ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

High-order ADER-Discontinuous Galerkin Method for Solving the Compressible Euler Equations with AMR

Fabiano Sasselli
Master of Science in Computational Science and Engineering

Master Thesis

September 18, 2023

Advisors: Dr. Roger Käppeli
Department of Mathematics, SAM, ETH Zürich

Abstract

This thesis presents a block structured Adaptive Mesh Refinement (AMR) software framework which employs a modal formulation of the Discontinuous Galerkin (DG) method coupled with Arbitrary Derivatives (ADER) time integration in order to solve general multi-dimensional systems of hyperbolic PDEs with high-accuracy in space and time.

It has been developed with astrophysical applications in sight, therefore as a first step, the compressible Euler equations are solved. Of interest in this field is the ability to accurately resolve discontinuities such as shocks. For this reason the TVB limiter has been implemented as shock detector.

The framework utilizes an MPI+OpenMP parallelization strategy and is built on the AMReX C++ library, which provides distributed data structures, classes and functions required for refinement and classic mesh operations.

The numerical methods and techniques used are presented. The solver implementation is validated through a series of tests problems that show its ability to generate high order accurate solutions in space and time, track and refine discontinuities (shocks) and also apply limiting locally.

In the context of DG applied to astrophysics, the conservation of angular momentum is an important requirement, which is satisfied by the scheme but violated when limiters are introduced. To this end a technique to recover the angular momentum after limiting is proposed.

Contents

Chapter 1

Introduction

Complex, dynamical astrophysical interactions involving hypersonic fluids [46],[28], [40] are characterized by conservative physical interactions [4], a wide range of spatio-temporal scales [26],[4] and the presence of discontinuities in the form of shocks [4], [7]. These characteristics make the use of systems of hyperbolic PDEs for their description a suitable choice.

The Discontinuous Galerkin (DG) method, popularized by Cockburn and Shu in a collection of papers [40], [41], [42],[27], is a cell local finite element method that solves the weak formulation of a hyperbolic PDE [26]. This ensures that it preserve conservation of the physical system. DG also provides great sub-cell resolution [23] thanks to its ability to be easily extended to high-order of accuracies [28]. Unfortunately, the high order accuracy comes with the cost of increased risk of developing spurious oscillation near discontinuities [36]. Currently, the solution to this problems involve the utilization of either pre or post-processing filters (limiters) to apply to the solution in order to reduce or remove the oscillatory components [34]. This approaches has been shown to work well, but there is the risk that the approximated solution in the cell is modified to the point of violating conservation or/and positivity of the physical quantities it represents [26]. For example, the angular momentum, which is an important variable for cosmological models, loses its conservation when limiters (filters) are applied to the linear momentum variables [7].

High spatial order is not enough if not accompanied by an equivalent high order in time [26]. The modern version of the Arbitrary Derivatives (ADER) time integration allows this and can be applied to DG in a very elegant way. Still, a fine resolution is needed if we want to capture all the feature of the flows we are studying [29]. Employing a strategy based on the high order capabilities of DG mixed with the efficient localized refinement provided by

Adaptive Mesh refinement (AMR) seems a promising choice. In addition, since in DG the interaction with the rest of the domain is very limited [28], the scheme is suitable for extreme parallelization and to be applied to unstructured meshes and AMR [7],[26], [37]

The presented thesis is a step towards this goal and aims at laying the foundations for the creation of a robust and flexible AMR software framework that can effectively be used in a research environment. For this reason a parallel implementation of the DG method to solve the compressible Euler equations using the AMR capabilities offered by AMReX has been created. To note that similar AMR-ADER-DG implementations have been done throughout the years by other authors in the context of Euler and MHD equations (non comprehensive list) [25], [23], [50], but their code is not available.

The structure of this document is the following: in chapter 2 the ADER-DG scheme is derived and all implementation relevant aspects are discussed. Chapter 3 presents the concept of AMR and the techniques required for a DG implementation. Some further remarks regarding the implemented software are made. In chapter 4 and 5 the considered equations are presented and the results of the numerical tests discussed. Finally, in chapter 6 concluding remarks are made.

The main component of this thesis has been code development, therefore limited time has been allocated for in-depth simulation campaign and parameter testing. Unfortunately this type projects have time restrictions and therefore not everything can be done.

Chapter 2

Modal ADER-DG scheme

2.1 Overview

The original formulation of ADER, proposed by Toro and Titarev [44], [45], [46] was based on expanding the time derivative of the PDE through a Taylor series up to the the desired order, and then utilize the Cauchy-Kovalewski procedure to replace the temporal derivatives with spatial derivatives by directly using the governing PDE [23], [25], [37]. This approach was hard to generalize to PDE systems of high order [25] and could not handle stiff source terms [37]. A modern reformulation by Dumbser and Balsara [47], [48], [49] has proven to be more flexible and elegantly formulated. It utilizes a cell local evolution, based on the weak integral form of the PDE [25], of a space-time predictor which takes also into consideration the initial conditions [23], [25], [37]. The predictor is utilized to evaluate the time evolution of the volume and boundary fluxes. This new version is also able to deal with stiff terms [37]. Compared to the more common Runge-Kutta schemes, ADER schemes are one-step methods which don't require the evaluation and storage of multiple stages [25]. In addition, in the context of AMR, ADER requires drastically less communication between levels during a single time step [25]. The formulation of the ADER-DG scheme for systems of hyperbolic PDEs is equivalent to evaluating the system component by component and then applying the scalar formulation of the method component-wise [6]. Therefore the following derivations are presented for the scalar case.

2.2 Discontinuous Galerkin corrector

Let $\Omega \subset \mathbb{R}^D$, D=1,2,3, be a bounded domain with cartesian coordinates. Let denote with T the final time and $\partial\Omega$ the domain boundary. We are considering the following IVP: find $u : \Omega \times (0,T) \to \mathbb{R}$ s.t [5]:

$$\frac{\partial}{\partial t}u(\mathbf{x},t) + \nabla \cdot \mathbf{f}(u(\mathbf{x},t),\mathbf{x},t) = s(u(\mathbf{x},t),\mathbf{x},t) \;\;,\;\; \Omega \times (0,T) \tag{2.1}$$

$$u(\mathbf{x},0) = u_0(\mathbf{x}) \;\;,\;\; \mathbf{x} \in \Omega \tag{2.2}$$

with flux $\mathbf{f} = (f_1,...,f_D) \in C^1(\mathbb{R}^D)$, source term s and the number of dimensions given by D. We approximate the domain with a block structured cartesian mesh made of K non overlapping discrete elements (overlaps only at the edge [3]), i.e quadrilaterals (D=2) or tetrahedrons (D=3) [5]:

$$\Omega \simeq \Omega_h = \bigcup_{k=1}^{K} D^k \tag{2.3}$$

The $p+1$-order accurate [36] piece-wise continuous approximation $u_h(\mathbf{x},t)$ of the global solution $u(\mathbf{x},t)$ is constructed from the individual cells local solution $u_{h,k}(\mathbf{x},t)$ [5]:

$$u(\mathbf{x},t) \simeq u_h(\mathbf{x},t) = \sum_{k=1}^{K} u_{h,k}(\mathbf{x},t) \tag{2.4}$$

where the element D^k local solution is further defined as a linear combination of smooth polynomial basis functions $\varphi_i^k(\mathbf{x})$ of order maximum p and their associated coefficients $\hat{u}_k^i$ [28]:

$$u_{h,k}(\mathbf{x},t) = \begin{cases} \sum_{i=1}^{N_p} \hat{u}_k^i(t)\varphi_i^k(\mathbf{x}) \;\;;\;\; \mathbf{x} \in D^k \\ 0 \;\;;\;\; else \end{cases} \tag{2.5}$$

The superscript k of the basis functions $\varphi_i^k(\mathbf{x})$ indicates that they are defined w.r.t the cell global coordinates. This is relevant because, as it will be shown later, a change of coordinates $\mathbf{x} \in D^k \to \xi \in [-1,1]^D$ is usually made so that the solution is expressed w.r.t a cell reference element. Then the basis function is written without the cell index: $\varphi_i^k(\mathbf{x}) = \varphi_i(\xi)$ [28].

The core idea of DG is the evolution of the cell local solution and its correct interaction with neighbouring cells by solving an integral weak formulation of the conservation law [5]. To obtain the global solution approximation we begin by solving the conservation law on the local element D^k [5]:

$$\frac{\partial}{\partial t}u_{h,k}(\mathbf{x},t) + \nabla \cdot \mathbf{f}(u_{h,k}(\mathbf{x},t),\mathbf{x},t) - s(u_{h,k}(\mathbf{x},t),\mathbf{x},t) = R_{h,k} \tag{2.6}$$

The residual should be minimized. This is achieved by defining a set of globally piecewise smooth test functions, which are locally defined in the same way as the local solution polynomial [5], i.e:

$$v_{h,k}(\mathbf{x},t) = \sum_{j=1}^{N_p} \hat{v}_k^j(t)\varphi_j^k(\mathbf{x}) \;\;;\;\; \mathbf{x} \in D^k \tag{2.7}$$

By using test functions defined w.r.t the same basis as our local solution polynomial we are be able to obtain N_p equations for the N_p unknown solution coefficients [15]. The local nature of the computations makes the interaction of the solution between neighbouring elements not straight forward compared e.g to the continuous Galerkin method, where the basis functions have overlapping support and information exchange between cells is obtained by construction [3],[5], [14]. After the weak DG formulation will be derived, it will be clear how this problem is solved. Continuing the derivation, the governing equation is projected into the members of the basis set [14] by imposing orthogonality w.r.t the residual and integrating in time, i.e we integrate over the discrete space-time control volume $D^k \times [t^n, t^{n+1}]$ [23]:

$$\int_{t^n}^{t^{n+1}} \int_{D^k} R_{h,k} v_{h,k}(\mathbf{x}, t) d\mathbf{x} = 0 \tag{2.8}$$

rewritten as

$$\begin{aligned} &\int_{t^n}^{t^{n+1}} \int_{D^k} \left(\frac{\partial}{\partial t} u_{h,k}(\mathbf{x}, t) \right) v_{h,k}(\mathbf{x}, t) d\mathbf{x} \\ + &\int_{t^n}^{t^{n+1}} \int_{D^k} \left(\nabla \cdot \mathbf{f}(u_{h,k}(\mathbf{x}, t), \mathbf{x}, t) \right) v_{h,k}(\mathbf{x}, t) d\mathbf{x} \\ - &\int_{t^n}^{t^{n+1}} \int_{D^k} s(u_{h,k}(\mathbf{x}, t), \mathbf{x}, t) v_{h,k}(\mathbf{x}, t) d\mathbf{x} = 0 \end{aligned} \tag{2.9}$$

Integration by parts is now applied to the second term in order to move the derivative from the non-linear flux to the test function [15], leading to the integral weak formulation:

$$\begin{aligned} &\int_{t^n}^{t^{n+1}} \int_{D^k} \left(\frac{\partial}{\partial t} u_{h,k}(\mathbf{x}, t) \right) v_{h,k}(\mathbf{x}, t) d\mathbf{x} - \int_{t^n}^{t^{n+1}} \int_{D^k} \mathbf{f}(u_{h,k}(\mathbf{x}, t), \mathbf{x}, t) \nabla v_{h,k}(\mathbf{x}, t) d\mathbf{x} \\ &+ \int_{t^n}^{t^{n+1}} \int_{\partial D^k} \mathbf{f}^*_{h,k} \cdot \mathbf{n} v_{h,k}(\mathbf{x}, t) d\mathbf{s} - \int_{t^n}^{t^{n+1}} \int_{D^k} s(u_{h,k}(\mathbf{x}, t), \mathbf{x}, t) v_{h,k}(\mathbf{x}, t) d\mathbf{x} = 0 \end{aligned} \tag{2.10}$$

with $\mathbf{n}$ being the boundary outward-pointing unit normal vector [1]. The definition of the non-linear flux evaluated at interface is problematic because, due to the local definition of the solution and lack of conditions on local solution and test functions [5], its definition at the boundary is non-unique with neighbouring cells; i.e the solution at each interior edge is double valued [3]. By defining the flux at the interface as a function of neighbouring cells solutions, the uniqueness problem is avoided and a mechanism for exchanging informations between cells is introduced [14]. The boundary flux is therefore substituted with numerical fluxes in the same way as in finite volume.

By expanding the test functions (2.7) and simplifying the equation, we get the ADER-DG weak formulation for the evolution of the $j-$th coefficient $\forall\, j = 1, ..., N_p$ of our solution polynomial [23], [28]:

$$\begin{aligned}\int_{t^n}^{t^{n+1}} \int_{D^k} \left(\frac{\partial}{\partial t} u_{h,k}(\mathbf{x},t)\right) \varphi_j^k(\mathbf{x})\, d\mathbf{x}dt - \int_{t^n}^{t^{n+1}} \int_{D^k} \mathbf{f}(u_{h,k}(\mathbf{x},t),\mathbf{x},t)\nabla\varphi_j^k(\mathbf{x})\, d\mathbf{x}dt \\ + \int_{t^n}^{t^{n+1}} \int_{\partial D^k} \mathbf{f}^*_{h,k}\cdot \mathbf{n}\varphi_j^k(\mathbf{x})\, d\mathbf{s}dt - \int_{t^n}^{t^{n+1}} \int_{D^k} s(u_{h,k}(\mathbf{x},t),\mathbf{x},t)\varphi_j^k(\mathbf{x})\, d\mathbf{x}dt = 0\end{aligned} \tag{2.11}$$

In ADER we make the assumption that our solution polynomial $u_{h,k}(\mathbf{x},t)$ is known only at discrete points in time, i.e time steps $t^n, t^{n+1}, ...$, while in between time steps the solution is described by a spatio-temporal predictor $h_{h,k}(\mathbf{x},t)$ [23]. The predictor coefficients are not time dependent, because the time dependence and evolution is modelled by additional set of modified time-dependent basis functions and respective coefficients. The name ″predictor″ and ″corrector″ comes from the fact that $h_{h,k}(\mathbf{x},t)$ models only the cell local evolution in space and time, and thus requires to be evolved further (i.e corrected) in order to account for neighbouring cell fluxes. The solution polynomial that takes into account the exchanges between cells is the corrector $u_{h,k}(\mathbf{x},t)$. The method to compute the predictor modes will be explained in the next section.

Continuing now with the corrector equation, the first integral is exactly integrated in time and in the remaining temporal integrals we include the predictor [23],[28]:

$$\begin{aligned}\int_{D^k} (u_{h,k}(\mathbf{x},t^{n+1}) - u_{h,k}(\mathbf{x},t^n))\varphi_j^k(\mathbf{x})\, d\mathbf{x} \\ - \int_{t^n}^{t^{n+1}} \int_{D^k} \mathbf{f}(h_{h,k}(\mathbf{x},t),\mathbf{x},t)\nabla\varphi_j^k(\mathbf{x})\, d\mathbf{x}dt \\ + \int_{t^n}^{t^{n+1}} \int_{\partial D^k} \mathbf{f}^*_{h,k}(h^+_{h,k},h^-_{h,k})\cdot\mathbf{n}\varphi_j^k(\mathbf{x})\, d\mathbf{s}dt \\ - \int_{t^n}^{t^{n+1}} \int_{D^k} s(h_{h,k}(\mathbf{x},t),\mathbf{x},t)\varphi_j^k(\mathbf{x})\, d\mathbf{x}dt = 0\end{aligned} \tag{2.12}$$

By employing the expansion of the solution (2.5) we obtain our modal evolution equation, $\forall j = 1, ..., N_p$ [23],[28]:

$$\begin{aligned}
&\sum_{i=1}^{N_p} (\hat{u}_k^i(t^{n+1}) - \hat{u}_k^i(t^n)) \int_{D^k} \varphi_i^k(\mathbf{x}) \varphi_j^k(\mathbf{x})\, d\mathbf{x} \\
&- \int_{t^n}^{t^{n+1}} \int_{D^k} \mathbf{f}(h_{h,k}(\mathbf{x},t),\mathbf{x},t) \nabla \varphi_j^k(\mathbf{x})\, d\mathbf{x}dt \\
&+ \int_{t^n}^{t^{n+1}} \int_{\partial D^k} \mathbf{f}_{h,k}^*(h_{h,k}^+, h_{h,k}^-) \cdot \mathbf{n} \varphi_j^k(\mathbf{x})\, d\mathbf{s}dt \\
&- \int_{t^n}^{t^{n+1}} \int_{D^k} s(h_{h,k}(\mathbf{x},t),\mathbf{x},t) \varphi_j^k(\mathbf{x})\, d\mathbf{x}dt = 0
\end{aligned} \tag{2.13}$$

The integrals are computed using a $D+1$ dimensional Gaussian quadrature with $p+1$ quadrature points per dimension by first appropriately changing coordinates to the cell reference element: $\xi \times \tau \to [-1,1]^D \times [-1,1]$ [7], [23], [28]:

$$\xi = (\xi_1, ..., \xi_D) = \frac{2}{\Delta \mathbf{x}_k}(\mathbf{x} - \mathbf{x}_k) \;\; ; \;\; \tau = \frac{2}{\Delta t}(t - t^n) - 1 \tag{2.14}$$

with time step size Δt, cell center $\mathbf{x}_k$ and $\Delta \mathbf{x}_k$ is the vector of mesh widths in all dimensions. The equation is rearranged and the coordinate transformation applied:

$$\begin{aligned}
&\sum_{i=1}^{N_p} \hat{u}_k^i(1) det(\mathbf{J}) \int_{[-1,1]^D} \varphi_i(\xi)\varphi_j(\xi)\, d\xi = \\
&\sum_{i=1}^{N_p} \hat{u}_k^i(-1) det(\mathbf{J}) \int_{[-1,1]^D} \varphi_i(\xi)\varphi_j(\xi)\, d\xi \\
&+ \frac{\Delta t}{2} \int_{[-1,1]} \Big[det(\mathbf{J}) \int_{[-1,1]^D} \mathbf{f}(h_{h,k}(\xi,\tau),\xi,\tau) \mathbf{J}^{-1} \nabla \varphi_j(\xi)\, d\xi \\
&\quad - det(\mathbf{J}_\partial) \int_{\partial[-1,1]^D} \mathbf{f}_{h,k}^*(h_{h,k}^+, h_{h,k}^-) \cdot \mathbf{n} \varphi_j(\xi)\, d\xi_\partial \Big] d\tau \\
&+ \frac{\Delta t}{2} \int_{[-1,1]} \Big[det(\mathbf{J}) \int_{[-1,1]^D} s(h_{h,k}(\xi,\tau),\xi,\tau) \varphi_j(\xi)\, d\xi \Big] d\tau
\end{aligned} \tag{2.15}$$

with $\mathbf{J}$, $\mathbf{J}_\partial$ being the (fully diagonal) Jacobian of the spatial volume and surface coordinate transformation respectively. The basis function superscript notation k has been dropped because of the coordinate transformation. In the considered framework, we have structured rectangular/tetrahedral cells, thus the boundary integral can be just written as the sum over dimensions of the differences between surface integrals, defined w.r.t the cell boundary plus and minus in the considered dimension, i.e b_d^+, b_d^-. The notation plus and minus refers to reference cell boundaries, i.e the boundary plus is the place where $\xi_d = 1$, similarly for the boundary minus. In the next step the

Jacobian determinant are simplified, the boundary integrals and divergence term rewritten in a more compact and algorithm friendly form, Quadrature is applied to the $D+1$ dimensional integrals to obtain [23], [28]:

$$\begin{aligned}\sum_{i=1}^{N_p} \hat{u}_k^i(1) \overbrace{\int_{[-1,1]^D} \varphi_i(\xi)\varphi_j(\xi)\, d\xi}^{(\mathbf{M})_{ji}} &= \sum_{i=1}^{N_p} \hat{u}_k^i(-1) \overbrace{\int_{[-1,1]^D} \varphi_i^k(\xi)\varphi_j^k(\xi)\, d\xi}^{(\mathbf{M})_{ji}} \\ &+ \sum_{d=1}^{D} \frac{\Delta t}{\Delta x_{d,k}} \sum_{q=1}^{(p+1)^{D+1}} \widetilde{f}_{d,k}^q \underbrace{\frac{\partial \varphi_j(\xi)}{\partial \xi_d}(\xi_q) w_{q,1}...w_{q,D+1}}_{(\mathbf{S}_d)_{jq}} \\ -\sum_{d=1}^{D} \frac{\Delta t}{\Delta x_{d,k}} &\left[\sum_{q=1}^{(p+1)^D} \widetilde{f}_{d,b^+}^{*,q} \underbrace{\varphi_j(\xi_{q,b^+}) w_{q,1}...w_{q,D}}_{(\partial \mathbf{M}_d^{b^+})_{jq}} - \sum_{q=1}^{(p+1)^D} \widetilde{f}_{d,b^-}^{*,q} \underbrace{\varphi_j(\xi_{q,b^-}) w_{q,1}...w_{q,D}}_{(\partial \mathbf{M}_d^{b^-})_{jq}} \right] \\ &+ \frac{\Delta t}{2} \left[\sum_{q=1}^{(p+1)^{D+1}} \widetilde{s}_k^q \underbrace{\varphi_j(\xi_q) w_{q,1}...w_{q,D+1}}_{(\mathbf{M}_s)_{jq}} \right]\end{aligned} \tag{2.16}$$

the superscript q and tilde notation of the fluxes $\widetilde{f}_{d,k}^q, \widetilde{f}_{d,b^\pm}^{*,q}$ indicates that we are evaluating them at the $q-$th quadrature point. Using matrix-vector notation we get the final reference element ADER-DG corrector time-step update equation:

$$\begin{aligned}\hat{\mathbf{u}}(1) = \mathbf{M}^{-1} &\Bigg[\mathbf{M}\hat{\mathbf{u}}(-1) + \sum_{d=1}^{D} \frac{\Delta t}{\Delta x_{d,k}} \mathbf{S}_d \tilde{\mathbf{f}}_{d,k} \\ -\sum_{d=1}^{D} \frac{\Delta t}{\Delta x_{d,k}} &\left[\partial \mathbf{M}_d^{b^+} \tilde{\mathbf{f}}_{d,b^+}^* - \partial \mathbf{M}_d^{b^-} \tilde{\mathbf{f}}_{d,b^-}^* \right] + \frac{\Delta t}{2} \mathbf{M}_s \tilde{\mathbf{s}}_k \Bigg]\end{aligned} \tag{2.17}$$

For smooth enough solutions, it can be shown that the obtained DG solution has $p+1$ order of convergence [5]:

$$\|u(\mathbf{x},t) - u_h(\mathbf{x},t)\| \leq Ch^{p+1} \tag{2.18}$$

with grid spacing h. Any explicit time-stepping scheme requires the time step size to be bounded by the CFL condition, this is done to ensure that waves originated at neighbouring cell interfaces do not intersect before reaching the next time step. This induces the following timestep constraint for a DG method of order $p+1$ [7], [23], [36]:

$$\Delta t \leq \frac{1}{2p+1} \frac{1}{D} \frac{\Delta \overline{x}}{|\lambda_{max}|} \tag{2.19}$$

with $\Delta\overline{x}$: the average cell size, λ_{max} : maximum signal velocity (equivalent to the maximum eigenvalue of the Jacobian of the non-linear flux).

2.2.1 Basis function

In the considered framework we employ a set of orthogonal polynomial basis functions of maximum order p, defined as the $D-$dimensional tensor product of Legendre polynomials [4], [6], [7]. Legendre polynomials are chosen because of their orthogonality property, which leads to sparse and easy to compute system matrices. The use of a Legendre basis motivates the naming of the polynomial expansion coefficient as "modes". The basis function are therefore defined as follows [7]:

$$\{\varphi_i(\xi)\}_{i=1}^{N_p} = \left\{ \prod_{d=1}^{D} P_{i_d}(\xi_d) \mid \forall\, i_1, ..., i_D \in \mathbb{N}_0 \wedge \sum_{d=1}^{D} i_d \leq p \right\} \tag{2.20}$$

with reference coordinates vector $\xi = (\xi_1, \xi_2, \xi_3)$ and Legendre polynomial $P_n(\xi$: of order n and p the maximum degree of the basis function φ [7]. The sums of their orders is maximum equal to the order p of our basis function. The mapping between the one-dimensional indices $i_1, ..., i_D$ and the index i is made such that each combination of the D indices leads to a unique index i. This definition of basis function induces the existence of N_p distinct basis functions. N_p defined as [1],[2],[7],[13]:

$$N_p(D) = \frac{(p+D)!}{D!p!} \tag{2.21}$$

In our framework we are dealing with $D \leq 3$; therefore we get

$$N_p(D) = \begin{cases} (p+1) \;\; ; \; D=1 \\ \frac{1}{2}(p+1)(p+2) \;\; ; \; D=2 \\ \frac{1}{6}(p+1)(p+2)(p+3) \;\; ; \; D=3 \end{cases} \tag{2.22}$$

2.3 Local space-time Galerkin predictor

Starting from the original PDE expressed for the predictor in cell D^k, i.e

$$\frac{\partial}{\partial t} h_{h,k}(\mathbf{x},t) + \nabla \cdot \mathbf{f}(h_{h,k}(\mathbf{x},t),\mathbf{x},t) - s(h_{h,k}(\mathbf{x},t),\mathbf{x},t) = R_{h,k} \tag{2.23}$$

a similar ansatz to the corrector one is made, but here we remove time dependence from the modes [23], [37], [36]:

$$h_{h,k}(\mathbf{x},t) = \sum_{i=1}^{\widetilde{N}_p} \hat{h}_k^i \widetilde{\varphi}_{i,i_t}^k(\mathbf{x},t) \equiv \sum_{i=1}^{N_p} \hat{h}_k^i \widetilde{\varphi}_{i,0}^k(\mathbf{x},t) + \sum_{i=N_p+1}^{\widetilde{N}_p} \hat{h}_k^i \widetilde{\varphi}_{i,i_t}^k(\mathbf{x},t) \tag{2.24}$$

The sum can be viewed as two distinct sums. The first term carries information regarding the spatial evolution of the solution and it is equivalent to the corrector evaluated at time t, while the second term describe the combined spatio-temporal evolution. The modified basis function is defined as [26]:

$$\widetilde{\varphi}^k_{i,i_t}(\mathbf{x}, t) = \varphi^k_i(\mathbf{x})\varphi^k_{i_t}(t) \tag{2.25}$$

with temporal basis function $\varphi^k_{i_t}(t) = \mathcal{O}(t^{i_t})$. The modified basis functions (w.r.t the corrector basis function, noted with tilde) have maximum order p, therefore the indices are constrained s.t:

$$\left(\sum_{d=1}^{D} i_d\right) + i_t \leq p \tag{2.26}$$

which induces the modified number of modes to be

$$\widetilde{N}_p(D) = \sum_{i_t=0}^{p} \sum_{i_1=0}^{p-i_t} \dots \sum_{i_D=0}^{p-i_t-i_1-\dots-i_{D-1}} 1 = N_p(D+1) \tag{2.27}$$

The PDE is now imposed to be orthogonal w.r.t the full spatio-temporal element by employing a test function $v_{h,k}$ expressed using the modified basis function $\widetilde{\varphi}^k_{j,j_t}$. Following the same procedure as for the corrector, with only difference that we allow the solution to be continuous also at the boundaries (i.e we don't apply the divergence theorem), we get $\forall\ j = 1, ..., \widetilde{N}_p$ [23], [37], [36]:

$$\begin{aligned}\int_{t^n}^{t^{n+1}} \int_{D^k} \widetilde{\varphi}^k_{j,j_t} \frac{\partial}{\partial t} h_{h,k}(\mathbf{x}, t)\ d\mathbf{x}dt + \int_{t^n}^{t^{n+1}} \int_{D^k} \widetilde{\varphi}^k_{j,j_t} \nabla \cdot \mathbf{f}(h_{h,k}(\mathbf{x}, t), \mathbf{x}, t)\ d\mathbf{x}dt \\ - \int_{t^n}^{t^{n+1}} \int_{D^k} \widetilde{\varphi}^k_{j,j_t} s(h_{h,k}(\mathbf{x}, t), \mathbf{x}, t)\ d\mathbf{x}dt = 0\end{aligned} \tag{2.28}$$

The same coordinate change done for the corrector is applied here, leading to

$$\begin{aligned}&\frac{\Delta t}{2} \int_{-1}^{1} \int_{[-1,1]^D} \widetilde{\varphi}_{j,j_\tau} \frac{2}{\Delta t} \frac{\partial}{\partial \tau} h_{h,k}(\xi, \tau)\ d\xi d\tau \\ +&\frac{\Delta t}{2} \int_{-1}^{1} \int_{[-1,1]^D} \widetilde{\varphi}_{j,j_\tau} \mathbf{J}^{-1} \nabla \cdot \mathbf{f}(h_{h,k}(\xi, \tau), \xi, \tau)\ d\xi d\tau \\ -&\frac{\Delta t}{2} \int_{-1}^{1} \int_{[-1,1]^D} \widetilde{\varphi}_{j,j_\tau} s(h_{h,k}(\xi, \tau), \xi, \tau)\ d\xi d\tau = 0\end{aligned} \tag{2.29}$$

in order to obtain the predictor evaluated at distinct point, the first term is now integrated by parts in time [23], [37], [36]

$$\begin{aligned}\frac{2}{\Delta t}\int_{[-1,1]^D}\widetilde{\varphi}_{j,j_\tau}(\xi,1)h_{h,k}(\xi,1)\,d\xi-\frac{2}{\Delta t}\int_{[-1,1]^D}\widetilde{\varphi}_{j,j_\tau}(\xi,-1)h_{h,k}(\xi,-1)\,d\xi\\ -\frac{2}{\Delta t}\int_{-1}^{1}\int_{[-1,1]^D}h_{h,k}(\xi,\tau)\frac{\partial}{\partial\tau}\widetilde{\varphi}_{j,j_\tau}(\xi,\tau)\,d\xi d\tau\\ +\int_{-1}^{1}\int_{[-1,1]^D}\widetilde{\varphi}_{j,j_\tau}\mathbf{J}^{-1}\nabla\cdot\mathbf{f}(h_{h,k}(\xi,\tau),\xi,\tau)\,d\xi d\tau-\\ \int_{-1}^{1}\int_{[-1,1]^D}\widetilde{\varphi}_{j,j_\tau}s(h_{h,k}(\xi,\tau),\xi,\tau)\,d\xi d\tau=0\end{aligned} \tag{2.30}$$

To compute the integrals of the non-linear flux and the source term, we utilize the so called ”quadrature-free” approach proposed by Aktins and Shu [8] and utilized e.g in [1], [5], [9], [13], [14] in the context of RKDG to compute the spatial integrals, and in [23],[24],[37] ,[36] for ADER-DG, where the source term and non-linear flux are expanded in the same way as the predictor polynomial in order to obtain a comparable space-time accuracy [26]. As noted in [24], advantages of modal ADER-DG formulations, such as the ability to include viscosities and resistivities, requires an efficient method for going from modal to nodal formulations and vice versa.

The ”quadrature free” approach is used in the DG literature due to its computational efficiency (especially in multi-dimensional problems) compared to classic Gaussian quadrature, because the fluxes have to be evaluated at less points. In addition in the quadrature free method, the choice of interpolation points is more flexible [8]. For these reasons the quadrature free method is more suited to handle unstructured meshes made of multi-dimensional complex elements, for which optimal quadrature formulas of the desired accuracy might not be available and thus it allows to implement DG using polynomial basis of any degree [8].

As downsides, the quadrature free method does not guarantee the exact integration, in fact it has a slightly lower achieved accuracy [13], and many theoretical results that have already been derived for quadrature based DG might not be readily transferable to a quadrature-free framework [15]. The increased freedom regarding the choice of interpolation points can become a disadvantage if they are not properly chosen. These downsides, when the method is used for RKDG schemes, leads to aliasing being more pronounced [15], [5]. We won’t go into details, but in general, aliasing is a variational crime caused by inaccurate integration of the non-linear volume and surface integrals. It leads to stability problems and require special care [5], [16].

Given the $d-$component of the non-linear flux (similarly also the source), we impose equivalence of its the modal expansion with a nodal basis expansion [1], [13]

$$f_d(h_{h,k}(\xi,\tau),\xi,\tau) = \sum_{i=1}^{\widetilde{N}_p} \hat{f}^i_{d,k}\widetilde{\varphi}_{i,i_t}(\xi,\tau) = \sum_{m=1}^{\widetilde{M}_p} \tilde{f}^m_{d,k}\widetilde{\psi}_{m,m_t}(\xi,\tau) \tag{2.31}$$

The transition between nodal and modal approximations can be made through the generalized Vandermonde matrix $\mathbf{V}$ [13] evaluated at the interpolation points:

$$\mathbf{V}^T\widetilde{\psi} = \widetilde{\varphi} \leftrightarrow \widetilde{\psi} = \mathbf{V}^{-T}\widetilde{\varphi} \;\; ; \;\; V_{ij} = \widetilde{\varphi}_j(\xi_i) \tag{2.32}$$

then the nodal to modal conversion is made for all $i = 1, ..., \widetilde{M}_p$, $j = 1, ..., \widetilde{N}_p$ as follows:

$$\hat{\mathbf{f}}_{d,k} = \mathbf{V}^{-1}\tilde{\mathbf{f}}_{d,k} \rightarrow \hat{f}^n_{d,k} = \sum_{m=1}^{\widetilde{M}_p}(V^{-1})_{nm}\tilde{f}^m_{d,k} \;\; ; \;\; (V^{-1})_{nm} = (V^{-T})_{mn} \tag{2.33}$$

The Vandermonde matrix is not square, therefore its pseudoinverse is computed using SVD like the authors in [1], [13]. The consequence is that the modal and nodal approximation will give non equal results [13]. In our framework we employ Gauss-Lobatto quadrature points because, due to the way we store point-wise evaluations of fluxes, it allows us to reuse the same data structures employed in the quadrature for the constructor step (the data structures are defined w.r.t the number of points; a different distribution would not be an issue). For more optimal points distributions the reader is redirected the the scientific literature, to start [1], [13]. In addition, as discussed in the original paper introducing the quadrature free approach, the number of interpolation point also has an impact on the DG scheme accuracy. In their case, the quadrature free approach was used for a RKDG scheme and it was noted that for non-linear fluxes, $\widetilde{M}_p \geq \widetilde{N}_{p+1} > \widetilde{N}_p$ quadrature points were at least needed to achieve a good polynomial representation and thus the desired $p+1$ convergence order [8].

In our case we made the assumption that their result transfer to ADER-DG and therefore when using $(p+1)^{D+1}$ Gauss Lobatto interpolation points for the modal to nodal conversion the condition is satisfied. To note that the method shown above for switching between nodal and modal basis is usually applied and reported in the literature for orthonormal modal basis functions. In our framework we are using orthogonal basis functions, but the method can still be used.
We continue now the derivation of the corrector update equation. The ADER method require an important simplification to be made, i.e that the space-time

predictor initial state is identical to the corrector [23], [24], [36], [37]:

$$h_{h,k}(\xi,-1) = u_{h,k}(\xi,-1)$$

By substituting all of the expansions in the corrector equation, dividing the modified basis functions into its temporal and spatial components and rearranging the terms we get the equation for the predictor modes $\forall\ j = 1,...,\widetilde{N}_p$ [23], [24], [36], [37]:

$$\sum_{i=1}^{\widetilde{N}_p} \hat{h}_k^i \overbrace{\Bigg(\varphi_{j_\tau}(1)\varphi_{i_\tau}(1) \int_{[-1,1]^D} \varphi_j(\xi)\varphi_i(\xi)\, d\xi - \left[\int_{-1}^{1} \varphi_{i_\tau}(\tau)\frac{\partial \varphi_{j_\tau}(\tau)}{\partial \tau} d\tau\right] \left[\int_{[-1,1]^D} \varphi_j(\xi)\varphi_i(\xi)\, d\xi\right]}^{(\mathbf{M}_h)_{ji}} \tag{2.34}$$

$$= \sum_{i=1}^{N_p} \hat{u}_k^i(-1)\, \underbrace{\varphi_{j_\tau}(-1) \int_{[-1,1]^D} \varphi_j(\xi)\varphi_i(\xi)\, d\xi}_{(\mathbf{M}_u)_{ji}} -$$

$$\sum_{d=1}^{D} \frac{\Delta t}{\Delta x_{d,k}} \sum_{m=1}^{\widetilde{M}_p} \tilde{f}_d^m \sum_{i=1}^{\widetilde{N}_p} (V^{-T})_{im} \underbrace{\left[\int_{-1}^{1} \varphi_{i_\tau}(\tau)\varphi_{j_\tau}(\tau) d\tau\right] \left[\int_{[-1,1]^D} \varphi_j(\xi)\frac{\partial}{\partial \xi_d}\varphi_i(\xi)\, d\xi\right]}_{(\mathbf{S}_{d,f})_{ji}}$$

$$+\frac{\Delta t}{2} \sum_{m=1}^{\widetilde{M}_p} \tilde{s}_d^m \sum_{i=1}^{\widetilde{N}_p} (V^{-T})_{im} \underbrace{\left[\int_{-1}^{1} \varphi_{i_\tau}(\tau)\varphi_{j_\tau}(\tau) d\tau\right] \left[\int_{[-1,1]^D} \varphi_j(\xi)\varphi_i(\xi)\, d\xi\right]}_{(\mathbf{M}_s)_{ji}} \tag{2.35}$$

The final equation is rewritten in system form:

$$\hat{\mathbf{h}} = \mathbf{M}_h^{-1} \left[\mathbf{M}_u \hat{\mathbf{u}}(-1) - \sum_{d=1}^{D} \frac{\Delta t}{\Delta x_{d,k}} \mathbf{S}_{d,f} \mathbf{V}^{-1} \tilde{\mathbf{f}}_d + \frac{\Delta t}{2} \mathbf{M}_s \mathbf{V}^{-1} \tilde{\mathbf{s}} \right] \tag{2.36}$$

In the surveyed literature, authors always use a monomial temporal basis function. In our case, in order to re-use the code that pre-computes the spatial integrals we will utilize a Legendre temporal basis function $\varphi_{i_\tau}(\tau) = P_{i_\tau}(\tau)$. The corrector modes are iteratively found. At the beginning of the iteration process, the predictor modes are set equal to the current step corrector modes. Afterwards, fixed point iteration is performed until convergence is reached [26]. The predictor modes at the current iteration are used to evaluate the fluxes and sources. In the literature it is reported that a sufficient number of iterations to reach the desired DG order, is equal to the modified basis function order, i.e p [24]; this has been also our observation. Once the iterations are finished, the corrector step begins.

2.4 Evaluation of system matrices

The system matrices for both predictor and corrector are computed only once at the beginning of the simulation. The corrector matrices entries just require the evaluation of the basis functions at the correct quadrature points and the respective quadrature weights to be computed. When using quadrature, we always employ $p + 1$ nodes per dimension, which allow the exact integration of polynomials up to degree $2p + 1$ on $[-1, 1]$, and thus , as noted in [6], the expected DG order of accuracy can be achieved. The approach to find the quadrature points is the same reported in [7], [51]. They are found by employing their symmetry and using a slightly modified version of the asymptotic approximations reported in [51] as initial guess, which is then iteratively improved with Newton Raphson. The exact formulas can be found in Appendix A. These formulas are best used for very high degree polynomials, and therefore in this framework simpler formulas could have been used.

The predictor matrices can be computed either analytically or using quadrature. For the mass matrix the analytical form is straight forward to compute and makes use of the orthogonality property. For the stiffness matrix it is a bit more involved and a derivation can be found in Appendix A. For consistency reasons, GL quadrature is used also for the predictor.

2.5 Numerical Flux

The numerical flux has the requirement to be monotone, consistent and conservative [6]. Monotonicity ensures that in presence of piecewise constant solutions, the DG scheme is stable and converges to the physically relevant solution (entropy solution) [6], [29]. Consistency means that the numerical flux, should correctly represent and approximate the non-linear flux at the interface [6], [29]. Authors of [6] suggest that increasing the DG order decreases the impact that the selected numerical flux has on the approximation. Therefore the chosen numerical flux should be tailored for the desired application for optimal results. In the context of the creation of the DG-AMR framework we put more importance on its ability to generalize to different applications and the implementation simplicity. For these reasons the local Lax-Friedrich (Rusanov) flux, which satisfies all requirements, has been used. The LF flux is commonly used in the scientific literature [29] as it can be applied to any non-linear hyperbolic system [6]. Its main downside is that, compared with other monotone fluxes, it introduces the most numerical viscosity [6], [29]. In their comparative paper, authors of [29] noted that the Lax-Friedrich flux is the least expensive from a computational time point of view but leads to the highest numerical errors when compared with other numerical fluxes.

The numerical fluxes at interface b are computed for all interfaces between cells by considering the interface b minus and plus states. In the constructed framework, numerical fluxes are stored in a face centred way. The numerical flux evaluated at the $m-$th interpolation/quadrature point on the selected boundary b in dimension d w.r.t cell k is a function of the solution of the two cells evaluated at the interface and their respective non-linear fluxes (without time notation) [25]:

$$\begin{aligned} f^{*,m}_{d,b} &= f^{*}_{d}(u_{h,k_d}(\xi_{m,b^+}), u_{h,k_d+1}(\xi_{m,b^-}), f_{h,d}(u_{h,k_d}(\xi_{m,b^+})), f_{h,d}(u_{h,k_d+1}(\xi_{m,b^-}))) \\ &= \frac{f_{h,d}(u_{h,k_d}(\xi_{m,b^+})) + f_{h,d}(u_{h,k_d+1}(\xi_{m,b^-})}{2} - \frac{C}{2}(u_{h,k_d+1}(\xi_{m,b^-}) - u_{h,k_d}(\xi_{m,b^+})) \end{aligned} \tag{2.37}$$

with constant $C = max(|\frac{\partial}{\partial u} f(u_{h,k_d}(\xi_{m,b^+}))|, |\frac{\partial}{\partial u} f(u_{h,k_d+1}(\xi_{m,b^-}))|)$ denoting the largest maximum absolute eigenvalue of the Jacobian matrix of the non-linear flux evaluated at plus and minus sides [25],[6].

2.6 Limiting

The limiting procedure aims to compute and apply bounds to the polynomial modes in order to dampen oscillations which appear near discontinuities for high-order DG methods [21], [34]. In general, limiters should be able to detect oscillations near discontinuities and generate a corrected solution which preserve conservation inside the cell [6]. If applied in smooth regions, limiters should also not disrupt the existing solution [6]. In addition, it desirable to preserve positivity of the conserved variables, e.g to avoid having negative densities and pressures [34], [26]. The Total Variation Bounded (TVB) limiter [6] operates only on the linear modes (i.e slopes) of the solution polynomial, discarting all the higher order ones [26]. The TVB limiter has all the previously mentioned properties [27] apart from it ability to fully preserve positivity [34].

In our framework we employ the TVB characteristic limiting technique formulation used in [45]. A similar formulation of the TVB limiter, based on the work of other authors [38], [34], is also used as troubled cell indicators for AMR refinement and is presented in the dedicated AMR chapter. The TVB limiter is widely used in the literature and is a modification of the even more famous TVD limiter, with the difference that the TVB limiter is not activated near smooth extrema [45]. This limiter locally trades the high-order approximation of the solution for an enhanced simulation stability. This limiter was chosen because its implementation is straightforward.

For the troubled cell D^k, limiting is applied independently in each dimension The presented formulation is for a general system of hyperbolic PDEs with cell local approximated solution vector $\mathbf{U}_{h,k_d}$. For scalar equations the characteristic projection is not required [34]. We begin by projecting the solution vector into its characteristic field, i.e the field of decoupled advected quantities [7]. As noted in [6], this approach leads to superior result compared to applying the limiter componentwise to the conserved variables. The minmod function is applied componentwise to the characteristic solution vector elements, and the result is transformed back into conserved field. We thus have:

$$\mathbf{V}_{h,k_d} = \mathbf{R}^{-1}\mathbf{U}_{h,k_d} = \mathbf{R}^{-1}\sum_{i=1}^{N_p}\hat{\mathbf{U}}^i_{k_d}\varphi^k_i(\mathbf{x}) = \sum_{i=1}^{N_p}\underbrace{\mathbf{R}^{-1}\hat{\mathbf{U}}^i_{k_d}}_{\hat{\mathbf{V}}^i_{k_d}}\varphi^k_i(\mathbf{x}) \tag{2.38}$$

with $\hat{\mathbf{V}}^i_{k_d}$ being the characteristic modes vector. The right eigenvectors matrix $\mathbf{R}$ is found by diagonalizing the Jacobian $\frac{\partial \mathbf{F}_d}{\partial \mathbf{U}}|_{\hat{\mathbf{U}}^1_k}$ evaluated at the cell average state $\hat{\mathbf{U}}^1_k$ [21], [27]:

$$\mathbf{R}(\hat{\mathbf{U}}^1_k) = (r^1_k, ..., r^Q_k) \;\; ; \;\; \mathbf{L}(\hat{\mathbf{U}}^1_k) = \mathbf{R}^{-1}(\hat{\mathbf{U}}^1_k) = (l^1_k, ..., l^Q_k)^T \tag{2.39}$$

where r^n_k is the n-th column made of the $n-$th right eigenvector, and l^n_k is the n-th row made of the respective left eigenvector. The following slopes are defined as [27], [26]:

$$\Delta_+\overline{\mathbf{V}}_{h,k_d} = \frac{1}{2}\mathbf{R}^{-1}(\hat{\mathbf{U}}^1_{h,k_d+1} - \hat{\mathbf{U}}^1_{h,k_d}) \;\; ; \;\; \Delta_-\overline{\mathbf{V}}_{h,k_d} = \frac{1}{2}\mathbf{R}^{-1}(\hat{\mathbf{U}}^1_{h,k_d} - \hat{\mathbf{U}}^1_{h,k_d-1}) \tag{2.40}$$

The coefficient $1/2$ comes from the fact that the slopes are computed w.r.t cell centres of neighbouring reference elements [28]. These slopes are then passed to the bounded minmod function together with the actual characteristic solution linear modes (noted with index s) $\hat{\mathbf{V}}^s_{h,k_d} = \mathbf{R}^{-1}(\hat{\mathbf{U}}^s_{h,k_d})$ and their sign is compared:

$$\hat{\mathbf{V}}^{s,mod}_{h,k_d} = minmodB(\hat{\mathbf{V}}^s_{h,k_d}, \Delta_+\overline{\mathbf{V}}_{h,k_d}, \Delta_-\overline{\mathbf{V}}_{h,k_d}) \tag{2.41}$$

with

$$minmodB(a_1, a_2, a_3) = \begin{cases} a_1 \;\; ; \; |a_1| \leq Mh^2 \\ minmod(a_1, a_2, a_3) \;\; ; \; else \end{cases} \tag{2.42}$$

$$minmod(a_1, a_2, a_3) = \begin{cases} s \cdot \min_{1\leq j\leq 3}|a_j| \;\; ; \; s = sign(a_1) = sign(a_2) = sign(a_3) \\ 0 \;\; ; \; else \end{cases} \tag{2.43}$$

The boundedness aspect of the limiter is applied when comparing a_1 to a chosen coefficient M and the mesh width. Low values of M leads to more discontinuities being detected, even if not dangerous for the simulation stability. By tuning this parameter we can control the oscillations. In our framework it is chosen depending on the case analysed, but an analytical formula exist to predict its value based on the second derivative of the initial conditions [41]. If oscillations are detected, then the minmod function compares the slopes between neighbouring cells. If of opposite signs, a complete removal of the linear mode is done, otherwise the linear mode is set to the minimum of the neighbours.

2.7 Initial conditions

An L_2 projection is required in order to obtain initial conditions for the solution modes as proposed by [7]. We aim to find the modes of the polynomial solution so that the squared difference w.r.t the true solution on the entire cell is minimized:

$$\min_{\hat{\mathbf{u}}_k} \int_{D^k} (u(\mathbf{x},t) - u_{h,k}(\mathbf{x},t))^2 d\mathbf{x} \tag{2.44}$$

by solving the minimization problem, and setting t=0. We get the modal initial conditions for all $j = 1, ..., N_p$:

$$\hat{u}_k^j(0) = \left(\prod_{d=1}^{D} \frac{2j_d + 1}{2} \right) \int_{[-1,1]^D} u(\xi, 0) \varphi_j(\xi) d\xi \tag{2.45}$$

the above integral is computed using a $(p+1)^D$ points Gauss-Legendre quadrature rule.

2.8 Boundary conditions

Boundary conditions in the DG context are enforced in the weak sense in the same manner as in the Finite Volume Method [1]. The solution at the boundary is specified on a set of ghost cells and its influence enters the DG formulation through the numerical flux [1], [2], [5], [6]. When evaluating the numerical fluxes at the boundaries, the boundary function will influence the solution polynomial value at the left or right interface. When given a continuous formulation of a boundary condition, we are required to cast it into a discrete form and enforce it through domain ghost cells. In the considered framework, the two most general types of boundary conditions have been implemented: Dirichlet and Neumann conditions.

Dirichlet BC

$$u(\mathbf{x},t) = g_D(\mathbf{x}) \;\; ; \;\; \forall \mathbf{x} \in \partial\Omega \tag{2.46}$$

The discrete formulation follows:

$$u_{h,k}(\mathbf{x},t) \simeq g_D(\mathbf{x}) \;\; ; \;\; \forall \mathbf{x} \in \partial D^K \subset \Omega_h \tag{2.47}$$

The implementation is straight forward and requires us to set the correct value of the ghost cells.

Neumann BC

$$\nabla u(\mathbf{x},t)\mathbf{n} = g_N(\mathbf{x}) \;\; ; \;\; \forall \mathbf{x} \in \partial\Omega \tag{2.48}$$

then the discrete formulation is

$$\nabla u_{h,k}(\mathbf{x},t)\mathbf{n} \simeq g_N(\mathbf{x}) \;\; ; \;\; \forall \mathbf{x} \in \; \partial D^K \subset \Omega_h \tag{2.49}$$

First order finite difference is used to discretize the gradients and in our framework they are defined to point outwards, i.e from the domain into the boundary. For both plus and minus boundaries we get

$$\frac{u_{h,k}(\mathbf{x},t) - u_{h,k}(\mathbf{x}_i,t)}{\Delta} = g_{N,\pm}(\mathbf{x}) \tag{2.50}$$

with $\mathbf{x}_i$ being a point inside of the domain (adjacent to the ghost cell). Neumann BC are then casted as Dirichlet BC and the value assigned to the ghost cell.

2.8.1 Modal formulation

In our framework the boundary conditions are applied at the beginning of each time step on the solution polynomial modes, after all the required multi-level synchronization has been done. The alternative would be to assign the boundary conditions when computing the pointwise evaluations of our solution polynomial at the cells plus and minus interfaces; which are needed for the numerical fluxes evaluations. This has to be done twice in each dimension. Either a datastructure which only holds boundary values which we read from when needed has to be used (and initialized with the correct values at each step) and/or cells index conditions checks inside the grid loops shall be used. Overall we felt that having the required data for predictor and corrector already prepared at the beginning of the step was the better choice. Therefore we need to compute the modal representation of the boundary conditions and store them in the same specialized distributed data structure provided by AMReX, which we already use to store our modal solution. To this end, we perform an L_2 projection of the boundary function onto its modes and evaluate the integral with GL-quadrature at $(p+1)^D$ quadrature points expressed on the reference element, exactly like we do for the initial conditions.
Dirichlet conditions are generally fixed values, therefore the projection will just initialize the cell average. In the case of Neumann conditions we can recover a modal representation by imposing the desired gradient on to neighbouring cells solution polynomials, evaluated at the quadrature points of their respective reference cells:

$$u_{k_{BC}}(\xi_q,t) = u_{k_{BC}\pm 1}(\xi_q,t) + g_N \Delta x_d \tag{2.51}$$

As it will be explained later, for the Euler equations the gradient g_N is expressed w.r.t conserved quantities and their respective boundary conditions. Some quantities (e.g momentum) require a point evaluation. In our framework they are evaluated at a point which is shifted, depending on the direction/location of the boundary, towards the reference cell ± 1 interface. This general modal formulation has been implemented in order to allow for further development and inclusion of more complex boundary functions or even boundary conditions expressed directly for the modes.

Chapter 3

Adaptive Mesh Refinement

We employ a block structured fully conforming AMR technique. Block meaning that, contrary to what is done in ”in cell” approaches e.g [7], [31], [10], [25], [28] ,[37] cells are not individually refined but instead aggregations of cells, called blocks, are constructed and equally refined [32]. The ensemble of these blocks form the grid of a specific refinement level [33]. In addition, blocks defined at a specific refinement level are fully enclosed by blocks defined at the refinement level preceding it. This is achieved (if needed) through buffer cells and ensures that jumps across levels are at most of one refinement ratio, which helps with stability [7], [25]. Buffer cells are cells which were not tagged for refinement but still get intentionally refined to ensure proper grid creation. We work with a refinement ratio of 2 and a buffer thickness of 1 cell. Conforming refers to the fact that refined cells perfectly fill their parent coarse cell interfaces, leaving no hanging nodes [31]. The ADER-DG scheme can thus be applied to any grid, independently of the level of its refinement and the one of its neighbours, without any change regarding the numerical fluxes computation [25],[31].

As already explained, interaction between cells is handled by the numerical fluxes. Cells with boundaries at the fine-coarse levels interfaces require thus to have access to the current modal expansion at the other level of refinement. The correct exchange of the solution at different levels of refinement needs to be conservative and not loose too much information [7]. This is accomplished through a coarse to fine modal L_2 projection, which is stored in the fine level ghost cells. This operation is often referred to as interpolation/scattering. The ghost cells surround every grid at each level of refinement (even if they don’t span the entire domain). The inverse communication operation requires to reconstruct the coarse cell solution from its individual fine cells contributions and it is also done using a projection technique. This inverse operation is called gathering [31]. In our framework, the ghost-cell layer has a thickness of 1 cell. Ghost cells are updated at the beginning of the simulation and after every time step. The communication happens between blocks of the same grid, i.e same level; and for ghost cells at coarse-fine interfaces which will receive data scattered from the coarser cell at one coarser level of refinement.

The current AMR strategy consist of evaluating a refinement criteria in order to tag cells that require refinement. New fine grids are created and the coarse solution is projected onto the fine grid. Coarse and fine grids are then evolved independently and communicate only through numerical fluxes at the boundaries [25],[17]. These fluxes are independently computed using up-to date solution data from other levels. After the time evolution, to ensure conservation, the levels gather the data from the overlapping finer levels. Subsequently a global ghost cells synchronization is done, where levels exchange ghost cells data with themselves and interpolate valid domain cells data into finer levels ghost cells. Synchronization is needed at the same level because the grid is distributed among multiple MPI ranks and each rank evolves its own ghost cells until the final solution update. The final solution is updated only on the grid without considering ghost cells, this is done because ghost cells are present only to allow the computation of the numerical fluxes. Therefore we need to be able to perform point wise evaluations of the solution and physical flux at the ghost cells beforehand.

After this synchronization, the new time step starts and cells are tagged for refinement. If re-meshing happens, some synchronization operations will be carried out again. Independently of the re-meshing of finer levels, the coarser ones will remain up-to date with the latest fine data. If the conditions for refinement don't hold any more and de-refinement is triggered, we will thus not need to gather the data, because it has already been done (and the solution has not evolved since then). Multiple refinement criteria exists, a selection of which have been implemented, the main one being the discontinuity TVB detector. The choice and combination strategy for refinement criteria is driven by the specific physical problem tackled and requires fine tuning.

We use a global time stepping approach where all levels are evolved w.r.t the same global time coordinate and the time step is given by the minimum across all levels. This choice was driven by time constraints of the project. More computationally efficient solutions (and most often used) employ local time stepping, i.e time-step sub cycles of fine levels. w.r.t a full coarse time-step are performed. This approach generally introduces some difficulties regarding the treatment of accumulated fluxes at fine-coarse interfaces [25]. As noted in [25], ADER schemes with subcycling don't have this problem because they use the predictor polynomial, which is constructed and models the evolution for the full time step; therefore for each fine time step, the fine

coarse interface fluxes can be computed using the same coarse predictor.
As explained, once the time-step has started each MPI ranks updates its own ghost cells values until the corrector step is performed. After the end of the time step the ghost cells will need to be communicated across MPI ranks. This choice of avoiding communication during the time step has been done because it was considered better limit communication between ranks at few locations in the code until a proper computation vs communication overhead analysis is performed. But in case it is worth exchanging data vs computing it, that would involve iterating across grids with ghost cells and skip those that are not at fine-coarse interfaces (index check w.r.t grid bounds). Skipped ghost cells data would then be exchanged between ranks.

3.1 Coarse to fine solution scattering

The solutions restricted to the coarse cell at level L, D^{k_L} , and to the fine cell at level $L+1$, $D^{k_{L+1}}$, are expressed w.r.t their reference element and given by:

$$u_{h,k_L}(\xi_L,t) = \sum_{m=1}^{N_p} \hat{u}^m_{k_L}(t)\varphi_m(\xi_L) \;\; ; \;\; u_{h,k_{L+1}}(\xi_{L+1},t) = \sum_{n=1}^{N_p} \hat{u}^n_{k_{L+1}}(t)\varphi_n(\xi_{L+1}) \tag{3.1}$$

with both $\varphi_m(\xi_L), \varphi_n(\xi_{L+1})$ defined in $[-1,1]^D$, with $\xi_L, \xi_{L+1} \in [-1,1]^D$. The L_2 projection is now made (dropping time dependence) [7],[10], [31]:

$$\min_{\hat{u}^n_{k_{L+1}}} \int_{[-1,1]^D} (u_{h,k_L}(\xi_L) - u_{h,k_{L+1}}(\xi_{L+1}))^2 d\xi \tag{3.2}$$

rewritten as

$$\sum_{n=1}^{N_p} \hat{u}^n_{k_{L+1}} \int_{[-1,1]^D} \varphi_n(\xi_{L+1})\varphi_j(\xi_{L+1})d\xi = \sum_{m=1}^{N_p} \hat{u}^m_{k_L} \int_{[-1,1]^D} \varphi_m(\xi_L)\varphi_j(\xi_{L+1})d\xi \tag{3.3}$$

the left hand side simplifies thanks to orthogonality, leading to to

$$\hat{u}^n_{k_{L+1}} = \left[\prod_d^D \frac{2}{2n_d+1}\right]^{-1} \sum_{m=1}^{N_p} \hat{u}^m_{k_L} \int_{[-1,1]^D} \varphi_m(\xi_L)\varphi_n(\xi_{L+1})d\xi \tag{3.4}$$

At their respective level of refinement, the basis functions are defined w.r.t their own reference element, and therefore if a projection is made without a rescaling of the fine level basis function, we will just copy the entire coarse cell data (because the reference elements fully overlaps). We thus need to rescale the fine level so that, with respect to the coarse one, the basis functions are defined on one of the coarse sub-cells. In this way, by integrating on the full reference domain (equivalent with the coarse one), the coarse cell will just project the solution contained in the region which overlaps with the fine

one.
From the reference frame of a coarse level $\xi_L \equiv \xi \in [-1,1]^D$, the fine cell is defined in a combination of one dimensional intervals which are defined either as $[-1,0]$ ("-" sub-cell) or $[0,1]$ ("+" sub-cell). The following change of coordinates is defined for a refinement ratio of 2 and formulated for the $d-$th component of the reference coordinates [7],[31]:

$$\xi_{d,L+1} = \frac{1}{2}\xi_d \pm \frac{1}{2} \;\; ; \;\; \xi_d \in [-1,1] \tag{3.5}$$

with ξ_d being a the component of the generic reference coordinate and $\xi_{d,L+1}$ now defined either in the minus $[-1,0]$ or the plus (+) sub-cell $[0,1]$. The integral in the projection is solved using the same quadrature approach we used throughout this work and the final form reads [7],[31]:

$$\hat{u}^n_{k_{L+1}} = \sum_{m=1}^{N_p} \hat{u}^m_{k_L} \underbrace{\left[\prod_d^D \frac{2}{2n_d+1}\right]^{-1} \int_{[-1,1]^D} \varphi_m(\xi)\varphi_n(\frac{1}{2}\xi \pm \frac{1}{2})d\xi}_{\mathbf{P}_{nm}} \tag{3.6}$$

the coordinates are shifted in the positive or negative direction depending on which sub-cell of the coarse one they represent. The projection can be written as vector matrix multiplication where the coarse modes vector is multiplied by the projection matrix **P** [7],[31]. This projection method is loss-less [7]. Authors of [31] note that depending on the variable (i.e solution component) that we want to communicate between levels, an interpolation or even a re-computation might be better. But in general this method works well for conserved quantities.

3.2 Fine to coarse solution gathering

A generic level L cell D^k_L is decomposed into $N_L = 2^D$ sub-cells D^l_{L+1}:

$$D^k_L = \bigcup_{l=1}^{N_L} D^l_{L+1} \tag{3.7}$$

The gathering operation is equivalent to the cumulative inverse scattering between the coarse cell and its N_L sub-cells. It is formulated as seeking the minimum [9]:

$$\sum_{l=1}^{N_L} \min_{\hat{u}^n_{k_L}} \int_{[-1,1]^D} (u_{h,k_L}(\xi_L) - u_{h,k^l_{L+1}}(\xi_{L+1}))^2 d\xi \tag{3.8}$$

which is simplified exactly as done for the scattering, with only difference that an additional coefficients, due to the sum over sub-cells, is present

[7],[31]:

$$\hat{u}^n_{k_L} = \sum_{m=1}^{N_p} \hat{u}^m_{k_{L+1}} \underbrace{\frac{1}{2^D} \left[\prod_d^D \frac{2}{2n_d + 1} \right]^{-1} \int_{[-1,1]^D} \varphi_n(\xi) \varphi_m(\frac{1}{2}\xi \pm \frac{1}{2}) d\xi}_{(\mathbf{P})^T_{nm}} \tag{3.9}$$

Contrary to the scattering operation, gathering will not always reconstruct the coarse solution exactly. Consecutive refinement and de-refinement of smooth functions is lossless but if discontinuities are present, the original coarse solution is not exactly reconstructed [7].

3.3 Tagging criteria

In the present work, the shock tracking capabilities of AMR were of interest. Therefore a general numerical discontinuity indicator was needed and the TVB detector formulation presented in [21] provided that. In their original work, the authors employed this discontinuity indicator to detect troubled cells, to which then apply a more robust WENO limiter. In addition to using directly limiters to detect discontinuities, other purely analytical (i.e no physical intuition) exist, notably the discontinuity detector presented by Krivodonova [34] and used in [35], [19].

3.3.1 TVB Discontinuity detector

As we have already seen, the TVB limiter is able to locate the onset of oscillations caused in the vicinity of discontinuities. We use the formulation found in [21], [34] which differs from the one used for limiting in the first argument of the minmod function. Here we don't use the linear mode but we use the slope given by the difference between cell boundary value and cell average (assumed to be located at the center of the cell). This "half cell" slope is compared to the slopes between neighbouring cells and current cells. Also these are interpreted as being defined between the current cell center and the outside of the cell boundary. The choice of using a different form of the same limiter was out of curiosity to see if there are differences or not. For the troubled cell D^k the bounded minmod function is evaluated in each dimension separately. If oscillations are detected, the cell is tagged as troubled and no other dimension/direction is checked. Since we need evaluations of cell boundary values, we loop over surfaces quadrature points and evaluate the minmod function fore each one of them. Also here, as soon as troubles get detected we stop looking for more.

Just like we did for limiting, the solution vector of the PDE system is first transformed in its characteristic formulation. Then we define the interface evaluated characteristic polynomial

$$\mathbf{V}^{+}_{h,k_d} = \mathbf{R}^{-1}(\mathbf{U}_{h,k_d}|_{b_d^+} - \hat{\mathbf{U}}^{1}_{h,k_d}) \ ; \ \mathbf{V}^{-}_{h,k_d} = \mathbf{R}^{-1}(\hat{\mathbf{U}}^{1}_{h,k_d} - \mathbf{U}_{h,k_d-1}|_{b_d^-}) \tag{3.10}$$

these modal vectors are evaluated with the minmod function separately [21]:

$$\mathbf{V}^{+,mod}_{h,k_d} = minmodB(\mathbf{V}^{+}_{h,k_d}, \Delta_{+}\overline{\mathbf{V}}_{h,k_d}, \Delta_{-}\overline{\mathbf{V}}_{h,k_d}) \tag{3.11}$$

$$\mathbf{V}^{-,mod}_{h,k_d} = minmodB(\mathbf{V}^{-}_{h,k_d}, \Delta_{+}\overline{\mathbf{V}}_{h,k_d}, \Delta_{-}\overline{\mathbf{V}}_{h,k_d}) \tag{3.12}$$

if the bounded minmod function is activated for both the plus and minus states, i.e if $\mathbf{V}^{+,mod}_{h,k_d} \neq \mathbf{V}^{+}_{h,k_d}$ and $\mathbf{V}^{-,mod}_{h,k_d} \neq \mathbf{V}^{-}_{h,k_d}$, it means that the solution inside cell D^k is oscillating more than its neighbours [21], and therefore it is tagged for refinement. Overall this formulation of the limiter checks for a possible oscillation in all directions and at all specified points and should therefore be reliable.

3.3.2 Other detectors

Curvature based detector: This detector often appears in the literature because of its general formulation and the fact that, similarly to the TVB based one, it is local [23], [25]. This means that it can be evaluated in every cell independently of the global behaviour of the indicator. It can be formulated w.r.t any indicator function θ of interest [23]:

$$\chi_{\partial^2,k} = \left(\frac{\sum_{n,l}(\partial^2\theta/\partial x_n \partial x_l)^2}{\sum_{n,l}\left[\frac{(|\partial\theta/\partial x_n|_{k+1} + |\partial\theta/\partial x_n|_k)}{\Delta x_k} + \varepsilon|\partial^2/\partial x_k \partial x_l||\theta|\right]^2} \right)^{1/2} \tag{3.13}$$

This indicator checks the curvature inside the cell across all dimensions and compares it to the one between the same cell and the neighbouring one. Refinement is triggered if, w.r.t a reference value, the following expression is true [23]:

$$\chi_{\partial^2,k} > \chi_{reF} \tag{3.14}$$

Curl, Divergence and Gradient based detector: These detectors have shown to be able to identify well various types of shock configurations [10], [18], [19]. Their evaluation is expensive because they require a first pass through the mesh to compute the cell indicator values, then the standard deviation over all cells is taken and another pass across the mesh is made to compare the cell indicator with a multiple of the standard deviation [10], [18] ,[19]. The indicators are defined as follows:

$$\chi_{r,k} = |\nabla \times \mathbf{u}|\Delta\overline{x}_k^{3/2} \ ; \ \chi_{d,k} = |\nabla \cdot \mathbf{u}|\Delta\overline{x}_k^{3/2} \ ; \ \chi_{g,k} = |\nabla\rho|\Delta\overline{x}_k^{3/2} \tag{3.15}$$

with $\overline{x}$ being the average cell size of cell k. They are compared to the refinement condition and if evaluated to true then the cell is tagged:

$$\chi_{r,k} > C_{AMR,r}\left(\frac{\sum_{k=1}^{K}\chi_{r,k}^2}{K}\right)^{1/2} \quad ; \quad \chi_{d,k} > C_{AMR,d}\left(\frac{\sum_{k=1}^{K}\chi_{d,k}^2}{K}\right)^{1/2}$$

$$\chi_{g,k} > C_{AMR,g}\left(\frac{\sum_{k=1}^{K}\chi_{g,k}^2}{K}\right)^{1/2} \tag{3.16}$$

with K being the total number of cells and C_{AMR} tunable coefficients. They have been implemented but due to time limitations they could not be tested.

3.4 Software considerations

3.4.1 AMReX

AMReX is a C++ software framework created for massively parallel (exascale), block-structured AMR simulations. The framework provides customization flexibility and a vast base of high performance implementations of core functions. The framework has been developed with parallelization in mind, therefore it already provides most of the tools to ensure proper communication across OpenMP threads and MPI ranks. As a user, some care regarding accessing shared and distributed data is still needed. Currently AMReX is being developed more towards full GPU support. For this project no GPU considerations were made and it is not therefore supported. The main difference between AMReX and other libraries is its block/patch based approach. The specifics and intricacies of how the library organizes its data structures and design philosophy will not be discussed here and are found in AMReX presentation paper [33]. From the base library, a suite of specialized code has been developed throughout the years for e.g combustion, electromagnetic and astrophysical simulations and can be found on the official website. AMReX also provides support for tiled matrix operations (cache blocking) and their parallelization among threads.

3.4.2 AMRDG

The code created for this thesis is based on the AMReX pure virtual class AMRCore, from which it inherits all the AMR capabilities. The ADER-DG code defines a main class called AmrDG, which contains the numerical implementation of the scheme and all the required operations around it (e.g quadrature, fluxes evaluations). The class provides two nested classes which contains a custom implementation for the boundary conditions and scattering-gathering operations. In order to enable an easier further development of the code, a pure virtual class for the implementation of the desired physical model equation is defined inside AmrDG. The user has to provide its own implementation of the test cases, physical fluxes, sources, boundary conditions, eigenvalues, etc. Currently the data is organized in nested vectors of MultiFabs (i.e in AoS style), which is the main distributed data-structure that stores data at each cell. In each cell the modes of the approximated solution polynomial are stored and in case the evaluation of the polynomial is required, the values at quadrature points are stored. The vectors are organized in order to first store level data, then store solution components data, then if required (e.g for fluxes) store dimension dependent data and finally the cell data. A performance analysis and modifications (if needed) should be carried out to ensure an optimal memory access pattern w.r.t the implemented model and w.r.t AMReX design philosophy. The main code comes together with a plotting script written in Python, which uses the yt library to generate AMR plots for the selected models. Also this script was written in way to make it easier to be extended and used as template in the future. Some I/O utilities (e.g restarting simulations, having a parameter file, pre-sets for simulations and plotting, etc.) have not been implemented but would be a nice addition in the future.

Some critical remarks regarding the current AMR approach and possible improvements. Gathering fine to coarse data at each time step to ensure proper synchronization and conservation is not optimal. As discussed, only the numerical fluxes at the grids/patches fine-coarse boundaries interfaces needs to be evaluated. Therefore all covered/overlapped cells evolution can be ignored since their wrong flux predictions will be replaced by the correct fine ones. This was also our strategy at the beginning but during development it was observed that performing this operation using a weighted averaging (AMReX provides functions for this) was not optimal and would lead to inaccuracies. The quadrature free method used for the predictor step fluxes is generally used for the nodal formulations of ADER-DG. Therefore the expansion coefficients are equivalent to point evaluation of the fluxes. In our modal case, we obtain modes of the fluxes, which are not usable as is and require conversion to nodal. Therefore, since the infrastructure for quadrature free computations and projections is there, the numerical fluxes modes could be projected from fine to coarse levels interfaces instead. And then be converted back into nodal for the evaluation at the coarse level.
In addition, the functions that currently perform the gathering operation, acts on all cells of the fine level and store the data in the respective coarse cells. Gathering should only be used when a cell is de-refined. Therefore an algorithm that interfaces with AMReX to check which cells have been coarsened, i.e removed from the fine level, has to be devised. Some development in this regard was started but due to time restrictions could not be developed further. Overall all the tools to allow a proper efficient implementation of AMR are present, and just require to be merged together. As last point, also it

might be worth o explore the possibility of subcycling the time steps; which as previously explained, interfaces well with an ADER-DG framework.

Chapter 4

DG-Euler

4.1 Compressible Euler equations

The Euler equations are a derived from the Navier-Stokes equations with the assumption of inviscid flow, i.e no shear stresses. This assumption holds for Reynolds numbers much greater than one and approaching infinity, such as those present in astrophysical plasmas [28]. For an ideal gas, the system of hyperbolic PDEs in conservative form

$$\frac{\partial}{\partial t}\mathbf{U}(\mathbf{x},t) + \nabla \cdot \mathbf{F}(\mathbf{U}(\mathbf{x},t),\mathbf{x},t). \tag{4.1}$$

has solution vector $\mathbf{U}(\mathbf{x},t) = (\rho, \rho u_1, \rho u_2, \rho u_3, \rho e)^T$ and physical flux tensor $\mathbf{F} = (F_1, F_2, F_3)$ defined as [7]:

$$F_1 = \begin{pmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ \rho u_1 u_3 \\ (\rho e + p) u_1 \end{pmatrix} \;\; ; \;\; F_2 = \begin{pmatrix} \rho u_2 \\ \rho u_2 u_1 \\ \rho u_2^2 + p \\ \rho u_2 u_3 \\ (\rho e + p) u_2 \end{pmatrix} \;\; ; \;\; F_3 = \begin{pmatrix} \rho u_3 \\ \rho u_3 u_1 \\ \rho u_3 u_2 \\ \rho u_3^2 + p \\ (\rho e + p) u_3 \end{pmatrix} \tag{4.2}$$

with velocity vector $\mathbf{u} = (u_1, u_2, u_3)$, density ρ, total energy per unit mass e (ρe is the total energy density) and pressure p. The pressure is given by the equation of state, for simplicity an ideal gas equation is used:

$$p = \rho\varepsilon(\gamma - 1) = [\rho e - \frac{1}{2}\rho\|\mathbf{u}\|^2](\gamma - 1) \;\; ; \;\; e = \varepsilon + \frac{1}{2}\|\mathbf{u}\|^2 \;\; ; \;\; c = \sqrt{\gamma\frac{p}{\rho}} \tag{4.3}$$

with adiabatic coefficient $\gamma = 1.4$, internal energy per unit mass ε and speed of sound c. If external forces acting on the fluids are considered, a source term $\mathbf{S}(\mathbf{x},t)$ is added to the right hand side. To obtain the two-dimensional equations, we just need to drop the third component of the fluxes and the flux in x_3 direction. The eigendecompositon of the flux Jacobian is found in the scientific literature and in our case the eigenvalues, left and right eigenvector matrices were taken from [20]. The maximum signal velocity λ_{max} needed to compute the timestep is given for the Euler equations by the maximum over all dimensions of the cell average absolute value of the velocity plus the speed of sound, i.e $|u_d| + c \quad \forall d = 1, 2, 3$ [7].

In the context of astrophysical simulations, another conserved quantity of interest is the angular momentum. The conservation law for angular momentum is derived by taking the cross product between the momentum conservation and the position vector, i.e $\mathbf{L} = \mathbf{x} \times \rho\mathbf{u}$, leading to a system of conservation laws to be added to the Euler system. The solution vector and flux function are defined as follows [22]:

$$\mathbf{U}(\mathbf{x}, t) = (\underbrace{x_2\rho u_3 - x_3\rho u_2}_{L_1}, \underbrace{x_3\rho u_1 - x_1\rho u_3}_{L_2}, \underbrace{x_1\rho u_2 - x_2\rho u_1}_{L_3})^T \quad ; \quad \mathbf{F} = (F_1, F_2, F_3) \tag{4.4}$$

with components

$$F_1 = \begin{pmatrix} x_2\rho u_3 u_1 - x_3\rho u_1 u_2 \\ x_3(\rho u_1^2 + p) - x_1\rho u_1 u_3 \\ x_1\rho u_1 u_2 - x_2(\rho u_1^2 + p) \end{pmatrix} \quad ; \quad F_2 = \begin{pmatrix} x_2\rho u_2 u_3 - x_3(\rho u_2^2 + p) \\ x_3\rho u_1 u_2 - x_1\rho u_3 u_2 \\ x_1(\rho u_2^2 + p) - x_2\rho u_1 u_2 \end{pmatrix}$$

$$F_3 = \begin{pmatrix} x_2(\rho u_3^2 + p) - x_3\rho u_2 u_3 \\ x_3\rho u_1 u_3 - x_1(\rho u_3^2 + p) \\ x_1\rho u_2 u_3 - x_2\rho u_1 u_3 \end{pmatrix} \tag{4.5}$$

Since the angular momentum conservation equation is directly derived from the linear momentum conservation equation, we are not introducing new unknowns. Still, its conservation is not always ensured by numerical methods [30]. The Discontinuous Galerkin method of order $p > 1$, allows the proper conservation of angular momentum by construction. This because angular momentum is a derived quantity, more importantly it is linearly dependent on the linear momentum components. This allows, by substitution of the linear momentum conservation equations, to recover a weak form conservation law for angular momentum [7]. When limiting is applied to DG, the dampening of the linear modes locally disrupts the conservation of the angular momentum [7]. Strategies should be devised to recover the angular momentum. In the next section we propose a way to achieve this goal.

4.2 Post-limiting angular momentum reconstruction

After limiting is applied we wish to modify the modes of the limited polynomials in order to recover as much as possible the angular momentum. This problem can be formulated as a constrained optimization (regularized regression). The constrain we enforce is that before and after limiting we preserve the angular momentum point-wise, i.e $L_d = \widetilde{L}_d$ respectively (all quantities derived after limiting are noted with $\widetilde{\bullet}$). We utilize as evaluation points the cell quadrature points, $M_p = (p+1)^D$. We utilize the squared norm for both the loss function and the regularizer:

$$\hat{\mathbf{w}} = argmin_{\mathbf{w}} \frac{1}{M_p} \sum_{m=1}^{M_p} \left(\ell(\mathbf{u}_{h,k}(\mathbf{x}_m), \widetilde{\mathbf{u}}_{h,k}(\mathbf{x}_m)) + \sum_{d}^{D} \lambda_d (L_d(\mathbf{x}_m) - \widetilde{L}_d(\mathbf{x}_m))^2 \right) \tag{4.6}$$

with loss function

$$\ell(\mathbf{u}_{h,k}(\mathbf{x}_m), \widetilde{\mathbf{u}}_{h,k}(\mathbf{x}_m)) = \|\mathbf{u}_{h,k}(\mathbf{x}_m) - \widetilde{\mathbf{u}}_{h,k}(\mathbf{x}_m)\|_2^2 = \sum_{q=1}^{D} (u_{h,q}(\mathbf{x}_m) - \widetilde{u}_{h,q}(\mathbf{x}_m))^2 \tag{4.7}$$

with $\mathbf{u}_{h,k}$ representing a DG solution vector with only the momentum components. The post-limiting solution polynomial is parametrized using its modal expansion. Since we want to preserve the cell averages, and thus conservation of linear momentum, the first mode is taken out of the sum. To keep a nice indexing scheme (*i.e* $i = 1, ..., N_p$) after the first mode is taken out, we set the first weight to zero (this allow later to have the full Vandermonde matrix in our system instead of having a version of it not containing the first column):

$$\widetilde{u}_{h,q}(\mathbf{x}_m) = \sum_{i=1}^{N_p} \widetilde{\hat{u}^i}_q \varphi_i(\mathbf{x}_m) \equiv \overline{u}_q + \sum_{i=1}^{N_p} w_q^i \varphi_i(\mathbf{x}_m) \;\; ; \;\; w_q^1 = 0 \tag{4.8}$$

Optimization is performed w.r.t the post-limiting polynomial solution modes (except for the cell average), here called "weights" (noted w). The index q denotes the $q-$th component of the reduced solution vector . This vector contains only the D momentum components of the DG solution vector, because they are the only one influencing the angular momentum. Here only the final result for 3 dimension is reported, but the interested reader may find the derivation, together with the 2 dimensional case in Appendix B. The regression problem is simplified into a system of equations that has to be solved after every limiting operation in every cell of the form:

$$\underbrace{\begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{w}_3 \end{bmatrix}}_{\in \mathbb{R}^{3N_p \times 1}} = \underbrace{\begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_1^- & \mathbf{K}_1^+ \\ \mathbf{K}_2^+ & \mathbf{K}_2 & \mathbf{K}_2^- \\ \mathbf{K}_3^- & \mathbf{K}_3^+ & \mathbf{K}_3 \end{bmatrix}^{-1}}_{\in \mathbb{R}^{3N_p \times 3M_p}} \cdot \underbrace{\begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix}}_{\in \mathbb{R}^{3M_p \times 1}} \tag{4.9}$$

with $\forall\, j = 1, 2, 3$:

$$\begin{aligned} \mathbf{r}_j = \mathbf{V}^T \mathbf{u}_{h,j} + (\lambda_{n_j} C_{n_j} (\mathbf{x}_{l_j} \mathbf{I}_{M_p}) - \lambda_{l_j} C_{l_j} (\mathbf{x}_{n_j} \mathbf{I}_{M_p}) - \mathbf{V}^T) \overline{\mathbf{u}}_j + \lambda_{n_j} C_{n_j} (\mathbf{x}_j \mathbf{I}_{M_p}) \overline{\mathbf{u}}_{l_j} \\ + \lambda_{l_j} C_{l_j} (\mathbf{x}_j \mathbf{I}_{M_p}) \overline{\mathbf{u}}_{n_j} + (\lambda_{n_j} C_{n_j} \mathbf{L}_{n_j} - \lambda_{l_j} C_{l_j} \mathbf{L}_{l_j}) \end{aligned} \tag{4.10}$$

and block matrices

$$\mathbf{K}_j = \mathbf{V}^T\mathbf{V} - \lambda_{n_j} C_{n_j}(\mathbf{x}_{l_j}\mathbf{I}_{M_p})\mathbf{V} + \lambda_{l_j} C_{l_j}(\mathbf{x}_{n_j}\mathbf{I}_{M_p})\mathbf{V}$$

$$\mathbf{K}_j^+ = (\lambda_{n_j} C_{n_j}(\mathbf{x}_j\mathbf{I}_{M_p})\mathbf{V}) \;\; ; \;\; \mathbf{K}_j^- = -(\lambda_{l_j} C_{l_j}(\mathbf{x}_j\mathbf{I}_{M_p})\mathbf{V}) \tag{4.11}$$

and coefficients

$$C_{n_j} = (\mathbf{x}_{l_j}\mathbf{I}_{M_p})\mathbf{V} \;\; ; \;\; C_{l_j} = -(\mathbf{x}_{n_j}\mathbf{I}_{M_p})\mathbf{V} \tag{4.12}$$

where $\mathbf{V}$ is the Vandermonde matrix, $\mathbf{x}_{l_j}$ is a vector containing the l_j-th component of all all M_p points (similarly for $\mathbf{x}_{n_j}$), $\overline{\mathbf{u}}_{n_j}$ is a vector containing M_p times the cell average of solution component n_j (similarly for $\overline{\mathbf{u}}_{n_j}$) and $\mathbf{I}_{M_p}$ is the identity matrix of size M_p.
The matrix $\mathbf{K}$ and the coefficients of the right hand side can be fully pre-computed. The vector of the cell average on the right hand side does not require point-wise evaluations. On the other hand, the evaluation of the polynomial solution should be done for all interpolation points. The regularization parameters λ should set equal to each other and their value fine tuned. This approach has not yet been implemented nor tested due to time constraints. It is expected to reconstruct the angular momentum and satisfy its conservation in the least square sense across all modes. It is clear, that this approach does not scale well with DG order and grid resolution. If proven able to reconstruct the angular momentum, a possible solution could be to solve this system on e.g every other cell and then give its result to neighbouring cells (under the assumption that the solution should not change too much in the neighbourhood). It is also possible that simpler formulations of the constrained optimization, in the sense of less degree of freedom, prove to be sufficient at reconstructing the angular momentum.

4.3 Primitive boundary conditions

It is practical to express our domain boundary conditions w.r.t the primitive variables $\rho, \mathbf{u}, T, p$. To be able to enforce boundary conditions, we thus need to first derive their equivalent conserved expression. In the case of Dirichlet BC we are given $\rho = g_{D,\rho}$, $u_j = g_{D,u_j}$ $\forall\ j = 1, ..., D$ and $p = g_{D,e}$ while for Neumann we have $\frac{\partial}{\partial x_d}\rho = g_{N,\rho}$, $\frac{\partial}{\partial x_d}u_j = g_{N,u_j}$ $\forall\ j = 1, ..., D$ and $\frac{\partial}{\partial x_d}p = g_{N,e}$. The temperature BC treatment has not been implemented because an equivalent pressure BC can directly be derived from it. From the definition of the conserved quantities, if we substitute the correct primitive BCs we obtain the following Dirichlet conditions:

$$\rho = g_{D,\rho} \;\; ; \;\; \rho u_j = g_{D,\rho} g_{D,u_j} \tag{4.13}$$

$$\rho e = \frac{g_{D,e}}{\gamma - 1} + \frac{1}{2}\sum_d^D (g_{D,\rho} g_{D,u_j})^2 \tag{4.14}$$

and Neumann follows:

$$\frac{\partial \rho}{\partial x_d} = g_{N,\rho} \ ; \ \frac{\partial \rho u_j}{\partial x_d} = \rho g_{N,u_j} + \frac{\rho u_j}{\rho} g_{N,\rho} \tag{4.15}$$

$$\frac{\partial \rho e}{\partial x_d} = \frac{g_{N,e}}{\gamma + 1} + \sum_d^D (\rho u_j) \frac{\partial \rho}{\partial x_d} \tag{4.16}$$

Values for density and momentum are recovered from the cell solution evaluated at the boundary interface. The angular momentum BC are derived directly from the Dirichlet/Neumann BC for the velocity. The Neumann BC for angular momentum requires the evaluation of the velocity, which is taken at the boundary interface.

Chapter 5

Numerical validation

In this chapter standard numerical tests are shown and the obtained results discussed. The goal of the presented test cases is to show that the ADER-DG method has been correctly implemented. For the convergence study, the test case solution is known. Therefore representative norms can be computed. The multi-level error norms are computed for individual components of the solution vector, but only the density error norm is reported. The formula for the $q-$th component of the solution is the following:

$$\begin{aligned} \|u_q(\mathbf{x},t) - u_{h,k,q}(\mathbf{x},t)\|_{L^m} &= \left(\frac{1}{|\Omega|}\int_\Omega |u_q(\mathbf{x},t) - u_{h,k,q}(\mathbf{x},t)|^m d\mathbf{x}\right)^{1/m} \\ &= \left(\frac{1}{\sum_l |\Omega_l|}\sum_l^L \int_{\overline{\Omega}_l} |u_q - u_{h,q}|^m d\mathbf{x} - \int_{\widetilde{\Omega}_l} |u_q - u_{h,q}|^m d\mathbf{x}\right)^{1/m} \end{aligned} \tag{5.1}$$

where $\Omega_l = \overline{\Omega}_l - \widetilde{\Omega}_l$ represents the level l contribution to the problem domain and presents some overlap with other levels. $\overline{\Omega}_l$ represents the full level (i.e with overlap) domain, while $\widetilde{\Omega}_l$ represents the portion of $\overline{\Omega}_l$ that overlaps with level $l+1$. The integral is evaluated using $m(p+1)$ quadrature points per dimension and we use $m = 1, 2$.

5.1 Test cases

5.1.1 Isentropic Vortex

The 2-dimensional inviscid isentropic vortex in a free stream test, simulates the passive advection of a vortex across the domain [7], [39], [25]. It is a common benchmark test case because it is smooth and has an analytical solution [39]. The free stream conditions are chosen sot that the vortex moves only in x-direction [39]:

$$\rho_0 = \left[1 - \frac{(\gamma-1)\beta^2}{8\gamma\pi^2}\exp(1-r^2)\right]^{1/(\gamma-1)}$$

$$u_0 = u_\infty + \delta u \ ; \ u_\infty = 1 \ ; \ \delta u = -(y-y_0)\frac{\beta}{2\pi}exp\left(\frac{1-r^2}{2}\right)$$

$$v_0 = v_\infty + \delta v \ ; \ v_\infty = 0 \ ; \ \delta v = (x-x_0)\frac{\beta}{2\pi}exp\left(\frac{1-r^2}{2}\right)$$

$$p_0 = \rho^\gamma$$

with vortex strength $\beta = 5$, $x_0 = 5$ and $y_0 = 5$ indicates the center point location of the vortex, and $\gamma = 1.4$. The periodic domain is $[0,10] \times [0,10]$ and the simulation runs until the final time $T = 10$. At the end of the simulation, the error norm between the DG approximate solution and the analytical one, i.e initial condition, is computed. The procedure is repeated for different mesh sizes (equal in both dimensions) and for different polynomial order. Finally the empirical convergence order is computed and compared to the theoretical predicted one. This test is conducted both with and without AMR. The AMR tagging criteria is a fixed value of the density so that the fine grid is advected with the vortex and always overlaps with approximately the same region. Cells satisfying the condition $\rho \leq 0.8$ are tagged for refinement. In addition, re-meshing checks are made at every time step, this is the worst case scenario and in real application would not happen. Since the problem is smooth, no limiter is needed. The refinement ratio is always of 2.

The goal of this test is to verify the correct implementation of the ADER-DG numerical method from an error norm convergence point of view and to verify if AMR has the same convergence as single level simulations and lower errors. We first look at some AMR convergence results in table 5.1. Shown are the error norms and the respective order of convergence for the density computed by DG-$\mathbb{P}_3$ and DG-$\mathbb{P}_4$ schemes. Density is chosen because it is representative of the entire system evolution. From this data we can see that for a given polynomial order, the empirical order of convergence (EOC) converges and/or oscillates w.r.t the theoretical order of convergence (TOC). The same can be said also for single level test cases presented in table 5.2. When comparing the error norm produced by AMR w.r.t its respective fine and coarse single level errors (tab 5.2), we can see that we are not obtaining the desired error reduction. The AMR simulation, even though it converges correctly, reaches at best the coarse level error. This is surprising, because when solving a linearly advected Gaussian pump in a periodic domain (test case used for development), the AMR error is, as it should, between the errors of its respective coarse and fine levels. Since the single level converges correctly, the issue has to be related only to the AMR implementation. A more in-depth look by using "dummy tests" could help find the problem; which is probably of minor nature.

To point out that the single level very high order $DG - \mathbb{P}_7$ appears to be having a harder time reaching and surpassing the TOC. Compared to other results, for this case the time step size had to be artificially reduced to 50% maximum allowed time step. This has been done for stability reasons. The domain size being not large enough and thus having non-zero values of the solution near the boundaries can be an issue in the context of a periodic domain with exponential functions and very high order approximations, because when advected, a small discontinuity is created at the boundary. The discontinuity is caused by different slopes of the initial condition at opposite boundaries. Increasing the domain size and having a couple more datapoints would definitely shed more light about this slight under-performance.
Overall the convergence results are convincing and pointing towards the correct implementation of the ADER-DG numerical scheme at a single level. The AMR implementation still needs some refinement.

DG-$\mathbb{P}_{(p+1)}$	N_c	$\|u-u_h\|_{L_1}$			$\|u-u_h\|_{L_2}$		
		Error	EOC	TOC	Error	EOC	TOC
DG-$\mathbb{P}_3^{\text{AMR}}$	8	5.4479E-3	-		7.9627E-3	-	
	16	5.3576E-4	3.35		1.9065E-3	2.06	
	32	1.5294E-4	1.81		5.6640E-4	1.75	
	64	1.0598E-5	3.85		4.3642E-5	3.70	
	128	1.1037E-6	3.26		7.8912E-6	2.47	
				3			3
DG-$\mathbb{P}_4^{\text{AMR}}$	8	1.9810E-3	-		2.7900E-3	-	
	16	1.9834E-4	3.32		6.2290E-4	2.16	
	32	1.2760E-5	3.96		4.4081E-5	3.82	
	64	7.4322E-7	4.10		2.9977E-6	3.88	
	128	4.1145E-8	4.17		1.8444E-7	4.022	
				4			4

Table 5.1: L_1, L_2 errors and empirical/theoretical orders of convergence of the density $\rho(\mathbf{x})$ for the 2D isentropic vortex problem for the ADER-DG-$\mathbb{P}_{p+1}$ schemes with AMR activated. One refined level is present. The number of cells is expressed w.r.t the coarse level

DG-$\mathbb{P}_{(p+1)}$	N	$\|u-u_h\|_{L_1}$ Error	EOC	TOC	$\|u-u_h\|_{L_2}$ Error	EOC	TOC
DG-$\mathbb{P}_2$	8	1.8363E-2	-		4.5349E-2	-	
	16	7.1435E-3	1.36		1.9039E-2	1.25	
	32	1.6757E-3	2.09		3.9868E-3	2.26	
	64	2.9186E-4	2.52		7.1226E-4	2.48	
	128	4.4151E-5	2.72		1.1007E-4	2.69	
				2			2
DG-$\mathbb{P}_3$	8	4.4952E-03	-		8.7973E-03	-	
	16	7.5640E-04	2.57		1.7008E-03	2.37	
	32	6.3484E-05	3.57		1.2845E-04	3.73	
	64	7.7768E-06	3.03		1.5800E-05	3.02	
	128	1.1154E-06	2.80		2.3255E-06	2.76	
				3			3
DG-$\mathbb{P}_4$	8	1.4067E-03	-		2.7705E-03	-	
	16	9.1312E-05	3.94		1.7583E-04	3.98	
	32	4.0289E-06	4.50		7.8639E-06	4.482	
	64	1.8033E-07	4.48		3.9797E-07	4.30	
	128	9.2438E-09	4.28		2.1753E-08	4.19	
				4			4
DG-$\mathbb{P}_7$	8	5.4456E-05	-		8.3997E-05	-	
	16	6.9997E-07	6.36		1.0199E-06	6.37	
	32	6.0719E-09	6.85		8.6343E-09	6.89	
	64	1.3763E-10	5.46		3.5373E-10	4.61	
				7			7

Table 5.2: L_1, L_2 errors and empirical/theoretical orders of convergence of the density $\rho(\mathbf{x})$ for the 2D isentropic vortex problem for the ADER-DG-$\mathbb{P}_{p+1}$ schemes on a single level.

5.1.2 Richtmeyer–Meshkov instability

This test cases involves the interaction of a strong shock with a discontinuous density field [17]. The settings for this test case, apart from boundary conditions, are the same used in [38]. We use a 2-dimensional domain of size $[0, 40/3] \times [0, 40]$ discretized with 32×96 cells. The initial conditions are:

$$\rho_0 = d_{1,0.25}\left(y - \left(18 + 2cos\left(\frac{6\pi x}{L}\right)\right)\right) + d_{3.22,0}(|y-4|-2)$$

$$u_0 = v_0 = 0$$

$$p_0 = d_{4.9,1}(|y-4|-2)$$

with $L = 40$ and the discontinuous function smoothed approximation $d_{a,b}(x) = a + 0.5(1 + tanh(sx))(b-a)$ with s=2. The boundary conditions are periodic in x-direction and no flow in y-direction. The goal of this test case is to see if the TVB troubled cell indicator for refinement is effective at tracking the generated flow features. Plots at different times are shown with and without the overlayed AMR mesh. The maximum allowed time step is taken and the simulation is run until a final time $T = 50$. This test case is run using second order DG ($DG - P_2$). The coarse level has a refinement parameter $M = 0.1$, while the finer level just close to it has $M = 10$. For the last finer level it is not needed to specify a refinement ratio. The results are displayed in the coming pages in Fig 5.1-Fig 5.6. In figure 5.1 we observe the strong shock (reddish) that is moving upwards and will impact the small valley. If we look at the refined mesh we notice how the shock front is being refined and also the extremities of the other fluid. After the impact, a secondary shock wave is produced, which is barely visible in Fig 5.3, but its outline can be inferred from where the fine grid has been placed (Fig 5.4). Finally at the end of the simulation we see that an uprising structure has appeared. This structure is being correctly recognized and certain areas refined.

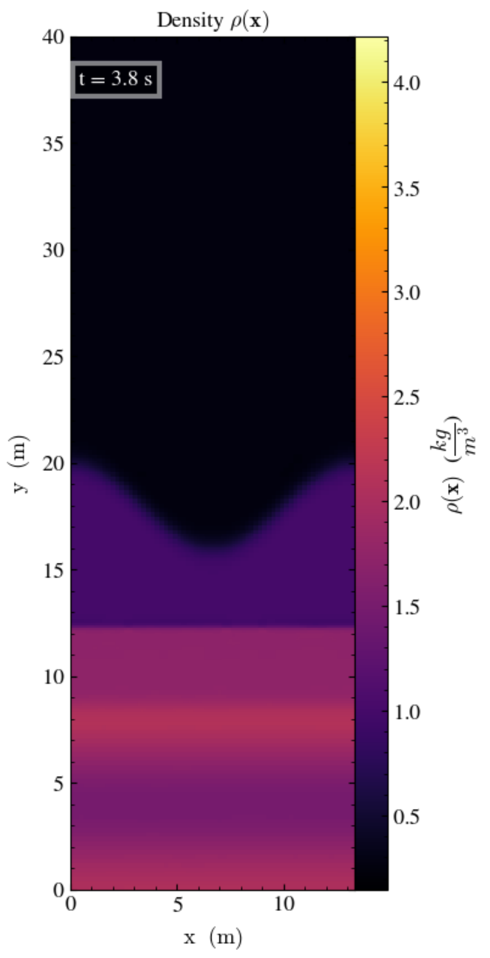


Figure 5.1: Visualization of the density field produced in the Richtmeyer–Meshkov instability test case at $t = 3.8$

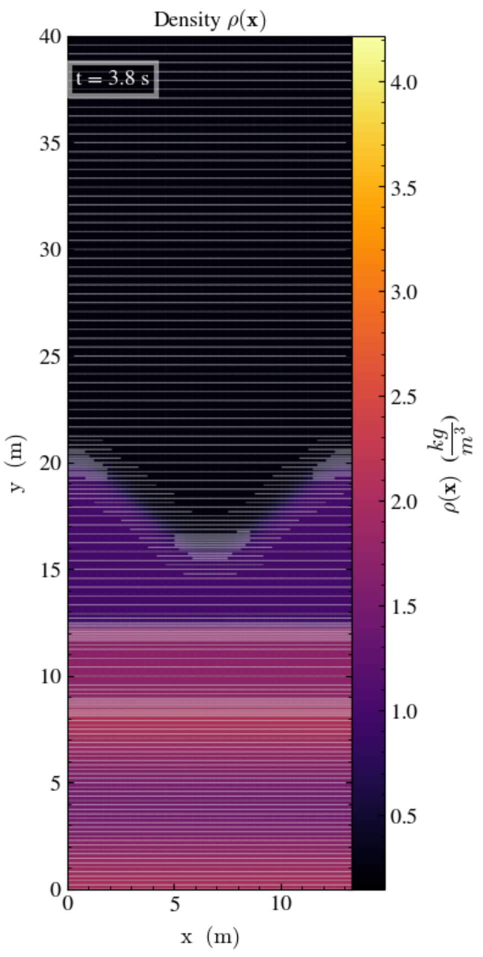


Figure 5.2: Visualization of the density field produced in the Richtmeyer–Meshkov instability test case at $t = 3.8$. This plot also displays the silhouette of the AMR grid.

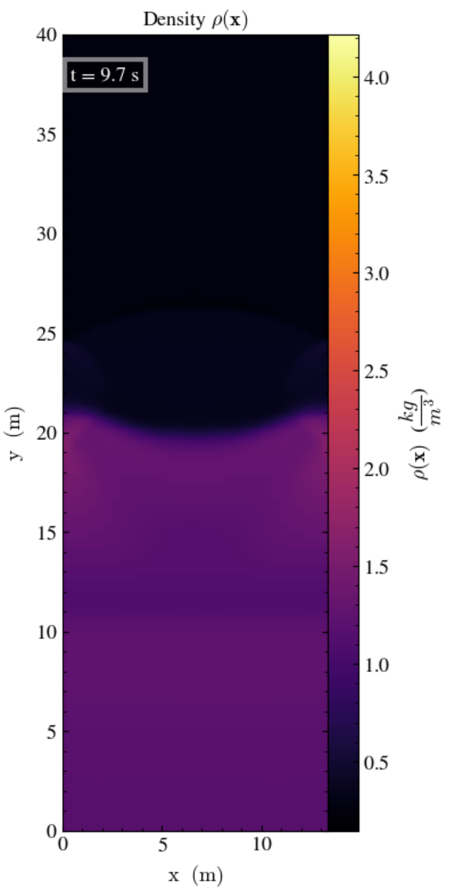


Figure 5.3: Visualization of the density field produced in the Richtmeyer–Meshkov instability test case at $t = 9.7$.

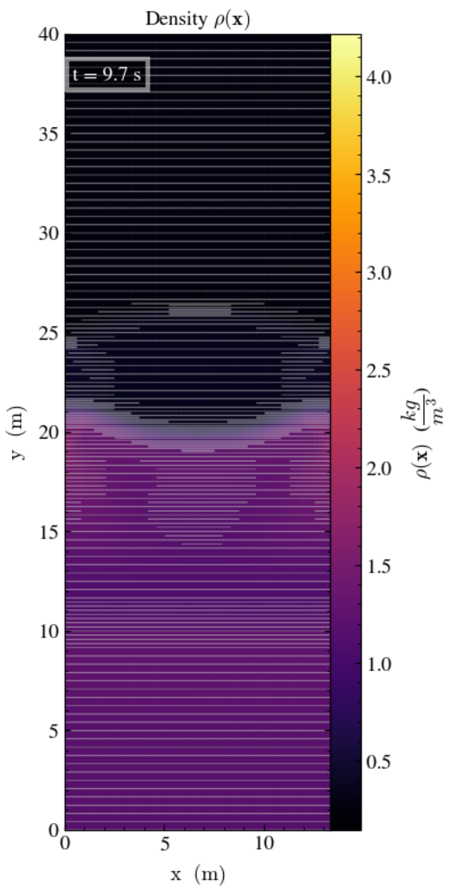


Figure 5.4: Visualization of the density field produced in the Richtmeyer–Meshkov instability test case at $t = 9.7$. This plot also displays the silhouette of the AMR grid.

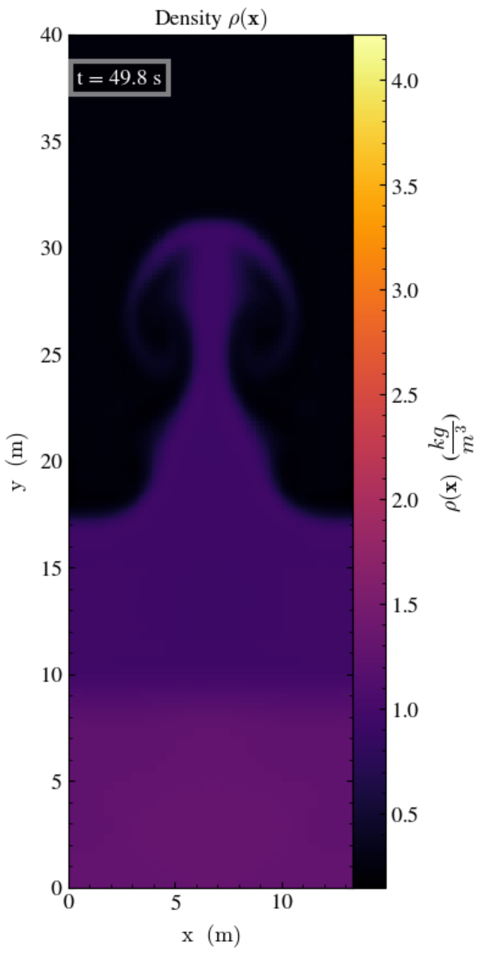


Figure 5.5: Visualization of the density field produced in the Richtmeyer–Meshkov instability test case at $t = 49.8$.

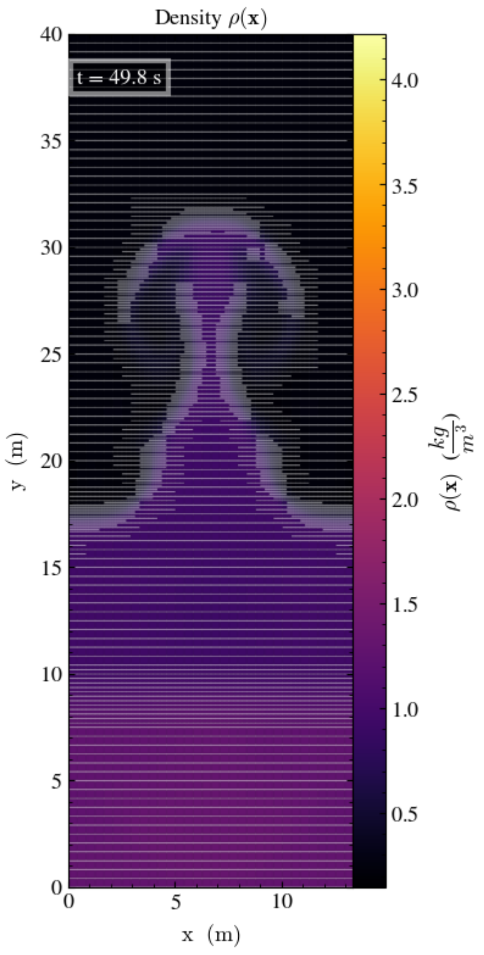


Figure 5.6: Visualization of the density field produced in the Richtmeyer–Meshkov instability test case at $t = 49.8$. This plot also displays the silhouette of the AMR grid.

5.1.3 Kelvin-Helmolz instability

This test case has been taken from [7]. The initial conditions are given by:

$$p = 2.5$$

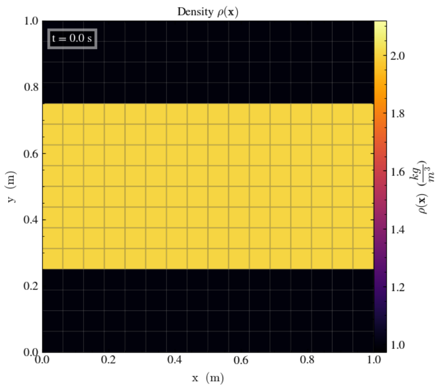
$$\rho(x,y) = \begin{cases} 2 \;\; ; \;\; 0.25 < y < 0.75 \\ -0.5 \;\; ; \;\; else \end{cases}$$

$$u(x,y) = \begin{cases} 0.5 \;\; ; \;\; 0.25 < y < 0.75 \\ -0.5 \;\; ; \;\; else \end{cases}$$

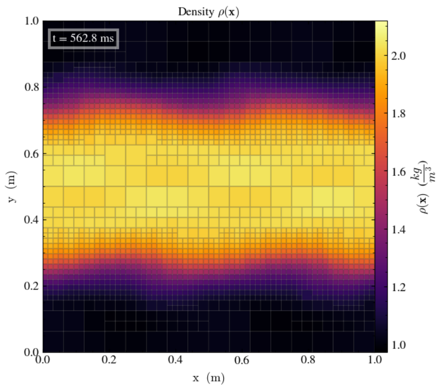
$$v(x,y) = w_0 sin(4\pi x) \left[\exp\left[-\frac{(y-0.25)^2}{2\sigma^2} \right] \right] + \exp\left[-\frac{(y-0.75)^2}{2\sigma^2} \right]$$

with $w_0 = 0.1$, $\sigma = 0.05/\sqrt{2}$, $\gamma = 1.4$. The domain size is $[0,1]^2$ and the coarsest level is made of 16 cells per dimension. Two additional refined layer are introduced, each one doubling the cell count, thus having at the finest level maximum 64 cells per dimensions. The simulation is run for $T = 10$. This problem contains two sharp discontinuities at the beginning and steep gradients can appear later on [7]. The goal of this test was to see how well the grid can adapt with rapidly changing level geometries. In addition the limiter could also be tested (it has also been done for the Richtmeyer instability, but the results not reported) This test is run with fourth order DG ($DG - P_4$) and limiting is necessary to avoid crashes at the beginning of the simulation. Limiting is applied only on the finest level. Refinement is based again on the TVB indicator. From the coarsest level to the finest one, the values of the TVB coefficient are $M = 30, 50, 60$. After the simulation has been done, the data is plotted and 4 selected representative plots showed here. The results can be seen in the next page in Fig. 5.7. Plotted is the density field.

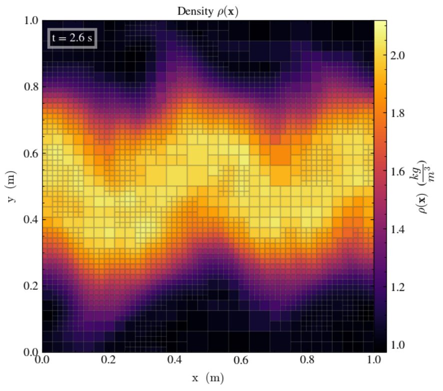
By looking at the results we see that the shear forces properly act on the fluids [7], making them roll up at the borders. The adaptive mesh follows well the profile and also tracks other flow structures moving across the central area. The interfaces between the two fluids appear a bit smeared out.



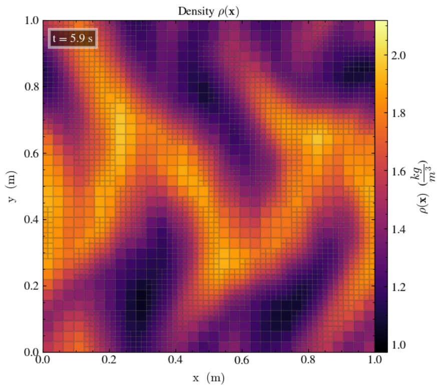
(a) $t = 0$



(b) $t = 0.562$



(c) $t = 2.6$



(d) $t = 5.9$

Figure 5.7: Visualization of the density field induced by the Kevin-Helmolz instability at 4 different times. Adaptive mesh is plotted on top of the field. There are 2 levels of refinement. Refining is triggered by the TVD indicator. Each level has a different TVB coefficient M: $(l_0, M = 30)$,$(l_1, M = 50)$,$(l_2, M = 60)$. Solution computed using $DG - P_4$. Limiting is applied at the finest layer.

Chapter 6

Conclusion

The ADER-DG method provides a powerful numerical tool to carry out simulations and utilize efficiently the benefits provided by AMR. The methodology behind these two numerical components has been explained and applied. Especially in the AMR part of the framework, there is still room for improvement, suggestions to this regard have been made in this document. The provided implementation reaches the desired order of accuracy but some errors are introduced in the solution when employing the AMR part, making it not yet fully operative. We have seen that a simple slope limiter can be employed to provide troubled cell detection in the context of AMR.

Acknowledgements: I'd like to thank Roger Käppeli for his technical support, insightful observations and the friendly co-operation. The simulations have been run the Euler HPC cluster of ETH Zürich.

Appendix A

Evaluation of system matrices

A.1 Quadrature formulas

Taken from their definition in [51]: $\forall\ 1 \leq q \leq \lfloor \frac{1}{3}(p+1) \rfloor$:

$$\xi_{q,d} = \left(1 - \frac{1}{8(p+1)^2} + \frac{1}{8(p+1)^3} - \frac{1}{384(p+1)^4}\left(39 - 28csc^2\left(\frac{4q-1}{4(p+1)+2}\right)\right)\right)\cos\left(\frac{4q-1}{4(p+1)+2}\right)$$

$\forall\ \frac{1}{3}(p+1)\rfloor < q \leq \lfloor \frac{1}{2}(p+1) \rfloor$:

$$\xi_{q,d} = \left(1 - \frac{1}{8(p+1)^2} + \frac{1}{8(p+1)^3}\right)\cos\left(\frac{4q-1}{4(p+1)+2}\right)$$

The quadrature weights canthen computed using the first associated Legendre polynomials (provided by the standard C++ library)

$$w_{q,d} = \frac{2}{(1-\xi_{q,d}^2)[P_n'(\xi_{q,d})]^2} = \frac{2}{[P_n^1(\xi_{q,d})]^2} \ ; \ n = (p+1)$$

A.2 Stiffness matrix analytic solution

$$\int_{[-1,1]^D} \varphi_i(\xi)\frac{\partial}{\partial \xi_d}\varphi_j(\xi)d\xi = \int_{[-1,1]^D}\left(\prod_{d=1}^{D} P_{i_d}(\xi_d)\right)\left(\prod_{a=1\neq d}^{D} P_{j_a}(\xi_a)\right)\frac{\partial}{\partial \xi_d}P_{j_d}(\xi_d)d\xi_1...d\xi_D$$

$$= \int_{[-1,1]^D}\left(\prod_{a=1\neq d}^{D} P_{i_a}(\xi_a)P_{j_a}(\xi_a)\right)P_{i_d}(\xi_d)\frac{\partial}{\partial \xi_d}P_{j_d}(\xi_d)d\xi_1...d\xi_D$$

$$= \left(\prod_{a=1\neq d}^{D} \frac{2}{2j_a+1}\delta_{i_a j_a}\right)\int_{-1}^{1} P_{i_d}(\xi_d)\frac{\partial}{\partial \xi_d}P_{j_d}(\xi_d)d\xi_d$$

the derivative can be rewritten using the first order ($P^1_{j_d}(\xi_d)$) associated Legendre polynomial and using Legendre recurrence formulas and identities [11],[12]. I.e starting from the definition of associated Legendre polynomial $P^m_{j_d}(\xi_d)$:

$$\frac{d^m}{d\xi_d^m}P_{j_d}(\xi_d) = P^m_{j_d}(\xi_d)(-1)^{-m}(1-\xi_d^2)^{-m/2}$$

we set $m = 1$ to find the first equality:

$$\frac{\partial}{\partial \xi_d}P_{j_d}(\xi_d) = -\frac{1}{\sqrt{1-\xi_d^2}}P^1_{j_d}(\xi_d)$$

by applying recurrence formulas found in [11],[12] to the right hand side,we then get:

$$\frac{\partial}{\partial \xi_d}P_{j_d}(\xi_d) = \frac{1}{2}P^2_{j_d+1}(\xi_d) + \frac{1}{2}j_d(j_d+1)P_{j_d+1}$$

where P_{j_d} indicates the 0th order Legendre polynomial $P^0_{j_d}$. Plugging back into the integral the derivative term leads to:

$$\int_{-1}^{1} P_{i_d}(\xi_d)\frac{\partial}{\partial \xi_d}P_{j_d}(\xi_d)d\xi_d = \frac{1}{2}\underbrace{\int_{-1}^{1} P_{i_d}(\xi_d)P^2_{j_d+1}(\xi_d)d\xi_d}_{(I)} + \frac{1}{2}j_d(j_d+1)\underbrace{\int_{-1}^{1} P_{i_d}(\xi_d)P_{j_d+1}(\xi_d)d\xi_d}_{(II)}$$

The second term (II) is easily computed using the Legendre polynomials orthogonality:

$$(II) = \frac{2}{2k_d+1}\delta_{i_d k_d} \;\; ; \;\; k_d = j_d + 1$$

It now remains to find expressions for the first integral (I):

$$\int_{-1}^{1} P_{i_d}(\xi_d)P^2_{j_d-1}(\xi_d)d\xi_d$$

In general, the associated Legendre polynomials $P^m_{k_d}(\xi_d)$ are of order k_d if m is even. If that is the case, then we can expand it in terms of zero order (classic) Legendre polynomials $P_l(\xi_d)$, $0 \leq l \leq k_d$ [11]. In our framework we are dealing with m=2, therefore such expansion is possible and reads as follows [11]:

$$P^2_{k_d}(\xi_d) = \sum_{l=0}^{k_d} c_l P_l(\xi_d) \;\; ; \;\; k_d = j_d + 1$$

$$c_{k_d} = -k_d(k_d - 1) \;\; ; \;\; c_l = (2l+1)(1 + (-1)^{l+k_d}) \;\; \forall \; l < k_d$$

Plugging the expansion in the integral and employing orthogonality allows us to therefore we can rewrite term (I) as:

$$(I) = \int_{-1}^{1} P_{i_d}(\xi_d) \sum_{l=0}^{k_d} c_l P_l(\xi_d) d\xi_d = \sum_{l=0}^{k_d} c_l \int_{-1}^{1} P_{i_d}(\xi_d) P_l(\xi_d) d\xi_d = \sum_{l=0}^{k_d} c_l \frac{2}{2l+1} \delta_{i_d l}$$

Finally the stiffness matrix entries can be computed with the following formula:

$$\int_{[-1,1]^D} \varphi_i(\xi) \frac{\partial}{\partial \xi_d} \varphi_j(\xi) d\xi = \\ \left(\prod_{a=1 \neq d}^{D} \frac{2}{2j_a + 1} \delta_{i_a j_a} \right) \left(\frac{1}{2} \sum_{l=0}^{k_d} c_l \frac{2}{2l+1} \delta_{i_d l} + \frac{1}{2} j_d(j_d+1) \frac{2}{2k_d+1} \delta_{i_d k_d} \right)$$

Appendix B

Angular momentum reconstruction

The index k to indicate cell D^k will be dropped for the remaining of the section. The post-limiting solution polynomial is parametrized using its modal expansion. Since we want to preserve the cell averages, and thus conservation of linear momentum, the first mode is taken out of the sum. To keep a nice indexing scheme ($i.e\ i = 1, ..., N_p$) after the first mode is taken out, we set the first weight to zero (this allow later to have the full Vandermonde matrix in our system instead of having a version of it not containing the first column):

$$\widetilde{u}_{h,q}(\mathbf{x}_m) = \sum_{i=1}^{N_p} \widetilde{\hat{u}^i}_q \varphi_i(\mathbf{x}_m) \equiv \overline{u}_q + \sum_{i=1}^{N_p} w_q^i \varphi_i(\mathbf{x}_m) \ \ ; \ \ w_q^1 = 0$$

We can therefore rewrite the problem as

$$\hat{\mathbf{w}} = argmin_{\mathbf{w}} \frac{1}{M_p} \sum_{m=1}^{M_p} \left(\sum_{q=1}^{D} (u_{h,q}(\mathbf{x}_m) - \overline{u}_q - \sum_{i=1}^{N_p} w_q^i \varphi_i(\mathbf{x}_m))^2 + \sum_{d}^{D} \lambda_d (L_d(\mathbf{x}_m) - \widetilde{L}_d(\mathbf{x}_m))^2 \right)$$

with $\mathbf{w} = (\mathbf{w}_1, ..., \mathbf{w}_D)$, $\mathbf{w}_q = \{w_q^i\}_{i=1}^{N_p}$ with $w_q^1 = 0$. The angular momentum in direction d can be written using a multi index notation as:

$$L_d = x_{n_d} u_{h,l_d} - x_{l_d} u_{h,n_d} \ \ ; \ \ \widetilde{L}_d = x_{n_d} \widetilde{u}_{h,l_d} - x_{l_d} \widetilde{u}_{h,n_d}$$

with $n_d, l_d = 1, ..., D$, $n_d \neq l_d$ and for $d = 1$ we have $(n_1, l_1) = (2, 3)$, $(n_2, l_2) = (3, 1)$ for $d = 2$ and finally $(n_3, l_3) = (1, 2)$ for $d = 3$. The limited angular momentum (evaluated at point $\mathbf{x}_m$) is rewritten w.r.t the weights:

$$\widetilde{L}_d(\mathbf{x}_m) = (\mathbf{x}_m)_{n_d} \widetilde{u}_{h,l_d}(\mathbf{x}_m) - (\mathbf{x}_m)_{l_d} \widetilde{u}_{h,n_d}(\mathbf{x}_m)$$

$$= (\mathbf{x}_m)_{n_d}(\overline{u}_{l_d} + \varphi(\mathbf{x}_m)\mathbf{w}_{l_d}) - (\mathbf{x}_m)_{l_d}(\overline{u}_{n_d} + \varphi(\mathbf{x}_m)\mathbf{w}_{n_d})$$

The notation $(\mathbf{x}_m)_{n_d}$ means that we take the n_d-th component of the point $\mathbf{x}_m$. The minimization problem is further simplified to obtain a matrix-vector formulation.

$$\nabla_{\mathbf{w}} \sum_{m=1}^{M_p} \left(\sum_{q=1}^{D} (u_{h,q}(\mathbf{x}_m) - \overline{u}_q - \underbrace{\sum_{i=1}^{N_p} w_q^i \varphi_i(\mathbf{x}_m)}_{=\varphi(\mathbf{x}_m)\mathbf{w}_q})^2 + \sum_{d=1}^{D} \lambda_d (L_d(\mathbf{x}_m) - \widetilde{L}_d(\mathbf{x}_m))^2 \right) = \mathbf{0}$$

with $\varphi(\mathbf{x}_m) = \{\varphi_i(\mathbf{x}_m)\}_{i=1}^{N_p}$. The full weights gradient is composed of D smaller gradient vectors $\nabla_{\mathbf{w}} = (\nabla_{\mathbf{w}_1}, ..., \nabla_{\mathbf{w}_D})$. By taking the gradient inside of the sum over the points, and only considering the weights we are taking the derivative with respect to; then $\forall j = 1, ...D$:

$$\nabla_{\mathbf{w}_j} \|\mathbf{u}_{h,j} - \overline{\mathbf{u}}_j - \mathbf{V}\mathbf{w}_j\|_2^2 + \sum_{d=1}^{D} \lambda_d \nabla_{\mathbf{w}_j} \|\mathbf{L}_\mathbf{d} - \widetilde{\mathbf{L}}_d\|_2^2 = \mathbf{0}$$

with

$$\mathbf{V} = \begin{bmatrix} \varphi_1(\mathbf{x}_1) & \cdots & \varphi_{N_p}(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ \varphi_1(\mathbf{x}_m) & \cdots & \varphi_{N_p}(\mathbf{x}_m) \\ \vdots & \vdots & \vdots \\ \varphi_1(\mathbf{x}_{M_p}) & \cdots & \varphi_{N_p}(\mathbf{x}_{M_p}) \end{bmatrix} \in \mathbb{R}^{M_p \times N_p}$$

$$\mathbf{u}_{h,j} = [u_{h,j}(\mathbf{x}_1), ..., u_{h,j}(\mathbf{x}_m), ..., u_{h,j}(\mathbf{x}_{M_p})]^T \in \mathbb{R}^{M_p \times 1} \ ; \ \overline{\mathbf{u}}_j = [\overline{u}_j, ..., \overline{u}_j]^T \in \mathbb{R}^{M_p \times 1}$$

$$\widetilde{\mathbf{L}}_d = (\mathbf{x}_{n_d}\mathbf{I}_{M_p})(\overline{\mathbf{u}}_{l_d} + \mathbf{V}\mathbf{w}_{l_d}) - (\mathbf{x}_{l_d}\mathbf{I}_{M_p})(\overline{\mathbf{u}}_{n_d} + \mathbf{V}\mathbf{w}_{n_d})$$

$$\mathbf{x}_{n_d} = [(\mathbf{x}_1)_{n_d}, ..., (\mathbf{x}_m)_{n_d}, ..., (\mathbf{x}_{M_p})_{n_d}]^T \in \mathbb{R}^{M_p \times 1}$$

To note that, the points used for the evaluation of the Vandermonde matrix share the same notation as the vector of points used for the angular momentum above; but they are not the same thing. Applying the derivative to both terms leads to:

$$\mathbf{V}^T(\mathbf{V}\mathbf{w}_j - \mathbf{u}_{h,j} + \overline{\mathbf{u}}_j) - \sum_{d=1}^{D} \lambda_d (\mathbf{L}_d - \widetilde{\mathbf{L}}_d) \nabla_{\mathbf{w}_j} \widetilde{\mathbf{L}}_d = \mathbf{0}$$

for convenience the term C_d is defined as the limited angular momentum derivative and it holds that $C_d \neq 0$ only if $d \neq j$ (i.e the angular momentum does not depend on the coordinate in its own direction):

$$C_d = \nabla_{\mathbf{w}_j} \widetilde{\mathbf{L}}_d = (\mathbf{x}_{n_d}\mathbf{I}_{M_p})\mathbf{V}\delta_{j,l_d} - (\mathbf{x}_{l_d}\mathbf{I}_{M_p})\mathbf{V}\delta_{j,n_d}$$

the system is rewritten with weights dependent term on the left hand side

$$\underbrace{\mathbf{V}^T\mathbf{V}\mathbf{w}_j - \sum_{d=1}^{D} \lambda_d C_d[(\mathbf{x}_{n_d}\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_{l_d}) - (\mathbf{x}_{l_d}\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_{n_d})]}_{(*)}$$
$$= \underbrace{\mathbf{V}^T(\mathbf{u}_{h,j} - \overline{\mathbf{u}}_j) - \sum_{d=1}^{D} \lambda_d C_d[\mathbf{L}_d - (\mathbf{x}_{n_d}\mathbf{I}_{M_p})(\overline{\mathbf{u}}_{l_d}) + (\mathbf{x}_{l_d}\mathbf{I}_{M_p})(\overline{\mathbf{u}}_{n_d})]}_{(**)}$$

Now we would like to remove the sum over the dimensions. For this reasons we try to see if some indexing patter appears between j and d. For e.g. $j = 1$, $(*)$ becomes:

$$\mathbf{V}^T\mathbf{V}\mathbf{w}_1 - \lambda_2 C_2[(\mathbf{x}_{n_2}\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_{l_2}) - (\mathbf{x}_{l_2}\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_{n_2})]$$
$$- \lambda_3[(\mathbf{x}_{n_3}\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_{l_3}) - (\mathbf{x}_{l_3}\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_{n_3})]C_3$$

we know that L_2 has coefficients $n_2 = 3$, $l_2 = 1$, and L_3 has coefficients $n_3 = 1$, $l_3 = 2$:

$$\mathbf{V}^T\mathbf{V}\mathbf{w}_1 - \lambda_2[(\mathbf{x}_3\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_1) - (\mathbf{x}_1\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_3)]C_2$$
$$- \lambda_3[(\mathbf{x}_1\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_2) - (\mathbf{x}_2\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_1)]C_3$$

$$C_2 = (\mathbf{x}_3\mathbf{I}_{M_p})\mathbf{V}\delta_{j,1} - (\mathbf{x}_1\mathbf{I}_{M_p})\mathbf{V}\delta_{j,3} = (\mathbf{x}_3\mathbf{I}_{M_p})\mathbf{V}$$
$$C_3 = (\mathbf{x}_1\mathbf{I}_{M_p})\mathbf{V}\delta_{j,2} - (\mathbf{x}_2\mathbf{I}_{M_p})\mathbf{V}\delta_{j,1} = -(\mathbf{x}_2\mathbf{I}_{M_p})\mathbf{V}$$

L_1 has coefficients $n_1 = 2$, $l_1 = 3$. By rearranging the terms we can rewrite the system as:

$$\left(\mathbf{V}^T\mathbf{V} - \lambda_2 C_2(\mathbf{x}_3\mathbf{I}_{M_p})\mathbf{V} + \lambda_3 C_3(\mathbf{x}_2\mathbf{I}_{M_p})\mathbf{V}\right)\mathbf{w}_1$$
$$+ (\lambda_2 C_2(\mathbf{x}_1\mathbf{I}_{M_p})\mathbf{V})\mathbf{w}_3 - (\lambda_3 C_3(\mathbf{x}_1\mathbf{I}_{M_p})(\mathbf{V})\mathbf{w}_2$$

By inspecting the indices relations between each other and w.r.t index j also for the remaining dimensions, we can rewrite a more general form

$$(*) = \underbrace{\left(\mathbf{V}^T\mathbf{V} - \lambda_{n_j} C_{n_j}(\mathbf{x}_{l_j}\mathbf{I}_{M_p})\mathbf{V} + \lambda_{l_j} C_{l_j}(\mathbf{x}_{n_j}\mathbf{I}_{M_p})\mathbf{V}\right)}_{\mathbf{K}_j}\mathbf{w}_j$$
$$+ \underbrace{(\lambda_{n_j} C_{n_j}(\mathbf{x}_j\mathbf{I}_{M_p})\mathbf{V})}_{\mathbf{K}_j^+}\mathbf{w}_{l_j} + \underbrace{-(\lambda_{l_j} C_{l_j}(\mathbf{x}_j\mathbf{I}_{M_p})\mathbf{V})}_{\mathbf{K}_j^-}\mathbf{w}_{n_j}$$

$$C_{n_j} = (\mathbf{x}_{l_j}\mathbf{I}_{M_p})\mathbf{V}$$

$$C_{l_j} = -(\mathbf{x}_{n_j}\mathbf{I}_{M_p})\mathbf{V}$$

We now focus on the right hand side (**) and try to simplify it using the same approach as before:

$$\mathbf{V}^T(\mathbf{u}_{h,j} - \overline{\mathbf{u}}_j) - \lambda_{n_j}C_{n_j}[\mathbf{L}_{n_j} - (\mathbf{x}_{l_j}\mathbf{I}_{M_p})(\overline{\mathbf{u}}_j) + (\mathbf{x}_j\mathbf{I}_{M_p})(\overline{\mathbf{u}}_{l_j})] \\ - \lambda_{l_j}C_{l_j}[\mathbf{L}_{l_j} - (\mathbf{x}_j\mathbf{I}_{M_p})(\overline{\mathbf{u}}_{n_j}) + (\mathbf{x}_{n_j}\mathbf{I}_{M_p})(\overline{\mathbf{u}}_j)]$$

which can be rewritten as (grouping similar terms)

$$(**) = \mathbf{V}^T\mathbf{u}_{h,j} + (\lambda_{n_j}C_{n_j}(\mathbf{x}_{l_j}\mathbf{I}_{M_p}) \\ - \lambda_{l_j}C_{l_j}(\mathbf{x}_{n_j}\mathbf{I}_{M_p}) - \mathbf{V}^T)\overline{\mathbf{u}}_j + \lambda_{n_j}C_{n_j}(\mathbf{x}_j\mathbf{I}_{M_p})\overline{\mathbf{u}}_{l_j} \\ + \lambda_{l_j}C_{l_j}(\mathbf{x}_j\mathbf{I}_{M_p})\overline{\mathbf{u}}_{n_j} + (\lambda_{n_j}C_{n_j}\mathbf{L}_{n_j} - \lambda_{l_j}C_{l_j}\mathbf{L}_{l_j}) = \mathbf{r}_j$$

Finally everything is condensed into a system of equations as shown in the apposite chapter.

2-dimensional case In 2-dimensions the sums over dimensions goes away since we only have the angular momentum defined w.r.t third direction component.

$$\mathbf{V}^T\mathbf{V}\mathbf{w}_j - \lambda_3C_3[(\mathbf{x}_1\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_2) - (\mathbf{x}_2\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_1)] = \\ \mathbf{V}^T(\mathbf{u}_{h,j} - \overline{\mathbf{u}}_j) - \lambda_3C_3[\mathbf{L}_3 - (\mathbf{x}_1\mathbf{I}_{M_p})(\overline{\mathbf{u}}_2) + (\mathbf{x}_2\mathbf{I}_{M_p})(\overline{\mathbf{u}}_1)]$$

$$C_3 = (\mathbf{x}_1\mathbf{I}_{M_p})\mathbf{V}\delta_{j,2} - (\mathbf{x}_2\mathbf{I}_{M_p})\mathbf{V}\delta_{j,1}$$

j=1:

$$\underbrace{(\mathbf{V}^T\mathbf{V} + \lambda_3C_3(\mathbf{x}_2\mathbf{I}_{M_p})\mathbf{V})}_{\mathbf{K}_1}\mathbf{w}_1 + \underbrace{-(\lambda_3C_3(\mathbf{x}_1\mathbf{I}_{M_p})\mathbf{V})}_{\mathbf{K}_1^-}\mathbf{w}_2 \\ = \underbrace{\mathbf{V}^T\mathbf{u}_{h,1} - (\mathbf{V}^T + \lambda_3C_3(\mathbf{x}_2\mathbf{I}_{M_p}))\overline{\mathbf{u}}_1 + \lambda_3C_3(\mathbf{x}_1\mathbf{I}_{M_p})\overline{\mathbf{u}}_2 - \lambda_3C_3\mathbf{L}_3}_{\mathbf{r}_1}$$

j=2:

$$\underbrace{(\mathbf{V}^T\mathbf{V} - \lambda_3C_3(\mathbf{x}_1\mathbf{I}_{M_p})\mathbf{V})}_{\mathbf{K}_2}\mathbf{w}_2 + \underbrace{(\lambda_3C_3(\mathbf{x}_2\mathbf{I}_{M_p})\mathbf{V})}_{\mathbf{K}_2^+}\mathbf{w}_1 \\ = \underbrace{\mathbf{V}^T\mathbf{u}_{h,2} + (\lambda_3C_3(\mathbf{x}_1\mathbf{I}_{M_p}) - \mathbf{V}^T)\overline{\mathbf{u}}_2 - \lambda_3C_3(\mathbf{x}_2\mathbf{I}_{M_p})\overline{\mathbf{u}}_1 - \lambda_3C_3\mathbf{L}_3}_{\mathbf{r}_2}$$

$$\underbrace{\begin{bmatrix}\mathbf{w}_1\\ \mathbf{w}_2\end{bmatrix}}_{\in\mathbb{R}^{2N_p\times1}} = \underbrace{\begin{bmatrix}\mathbf{K}_1 & \mathbf{K}_1^-\\ \mathbf{K}_2^+ & \mathbf{K}_2\end{bmatrix}^{-1}}_{\in\mathbb{R}^{2N_p\times2M_p}} \cdot \underbrace{\begin{bmatrix}\mathbf{r}_1\\ \mathbf{r}_2\end{bmatrix}}_{\in\mathbb{R}^{2M_p\times1}}$$

Test of indices correctness with $j = 2$ in $D = 3$

$$\mathbf{V}^T\mathbf{V}\mathbf{w}_2 - \lambda_1 C_1[(\mathbf{x}_2\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_3) - (\mathbf{x}_3\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_2)] - \lambda_3 C_3[(\mathbf{x}_1\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_2) - (\mathbf{x}_2\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_1)]$$

$$C_1 = (\mathbf{x}_2\mathbf{I}_{M_p})\mathbf{V}\delta_{j,3} - (\mathbf{x}_3\mathbf{I}_{M_p})\mathbf{V}\delta_{j,2} = -(\mathbf{x}_3\mathbf{I}_{M_p})\mathbf{V} \rightarrow C_{l_j} = -(\mathbf{x}_{n_j}\mathbf{I}_{M_p})\mathbf{V}$$

$$C_3 = (\mathbf{x}_1\mathbf{I}_{M_p})\mathbf{V}\delta_{j,2} - (\mathbf{x}_2\mathbf{I}_{M_p})\mathbf{V}\delta_{j,1} = (\mathbf{x}_1\mathbf{I}_{M_p})\mathbf{V} \rightarrow C_{n_j} = (\mathbf{x}_{l_j}\mathbf{I}_{M_p})\mathbf{V}$$

$$\mathbf{V}^T\mathbf{V}\mathbf{w}_j - \lambda_{l_j} C_{l_j}[(\mathbf{x}_j\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_{n_j}) - (\mathbf{x}_{n_j}\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_j)] - \lambda_{n_j} C_{n_j}[(\mathbf{x}_{l_j}\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_j) - (\mathbf{x}_j\mathbf{I}_{M_p})(\mathbf{V}\mathbf{w}_{l_j})]$$

$$(\mathbf{V}^T\mathbf{V} - \lambda_{n_j} C_{n_j}(\mathbf{x}_{l_j}\mathbf{I}_{M_p})\mathbf{V} + \lambda_{l_j} C_{l_j}(\mathbf{x}_{n_j}\mathbf{I}_{M_p})\mathbf{V})\mathbf{w}_j + \lambda_{n_j} C_{n_j}(\mathbf{x}_j\mathbf{I}_{M_p})\mathbf{V})\mathbf{w}_{l_j} - \lambda_{l_j} C_{l_j}(\mathbf{x}_j\mathbf{I}_{M_p})\mathbf{V})\mathbf{w}_{n_j}$$

We get same form as for $j = 1$.

Bibliography

[1] D. Wirasae *et al.*2014.DISCONTINUOUS GALERKIN METHODS WITH NODAL AND HYBRID MODAL/NODAL TRIANGULAR, QUADRILATERAL AND POLYGONAL ELEMENTS FOR NONLINEAR SHALLOW WATER FLOW.Comp.Methods Appl. Mech. Engrg.http://dx.doi.org/10.1016/j.cma.2013.11.006

[2] Satyvir Singh.2022.A MIXED-TYPE MODAL DISCONTINUOUS GALERKIN APPROACH FOR SOLVING NONLINEAR REACTION DIFFUSION EQUATIONS.AIP Conference Proceeding.https://doi.org/10.1063/5.0103736

[3] T.C. Warburton *et al.*1998.GALERKIN AND DISCONTINUOUS GALERKIN SPECTRAL/HP METHODS.Comp.Methods Appl. Mech. Engrg.https://doi.org/10.1016/S0045-7825(98)00360-0

[4] T.C. Warburton and G.E. Karniadakis.1999.A DISCONTINUOUS GALERKIN METHOD FOR THE VISCOUS MHD EQUATIONS.J. Computational Physics.https://doi.org/10.1006/jcph.1999.6248

[5] Jan S. Hesthaven and Tim Warburton.2008.NODAL DISCONTINUOUS GALERKIN METHODS.TAM Springer.https://doi.org/10.1007/978-0-387-72067-8

[6] Bernardo Cockburn and Chi-Wang Shu.2001.RUNGE-KUTTA DISCONTINUOUS GALERKIN METHODS FOR CONVECTION-DOMINATED PROBLEMS.Journal of Scientific Computing.https://doi.org/10.1023/A:1012873910884

[7] Kevin Schaal *et. al.*.2015.ASTROPHYSICAL HYDRODYNAMICS WITH A HIGH-ORDER DISCONTINUOUS GALERKING SCHEME AND ADAPTIVE MESH REFINEMENT.MRAS.https://doi.org/10.1093/mnras/stv1859

[8] Harold L. Atkins and Chi-Wang Shu1998.QUADRATURE-FREE IMPLEMENTATION OF DISCONTINUOUS GALERKING METHOD FOR HYPERBOLIC EQUATIONS.AIAA Journal.https://doi.org/10.2514/2.436

[9] Yu *et. al.*.2014.ON THE ACCURACY AND EFFICIENCY OF DISCONTINUOUS GALERKIN, SPECTRAL DIFFERENCE AND CORRECTION PROCEDURE VIA RECONSTRUCTION METHODS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2013.11.023

[10] Xiangyi Meng and Yan Xu.2022.ADAPTIVE LOCAL DISCONTINUOUS GALERKIN METHODS WITH SEMI-IMPLICIT TIME DISCRETIZAION FOR THE NAVIER-STOKES EQUATIONS.Advances in Aerodynamics.https://doi.org/10.1186/s42774-022-00110-4

[11] D.Westra.IDENTITIES AND PROPERTIES FOR ASSOCIATED LEGENDRE FUNCTIONS.University of Vienna.https://www.mat.univie.ac.at/~westra/associatedlegendrefunctions.pdf

[12] A.R. DiDonato.1982.RECURRENCE RELATIONS FOR THE INDEFINITE INTEGRALS OF THE ASSOCIATED LEGENDRE FUNCTIONS.Mathematics of Computation.https://doi.org/10.2307/2007289

[13] Gregor J. Gassner *et. al.*.2008.POLYMORPHIC NODAL ELEMENTS AND THEIR APPLICATION IN DISCONTINUOUS GALERKING METHODS. J. Computational Physics.https://doi.org/10.1016/j.jcp.2008.11.012

[14] Abdelkader Baggaget. *al.*.1999.PARALLEL IMPLEMENTATION OF THE DISCONTINUOUS GALERKING METHOD. Parallel Computational Fluid Dynamics Towards Teraflops, Optimization and Novel Formulations (p.115-122)

[15] Gregor J. Gassner and Andre R. Winters.2021.A NOVEL ROBUST STRATEGY FOR DISCONTINUOUS GALERKING METHODS IN COMPUTATIONAL FLUID MECHANICS: WHY? WHEN? WHAT? WHERE?.Front. Phys.https://doi.org/10.3389/fphy.2020.500690

[16] Sebastian Hennemann *et. al.*.2021.A PROVABLY ENTROPY STABLE SUBCELL SHOCK CAPTURING APPROACH FOR HIGH ORDER SPLIT FORM DG FOR THE COMPRESSIBLE EULER EQUATIONS. J. Computational Physics.https://doi.org/10.1016/j.jcp.2020.109935

[17] B.van der Holst and R.Keppens.2007.HYBRID BLOCK-AMR IN CARTESIAN AND CURVILINEAR COORDINATES: MHD APPLICATIONS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2007.05.007

[18] Jianming Liu *et. al.*.2013.ADAPTIVE RUNGE-KUTTA DISCONTINUOUS GALERKING METHOD FOR COMPLEX GEOEMTRY PROBLEMS ON CARTESIAN GRIDS.Int. J. Numer. Meth. Fluids.https://doi.org/10.1002/fld.3825

[19] Jianming Liu *et. al.*.2016.POSITIVITY-PRESERVING RUNGE-KUTTA DISCONTINUOUS GALERKING METHOD ON ADAPTIVE CARTESIAN GRID FOR STRONG MOVING SHOCK.Num. Math. T. M. and A.https://doi.org/10.4208/nmtma.2015.m1416

[20] Axel Rohde.2001.EIGENVALUES AND EIGENVECORS OF THE EULER EQUATIONS IN GENERAL GEOMETRIES. AIAA Computational Fluid Dynamics Conference.https://doi.org/10.2514/6.2001-2609

[21] Xinghui Zhong and Chi-Wang Shu.2013.A SIMPLE WEIGHTED ESSENTIALLY NONOSCILLATORY LIMITER FOR RUNGER-KUTTA DISCONTINUOUS GALERKIN METHOD.J. Computational Physics.https://doi.org/10.1016/j.jcp.2012.08.028

[22] R.Abgrall and F.N.Mojarrad.2022.CONSERVATIVE SCHEME COMPATIBLE WITH SOME OTHER CONSERVATION LAWS: CONSERVATION OF THE LOCAL ANGULAR MOMENTUM. Computers of Fluidshttps://doi.org/10.1016/j.compfluid.2022.105663

[23] M. Dumbser *et. al.*.2015.SOLVING THE RELATIVISTIC MAGNETOHYDRODYNAMICS EQUATION WITH ADER DISCONTINUOUS GALERKIN METHODS, A POSTERIORI SUBCELL LIMITING AND ADAPTIVE MESH REFINEMENT.MNRAS.https://doi.org/10.1093/mnras/stv1510

[24] D.S. Balsar *et. al.*.2013.EFFICIENT IMPLEMENTATION OF ADER SCHEMES FOR EULER AND MAGNETOHYDRODYNAMICAL FLOWS ON STRUCTURED MESHES-COMPARISON WITH RUNGE-KUTTA METHODS .J. Computational Physics.https://doi.org/10.1016/j.jcp.2012.04.051

[25] M. Dumbser *et. al.*.2009.ADER-WENO FINITE VOLUME SCHEMES WITH SPACE-TIME ADAPTIVE MESH REFINEMENT.J. Computational Physics.https://doi.org/10.1016/j.jcp.2008.12.003

[26] D.S. Balsara *et. al.*.2017.HIGH-ORDER ACCURATE SPACE-TIME SCHEMES FOR COMPUTATIONAL ASTROPHYSICS-PART I: FINITE VOLUME METHODS. Living Rev Comput Astrophys.https://doi.org/10.1007/s41115-017-0002-8

[27] B. Cockburn and Chi-Wang Shu.1998.THE RUNGE-KUTTA DISCONTINUOUS GALERKIN METHOD FOR CONSERVATION LAWS V-MULTIDIMENSIONAL SYSTEMS.J. Computational Physics.https://doi.org/10.1006/jcph.1998.5892

[28] Thomas Guillet *et. al.*.2019.HIGH-ORDER MAGNETOHYDRODYNAMICS FOR ASTROPHYSICS WITH AN ADAPTIVE MESH REFINEMENT DISCONTINUOUS GALERKIN SCHEME MRAS.https://doi.org/10.1093/mnras/stz314

[29] Jianxian Qui *et. al.*.2006.A NUMERICAL STUDY OF THE PERFORMANCE OF THE RUNGE-KUTTA DISCONTINUOUS GALERKING METHOD BASED ON DIFFERENT NUMERICAL FLUXES. J. Computational Physics.https://doi.org/10.1016/j.jcp.2005.07.011

[30] B. Després and E. Labourasse.2015.ANGULAR MOMENTUM PRESERVING CELL-CENTERED LAGRANGIAN AND EULERIAN SCHEMES ON ARBITRARY GRIDS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2015.02.032

[31] Michal A. Kopera and Francis X. Giraldo.2014.ANALYSIS OF ADAPTIVE MESH REFINEMENT FOR IMEX DISCONTINUOUS GALERKIN SOLUTION OF THE COMPRESSIBLE EULER EQUATIONS WITH APPLICATIONS TO ATMOSPHERIC SIMULATIONS. J. Computational Physics.https://doi.org/10.1016/j.jcp.2014.06.026

[32] H. Yuan *et. al.*.2017.AN ADAPTIVE MESH REFINEMENT -MULTIPHASE LATTICE BOLTZMANN FLUX SOLVER FOR SIMULATION OF COMPLEX BINARY FLUID FLOWS.Phys.of Fluids.https://doi.org/10.1063/1.5007232

[33] Weiqun Zhang *et. al.*.2021.AMREX: BLOCK-STRUCTURED ADAPTIVE EMSH REFINEMENT FOR MULTIPHYSICS APPLICATIONS.The International Journal of High Performance Computing Applications.https://doi.org/10.1177/10943420211022811

[34] L.Krivodonova *et. al.*.2004.SHOCK DETECTION AND LIMITING WITH DISCONTINUOUS GALERKIN METHODS FOR HYPERBOLIC CONSERVATION LAWS.Applied Numerical Mathematics.https://doi.org/10.1016/j.apnum.2003.11.002

[35] Jun Zhu *et. al.*.2020.HIGH-ORDER RUNGE-KUTTA DISCONTINUOUS GALERKIN METHODS WITH A NEW TYPE OF MULTI-RESOLUTION WENO LIMITERS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2019.109105

[36] M.Dumbser *et. al.*.2014.A POSTERIORI SUBCELL LIMITING OF THE DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR HYPERBOLIC CONSERVATION LAWS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2014.08.009

[37] Olindo Zanotti *et. al.*.2015.SPACE-TIME ADAPTIVE ADER DISCONTINUOUS GALERKIN FINITE ELEMENT SCHEMES WITH A POSTERIORI SUBCELL FINITE VOLUME LIMITNG. Computers and Fluids.https://doi.org/10.1016/j.compfluid.2015.06.020

[38] Jesse Chan *et. al.*.2022.ON THE ENTROPY STABLE DISCONTINUOUS GALERKIN SCHEMES FOR UNDER-RESOLVED FLOWS.Frontiers in Physics.https://doi.org/10.3389/fphy.2022.898028

[39] H.C. Yee *et. al.*.1999.LOW-DISSIPATIVE HIGH-ORDER SHOCK-CAPTURING METHODS USING CHARACTERISTIC-BASED FILTERS.J. Computational Physics.https://doi.org/10.1006/jcph.1998.6177

[40] Bernardo Cockburn and Chi-wang Shu.1991.TVB RUNGE-KUTTA LOCAL PROJECTION DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR CONSERVATION LAWS.Mathematical Modeling and Numerical Analysis.http://www.numdam.org/item/M2AN_1991__25_3_337_0/

[41] Bernardo Cockburn and Chi-wang Shu.1989.TVB RUNGE-KUTTA LOCAL PROJECTION DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD

FOR CONSERVATION LAWS II: GENERAL FRAMEWORK.Mathematics of Computation.https://doi.org/10.2307/2008474

[42] Bernardo Cockburn and Chi-wang Shu.1989.TVB RUNGE-KUTTA LOCAL PROJECTION DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR CONSERVATION LAWS III: ONE-DIMENSIONAL SYSTEMS.J. Computational Physics.https://doi.org/10.1016/0021-9991(89)90183-6

[43] Bernardo Cockburn and Chi-wang Shu.1990.TVB RUNGE-KUTTA LOCAL PROJECTION DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR CONSERVATION LAWS IV: THE MULTIDIMENSIONAL CASE.Mathematics of Computation.https://doi.org/10.2307/2008501

[44] E.F Toro and V.A Titarev.2006.DERIVATIVE RIEMANN SOLVERS FOR SYSTEMS OF CONSERVATION LAWS AND ADER METHODS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2005.06.018

[45] E.F Toro and V.A Titarev.2005.ADER SCHEMES FOR THREE DIMENSIONAL NON-LINEAR HYPERBOLIC SYSTEMS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2004.10.028

[46] S.Busto *et. al.*.2016.DESIGN AND ANALYSIS OF ADER-TYPE SCHEMES FOR MODEL ADVECTION-DIFFUSION-REACTION EQUATIONS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2016.09.043

[47] Arturo Hidalgo and Michael Dumbser.2011.ADER SCHEMES FOR NONLINEAR SYSTEMS OF STIFF ADVECTION-DIFFUSION-REACTION EQUATIONS.Journal of Scientific Computing.https://doi.org/10.1007/s10915-010-9426-6

[48] Michael Dumbser *et. al.*.2017.HIGH-ORDER ADER SCHEMES FOR A UNIFIED FIRST ORDER HYPERBOLIC FORMULATION OF NEWTONIAN CONTINUUM MECHANICS COUPLED WITH ELECTRO-DYNAMICS.J. Computational Physics.https://doi.org/10.1016/j.jcp.2017.07.020

[49] Michael Dumbser *et. al.*.2008.A UNIFIED FRAMEWORK FOR THE CONSTRUCTION OF ONE-STEP FINITE VOLUME AND DISCONINUOUS GALERKIN SCHEMES ON UNSTRUCTURED MESHES.J. Computational Physics.https://doi.org/10.1016/j.jcp.2008.05.025

[50] Michael Dumbser *et. al.*.2018.EFFICIENT IMPLEMENTATION OF ADER DISCONTINUOUS GALERKING SCHEMES FOR SCALABLE HYPERBOLIC PDE ENGINE.Axioms. https://doi.org/10.3390/axioms7030063

[51] F.G. Lether and P.R Wenston.1995.MINMAX APPROXIMATIONS TO THE ZEROS OF Pn AND GAUSS-LEGENDRE QUADRATURE.J. Computational Physics. https://doi.org/10.1016/0377-0427(94)00030-5

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

HIGH-ORDER ADER-DISCONTINUOUS GALERKIN METHOD FOR SOLVING THE COMPRESSIBLE EULER EQUATIONS WITH AMR

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
FABIANO	SASSELLI

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
18.9.2023	Fabiano Sasselli

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.