# Explicit Discontinuous Galerkin Methods for Magnetohydrodynamics

A thesis accepted by the Faculty
of Aerospace Engineering and Geodesy of the University of Stuttgart
in partial fulfillment of the requirements for the degree of
Doctor of Engineering Sciences (Dr.-Ing.)

by

**Christoph Altmann**

born in Nuremberg

| | |
|---|---|
| Main-referee: | Prof. Dr. Claus-Dieter Munz |
| Co-referee: | Prof. James Rossmanith, Ph.D. |
| Date of defence: | 14.09.2012 |

Institute of Aerodynamics and Gas Dynamics
University of Stuttgart

2012

Für meine Eltern und meinen Bruder.

## Preface

This thesis was developed during my work as academic employee at the Institute of Aerodynamics and Gas Dynamics (IAG) of the University of Stuttgart.

I thank my doctoral supervisor Prof. Dr. Claus-Dieter Munz for the excellent working conditions in his research group. Furthermore, I thank Prof. James Rossmanith for being my co-referee.

Many thanks also to all my colleagues at the IAG for the friendly working atmosphere. Moreover, I thank Andrea Beck and Gregor Gassner for their review and their scientific comments.

This thesis was developed within a project financed by the Deutsche Forschungsgemeinschaft (DFG).

Stuttgart, 14.09.2012

Christoph Altmann

# Contents

## Symbols

| | |
|---|---|
| $A$, $B$, $C$ | Flux matrices in x, y, z-direction |
| $a$ | Evolution matrix |
| $a_{ij}$, $b_i$, $b_{ij}^t$ | CERK coefficients |
| $\vec{B}$ | Magnetic field |
| $c$ | Speed of sound |
| $c_A$ | Alfvén wave speed |
| $c_f$ | Fast magneto-acoustic wave speed |
| $c_s$ | Slow magneto-acoustic wave speed |
| $c_h$ | Divergence correction speed |
| $c_p$ | Damping factor |
| $E$ | Total energy |
| $\vec{F}$ | Advection flux matrix |
| $\vec{F}^D$ | Diffusion flux matrix |
| $F_{kl}^r$, $F_{kl}^l$ | Left and right advection flux matrices |
| $K_{kl}$ | Stiffness matrix |
| $\mathbb{L}_2$ | Space of square-integrable functions |
| $M_{kl}$ | Mass matrix |
| $Ma$ | Mach number |
| $N$ | Number of interpolation points, polynomial degree |
| $\vec{n}$ | Normal vector |
| $P$ | Polynomial degree |
| $Pr$ | Prandtl number |
| $p$ | Pressure |
| $Q_i$ | Grid cell |
| $\partial Q_i$ | Grid cell boundary |
| $q^*$ | Normal intermediate velocity component |
| $\mathbb{R}$ | Residual |
| $R_S$ | Entropy residual |
| $r$ | Radius |
| $S$ | Entropy |
| $S_L$, $S_R$ | Left and right signal speeds |

| | |
|---|---|
| $s$ | Surface |
| $T$ | Temperature |
| $t$ | Time |
| $U$ | Exact solution |
| $U_h$ | Numerical solution |
| $u_0 \cdots u_9$ | Polynomial bases |
| $\hat{u}_i(t)$ | Degrees of freedom |
| $u_x$, $u_y$ | x- and y-slopes of the solution |
| $u$, $v$, $w$ | Velocity in x, y and z direction |
| $v$ | Velocity magnitude |
| $W$ | Prediction |
| $\vec{x}$ | Vector of spacial coordinates |
| $\Delta x$, $\Delta y$ | Cell size in x and y-direction |
| $\Delta t$ | Timestep |
| $\epsilon$ | Total energy density |
| $\eta$ | Resistivity |
| $\eta^{\text{dof}}$ | DOF energy indicator value |
| $\eta^{\text{press}}$ | Pressure difference indicator value |
| $\gamma$ | Adiabatic coefficient |
| $\mu$ | Physical viscosity |
| $\nu$ | Artificial viscosity |
| $\Phi$ | Polynomial basis and test function |
| $\psi$ | Divergence cleaning error variable |
| $\rho$ | Density |
| $\tau$ | Viscous stress tensor |
| $\Omega$ | Computational domain |

## Abbreviations

| | |
|---|---|
| ADER | Arbitrary High Order Using Derivatives |
| CERK | Continuous Extension Runge-Kutta |
| CFL | Courant-Friedrichs-Levy |
| CK | Cauchy-Kovalewskaya |
| DG | Discontinuous Galerkin |
| dGRP | Diffusive Generalized Riemann Problem |
| DGSEM | Discontinuous Galerkin Spectral Element Method |
| DNS | Direct Numerical Simulation |
| DOF | Degree(s) of Freedom |
| ENO | Essentially Non-oscillatory |
| FV | Finite Volume |
| GLM | Generalized Lagrange Multiplier |
| HLLC | Harten-Lax-van Leer-Contact Riemann solver |
| HPC | High Performance Computing |
| HWENO | Hermite Weighted Essentially Non-oscillatory |
| LES | Large Eddy Simulation |
| LTS | Local Time Stepping |
| MHD | Magnetohydrodynamic |
| ODE | Ordinary Differential Equation |
| RANS | Reynolds Averaged Navier Stokes |
| RCS | Software Versioning and Revision Control System |
| RK | Runge Kutta |
| RKDG | Runge Kutta Discontinuous Galerkin |
| STE-DG | Space-Time Expansion Discontinuous Galerkin |
| TV | Total Variation |
| TVB | Total Variation Bounded |
| TVD | Total Variation Diminishing |
| WENO | Weighted Essentially Non-oscillatory |

## Kurzfassung

In dieser Arbeit wird die Anwendung des expliziten space-time expansion discontinuous Galerkin (STE-DG) Verfahrens für die idealen und viskosen Gleichungen der Magnetohydrodynamik (MHD) beschrieben. Unter besonderem Hinblick auf Shock-Capturing und Divergenzkorrektur des magnetischen Felds werden Erweiterungen des STE-DG Verfahrens vorgeschlagen, die im Zusammenhang mit MHD-Rechnungen erforderlich werden.

Discontinuous Galerkin Verfahren erfreuen sich einer wachsenden Beliebtheit, da sie eine große Flexibilität im Umgang mit komplexen Geometrien, eine variable Anpassung an das zu berechnende Problem und eine effiziente parallele Implementierung miteinander verbinden. Diese Vorteile machen sie für moderne numerische Berechnungen in vielfältigen Anwendungsgebieten, so auch Astro- und Plasmaphysik, interessant. Das hier vorgestellte STE-DG Verfahren kann dank expliziter lokaler Zeitschritte, bei denen jede Zelle im Rechengebiet ihren eigenen Zeitschritt hat, Berechnungen zusätzlich beschleunigen.

Die dafür benötigte zell-lokale Formulierung birgt zum einen zusätzliche Komplexität für die Implementierung neuer Gleichungssysteme und numerischer Funktionalitäten; zum anderen ermöglicht sie Mechanismen, die für herkömmliche numerische Verfahren ungeeignet erscheinen, da sie sonst den Zeitschritt global zu stark beschränken würden. Das hier vorgestellte Shock-Capturing fällt in diese Kategorie: Künstliche Viskosität wird dazu verwendet, das Stoß-Profil zu glätten und auflösbar zu machen. Die dadurch verursachte starke Zeitschritt-Reduktion wird vom Verfahren größtenteils aufgefangen. Hierfür wurden geeignete Indikatoren entwickelt und evaluiert. Bei der Divergenzkorrektur ermöglichen lokale Zeitschritte eine zusätzliche Effizienzsteigerung. Darüber hinaus werden Postprocessing-Strategien, sowie Ansätze zur Datenreduktion vorgestellt, die gerade bei Verfahren hoher Ordnung von Interesse sind. Außerdem werden Codeentwicklungsstrategien präsentiert.

Für die Validierung des Verfahrens und seiner numerischen Funktionalitäten werden schließlich verschiedene mehrdimensionale Testbeispiele gerechnet, inklusive Konvergenzstudien und Stoßrohr-Rechnungen. Das Verfahren wird dann auf zwei- und dreidimensionale größere und komplexere Testbeispiele der Astrophysik angewandt.

## Abstract

In this work, the explicit space-time expansion discontinuous Galerkin (STE-DG) method is adapted and applied to unsteady ideal and viscous magnetohydrodynamic (MHD) computations. With a special emphasis on shock-capturing and divergence correction of the magnetic field, enhancements to the STE-DG method are proposed that are necessary within the MHD context.

Discontinuous Galerkin schemes enjoy continuously growing popularity, since they combine the flexibility in handling complex geometries, a variable adaptivity to the calculated problem and efficiency of parallel implementations. These are big advantages for modern numerical calculations of various fields of interest, also for MHD calculations e.g. in astrophysics or plasma physics. The presented STE-DG scheme can further enhance explicit computations by its local timestepping functionality, allowing each cell to run with its own determined timestep. The necessary local formulation adds additional constraints to the implementation of new equation systems and numerical ingredients and not every method is suitable. On the other hand it enables mechanisms that would generally be considered to be ineffective for explicit numerical schemes, since they would drastically decrease the timestep of the calculation. The proposed use of artificial viscosity for shock capturing falls in this category: Artificial viscosity is used to capture shocks with a high order scheme. The thereby caused strong influence on the scheme's timestep is substantially reduced by the local timestepping. For this purpose, suitable oscillation indicators were found and evaluated. For the divergence correction of the magnetic field, the local timesteps enable a sub-cycling feature to increase the correction efficiency.

In addition, postprocessing and data reduction techniques are presented, that are especially of interest for high order schemes. Further more, the parallel efficiency of the STE-DG implementation and code development strategies are considered.

To validate the STE-DG implementation for MHD and the proposed ingredients, several multi-dimensional test cases have been set up, including convergence studies and shock tube tests. The scheme is then applied to two- and three-dimensional more complex astrophysical test cases of larger computational scale.

# 1. Introduction

During recent years, the interest in numerical flow simulations has moved towards complex large scale massively parallel computations. Modern computing architectures and configurations are pushing computationally excessive computations like Large Eddy Simulations (LES) or even Direct Numerical Simulations (DNS). In that context, the class of discontinuous Galerkin (DG) high order schemes has gained significantly in popularity since it combines the advantageous features of a high order scheme with an enormous flexibility, making it an ideal candidate for modern large scale parallel calculations. This is a result of the element-local structure of DG schemes. Similar to Finite Element (FE) schemes, the DG approximate solution is defined as a polynomial expansion, where the time dependent polynomial coefficients are the degrees of freedom (DOF). This polynomial approximation is inserted into a variational formulation of the governing equations to determine the DOF. The difference to the classical FE schemes is that the approximation as well as the test functions may be discontinuous at the interface between grid cells similar to the Finite Volume (FV) framework.

The objective of this work is to develop and enhance efficient large scale discontinuous Galerkin (DG) schemes for solving partial differential equations as they occur in flow problems. The equation systems range from purely hyperbolic Euler equations to complex mixed hyperbolic/parabolic equations of Magnetohydrodynamics (MHD). Several critical aspects are covered:

- The treatment of singularities within the hyperbolic equations, namely shocks, as they arise in supersonic flows. This is especially crucial for plasma flows, described by the MHD equations. Here, all problems show strong gradients and therefore require some sort of shock capturing/ limiting technique.

- The numerical handling of the divergence-free condition of the magnetic field for the MHD equations. This is a key issue when trying to calculate MHD flows, since the numerical scheme does not automatically take care of this.

- All ingredients must fit into a high order time explicit local timestepping (LTS) context. This poses severe constraints on the locality, since the LTS algorithm truly narrows the data horizon of the scheme. That is the area of the scheme, where active data is available and can be used for computations.

## 1.1. Shock Capturing

Hyperbolic differential equations like the Euler or MHD equations allow two types of singularities, which are contact discontinuities and shocks. Both phenomena cannot be resolved natively by a numerical scheme. While a contact discontinuity does not sharpen itself after being made resolvable, the shock profile will again begin to steepen, causing the DG polynomial to oscillate once more. Since singularities will produce spurious oscillations [20], the so-called Gibbs phenomenon, the solution will be badly degraded. If these oscillations are allowed to continue to grow, they eventually affect the underlying physics (e.g. positivity of density and pressure) and will cause the computation to fail. Therefore, suitable mechanisms have to be found. For low order schemes like FV, several techniques have been developed. The class of total variation diminishing (TVD) limiters have proven to be very robust. A method is said to be TVD if the total variation (TV), discretely defined as $TV = \sum_j |u_{j+1} - u_j|$, diminishes over time:

$$TV(u^{n+1}) \leq TV(u^n), \tag{1.1}$$

were $n$ denotes the time level. Harten showed in [22] that a TVD scheme is monotonicity preserving, a key feature to handle discontinuities. Well known TVD limiters in the FV context are e.g. the minmod or the superbee limiter of Roe [46]. However, TVD schemes may switch back to first order in the vicinity of an extremum, loosing accuracy even in smooth regions of the solution. High order schemes usually generate oscillations that have to be handled. In general, there is still no golden rule for shock capturing in a high order context. Nevertheless, several ideas are available:

- ENO/WENO reconstruction: By reconstructing high order data out of low order information, one can ensure that the reconstruction process takes care of the oscillatory behavior by selecting the least oscillatory polynomial. In the case of an oscillating discontinuity, all high order information that carry oscillations is removed. A reconstruction would then e.g. not take elements across a discontinuity, but would rather just

use one-sided data. Essentially non oscillatory (ENO) [23] and weighted ENO (WENO) [36] schemes do have oscillation measures that will be used for shock capturing. Unfortunately, although a proper reconstruction on general elements in multi dimensions is feasible, see Dumbser et al. [13], it can be quite cumbersome and computationally very expensive. However, this approach is not possible for a LTS method, since large stencils cannot be built there.

- Moment limitation: For a high order scheme, the degrees of freedom (DOF) can be limited directly, as done by Krivodonova [33]. Her approach can be seen as a generalization of a minmod limiter, sequentially applied to the high order DOF. The limiter generally works with characteristic variables in one space dimension. Extension to two and three space dimensions can only be done on tensor product meshes, general elements may therefore pose difficulties.

- Filtering: Ideas from finite differences, as developed by Yee et al. in [57] or [58] for MHD can also be applied locally within the DG context. Filters modify selective DOF directly. This can also be used to provide some sort of shock capturing by removing or damping the oscillating DOF. The filter framework of Hesthaven [24] can also be modified for shock capturing purposes. Unfortunately, filtering within the DG context has limitations, especially when it comes to strong shocks, as they arise in MHD. The filter eventually reduces the solution to first order, destroying the accuracy of the overall computation.

- Artificial viscosity: This idea originally comes from the finite element community, see e.g. Jaffre et al. [26] or Jameson [27] for FV schemes. Recently, Persson and Perraire [41] revived the theory in the context of DG schemes and proposed a consistent treatment of the artificial viscosity terms (in the volume- and surface integral). The idea is to add viscosity to the discontinuity to be able to directly resolve it with high order. In general, this approach is quite local, especially for their setting, where the oscillation detection process also only needs local data. An application within a LTS scheme is therefore possible.

A shock capturing method consists of two main parts: In the first part, cells that contain oscillating polynomials have to be identified and then be treated in the second part to reduce the oscillations. The first step is crucial for the accuracy of the scheme, as well as the computational efficiency. Shock capturing

takes time and destroys information which is wrong, dangerous or useless in the case of a discontinuity, but may be essential in all other cases. One carefully has develop detection routines to flag just the oscillating cells.

## 1.2. Divergence Correction

Within the Maxwell-part of the MHD equations, the divergence free condition of the magnetic field is not modeled directly. There are different techniques available to ensure that a numerical produces the right results, although the divergence-free condition is not modeled directly into the equation system. They can be distinguished either by their locality or their divergence treatment. Global methods affect the whole computational domain at a time and are usually run in an intermediate step between the field updates. A typical representative of such method is an elliptic projection, based on the Helmholtz decomposition, originally developed by Chorin in 1967 [5]. Here, the solution is divided into a divergence-free part $u_{div}$ and a rotationally invariant part $u_{rot}$:

$$u = u_{div} + u_{rot}, \text{ where } u_{rot} = \nabla\phi, \tag{1.2}$$

with $\phi$ being a scalar potential. Taking the divergence of Equation (1.2) yields

$$\nabla \cdot u = \Delta\phi, \text{ since } \nabla \cdot u_{div} = 0. \tag{1.3}$$

Since the divergence has to be zero, we have to solve $\Delta\phi = 0$ for $\phi$, insert it into (1.2) and solve for $u_{div}$, which then is our projected divergence-free solution. Brackbill and Barnes [4] first developed a projection method in the context of MHD. The general downside may be the high computational costs that come along with the projection. One therefore has to choose the ratio of computation and projection steps carefully. This global correction technique is deferrable in a LTS context. Another way of divergence treatment is the method of constrained transport (CT), developed by Evans and Hawley [14] or Balsara and Spicer [3]. The key idea is to introduce staggered magnetic and electric fields at the cell faces and edges respectively, and to perform the update of the magnetic field in such way that a discrete divergence-free condition is satisfied. However, this technique may be cumbersome on elements other than quads or hexahedra, although CT schemes were developed on general elements as well, see e.g. Torrilhon [52]. In addition, depending on the formulation, too much coupling is introduced so that an LTS scheme can no longer work efficiently. Another quite popular approach is to add a source term, depending on the divergence of the magnetic field to ensure the condition. This method

was developed by Powell et al. [43]. Since this source term makes the MHD system non-conservative, a fix was released by Janhunen [28] in 2000 by adding a source term also to the magnetic field equations. Although this method works well, it cannot handle an "ab initio" divergence error, as it may be the case for a poor initial projection of an originally divergence-free field. The hyperbolic transport method of Dedner et al. from 2002 [12] is capable of handling this. It is equally easy to implement, since it only adds a ninth equation to the MHD system, that can be calculated separately and simply added later. This method is completely local and couples only to direct neighbors via fluxes, just as the MHD system does. It is therefore well suited for a LTS scheme.

## 1.3. High Performance Computing

The computational fluid dynamics community is still one of the main applicants of high performance computing (HPC). This is due to the fact that almost all industrial flow problems are turbulent and are determined by the interaction of phenomena on multiple scales. The state of the art in industry is such that many problems can be solved with acceptable accuracy as a steady solution of the time averaged equations using Reynolds Averaged Navier Stokes (RANS) equations together with well known turbulence models. Nevertheless, the simulation of problems which are inherently unsteady or for which the turbulence models fail or contain a large amount of uncertainty today is still a challenge. Improved numerical methods as well as today's and future large scale massively parallel computer architectures with well adopted numerical algorithms do even make these calculations feasible.

## 1.4. Objectives

The objectives of this thesis are the following:

- Implementation of MHD equations (ideal and viscous)
- Development of robust shock capturing
- Enhancement/Implementation of divergence correction technique for space-time adaptive schemes
- Large scale calculations

## 1.5. Outline

The structure of this thesis is as follows: We begin with the mathematical models, namely the equations and proceed via numerical methods to coding aspects, such as parallelization. Finally, we reach the calculation section and a discussion of future prospects in the end. With Chapter 2, we are starting with an introductory part about the equation systems, namely the Euler, ideal and viscous MHD equations. The following Chapter 3 describes all numerical ingredients needed to handle these equations for complex problems. After some basic considerations, the numerical schemes are described, together with necessary numerical ingredients. Chapter 4 then deals with code specific topics, such as parallelization, optimization and HPC aspects as well as regression checks. After that, Chapter 5 shows various calculation in one, two and three space dimensions for all equation systems, including convergence studies, while Chapter 6 provides summary of recent and especially outlook for future work.

# 2. Equations

## 2.1. Conservation Equations

We are interested in solutions of conservation equations. These equations describe the way information is advected (and diffused). Since this transportation is characterized by variations in space and time, differential equations are used to describe these phenomena. A differential equation has a hyperbolic character, if pure advection is considered. When diffusion comes into play, the equation obtains a parabolic component. While the developed numerical schemes are suitable for several types of these hyperbolic/parabolic differential equations, we want to focus in the following on Euler and MHD equations.

## 2.2. Euler Equations

This set of equations describes inviscid flows. It consists of conservation equations for mas,s momentum and energy and is written in differential form as

$$\begin{array}{lcl} \frac{\partial}{\partial t}\rho & = & -\nabla\cdot\left(\rho\vec{v}\right), \\ \frac{\partial}{\partial t}(\rho\vec{v}) & = & -\nabla\cdot\left(\rho\vec{v}\vec{v}^{t}\right)+\nabla p, \\ \frac{\partial}{\partial t}E & = & -\nabla\cdot\left(\left(E+p\right)\vec{v}\right), \end{array} \tag{2.1}$$

where the superscript $(.)^t$ denotes the transpose of a column vector. Hereby, the pressure $p$, that closes the equation, is derived from $p=(\gamma-1)\left(E-\frac{1}{2}\rho\vec{v}^{2}\right)$, assuming an ideal gas. When expressing these equations in flux-Jacobian form, they read

$$\frac{\partial}{\partial t}U+\nabla\cdot\left(\mathbf{A}(U)\right)=0, \tag{2.2}$$

with $U$ being the vector of the conservation quantities, the state vector, and the tensor $\mathbf{A}\in\mathcal{R}^{d\times m\times m}$ containing the Jacobi matrices $\mathbf{A}_i\in\mathcal{R}^{m\times m}$ for each direction in dimension $d$ of the equation. The eigenvalues of these Jacobian matrices correspond directly to the different system speeds that transport in-

formation. In the case of Euler's equations, there are three of them

$$\begin{aligned} &v - c, \\ &v, \\ &v + c, \end{aligned} \tag{2.3}$$

with $c = \sqrt{\gamma \frac{p}{\rho}}$ being the speed of sound. Since we are examining a hyperbolic system, these eigenvalues are real.

## 2.3. MHD Equations

When the fluid is an electrically conductive plasma, the Euler equations from 2.1 do not describe the full physics, since the electric and magnetic field and their interaction is not modeled. In general, a plasma, although neutral in total, contains species of different charges. All gases will change to a plasma at high temperatures due to ionization. As long as a continuum assumption is valid, a plasma can be described by the equations of Magnetohydrodynamics. They are developed by considering the conservation of mass, momentum and energy just as done for the Euler equations and taking into account Maxwell's equations that describe the electric and magnetic field. By neglecting resistivity and viscous effects, we derive the ideal MHD equations in their differential form to

$$\begin{aligned} \tfrac{\partial}{\partial t}\rho &= -\nabla \cdot (\rho \vec{v}), \\ \tfrac{\partial}{\partial t}(\rho \vec{v}) &= -\nabla \cdot \left( \rho \vec{v}\vec{v}^t - \vec{B}\vec{B}^t + \left( p + \tfrac{1}{2}|\vec{B}|^2 \right) I \right), \\ \tfrac{\partial}{\partial t} E &= -\nabla \cdot \left( (E + p)\, \vec{v} + \left( \tfrac{1}{2}|\vec{B}|^2 I - \vec{B}\vec{B}^t \right) \cdot \vec{v} \right), \\ \tfrac{\partial}{\partial t} \vec{B} &= -\nabla \cdot \left( \vec{B}\vec{v}^t - \vec{v}\vec{B}^t \right). \end{aligned} \tag{2.4}$$

Again, a pressure $p$ is needed to close the system. This pressure, called total pressure, consists of both a hydrodynamical and a magnetic part. Analogous to the Euler equations, we assume an ideal plasma and derive $p = (\gamma - 1)\left( E - \frac{1}{2}\rho|\vec{v}^2| - \vec{B}^2 \right)$. Please note that for simplicity reasons, the magnetic permeability constant $\mu_0$ is set to 1. Similar to Euler's equations before, the MHD equations can also be brought into flux-Jacobian form and the eigenvalues of the Jacobi matrix can be determined. This reveals its seven wave

speeds, since we are dealing with seven conserved variables. In detail, we get

$$\begin{aligned} &v - c_f,\ v - c_A,\ v - c_s, \\ &v, \\ &v + c_f,\ v + c_A,\ v + c_s, \end{aligned} \tag{2.5}$$

with $c_f, c_s$ being the fast and slow magneto-acoustic wave speed and $c_A$ the Alfvén wave speed with

$$\begin{aligned} c_A &= \sqrt{\tfrac{B_x}{\rho}}, \\ c_f &= \sqrt{\tfrac{1}{2}\left(a^2 + b^2 + \sqrt{(a^2+b^2)^2 - 4a^2c_A}\right)}, \\ c_s &= \sqrt{\tfrac{1}{2}\left(a^2 + b^2 - \sqrt{(a^2+b^2)^2 - 4a^2c_A}\right)}, \end{aligned} \tag{2.6}$$

where $a = \sqrt{\frac{\gamma p}{\rho}}$ and $b^2 = \frac{\mathbf{B}^2}{\rho}$. The latter, $c_A$, is quite special, because, unlike the other waves, oscillations in density and longitudinal velocity cannot be observed [51]. Nevertheless, it is a very characteristic phenomenon for plasmas and the magnetic field is its moving power. The flow is bending magnetic stream lines that will on their part create flow again. Unfortunately, one important aspect of the magnetic field is not modeled directly within the equation system, but necessary for physically correct results: its solenoidal character.

### 2.3.1. Divergence constraint

Physically, the magnetic field is required to be divergence free at all time. Gauss' law for the magnetic field as part of Maxwell's equations states

$$\nabla \cdot \vec{B} = 0. \tag{2.7}$$

Since the solution of the MHD equations is already unique due to the evolution equation for the magnetic field in 2.4, the divergence free condition would result in an overdetermined system. Because its evolution is implicitly fulfilled by the rest of Maxwell's equations, since $\nabla \cdot rot(B) = 0$, it acts as a constraint here. While physically fulfilled at any time, we will see later that we have to take care of this constraint numerically. Please note that although the divergence is of elliptical character, its region of influence is local. The finite propagation speed of the MHD system is not modified. A close look at the system reveals that the system degenerates to the Euler equations when the magnetic field is set to zero.

## 2.4. Viscous MHD

When viscosity and resistivity are taken into account, the viscous MHD equation system is derived. With added viscosity and resistivity, the former purely hyperbolic character of the equations changes to a mixed hyperbolic parabolic type. As chapter 3 reveals, this requires some modifications of the numerical scheme. The complete system in differential form is written as

$$\begin{aligned}
\tfrac{\partial}{\partial t}\rho &= -\nabla\cdot(\rho\vec{v})\,, \\
\tfrac{\partial}{\partial t}(\rho\vec{v}) &= -\nabla\cdot\left(\rho\vec{v}\vec{v}^{t} - \vec{B}\vec{B}^{t} + \left(p + \tfrac{1}{2}|\vec{B}|^2\right) I - \tau\right), \\
\tfrac{\partial}{\partial t}E &= -\nabla\cdot\left((E+p)\,\vec{v} + \left(\tfrac{1}{2}|\vec{B}|^2 I - \vec{B}\vec{B}^{t}\right)\cdot\vec{v},\right. \\
&\quad \left. -\vec{v}\tau + \eta\left(\vec{B}\cdot\nabla\vec{B} - \nabla\left(\tfrac{1}{2}|\vec{B}|^2\right)\right) - \mu\tfrac{1}{Pr}\nabla T\right), \\
\tfrac{\partial}{\partial t}\vec{B} &= -\nabla\times\left(\vec{B}\times\vec{v} + \eta\nabla\times\vec{B}\right),
\end{aligned} \tag{2.8}$$

similar to [54],where $\tau := \mu(\vec{\nabla}\vec{v} + (\vec{\nabla}\vec{v})^T - \frac{2}{3}(\vec{\nabla}\cdot\vec{v})\ I)$ denotes the viscous stress tensor and $Pr = \frac{c_p\mu}{\kappa}$ the Prandtl number. The pressure $p$ remains unchanged from the ideal MHD equations. Again, we have the solenoidal property of the magnetic field, $\nabla\cdot\vec{B} = 0$, as an additional constraint. Please also note the similarity to the compressible Navier-Stokes equations when assuming zero magnetic field.

# 3. Numerics

We will now focus on DG schemes and their implementation for the simulation of fluid flows, defined by the equation systems that were introduced in the previous chapter. Several characteristics that come with DG schemes will also be addressed.

## 3.1. Conventions

Throughout this work, several operators and notations are used frequently. This first section is used to introduce these mathematical basics, conventions and notations. They will help to clarify mathematical expressions as well as simplify more complex connections.
The scalar product in $\mathbb{L}_2$ is defined as

$$f, g \in \mathbb{L}_2(C) : \langle f, g \rangle_{\mathbb{L}_2(C)} = \int_C f(x)g(x)dx. \tag{3.1}$$

The $\mathbb{L}_2$ norm is defined as

$$f \in \mathbb{L}_2(C) : ||f||_{\mathbb{L}_2(C)} = \sqrt{\langle f, f \rangle_{\mathbb{L}_2(C)}}. \tag{3.2}$$

With the help of the nabla operator, defined as

$$\vec{\nabla} = \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_d} \end{pmatrix}, \tag{3.3}$$

the differential operators "grad" and "div" are then defined as

$$\begin{aligned} \text{div}(\vec{a}V) = \vec{\nabla} \cdot \vec{a}V, \text{ with } \vec{\nabla} \cdot \vec{a}V[j] := \sum_{i=1}^{d} \frac{\partial}{\partial x_i} \vec{a}[i]\, V[j],\ j = 1, ..., m, \\ \text{grad}(V) = \vec{\nabla}V, \text{ with } \vec{\nabla}V[i, j] := \frac{\partial}{\partial x_i} V[j],\ i = 1, ..., d,\ j = 1, ..., m. \end{aligned} \tag{3.4}$$

Here, $\vec{a} \in \mathbb{R}^d$ denotes a vector in physical space, containing $d$ entries as the number of space dimensions, while $V \in \mathbb{R}^m$ is a vector with the number of state variables, $m$, as entries.

## 3.2. Numerical Schemes

We now move on to numerical schemes for the solution of nonlinear partial differential equations, like ideal or viscous MHD. An explicit modal Discontinuous Galerkin method with nodal integration and a time consistent local time-stepping enhancement is described in the following. We will start with the Discontinuous Galerkin formulation basics.

### 3.2.1. Space Discretization Basics

A key point for efficient numerical schemes is the transformation into reference space. We therefore need to search for a transformation from physical space to a so-called reference space, where each element or the whole computational domain is regularized. For each type of element, only a single reference element has to be build. Still, the transformation will vary. Doing computations in reference space has two main advantages: First, several computations in reference space only have to be done once in the initialization phase of the computation and can then be stored for later use. This saves a high amount computational work. Second, computations, e.g. integrations, are done on a regularized element which will avoid round-off problems that may occur at small elements. Depending on the element shape, the transformation will be nonlinear, e.g. when curved elements are transformed.

### 3.2.2. Modal Discontinuous Galerkin Basics

We will now move on to focus on Discontinuous Galerkin schemes and their formulations. In the following, a modal DG, as e.g. described by [19], with nodal integration, is presented.
Without the loss of generality, let us consider a system of nonlinear partial differential equations (PDEs) of the form

$$U_t + \nabla \cdot \left( \vec{F}(U) - \nu(\vec{x})(\vec{\nabla} U) \right) = 0, \tag{3.5}$$

where $U$ denotes the vector of conservative variables and $\vec{F}(U)$ the advection flux matrix. The gradient of $U$ in our case represents an artificial viscosity term

that is added for shock capturing purposes, while $\nu$ is the amount of element-wise added viscosity. The above PDE system could be the representation of any nonlinear equation system like the Euler equations or MHD equations, both extended by an artificial viscosity term.
We will next introduce the DG framework by defining an approximate (numerical) solution $U_h = U_h(\vec{x}, t)$. We then choose a finite element (FE) like ansatz, where the appropriate solution in an arbitrary space-time cell $Q_i^n := Q_i \times [t_n, t_{n+1}]$ is given by

$$U\Big|_{Q_i^n} \approx U_h(\vec{x}, t) := \sum_{j=1}^{N} \hat{U}_j^{n,i}(t)\Phi_j^i(\vec{x}) \text{ for all } x \in Q_i. \tag{3.6}$$

The $\hat{U}_j^{n,i}$ are the time dependent degrees of freedom (DOF) and the basis functions
$\{\Phi_j(x)\}_{j=1,\dots,N}$ span the space of polynomials with degree $\leq P$ over the spatial grid cell $U_i$, with $N$ being the number of interpolation points. This ansatz will then be inserted into the weak formulation and tested with each of the test functions $\Phi_j$ that are chosen to be from the same space as our basis functions. To keep things simple in the following, we will omit the $h$ from the numerical solution $U_h$ within $Q_i$ as long as no misunderstanding can occur. We will also suppress the index $i$ in the notation of $U^i$ and $\Phi^i$.
To derive the weak formulation, we first multiply the equation system (3.5) by a test function $\Phi = \Phi(\vec{x})$, integrate over an arbitrary space-time cell and perform a spatial integration by parts:

$$\int_{Q_i^n} U_t \Phi \, d\vec{x}dt + \int_{\partial Q_i^n} \left(\vec{F}(U) - \nu\vec{\nabla}U\right)\cdot\vec{n}\Phi \, dsdt - \int_{Q_i^n} \left(\vec{F}(U) - \nu\vec{\nabla}U\right)\cdot\vec{\nabla}\Phi \, d\vec{x}dt = 0. \tag{3.7}$$

For the artificial viscosity part of our equation (3.7) which contains second order derivatives, we proceed with a second integration by parts:

$$\int_{Q_i^n} \nu\vec{\nabla}U\cdot\vec{\nabla}\Phi \, d\vec{x}dt = \int_{Q_i^n} \vec{\nabla}U\cdot\nu\vec{\nabla}\Phi \, d\vec{x}dt = \int_{\partial Q_i^n} U\nu\vec{\nabla}\Phi\cdot\vec{n}dsdt - \int_{Q_i^n} U\nabla\cdot\left(\nu\vec{\nabla}\Phi\right) \, d\vec{x}dt. \tag{3.8}$$

The integration by parts is used twice, back and forth, under the assumption that the volume integral is calculated from data inside $Q_i$ and only the surface integral covers the interaction between the grid cells. Data from the elements interior is marked with a subscript $INT$, which stands for "internal". The

objective is to lift a jump between the functional values from the interior and the outer interface state at the grid cell boundary into the discrete variational formulation, see [18] in addition.
We then end up with the weak formulation

$$\int_{Q_i^n} U_t \Phi \, d\vec{x}dt + \int_{\partial Q_i^n} \left( \vec{F}(U) - \nu \vec{\nabla} U \right) \cdot \vec{n} \Phi \, dsdt + \int_{\partial Q_i^n} \left( \nu U \vec{\nabla} \Phi - \left[ \nu U \vec{\nabla} \Phi \right]_{INT} \right) \cdot \vec{n} \, dsdt - \int_{Q_i^n} \left( \vec{F}(U) - \nu \vec{\nabla} U \right) \cdot \vec{\nabla} \Phi \, d\vec{x}dt = 0. \tag{3.9}$$

The identity approach allows us to combine the volume integral portion of both the advective volume integral and the volume integral part obtained by the treatment of the diffusive terms for the numerical implementation. With the use of orthonormal basis functions which are constructed via the Gram-Schmidt orthogonalization algorithm, this discretization yields a diagonal mass matrix, even for elements with curved boundaries. Because our approximation $U_h$ is discontinuous across element interfaces, we have to introduce numerical flux functions to guarantee both the stability and consistency of the discretization.

### 3.2.3. Time Discretization Basics

Having derived a variational formulation, the next thing to do is updating in time. We now have to solve an ordinary differential equation (ODE), where the unknowns are the DG degrees of freedom. A usual method for a high order DG scheme is using a Runge-Kutta (RK) one-step scheme. For DG schemes, these were developed by Cockburn and Shu (RKDG), see their article series [6–10]. Explicit methods need to obey a time step restriction to remain stable. The timestep reads as

$$\begin{aligned} \Delta t_{adv} &= \frac{CFL(P)}{\lambda_{adv}^{max}} \left( \frac{\Delta x}{2P+1} \right) \text{ for advection,} \\ \Delta t_{dif} &= \frac{DFL(P)}{\lambda_{dif}^{max}} \left( \frac{\Delta x}{2P+1} \right)^2 \text{ for diffusion,} \end{aligned} \tag{3.10}$$

where $\lambda_{adv}^{max}$ and $\lambda_{adv}^{dif}$ denote the maximum eigenvalues of the advection and diffusion flux matrices. For advection-diffusion systems, the timestep is deter-

mined by superposition, resulting in

$$\Delta t_{advdif} = \frac{1}{\sqrt{\frac{1}{\Delta t_{adv}^2} + \frac{1}{\Delta t_{dif}^2}}}. \tag{3.11}$$

RK methods can further be built in an implicit or hybrid context, the latter being called IMEX Runge-Kutta schemes, as e.g. shown in [29]. They are especially useful when different terms in the DG formulation trigger different time scales. In the context of the MHD equations with divergence cleaning, a sub-cycling of the cleaning mechanism could then be treated implicitly, since it would otherwise significantly reduce the timestep. If running explicitly, the timestep to take is determined by finding its minimum among all cells of the computational domain. This can reduce computational efficiency if the timestep varies significantly within the domain. Think of e.g. very small stretched cells (so-called slivers) that result from a local h-adaptation. These cells will now reduce timestep of all cells and therefore of the whole computation. We will now describe a different method that does not need global timesteps, but instead uses local timesteps, depending on the local timestep restriction of each cell.

### 3.2.4. STE and Local Time Stepping

Equation (3.9) contains space-time integrals which have to be approximated by an appropriate quadrature rule, e.g. Gaussian quadrature. For its evaluation, solution values at intermediate time levels are needed. We will use the strong form of Equation (3.5) and build a local predictor for the intermediate solution that does not depend on neighboring information. The correction is then performed by taking into account the predictions from neighboring cells. To obtain the strong form of Equation (3.5), we start from Equation (3.7) and again perform a backward integration by parts. As long as only data of the element $Q_i$ is involved, this is an equivalence transformation. We therefore explicitly do not take neighboring information into account. Data from the elements interior is again marked with a subscript $INT$, which stands for "internal". We now get

$$\begin{aligned} &\int_{Q_i^n} \left(U_t + \nabla \cdot \vec{F}(U) - \nu(\vec{\nabla} U)\right) \Phi \, d\vec{x}dt + \\ &\int_{\partial Q_i^n} \left(\left(\vec{F}(U) - \nu\vec{\nabla}U\right) - \left(\vec{F}(U_{INT}) - \nu\vec{\nabla}U_{INT}\right)\right) \cdot \vec{n}\Phi \, dsdt = 0. \end{aligned} \tag{3.12}$$

In a local time stepping setting, a predictor will not be able to take neighboring data into account. We therefore solve the so-called local Cauchy problem within our update-cell and neglect all neighboring data. Instead, we extent our DG polynomial that holds as initial condition for the Cauchy problem onto $\mathbb{R}^n$. The local Cauchy problem then reads as

$$\int_{Q_i^n} W_t \Phi \, d\vec{x}dt = \int_{Q_i^n} \left( \nabla \cdot \vec{F}(W) - \nu(\vec{\nabla} W) \right) \Phi \, d\vec{x}dt, \tag{3.13}$$

with $W_{Q_i^n}(t=0) = U_{Q_i^n}(t=0)$, the DG polynomial as initial condition. After using appropriate integration rules, we can now solve this ODE in time with standard schemes, providing the scheme is able to offer a time series of the solution of the Cauchy problem. We need that series to evaluate the integrals using Gaussian quadrature. Two approaches are shown in the following. Once the solution is obtained, we move back to the strong form, now inserting the predictors of our cell and its neighbors into the surface integral. We end up with our predictor-corrector formulation being derived as

$$\begin{aligned} &\int_{Q_i^n} \left( U_t + \nabla \cdot \vec{F}(W) - \nu(\vec{\nabla} W) \right) \Phi \, d\vec{x}dt + \\ &\int_{\partial Q_i^n} \left( \left( \vec{F}(W) - \nu \vec{\nabla} W \right) - \left( \vec{F}(W_{INT}) - \nu \vec{\nabla} W_{INT} \right) \right) \cdot \vec{n} \Phi \, dsdt = 0. \end{aligned} \tag{3.14}$$

We can see that the correction is done via the surface integral, using local predictions of neighboring cells.
The first approach to build a time series of the solution of Equation (3.13) is based on a Taylor expansion in space and time

$$U(\vec{x}, t) = U(\vec{x}_i, t_n) + \sum_{j=1}^{N} \frac{1}{j!} \left( (t - t_n) \frac{\partial}{\partial t} + (\vec{x} - \vec{x}_i) \cdot \vec{\nabla} \right)^j U \Big|_{(\vec{x},t)=(\vec{x}_i,t_n)}, \tag{3.15}$$

about the barycenter $\vec{x}_i$ of the element $U_i$ at time $t_n$, with $N$ being the polynomial degree. The space-time Taylor expansion provides approximate values for $U$ and $\vec{\nabla} U$ at all space-time points $(\vec{x}, t) \in Q_i^n$, if the values of both the pure space and time derivatives as well as the mixed space-time derivatives at the point $(\vec{x}_i, t_n)$ of the expansion are known. While the pure space derivatives at $(\vec{x}_i, t_n)$ are readily available within the DG framework, the time derivatives and

mixed space-time derivatives have to be computed using the so-called Cauchy-Kovalevskaya (CK) procedure. It allows to compute all space-time derivatives starting from pure space derivatives. A construction guideline for this procedure is shown in Appendix A.
The second approach is using a continuous extension Runge-Kutta (CERK) scheme, introduced by [39, 40], to advance the Cauchy problem in time. This special Runge-Kutta scheme provides a time series solution and reads as

$$\hat{W}^{n+1} = \hat{U}^n + \Delta t \sum_{i=1}^{n_{stages}} b_i \hat{K}_i,$$

where

$$\hat{K}_i = \mathbb{R}(\hat{W}_i^n), \tag{3.16}$$

with

$$\hat{W}_i^n = \hat{U}^n + \sum_{j=1}^{n_{stages}} a_{ij} \hat{K}_j.$$

We then get a time series for our predictive solution

$$\hat{W}(t) = \sum_{i=1}^{p_t+1} c_i^t t_j,$$

where (3.17)

$$c_i^t = \sum_{j=1}^{n_{stages}} b_{ij}^t \hat{K}_j.$$

Here, $n_{stages}$ and the CERK coefficients $a_{ij}$, $b_i$ and $b_{ij}^t$ depend on the desired time order.
Other methods are possible as well, especially Continuous and Discontinuous Galerkin methods, as described in [16], giving the framework excellent adaptation abilities for various numerical problems. This space-time setting now allows us to perform a high order time accurate and fully conservative explicit local time stepping, where each cell within the computational domain runs with its own optimal time step. Via the outlined Taylor expansion, necessary values at the cell boundaries can be reconstructed in time. It is therefore ensured that no information that has to be exchanged with the neighbors will be lost during the time updates. In the following, the methodology of the local time stepping is briefly outlined.

A sequence of four time steps of three adjacent grid cells in Figure 3.1, taken from [37], starting from a common time level $t^0 = 0$, shows the work flow:

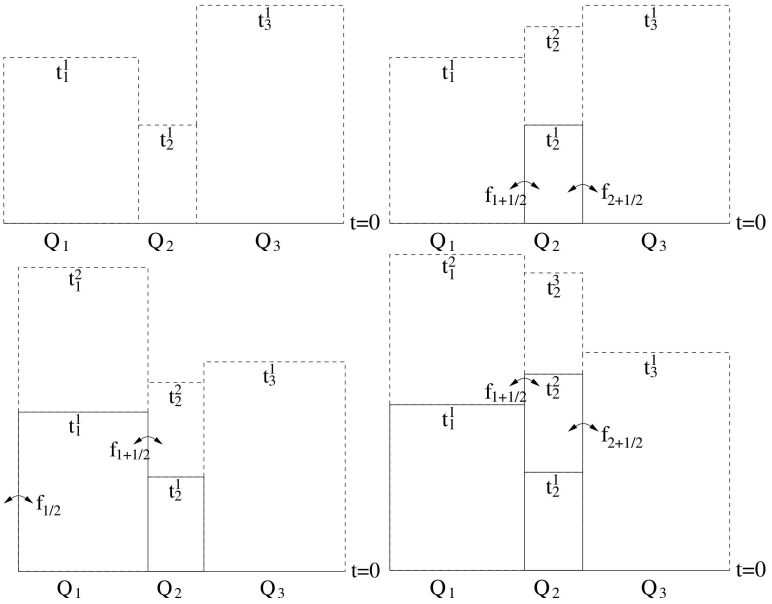



**Figure 3.1.:** Sequence of steps 1-4 of a computation with 3 different elements and local time stepping

After the determination of the local time steps, which are assumed to be different in our example due to the local stability restriction, the space-time Taylor expansions are calculated in each grid cell. This results in a predictive approximate solution in the space-time cells $Q_i \times [t_i^0, t_i^1]$ - in our example for $i = 1, 2, 3$. These space-time polynomials are stored. We note that after this step the DOF $\vec{\hat{U}}_i^0$ at the time level $t_i^0$ are not needed any longer and may be overwritten. First, the volume integrals are calculated for each element $Q_i$. They rely only on the local space-time polynomials. The contribution of these terms are added to the DOF of the old time level. We call these values $\vec{\hat{U}}_i^*$. In the end, we have for each grid element a space-time polynomial already containing the

volume integral contributions. Next, the surface flux contributions involving neighboring grid cells have to be considered.
The local time stepping algorithm relies on the following evolve condition. The update of the DOF can only be completed, if

$$t_i^{n+1} \leq \min\left\{t_j^{n+1}\right\}, \forall j : Q_j \cap Q_i \neq \emptyset \tag{3.18}$$

is satisfied. This condition guarantees that all the data for the interface fluxes are available. In our example, the first grid cell satisfying this condition is $Q_2$. So $Q_2$ can now be evolved to $t_2^1$. To do so, the flux contributions at the right and left cell interface have to be computed and its contribution is then added to the local $\vec{\hat{U}}_2^*$. The flux integrals are calculated from $t = t_2^0$ to $t = t_2^1$ at the right interface $\partial Q_{2+\frac{1}{2}}$ and the left interface $\partial Q_{2-\frac{1}{2}}$. The arguments for the numerical flux functions at the time Gaussian points are obtained from the left and right space-time polynomials.
In order to keep this calculation exactly conservative as well as efficient, the flux contributions computed for the evolution of $Q_2$ are added simultaneously to the corresponding neighbors $\vec{\hat{U}}_1^*$ and $\vec{\hat{U}}_3^*$ with the minus sign. Then the update for $Q_2$ is completed and the DOF at the new time level $t_2^1$ are fully derived. We can now start the procedure again: A new space-time polynomial is constructed in $Q_2 \times [t_2^1, t_2^2]$ and the volume integral contribution is added to the local DOF $\vec{\hat{U}}_2^1$, now named by $\vec{\hat{U}}_2^*$.
If boundary values at the left are given, now $Q_1$ satisfies the evolve condition and can be advanced to $t_1^1$. As before, the volume integral contribution is already added. But in this case, also a part of the flux contributions has already been added to the $\vec{\hat{U}}_1^*$ during the previous evolution of $Q_2$. Thus, only the missing flux contributions, which are sketched in the lower left corner of Figure 3.1, have to be added to the $\vec{\hat{U}}_1^*$ in order to get the $\vec{\hat{U}}_1^1$. Namely, on the interface $\partial Q_{1+\frac{1}{2}}$, the flux integral has to be computed with a quadrature formula from $t_2^1$ to $t_1^1$. As before, the flux integral computed on this shared interface is not only added to $\vec{\hat{U}}_1^*$, but also to $\vec{\hat{U}}_2^*$. The time interval, for which the flux contribution at the interface shared by an element $Q_i$ and an adjacent element $Q_j$ has to be computed when evolving $\vec{\hat{U}}_i^*$ to $\vec{\hat{U}}_i^{n+1}$, is generally

$$[t_{ij}^\star, t_i^{n+1}] = [max(t_i^n, t_j^n), t_i^{n+1}]. \tag{3.19}$$

In this manner, the algorithm continues by searching for elements satisfying the evolve condition (3.18). At each time, the interface fluxes are defined

uniquely for both adjacent elements, making the scheme exactly conservative. Since small elements usually perform many timesteps until the evolve condition cannot be fulfilled any more, the algorithm checks the current element again, before continuing with other elements. This speeds up the algorithm and can be advantageous in terms of cache reuse.
The presented local time stepping algorithm minimizes the total number of time steps for a computation with fixed end time. However, in some cases where the difference of time levels of adjacent grid cells are very small compared to the local time steps, the efficiency of the presented algorithm may decrease. To overcome this deficiency, we locally synchronize the time levels of those cells. It is also not difficult to introduce some common global time levels as needed for example at the end of the computation. Please note that this procedure has absolutely no influence on the accuracy of the underlying numerical scheme, as convergence tests e.g. in [37] and [18] verify.

## 3.3. Numerical Ingredients

### 3.3.1. Numerical Fluxes

For the approximation of the fluxes between cell edges, the rotational invariance of the MHD equations is exploited. We can therefore rotate the state vector $U$ and the gradient of the state vector $\vec{\nabla}U$ from the global $(x, y)$-system into an edge aligned $(\widetilde{x}, \widetilde{y})$-system.
We will first have a look at the inviscid fluxes. Here, a wide variety of exact and approximate Riemann solvers are available and would all be valid choices. Since finite volume schemes strongly depend on the effectiveness and efficiency of Riemann solvers to handle the jumps at cell edges, a strong development in this area began as these schemes became popular. Riemann solvers for all kind of equation systems with different special features were developed. The book of Toro [49] is a good start for further reading. However, we are interested in high order DG schemes. These schemes tend to diminish jumps at element interfaces, the higher the polynomial order is chosen. After all, with the exception of shocks, a fully resolved flow field should be nearly continuous. Therefore, the importance of the Riemann solver within the numerical framework decreases with the scheme's increasing order. This was e.g. discovered previously in [45] and [44]. We are therefore looking for a simple, robust and computationally cheap Riemann solver that still captures the physics well.
In our case, inspired by the good performance in the hydrodynamical case, we have chosen the HLLC flux adaption for multidimensional MHD equations as

proposed by Li, described in [35],

$$F_{HLLC} = \begin{cases} F_l, & \text{if } 0 \leq S_L, \\ F_l + S_L\left(\vec{U}_l^* - \vec{U}_r\right), & \text{if } S_L \leq q^*, \\ F_r + S_R\left(\vec{U}_r^* - \vec{U}_r\right), & \text{if } q^* \leq S_R, \\ F_r, & \text{if } 0 \geq S_R. \end{cases} \tag{3.20}$$

Here, the signal speeds $S_L$ and $S_R$, that approximate left- and right-going characteristics, and the normal intermediate velocity component $q^*$ are derived as

$$\begin{aligned} S_L &= u + c_f, \\ S_R &= u - c_f, \\ q^* &= \tfrac{\rho_r u_r (S_R - u_r) - \rho_l u_l (S_L - u_l) + p_l - p_r - B_{x_l}^2 + B_{x_r}^2}{\rho_r (S_R - u_r) - \rho_l (S_L - u_l)}, \end{aligned} \tag{3.21}$$

and the middle state, denoted with $U_{l/r}^*$ as

$$\begin{aligned} \rho_{l/r}^* &= \rho \tfrac{S_{L/R} - u}{S_{L/R} - u^*}, \\ (\rho u)_{l/r}^* &= \rho_{l/r}^* u^*, \\ (\rho v)_{l/r}^* &= (\rho v)\, \tfrac{S_{L/R} - u}{S_{L/R} - u^*} - \tfrac{\left(B_{x_{l/r}}^* B_{y_{l/r}}^* - B_{x_{l/r}} B_{y_{l/r}}\right)}{S_{L/R} - u^*}, \\ (\rho w)_{l/r}^* &= (\rho w)\, \tfrac{S_{L/R} - u}{S_{L/R} - u^*} - \tfrac{\left(B_{x_{l/r}}^* B_{z_{l/r}}^* - B_{x_{l/r}} B_{z_{l/r}}\right)}{S_{L/R} - u^*}, \\ B_{x_{l/r}}^* &= B_x^{HLL} = \tfrac{S_R B_{x_r} - S_L B_{x_l}}{S_R - S_L}, \\ B_{y_{l/r}}^* &= \frac{\left(S_{L/R} - u_{l/r} - \frac{B_{x_{l/r}}^* B_{x_{l/r}}}{\rho\left(S_{L/R} - u_{l/r}\right)}\right) B_{y_{l/r}} - \left(B_{x_{l/r}}^* - B_{x_{l/r}}\right) v_{l/r}}{S_{L/R} - u^* - \frac{\left(B_{x_{l/r}}^*\right)^2}{\rho\left(S_{L/R} - u_{l/r}\right)}}, \\ B_{z_{l/r}}^* &= \frac{\left(S_{L/R} - u_{l/r} - \frac{B_{x_{l/r}}^* B_{x_{l/r}}}{\rho\left(S_{L/R} - u_{l/r}\right)}\right) B_{z_{l/r}} - \left(B_{x_{l/r}}^* - B_{x_{l/r}}\right) w_{l/r}}{S_{L/R} - u^* - \frac{\left(B_{x_{l/r}}^*\right)^2}{\rho\left(S_{L/R} - u_{l/r}\right)}}, \\ E_{l/r}^* &= E_{l/r} \tfrac{S_{L/R} - u}{S_{L/R} - u^*} + \tfrac{\left(p^* u^* - p u\right) - \left(B_{x_{l/r}}^* \left(\mathbf{B}^{HLL} \cdot \mathbf{u}^{HLL}\right) - B_{x_{l/r}} (\mathbf{B} \cdot \mathbf{u})\right)}{S_{L/R} - u^*}, \end{aligned} \tag{3.22}$$

where

$$u^* = \frac{\rho_r u_r \left(S_R - u_r\right) - \rho_l u_l \left(S_L - u_l\right) + p_l - p_r}{\rho_r \left(S_R - u_r\right) - \rho_l \left(S_L - u_l\right)}, \tag{3.23}$$

is the normal component of the velocity of the middle state $*$, just as in the case of the hydrodynamical HLLC Riemann solver.
For approximating the diffusion fluxes, a method developed by Gassner *et al.* [17] is used. This method is based on the solution of the generalized Riemann problem for systems of linear diffusion equations and is called the dGRP-flux (diffusive Generalized Riemann Problem). Hereby, the character of the second order differential equation is taken into account by using at least piecewise linear initial data for solving the Riemann problem. Since the character of the diffusion fluxes of the MHD equations, either with physical (viscous MHD) or artificial diffusion, directly correspond to its hydrodynamical equivalence, the theory behind the dGRP holds also in the case of MHD. The diffusive part of the viscous MHD as described by Karniadakis [54] is similar to the Navier-Stokes equations by means of its homogeneity with respect to the gradient

$$\vec{F}^D\left(U, \vec{\nabla}U\right) = \mathbf{D}(U)\vec{\nabla}U, \tag{3.24}$$

where the diffusion matrix $\mathbf{D}(U)$ contains $d^2$ diffusion matrices of full rank, containing eigenvalues greater or equal to zero. Those positive semi-definite matrices will guarantee physically meaningful results. Since the electric resistivity $\eta$ of the viscous MHD equations is the magnetic equivalent of the viscosity coefficient $\mu$ for the hydrodynamic part, we can adopt the calculation of the diffusion fluxes from [17] with only little modifications. In this context, we can also describe the influence of these additional magnetic diffusion terms on the parabolic time step restriction as it is done for the Navier-Stokes equations. The time step restriction is thus simply built with the maximum of viscosity and resistivity.

### 3.3.2. Shock Capturing

Since shocks are part of many MHD problems, a suitable shock capturing mechanism has to be found, as it is well known that a high order DG scheme suffers from spurious oscillations at shock waves or strong gradients. Especially for MHD, shocks can be very strong, so special care of this matter has to be taken. In the context of the proposed local time stepping scheme, we have to face a major difficulty: Because neighboring cells may be at different time levels in regard to the cell that has to be limited, most common and often used reconstruction methods like ENO [23] or WENO [36] schemes are almost impossible to implement, since the necessary information to build a stencil at a common time level from a set of neighboring cells is not available. Setting

all ENO/WENO stencil cells to the same time level would destroy the local time stepping functionality: A stencil cells would first need to be brought to the time level of the reconstruction-cell. Within the LTS scheme, its time level could be above or below the time level of the oscillating cell that needs to be reconstructed. This implies that we would need to either store a large amount of old time levels or update stencil cells first. Additionally, since cells that are part of a stencil could be oscillating cells themselves, the reconstruction procedure would get an iterative and completely inefficient character. If we would like to save this functionality, we would need to store the polynomials at older time levels since we probably need them within a stencil. This would lead to a totally inefficient or even impossible method, both computationally and storage-wise.
As a key to efficient shock capturing for an explicit local time stepping scheme, we therefore need a limiting mechanism that is as local as possible. In other words: A method that, in the best case, only needs data of the oscillating element itself.
Persson and Peraire [41] developed such a strategy by adding a certain amount of artificial viscosity to the numerical scheme determined by a sensor that needs only local information. In that way we get a method that is completely element-local. Initially, their mechanism was designed only for steady-state problems. As the scheme presented in this thesis is especially designed to handle unsteady problems, we found out that it works remarkably well in that context, too. This is remarkable, since the shock usually does not remain in a single position in unsteady calculations. However, we found out that their oscillation indicator responds very fast to changes in the DG polynomial. Together with the effect of the viscosity, these features allow a robust unsteady usage. Since the viscous timestep dominates at shocks, when using viscosity for shock capturing, a global timestepping scheme would drastically loose efficiency. So far, this methodology was only used in implicit schemes. For the first time, the local timestepping mechanism allows this shock capturing technique to be efficiently used in an explicit scheme as well. The artificial viscosity strategy of Persson and Peraire simply adds Laplacian viscosity to all state equations. We will outline this method in the following.

#### 3.3.2.1. Oscillation Indicators

The first step is to detect grid cells that carry spurious oscillations in the approximate polynomial. This task is done via an oscillation indicator, whose output is then directly used to calculate the suitable amount of artificial vis-

cosity that has to be added in order to smooth the profile. There are several possibilities available on how to build such a sensor, but we will focus on two variants in the following.

#### 3.3.2.2. DOF Energy Indicator

We set up the following indicator that measures cell-wise the maximum energy decay in the highest and second highest degrees of freedom of our DG polynomial for any primitive or conservative state variable $u$:

$$\eta^{\mathrm{dof}}(u) = \log_{10}\left\{\max\left[\left(\frac{\sum_{j=N(P-1)+1}^{N(P)}(\tilde{u}_j)^2}{\sum_{j=1}^{N(P)}(\tilde{u}_j)^2}\right),\left(\frac{\sum_{j=N(P-2)+1}^{N(P-1)}(\tilde{u}_j)^2}{\sum_{j=1}^{N(P-1)}(\tilde{u})_j)^2}\right)\right]\right\}, \tag{3.25}$$

where $N(P)$ denotes the number of degrees of freedom for a DG polynomial of degree $P$ and $\tilde{u}_j$ the $j^{th}$ degree of freedom of our polynomial representation of $u$. This setup respects the influence of odd/even order effects in the DG polynomial. Since we want to distinguish between shocks and contact discontinuities, the pressure could be the preferable indicator for shock capturing, since it jumps at shocks but not at contact discontinuities. Too much diffusion at a contact discontinuity can therefore be avoided. Nevertheless, also a contact discontinuity has to be handled properly, otherwise causing oscillations as well. A small amount of artificial viscosity, using the density as an indicator, may also help in this context. Since a contact discontinuity will not sharpen itself, no further viscosity is added once the numerical scheme can fully resolve its profile. While this indicator is used to determine the necessary artificial viscosity, other variables, such as the density, can be applied for tasks like adaptive mesh refinement.

After calculating the indicator, the artificial viscosity is then determined by

$$\nu = \nu\left(\eta^*, h, P\right), \tag{3.26}$$

where $\eta$ is the indicator value, e.g. $\eta^{\mathrm{res}}$, $h$ is the spatial resolution and $P$ is the polynomial order.

In detail, as done in [41], we use a smooth distribution of the form

$$\nu\left(\eta\left(u\right)\right) = \begin{cases} \varepsilon_0 & \eta \geq \eta_{max} \\ \frac{\varepsilon_0}{2}\left(1+\sin\left(\frac{\pi\left(\eta-\frac{1}{2}\left(\eta_{max}+\eta_{min}\right)\right)}{\eta_{max}-\eta_{min}}\right)\right) & \text{if} \quad \eta_{min} \leq \eta \leq \eta_{max} \\ 0 & \eta_{min} \geq \eta \end{cases}, \tag{3.27}$$

where $\epsilon_0$ is a gain factor defined by the user. This factor may be problem dependent, especially for different spatial resolutions.
If several shocks of different strength have to be captured, it may be better to just focus on the derivatives in our DG solution polynomial. We therefore replace the mean value $\widetilde{u}_1$ in (3.25) by an artificial one. Different mean values, as they arise when dealing with different shocks at different states, could otherwise result in a non-optimal viscosity distribution when using just one general gain factor.
The following example will demonstrate the capabilities of the above defined sensor: A Mach 10 shock is moving through a computational domain. In addition to a $5^{th}$ order run with the STE-DG scheme, we performed the calculation also with a $4^{th}$ order WENO reconstructed finite volume scheme, see [53] and a $3^{rd}$ order DG scheme with HWENO limiting as presented in [2]. The shock profiles in density together with the grid sizes after $t = 0.16$ are plotted in Figure 3.2.
The $3^{rd}$ order DG scheme and the $4^{th}$ order reconstructed (rec.) FV scheme both ran on 100 cells, while for the $\mathcal{O}5$ STE-DG scheme only 25 cells were used. One notices that all profiles look rather similar. As expected, the HWENO-DG and the rec. FV scheme both smeared the shock over about 4 cells which is common for these limiting strategies and schemes. In contrast, the artificial viscosity limiter was able to capture the profile within one cell with almost the same resolution, but using only 1/4 of the HWENO-DG/rec. FV cells.

#### 3.3.2.3. Pressure Difference Indicator

The second approach that is presented here uses a technique initially introduced by Jameson [27] for finite volume schemes, but also commonly used within the finite difference community. He builds a second order viscosity to damp oscillations. We use his ideas as a starting point to build a shock indicator from a second order pressure derivative. The so-called pressure indicator reads as

$$\eta_i^{\text{press}}\left(p\right) = \frac{\left|p_{\min} - 2p_i + p_{\max}\right|}{\left|p_{\min} + 2p_i + p_{\max}\right|}. \tag{3.28}$$

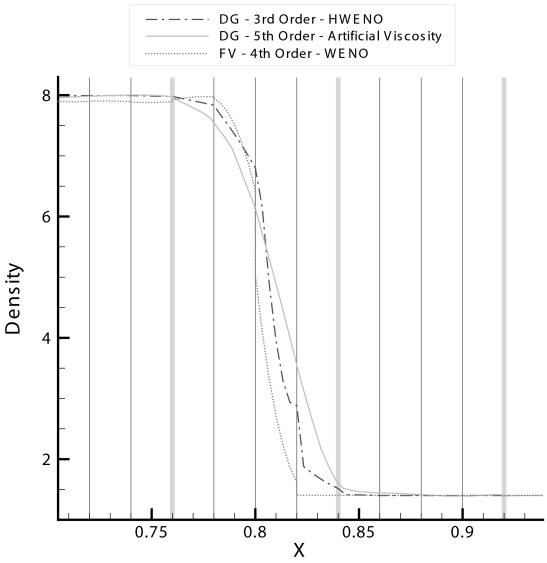



**Figure 3.2.:** Comparison of shock capturing properties. Plot of a $3^{rd}$ order DG scheme, a $4^{th}$ order WENO reconstructed FV scheme and the $5^{th}$ order STE-DG scheme with artificial viscosity

It estimates the change of the pressure from the von Neumann neighbors, where $p_{\min}$ is set as the minimum and $p_{\max}$ as the maximum mean value of the pressure of all neighboring cells.
To build this viscosity, the indicator value $\eta_i^{\text{press}}$ is multiplied with factors of geometry, flow field and a user determined amplification factor $D0$. We then end up with

$$\nu\left(\eta\left(u\right)\right) = D0\ \lambda_{max}\ h\ \eta. \tag{3.29}$$

In a local time stepping setting, one would use the zeroth DoF of the neighboring cells' prediction that is available at the element boundary. This indicator introduces some sort of coupling, making it more robust but also less accurate when comparing to the local resolution indicator.

#### 3.3.2.4. Entropy Residual Indicator

The last approach was introduced by Guermond and Pasquetti in [21]. It uses an entropy residual $R_S$ to determine oscillating cells as well as the correct amount of a point-wise viscosity distribution needed for shock capturing.

$$\begin{aligned} R_S &= S_t + \nabla \cdot (\vec{v}S)\,, \\ \nu &= \alpha \rho h^2 \left|R_S\right|, \end{aligned} \tag{3.30}$$

with $S$ being the entropy of the equation system, derived from $S = \frac{\rho}{\gamma - 1} ln\left(\frac{p}{\rho^\gamma}\right)$ and $\vec{v}$ being the velocity vector. To limit the maximum amount of viscosity introduced by the system, an upper bound is established as

$$\nu_{max} = \alpha_{max}\ h\ max\left(|\vec{v}| + c\right). \tag{3.31}$$

The parameters $\alpha$ and $\alpha_{max}$ are user defined to adjust the bounds in order to set the right amount of viscosity.

#### 3.3.2.5. Indicator Comparison

We will now compare all the three indicators for two different test problems. This includes a qualitative comparison, defined by the shock sharpness, resolution and remaining oscillations, as well as some quantitative numbers concerning calculation times. The first calculation is a Mach 10 shock in a tube. The post-shock state was determined via the Rankine-Hygoniot conditions. The pre-shock state was user-given, the density was set to 1. All three indicators where properly adjusted to discover their near optimal settings. Figure 3.3 shows the outcome of this calculation. The vertical lines indicate cell boundaries. This way we can determine the sub-cell shock resolution for the different indicators. We can see that all three indicators perform well overall and are quite similar. There is almost no difference for the resolution indicator (Persson) and pressure indicator (Jameson), while the entropy residual indicator (Pasquetti) is little behind.
The second calculation is the famous one-dimensional interaction problem by Osher and Shu from [48]. This testcase is used to show the advantages of high order schemes. A $M = 3.0$ shock travels through a sinusoidal wave. Its initial condition is given by

$$(\rho, u, p) = \begin{cases} \begin{pmatrix} 3.8571, & 2.6294, & 10.333 \end{pmatrix} & \text{if } x < -4.0, \\ \begin{pmatrix} 1 + 0.2\sin(5x), & 0.0, & 1.0 \end{pmatrix} & \text{if } x \geq -4.0. \end{cases} \tag{3.32}$$

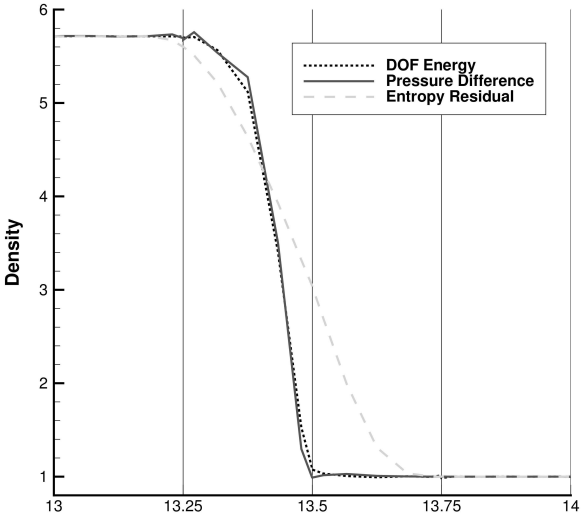



**Figure 3.3.:** Indicator comparison for the shock problem.

For this calculation, the methods show stronger differences than within the first example, see Figure 3.4. We can see that shock capturing done with the pressure indicator gives the least accurate results, while being the most robust. This is due to the fact that we are only using a second order viscosity. We are therefore losing accuracy especially in regions where high order information would significantly contribute to the viscosity distribution calculation. Figure 3.4 shows the state at time $t = 1.8$.

To get an idea of the performance of all three methods, the local timestepping feature will be used: By calculating a performance factor, defined as the average timestep divided by the minimum timestep of a computation, $\frac{\Delta t_{avg}}{\Delta t_{min}}$, one could easily measure the speed-up of the local timestepping. The factor directly tells how much faster the computation runs compared to a global timestepping scheme. By comparing this number with the computational time for the calculation, the amount and/or spread of artificial viscosity can be determined. A high speed-up factor combined with a short calculation time is an indicator

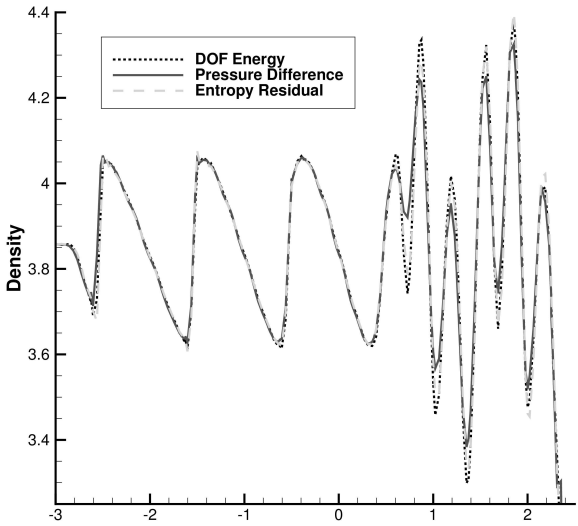



**Figure 3.4.:** Indicator comparison for the Osher-Shu problem.

for a very local viscosity region which is preferable for shock capturing.

- Osher-Shu LTS performance factor (avg timestep/min timestep)
  - Resolution indicator (Persson): 6.3
  - Pressure indicator (Jameson): 3.95
  - Entropy residual indicator (Pasquetti): 2.67
- Osher-Shu calculation time factor (averaged)
  - Resolution indicator (Persson): 1.0
  - Pressure indicator (Jameson): 1.1
  - Entropy residual indicator (Pasquetti): 1.8

We see that the resolution indicator in the spirit of Persson does perform best for the Osher-Shu problem. The pressure indicator comes close, but does not allow a similar speedup, while still maintaining a good calculation time. When

keeping in mind that this indicator gave the worst results, as shown in Figure 3.4, this leads to the assumption that it uses less viscosity as the resolution indicator (worse speed-up while still good calculation time) but spreads viscosity too much (again, worse speed-up and worse results). The entropy residual indicator, however, does need the most computational time and therefore triggers too much viscosity in general. This also explains the bad speed-up factor.

#### 3.3.2.6. Parameter Studies

Since the pressure difference indicator from Section 3.3.2.3 does only have a single adjustment parameter, it is well suited for more detailed parameter tests. These tests should help to identify hidden dependencies of the indicator setting as well as find parameter-free settings that automatically adjust the necessary viscosity depending on the local cell values. It is therefore necessary to discover parameter dependencies between polynomial order, spacial resolution, shock strength and shock alignment as well as CFL/DFL number or synchronization intervals of local timestepping mechanism. All tests were performed by varying the user determined amplification factor $D0$ with a bisection algorithm to determine the minimum value for a successful calculation. The following test matrices will show varied parameter and the resulting $D0$ factors. Tables 3.1 and 3.2 show the resulting amplification factors when varying different parameters. Table 3.1 helps to identify dependencies on the polynomial order of the scheme and the strength of the shock (characterized by the shock Mach number $Ma_s$). It is obvious that the shock strength is perfectly covered by the indicator formula, so no dependencies occur. There are slight variations for different polynomial order, though. Some differences occur when using first order polynomials, resulting in a second order accurate scheme.

Table 3.2 shows dependencies on the polynomial order of the scheme and the number of elements used for discretization. Again, we can only see small variations here, indicating that the indicator formulation works indeed well and robust for different problems.

By plotting the amplification factors over the shock strength indicator Ms for the different polynomial degrees, as done in Figure 3.5, one realizes that all curves look similar. A trial-and-error approach gave a factor of $\frac{Ms}{7}$, which is also shown. This factor can then be used in the indicator formulation 3.29.

| $Ma_s$ \ $N$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 2.0 | 0.0005 | 0.7132 | 1.0859 | 1.2967 | 1.5937 | 2.5544 | 3.0053 |
| 6.0 | 0.1576 | 0.6376 | 0.7304 | 1.1241 | 0.6985 | 1.2476 | 1.5290 |
| 10.0 | 0.1370 | 0.6016 | 0.6075 | 0.7282 | 0.4782 | 0.7065 | 0.8634 |
| 14.0 | 0.1328 | 0.4111 | 0.4081 | 0.5146 | 0.3385 | 0.5012 | 0.6929 |
| 18.0 | 0.1381 | 0.3307 | 0.4275 | 0.4162 | 0.2660 | 0.3833 | 0.4838 |
| 22.0 | 0.1290 | 0.3666 | 0.3563 | 0.3140 | 0.2667 | 0.3060 | 0.3997 |
| 26.0 | 0.1441 | 0.3952 | 0.3085 | 0.3157 | 0.2128 | 0.2549 | 0.3274 |
| 30.0 | 0.1000 | 0.4099 | 0.2470 | 0.2528 | 0.1865 | 0.2145 | 0.2803 |
| 34.0 | 0.1080 | 0.1791 | 0.2050 | 0.2268 | 0.1761 | 0.1881 | 0.2688 |

**Table 3.1.:** Smallest necessary amplification factors for a variation of polynomial degree and shock strength for the pressure difference indicator.

| $N$ \ #elems | (20,2) | (40,2) | (60,2) | (80,2) | (100,2) |
|---|---|---|---|---|---|
| 2 | 0.50 | 0.50 | 0.59 | 0.50 | 0.49 |
| 3 | 0.54 | 0.54 | 0.55 | 0.54 | 0.59 |
| 4 | 0.73 | 0.73 | 0.73 | 0.72 | 0.73 |
| 5 | 0.49 | 0.66 | 0.47 | 0.46 | 0.66 |

**Table 3.2.:** Smallest necessary amplification factors for a variation of element number and polynomial degree for the pressure difference indicator.

### 3.3.3. MHD Divergence Cleaning

When dealing with MHD, the divergence free ($\nabla \cdot \mathbf{B} = 0$) constraint has to be maintained. Not doing so, numerical schemes may generate divergence errors that can have a negative influence on the solution and may eventually lead to an abort of the calculation. While some methods will clean divergence errors to machine precision, others are designed to keep errors bounded and prevent their growth during the computation. Since our scheme is running with a local time stepping mechanism, all divergence cleaning methods that are based on operations affecting all cells at the same time (e.g. projection methods) cannot

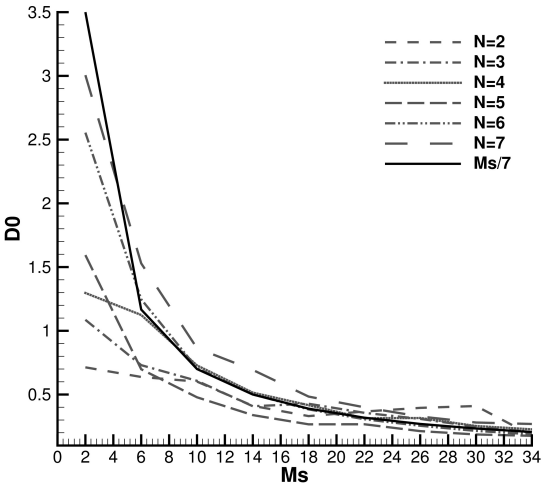



**Figure 3.5.:** Amplification factors over shock strength Ms for different polynomial degrees. A reference fitting of Ms/7 is also drawn.

be applied. As in the case of shock capturing, we are again looking for a very local method.

Dedner et al. [12] presented a hyperbolic divergence cleaning that is easy to implement, can be applied very locally and still yields the desired effect on the divergence errors. It adds a divergence correction variable to the MHD equation system that can be solved in a very fast and straight-forward way. Since the effect of this variable on the whole system is similar to a Lagrangian multiplier, they called this method the Generalized Lagrange Multiplier (GLM) divergence correction method. With that addition, the viscous MHD equations

as shown in [55] with GLM modification result to

$$\begin{aligned}
\frac{\partial}{\partial t}\rho &= -\nabla\cdot(\rho\vec{v}) \\
\frac{\partial}{\partial t}(\rho\vec{v}) &= -\nabla\cdot\left(\rho\vec{v}\vec{v}^t - \vec{B}\vec{B}^t + \left(p+\tfrac{1}{2}|\vec{B}|^2\right)I - \tau\right) \\
\frac{\partial}{\partial t}E &= -\nabla\cdot\left((E+p)\,\vec{v} + \left(\tfrac{1}{2}|\vec{B}|^2 I - \vec{B}\vec{B}^t\right)\cdot\vec{v}\right. \\
&\quad \left. -\vec{v}\tau \;+\; \eta\left(\vec{B}\cdot\nabla\vec{B} - \nabla\left(\tfrac{1}{2}|\vec{B}|^2\right)\right) - \mu\tfrac{1}{Pr}\nabla T\right) \\
\frac{\partial}{\partial t}\vec{B} &= -\nabla\times\left(\vec{B}\times\vec{v} + \eta\nabla\times\vec{B} + \psi I\right) \\
\frac{\partial}{\partial t}\psi &= -\nabla\cdot\left(c_h^2\vec{B}\right) - \tfrac{c_h^2}{c_p^2}\psi,
\end{aligned} \tag{3.33}$$

with the divergence constraint $\nabla\cdot\vec{B}=0$. Again, $\tau := \mu(\vec{\nabla}\vec{v}+(\vec{\nabla}\vec{v})^T - \frac{2}{3}(\vec{\nabla}\cdot\vec{v})\,I)$ is the viscous stress tensor and $Pr = \frac{c_p\mu}{\gamma}$ the Prandtl number. The pressure (perfect gas) is derived as $p = \rho RT = (\gamma-1)\left(E - \frac{1}{2}\rho\vec{v}^2 - \frac{1}{2}\vec{B}^2\right)$. As has been said before, setting viscosity $\mu$ and resistivity $\eta$ to zero results in the ideal MHD system. The additional variable $\psi$ is introduced and propagates the divergence error out of the computational domain. With the addition of $-\frac{c_h^2}{c_p^2}\psi$ to the evolution equation for $\psi$ on the right hand side of (3.33), we do not only transport the errors out of the computational domain but also damp them. The damping effect can be scaled by setting the value of $c_p$. The original paper suggests a value of 0.18 for $c_p$, which is a good compromise between damping and hyperbolic transport. If more damping is forced, the hyperbolic removal of divergence errors backs out. With that modification, the divergence cleaning method is then called mixed GLM method. For MHD equations, it was proposed in [12] to set the hyperbolic transport speed $c_h$ to the fastest system wave, so that errors are at least spread with the same velocity as they may be generated. Furthermore, it is ensured that the correction subsystem is not affecting the time step of the overlying MHD calculation.
Treating the divergence correction system within a split scheme we get the following stand-alone problem:

$$\begin{aligned}
\vec{B}_t + \quad \nabla\psi \quad &= 0, \\
\psi_t + \quad c_h^2\,\nabla\cdot\vec{B} \quad &= 0.
\end{aligned} \tag{3.34}$$

Dedner et al. showed in [12] a simple way for the calculation of the flux of this linear system. Since the STE-DG scheme relies on local time steps, each cell allows for a divergence cleaning at a different speed. We therefore need to adapt the original setting of the GLM system based on global time steps.

This is done by adjusting the calculation of the numerical flux of the correction system. By taking into account the corresponding local Riemann problem, the numerical flux then reads in one space dimension as:

$$\begin{pmatrix} B_{x,m} \\ \psi_m \end{pmatrix} = \begin{pmatrix} B_{x,l} \\ \psi_l \end{pmatrix} + \begin{pmatrix} \frac{1}{2}(B_{x,r} - B_{x,l}) - \frac{1}{2c_h}(\psi_r - \psi_l) \\ \frac{1}{2}(\psi_r - \psi_l) - \frac{c_h}{2}(B_{x,r} - B_{x,l}) \end{pmatrix} . \quad (3.35)$$

Here, $B_x$ denotes the x component of the magnetic field vector $\mathbf{B}$. The correction speed $c_h$ in the above formula will be cell local. Consider two adjacent elements $Q_i$ and $Q_j$ with different correction speeds $c_{h,i}$ and $c_{h,j}$. The flux out of element $Q_i$ into element $Q_j$ will then be calculated with the speed $c_{h,i}$, the flux from element $Q_j$ to $Q_i$ with $c_{h,j}$. A similar approach within a different environment was already shown in [31]. Please keep in mind that for vanishing divergence errors, our proposed DG scheme is exactly conservative at all time. This flux can be added directly to the numerical flux of the MHD equations. As this system is invariant to rotations it can easily be transformed into the normal direction via a simple vector-matrix multiplication and extended to multi dimensions. Since we are solving one-dimensional Riemann problems when calculating inter-cell fluxes from one element's side to its neighboring side, Equation (3.35) is all we need even for a multi-dimensional scheme.
By increasing the divergence error transportation speed $c_h$, one will be able to remove errors even faster. Due to the increased number of time steps we then have to perform for the cleaning, this method has as a sub-cycling behavior: For each physical time step, we perform several divergence cleaning steps. When using this method, a global time step scheme would become computationally inefficient. However, by making use of local time steps, these limitations can be minimized. The local time stepping hereby ensures a computationally efficient operation.

### 3.3.4. Divergence Test Case

In order to test different divergence cleaning aspects and strategies, a special testcase was set up. Consisting of a Gaussian peak in the x-magnetic field, it has a non-zero divergence in the beginning. The cleaning mechanism then starts to reduce the divergence errors. We can determine the cleaning efficiency by taking the cleaning rate (divergence reduction over time) and the obtained minimum divergence error into account. Equation 3.36 shows the initial condition of the problem

$$\rho = 1.0; \quad e = 6.0; \quad B_x = exp\left(-0.5\left(\frac{\vec{x}}{0.11}\right)^2\right), \quad (3.36)$$

all other conservative variables are set to zero. We are therefore starting with an initial divergence error. Since no divergence sources are added, we see the error approaching zero. The testcase is calculated on a quadratic computational domain, spanning from $-1$ to 1 in each direction. By adding a non-zero velocity to the initial condition, the peak can be advected through the domain.

### 3.3.5. Correction Speed

It seems clear that strong divergence errors should be treated with high correction speeds while only small errors do not require going beyond the largest system speed of the MHD system. We are therefore directly making use of the divergence error itself by calculating the $\mathbb{L}_2$ norm of the cell divergence and correlating it to the corresponding correction speed. The divergence itself is derived via lifting

$$\nabla \cdot \vec{B} \quad = \int_Q (\nabla \cdot \vec{B}) \; \Phi \; d\vec{x} = \int_{\partial Q} \; \vec{B} \cdot \vec{n} \; \Phi \; ds - \int_Q \; \vec{B} \cdot \nabla \Phi \; d\vec{x}, \tag{3.37}$$

An important question is the correlation between the indicator value and the correction speed, which determines the sub-cycling strength. Whenever the correction speed jumps heavily from cell to cell, spurious reflections are generated. These reflections can prevent the divergence error from falling below a certain, user defined threshold. Several approaches have been tested, using more distinct steps, linear or smooth transitions or a direct correlation. However, the less steep steps occur during the computation, the better results were obtained.
Nevertheless, by applying two different speed steps for the correction, the setting is most efficient:

$$c_h = \begin{cases} c_{min} & & \| \nabla \cdot \vec{B} \|_{L_2} \leq 5.0e-3 \\ \frac{c_{max}}{2} & \text{for} \quad 5.0e-3 \leq & \| \nabla \cdot \vec{B} \|_{L_2} \leq 5.0e-2 \\ c_{max} & & \| \nabla \cdot \vec{B} \|_{L_2} \geq 5.0e-2 \end{cases}, \tag{3.38}$$

where $c_{min}$ denotes the maximum physical speed within the cell, determined by the CFL condition and $c_{max}$ a user defined maximum correction speed, $c_{max} \geq c_{min}$. The introduction of speed steps is balancing areas of slightly varying divergence errors, building more consistent sub-cycling zones. A direct mapping would otherwise introduce to many steps that generate oscillations, so the cleaning process takes longer then.
For analyzing efficiency, the divergence $\mathbb{L}_2$-norm of the test problem with initially non-zero divergence, described above, is plotted over the CPU time for

different correction settings in Figure 3.6. Thus, the first task of the scheme is to remove the ab initio divergence errors. Runs were performed by setting the divergence correction speed to be equal to the maximum CFL determined system speed, to be seven times this speed and to be in between these both speeds by using the setting (3.38) described above. It is obvious that the divergence correction with variable speed outperforms both other settings. This is due to the local time stepping functionality acting on the different correction speed steps.

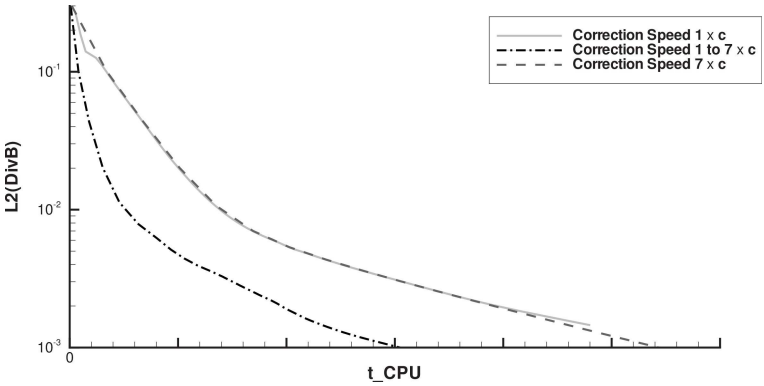



**Figure 3.6.:** Efficiency of the mixed GLM divergence correction with different correction speed settings.

Figure 3.6 shows the divergence error over computational time. The steeper the gradient of the curve, the faster the scheme removes divergence errors. We can see that both correction schemes with global speed perform equally efficient. Although the correction speed is higher for one calculation, the timestep is smaller and we therefore end exactly on the same efficiency level. However, the variable calculation benefits from the local timestepping setting, since only cells with strong divergence errors do need a high correction speed. The LTS mechanism ensures that the scheme stays efficient, since only the timestep in the high-correction-speed-cells is reduced. Once strong divergence errors are corrected, the variable method reduces to a global correction with $c_{min}$ which explains the alignment of all curves for higher computational times.

### 3.3.6. Influence of Artificial Viscosity on Divergence Errors

Several performed calculations showed remarkably good results even in the case where the hyperbolic divergence cleaning was switched off. This discovery may seem strange at first, because DG schemes are not in any case superior to other numerical schemes when it comes to handling divergence errors. In contrast, high order schemes are even said to be inferior here, since their oscillations at shocks would even worsen the problem. Nevertheless, the shock treatment of our scheme is the answer to this behavior. Since jumps or oscillations in the (perpendicular) magnetic field over MHD shocks are smoothed out by artificial viscosity, the production rate of divergence errors, which is related to these magnetic field discontinuities, is reduced. However, Section 5.1 shows that the design order of convergence as well as the accuracy of the scheme is not met any more. The computation remains stable but becomes less accurate. The following two pictures of the Orszag-Tang Vortex sow exactly this behavior. In Figure 3.7, we plotted the density distribution of runs with and without divergence cleaning at time $t = 0.5$. Also stated is the $\mathbb{L}_2$ norm of the divergence error at that time. We can see small changes in the solution, mainly oscillations, as well as differences in the $\mathbb{L}_2$ error norm of several magnitudes. Nevertheless, both computations remained stable up to the simulation end time of $t = 0.5$.

### 3.3.7. Low Order Divergence Cleaning for High Order Schemes

Since high order scheme do have less numerical dissipation than low order schemes, divergence errors are advected over longer distances before damed out by numerical dissipation. This is especially important, if no additional damping is added to the correction system. It is therefore obvious to test, if a low order divergence correction is more efficient than a correction with the same order of accuracy as the main numerical method, simply by adding more dissipation to damp the errors. By setting the correspondent DOF to zero before the time update is done, a low order correction can be tested easily. In Figure 3.8 below, a correction with different orders of accuracy for a initially non-zero divergence error is shown. All diagrams show the $\mathbb{L}_2$ divergence error over time.

It is clearly visible that a low order correction will not make divergence correction more efficient. Quite the contrary, a very low order correction makes results worse in the beginning. When going to higher order, results do not change that much any more. Still, using the same order for the correction sys-

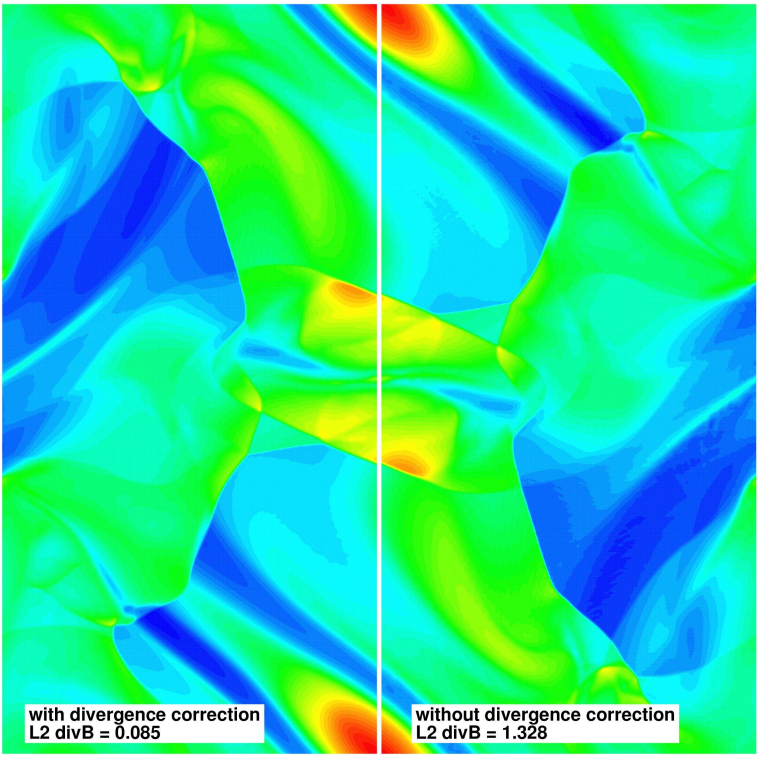



**Figure 3.7.:** Density plot of the Orszag-Tang vortex without and with divergence correction. $\mathbb{L}_2$ divergence error norm is also shown.

tem as well as the MHD equations is the native way that should be preferred.

### 3.3.8. Treatment of Boundaries

The scheme's hyperbolic transport mechanism propagates the divergence errors out of the computational domain. Classical supersonic outflow boundaries can directly be adopted to the additional correction system, while all

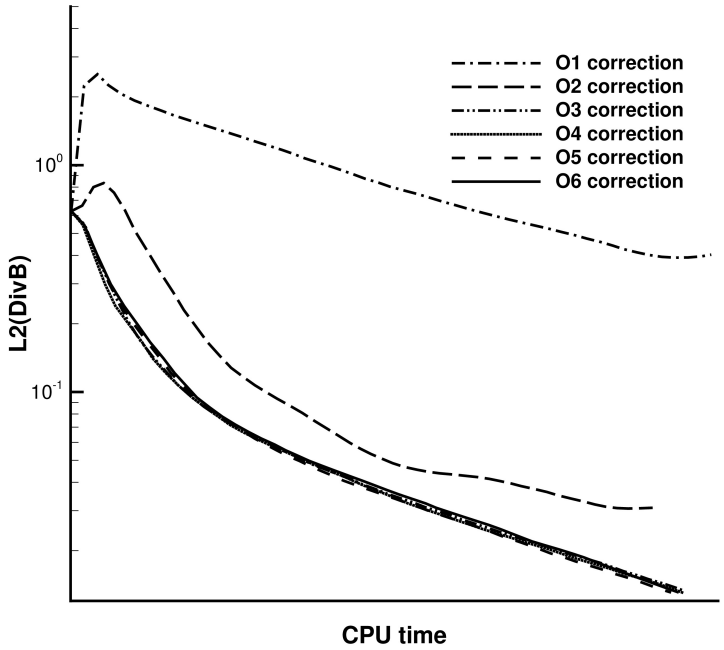



**Figure 3.8.:** Plot of $\mathbb{L}_2$ divergence error over time for different orders of accuracy of the correction system.

other boundaries do require some sort of treatment. One could implement a "semipermeable" boundary that always let the divergence errors pass through. Unfortunately, such setting may lead to inconsistencies within the state variables: Since the GLM system is coupled the the MHD system, divergence errors do also produce error in the remaining quantities. By just letting the divergence errors themselves pass through the boundary, additional errors are generated since remaining errors in other variables are still advected against the boundary. Setting the damping parameter to larger values directly at the boundary can be used to stabilize the system. This way errors that remain within the computational domain will be damped. The above described test

case is once again used to demonstrate the desired behavior. Figure 3.9 shows two calculations with no-slip wall boundaries.

### 3.3.9. Data Manipulation

This section deals with the solution data that is produced by running calculations. Two main aspects are covered here: Classical postprocessing mechanisms to obtain desired derived quantities or enhance the solution quality and data reduction techniques that reduce the total amount of data in favor of data handling or storage.

#### 3.3.9.1. Data Postprocessing

Today's visualization software is still not able to directly visualize high order data. The schemes are stuck at interpolating linearly between solution points. Thus, visualizing high order data needs postprocessing steps to prepare the data and retain high order information. Using an equidistant super-sampling with a sufficiently large number of sampling points, one can reasonably resolve high order information. Usually, the number of sampling points $N_{visu}$ should be significantly higher than the number of integration points $N$. Using just evaluations at the integration points itself will not reveal the most accurate results. Figures 3.10 to 3.12 show three different types of visualization of the same calculation result. In Figure 3.10, the solution was visualized on exactly the same nodal points that were also used for computation, in this case Legendre-Gauss-Lobatto nodes. In Figure 3.11, the number of points were kept the same, but this time, equidistant points were used. Finally, in Figure 3.12, twice the number of sample points were used to provide a sufficient super-sampling.

Using just the calculation nodes or other sets with the same number of points clearly does not reveal the full high order information when using standard visualization software.

When the grid alignment is chosen such that no visualization points come to lie at an element boundary, discontinuities, as they arise when using DG schemes, will not be visible. This is important when displaying streamlines or contour lines that heavily rely on the smoothness of the data. By shifting the equidistant visualization grid by half of its spacing, the desired effect is achieved. Figure 3.13 shows a schematic view of a solution visualized with and without shifted visualization points.

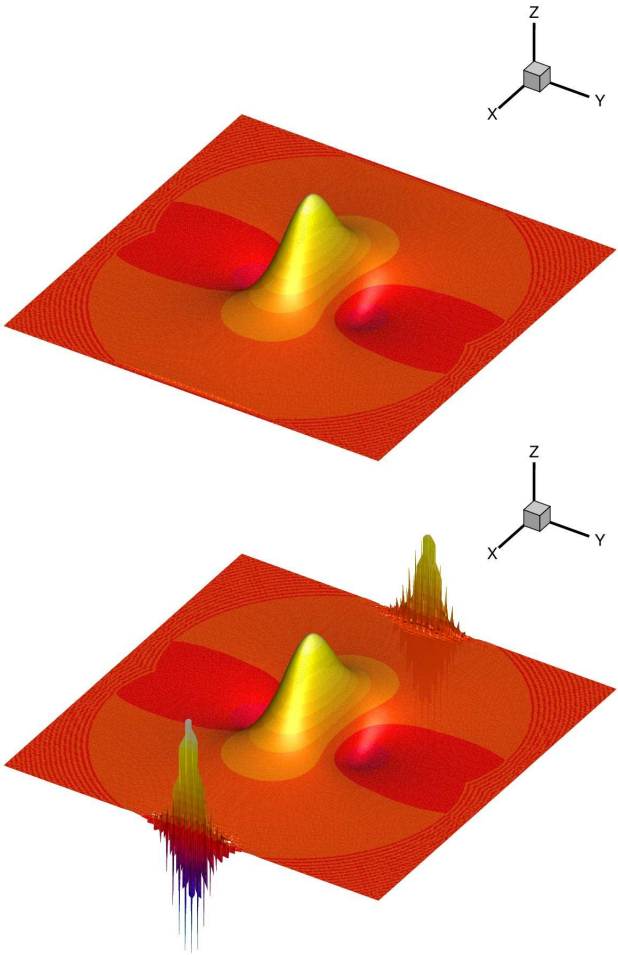



**Figure 3.9.:** Divergence error for a test case containing only no-slip wall boundaries with (top) and without (bottom) special damping treatment.

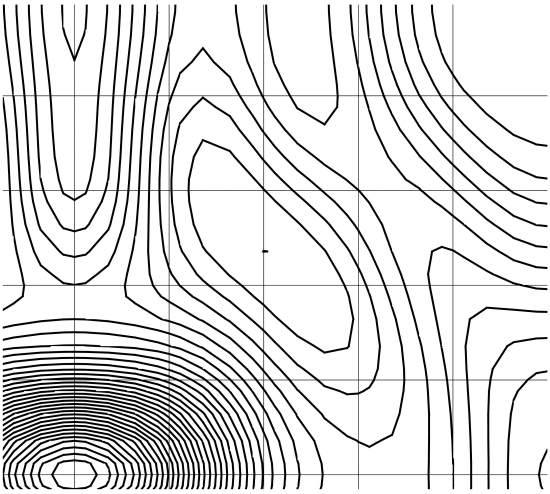


**Figure 3.10.:** Three different visualizations of the same calculation result. Visualization using Legendre-Gauss-Lobatto calculation nodes.

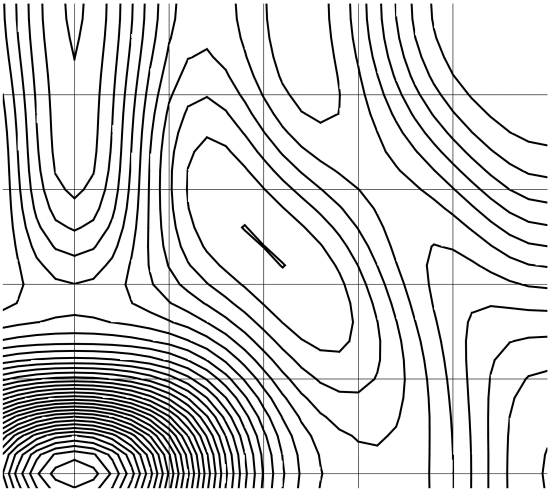


**Figure 3.11.:** Three different visualizations of the same calculation result. Visualization using equidistant points of calculation order.

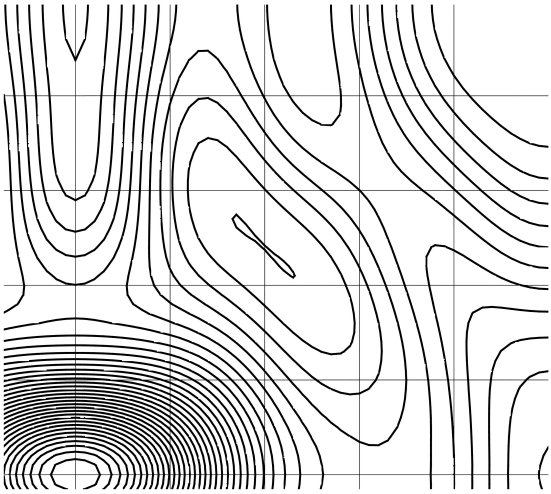


**Figure 3.12.:** Three different visualizations of the same calculation result. Visualization using equidistant points of 2× calculation order.

#### 3.3.9.2. Data Reduction

Especially when it comes to large scale HPC calculations, a huge amount of data is produced. Since this can easily overburden current visualization software both in memory and CPU consumption, strategies for data reduction have to be found to reduce the amount of data and therefore the workload for visualization software, without stripping desired information. Several ways to do so are possible:

- By reducing the computational domain to the region of interest in a postprocessing step, the total amount of data to be visualized can be significantly reduced. The downside of this method is that the region of interest has to be known a priori.

- By writing out just exactly the variables that e.g. will be visualized, additional data reduction can be achieved. Unfortunately, if other data

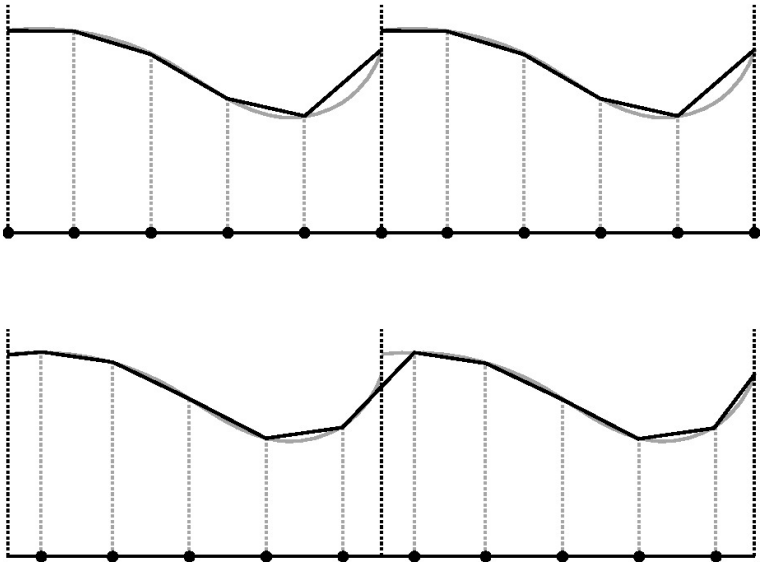


**Figure 3.13.:** Numerical solution, super-sampled equidistantly. Discontinuous visualization (top). Smooth solution, using shifted visualization points (bottom).

that depends on these stripped variables is required later on, the post-processing mechanism must be run again to include this data.

- Fortunately, with a DG polynomial representation at hand, another way of reducing data is possible: stripping higher order DOF in areas where they do not contain information. This should be coupled with a suitable a posteriori sensor, e.g. as described in 3.3.2.1. If the sensor value falls below a certain threshold, the higher order DOF do not contain valuable data and may be removed. By adjusting the sensor's threshold, one could easily adjust the amount of data to fit onto a specific computer and/or visualization system at the cost of a lower resolution. By putting this into an self-adaptive algorithm, one is able to consecutively strip DOFs as they do not contain necessary information.

# 4. Code

This chapter will focus on programming aspects of the code, developed using the numerical ingredients from the previous chapter. Here, necessary steps to enable very large scale scientific computations will be mentioned, alongside with their special requirements. Since developing and maintaining a huge code usually means working within a team, some management aspects are also addressed.

## 4.1. Parallelization

Due to the serial local time stepping mechanism, the parallelization of the nodal DG scheme is not straight forward. At least a minimum – yet to be determined – amount of cells is necessary for a proper run. Moreover, due to the variable time stepping, the computational load is varying too and has to be balanced dynamically. We use the software parMETIS from George Karypis [30] to provide a partitioning. Once a new partition is determined, elements are being sent from one processor to the other one-by-one. There is no global communication involved. The message sizes are therefore small, but come in great quantities. For consistency, a communication-evolve-condition is necessary. We therefore always let the processor with the lower rank start the evolve operation. Custom MPI data types take care of which data will be sent.

### 4.1.1. Shock Capturing

In general, the shock capturing process should not pose a problem for the parallelization. If flagging and shock capturing is done locally without neighboring information, e.g. when using Persson's and Peraire's ideas, no modifications of the parallelization mechanism are necessary. A transfer of data is only necessary, if neighboring information is taken into account. This is the case for the Jameson inspired method, since it needs pressure data from neighboring cell-faces. We additionally send this data when performing the communication step by adding the appropriate values of the mean pressure to the MPI data type that handles all communication information.

### 4.1.2. Divergence Cleaning

For the Riemann solver of the divergence cleaning sub-system, additional neighboring data has to be communicated, such as the local correction speed $c_h$. This is done exactly the same way as the data communication for the Jameson shock capturing method by extending the MPI data type that defines the communication data.

## 4.2. HPC aspects

HPC stands for High Performance Computing and basically means the numerical approach to solve complex scientific problems using large scale computational resources, like supercomputers or large computer clusters. For successfully solving HPC problems, special attention has to be given to code parallelization and optimization, strongly dependent on the computer's system architectures, as well the handling of large data amounts. These aspects will now be addressed.

### 4.2.1. System Architectures

#### 4.2.1.1. JUGENE

The JUGENE supercomputer at the Forschungszentrum Jülich (FZJ) is an IBM BlueGene powered system, consisting of 72 computer racks with a total of 294912 processor cores and 144 Terabytes of main memory. The overall peak performance reaches 1 Petaflops. This system has relatively slow CPUs, but a good computation/bandwidth ratio due to a sophisticated communication layout. One also should mention a relatively small amount of memory per core.

#### 4.2.1.2. HLRB II

The SGI Altix 4700 of the Leibniz Rechenzentrum (LRZ) in Munich, called HLRB II, consists of 9728 Itanium2 processors at 1,6 GHz and 39 Terabytes of main memory. The system consists of 19 NUMA compute partitions with 512 cores each. Being built in 2007, the HLRB II has medium paced CPU's. Since memory is shared amongst 510 compute cores, a massive amount is available.

#### 4.2.1.3. LAKI

This cluster of the Höchstleistungsrechenzentrum Stuttgart (HLRS) consists of 700 Intel Xeon 5560 compute nodes (5600 cores in total) with Nvidia Tesla GPU support for 32 nodes. While the Xeon CPU is quite powerful, the InfiniBand interconnect as a standard solution for cluster systems cannot keep up with it.

### 4.2.2. Code Optimization and Performance

The nodal STE-DG code "HALO" is mainly written according to the FORTRAN90 standard, using INTEL's FORTRAN compiler IFORT to generate the binaries. Since efforts exist to extend the code to more object oriented programming, it is preferable to compile the code with up-to-date FORTRAN2003 compilers, although no special FORTRAN2003 functionalities are yet built in. Since HALO is an unstructured STE-DG code with local time stepping that is capable of performing h- and p-adaptations, the data structure is made of linked pointer lists of elements that contain all necessary calculation data. That structure, together with the serial behavior of the local time stepping algorithm, will make a vectorization of loops almost impossible, since they are always written as a do-while statement. As an example, a loop over all elements within a linked list in HALO will typically look like

```
iElem=>MESH%firstElem
DO WHILE(ASSOCIATED(iElem))
  ...
  <Perform operations>
  ...
  iElem=>iElem%nextElem
ENDDO.
```

Here, the `DO WHILE` statement cannot be vectorized easily, since this loop has no finite end and will probably change during h/p-adaptation. When compiling with `-O3 -qhot=simd` and `-qreport -qxflag=diagnostic` on JUGENE, it can be seen that only very few loops can be optimized by the compiler. As long as the operations to be performed within such a loop are big enough, optimizations could take place there.
When compiling the code for the first time using the IBM XL Fortran compiler, several obstacles lay in the way. One problem could be traced down to the use of pointer sub-arrays as subroutine arguments in HALO. For example,

```
CALL SubroutineXYZ(pointerarray(:,:,ii), &
```

```
bar,pointerarray(:,1,jj),'foo').
```

Here, a subset of the pointer array `pointerarray` occurs twice as a subroutine (or function) argument. In several cases, this caused the compiler to perform optimizations for an infinite time on this routine or the calculation to fail, once this routine was called. In general, using pointers in routine headers may decrease computational efficiency. After disabling polymorphic items by invoking `-qxlf2003= nopolymorphic`, the remaining problematic subroutines have been rewritten by copying the necessary pointer parts into an auxiliary array and/or renaming problematic parts inside the subroutine.
For large scale computations with the STE-DG scheme, a proper code scale-up is necessary. Special scaling tests were set up to test this feature. Herein, we have to distinguish between two substantially different scaling properties: weak scaling and strong scaling. For the first one, the problem size per single processor is kept exactly constant. In other words, the more processors are used, the bigger the problem size is chosen by adding more cells according to the processor layout. By doing so, the overall computational time for the test should ideally not change, since each processor has exactly the same starting position. In reality, MPI communication overhead will result in a longer runtime. By comparing these run times, a weak scale-up table can be arranged. For the latter, the total problem size is kept constant. Thus, the more processors are used, the smaller the problem size gets for a single processor. This test case is not suitable for a local timestepping scheme, since the serial timestepping algorithm always needs a sufficiently large set of elements to be able to perform efficiently. The smaller the set of available elements the more likely the algorithm has to wait for a neighboring processor in order to proceed, since there is no cell left able to perform a time update. Table 4.1 shows the good weak scale-up efficiency of our scheme for up to 4080 processors. The efficiency when running on N processors is calculated as the runtime on one processor divided by the time needed for a calculation on N processors. The load per processor was kept constant at 10.500 DOF ($5^3$ elements with a $7^{th}$ order approximation). We see that there is only very little MPI overhead, even on large

| Number of processors | 1 | 1000 | 2197 | 4080 |
|---|---|---|---|---|
| Efficiency [%] | - | 99.1 | 97.8 | 98.8 |

**Table 4.1.:** Scale-up efficiency of the code on the HLRBII cluster.

numbers of processors.
To determine the weak scaling on the JUGENE supercomputer, larger test cases were set up. $7^{th}$ order calculations from $3^3$ to $9^3$ elements per processor were set up and run on up to 64.000 processor cores. Altogether, a very good weak scaling efficiency was achieved, although a higher processor load gave better scaling results. Due to the limited available memory on JUGENE and the increasing MPI overhead, the maximum amount of DOF per processor for very large scale computations was limited. Table 4.2 shows the results.

| Nb. of procs | 8 | 64 | 512 | 4096 | 29791 | 32768 | 64000 |
|---|---|---|---|---|---|---|---|
| 945 DOF/processor ($3^3$ elements) | | | | | | | |
| Efficiency [%] | 97.05 | 88.24 | 87.27 | 85.39 | | 84.19 | 81.11 |
| 4375 DOF/processor ($5^3$ elements) | | | | | | | |
| Efficiency [%] | 100.38 | 100.0 | | 100.38 | 100.6 | | |
| 7560 DOF/processor ($6^3$ elements) | | | | | | | |
| Efficiency [%] | 99.13 | 98.93 | 98.39 | 98.19 | | | |
| 25515 DOF/processor ($9^3$ elements) | | | | | | | |
| Efficiency [%] | 99.28 | 99.23 | 99.22 | 99.65 | | | |

**Table 4.2.:** Scale-up efficiency of the code on JUGENE cluster.

## 4.3. Code Maintenance

Careful code maintenance is extremely useful. This is especially the case when the numerical codes is developed by a lot of people simultaneously, works for different problems with very little similarities or has to run for a very long time. Two important examples will be pointed out in the following: managing development by using revision control systems and checking for code regressions.

### 4.3.1. Managing Development

When code development is done in a team, care has to be taken to ensure an efficient workflow. It is important that enhancements as well as fixes will be available to the whole team. It is also important that there is no interference

between the team members, even if working on the same part of code. A modification that one developer is programming should not be overwritten by other developers working on the same code section. The team-work therefore needs to be managed. A so-called software versioning and revision control system (RCS) will help to reach these goals. These systems keep track of software changes and, important, also merge different changes of the same code part, as long as they do not affect the exact same line of the code. If this happens, the developer has to decide what to do to resolve the conflicting code lines.

- A central repository herein ensures that there exists only one main version of the code. If more comprehensive modifications are to be made or new functionalities are to be added, a direct copy of the code can be prepared, known as a "branch", which acts as a sandbox for developers to test and evaluate their modifications. Once the features are fully developed and working, the branch can be reintegrated into the main code version, called "trunk". *Pros:* A central working version of the code is always available, where everyone is contributing to. *Cons:* All features have to go to the trunk, which will get cluttered and end up in a non-working state.

- In a distributed RCS, no central repository exists. Instead, each developer hosts and develops his own code version. Developed features can be shared with other repositories similar to a merge-process from a branch into the trunk. *Pros:* More flexible handling, easier to adapt code to ones needs. *Cons:* Needs strong self-control skills of developers, common routines could serve as a coding basis.

Another useful feature stems from the "check in" methodology. Developers have to check in their changes which will be logged by the versioning program. These log files provide useful information about the chances that were made. Since the new data is stored incrementally, the storage space consumption remains low. In addition, old versions of the code can always be restored.

### 4.3.2. Regression Check

Once the code has grown and covers a lot of numerical aspects of different developer teams, modifications can easily have effects beyond the scope of the responsible developer team. Since it cannot be guaranteed that all teams know each others' work at all time, this could pose a threat to the code's integrity, resulting in a growing instability over developing time. By defining test cases

that check certain functionalities and automatically performing defined computations, one could ensure a working code and betimes avoid code regressions. This framework is called regression check and – in our case – is executed daily, only if the code was changed since its last successful run. It provides a basic functionality to easily set up new examples but can be enhanced to perform more complex operations via pre- or postprocessing-scripts written in shell or python. By default, calculation time and $\mathbb{L}_2$-Error norms are being compared to a reference solution that was calculated with a tagged "working" version of the code. In this version, all examples that are being checked should work as expected. Figure 4.1 shows a schematic overview of the framework. The pre- or postprocessing-scripts can basically perform unlimited additional tasks and can therefore be used to modify the regression check to one's own special needs. At all stages of the check, log files are generated, so that the user can follow computations and detect errors easily. When adding new test cases to the regression-check, some topics are important:

- Since each run performs differently in terms of outcome and computation time, suitable tolerances have to be found. Also, the shorter the computation time of an example is, the more the fluctuations in execution time matter. Test computations should give an insight into a suitable run time, before the example is added to the check.

- It is necessary to choose the features to be tested selectively. If too many functionalities are packed into a single test run, the outcome in case of a failure may not be clear enough to determine which feature is the cause. It is better to set up more examples with distinct functionalities.

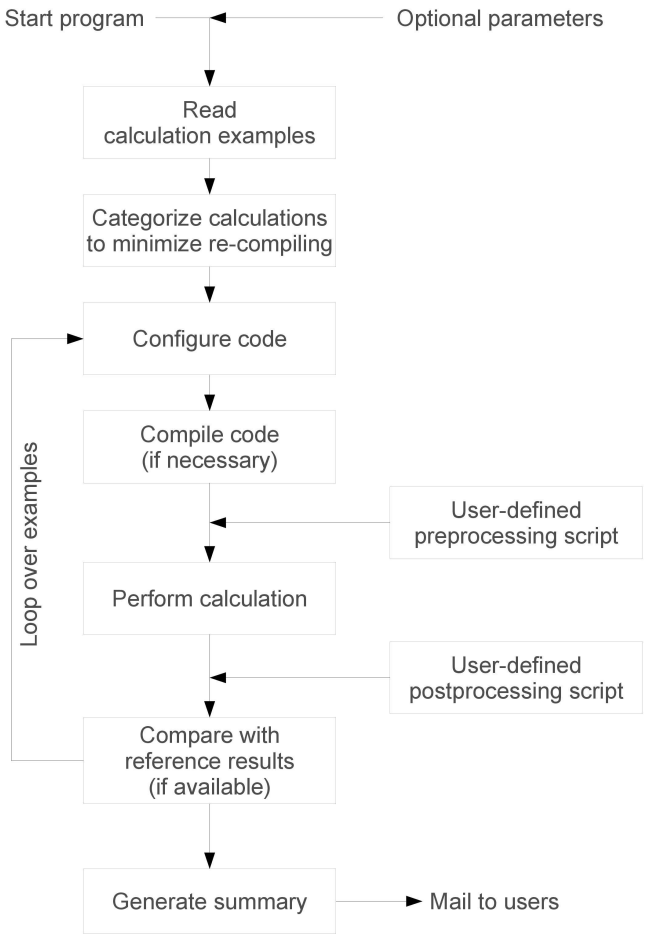



**Figure 4.1.:** Application flow diagram for the regression check.

# 5. Calculations

This chapter deals with computations performed with the developed codes and its ingredients. First, the codes are validated by undergoing a series of convergence tests under different circumstances. We will hereby discover that in the case of MHD, divergence correction is essential for obtaining correct results. Next, special one-dimensional computations will further proof consistency by addressing special aspects of the MHD equation system: non-regular shocks and ambiguous solutions. Once convergence and consistency are successfully validated, two-dimensional test cases are used to investigate the code performance, while three-dimensional large production runs will show the HPC potential of the numerical schemes.

## 5.1. MHD Convergence Tests

We will start with a two-dimensional wave propagation example stated in [1]. Although it was initially not intended to, we use this test problem to perform convergence tests since it has the nice property of an analytical solution easily available at any time. It consists of fluctuations in both the magnetic field and the velocity and is called Alfvén wave decay. It was originally intended to examine the dissipation of torsional Alfvén waves that are made to propagate at a small angle relative to the mesh. The computational domain spans $[-r/2, r/2] \times [-r/2, r/2]$ in the xy-plane with $r = 6$. A uniform density, $\rho_0 = 1$, and pressure, $p_0 = 1$, are initialized on the mesh. The unperturbed velocity is set to $v_0 = 0$, and the unperturbed magnetic field to $B_0 = 1$. The amplitude of the Alfvén wave fluctuation is parametrized in terms of the velocity fluctuation which has a value of $\epsilon = 0.2$ in our calculations. The Alfvén wave is made to propagate at an angle of $\tan^{-1}(1/r) = \tan^{-1}(1/6) = 9.462$ with respect to the y-axis. The direction of wave propagation is along the unit vector

$$\hat{n} = n_x \hat{i} + n_y \hat{j} = \frac{1}{\sqrt{r^2+1}} \hat{i} + \frac{r}{\sqrt{r^2+1}} \hat{j}. \tag{5.1}$$

The phase of the wave is taken to be

$$\phi = \frac{2\pi}{n_y}\left(n_x x + n_y y - V_a t\right), \text{ where } V_a = \frac{B_0}{\sqrt{4\pi\rho_0}}. \tag{5.2}$$

The velocity is set to

$$\mathbf{v} = \left(v_0 n_x - \epsilon n_y \cos\phi\right)\hat{i} + \left(v_0 n_y + \epsilon n_x \cos\phi\right)\hat{j} + \left(\epsilon \sin\phi\right)\hat{k}. \tag{5.3}$$

The magnetic field is given by

$$\begin{aligned}\mathbf{B} = &\left(B_0 n_x + \epsilon n_y \sqrt{2\pi\rho_0}\cos\phi\right)\hat{i} + \left(B_0 n_y - \epsilon n_x \sqrt{4\pi\rho_0}\cos\phi\right)\hat{j} \\ &- \left(\epsilon\sqrt{4\pi\rho_0}\sin\phi\right)\hat{k}.\end{aligned} \tag{5.4}$$

Here, $\hat{i}, \hat{j}, \hat{k}$ denote the unit vectors of the coordinate system. Structured meshes up to $32 \times 32$ quadrilateral cells were used. The following tables show convergence rates in $B_x$ for $2^{nd}$ to $8^{th}$ order of accuracy.
This problem also holds as a three-dimensional convergence test case. No modifications to the initial conditions had to be performed, the computational domain was simply extended to the third dimension.
First, convergence rates for the two-dimensional ideal MHD case are shown in Table 5.1. These are compared to a calculation without the divergence cleaning mechanism. As clearly visible, the $\mathbb{L}_2$ error results show a degradation of up to one order of magnitude, in some cases even resulting in a degraded order of convergence, if divergence cleaning is neglected.

| **Nb cells**$^2$ | **Nb DOF** | $\|B_x\|_{L_2}$ **w divB corr.** | $\mathcal{O}_{L_2}$ | $\|B_x\|_{L_2}$ **w/o divB corr.** | $\mathcal{O}_{L_2}$ |
|---|---|---|---|---|---|
| | | $\mathcal{P}1$ **DG** | | | |
| **4** | **48** | 5.36E-02 | | 4.92E-02 | |
| **8** | **192** | 1.55E-02 | **1.8** | 1.41E-02 | **1.8** |
| **16** | **768** | 3.52E-03 | **2.1** | 3.32E-03 | **2.1** |
| **32** | **3072** | 7.89E-04 | **2.2** | 7.73E-04 | **2.1** |
| | | $\mathcal{P}2$ **DG** | | | |
| **4** | **96** | 1.20E-02 | | 1.16E-02 | |
| **8** | **384** | 1.29E-03 | **3.2** | 1.40E-03 | **3.1** |
| **16** | **1536** | 1.44E-04 | **3.2** | 1.85E-04 | **2.9** |
| **32** | **6144** | 1.66E-05 | **3.1** | 2.50E-05 | **2.9** |

| | | | | | |
|---|---|---|---|---|---|
| | | $\mathcal{P}3$ **DG** | | | |
| **4** | **160** | 2.10E-03 | | 2.38E-03 | |
| **8** | **640** | 1.25E-04 | **4.1** | 1.62E-04 | **3.9** |
| **16** | **2560** | 7.09E-06 | **4.1** | 1.11E-05 | **3.9** |
| **32** | **10240** | 4.51E-07 | **4.0** | 6.91E-07 | **4.0** |
| | | $\mathcal{P}4$ **DG** | | | |
| **4** | **240** | 2.93E-04 | | 3.54E-04 | |
| **8** | **960** | 9.21E-06 | **5.0** | 1.33E-05 | **4.7** |
| **16** | **3840** | 3.00E-07 | **4.9** | 4.61E-07 | **4.8** |
| **32** | **15360** | 9.55E-09 | **5.0** | 1.50E-08 | **4.9** |
| | | $\mathcal{P}5$ **DG** | | | |
| **2** | **84** | 2.86E-03 | | 3.38E-03 | |
| **4** | **336** | 4.09E-05 | **6.1** | 5.40E-05 | **6.0** |
| **8** | **1344** | 7.02E-07 | **5.9** | 1.20E-06 | **5.5** |
| **16** | **5376** | 1.09E-08 | **6.0** | 3.86E-08 | **5.0** |
| | | $\mathcal{P}6$ **DG** | | | |
| **2** | **112** | 3.63E-04 | | 3.68E-04 | |
| **4** | **448** | 4.69E-06 | **6.3** | 6.51E-06 | **5.8** |
| **8** | **1792** | 4.12E-08 | **6.8** | 5.90E-08 | **6.8** |
| **16** | **7168** | 3.47E-10 | **6.9** | 5.13E-10 | **6.9** |
| | | $\mathcal{P}7$ **DG** | | | |
| **2** | **144** | 1.36E-04 | | 1.61E-04 | |
| **4** | **576** | 4.36E-07 | **8.3** | 6.12E-07 | **8.0** |
| **8** | **2304** | 1.81E-09 | **7.9** | 3.62E-09 | **7.4** |
| **16** | **9216** | 7.14E-12 | **8.0** | 2.16E-11 | **7.4** |

**Table 5.1.:** Experimental order of convergence for the two-dimensional Alfvén wave test problem for ideal MHD on an equidistant grid.

For the three-dimensional extension of this problem, convergence rates of polynomial degree $\mathcal{P}4$ and $\mathcal{P}5$ with divergence cleaning are illustrated in Table 5.2 that show the designed order of accuracy. Structured hexahedral meshes up to $16^3$ elements were used.

| **Nb cells**$^3$ | **Nb DOF** | $\|B_x\|_{L_2}$ **w divB corr.** | $\mathcal{O}_{L_2}$ |
|---|---|---|---|
| | | $\mathcal{P}4$ **DG** | |
| **2** | **280** | 1.84E-02 | |
| **4** | **2240** | 4.12E-04 | **5.5** |
| **8** | **17920** | 1.06E-05 | **5.3** |
| **16** | **143360** | 3.32E-07 | **5.0** |
| | | $\mathcal{P}5$ **DG** | |
| **2** | **448** | 7.78E-03 | |
| **4** | **3584** | 5.01E-05 | **7.3** |
| **8** | **28672** | 1.11E-06 | **5.5** |
| **16** | **229376** | 1.75E-08 | **6.0** |

**Table 5.2.:** Experimental order of convergence for the three-dimensional Alfvén wave test problem for ideal MHD on an equidistant grid.

So far, we were able to show convergence for the ideal MHD equations in two and three dimensions. For viscous MHD, the Alfvén test case is not sufficient, since it does not provide an exact solution for the viscous part.
Therefore, we were now using the so-called manufactured solution technique for performing convergence tests of the numerical scheme for viscous MHD. To be able to find an exact solution, we choose an arbitrary smooth analytical function and insert it directly into the viscous MHD equations. For our test, we have chosen

$$\begin{array}{rclrclrcl} \rho & = & sin(\beta)+2; & u & = & 1; & v & = & 1; \\ e & = & sin(\beta)+2; & B_x & = & sin(\beta)+2; & B_y & = & -sin(\beta)+2, \end{array} \tag{5.5}$$

with $\beta = 2\pi \sum_{j=1}^{dim} x_j - 4t$, where $x_j$ are the spatial coordinates and $t$ the time. Viscosity $\mu$ and resistivity $\eta$ were set to 0.05. The resulting right hand side (see Appendix B for more details) is then added as a source term directly into the code. Table 5.3 shows the convergence order of our numerical scheme for two-dimensional viscous MHD using third and fourth order DG polynomials. The $\mathbb{L}_2$ error norm of the energy is used for the calculation. Table 5.3 shows that we do not obtain the desired order of accuracy when the divergence correction is neglected. In addition, also the absolute values of the error norms are significantly worse in that case. By using our proposed correction method,

| **Nb cells**$^2$ | **Nb DOF** | $\|E\|_{L_2}$ **w divB** | $\mathcal{O}_{L_2}$ **corr.** | $\|E\|_{L_2}$ **w/o divB** | $\mathcal{O}_{L_2}$ **corr.** |
|---|---|---|---|---|---|
| | | $\mathcal{P}3$ **DG** | | | |
| **4** | **160** | 1.82E-01 | | 2.70E-01 | |
| **8** | **640** | 9.29E-03 | **4.3** | 1.98E-02 | **3.8** |
| **16** | **2560** | 4.98E-04 | **4.2** | 9.60E-04 | **4.4** |
| **32** | **10240** | 2.95E-05 | **4.1** | 6.58E-05 | **3.9** |
| | | $\mathcal{P}4$ **DG** | | | |
| **4** | **240** | 3.63E-02 | | 4.34E-02 | |
| **8** | **960** | 9.82E-04 | **5.2** | 1.22E-03 | **5.2** |
| **16** | **3840** | 3.07E-05 | **5.0** | 5.19E-05 | **4.6** |
| **32** | **15360** | 9.36E-07 | **5.0** | 2.28E-06 | **4.5** |

**Table 5.3.:** Order of accuracy of the STE-DG scheme for 2D viscous MHD with and without divergence correction.

we where able to achieve the accuracy for both odd and even orders for the viscous MHD equations.
We now want to get a deeper understanding of how the correction method fits into the DG scheme and how exactly it establishes the correct convergence rate. It is therefore important to have a look at the error norm of the divergence itself. Since the divergence operator is formed by derivatives of the magnetic field, we expect the $\mathbb{L}_2$ norm to be one order of accuracy lower than the parameter it is acting on, in our case the magnetic field $B$. In Table 5.4 we are plotting the results, again with and without divergence correction. We can clearly see that our assumptions are only true in the case where we are performing a divergence correction. Otherwise, the $\mathbb{L}_2$ norm of the divergence is much worse. We can also see that divergence errors are not kept at a machine zero, even in the case we perform a divergence correction. As long as the decay of the divergence goes along the schemes order of accuracy, there is no need to do better than this, since a further improvement of the solution may not be achieved. The convergence test proved that the evolution of the magnetic field's divergence goes along the other variables and is therefore treated consistently.

| Nb cells$^2$ | Nb DOF | $\|DivB\|_{L_2}$ w divB corr. | $\mathcal{O}_{L_2}$ | $\|DivB\|_{L_2}$ w/o divB corr. | $\mathcal{O}_{L_2}$ |
|---|---|---|---|---|---|
| | | $\mathcal{P}3$ **DG** | | | |
| **4** | **160** | 1.93E-01 | | 7.48E-01 | |
| **8** | **640** | 1.62E-02 | **3.6** | 1.49E-01 | **2.3** |
| **16** | **2560** | 2.87E-03 | **2.5** | 2.63E-02 | **2.5** |
| **32** | **10240** | 4.55E-04 | **2.7** | 5.08E-03 | **2.4** |
| | | $\mathcal{P}4$ **DG** | | | |
| **4** | **240** | 1.48E-01 | | 2.62E-01 | |
| **8** | **960** | 1.87E-03 | **6.3** | 2.20E-02 | **3.6** |
| **16** | **3840** | 1.00E-04 | **4.2** | 2.42E-03 | **3.2** |
| **32** | **15360** | 6.50E-06 | **4.0** | 2.92E-04 | **3.1** |

**Table 5.4.:** Order of accuracy for the divergence of the magnetic field, divB, with and without divergence correction.

## 5.2. One-Dimensional Calculations

Amongst basic testing, the one-dimensional calculations mainly target selected specialties of the MHD equations. Divergence cleaning is not necessary here, since the magnetic field is constant in $x$. We will show two distinct Riemann problems, initialized in a shock-tube fashion.

### 5.2.1. Compound Shock

This test case shows the formation of a left-going slow compound wave together with a weak right-going slow shock and a contact discontinuity. Left- and right-going rarefactions flank both sides of the Riemann problem. A compound shock is a MHD specialty: It is a slow shock with an attached rarefaction, which is not possible to develop in pure hydrodynamics. Since the MHD system is not strictly hyperbolic, because it contains linearly degenerated characteristic fields, irregular over-compressive shock patterns can occur. Figure 5.1 shows density and magnetic field in y direction.
In all one-dimensional calculations, $\gamma = 5/3$, L and R denote the left and right states, $B_x$ is constant and $x_c$ denotes the position where the discontinuity

was initialized. The simulation end-time is denoted by $t_s$. The computational domain for all test cases is $[-0.5, 0.5]$. For a first impression, images of the density and the $y$-component of the magnetic field for these problems are shown, if possible. They were computed with the help of an exact Riemann solver.
The initial condition for this shock tube problem are given according to Table 5.5

| Pos | $\rho$ | $p$ | $u$ | $v$ | $w$ | $B_x$ | $B_y$ | $B_z$ | $x_c$ | $t_s$ |
|---|---|---|---|---|---|---|---|---|---|---|
| L | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.75 | 1.0 | 0.0 | 0.0 | 0.1 |
| R | 0.125 | 0.1 | 0.0 | 0.0 | 0.0 | 0.75 | -1.0 | 0.0 | | |

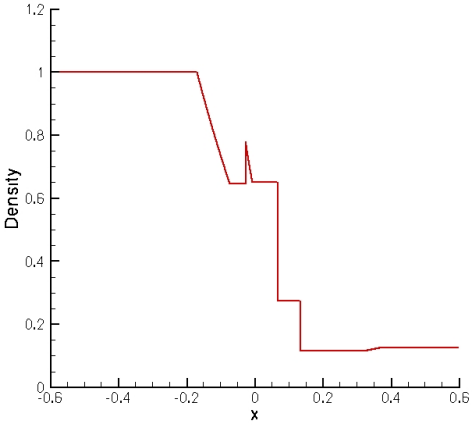
**Table 5.5.:** Initial values for the compound shock test case.

### 5.2.2. Pseudo Convergence

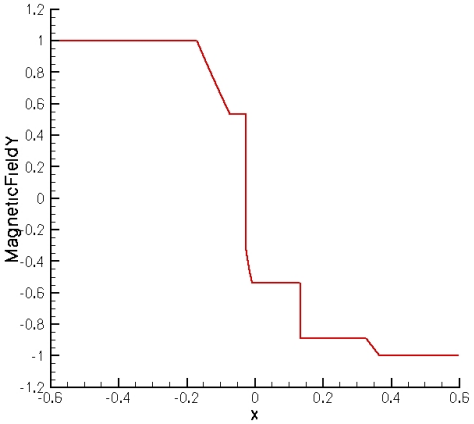
This one-dimensional test case shows a pseudo convergence behavior that is linked to ambiguous MHD solutions. For this problem, a low order coarse grid solution would always show a compound wave. The images in Figure 5.2 show both the coarse (solid) and the fine (dashed) grid solution. These phenomena are studied extensively in [51] and [50], so the reader is referred there.

| Pos | $\rho$ | $p$ | $u$ | $v$ | $w$ | $B_x$ | $B_y$ | $B_z$ | $x_c$ | $t_s$ |
|---|---|---|---|---|---|---|---|---|---|---|
| L | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.4 |
| R | 0.2 | 0.2 | 0.0 | 0.0 | 0.0 | 1.0 | cos(3.0) | sin(3.0) | | |

**Table 5.6.:** Initial values for the pseudo convergence test case.

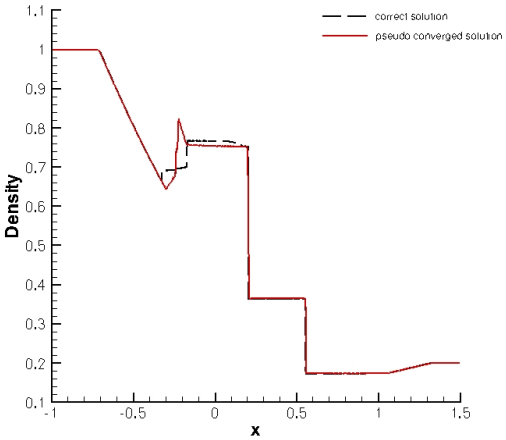



(a) density plot

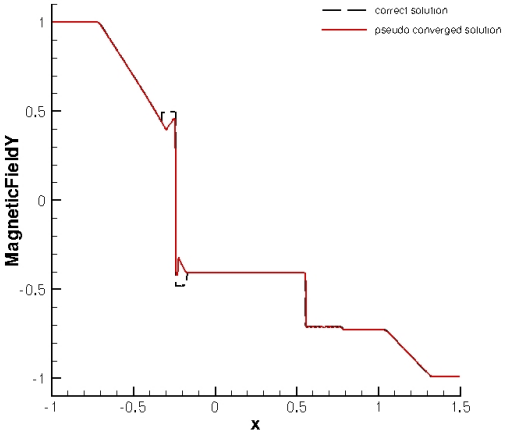



(b) $B_y$ plot

**Figure 5.1.:** Reference solution of the compound shocks problem.




(a) density plot




(b) $B_y$ plot

**Figure 5.2.:** Solution of the pseudo convergence problem.

## 5.3. Two-Dimensional Calculations

We are now moving on to two-dimensional calculations well known from literature. It is therefore no problem to find reference results to compare our scheme to.

### 5.3.1. Double Mach Reflection

This Euler test case aims at the shock capturing properties of the numerical scheme. Originally shown in [56], it consists of a strong shock at Mach number $Ma = 10$, approaching a wall at an angle of 60°. Once the shock hits the wall, a complex shock reflection pattern develops: In the front region of the shock, a contact discontinuity starts to roll up and gets unstable, surrounded by two shocks of different strength, all meeting in a single point. This so-called triple-point region is the most interesting part, since the necessary shock capturing or limiting process could badly influence the flow development here. Especially the contact discontinuity would not show its distinct unstable behavior if the shock capturing mechanisms had falsely acted on it. The results can easily be compared to other numerical schemes and methods. This test case was set up with mesh adaptation, using the pressure difference indicator for shock capturing, see Section 3.3.2.3. The mechanism was set up such that two refining cycles were allowed and that refining was preferred to adding artificial viscosity in shock regions. The computational setup is shown in Figure 5.3.

Figure 5.4 shows the triple-point region at end time $t = 2.0$. The roll-up of the contact-discontinuity as well as its developing instability is clearly visible. Additionally, the adapted grid lines are shown.

### 5.3.2. Magnetic Rotor

The first example, also presented in [1], consists of a spinning dense circle within an initially stationary fluid. Inside the circle, we have a density of $\rho = 10$ which is set to $\rho = 1$ in the ambient fluid. Pressure and magnetic field are uniquely defined as $p = 1$ and $\mathbf{B} = \left(\frac{2.5}{\sqrt{4\pi}}, 0, 0\right)$. Within the circle, we set a uniform angular velocity up to a radius of $r = 0.1$ and a value of $v_a = 1$ at that position. The ratio of specific heats is $\gamma = \frac{5}{3}$. Periodic boundaries were used and this example was calculated up to a time of $t = 0.29$, which corresponds to

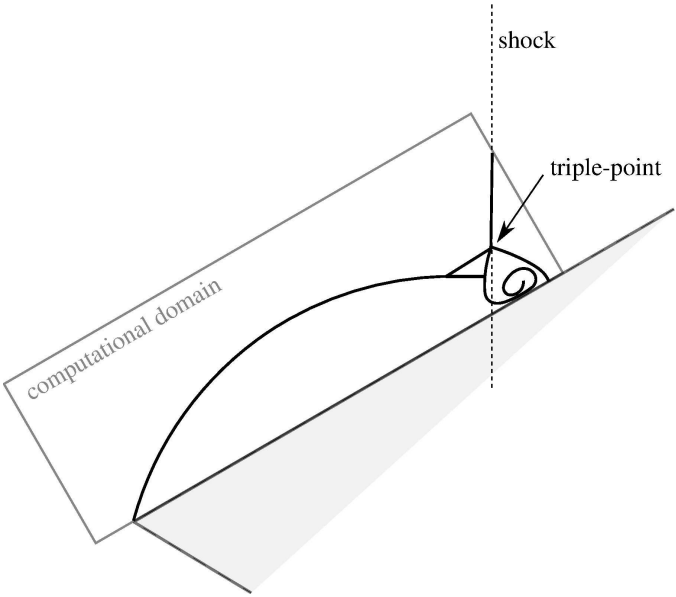



**Figure 5.3.:** Layout of the double mach reflection problem.

calculations found in the literature. The following Figure 5.5 shows the density distribution and the magnetic field magnitude of a $5^{th}$ order calculation on a $200 \times 200$ grid. Divergence cleaning was performed as described previously. When the spinning fluid interacts with the ambient magnetic field, Alfvén waves are sent out, which can be seen best on the image to the right.

### 5.3.3. Inviscid Orszag-Tang Vortex

This vortex system of Orszag and Tang [38] which was studied extensively in [42] and [11] is an ambitious test problem for any numerical scheme. In our case, the computational domain is $[0; 1] \times [0; 1]$ with periodic boundaries. The

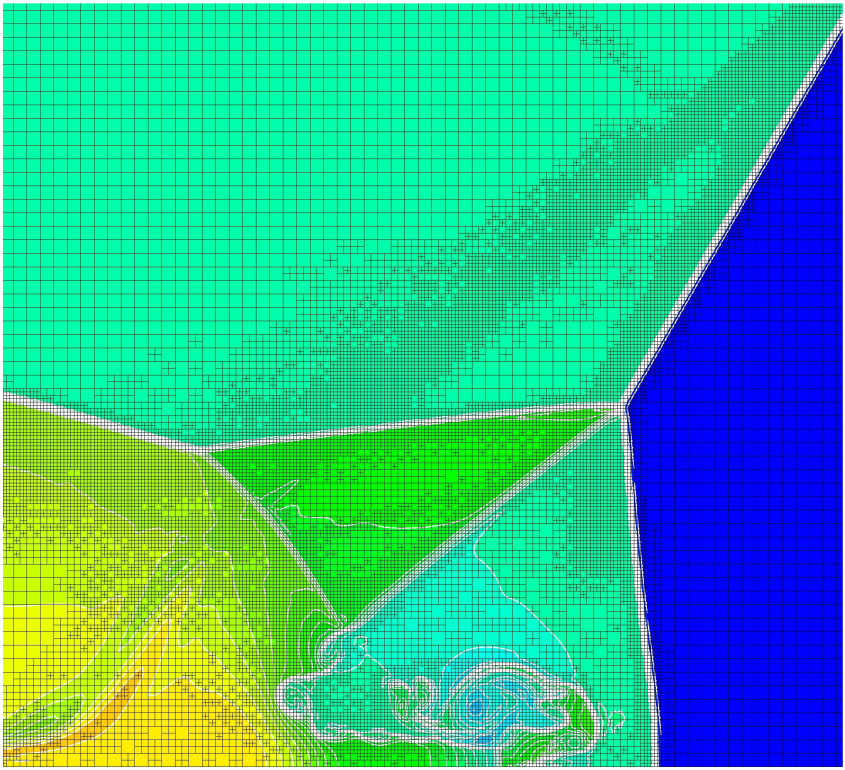
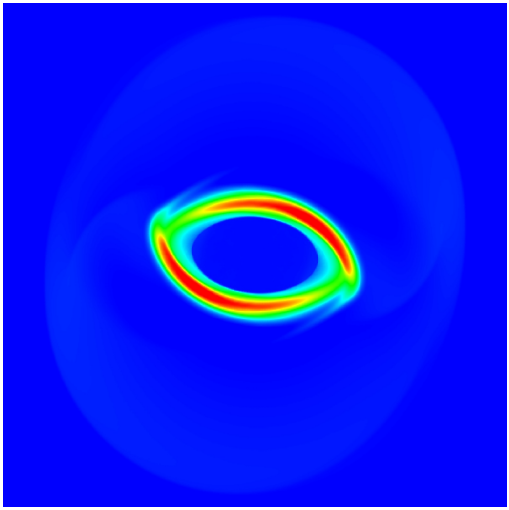
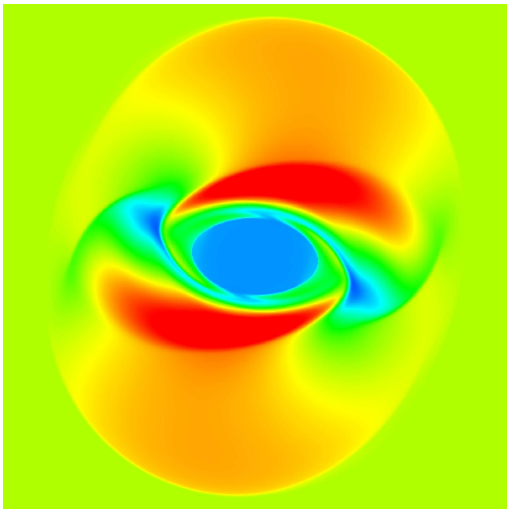


**Figure 5.4.:** Plots of the triple-point region of the double mach reflection problem at end time $t = 2.0$. Colors and white contour lines indicate density, adapted mesh is shown in black.



(a) density plot at $t = 0.29$



(b) magnetic field magnitude plot at $t = 0.29$

**Figure 5.5.:** Plots of the density distribution and the magnetic field magnitude for the magnetic rotor problem at time $t = 0.29$.

initial condition of the problem is given by

$$\begin{array}{lllllllllll} \rho & = & \gamma\, e; & u & = & -sin(2\pi y); & v & = & sin(2\pi x); \\ e & = & \frac{10}{24}\pi; & B_x & = & -\frac{1}{\sqrt{4\pi}}\, sin(2\pi y); & B_y & = & \frac{1}{\sqrt{4\pi}}\, sin(4\pi x), \end{array} \tag{5.6}$$

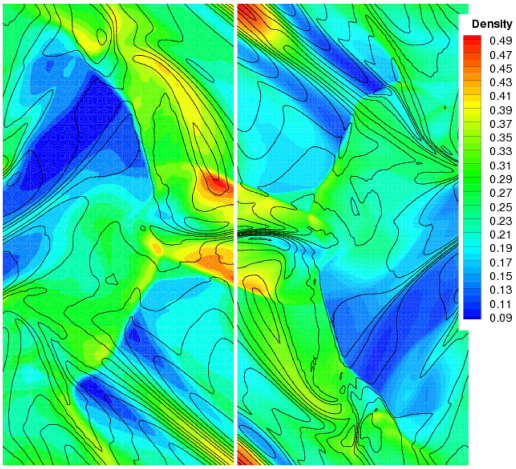
with $\gamma = \frac{5}{3}$. The Mach number is set to 1.0. Ideal MHD calculations on a $50 \times 50$, a $100 \times 100$ and a $200 \times 200$ grid run up to $t = 0.5$. By then, several shocks have crossed the computational domain and a vortex system is formed near the center. Figures 5.6(a) and 5.6(b) show a very good agreement with reference results in the literature, e.g. [34] or [47]. For various numerical schemes without divergence cleaning, this test problem fails or will at least produce severe errors, see [34]. For the $100 \times 100$ calculation, we were able to keep the $\mathbb{L}_2$ norm of the divergence errors below $1 \cdot 10^{-2}$. It can be seen clearly that both the $100 \times 100$ and the $200 \times 200$ calculation resolves the small-scale structures much better. We can also see a grid convergence, since the results from the $100 \times 100$ and the $200 \times 200$ calculation are almost identical. Due to different shock capturing settings, the $200 \times 200$ calculation shows some more oscillations, though.

### 5.3.4. Viscous Orszag-Tang Vortex

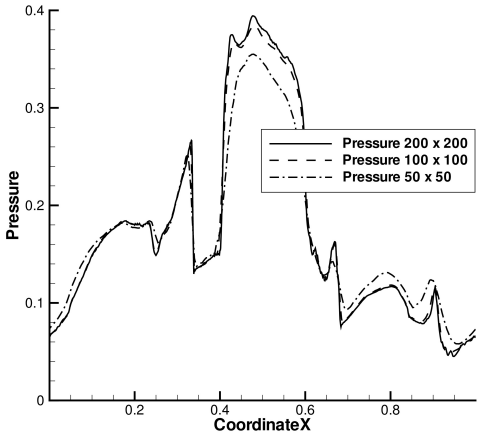
In this calculation, we are using the same initial conditions as shown under 5.3.3. However, we set up a viscous calculation, where viscosity and resistivity were defined to be $\mu = \frac{1}{100} \cdot \frac{1}{4\pi}$ and $\eta = \frac{1}{100} \cdot \frac{1}{4\pi}$ respectively and the Prandtl number was set to 1. This setting closely corresponds to viscous and resistive Lundquist numbers of $S_v = S_r = 100 \cdot 4\pi$ for a system length of $L_0 = 1$. Figure 5.7(a) shows plots of density and magnetic field. Figure 5.7(b) displays the magnetic field streamlines at time $t = 0.5$ over the density distribution. The results are linked closely to [11] and [42] where calculations with differentiation Mach numbers were performed.

### 5.3.5. Magnetic Cloud

This test case consists of a shock traveling over a dense magnetic cloud. The problem is set up according to Figure 5.8 and can easily be transformed into three space dimensions. The computational domain spans a $[-1.0, -1.5] \times [3.5, 1.5]$ rectangle. In the beginning, a $Ma = 8$ shock is initialized in the right half of the computational domain at $x = 2.5$, while a circular dense cloud with a radius of $r = 0.3$ is placed at $x = 2.85$. The pre-shock density and pressure

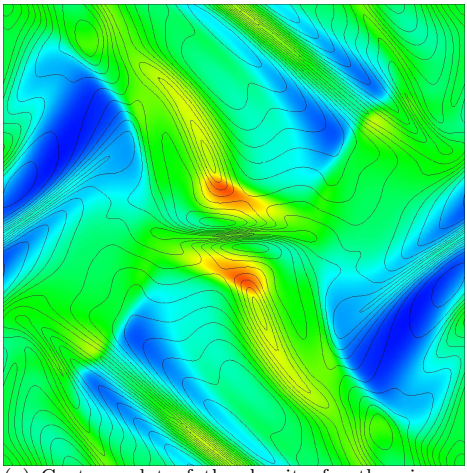



(a) Density plot for the Orszag-Tang vortex at $t = 0.5$ on a $50 \times 50$ grid (left) and a $100 \times 100$ grid (right). Contour levels of the magnetic field magnitude are also shown.

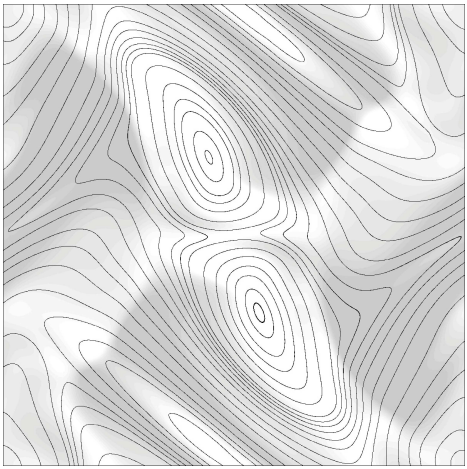



(b) 1D profile of the Orszag-Tang vortex at $y = 4.275$ for all three calculations.

**Figure 5.6.:** Plots of the Orszag-Tang vortex at time $t = 0.5$.



(a) Contour plot of the density for the viscous Orszag-Tang vortex at time level $t = 0.5$. Contour levels of the magnetic field magnitude are also shown.



(b) Magnetic field streamlines over density distribution for the viscous Orszag-Tang vortex at time level $t = 0.5$.

**Figure 5.7.:** Plots of the viscous Orszag-Tang vortex at time $t = 0.5$.

outside of the cloud are both set to unity, the flow speed in x-direction is set to $-\gamma Ma_{shock}$, where again $\gamma = \frac{5}{3}$. This way, the shock is kept "static" during the calculation. The post-shock state is determined by the Rankine-Hugoniot conditions. The magnetic field is zero everywhere in the domain except for the cloud, where it is initialized as circular with a maximum of 1.3. Inside the cloud, a smooth function is applied to bring the cloud density of $\rho_{cloud} = 2.0$ in line with the surroundings. Figure 5.8 shows the computational setup for this calculation.

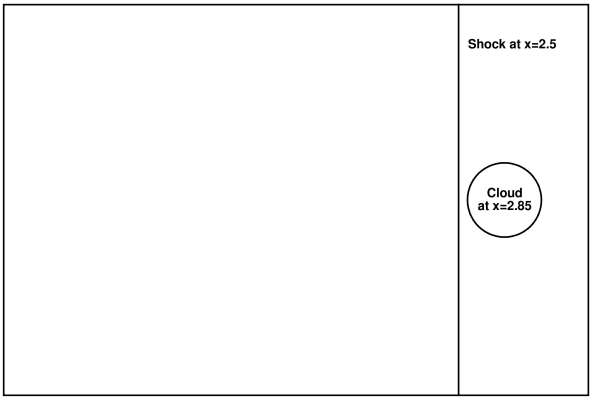



**Figure 5.8.:** Schematic view of the computational setup for the magnetic cloud test case.

Figure 5.9 shows a density plot at $t = 0.9$. The cloud has traveled over the shock and is now deformed. We can see its characteristic roll-up.
A purely hydrodynamic Euler calculations was performed with the same setting and is shown in Figure 5.10. Here, the absence of the magnetic field permits a stronger roll-up phenomenon, once the cloud interacts with the shock. This corresponds to findings in the literature, e.g. [54].

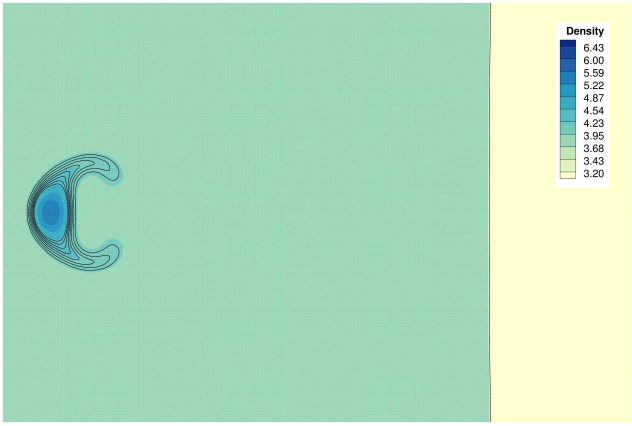



**Figure 5.9.:** Density plot for the magnetic cloud test case (MHD version) at $t = 0.9$.

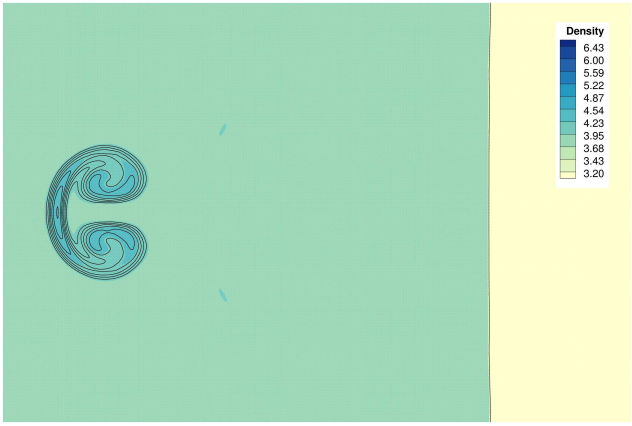



**Figure 5.10.:** Density plot for the magnetic cloud test case (purely hydrodynamic Euler version) at $t = 0.9$.

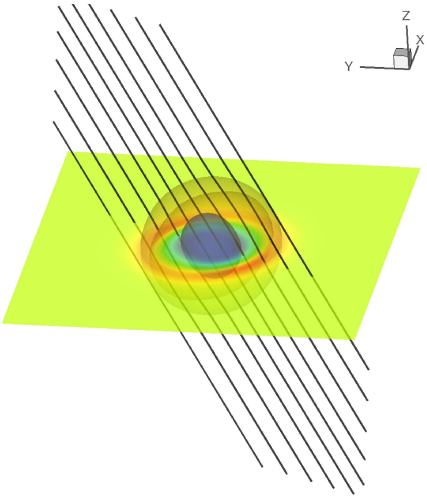
## 5.4. Three-Dimensional Calculations

We now move on to a more complex calculation in three space dimensions. This setup is a three-dimensional extension from a well known two-dimensional test case.

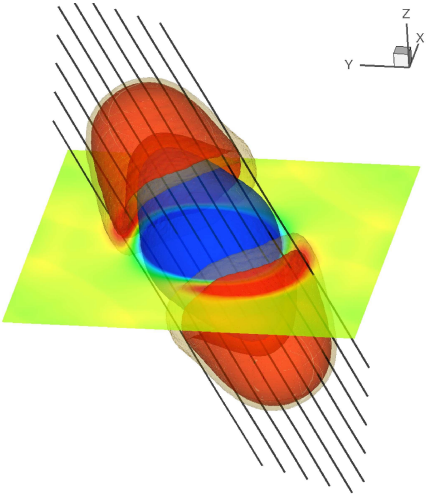
### 5.4.1. Magnetic Blast

This originally two-dimensional test case in the spirit of [3] aims especially at challenging the shock capturing properties of the numerical scheme. The presented three dimensional extension of this problem is achieved easily, since the simple initial conditions only need to be changed slightly. The initial setting is the following:
At the origin of the $[-0.5, 0.5]\times[-0.5, 0.5]\times[-0.5, 0.5]$ domain, a spherical "bubble" of radius $r = 0.1$ with initial pressure $p = 10$ is located. The ambient fluid has a pressure of $p_0 = 0.1$ as well as a magnetic field of $\mathbf{B} = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)\cdot\sqrt{4\pi}$. Using this setting, the magnetic field lines are diagonally aligned, resulting in a real three-dimensional characteristic of this test case. The initial density is set to 1 and the velocity remains zero. In our case, the problem is set up with periodic boundaries, although other boundary conditions are possible as well. Shock capturing is performed using our artificial viscosity approach.
In Figure 5.11 one can easily see that the propagation of the density shock-wave is stretched along the magnetic field lines, as the calculation continues.
Because strong shocks develop, Figure 5.12 shows a close-up of the blast-region with shocks fronts displayed as iso-contours. Also shown are shock cells that are treated with viscosity, mainly located at the shock fronts.

Since the problem run on 125 processors in parallel, a load-balancing mechanism had to ensure the computational efficiency of the calculation in multi processor mode. Figure 5.13 shows a snapshot of the domain partitioning at the end of the calculation. In the outer region of the domain, the initial setup, consisting of a Cartesian distribution, remained almost unchanged. However, in the inner region, where the flow field is developed, fine-grained partitions are visible. This ensures an almost equal computational load on all processors. Latency differences are efficiently covered by the local timestepping mechanism.




(a) Density plot at $t = 0.1$.




(b) Density plot at $t = 0.5$.

**Figure 5.11.:** Plot of the density distribution as a slice in the y-z plane together with contour levels for the MHD blast problem at time levels $t = 0.2$ and $t = 2.0$. The rakes indicate the magnetic field direction.

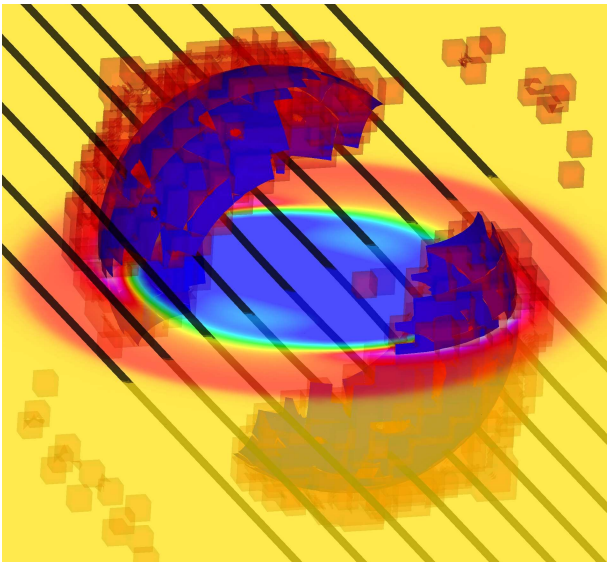


**Figure 5.12.:** Plot of shock fronts and viscous cells for the MHD blast problem.

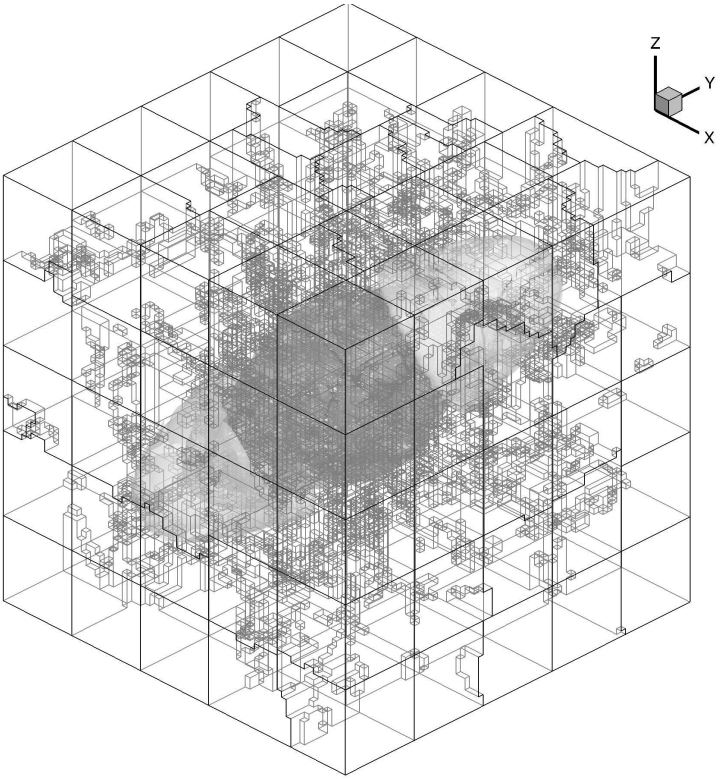



**Figure 5.13.:** Plot of the domain partitioning for a 125 processor calculation of the MHD blast test case at end time $t = 2.0$.

# 6. Conclusion and Prospects

The aim of this work is to provide numerical ingredients for Discontinuous Galerkin schemes ranging from inviscid Euler equations to viscous MHD equations. To achieve this, numerical approaches were shown, beginning with basic mathematical models and ending with code optimization aspects. Hereby, special attention was drawn to shock capturing and – in the context of MHD equations – divergence cleaning. Artificial viscosity is used to smear the shock profiles to make them resolvable for high order schemes. Shock capturing mechanisms by Persson and Peraire, initially developed for implicit schemes, were modified and put into an explicit context. Inspired by the ideas of Jameson, a robust and easy to use shock capturing mechanism was developed that makes use of the jumps in pressure between cells to determine the necessary amount of artificial viscosity. Parameter studies were set up to determine their optimal settings. The hyperbolic GLM divergence cleaning strategy, initially developed by Dedner et al. was localized and a sub-cycling mechanism was developed. Via sub-cycling, even strong divergence errors can be efficiently removed. In that context, additional considerations regarding boundary conditions were made. Increasing the divergence error damping factor close to the boundary can prevent unwanted reflections there.

Since a highly efficient local timestepping DG scheme with a limited information horizon was used, we had to ensure that all developed numerical ingredients act as local as possible. By affecting to many grid cells at once, the efficiency of the timestepping algorithm would have been reduced, if not, disabled. By allowing each cell to run with its own local timestep, solely determined by its local CFL condition, a significant speed-up in comparison to standard global method is achieved. All presented numerical ingredients do not only work under such local conditions, but are also explicitly making use out of them to gain a better efficiency and/or accuracy. In addition, postprocessing and data reduction aspects, as well as management tools were discussed, such as revision control programs or code regression checks that may help to organize code development in larger working groups.

Thus, the local timestepping functionality could be successfully brought to the MHD equations and its characteristics and, furthermore, could be used

to enhance the efficiency of numerical MHD calculations. Still a lot of open questions remain for the addressed topics:

- Still no "natural" way of high order shock capturing has been discovered. Here, the word natural means an easy and robust mechanism that may use high order data- and grid-settings but does not depend on complex and unclear parameter settings. Although the strategies shown in this thesis may be seen as a first small step towards that goal, there is still plenty of work to do. First approaches using a low order sub-cell (re)construction, shown by Huerta in [25] may be another promising way to solve this problem.

- Regarding the DG method's ability to handle complex boundaries, more work has to be done to find applications that make use of this feature. Especially in the context of astrophysics, this could be quite difficult, since applications there are often too easy with respect to boundaries, but are otherwise more physics focused.

- For large scale turbulent MHD computations, suitable examples are available. Especially in the context of planet formation, as seen within so-called accretion discs, turbulent flows can be simulated. Additional radiation effects and special shear-periodic boundaries that reflect the rotational movement of such discs may pose additional difficulties that have to be solved first. A DFG renewal-proposal to address that matter has been filed and positively reviewed.

- For calculations on Cartesian grids, as often the case for MHD computations, the highly efficient Discontinuous Galerkin spectral element method (DGSEM), see [32], could be used. The tensor-product formulation of this nodal DG scheme with its collocation of interpolation and integration points provides for a superb accuracy and efficiency, especially for massively parallel calculations. This can be extremely advantageous for large scale MHD problems.

# A. CK Procedure

The CK procedure in two space dimensions is built up the following way: From the MHD equations with artificial viscosity shock capturing, we directly obtain

$$\frac{\partial U}{\partial t} = -\frac{\partial}{\partial x}(F_1 - F_1^v) - \frac{\partial}{\partial y}(F_2 - F_2^v), \tag{A.1}$$

where $F_1^v = \nu(\vec{x})\frac{\partial U}{\partial x}$, or more general

$$\frac{\partial^{m+p+o} U}{\partial x^m y^p t^o} = -\frac{\partial^{m+p+o}}{\partial x^{m+1} y^p t^{o-1}}(F_1 - F_1^v) - \frac{\partial^{m+p+o}}{\partial x^m y^{p+1} t^{o-1}}(F_2 - F_2^v). \tag{A.2}$$

We now need to introduce auxiliary variables to avoid quotients of two functions. This way is favorable for the construction of an efficient algorithm to compute all space-time derivatives starting from pure space derivatives. In order to evaluate, we express nonlinear terms, e.g. in the conservative variables in terms of previously computed space-time derivatives. A tool to achieve this is the generalized Leibniz rule

$$\begin{aligned}&\frac{\partial^{m+p+o}}{\partial x^m x_2{}^p t^o}(fg)\\ &= \sum_{j=0}^{m}\sum_{l=0}^{p}\sum_{k=0}^{o}\begin{pmatrix} m \\ j \end{pmatrix}\begin{pmatrix} p \\ l \end{pmatrix}\begin{pmatrix} o \\ k \end{pmatrix}\frac{\partial^{m-j+p-l+o-k}}{\partial x^{m-j} y^{p-l} t^{o-k}}(f)\frac{\partial^{j+l+k}}{\partial x^j y^l t^k}(g),\end{aligned} \tag{A.3}$$

where $f = f(\vec{x}, t)$ and $g = g(\vec{x}, t)$ are sufficiently regular functions. The differentiation of a quotient of two functions is avoided by the introduction of the auxiliary variables and the application of the generalized Leibniz rule in the form

$$\begin{aligned}&\frac{\partial^{m+p+o}}{\partial x^m y^p t^o}(\frac{f}{g}) = \frac{\partial^{m+p+o}}{\partial x^m y^p t^o}(f)\\ &-\frac{1}{g}\underbrace{\sum_{j=0}^{m}\sum_{l=0}^{p}\sum_{k=0}^{o}}_{j+l+k\neq m+p+o}\begin{pmatrix} m \\ j \end{pmatrix}\begin{pmatrix} p \\ l \end{pmatrix}\begin{pmatrix} o \\ k \end{pmatrix}\frac{\partial^{m-j+p-l+o-k}}{\partial x^{m-j} y^{p-l} t^{o-k}}(g)\frac{\partial^{j+l+k}}{\partial x^j y^l t^k}(\frac{f}{g}).\end{aligned} \tag{A.4}$$

The CK-algorithm consists of an exterior loop, which begins from $o = 1$ up to $N$, where $N$ is the polynomial degree of the trial function $U_i$. Two inner loops begin from $m = 1$ up to $N - o$, and $p = 1$ up to $N - o - m$, respectively. All space-time derivatives of $U$ are then computed using equation (A.2) by applying equation (A.3). The derivatives of the auxiliary variables have to be computed in advance using (A.3) and (A.4) and have to be stored during the procedure.

## A.1. Information for building a 2D Cauchy-Kovalevskaya procedure for viscous MHD

The following conservative (C) and auxiliary (A) variables are used to compute the 2D Cauchy-Kovalevskaya procedure:

$$
\begin{array}{lcllcllcl}
C_1 & = & \rho & H_9 & = & u & & & \\
C_2 & = & \rho u & H_{10} & = & v & H_{21} & = & u\left(\rho u^2 + \rho v^2 + \rho w^2\right) \\
C_3 & = & \rho v & H_{11} & = & w & H_{22} & = & v\left(\rho u^2 + \rho v^2 + \rho w^2\right) \\
C_4 & = & \rho w & H_{12} & = & \rho u^2 & H_{24} & = & e \\
C_5 & = & \rho e & H_{13} & = & \rho uv & H_{25} & = & u^2 \\
C_6 & = & Bx & H_{14} & = & \rho uw & H_{26} & = & v^2 \\
C_7 & = & By & H_{15} & = & \rho v^2 & H_{27} & = & w^2 \\
C_8 & = & Bz & H_{16} & = & \rho vw & H_{28} & = & \mu \frac{\kappa}{Pr}\left(e - \frac{1}{2}\left(u^2 + v^2 + w^2\right)\right) \\
 & & & H_{17} & = & \rho w^2 & H_{29} & = & -\frac{2}{3}uv_y + vu_y \\
 & & & H_{18} & = & \rho eu & H_{30} & = & -\frac{2}{3}vu_x + uv_x \\
 & & & H_{19} & = & \rho ev & & & \\
 & & & H_{20} & = & \rho ew & & &
\end{array}
$$

$$
\begin{aligned}
H_{32} &= Bx^2 \\
H_{33} &= By^2 \\
H_{34} &= Bz^2 \\
H_{35} &= uBx \\
H_{36} &= uBy \\
H_{37} &= uBz \\
H_{38} &= vBx \\
H_{39} &= vBy \\
H_{40} &= vBz \\
H_{41} &= wBx \\
H_{42} &= wBy \\
H_{43} &= wBz
\end{aligned}
\qquad
\begin{aligned}
H_{44} &= BxBy \\
H_{45} &= BxBz \\
H_{46} &= ByBz \\
H_{47} &= u\left(Bx^2 + By^2 + Bz^2\right) \\
H_{48} &= v\left(Bx^2 + By^2 + Bz^2\right) \\
H_{49} &= Bx\left(uBx + vBy + wBz\right) \\
H_{50} &= By\left(uBx + vBy + wBz\right) \\
H_{51} &= By\left(By_x - Bx_y\right) + Bz\left(Bz_x\right) \\
H_{52} &= Bx\left(Bx_y - By_x\right) + Bz\left(Bz_y\right)
\end{aligned}
$$

## A.2. Information for building a 3D CK procedure for viscous MHD

In three space dimension, the viscous MHD equations without any normalization may be written in a conservation form as:

$$
\begin{aligned}
0 = &\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \\ Bx \\ By \\ Bz \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p + \frac{\mathbf{B}^2}{2} - Bx^2 \\ \rho uv - BxBy \\ \rho uw - BxBz \\ u\left(\rho e + p + \frac{\mathbf{B}^2}{2}\right) - Bx\left(\mathbf{v}\cdot\mathbf{B}\right) \\ 0 \\ uBy - vBx \\ uBz - wBx \end{pmatrix}_x + \\
&\begin{pmatrix} \rho v \\ \rho uv - BxBy \\ \rho v^2 + p + \frac{\mathbf{B}^2}{2} - By^2 \\ \rho vw - ByBz \\ v\left(\rho e + p + \frac{\mathbf{B}^2}{2}\right) - By\left(\mathbf{v}\cdot\mathbf{B}\right) \\ vBx - uBy \\ 0 \\ vBz - wBy \end{pmatrix}_y + \begin{pmatrix} \rho w \\ \rho uw - BxBz \\ \rho vw - ByBz \\ \rho w^2 + p + \frac{\mathbf{B}^2}{2} - Bz^2 \\ w\left(\rho e + p + \frac{\mathbf{B}^2}{2}\right) - Bz\left(\mathbf{v}\cdot\mathbf{B}\right) \\ wBx - uBz \\ wBy - vBz \\ 0 \end{pmatrix}_z - \\
&\begin{pmatrix} 0 \\ \frac{4}{3}\mu u_x - \frac{2}{3}\mu\left(v_y + w_z\right) \\ \mu\left(u_y + v_x\right) \\ \mu\left(u_z + w_x\right) \\ C_1 \\ 0 \\ \eta\left(By_x - Bx_y\right) \\ \eta\left(Bz_x - Bx_z\right) \end{pmatrix}_x - \begin{pmatrix} 0 \\ \mu\left(u_y + v_x\right) \\ \frac{4}{3}\mu v_y - \frac{2}{3}\mu\left(u_x + w_z\right) \\ \mu\left(v_z + w_y\right) \\ C_2 \\ \eta \\ 0 \\ \eta\left(Bz_y - By_z\right) \end{pmatrix}_y - \\
&\begin{pmatrix} 0 \\ \mu\left(u_z + w_x\right) \\ \mu\left(v_z + w_y\right) \\ \frac{4}{3}\mu w_z - \frac{2}{3}\mu\left(u_x + v_y\right) \\ C_3 \\ \eta\left(Bx_z - Bz_x\right) \\ \eta\left(By_z - Bz_y\right) \\ 0 \end{pmatrix}_z ,
\end{aligned}
\tag{A.5}
$$

with $C_1, C_2, C_3$ being

$$\begin{aligned}
C_1 = \;& \tfrac{4}{3}\mu u u_x - \tfrac{2}{3}\mu u\left(v_y + w_z\right) + \mu v\left(v_x + u_y\right) + \mu w\left(w_x + u_z\right) + kT_x \\
& + \eta By\left(By_x - Bx_y\right) + \eta Bz\left(Bz_x - Bx_z\right), \\
C_2 = \;& \tfrac{4}{3}\mu v v_y - \tfrac{2}{3}\mu v\left(u_x + w_z\right) + \mu u\left(v_x + u_y\right) + \mu w\left(w_y + v_z\right) + kT_y \\
& + \eta Bx\left(Bx_y - By_x\right) + \eta Bz\left(Bz_y - By_z\right), \\
C_3 = \;& \tfrac{4}{3}\mu w w_z - \tfrac{2}{3}\mu w\left(u_x + v_y\right) + \mu u\left(u_z + w_x\right) + \mu v\left(v_z + w_y\right) + kT_y \\
& + \eta Bx\left(Bx_z - Bz_x\right) + \eta By\left(By_z - Bz_y\right).
\end{aligned} \tag{A.6}$$

The temperature $T$ can be derived from the following equations:

$$T = \frac{p}{\rho R} = \frac{1}{c_v}\epsilon, \tag{A.7}$$

where

$$c_v = \frac{c_p}{\kappa} = \frac{Prk}{\kappa\mu}. \tag{A.8}$$

Hence, we will obtain the temperature by

$$T = \frac{\kappa}{c_v k}\left(e - \frac{1}{2}\left(u^2 + v^2 + w^2\right)\right) = \frac{\mu\kappa}{Prk}\left(e - \frac{1}{2}\left(u^2 + v^2 + w^2\right)\right), \tag{A.9}$$

where

$$k = \frac{c_p \mu}{Pr} \text{ and } c_p = \kappa c_v. \tag{A.10}$$

The following conservative (C) and auxiliary (A) variables are used to compute the Cauchy-Kovalevskaya procedure:

$$\begin{array}{lcllcllcl}
C_1 & = & \rho & A_9 & = & u & A_{21} & = & u\left(\rho u^2 + \rho v^2 + \rho w^2\right) \\
C_2 & = & \rho u & A_{10} & = & v & A_{22} & = & v\left(\rho u^2 + \rho v^2 + \rho w^2\right) \\
C_3 & = & \rho v & A_{11} & = & w & A_{23} & = & w\left(\rho u^2 + \rho v^2 + \rho w^2\right) \\
C_4 & = & \rho w & A_{12} & = & \rho u^2 & A_{24} & = & e \\
C_5 & = & \rho e & A_{13} & = & \rho u v & A_{25} & = & u^2 \\
C_6 & = & Bx & A_{14} & = & \rho u w & A_{26} & = & v^2 \\
C_7 & = & By & A_{15} & = & \rho v^2 & A_{27} & = & w^2 \\
C_8 & = & Bz & A_{16} & = & \rho v w & A_{28} & = & \mu\frac{\kappa}{Pr}\left(e - \frac{1}{2}\left(u^2 + v^2 + w^2\right)\right) \\
 & & & A_{17} & = & \rho w^2 & A_{29} & = & -\frac{2}{3}u\left(v_y + w_z\right) + v u_y + w u_z \\
 & & & A_{18} & = & \rho e u & A_{30} & = & -\frac{2}{3}v\left(u_x + w_z\right) + u v_x + w v_z \\
 & & & A_{19} & = & \rho e v & A_{31} & = & -\frac{2}{3}w\left(u_x + v_y\right) + u w_x + v w_y \\
 & & & A_{20} & = & \rho e w & & &
\end{array}$$

$$
\begin{aligned}
A_{32} &= Bx^2 & A_{44} &= BxBy \\
A_{33} &= By^2 & A_{45} &= BxBz \\
A_{34} &= Bz^2 & A_{46} &= ByBz \\
A_{35} &= uBx & A_{47} &= u\left(Bx^2 + By^2 + Bz^2\right) \\
A_{36} &= uBy & A_{48} &= v\left(Bx^2 + By^2 + Bz^2\right) \\
A_{37} &= uBz & A_{49} &= w\left(Bx^2 + By^2 + Bz^2\right) \\
A_{38} &= vBx & A_{50} &= Bx\left(uBx + vBy + wBz\right) \\
A_{39} &= vBy & A_{51} &= By\left(uBx + vBy + wBz\right) \\
A_{40} &= vBz & A_{52} &= Bz\left(uBx + vBy + wBz\right) \\
A_{41} &= wBx & A_{53} &= By\left(By_x - Bx_y\right) + Bz\left(Bz_x - Bx_z\right) \\
A_{42} &= wBy & A_{54} &= Bx\left(Bx_y - By_x\right) + Bz\left(Bz_y - By_z\right) \\
A_{43} &= wBz & A_{55} &= Bx\left(Bx_z - Bz_x\right) + By\left(By_z - Bz_y\right)
\end{aligned}
$$

# B. Analytical Solutions for Convergence Tests

In order to perform convergence studies, we need to compare the calculated solution to a known exact solution after some calculation time. Unfortunately, an exact solution is usually not known, especially when it comes to three space dimensions. We therefore use a method called "manufactured solution" to build an exact solution as done similarly in [15] for the Navier-Stokes equations. We hereby choose an arbitrary smooth state and insert it into the equation system. Since this state usually is not an exact solution, we end up with a right hand side, which we then put as a source term into our numerical scheme.
We are choosing, in primitive variables,

$$U = (sin(\beta) + C, 1, 1, 0, sin(\beta) + C, sin(\beta) + C, -sin(\beta) - C, 0)\,, \tag{B.1}$$

with $\beta := \sum_{j=1}^{2} x_j\omega - at$ for the two-dimensional viscous MHD equations and get a source term $S$

$$S = \begin{pmatrix} cos(\beta)(-a+2\omega) \\ (-a+3\omega+2\omega C)cos(\beta) + \omega sin(2\beta) \\ (-a+3\omega+2\omega C)cos(\beta) + \omega sin(2\beta) \\ 0 \\ \frac{\eta}{\kappa-1}((4\omega^2 C\kappa - 4\omega^2 C)sin(\beta) + \\ \quad (-4\omega^2\kappa + 4\omega^2)cos(2\beta)) + \\ \quad \frac{1}{\kappa-1}((-4\omega - a\kappa + a + 4\omega\kappa)sin(2\beta) + \\ \quad (-2\omega + 8\omega C\kappa + 4\omega\kappa - 2aC\kappa + 2aC - a\kappa - 8\omega C)cos(\beta)) \\ cos(\beta)(-a+2\omega) + 2\eta sin(\beta)\omega^2 \\ cos(\beta)(a-2\omega) - 2\eta sin(\beta)\omega^2 \\ 0 \end{pmatrix}, \tag{B.2}$$

with

$$\begin{aligned} A &= -\omega + \tfrac{k}{d-1}\left((-1)^{d-1} + \kappa\,(2\,d-1)\right) \\ B &= \tfrac{1}{2}\left((d^2 + \kappa(6+3\,d))\,k - 8\,\omega\right). \end{aligned} \tag{B.3}$$

Hereby, $\gamma$ $\omega$ and $C$ can be arbitrarily chosen to fit the test case into spacial and temporal resolution. In addition, the assumption of $Pr$, $\mu$ and $\eta$ being constant was made.

# Bibliography


[1] D.S. Balsara. Second-Order Accurate Schemes for Magnetohydrodynamics with Divergence-Free Reconstruction. *The Astrophysical Journal Supplement Series*, 151:149–184, 2004.

[2] D.S. Balsara, C. Altmann, C.-D. Munz, and M. Dumbser. A sub-cell based indicator for troubled zones in RKDG schemes and a novel class of hybrid RKDG+HWENO schemes. *J. Comput. Phys.*, 226:586–620, 2008.

[3] D.S. Balsara and D.S. Spicer. A staggered mesh algorithm using high order Godunov fluxes to ensure solenodial magnetic fields in magnetohydrodynamic simulations. *J. Comput. Phys.*, 149(2):270–292, 1999.

[4] J.U. Brackbill and D.C. Barnes. The effect of nonzero $\nabla\cdot$B on the numerical solution of the magnetohydrodynamic equations. *J. Comput. Phys.*, 35:426–430, 1980.

[5] A.J. Chorin. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull. Amer. Math. Soc.*, 73:928–931, 1967.

[6] B. Cockburn, S. Hou, and C.-W. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case. *Math. Comput.*, 54:545–581, 1990.

[7] B. Cockburn, S. Y. Lin, and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One dimensional systems. *J. Comput. Phys.*, 84:90–113, 1989.

[8] B. Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework. *Math. Comput.*, 52:411–435, 1989.

[9] B. Cockburn and C.-W. Shu. The Runge-Kutta local projection $p^1$-discontinuous Galerkin method for scalar conservation laws. $M^2AN$, 25:337–361, 1991.

[10] B. Cockburn and C.-W. Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws V: Multidimensional systems. *J. Comput. Phys.*, 141:199–224, 1998.

[11] R. B. Dahlburg and J. M. Picone. Evolution of the Orszag-Tang vortex system in a compressible medium. I. Initial average subsonic flow. *Phys. Fluids B*, 1:2153–2171, 1989.

[12] A. Dedner, F. Kemm, D. Kröner, C.-D. Munz, T. Schnitzer, and M. Wesenberg. Hyperbolic divergence cleaning for the MHD equations. *J. Comput. Phys.*, 175(2):645–673, 2002.

[13] M. Dumbser, M. Käser, V. A. Titarev, and E. F. Toro. Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems. *J. Comput. Phys*, 226:204–243, 2007.

[14] C.R. Evans and J.F. Hawley. Simulation of magnetohydrodynamic flow: A constrained transport method. *The Astrophysical Journal*, 332(2, Part 1):659–677, 1988.

[15] G. Gassner. *Discontinuous Galerkin Methods for the Unsteady Compressible Navier-Stokes Equations*. PhD thesis, Universität Stuttgart, 2009.

[16] G. Gassner, M. Dumbser, F. Hindenlang, and C.-D. Munz. Explicit one-step time discretizations for discontinuous Galerkin and finite volume schemes based on local predictors. *J. Comput. Phys.*, 230(11):4232–4247, 2011.

[17] G. Gassner, F. Lörcher, and C.-D. Munz. A Contribution to the Construction of Diffusion Fluxes for Finite Volume and Discontinuous Galerkin Schemes. *J. Comput. Phys.*, 224(2):1049–1063, 2007.

[18] G. Gassner, F. Lörcher, and C.-D. Munz. A discontinuous Galerkin scheme based on a space-time expansion. II. Viscous flow equations in multi dimensions. *J. Sci. Comp.*, 34(3):260–286, 2008.

[19] G. J. Gassner, F. Lörcher, C.-D. Munz, and J. S. Hesthaven. Polymorphic nodal elements and their application in discontinuous Galerkin methods. *J. Comput. Phys.*, 228(5):1573–1590, 2009.

[20] J.W. Gibbs. Fourier Series. *Nature*, 59(200 and 606), 1899.

[21] J.-L. Guermond, R. Pasquetti, and B. Popov. Entropy viscosity method for nonlinear conservation laws. *J. Comput. Phys.*, 230(11):4248–4267, 2011.

[22] A. Harten. High Resolution Schemes for Hyperbolic Conservation Laws. *J. Comput. Phys.*, 135(2):260–278, 1997.

[23] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy. Uniformly high order essentially non-oscillatory schemes, III. *J. Comput. Phys.*, 71:231–303, 1987.

[24] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications.* Springer Verlag, New York, 2008.

[25] A. Huerta, E. Casoni, and J. Peraire. A simple shock-capturing technique for high-order Discontinuous Galerkin methods. *Int. J. Numer. Methods Fluids*, 69(10):1614–1632, 2012.

[26] J. Jaffre, C. Johnson, and A. Szepessy. Convergence of the discontinuous Galerkin finite element method for hyperbolic conservation laws. *Math. Mod. Meth. Appl. Sci.*, 182:546–585, 1995.

[27] A. Jameson, W. Schmidt, and E. Turkel. Numerical Solution of the Euler equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes. In *Proc. of the AIAA 14th Fluid and Plasma Dynamic Conference*, AIAA-1981-1259, Paolo Alto, California, USA, 1981.

[28] P. Janhunen. A Positive Conservative Method for Magnetohydrodynamics Based on HLL and Roe Methods. *J. Comput. Phys.*, 160(2):649–661, 2000.

[29] A. Kanevsky, M.H. Carpenter, D. Gottlieb, and J.S. Hesthaven. Application of implicit-explicit high-order Runge-Kutta methods to discontinuous Galerkin schemes. *J. Comput. Phys.*, 225(2):1753–1781, 2007.

[30] G. Karypis and V. Kumar. A Parallel Algorithm for Multilevel Graph Partitioning and Sparse Matrix Ordering. *Journal of Parallel and Distributed Computing*, 48:71–85, 1998.

[31] M. Käser and M. Dumbser. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes - I. The two-dimensional isotropic case with external source terms. *Geophysical Journal International*, 166(2):855–877, 2006.

[32] D.A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations*. Springer, 2009.

[33] L. Krivodonova. Limiters for high-order discontinuous Galerkin methods. *J. Comput. Phys*, 226(1):879–896, 2007.

[34] F. Li and C.-W. Shu. Locally Divergence-Free Discontinuous Galerkin Methods for MHD Equations. *J. Sci. Comp.*, 22-23(1):413–442, 2005.

[35] S. Li. An HLLC Riemann solver for magneto-hydrodynamics. *J. Comput. Phys.*, 203(1):344–357, 2005.

[36] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially nonoscillatory schemes. *J. Comput. Phys.*, 115, 1994.

[37] F. Lörcher, G. Gassner, and C.-D. Munz. A discontinuous Galerkin scheme based on a space-time expansion. I. Inviscid compressible flow in one space dimension. *J. Sci. Comp.*, 32(2):175–199, 2007.

[38] S. A. Orszag and C. M. Tang. Small-scale structure of two-dimensional magnetohydrodynamic turbulence. *Journal of Fluid Mechanics*, pages 90–129, 1979.

[39] B. Owren and M. Zennaro. Order barriers for continuous explicit Runge-Kutta methods. *Math. Comp.*, 56:645–661, 1991.

[40] B. Owren and M. Zennaro. Derivation of efficient continuous explicit Runge-Kutta methods. *SIAM J. Sci. Stat. Comput.*, 13:1488–1501, 1992.

[41] P.-O. Persson and J. Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, January 2006.

[42] J. M. Picone and R. B. Dahlburg. Evolution of the Orszag-Tang vortex system in a compressible medium. II. Supersonic flow. *Phys. Fluids B*, 3:29–44, 1991.

[43] K.G. Powell, P.L. Roe, T.J. Linde, T.I. Gombosi, and D.L. De Zeeuw. A solution-adaptive upwind scheme for ideal magnetohydrodynamics. *J. Comput. Phys.*, 154(2):284–309, 1999.

[44] J. Qiu. A Numerical Comparison of the Lax-Wendroff Discontinuous Galerkin Method Based on Different Numerical Fluxes. *J. Sci. Comp.*, 30:345–367, 2007.

[45] J. Qiu, B. C. Khoo, and C.-W. Shu. A Numerical Study for the Performance of the Runge-Kutta Discontinuous Galerkin Method Based on Different Numerical Fluxes. *J. Comput. Phys.*, 212:540 – 565, 2006.

[46] P.L. Roe. Characteristic-based schemes for the Euler equations. *Annual Review of Fluid Mechanics*, 1:337–365, 1986.

[47] D. Ryu, F. Miniati, T. W. Jones, and A. Frank. A Divergence-free Upwind Code for Multidimensional Magnetohydrodynamic Flows. *ApJ*, 509:244–255, 1998.

[48] C.W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes II. *J. Comput. Phys.*, 83:32–78, 1989.

[49] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, June 1999.

[50] M. Torrilhon. Non-uniform convergence of finite volume schemes for Riemann problems of ideal magnetohydrodynamics. *J. Comput. Phys.*, 192:73–94, November 2003.

[51] M. Torrilhon. *Zur Numerik der idealen Magnetohydrodynamik*. PhD thesis, ETH Zürich, 2003.

[52] M. Torrilhon. Locally divergence-preserving upwind finite volume schemes for magnetohydrodynamic equations. *SIAM J. Sci. Comput.*, 26(4):1166–1191, 2005.

[53] J. Utzmann. *A domain decomposition method for the efficient direct simulation of aeroacoustic problems*. PhD thesis, Universität Stuttgart, 2008.

[54] T. C. Warburton and G. E. Karniadakis. A Discontinuous Galerkin Method for the Viscous MHD Equations. *J. Comput. Phys.*, 152(2):608–641, 1999.

[55] T. C. Warburton and G. E. Karniadakis. A discontinuous Galerkin method for the viscous MHD equations. *J. Comput. Phys.*, 152(2):608–641, 1999.

[56] P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *J. Comput. Phys.*, 54(1):115–173, 1984.

[57] H. C. Yee, N. D. Sandham, and M. J. Djomehri. Low-Dissipative High-Order Shock-Capturing Methods Using Characteristic-Based Filters. *J. Comput. Phys*, 150(1):199–238, 1999.

[58] H. C. Yee and B. Sjögreen. Non-Linear Filtering and Limiting in High Order Methods for Ideal and Non-Ideal MHD. *Journal of Scientific Computing*, 27(1):507–521, 2006.

# List of Tables

# List of Figures

# Lebenslauf

| | |
|---|---|
| 14.03.1980 | Geboren in Nürnberg |
| 1986 - 1990 | Grundschule, Zirndorf |
| 1990 - 1999 | Heinrich-Schliemann-Gymnasium, Fürth |
| 1999 | Allgemeine Hochschulreife |
| 2000 - 2006 | Studium der Luft- und Raumfahrttechnik an der Universität Stuttgart, Vertiefungsrichtungen: Strömungslehre, Regelungstechnik |
| 2006 | Diplomarbeit an der University of Notre Dame, USA: *HWENO Limiting and Divergence Cleaning for Euler- and Magnetohydrodynamic Equations* |
| 2007 - 2011 | Wissenschaftlicher Mitarbeiter am Institut für Aerodynamik und Gasdynamik der Universität Stuttgart<br>Dissertation: *Explicit Discontinuous Galerkin Methods for Magnetohydrodynamics* |
| seit 01.10.2011 | Beschäftigt bei der Robert Bosch GmbH im Bereich High Performance Computing |

Stuttgart, 14.09.2012 Christoph Altmann