the global linear system more often because of the smaller time step. In practice, one typically deals with this by acknowledging that it's could not make the time step arbitrarily small, and consequently can not make the local mesh size arbitrarily small. Rather, a maximal level of refinement is setting and when cells for refinement are flagged, it's simply did not refine those cells whose children would exceed this maximal level of refinement. To avoid being caught flat footed with too coarse a mesh in areas where it's suddenly needed a finer mesh, it's will also enforced in the code a minimal mesh refinement level.

Chapter 5

Numerical simulations

Introduction

In the previous chapters, a general framework to find numerical solutions for hyperbolic systems of conservation laws using the discontinuous Galerkin method was introduced. Additionally some techniques to deal with oscillations around discontinuities of the solutions were presented. This chapter shows, through different two-dimensional test cases, the application of the concepts previously introduced on the Euler equations. In the following sections the test cases will be explained together with the numerical results obtained through a `C++` implementation developed using the `deal.II C++` libraries [4].

5.1 Implementation details

5.1.1 The `deal.II` libraries

`deal.II C++` is a set of libraries programmed in `C++` (this work uses the deal.II 9.3.0 version introduced in [4]), which aids the implementation of grid based numerical methods. Its structure is based on object oriented programming, it is specialized in Cartesian grids and supports the necessary tools to implement FEM, FVM and DG methods. The major advantages that it provides with respect to other open-source codes reside on the continuous development of the code and the community working around it. Therefore it has an up-to-date and very complete documentation, plus a very active users group, where questions are posted almost every day. Many of the classes in the `deal.II` library can be grouped into modules. These modules form around the building blocks of any finite element program. An outline of how the primary groups of classes in `deal.II` interact is given by the following Figure5.1 (gray boxes denote a subset of the optional external libraries, gray ovals a subset of

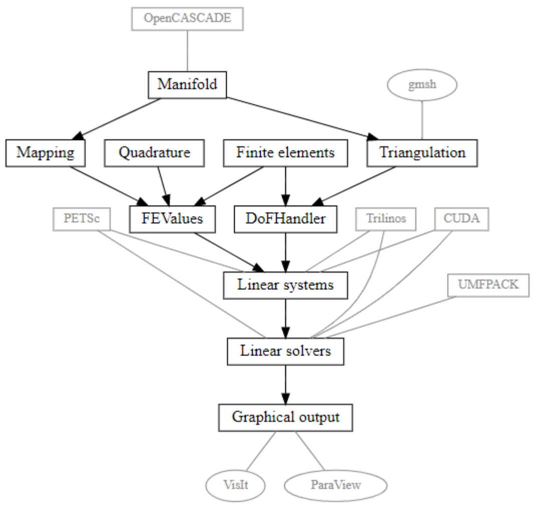


Figure 5.1: Outline of how the primary groups of classes in `deal.II` interact

the optional external applications with which `deal.II` can interact).

5.1.2 Characteristics of the `Eulero` code

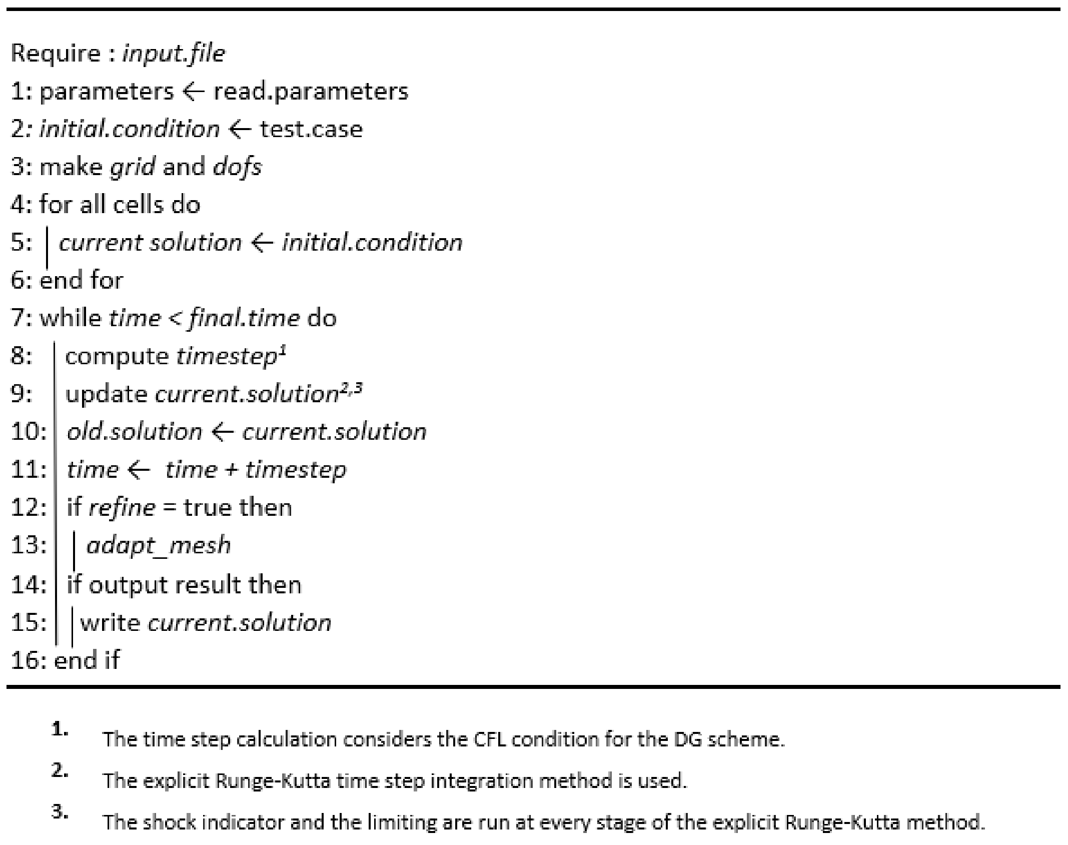
This is a `C++` code developed to compute solutions for systems of conservation laws, which uses the `deal.II` libraries previously introduced. The `Eulero` code uses the general structure of dealii's tutorial 67 [26] in which the basic functions for the DG are implemented. The tutorial has been modified to implement other numerical fluxes, limiters and indicator, adaptivity of the mesh, supersonic boundary conditions and explicit SSP integration. The solution of common test cases show the capability of the method. All the numerical results presented in this chapter have been obtained from simulations implemented in the `Eulero` code.
The code code has the following characteristic:

- works on a Cartesian grid for two dimensional problems;
- uses different numerical fluxes: Lax-Friedrichs, Harten-Lax-VanLeer, HLLC, SLAU or Roe can be selected;
- uses SSP3 explicit time integration;
- uses TVB limiters (minmod or Van Albada function) or filtering technique;
- refines the mesh using as indicator the gradient of density or the gradient of pressure;

The `Eulero` code is available in the following repository:

`https://github.com/Felotti/Eulero.git`

5.1.3 General structure of the `Eulero` code



```
Require : input.file
1: parameters ← read.parameters
2: initial.condition ← test.case
3: make grid and dofs
4: for all cells do
5:    current solution ← initial.condition
6: end for
7: while time < final.time do
8:    compute timestep[1]
9:    update current.solution[2,3]
10:   old.solution ← current.solution
11:   time ← time + timestep
12:   if refine = true then
13:      adapt_mesh
14:   if output result then
15:      write current.solution
16: end if
```

1. The time step calculation considers the CFL condition for the DG scheme.
2. The explicit Runge-Kutta time step integration method is used.
3. The shock indicator and the limiting are run at every stage of the explicit Runge-Kutta method.

This section is dedicated to the description of the code and to the justification of the technical choices that it's made while writing it. The `Euler` code is divided into three main namespaces:

- *Parameters* → the first struct `Solver` selects a detail of the spatial discretization, namely the numerical flux at the faces between cells, while the `Data_Storage` struct collects all parameters that control the execution of the program. This class has function `read_data`, which it reads all declared parameters from an input file.

- *EquationData* → it collects three classes `ExactSolution`, `InitialData` and `BoundaryData`, which define the analytical solution for the Sod-Shock tube problem(1D), the initial values, the boundary values for all test cases, respectively. Given that definition, it is returned either the density, the momentum, or the energy depending on which component is required.

- *Euler_DG* → it collects the two main classes and many functions grouped in the `operator.h` file.

– The file `operator.h` collects all those functions which are decorated with the keyword `DEAL_II_ALWAYS_INLINE` and which are called millions or billions of times in the code. This special macro is critical for performance of code. It's implemented the various problem-specific operators pertaining to the Euler equations. Each function acts on the vector of the conserved variables $[\rho, \rho\mathbf{u}, E]$ that it's hold in the solution vectors, and computes various derived quantities. First out is the computation of the velocity in `euler_velocity`, that it's derived from the momentum variable $\rho\mathbf{u}$ by division by ρ. The function `euler_pressure` computes the pressure from the vector of conserved variables, using the formula (2.4). The function `euler_flux` is the definition of the Euler flux, given the velocity and pressure, this is straight-forward given the equation stated in (2.2). The function `euler_numerical_flux` implements the numerical flux (Riemann solvers presented in the chapter (4)). It gets the state from the two sides of an interface and the normal vector, oriented from the side of the solution $\mathbf{w}^-$ towards the solution $\mathbf{w}^+$.

– `EulerOperator` is a class that implements the evaluators for the Euler problem. It sets effectively the weak formulation of the problems for the different stages. The template parameters are the dimension of the problem, the polynomial degree, the number of quadrature points for integrals. The code is in a DG-MatrixFree framework, so it is convenient to compute separately cell contribution, internal faces contributions and boundary faces contributions. It's provided a few functions to allow a user to pass in various forms of boundary conditions on different parts of the domain boundary marked by `type::boundary_id` variables. Different member functions that are the ones that must be called from outside to specify the various typed of boundaries are implemented.
`set_inflow_boundary`, `set_subsonic_outflow_boundary`,
`set_supersonic_outflow_boundary` and `set_wall_boundary` member functions store the function alongside the respective boundary id in a map member variable of this class. For the initialization of the Euler operator, in `reinit`, it's set up the `MatrixFree` variable contained in the class. It's proceed to the local evaluators for the Euler, which implements the core operation of the program, the integration over a range of cells for the nonlinear operator of the Euler problem. The `local_apply_cell` function performs the time stepping routine based on the cell-local strategy. The `local_apply_face` function concerns the computation of integrals on interior faces, where it's needed evaluators from both cells adjacent

to the face. For faces located at the boundary, it's needed to impose the appropriate boundary conditions. The discontinuous Galerkin method imposes boundary conditions not as constraints, but only weakly. Thus, the various conditions are imposed by finding an appropriate exterior quantity $\mathbf{w}^+$, as stated in the section (2.2.1), that is then handed to the numerical flux function also used for the interior faces. In the implementation `local_apply_boundary_face`, it's checked for the various types of boundaries at the level of quadrature points.
The `local_apply_inverse_mass_matrix` implements the inverse mass matrix operation. The `apply` function implements the evaluation of the Euler operator as a whole, i.e., $\mathcal{M}^{-1}\mathcal{L}(t, \mathbf{w})$, calling into the local evaluators presented above.
`project`, `compute_errors` and `compute_cell_transport_speed` are functions that compute projections, evaluate errors, and compute the speed of information transport on a cell.

– `EulerProblem` class combines the `EulerOperator` class with the time integration and the usual global data structures, to actually run the simulations of the Euler Problem. The member are a triangulation, a finite element, a mapping, and a DofHandler to describe the degrees of freedom. In addition, it's kept an instance of the `EulerOperator` described above around, which will do all heavy lifting in terms of integrals, and some parameters for time integration like the current time or the time step size. Furthermore, it's used a `PostProcessor` instance to write some additional information to the output file. The `make_grid` member function implements all the grid of the testcases, and impose the boundary conditions all around the domain. The `make_dofs` function create the unknown numbering from the DofHandler, and hand the DofHandler and Mapping object to the initialization of the `EulerOperator`. Furthermore, for each cell find the neighbor cells, needed for the limiter. The `adapt_mesh` takes care of the adaptive mesh refinement described before. At the beginning, it's looped over all the cells and marks those that it's thought should be refined. Then it's needed to transfer the various solution vectors from the old to the new grid. The following member functions are defined with the aim of implementing the indicator and the limiter. In particular, `compute_cell_average` is a function that compute the average in a cell average solution; `compute_shock_indicator` computes the KXRCF shock indicator, described in the section (4.1.2); `apply_limiter_TVB` computes the TVB version of minmod and van Albada limiter, applying the function `minmod` or `vanAlbada`; `apply_filter` compute the filter technique, described in section (4.2). The `update` function computes the SSP order 3 Runge Kutta method. At each stage, it's applied or the filter technique or the TVB limiter, in order to deal with the discontinuities. The `run` function puts all pieces together. It starts off by calling the function that creates the mesh and sets up data structures, and then, before starting the time loop, it's computed the time step size by `compute_cell_transport_speed` function. The initial values to the grid are projected and the first data for the solution vector are obtained. Then, it's initialized time step number and time step and start the time loop. The results obtained there are compared with the minimal mesh size and print them to screen. The time loop is started, which it's run until the time has reached the desired end time. Every 5 time steps, it's computed a new estimate for the time step - since the solution is nonlinear, it is most effective to adapt the value during the course of the simulation. In case the Courant number was chosen too aggressively, the simulation will typically blow up with time step NaN, so that is easy to detect here.

5.2 Euler equations - Test Cases

- Sod shock tube – these shock tubes are classical tests to assess the ability of a numerical method to deal with simple waves (rarefaction, contact discontinuity and shock wave);

- Forward facing step – this classical test simulates a supersonic flow over a forward facing step. The solution approaches a steady-state solution composed of multiple interacting shock waves and vortex like structures which are often dissipated with low order accurate schemes.

- Double Mach reflection – this classical test is meant to measure visually the ability of a numerical method to capture complex patterns created after the interaction of shock waves.

- 2D Riemann problems - this classical suite of test problems is meant to assess the ability of a numerical method to solve genuinely two-dimensional Riemann problems emerging from four piece-wise constant states joining at the origin.

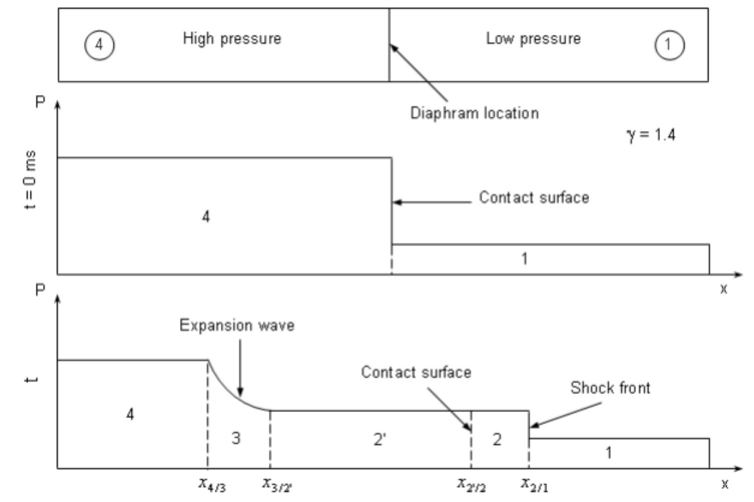


Figure 5.2: Schematic shock tube problem with pressure distribution for pre- and post-diaphragm removal

5.2.1 Shock tube problem

The Sod-Shock tube problem is frequently used to test the accuracy of computational methods, proposed by P. Woodward and P. Colella [27]. An analytical solution is in fact available, therefore it is possible to compare numerical results with it and understand the strengths and weaknesses of the scheme implemented. A shock tube consists of a pipe with rectangular cross section filled with a fluid with a diaphragm splitting the tube in two halves. The diaphragm is numerically simulated as a discontinuity in different fluid conditions (pressure and density) across that specific surface. The left side of the tube has higher values for the fluid properties. Given such initial conditions, the system is allowed to evolve in time. A shock wave will move towards the right (low pressure region) and a refractive wave will move to the left (high pressure region). A contact discontinuity separates the two regions and is visible in Figure 5.2. Initial left and right conditions for the density ρ $[kg/m^3]$, velocity $\mathbf{u}$ $[m/s]$ and pressure p $[Pa]$ are given as:

$$
\begin{aligned}
\rho_L &= 1.0 & \qquad \rho_R &= 0.125 \\
\mathbf{u}_L &= 0.0 & \qquad \mathbf{u}_R &= 0 \\
p_L &= 1.0 & \qquad p_R &= 0.1
\end{aligned}
$$

The solution is obtained at time $t = 0.2s$. For the two-dimensional setup, the computational domain is given by $\Omega = [-0.5; 0.5]X[0; 0.2]$. The ratio of specific heats is $\gamma = 1.4$ and the initial discontinuity is located in $x = 0$. All the simulations are scaled in the x-axis $[0, 1.0]$. The computations are performed using meshes in

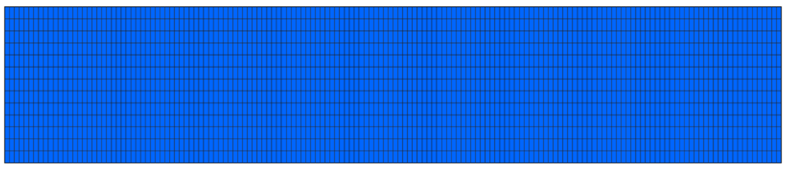


Figure 5.3: Computational mesh (160x13 elements)

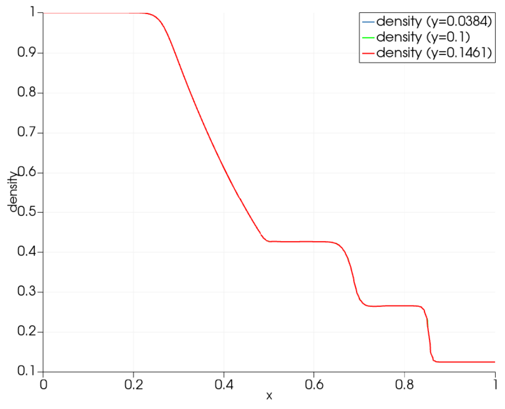


Figure 5.4: Numerical solution at $y = 0.0384$, $y = 0.1$ and $y = 0.1461$

figure 5.3, fine meshes that have 160 and 13 cells in horizontal and vertical direction, respectively. It's also proved that the only variability in the solution is in the x-direction where it's imposed the initial conditions for the internal flow. The extra dimension added for the 2D case does not really have an effect to the overall solution of the problem, for this reason, it has chosen a less refined mesh along the y-axis. It can see in figure 5.4 that there is no variability along the y axis since the initial conditions show a discontinuity along the x axis only.

To demonstrate the accuracy of `Eulero` code, it's compared the analytical solution to the numerical one by plotting density, pressure and velocity curves along the shock tube. It can see that the trends are the classic ones of a shock tube, on which much has been written in the literature, and confirm the correctness of the written code. It is interesting to note how the expansion, the leftmost wave, and the impact, the wave to the right, are clearly visible in both graphs presented; while, the contact discontinuity is visible only in the density graph. In fact, across a contact surface the normal component of the velocity and the pressure remain constant. It is possible to see how the contact surface is the one that diffuses more than other two; in fact, it is poorly represented by the numerical solution. In the following simulations it can seen how, at different numerical fluxes, the discontinuity is captured differently. It's made a first comparison imposing the TVB as limiter using the minmod function, see Figures 5.5 - 5.6 - 5.7 - 5.8 - 5.9. Then, a comparison between the numerical fluxes with the TVB limiter but with the vanAlbada function

(Figures 5.11 - 5.12 - 5.13 - 5.14 - 5.15) and finally it's plotted the results obtained with the filter technique (Figures 5.16 - 5.17 - 5.18 - 5.19). The variations in density, velocity and pressure represents that the scheme can capture discontinuities with good accuracy. In particular, considering the TVB limiter, it is possible to capture the desired trend for all numerical fluxes, both using the minmod function and the vanAlbada function. Furthermore, if the simulations with the various fluxes and the TVB limiter(minmod function) are considered, the error can be observed in the figures 5.10. The error was calculated as $\frac{|numerical\_solution - analitycal\_solution|}{|analitycal\_solution|}$. In the density error plot, it can be seen that the contact discontinuity is captured well by all numerical fluxes. In spite of its simplicity, the main disadvantage of the Lax-Friedrichs scheme is the excessive numerical dissipation which reduces its resolution at discontinuities. In the area where the expansion wave occurs, the worst behavior is of the HLLC flux. The HLLC scheme should have shown good results, however it did not. It is not clear where the error lied but it is possible the chosen wave speed calculation was not sufficient. Other wave speed estimates for the Star Region are available, and further investigation would be warranted to improve the performance of the HLLC scheme, as it is known to perform better than HLL in these regions. The HLL and Roe solvers prove to be a very powerful solver, having several qualities. The results have shown the high resolution and the non-oscillatory property of the van Albada limiter, but the vA function limiter not yields superior performance over the minmod function limiter. The solution with polynomial degree k=2 is much closer to the solution with polynomial degree zero. As regards the application of the filter technique, it was to be expected that the solution with the HLL numerical flux would have produced good results, as the study carried out in [21] demonstrated the potential of this filter with this flux. However, with the other numerical fluxes, good results have not been obtained. The solution is still oscillatory with various values of the beta parameter. The question of the beta parameter should be further investigated, perhaps making it dependent on the fluxes used.
Finally, the refinement was applied to the mesh using both the pressure gradient indicator and the density gradient indicator, presented in the section 4.3. The simulation is achieved with TVB limiter, minmod function and HLL flux. Figure 5.20 shows the meshes at t=0 and at t=0.2s after mesh refinement, with the density and pressure gradient, respectively. It can be seen that the mesh obtained with the density gradient highlights the expansion wave zone, the contact surface zone and the shock zone. As expected, as explained above, the contact discontinuity is not captured better with the mesh refinement with the pressure indicator. As can be seen from the figure 5.21, as the resolution of the mesh is increased the obtained distribution matches the reference study better.

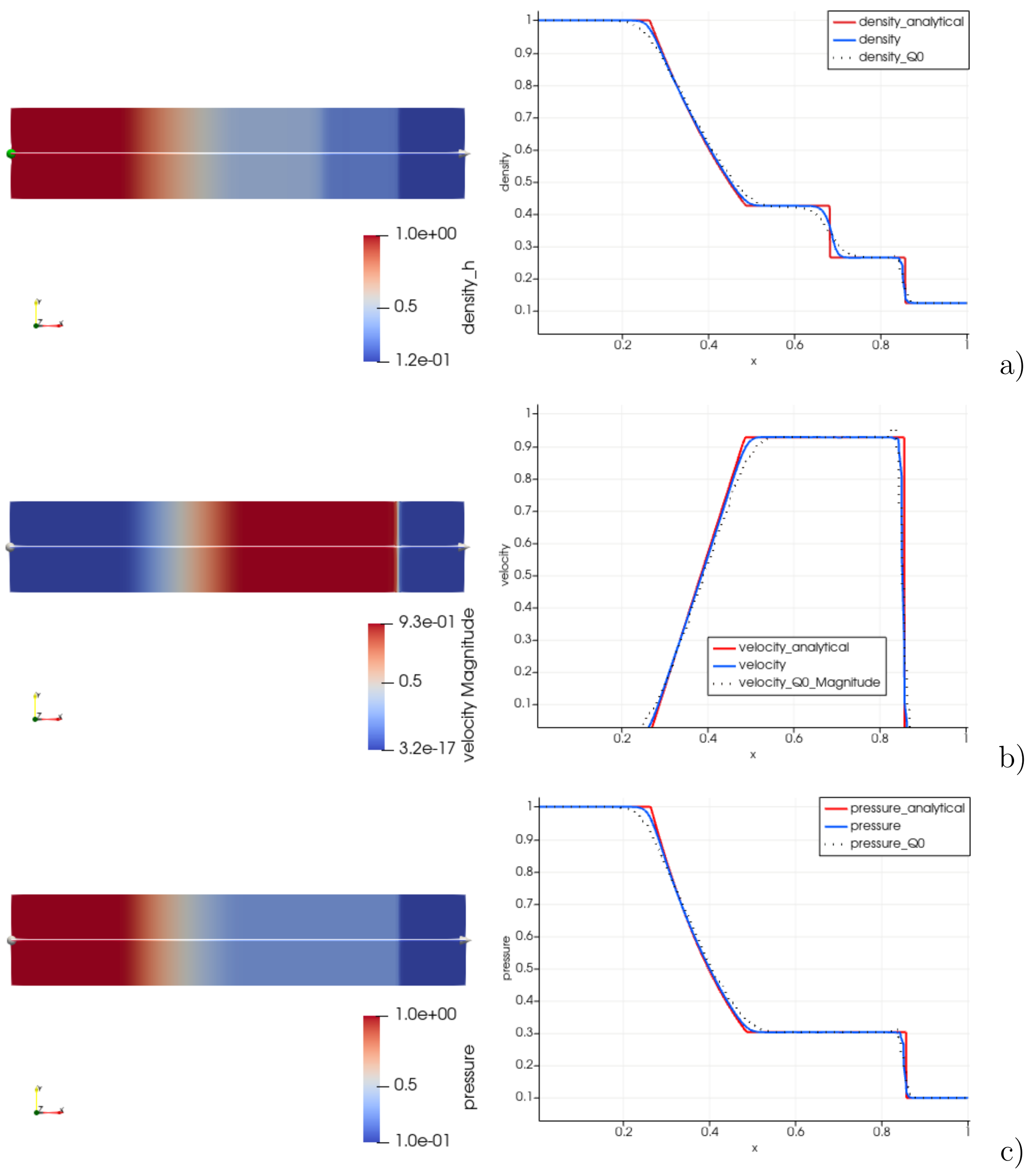
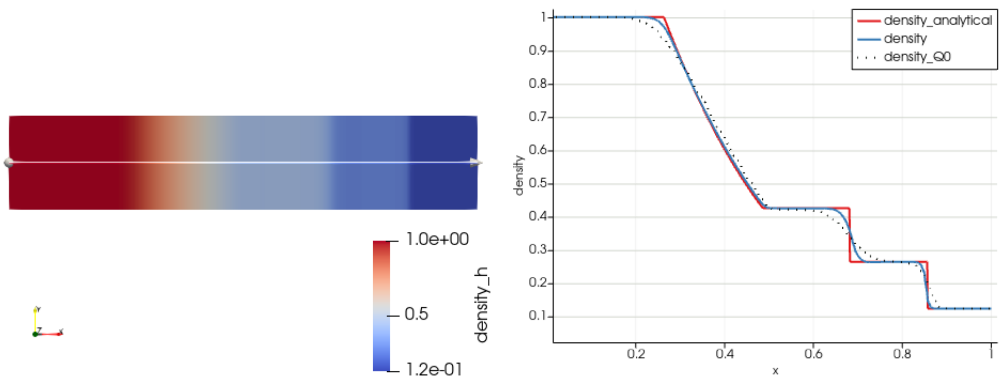
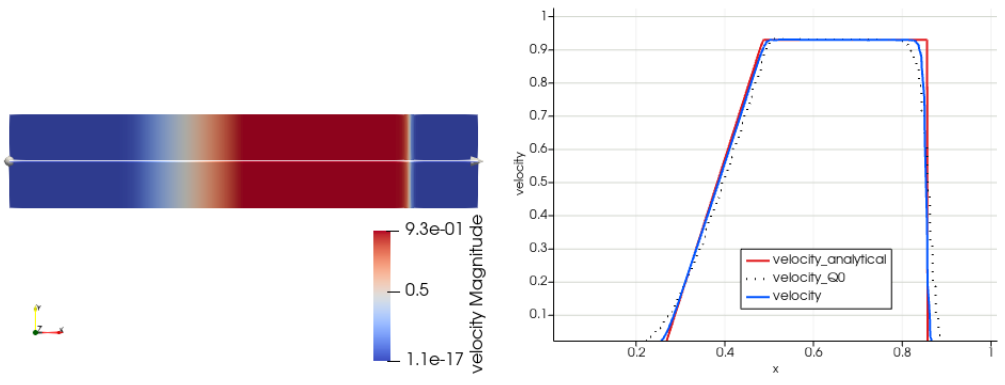


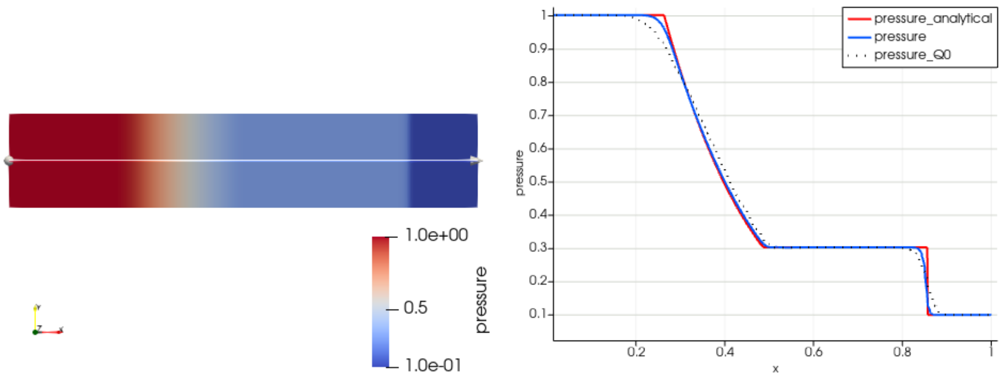
Figure 5.5: Computational results for Sod shock tube problem at t = 0.2s, with Lax-Friedrichs flux, TVB limiter and minmod function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.



a)



b)



c)

Figure 5.6: Computational results for Sod shock tube problem at t = 0.2s, with Harten Lax vanLeer flux, TVB limiter and minmod function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution $(k = 2)$, the dark dots denotes the $Q0$_solution.

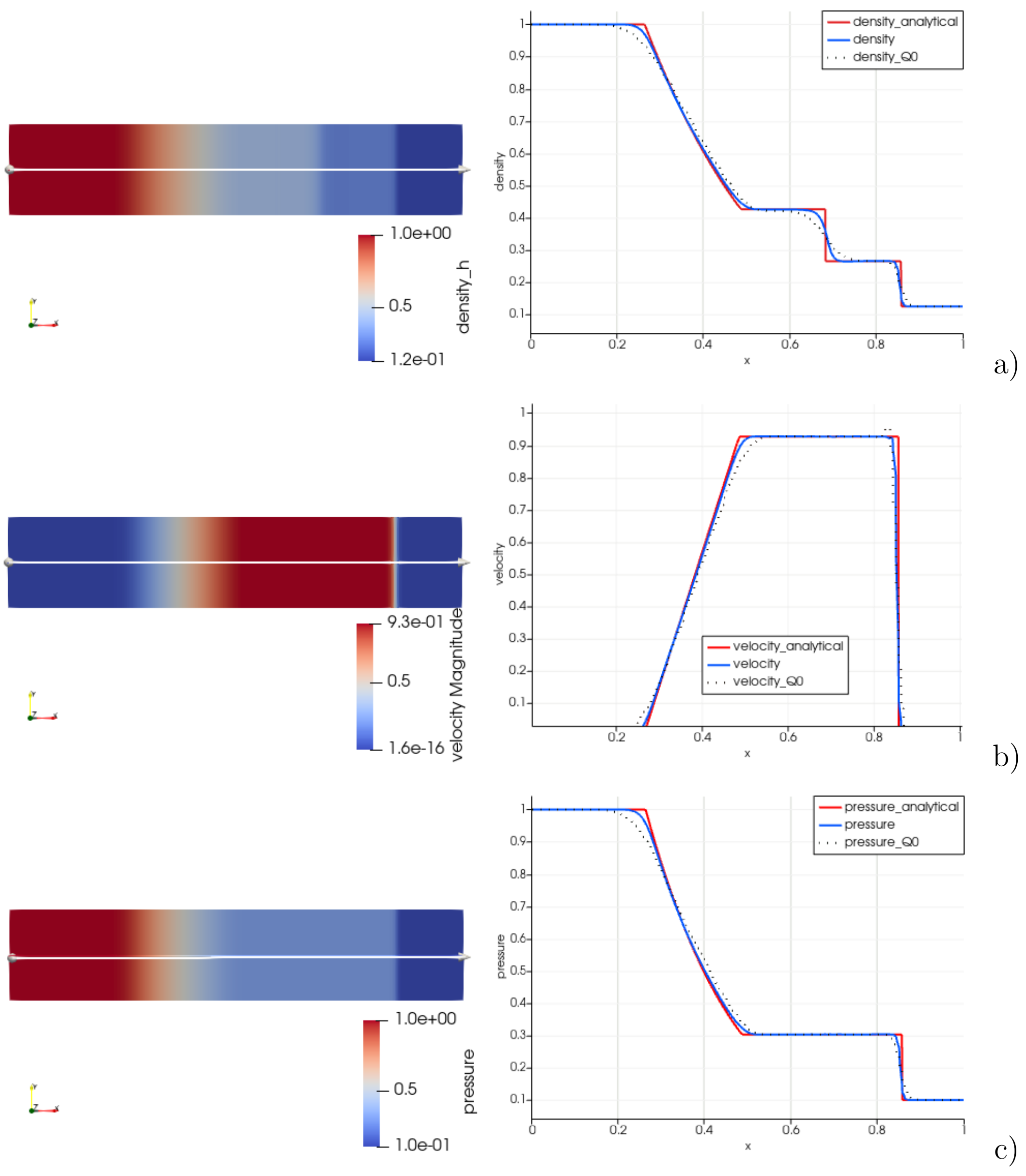


Figure 5.7: Computational results for Sod shock tube problem at t = 0.2s, with HLLC numerical flux, TVB limiter and minmod function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.

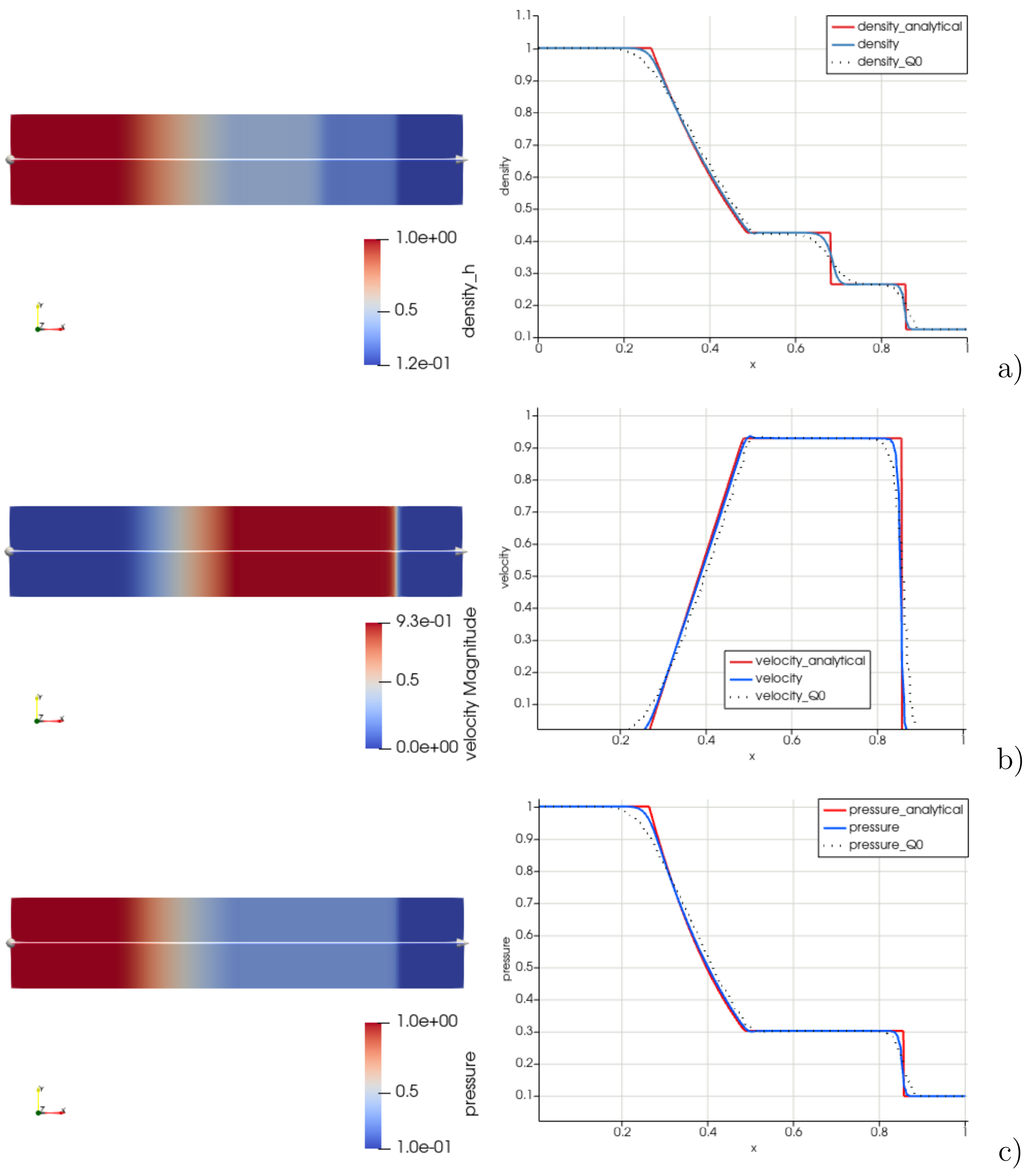


Figure 5.8: Computational results for Sod shock tube problem at t = 0.2s, with Roe numerical flux, TVB limiter and minmod function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.

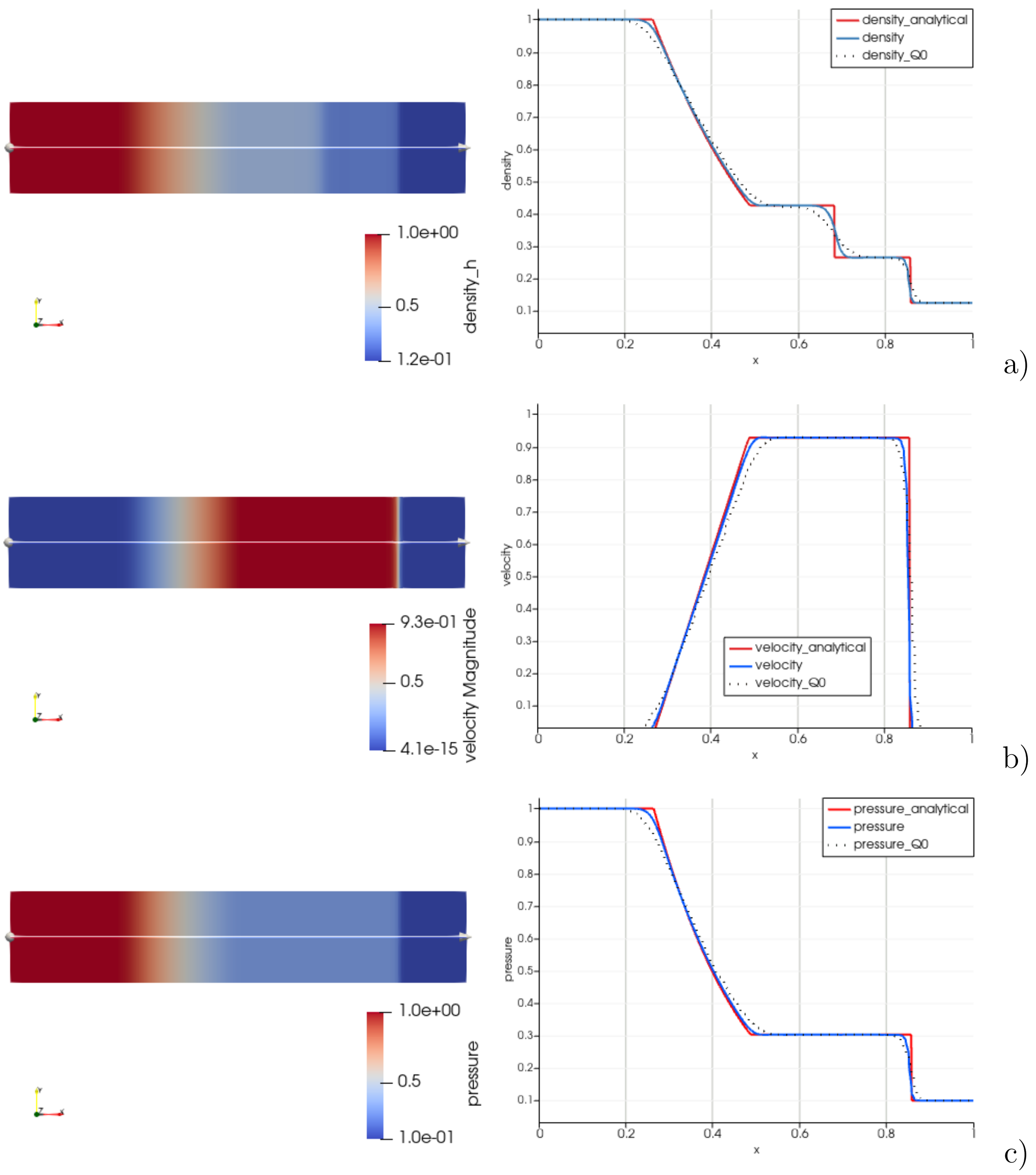


Figure 5.9: Computational results for Sod shock tube problem at t = 0.2s, with SLAU numerical flux, TVB limiter and minmod function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.

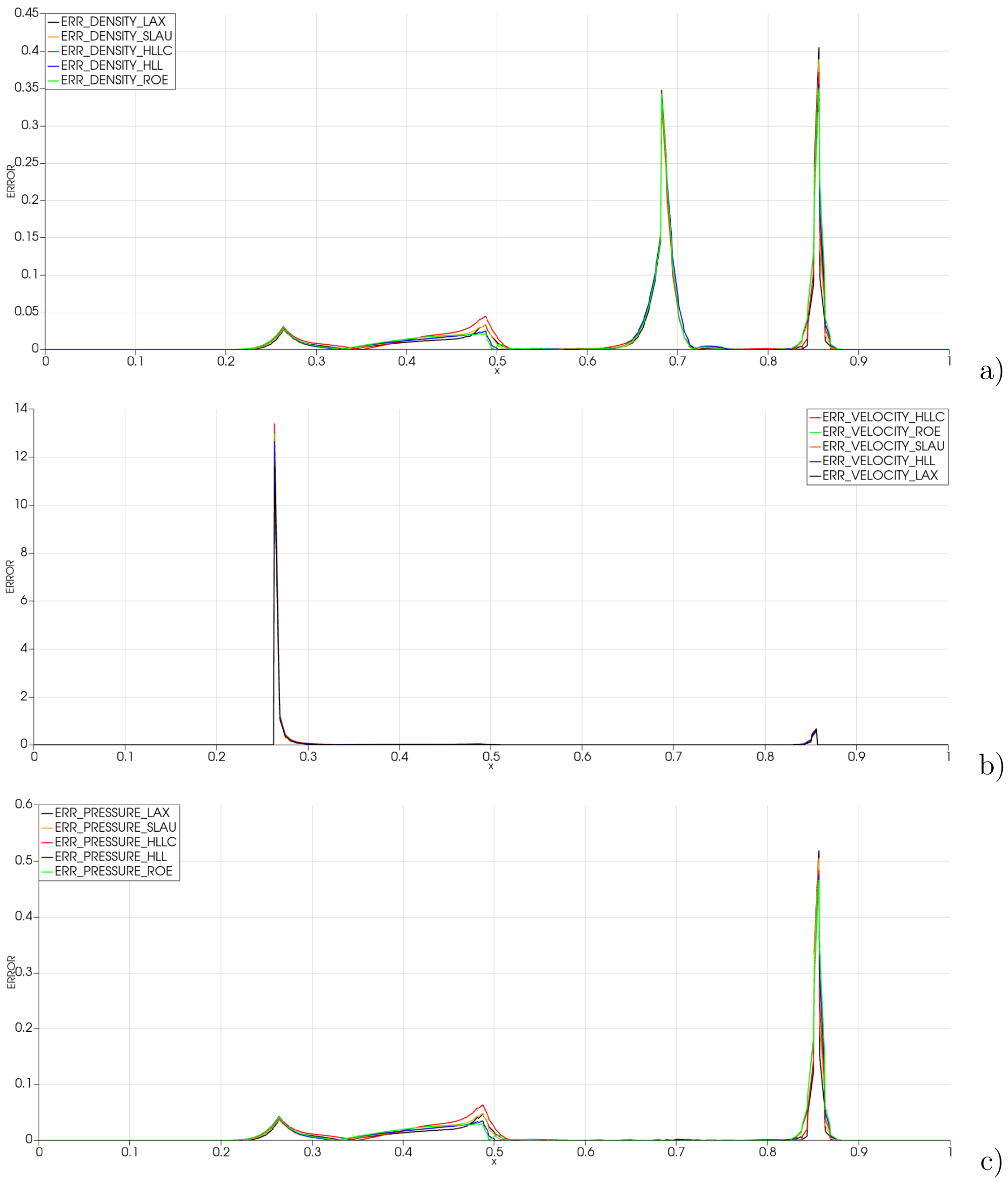


Figure 5.10: a Relative error of density, b) relative error of velocity, c) relative error of pressure of numerical solutions computed with different numerical fluxes, using TVB limiter (minmod function)

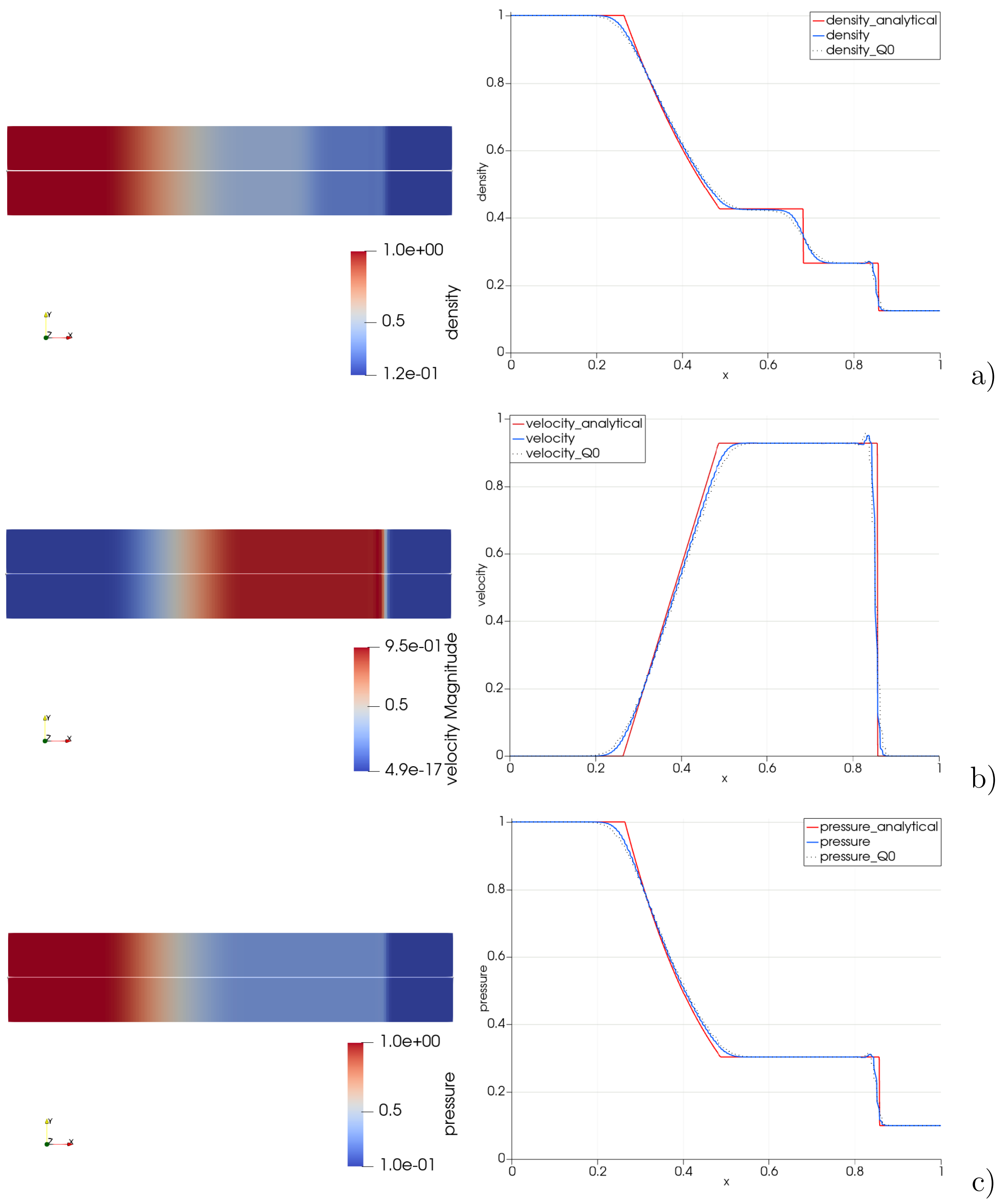


Figure 5.11: Computational results for Sod shock tube problem at t = 0.2s, with Lax-Friedrichs flux, TVB limiter and van Albada function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.

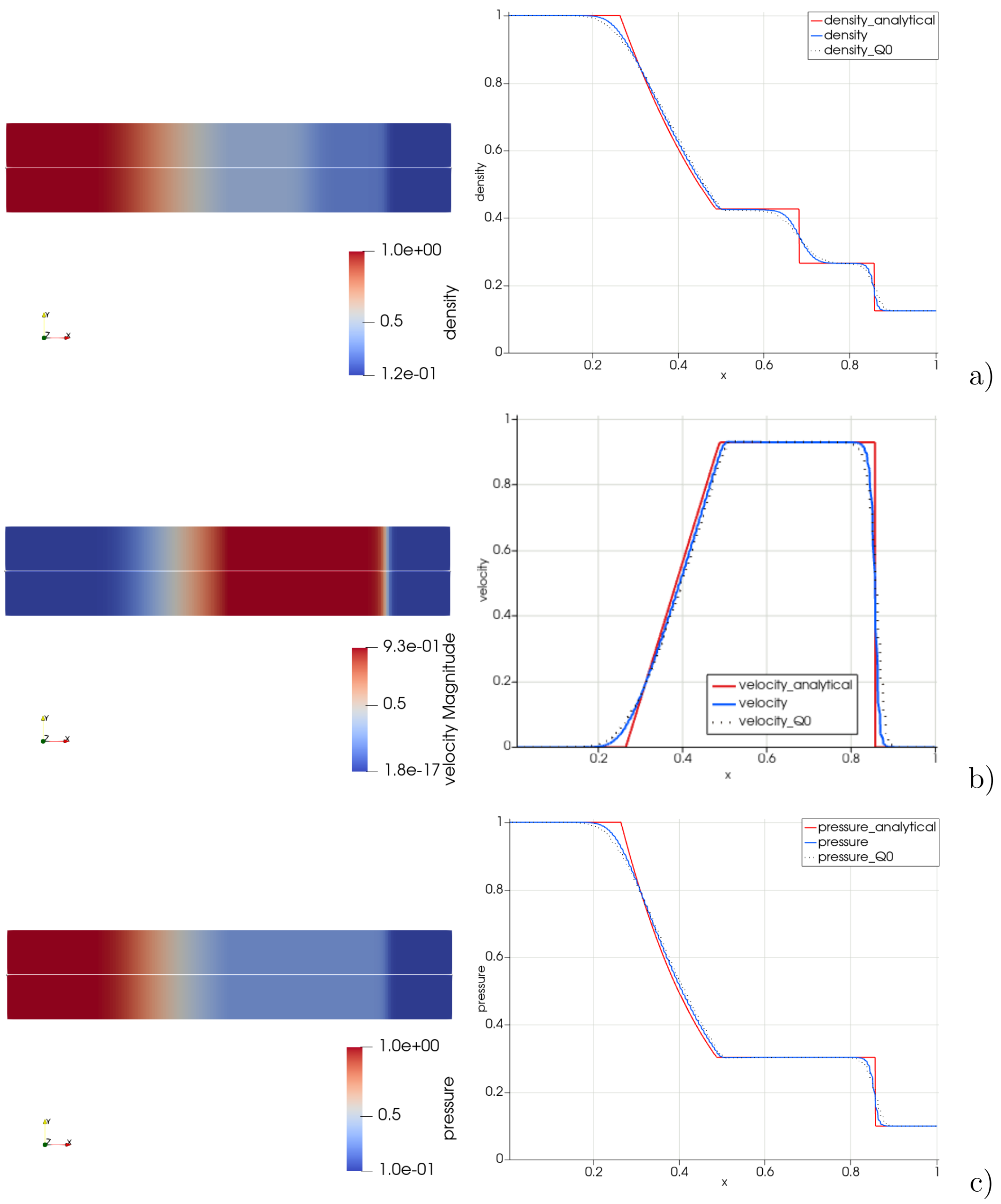
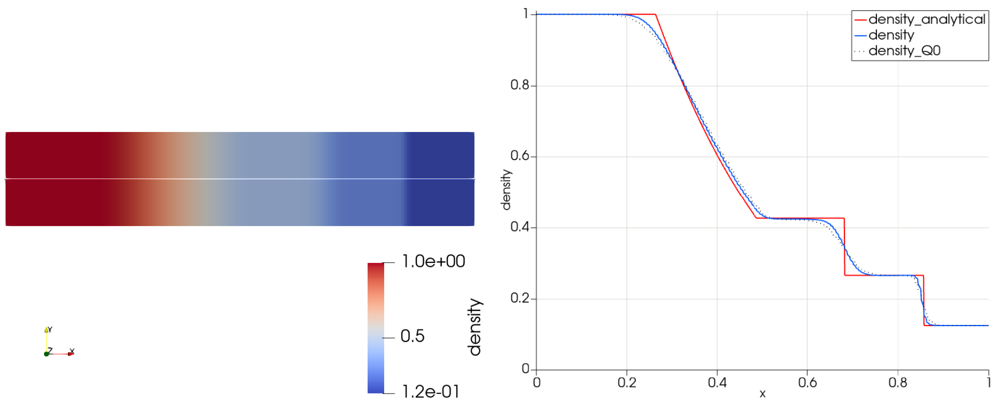
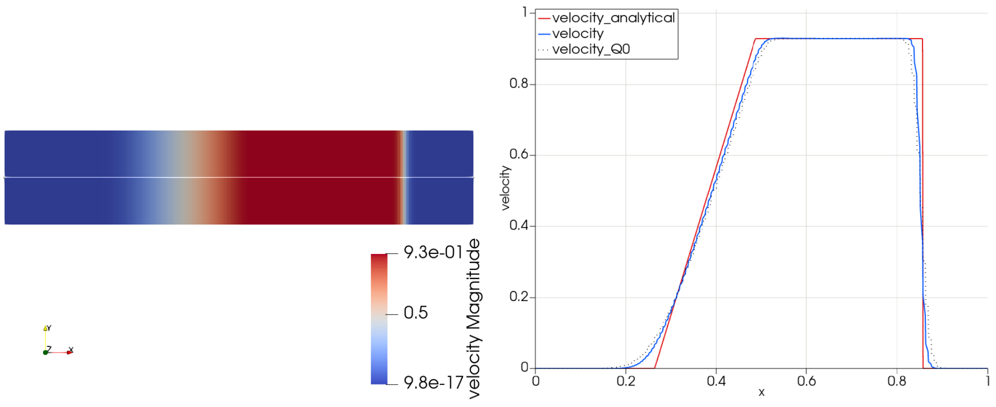


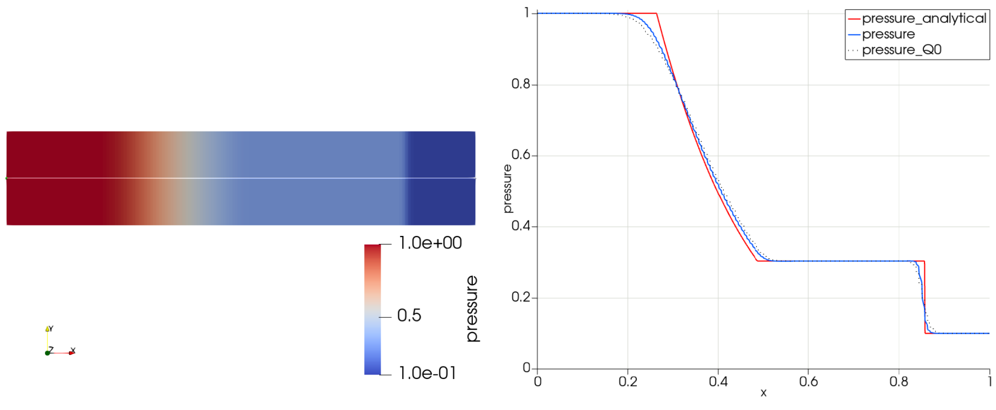
Figure 5.12: Computational results for Sod shock tube problem at t = 0.2s, with Harten Lax vanLeer flux, TVB limiter and van Albada function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.



a)



b)



c)

Figure 5.13: Computational results for Sod shock tube problem at t = 0.2s, with HLLC numerical flux, TVB limiter and van Albada function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.

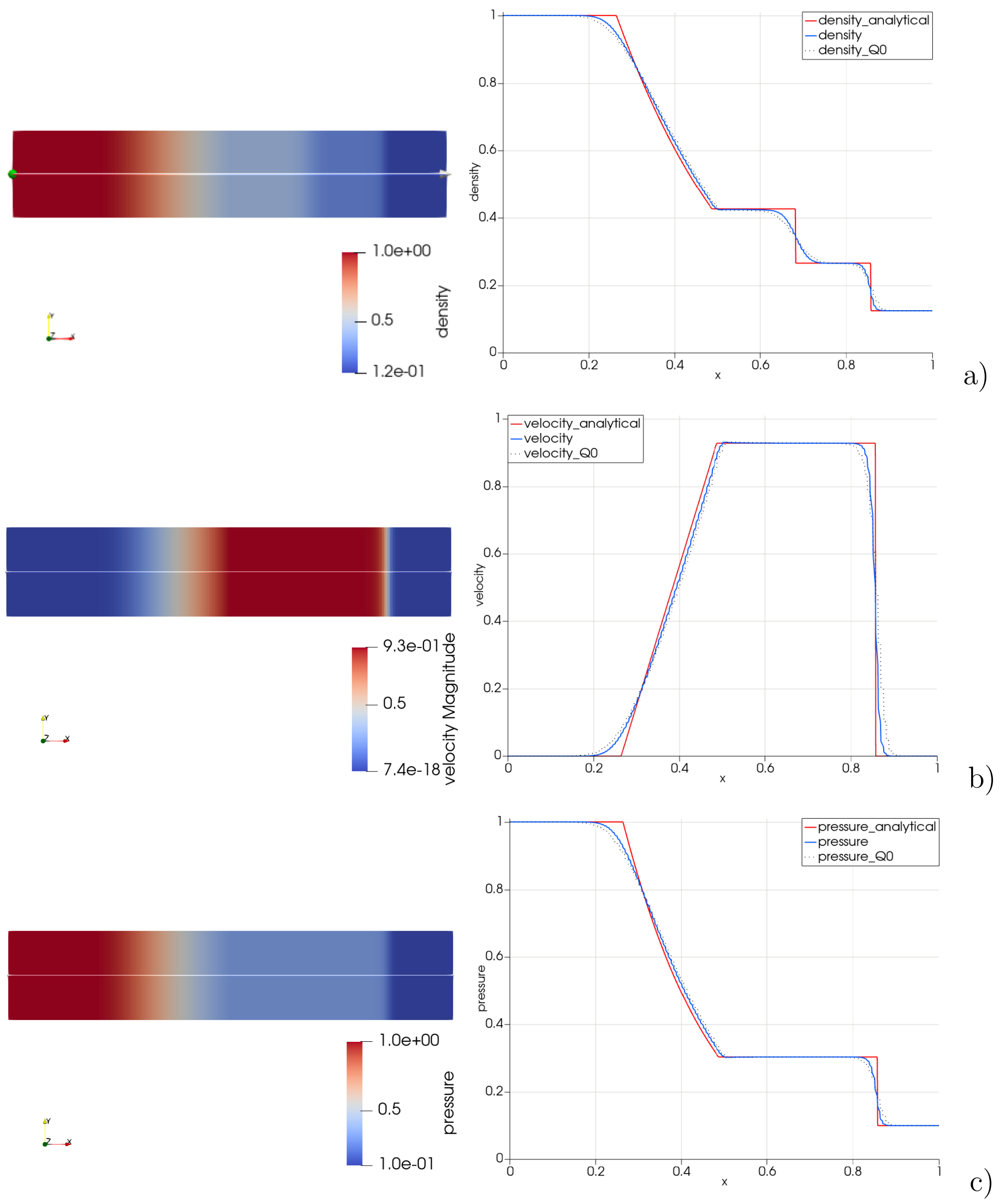


Figure 5.14: Computational results for Sod shock tube problem at t = 0.2s, with Roe flux, TVB limiter and van Albada function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.

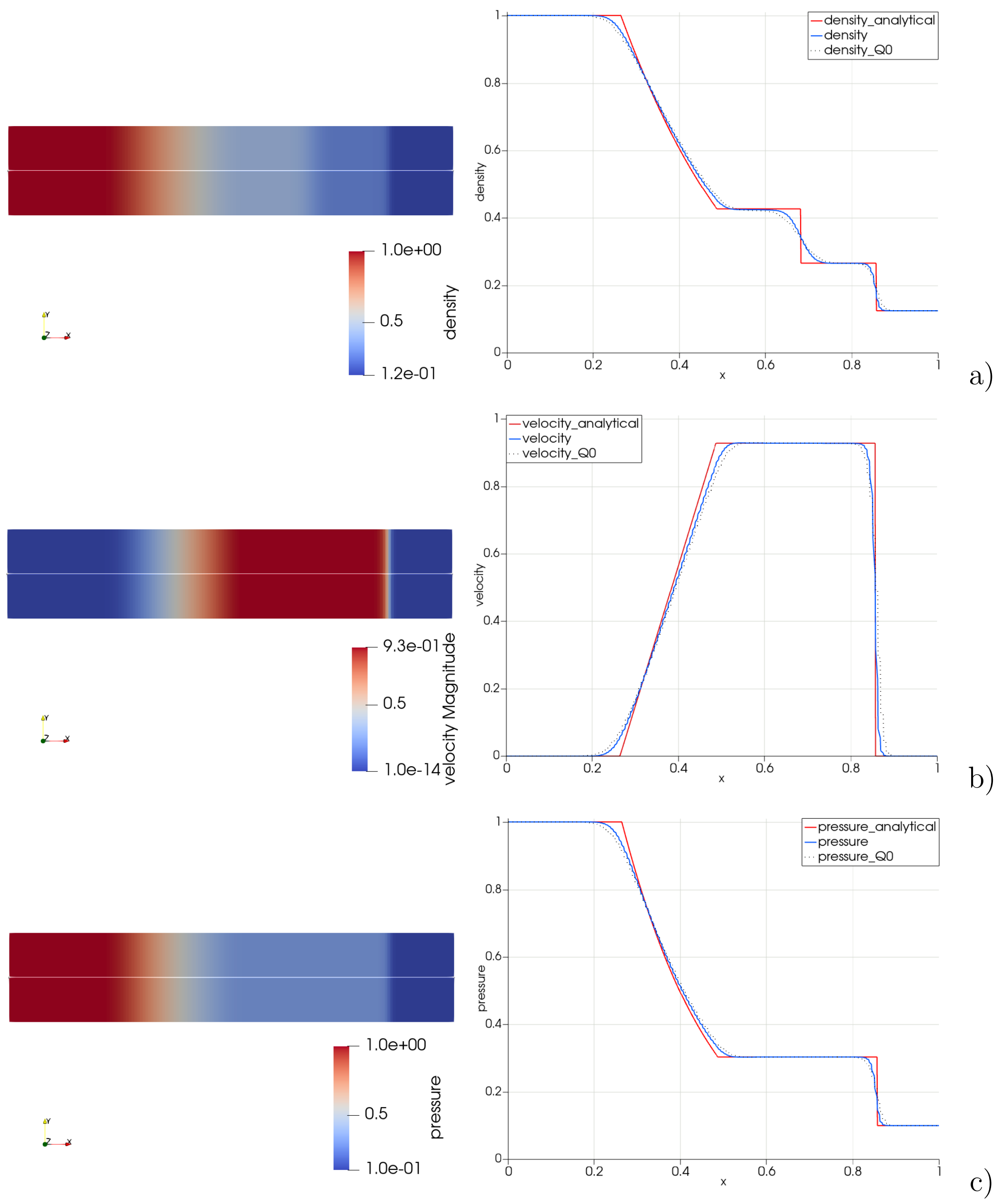
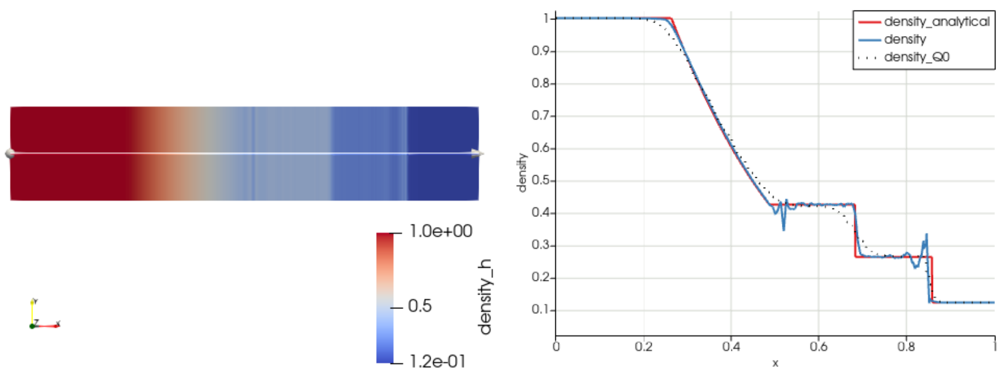
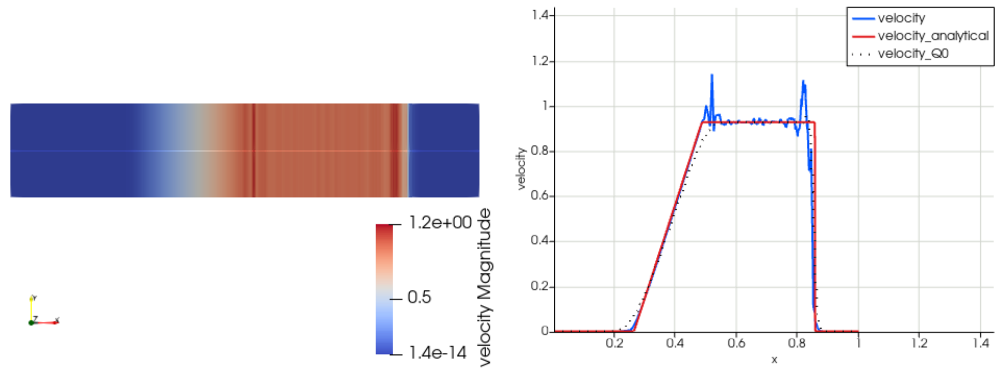


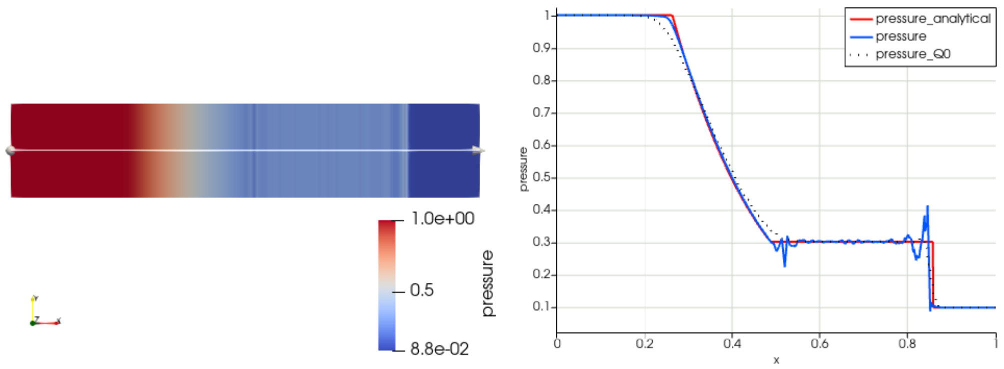
Figure 5.15: Computational results for Sod shock tube problem at t = 0.2s, with SLAU numerical flux, TVB limiter and van Albada function. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$), the dark dots denotes the $Q0$_solution.



a)

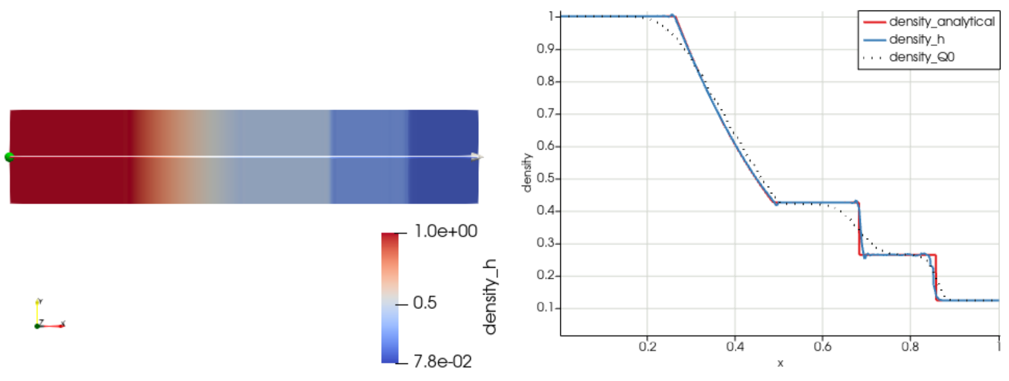


b)

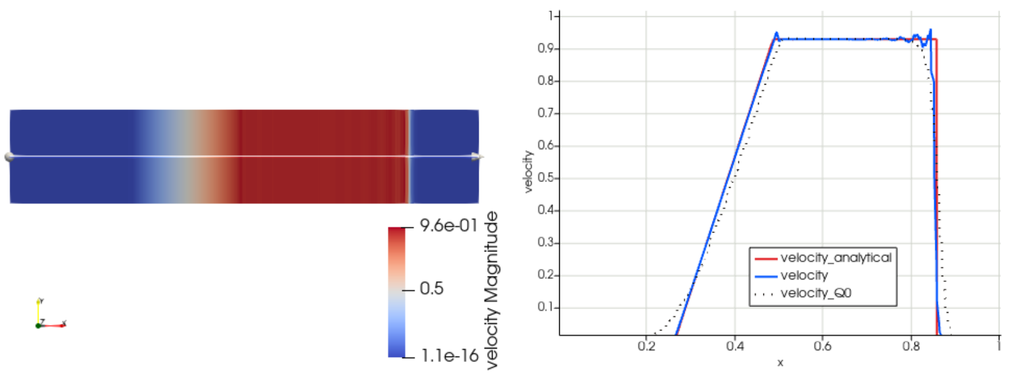


c)

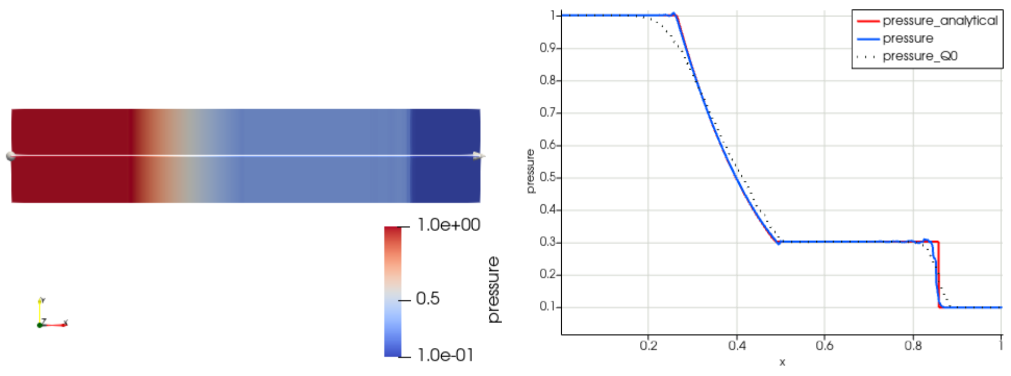
Figure 5.16: Computational results for Sod shock tube problem at t = 0.2s, with Lax-Friedrichs numerical flux, filter technique. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$, $\beta_\rho = \beta_{\rho\mathbf{u}} = \beta_{\rho E} = 0.9$), the dark dots denotes the $Q0$_solution.



a)



b)



c)

Figure 5.17: Computational results for Sod shock tube problem at t = 0.2s, with Harten Lax vanLeer numerical flux, filter technique. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$, $\beta_\rho = \beta_{\rho\mathbf{u}} = \beta_{\rho E} = 0.5$), the dark dots denotes the $Q0$_solution.

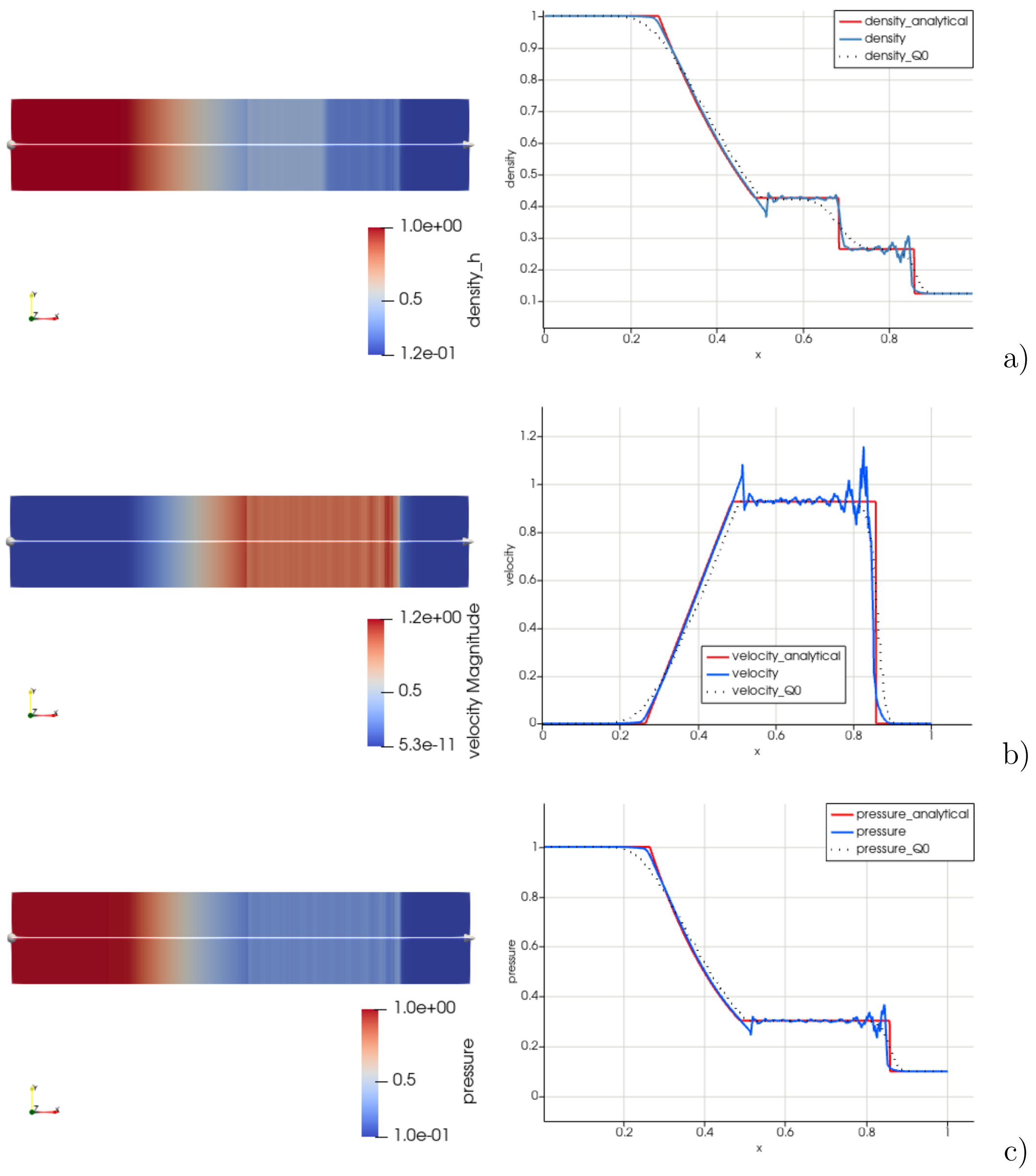
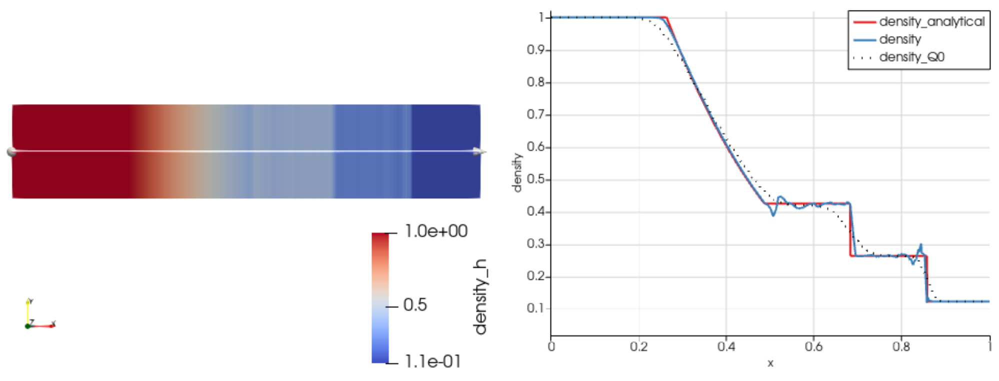
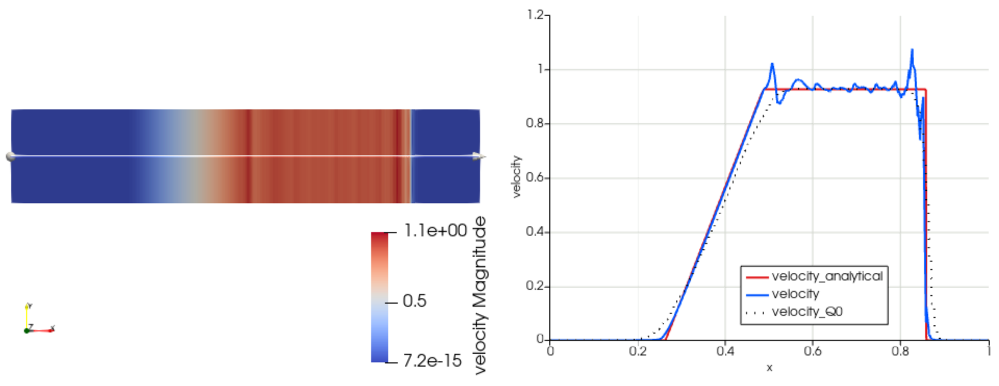


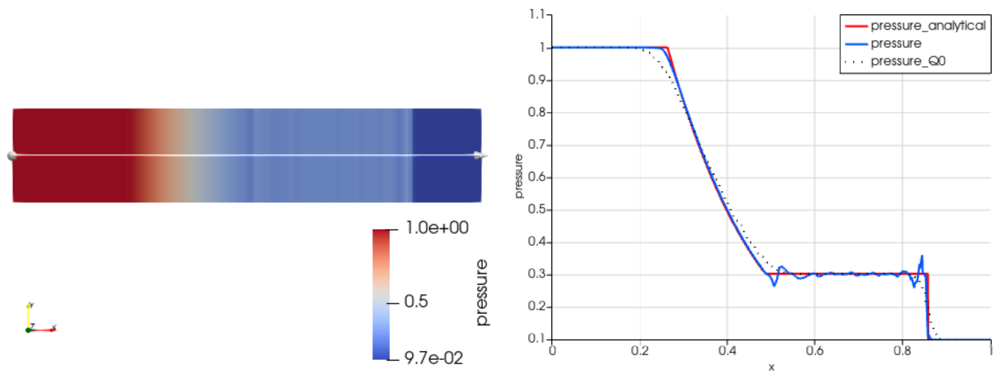
Figure 5.18: Computational results for Sod shock tube problem at t $= 0.2$s, with Roe numerical flux, filter technique. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$, $\beta_\rho = \beta_{\rho\mathbf{u}} = \beta_{\rho E} = 0.9$), the dark dots denotes the $Q0$_solution.



a)



b)



c)

Figure 5.19: Computational results for Sod shock tube problem at t = 0.2s, with SLAU numerical flux, filter technique. a) density, b) velocity, c) pressure. Left: 2D solution, Right: 1D solution plotted over line $y = 0.1$. The red line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$, $\beta_\rho = \beta_{\rho\mathbf{u}} = \beta_{\rho E} = 0.9$), the dark dots denotes the $Q0$_solution.

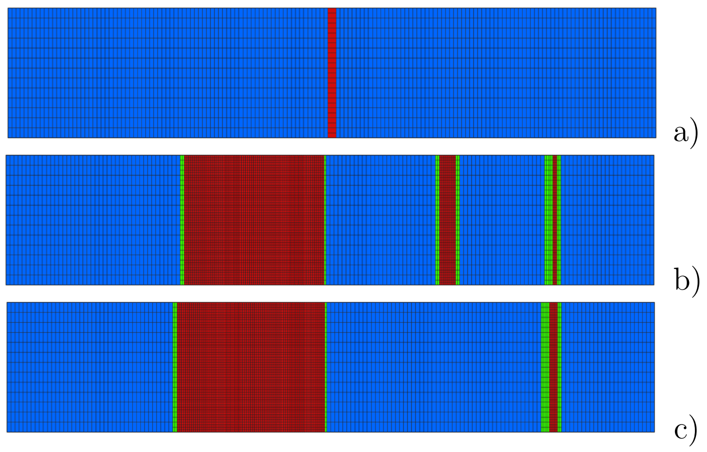


Figure 5.20: Computational meshes. a) mesh at t=0s composed by 160x13 elements; b) mesh at t=0.2s using as refinement indicator the gradient of the density; c) mesh at t=0.2s using as refinement indicator the gradient of the pressure.

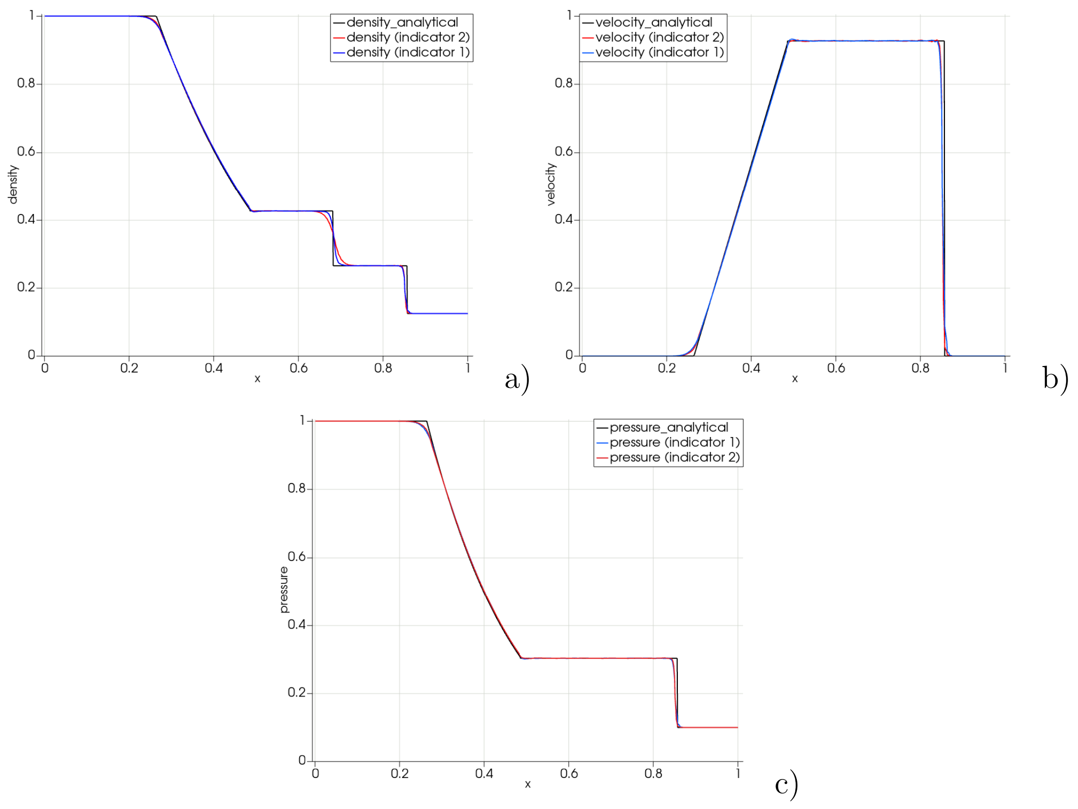


Figure 5.21: Computational results for Sod shock tube problem at t $= 0.2$s. a) density, b) velocity, c) pressure. The solution is plotted over line $y = 0.1$. The dark line reports the analytical solution, the blue line denotes the numerical solution ($k = 2$) computed using the gradient of density as h-indicator, the red dots denotes the numerical solution computed using the gradient of pressure as h-indicator.

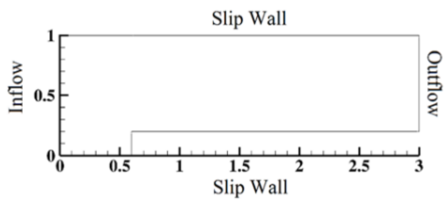


Figure 5.22: Geometry used in forward facing step problem

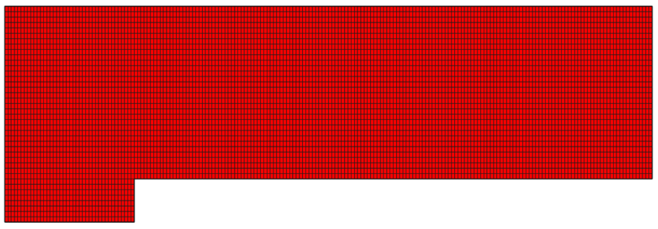
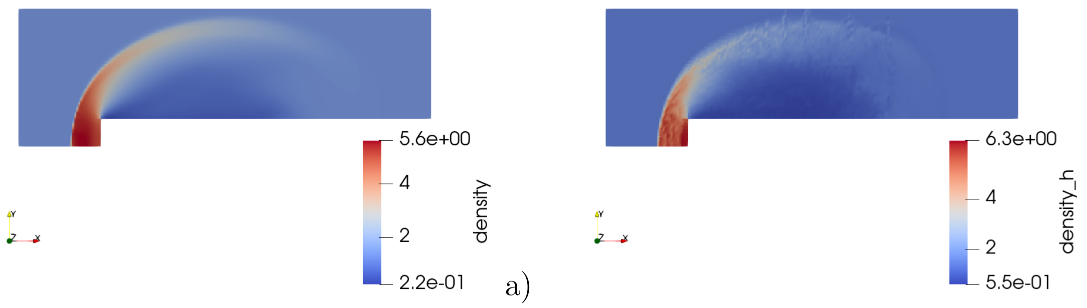


Figure 5.23: Computational mesh of FFS problem

5.2.2 Forward facing step

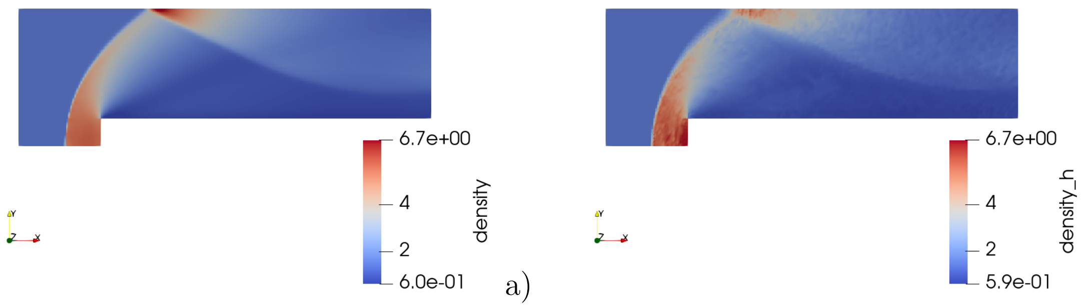
Next, the so called forward facing step (FFS) problem is considered, also proposed by Woodward and Colella in [27]. This problem is usually used to test the ability of the shock capturing method to handle strong discontinuities. A freestream at Mach 3 approaches a step as shown in Figure 5.22. A detached curve shock form in front of the step and reflects from the first top then bottom walls. The initial condition consists in a uniform gas with density $\rho = \gamma = 1.4$, pressure $p = 1$, velocity components $u_1 = 3$, $u_2 = 0$. The computational domain is given by $\Omega = [0;3]X[0;1]\backslash[0.6;3]X[0;0.2]$. Reflective boundary conditions are applied on the upper and lower boundary of the domain, whereas inflow and outflow boundary conditions are applied at the entrance and the exit. The solution of this problem involves shock waves interacting with the wall boundaries. A Cartesian meshes with cell sizes of $\Delta x = 0.015$ and $\Delta y = 0.025$ are generated and used computations (Figure 5.23). All the simulations are is achieved with HLL numerical flux. At time $t = 4s$, flow is unsteady. The steady flow has very little structure, thus it focused on more interesting flow at early times, here $t = 4s$. At $0.5s$, a detached bow shock develops ahead of the step (Figures 5.24). It curves strongly toward the upper surface of the step. The figures 5.25 at $1.0s$ shows that the bow shock strikes the upper boundary and the curvature decreases rapidly. The shock is reflected downwards from the upper boundary and strikes the upper surface of the step, as the figures 5.26 at $2.0s$ shows. A Mach reflection starts to be formed when the incident angle to the upper boundary of the domain is large enough. From $2.0s$ to $4.0s$, the triple point of incident, normal and reflected waves gradually moves upstream and to the lower surface, figures 5.27. The results are same as the standard result provided



a)

b)

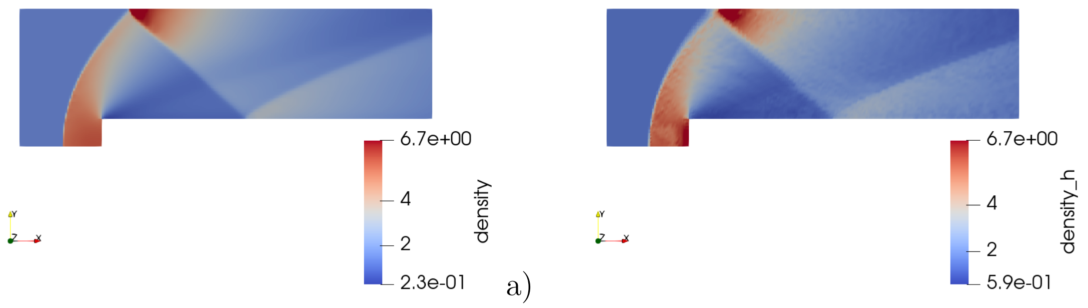
Figure 5.24: Comparisons of the obtained density a) TVB limiter and b) filter limiter at t = 0.5s



a)

b)

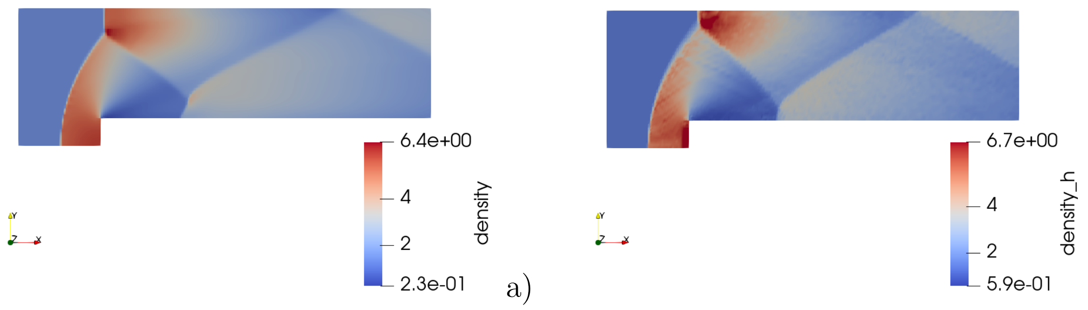
Figure 5.25: Comparisons of the obtained density a) TVB limiter and b) filter limiter at t = 1.0s



a)

b)

Figure 5.26: Comparisons of the obtained density a) TVB limiter and b) filter limiter at t = 2.0s



a)

b)

Figure 5.27: Comparisons of the obtained density a) TVB limiter and b) filter limiter at t = 4.0s

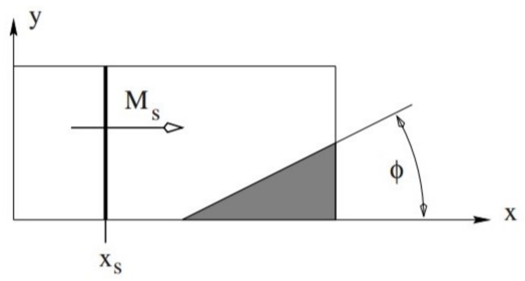


Figure 5.28: Initial Conditions for Shock Reflection from a Wedge

by Woodward [27]. For the density solution, it can be seen that strong discontinuity ahead and over the step is generated by the scheme. The major challenge is capturing the expansion from the corner of the step. `Eulero` code has accurately captured these expansion at the corner. The Mach stem, which is generated due to interaction of two shock waves, is also visible in the solution. This represent the ability of the scheme to capture Mach stems also.

5.2.3 Double Mach reflection

This test problem was presented by Woodward and Colella in [27]. Reflection of a shock wave from a wedge placed at an angle to the incident shock wave direction is shown in the figure 5.28. At an initial time a shock wave of mach no M_s, perfectly perpendicular to the $x-$direction is placed at a position x_s. The shock travels in the $x-$direction and encounters a wedge that makes an angle with the shock propagation direction. At first the simple planar shock meets the walls of the tube at right angles, but when one wall begins to slope a complicated shock reflection occurs. This test case is good to demonstrate the ability of the scheme for capturing the Mach stem well. In this problem, a Mach 10 shock moving horizontally meets a reflecting wedge inclined at 30° with horizontal and produces a double Mach reflection and a self similar flow. The horizontally moving shock is at an angle of 60° with the wedge. The results are obtained a time of 0.2 seconds. In order to perform the analysis on an orthogonal Cartesian grid, the test geometry is selected as a rectangle of 1 unit wide and 4 units long. Reflecting wall is aligned with the bottom of the rectangular geometry and starts at a distance 1/6 units from left side of rectangle. And the flow is replaced with an equivalent flow where shock is moving diagonally to the reflecting wall. The initial conditions and boundary conditions of the equivalent test set up are given below.

- Initial conditions: Since the shock is moving towards right, to the left of the shock it's haven post shock conditions and to the right side of shock it's had pre-shock conditions. For pre-shock conditions, let the density be 1.4, pressure

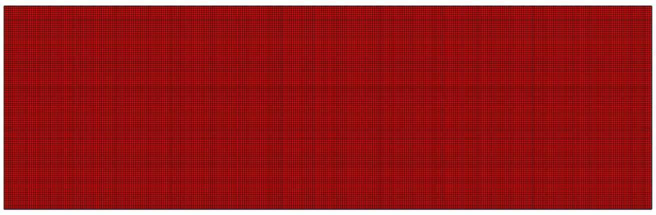


Figure 5.29: Computational mesh of double mach reflection problem

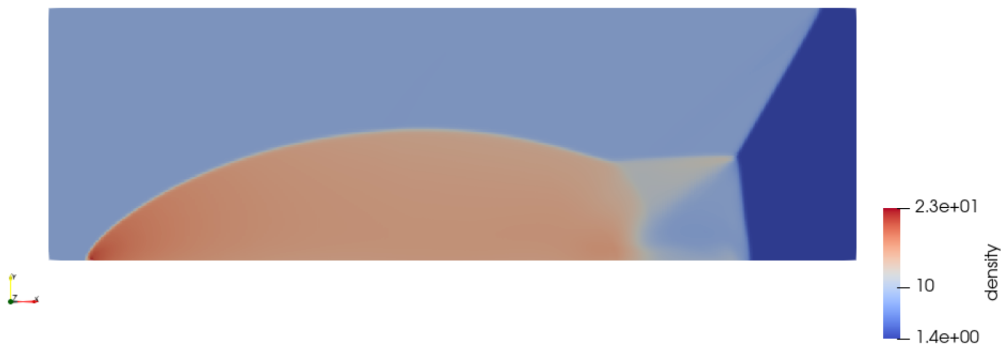


Figure 5.30: Density plot obtained with TVB limiter

be 1.0 and the corresponding post-shock values are found to be 8.0 and 116.5 respectively from the shock relations. At the beginning the shock is assumed to be passing through the left end of wedge. Therefore, the initial density is 8.0 for $x < 1/6 + y(1/\sqrt{3})$ and 1.4 for $x \geq 1/6 + y(1/\sqrt{3})$. Initial pressure is 116.5 for $x < 1/6 + y(1/\sqrt{3})$ and 1.0 for $x \geq 1/6 + y(1/\sqrt{3})$. Initial velocity $u_1 = 8.25cos(30°)$, $u_2 = -8.25sin(30°)$ for $x < 1/6 + y(1/\sqrt{3})$ and both components of velocity are zero for $x \geq 1/6 + y(1/\sqrt{3})$.

- Boundary conditions: The left side is an inflow boundary where post shock values are used. The right side is assigned with simple out flow boundary conditions. For the lower boundary ghost cells with post shock values are used for $x < 1/6$ and reflecting wall boundary condition is used for $x > 1/6$.For the top boundary ghost cells are assigned with values considering the motion of the shock. The intersection of the diagonal shock with top boundary moves at a speed of $10/cos(30°)$ and the location of this intersecting point at a time t, $x_s(t) = 1/6 + (1 + 20t)/\sqrt{3}$. So the top boundary ghost cells are assigned with post shock values for $x < x_s$ and pre-shock values for $x > x_s$.

The results obtained for this test case are as given in Figure 5.30 and 5.31. This is one of the difficult problem to simulate due to complex flow patterns. But the schemes have accurately developed the double Mach reflection of the shock at the wall. Two Mach stems form, with two contact discontinuities. The second con-

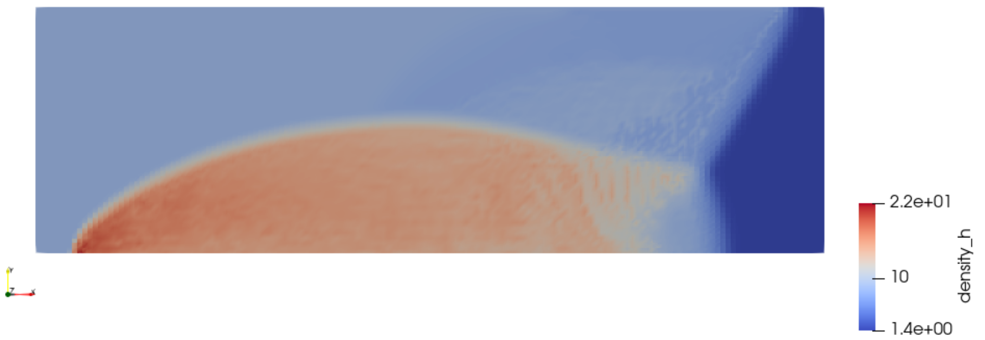


Figure 5.31: Density plot obtained with filter technique, $\beta_\rho = \beta_{\rho\mathbf{u}} = \beta_{\rho E} = 0.1$

tact discontinuity is extremely weak. The density contours generated matches that obtained by Woodward in his paper.

5.2.4 2D Riemann problem

In this section, a 2D Riemann problem, corresponding to the Configuration 4 proposed in [7], is proposed. The computational domain is $\Omega = [0,1]^2$ and the initial conditions are given by

$$(\rho_0, u_0, v_0, p_0) = \begin{cases} (1.1, 0, 0, 1) & if\ x > 0.5;\ \ y > 0.5 \\ (0.5065, 0.8939, 0, 0.35) & if\ x < 0.5;\ y > 0.5 \\ (1.1, 0.8939, 0.8939, 1.1) & if\ x < 0.5;\ \ y < 0.5 \\ (0.5065, 0, 0.8939, 0.35) & if\ x > 0.5;\ \ y < 0.5 \end{cases} \tag{5.1}$$

The final time is $t = 0.25s$. The initial mesh is composed by 80 along each direction. High order polynomial degree $k = 2$ are considered. Figure **??** shows the results obtained for the density using $\beta = 0.25$. The filter tends to add more dissipation than needed, but this is necessary to avoid large undershoots and overshoots and more in general oscillations in which completely corrupt the unfiltered solution. While not optimal, the results highlight the robustness of the proposed approach and show that the primary goal of the filter, namely avoid or at least reduce the oscillations, is achieved. Moreover, as pointed out in **??** the effects of Kelvin-Helmholtz instability with several small-scale features emerge at high resolution along the diagonal of the cocoon structure and this confirms that the test is particularly challenging. The TVB limiter confirms its power also for this test, obtaining the result proposed in [7].

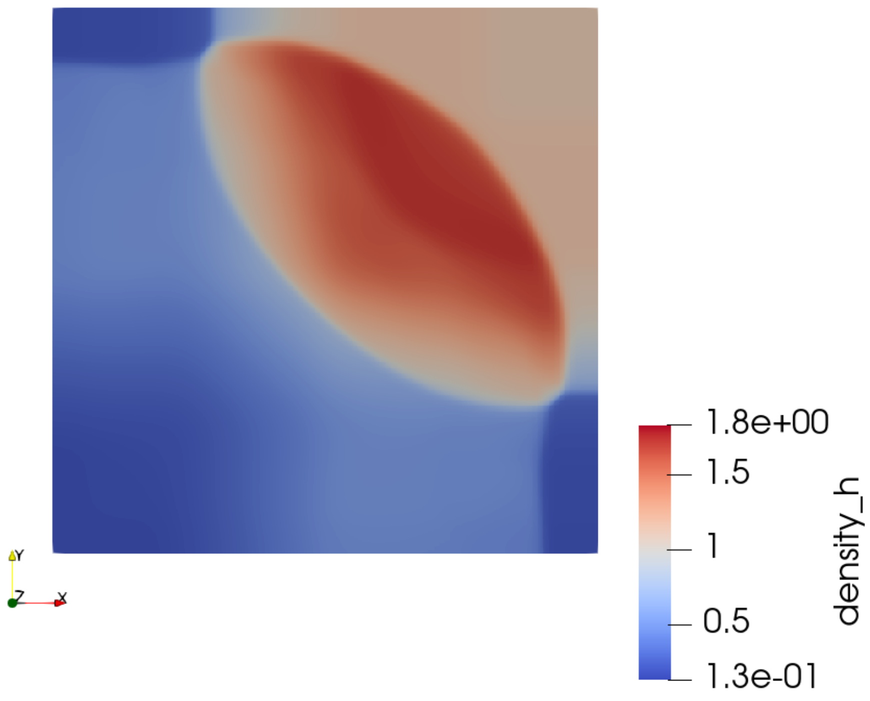


Figure 5.32: Density plot obtained with TVB limiter, HLL flux, and minmod function at $t = 0.2s$

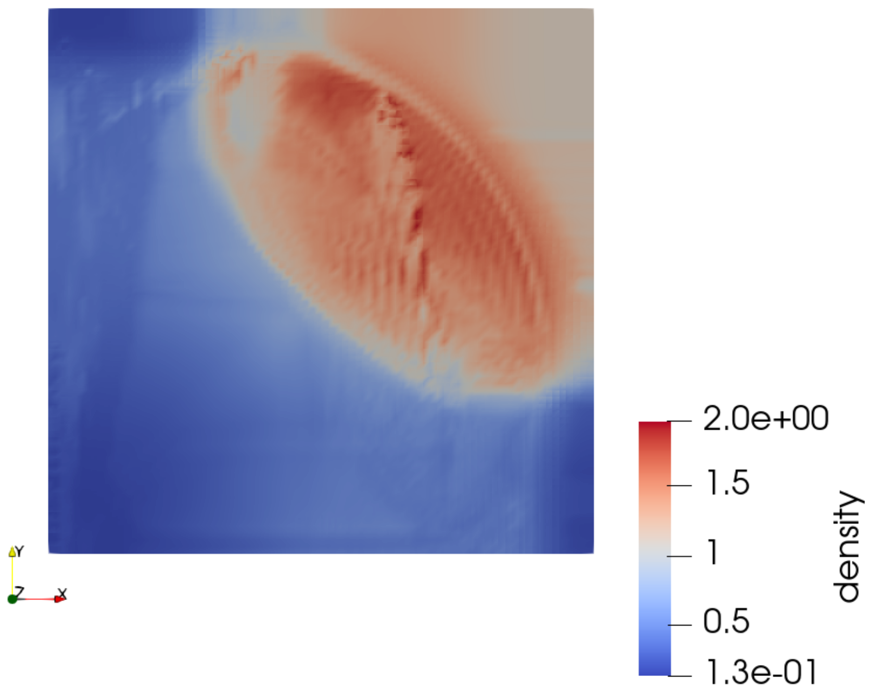


Figure 5.33: Density plot obtained with filter techniques, HLL flux and $\beta = 0.2$.

Chapter 6

Conclusions and perspectives

An implementation based on `deal.II C++` libraries was developed to numerically solve the compressible Euler equations of gas dynamics. The implementation uses a discontinuous Galerkin scheme. Different tools to control spurious oscillations around discontinuous solutions were reviewed. The tools mainly used for the numerical results were a TVDtype limiter and the KXRCF shock indicator. The KXRCF shock indicator seems to detect well the locations of the shocks in the 2D examples. The implementation of the shock indicator can be justified as it avoids the usage of limiters all over the domain, which in turn may lead to an improvement in the efficiency of the algorithm. The numerical experiments with the scheme were performed using an explicit third order SSP Runge-Kutta time integration, in combination especially with a Harten Lax vanLeer. Furthermore, a filtering technique for obtaining a monotonic Discontinuous Galerkin discretization of hyperbolic equations has been presented. The scheme is inspired by the approach originally proposed in [20] and it is based on a filter function that keeps the high order solution if it is regular and switches to a monotone low order approximation otherwise, according to the value of one or more parameters. Its potential has been demonstrated in a number of classical benchmarks for Euler equations, but in future works, a development concerns the tuning of the parameters is necessary, in order to use it for other type of numerical fluxes.

In the future, a new test-case will be analyzed in detail. In particular, the case of the supersonic flow past a circular cylinder will be implemented, in order to be able to see, with the various numerical flux, up to which mach number one can go before obtaining the carbuncle phenomenon.

Bibliography

[1] P.D.Lax A.Harten and B.Van Leer. “On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws”. In: *SIAM Review* 25 (1983), pp. 3–61.

[2] S. Adjerid and T. C. Massey. “Superconvergence of discontinuous galerkin solutions for a nonlinear scalar hyperbolic problem”. In: *Computer Methods in Applied Mechanics and Engineering* 195 (2006), pp. 331–334.

[3] L. Krivodonova et all. “Shock detection and limiting with discontinuous galerkin methods for hyperbolic conservation laws”. In: *Applied Numerical Mathematics* 48 (2004), pp. 323–338.

[4] Daniel Arndt et all. “The `deal.II` Library, Version 9.4”. In: *Journal of Numerical Mathematics* 30.3 (2022), pp. 231–246. DOI: `10.1515/jnma-2022-0054`. URL: `https://dealii.org/deal94-preprint.pdf`.

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

[6] S.Y. Lin B. Cockburn and C.W. Shu. “Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws III: Onedimensional systems”. In: *Journal of Computational Physics* 84 (1989), pp. 90–113.

[7] A.M. Oberman B.D. Froese. “Convergent filtered schemes for the Monge-Ampère partial differential equation”. In: *SIAM J. Numer. Anal.* 51 (2013), pp. 423–444.

[8] Y. Cheng and C.W. Shu. “Superconvergence of discontinuous galerkin and local discontinuous galerkin schemes for linear hyperbolic and convection-diffusion equations in one space dimension”. In: *SIAM Journal on Numerical Analysis* 47 (2010), pp. 4044–4072.

[9] B. Cockburn and C.W. Shu. “The runge-kutta discontinuous galerkin method for conservation laws V: multidimensional systems”. In: *Journal of Computational Physics* 141 (1998), pp. 199–224.

[10] B. Cockburn and C.W. Shu. “Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws. II. general framework.” In: *Mathematics of computation* 52 (1989), pp. 411–435.

[11] Luskin Cockburn, Shu, and Suli. “Enhanced accuracy by postprocessing for finite element methods for hyperbolic equations”. In: *Mathematics of Computation* 72 (2003), pp. 577–606.

[12] M. Spruce E. F. Toro and W. Speares. “Restoration of the Contact Surface in the HLL–Riemann Solver”. In: *Department of Aerospace Science, UK* 43 (1992), pp. 357–372.

[13] Bram van Leer Gale Dick van Albada and WW Roberts Jr. “A comparative study of computational methods in cosmic gas dynamics”. In: *Astronomy and Astrophysics* 108 (1982), pp. 76–84.

[14] S. K. Godunov. “A Finite Difference Method for the Computation of Discontinuous Solutions of the Equations of Fluid Dynamics”. In: *Mat. Sb.* 47 (1959), pp. 357–393.

[15] A. Harten. “High Resolution Schemes for Hyperbolic Conservation Laws”. In: *J.Comput. Phys.* 49 (1983), pp. 357–393.

[16] R. J. LeVeque. *Numerical Methods for Conservation Laws.* 1992.

[17] Meng-Sing Liou. “A Sequel to AUMS : $AUMS^{+}$”. In: *Journal of Computational Physics* 129 (1996), pp. 364–382.

[18] Meng-Sing Liou. “A sequel to AUSM, Part II: $AUSM^{+} - up$ for all speeds”. In: *Journal of Computational Physics* 214 (2006), pp. 137–170. DOI: `doi : 10.1016/j.jcp.2005.09.020`.

[19] Feng Qu Nan Li and Di Sun. “An Effective AUSM-Type Scheme for Both Cases of Low Mach Number and High Mach Number”. In: *Appl. Sci.* 5464 (2022), pp. 113–125. DOI: `https://doi.org/10.3390/app12115464`.

[20] M. Falcone O. Bokanowski and S. Sahu. “An efficient filtered scheme for some first order time dependent Hamilton Jacobi equations”. In: *SIAM Journal on Scientific Computing* 38 (2016), pp. 171–195.

[21] Giuseppe Orlando. “A filtering monotonization approach for DG discretizations of hyperbolic problems”. In: *Computers & Mathematics with Applications* 109 (Jan. 2023), pp. 113–125. DOI: `https://doi.org/10.1016/j.camwa.2022.11.017`.

[22] P. L. Roe. “Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes”. In: *J. Comput. Phys.* 43 (1981), pp. 357–372.

[23] Sandro Salsa. *Equazioni a derivate parziali: Metodi, modelli e applicazioni. Chapter 4.* Mathematical Engineering. Springer Verlag, 2016. ISBN: 8847057833.

[24] David I. Ketcheson Sigal Gottlieb and Chi Wang Shu. “High Order Strong Stability Preserving Time Discretizations”. In: *J Sci Comput* 38 (2009), pp. 251–289. DOI: `10.1007/s10915-008-9239-z`.

[25] Eleuterio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction.* Springer Verlag, 2009. ISBN: 978-3-540-25202-3.

[26] *tutorial 67 di* ***deal.II***. URL: `https://www.dealii.org/current/doxygen/deal.II/step_67.html`.

[27] P. Woodward and P. Colella. *The numerical simulation of two dimensional fluid flow with strong shocks.* Vol. 54. 1984, pp. 115–173.

[28] X. Zhang and C.W. Shu. “On maximum-principle-satisfying high order schemes for scalar conservation laws”. In: *Journal of Computational Physics* 229 (2010), pp. 3091–3120.

[29] X. Zhang and C.W. Shu. “On positivity-preserving high order discontinuous galerkin schemes for compressible euler equations on rectangular meshes”. In: *Journal of Computational Physics* 229 (2010), pp. 8918–8931.

[30] X. Zhang and C.W. Shu. “Positivity-preserving high order discontinuous galerkin schemes for compressible euler equations with source terms”. In: *Journal of Computational Physics* 230 (2011), pp. 1238–1248.