University of West Bohemia in Pilsen
Faculty of Electrical Engineering

Doctoral thesis

Large-scale Numerical Simulations of Magneto-Hydrodynamics Phenomena in Astrophysics

Mgr. Lukáš Korous

TODO

Check ze mam CFL podminky a flux dobre - vlastni cisla nejsou s hybnosti, ale s rychlosti

$[Vh]^8$ tak neni kdyz mam ten spesl prostor

smazal jsem linearizaci, jeste smazat vsechny zminky o implicitni

doplnit citace na Trilinos, P4EST, Intel Parallel Studio

Mam nejak divne rovnice pro energii - $varepsilon_{ijk}$ je divny, a taky poresit jestli je $p + U_m s\frac{1}{2}$ predtim nebo ne

- popsat numerical flux, ze je non-differentiable, ze se vyhodnocuje na hranach a ze to je challenging z pohledu adaptivity, paralelismu a periodickych podminek
- obecne MHD num fluxy, kolik je tam vlastnich cisel, cili kolik mezistavu, jake jsou

ruzne toky a proc se nepouziva presny resic
- HLLD, presny popis, proc ne ten flux s jeste vice stavy (drahy)

do uvodu napsat vice o state-of-the-art, odkaz na Mirovo clanek, ukazat tam ty FD vysledky, ze je chceme nahradit a ze je tam hodne dulezita adaptivity
pak teda popsat az tam z toho budu davat vysledky, tak vykopirovat nejaky rovnice IC / BC z clanku, jak se to implementuje, atd.
srovnani

adaptivita - ukazat na 2d prikladu z Hermesu co to je referencni reseni, co to je ||zprojektovane - presne|| (v dealu?), co to je tohle bez normy, zminka o distribuovanosti, ze musim napocitat nejaky thresholdy mapReducem, atd.
napsat pak algoritmus tedy cely, kde je casovy krok, adaptivita (ze se vraci casovy krok po zjemneni / zhrubeni, aby se neztracela informace), atd.

popis a vysledky Blastu, O-T
- odkazy na clanky, popis
- bez adaptivity jen "pocita to"
- s adaptivitou "a navic tak presne to spoctu a usetrim 80% dofu"
– tohle ukazovat na rezech hustotou - bylo by fajn nejaky najit

Abstract

The objective of this Doctoral Thesis was to develop, implement and test new algorithms for the large-scale solution of nonstationary compressible MHD equations based on higher-order discontinuous Galerkin (DG) methods. The basis for the new methods will be the discontinuous Galerkin methods and adaptive mesh refinement (AMR) algorithms. The new algorithms will be implemented and tested in the framework of the open source library deal.II, and they will be applied to selected problems of MHD in astrophysics, namely the magnetic reconnection phenomena.

Keywords

numerical simulation, finite element method, MHD equations, adaptivity, discontinuous Galerkin method, astrophysics, solar flares, CME, magnetic reconnection

Abstract [CZ]

Záměrem této práce je navrhnout, implementovat a otestovat nové algoritmy pro rozsáhlé simulace nestacionárních jevů spadajících do oblasti stlačitelné magnetohydrodynamiky. Vytvořený software bude založen na použití nespojité Galerkinovy metody (discontinuous Galerkin, DG) s vyššími řády přesnosti. Zároveň bude použita metoda automatického zjemňování výpočetní triangulace (automatic mesh refinement, AMR). Vytvořené algoritmy budou testovány ve frameworku deal.II a budou aplikovány na skutečné problémy v astrofyzice, jmenovitě na simulaci jevu magnetické rekonexe.

Keywords [CZ]

numerická simulace, metoda konečných prvků, MHD rovnice, adaptivní algoritmy, nespojitá Galerkinova metoda, astrofyzika, sluneční erupce, CME, magnetická rekonexe

Acknowledgement

I hereby acknowledge and thank for the lead and support of my supervisor, doc. Ing. Pavel Karban, Ph.D., as well as my advisor prof. Ing. Ivo Doležel, CSc. I would also like to thank Ing. Jan Skála, Ph.D. to give me valuable insight into the problems of astrophysics, and I also hereby appreciate the work of the entire deal.II development team for their professional approach to software development and support.

Statement

I am presenting this doctoral thesis, created during my doctoral studies at the Faculty of Electrical Engineering of the University of West Bohemia.

I confirm having prepared this work by my own, and having listed all used sources of information in the bibliography. All license conditions of all works of software and other nature were respected.

In Pilsen, June 16, 2018, Lukáš Korous

Contents

Notation

Symbol	Meaning
a	a scalar quantity "a"
$\boldsymbol{a}$	a 3-element vector "a"
a_i	the i-th component of a vector $\boldsymbol{a}$
$\mathbf{A}$	an 8-element vector "A"
ρ	Density
$\boldsymbol{\pi}$	Momentum
μ	Permeability
μ_0	Permeability of vacuum
t	Time variable
$\boldsymbol{x}$	Space variable
p	Pressure
$\boldsymbol{u}$	Velocity
u	Magnitude of velocity, i.e. $\lvert\boldsymbol{u}\rvert$
$\boldsymbol{B}$	Magnetic flux density
$\boldsymbol{J}$	Current density
B	Magnitude of magnetic flux density, i.e. $\lvert\boldsymbol{B}\rvert$
$\boldsymbol{E}$	Electric field
e	Internal energy density
c	Speed of light
$\boldsymbol{g}$	Gravitational acceleration
σ	Conductivity, $\sigma = \frac{1}{\eta}$
η	Electrical restisistivity, $\eta = \frac{1}{\sigma}$
c_v	Specific heat at constant volume
c_p	Specific heat at constant pressure
γ	Poisson adiabatic constant
θ	Absolute temperature
$\boldsymbol{q}$	Heat flux
q	Electric charge
$\boldsymbol{f}$	Density of force acting on fluid
$\boldsymbol{F}_L$	Lorentz force, $\boldsymbol{F}_L = q\left(\boldsymbol{E} + \boldsymbol{u} \times \boldsymbol{B}\right)$
$\boldsymbol{f}_L$	Lorentz force density, $\boldsymbol{f}_L = \rho_q \boldsymbol{E} + \boldsymbol{J} \times \boldsymbol{B}$
ε	Electrical permittivity
ε_0	Electrical permittivity of vacuum
ρ_q	Charge density
U_k	Kinetic energy, $U_k = \rho\frac{u^2}{2}$
U_m	Magnetic energy, $U_m = \frac{B^2}{2\mu_0}$
U	Total energy, $U = U_k + U_m + \rho e$
$\tilde{U}$	Hydrodynamic energy, $\tilde{U} = U - U_m$
$\boldsymbol{n}$	Unit outer normal

Table 0.1: Notation

1 Introduction

The term magnetohydrodynamics (MHD) covers all physical phenomena that involve both electromagnetic (EM) field and a fluid that carries the EM field. Such phenomena are very interesting, yet very complex to study. The behavior of such a fluid is utilized in some industrial applications - liquid-metal cooling of nuclear reactors, magnetic fluid in dampers, sensors for precise measuring of angular velocities, etc. Such phenomena occur in nature as well - the most significant of which are definitely the processes that take place inside and on the surface of stars - which is the topic of the next section.

1.1 Magnetohydrodynamics in Astrophysics

There are several phenomena in the universe that we can look at as magnetohydrodynamic in nature - planets consisting of metals, interplanetary space, but mainly - stars. If we talk about the nearest star - and the only one we are able to study well enough - the Sun - these phenomena include those that occur in the Sun's photosphere (the layer of Sun that is visible): Sun spots, but also phenomena that occur above the Sun (further from the center of the Sun): in Sun's chromosphere, or even corona (solar flares) - even phenomena that originate from the Sun, but then spread through our solar system - solar winds, space weather. All these phenomena have a large impact on the lives of all of us. For example solar flares (that are often followed by ejection of mass out of the Sun - the so called coronal mass ejections - CMEs) have impact on the Earth's magnetic field which in turn has impact on the electronic communication down on Earth (because the communication satellites used for transmissions may be damaged by the disturbances in the magnetic field). Also people operating at high altitudes, both in airplanes and manned space missions are exposed to the energetic particles coming from the Sun (this term is sometimes called *cosmic rays*). For all the above reasons, it is of great importance to understand the phenomena of space weather, and other MHD phenomena that occur in space.

1.1.1 Magnetic reconnection

Magnetic reconnection occurs within electrically charged gases called plasmas. These charged particles interact strongly with the magnetic field, but at the same time their motions modify the magnetic field. Plasmas behave unlike what we regularly experience on Earth because they travel with their own set of magnetic fields entrapped in the material. Changing magnetic field affects the way charged particles

move and vice-versa, so the net effect is a complex, constantly-adjusting system that is sensitive to minute variations.

Under normal conditions, the magnetic field lines inside plasmas don't break or merge with other field lines. But sometimes, as field lines get close to each other, the entire pattern changes and everything realign into a new configuration. The amount of energy released can be formidable. Magnetic reconnection taps into the stored energy of the magnetic field, converting it into heat and kinetic energy that sends particles streaming out along the field lines.

Solar flares, which are among the phenomena which are the most important to study, are driven by magnetic reconnection - and thus studying magnetic reconnection is of great importance.

1.1.2 Magnetic flux rope model of plasma

We are interested in the process of evolution of the flux rope eruption. For this, we utilize the magnetic field model by Titov and Demoulin (TODO citace) which describes a twisted flux rope as part of a torus with minor and major radii a and R, and winding number N_t. The magnetic configuration is kept in a global equilibrium by the action of the Lorentz force due to the overlying magnetic field. The sources of this ambient field are modeled by a subphotospheric line current I_0 and a pair of magnetic charges $+q, -q$, all located at the major axis of the torus.

1.2 State of the art

Only recently, the scientific computation community, due to the advances in computer and supercomputer capabilities, has started with non-trivial numerical simulations of such complex physical phenomena that the MHD model describes. Since both for industrial applications, and obviously for astrophysical application of the MHD model, it is quite expensive (or downright impossible) to perform any experiments, the benefit of being able to simulate the phenomena on a computer is very large.
There exist several available numerical simulation codes, such as [15], [9], [4], [13]. These codes have been successfully applied to a range of problems in astrophysics. There are many numerical methods implemented in these codes, such as the finite difference method ([13]), finite volume method ([4]), and the (continous) finite element method ([12]).

There have been some attempts to employ also the discontinuous Galerkin method ([11], [8]), but so far no open-source generic software employing this method is available. Moreover, as the astrophysical interest lies in multi-scale problem, a software that can handle such problems would be much more beneficial. An approach that can achieve this capability of solving multi-scale problems is the Adaptive Mesh Refinement technique (AMR). What we understand under this term is not only a mesh refinement that is local (i.e. non-uniform), but a mesh refinement that does

not originate in the problem description, and neither is invoked programatically with user input. The term 'adaptivity' means that through a predefined *refinement indicator*, which is a function operating on the set of elements of the triangulation, elements to be refined are chosen automatically. This *refinement indicator* is calculated from the solution, and in effect makes the triangulation 'adapt' to the solution - hence, AMR.

The reason for the development of a new code is two-fold. First, there is a unique collaboration between the Astronomical Institute of the Czech Academy of Sciences and the University of West Bohemia, where astrophysicists work together with electrical engineers (from theoretical and numerical modeling backgrounds), and the developed code will be usable for both simulating of astrophysical MHD phenomena, and industrial MHD applications.

Second, the newly developed code is based on locally-adaptive Discontinuous Galerkin method, which yields several advantages over the existing codes (which use e.g. finite difference, or finite volumes methods) developed at institutions of such high quality as *Princeton* - [15], [9]. The advantages are especially of performance, and automation nature - method of higher order together with AMR yields results qualitatively and quantitatively comparable to low order uniform mesh methods, but with computational cost that can easily be an order of magnitude smaller. Automation is mentioned here related to the AMR, which, without user interaction, can optimize the computational triangulation for a particular time instance in the evolution of the modeled phenomena.

Another benefit (namely over [9])of the newly created software are the use of modern object-oriented programming techniques and experience gained on creating finite element software ([14], [6], [5]). The implementation related to this work is written in the C++ language, with the use of existing software packages that are proven, and used by a wide community of researchers all over the world - deal.II ([1]), Trilinos, P4EST, Intel Parallel Studio, UMFPACK ([3]), Paraview(`http://www.paraview.org/`), and others.

The state of the art of numerical simulation of magnetohydrodynamics can be summarized as a state when the mathematical theory of the equations is quite solid, but the methods to solve the equations numerically in the most optimal and fast way are still being improved. The numerical solution is not merely about theoretical convergence rates and attributes of the particular method, but also the actual implementation plays an important role - i.e. programming, hardware, and software, and execution, both before, during, and very importantly after the actual method invocation (of the so-called *postprocessing* of results). In all aspects of implementation, there is space for new approaches, new ideas, new milestones, that can expand the capabilities of today's numerical solution of magnetohydrodynamics phenomena.

2 Mathematical model

In this chapter, the mathematical model will be derived from basic equations governing the studied physical phenomena, its weak formulation will be presented, and the complete mathematical problem which is solved will be described.

2.1 Derivation of the mathematical model

TODO - tady neco musi byt, nemuze hned byt subsekce

2.1.1 Initial setup

We consider a time interval $(0, T)$ and space domain $\Omega_t \subset \mathbb{R}^3$ occupied by a fluid at time t. By $\mathcal{M}$ we denote the space-time domain in consideration:

$$\mathcal{M} = \{(\boldsymbol{x}, t)\,; \boldsymbol{x} \in \Omega_t, t \in (0, T)\}\,. \tag{2.1}$$

Moreover we assume that $\mathcal{M}$ is an open set.

Assumptions

When dealing with MHD phenomena in plasma, the following rules apply

- We assume that the fluid is inviscid.
- We assume that the fluid is compressible.
- We consider only the so-called *perfect gas* or *ideal gas* whose state variables satisfy the following *equation of state*

  $$p = R\theta\rho, \tag{2.2}$$

  where ρ denotes density, θ denotes the absolute temperature, and R is the *gas constant*, which is defined as

  $$R = c_p - c_v. \tag{2.3}$$

  In the above, c_p, c_v are specific heats at constant pressure, and at constant volume.

2.1.2 Euler's equations of compressible flow

This system of equation reads

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\boldsymbol{\pi}) &= 0 && (2.4)\\ \frac{\partial \boldsymbol{\pi}}{\partial t} + \nabla \cdot (\boldsymbol{\pi} \otimes \boldsymbol{u}) &= \rho \boldsymbol{f} - \nabla p, && (2.5)\\ \frac{\partial \tilde{U}}{\partial t} + \nabla \cdot \left(\tilde{U}\boldsymbol{u}\right) &= \rho \boldsymbol{f} \cdot \boldsymbol{u} - \nabla \cdot (p\boldsymbol{u}) + \nabla \cdot \boldsymbol{q}, && (2.6) \end{aligned}$$

where $\boldsymbol{\pi}$ is momentum, p pressure, U total energy. Moreover, $\boldsymbol{u}$ denotes velocity, $\boldsymbol{f}$ density of the force acting on the fluid, and $\boldsymbol{q}$ is the heat flux. By $\otimes$, we denote the *tensor product*:

$$\boldsymbol{a} \otimes \boldsymbol{b} = \begin{pmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{pmatrix}.$$

Moreover, the following relations hold:

$$\begin{aligned} \tilde{U} &= \rho e + U_k, && (2.7)\\ p &= (\gamma - 1)\left(\tilde{U} - U_k\right), && (2.8)\\ \theta &= \left(\tilde{U}/\rho - |\boldsymbol{u}|^2/2\right)/c_v. && (2.9) \end{aligned}$$

This system is simply called the *compressible Euler equations* for a heat-conductive perfect gas. The individual equations are called the *continuity equation*(2.4), the *Navier-Stokes equations* (2.5), and the *energy equation* (2.6).

For the force density f we assume that only the Lorentz force and gravity act upon the fluid:

$$\boldsymbol{f} = \boldsymbol{f}_L + \boldsymbol{g}.$$

2.1.3 Maxwell's equations of electromagnetism

In this work, we use Maxwell's equations with the assumption of constant electrical permittivity, and constant permeability. This system of equation reads

$$\begin{aligned} \nabla \times \boldsymbol{B} &= \mu_0 \left(\boldsymbol{J} + \varepsilon_0 \frac{\partial \boldsymbol{E}}{\partial t}\right) && (2.10)\\ \nabla \times \boldsymbol{E} &= -\frac{\partial \boldsymbol{B}}{\partial t} && (2.11)\\ \nabla \cdot \boldsymbol{E} &= \frac{\rho_q}{\varepsilon_0} && (2.12)\\ \nabla \cdot \boldsymbol{B} &= 0, && (2.13) \end{aligned}$$

where $\boldsymbol{B}$ denotes magnetic flux density, $\boldsymbol{E}$ denotes electric field, $\boldsymbol{J}$ denotes current density, ε_0 is permittivity of vacuum, and ρ_q is electric charge density. The individual equations are known as Faraday's law(2.11), Ampere's law(2.10), and Gauss's laws(2.12, 2.13).

2.1.4 Derived relations between electromagnetic quantities

Further relations that are useful when deriving the MHD equations are:

$$\frac{\mathrm{d}}{\mathrm{d}t}U_m = \frac{1}{\mu_0}\nabla\cdot(\boldsymbol{B}\times\boldsymbol{E}) - \boldsymbol{E}\cdot\boldsymbol{J}, \tag{2.14}$$

$$\boldsymbol{E} = -u\times\boldsymbol{B} + \frac{\eta}{\mu_0}\boldsymbol{J}, \tag{2.15}$$

$$\frac{\partial\boldsymbol{B}}{\partial t} + \nabla\times(\boldsymbol{B}\times\boldsymbol{u}) = -\frac{1}{\mu_0}\nabla\times(\eta\nabla\times\boldsymbol{B}), \tag{2.16}$$

where μ_0 denotes permeability of vacuum, η is resistivity, and U_m is magnetic energy. The equation 2.15 is the differential form of the *Ohm's law*, the equation 2.16 is the *induction equation*.

Applying now 2.1.2 to 2.5, 2.6, we obtain:

$$\frac{\partial\boldsymbol{\pi}}{\partial t} + \nabla\cdot(\boldsymbol{\pi}\otimes\boldsymbol{u}) = \rho_q\boldsymbol{E} + \boldsymbol{J}\times\boldsymbol{B} + \rho\boldsymbol{g} - \nabla p, \tag{2.17}$$

$$\frac{\partial\tilde{U}}{\partial t} + \nabla\cdot\left(\tilde{U}\boldsymbol{u}\right) = \rho_q\boldsymbol{E}\cdot\boldsymbol{u} + \boldsymbol{J}\times\boldsymbol{B}\cdot\boldsymbol{u} + \rho\boldsymbol{g}\cdot\boldsymbol{u} - \nabla\cdot(p\boldsymbol{u}) + \nabla\cdot\boldsymbol{q} \tag{2.18}$$

The adjusted energy equation 2.18 does not include the magnetic energy U_m, which we do want to include in the MHD equations. To achieve this, we employ 2.14 - 2.16 and rearrange. After rearranging we obtain

$$\frac{\partial U}{\partial t} = -\nabla\cdot\left[\boldsymbol{\pi}\left(\frac{u^2}{2}+e\right) + p\boldsymbol{u} - \frac{1}{\mu_0}\boldsymbol{B}\times\boldsymbol{E}\right] + \rho\boldsymbol{g}\cdot\boldsymbol{u} + \nabla\cdot\boldsymbol{q}. \tag{2.19}$$

2.1.5 Simplifying assumptions

In what follows, we will make several simplifying assumptions, according to which we will get a system of equations that will adequately respect the physical model, yet will be easier to be solved.

Negligible time derivative of electric field

For the time increments that we are concerned with, the time derivative in 2.10 (the so-called *Maxwell's displacement current*) is very small. To estimate the minimum time increment value τ which would allow us to neglect the derivative, take the ratio

of the two terms on the right hand side of 2.10:

$$\varepsilon_0 \frac{\frac{\partial \boldsymbol{E}}{\partial t}}{\boldsymbol{J}} \approx \frac{\frac{\varepsilon_0 \boldsymbol{E}}{\tau}}{\sigma \boldsymbol{E}} \approx \frac{\varepsilon_0}{\sigma\tau} \approx \frac{10^{-11}}{\tau}. \tag{2.20}$$

This means for time scales much greater than 10^{-11} seconds, the time derivative of $\boldsymbol{E}$ can be neglected. As a consequence Equation 2.10 can be written as:

$$\nabla \times \boldsymbol{B} = \mu_0 \boldsymbol{J}. \tag{2.21}$$

Using 2.21, we can write

$$\mu_0 \boldsymbol{J} \times \boldsymbol{B} = (\boldsymbol{B} \cdot \nabla)\, \boldsymbol{B} - \nabla \frac{B^2}{2} = \nabla \cdot (\boldsymbol{B}\boldsymbol{B}) - \nabla \frac{B^2}{2}, \tag{2.22}$$

where the last equality comes from 2.13. And using 2.22 we can rewrite 2.17 as

$$\frac{\partial \boldsymbol{\pi}}{\partial t} + \nabla \cdot (\boldsymbol{\pi} \otimes \boldsymbol{u}) = q\boldsymbol{E} + \nabla \cdot \left(\frac{1}{\mu_0} \boldsymbol{B}\boldsymbol{B} - \frac{B^2}{2} \mathrm{I} \right) + \rho \boldsymbol{g} - \nabla p \tag{2.23}$$

Negligible electric field in the Navier-Stokes equations

The magnitude of electric field is smaller than the magnetic field by the factor $\frac{u^2}{c^2}$, so we can neglect the term $\rho_q \boldsymbol{E}$ on the right-hand-side of 2.23.

Negligible heat fluxes

Since the heat transfer accounts for a negligible contribution to the overall energy transfer, we neglect the heat flux terms, i.e. we set $\boldsymbol{q} = \boldsymbol{0}$ in the energy equation 2.19.

2.1.6 Adding the induction equation

The induction equation 2.16 can be rearranged in the following way:

$$\frac{\partial \boldsymbol{B}}{\partial t} + \nabla \times (\boldsymbol{B} \times \boldsymbol{u}) = -\frac{1}{\mu_0} \nabla \times (\eta \nabla \times \boldsymbol{B}) \tag{2.24}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} = -\nabla \times (\boldsymbol{u} \otimes \boldsymbol{B} - \boldsymbol{B} \otimes \boldsymbol{u}) + \frac{1}{\mu_0 \sigma} \left(\nabla^2 \boldsymbol{B} \right). \tag{2.25}$$

Now we can form the system of MHD equations:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\boldsymbol{\pi}), \tag{2.26}$$

$$\frac{\partial \boldsymbol{\pi}}{\partial t} = -\nabla \cdot (\boldsymbol{\pi} \otimes \boldsymbol{u}) + \nabla \cdot \left(\frac{1}{\mu_0} \boldsymbol{B}\boldsymbol{B} - \frac{B^2}{2} \mathrm{I} \right) + \rho \boldsymbol{g} - \nabla p, \tag{2.27}$$

$$\frac{\partial U}{\partial t} = -\nabla \cdot \left[\boldsymbol{\pi} \left(\frac{u^2}{2} + e \right) + p\boldsymbol{u} - \frac{1}{\mu_0} \boldsymbol{B} \times \boldsymbol{E} \right] + \rho \boldsymbol{g} \cdot \boldsymbol{u} \tag{2.28}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} = -\nabla \times (\boldsymbol{u} \otimes \boldsymbol{B} - \boldsymbol{B} \otimes \boldsymbol{u}) + \frac{1}{\mu_0 \sigma} \left(\nabla^2 \boldsymbol{B} \right). \tag{2.29}$$

This form suggests that rewriting these equations into a more suitable (for numerical calculations) conservative form shall be possible.

2.1.7 Conservative form of the MHD equations

A conservative form of a system of equations takes the form of

$$\frac{\partial \boldsymbol{\Psi}}{\partial t} + \nabla \cdot \mathbf{F}(\boldsymbol{\Psi}) = \mathbf{S}, \tag{2.30}$$

where $\boldsymbol{\Psi}$ is the so-called *state vector*, $\mathbf{F}_i,\ i = 1, 2, 3$ are the so-called *fluxes*, and $\mathbf{S}$ is the so-called *source term*.

Rewriting the system of equations 2.26 - 2.29 to the form 2.30 is fairly straightforward. We obtain the following:

$$\boldsymbol{\Psi} = \begin{pmatrix} \rho \\ \pi_1 \\ \pi_2 \\ \pi_3 \\ U \\ B_1 \\ B_2 \\ B_3 \end{pmatrix}, \tag{2.31}$$

$$\mathbf{F}_i = \begin{pmatrix} \pi_i \\ \frac{\pi_1\pi_i}{\rho} - B_1B_i + \frac{1}{2}\delta_{1i}\left(p + U_m\right) \\ \frac{\pi_2\pi_i}{\rho} - B_1B_i + \frac{1}{2}\delta_{2i}\left(p + U_m\right) \\ \frac{\pi_3\pi_i}{\rho} - B_1B_i + \frac{1}{2}\delta_{3i}\left(p + U_m\right) \\ \frac{\pi_i}{\rho}\left(\frac{\gamma}{\gamma-1}p + U_k\right) + 2\eta\varepsilon_{ijk}J_jB_k + \frac{2}{\rho}\varepsilon_{ijk}\left(\pi_kB_i - \pi_iB_k\right)B_k \\ \frac{\pi_iB_1 - \pi_1B_i}{\rho} + \eta\varepsilon_{1ij}J_j \\ \frac{\pi_iB_2 - \pi_2B_i}{\rho} + \eta\varepsilon_{2ij}J_j \\ \frac{\pi_iB_3 - \pi_3B_i}{\rho} + \eta\varepsilon_{3ij}J_j \end{pmatrix}, \tag{2.32}$$

$$\mathbf{S} = \begin{pmatrix} 0 \\ \rho g_1 \\ \rho g_2 \\ \rho g_3 \\ \boldsymbol{\pi} \cdot \boldsymbol{g} \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{2.33}$$

where $J_j = (\nabla \times \boldsymbol{B})_j$.

2.1.8 Solution considerations

For mathematical clarity, we should state, that the solution to the equations 2.30 is such a function

$$\boldsymbol{\Psi} \in C^1\left((0,T), \left[C^1\left(\Omega_t\right)\right]^8\right);\ \boldsymbol{\Psi_i} \in C^1\left((0,T), C^2\left(\Omega_t\right)\right),\ i = 6,7,8, \tag{2.34}$$

for which 2.30 holds for all $\boldsymbol{x} \in \Omega_t,\ t \in (0,T)$. The kind of spaces we used in the definition is called the Bochner spaces.

Because such a requirement on the solution $\boldsymbol{\Psi}$ is rather strong, we shall instead look for a so-called *weak solution*, which is specified in the coming sections.

2.2 Weak formulation of the problem

The DG method is defined by first considering the weak formulation of the equations 2.30 obtained by multiplying the equations at every time instance t by a (vector-valued) test function $\mathbf{v} \in W_t$, where W_t is a suitable space of (vector-valued) functions $W_t = (W_1, ..., W_8)$, integrating over the space domain Ω_t, and performing integration by parts. We start with multiplying 2.30 by a test function:

$$\frac{\partial \mathbf{\Psi}}{\partial t}\mathbf{v} + (\nabla \cdot \mathbf{F}(\mathbf{\Psi}))\,\mathbf{v} = \mathbf{S}\mathbf{v}, \tag{2.35}$$

then we integrate over Ω_t:

$$\int_{\Omega_t} \frac{\partial \mathbf{\Psi}}{\partial t}\mathbf{v} + \int_{\Omega_t} (\nabla \cdot \mathbf{F}(\mathbf{\Psi}))\,\mathbf{v} = \int_{\Omega_t} \mathbf{S}\mathbf{v}, \tag{2.36}$$

and finally we integrate by parts:

$$\int_{\Omega_t} \frac{\partial \mathbf{\Psi}}{\partial t}\mathbf{v} - \int_{\Omega_t} \mathbf{F}(\mathbf{\Psi})\,(\nabla \cdot \mathbf{v}) + \int_{\partial\Omega_t} (\mathbf{F}(\mathbf{\Psi}) \cdot \boldsymbol{n})\,\mathbf{v} = \int_{\Omega_t} \mathbf{S}\mathbf{v}, \tag{2.37}$$

where the terms $\mathbf{F}(\mathbf{\Psi})\,(\nabla \cdot \mathbf{v})\,; (\mathbf{F}(\mathbf{\Psi}) \cdot \boldsymbol{n})\,\mathbf{v}$ are meant as a component-wise multiplication.

Now without going into detail, we can conclude, that a suitable space W would be

$$W_t = \left[H^1(\Omega_t)\right]^8, \tag{2.38}$$

and we shall look for a solution to the 2.37 in the same space W_t at every time instance t. We shall not relax the requirement for continuity of time-derivatives we imposed for the *hard solution* 2.34 for reasons discussed later, and we arrive at the following Bochner space in which we are looking for a weak solution of 2.30:

$$W = C^1((0,T), W_t). \tag{2.39}$$

To sum up we can define the *weak solution* $\mathbf{\Psi} = \mathbf{\Psi}((t, \boldsymbol{x}))$ of MHD equations 2.30 as

1. $\mathbf{\Psi}((t, \boldsymbol{x})) \in W$ defined in 2.39
2. 2.37 holds for all $t \in (0, T)$, and all $\mathbf{v} \in W_t$.
3. $\mathbf{\Psi}(0, \boldsymbol{x}) = \Pi\mathbf{\Psi}^0(\boldsymbol{x})$,

where Π is a projection of the initial condition $\mathbf{\Psi}^0$ onto W_0.

2.3 Boundary conditions

For the problem of finding the solution 2.2 to be complete, we need to specify the proper boundary conditions.

2.3.1 Essential (inflow) boundary conditions

Since the solution 2.2 of the problem specified in 2.37 is not influenced by its values on the boundary Ω_t, we are not able to employ standard essential boundary conditions of the form $\boldsymbol{u}(x,y,z) = \boldsymbol{u}_D(x,y,z)$ with a known $\boldsymbol{u}_D$. If such a condition is required from the physical nature of the described phenomenon (as often is the case), it is only implied by the use of fluxes - as one can see in the last integrand $\mathbf{F}(\boldsymbol{\Psi})\mathbf{v}$ in 2.37.

2.3.2 Outflow (do-nothing) boundary conditions

If we want to model free boundary, that is not in any way present as an actual physical boundary or interface, this is usually achieved by specifying the fluxes through the boundary $\mathbf{F}(\boldsymbol{\Psi})\mathbf{v}$ in 2.37 to be zero:

$$\mathbf{F}(\boldsymbol{\Psi}(\boldsymbol{x})) = 0 \; \forall \boldsymbol{x} \in \partial\Omega_t \, \forall t \in (0,T).$$

2.3.3 Periodic boundary conditions

Periodic boundary condition is always specified on two parts Γ_1, Γ_2 of the domain boundary that share the bijection mapping between points:

$$[x_1,y_1,z_1] \leftrightarrow [x_2,y_2,z_2] \, \forall \, [x_1,y_1,z_1] \in \Gamma_1, \; \forall \, [x_2,y_2,z_2] \in \Gamma_2, \tag{2.40}$$

so that for each pair of related points $[x_1,y_1,z_1] \leftrightarrow [x_2,y_2,z_2]$, the values must be the same:

$$u([x_1,y_1,z_1]) = u([x_2,y_2,z_2]) \, \forall \, [x_1,y_1,z_1] \in \Gamma_1, \; \forall \, [x_2,y_2,z_2] \in \Gamma_2 : \; [x_1,y_1,z_1] \leftrightarrow [x_2,y_2,z_2]. \tag{2.41}$$

3 Numerical approach

The weak formulation of the problem we obtained in 2.2 still posses a problematic attribute - the space defined in 2.39 is of infinite dimension, and therefore we would need to employ analytical methods to find the solution 2.2 in such a space. The equation 2.37 is however rather impossible to be solved analytically, and we have to utilize some sort of numerical simulation - which in turn needs to operate on finite-dimensional spaces. But we need to make sure that the simplifying (reducing) assumptions we make on the way to the numerical model are acceptable so that the numerical solution we obtain converges (as we reduce the discretization size) to the solution defined in 2.2.

In this chapter we shall consider that $\Omega_t = \Omega \, \forall t \in (0, T)$, i.e. the computational domain does not change with respect to time. There are approaches to numerical simulation of MHD phenomena without this condition in place, which utilize the exact same general approach described in this work plus they add additional steps in the algorithm. These are outside of the scope of this work. Also, we always take $\Omega \subset \mathbb{R}^3$.

3.1 Triangulation

We start with leaving the time-derivative untouched, and focus on the discretization in space for now - we are performing a *space semidiscretization*.

First step in the process of the discretization is to divide the computational domain $\overline{\Omega}$ into a finite number of subsets with properties described below. These subsets form the set, further denoted by T_h, called the *triangulation or mesh of the domain* Ω. Please note that the terms *triangulation* and *mesh* shall be used in the text interchangeably. The parameter $h > 0$ of the triangulation usually represents maximum of diameters of all elements $K \in T_h$. The elements $K \in T_h$ are in the context of the finite volume method called *finite volumes*.

Properties of T_h:

1. Each $K \in T_h$ is closed and connected with its interior $K^\circ \neq \emptyset$.
2. Each $K \in T_h$ has a Lipschitz boundary.
3. $\cup_{K \in T_h} K = \overline{\Omega}$

4. If $K_1, K_2 \in T_h$, $K_1 \neq K_2$, then $K_1^\circ \cap T_2^\circ = \emptyset$.

In our case of the three-dimensional problem, we assume that the domain Ω is obtained as an approximation of the original computational domain (also denoted by Ω), and the triangulation is chosen accordingly to the following attributes:

A) Each $K \in T_h$ is a closed rectangular parallelepiped, possibly with curved edges.

B) For $K_1, K_2 \in T_h$, $K_1 \neq K_2$ we have either $K_1 \cap K_2 = \emptyset$ or K_1, K_2 share one edge (if the shared edge is a whole common edge, we call the triangulation *regular*), or K_1, K_2 share one vertex, or K_1, K_2 share one face.

C) $\cup_{K \in T_h} K = \overline{\Omega}$.

Furthermore

$$T_h = \{K_i, i \in I\}, \tag{3.1}$$

where $I \subset Z^+ = \{0, 1, 2, ...\}$ is a suitable index set.
By Γ_{ij} we denote a common face between two neighboring elements K_i and K_j. We set

$$s(i) = \{j \in I; K_j \text{ is adjacent to } K_i\}.$$

The boundary $\partial\Omega$ is formed by a finite number of faces of elements K_i adjecent to $\partial\Omega$. We denote all these boundary faces by S_j, where $j \in I_b \subset Z^- = \{-1, -2, ...\}$. Now we set

$$\gamma(i) = \{j \in I_b; S_j \text{ is a face of } K_i \in T_h\}$$

and

$$\Gamma_{ij} = S_j \text{ for } K_i \in T_h \text{ such that } S_j \subset \partial K_i, j \in I_b.$$

For K_i not containing any boundary face S_j we set $\gamma(i) = \emptyset$.
Obviously, $s(i) \cap \gamma(i) = \emptyset$ for all $i \in I$. If we write $S(i) = s(i) \cup \gamma(i)$, we have

$$\partial K_i = \cup_{j \in S(i)} \Gamma_{ij}, \quad \partial K_i \cap \partial\Omega = \cup_{j \in \gamma(i)} \Gamma_{ij}.$$

Furthermore we define the set of internal (i.e. not lying on the boundary $\partial\Omega$) edges as

$$\Gamma_I = \cup_{i \in I} \cup_{j \notin \gamma(i)} \Gamma_{ij}, \tag{3.2}$$

and the set of boundary (i.e. lying on the boundary $\partial\Omega$) edges as

$$\Gamma_B = \cup_{i \in I} \cup_{j \in \gamma(i)} \Gamma_{ij}. \tag{3.3}$$

Note If we were to use not $\Omega \subset \mathbb{R}^3$, but rather $\Omega \subset \mathbb{R}^4$, we may just employ the following machinery also to the time-derivative - this is not an uncommon approach. Why the approach described in this work is favored by the author is twofold:

- Data (in a general sense - e.g. algebraic systems, function bases, etc.) are smaller when using a separate handling for time-derivative
- The dependency on time and space may (and usually does) vary a lot for physical phenomena - to have a separate approach is therefore beneficial

3.1.1 Distributed triangulation

The standard approach to handle large problems that are impossible to be calculated on a single processor in mesh-based numerical simulations (such as Discontinuous Galerkin method) is to employ a *domain decomposition* method, where each of the processors on which the simulation runs holds data about a subset of elements of the mesh T. Consequently, also the matrix and vector assembly (described in 2), the linear problem solution, slope limiting, and AMR procedures are performed by all processors using data they have available. The aim here is not to go into deep technical details of distributing data, etc.

TODO Add pictures of domain decomposition

3.2 Discontinuous Galerkin method

For complex problems of compressible flow, and of course for even more complex problems of compressible MHD, there has been a number of attempts to use standard and well known Finite Element Methods that replace the spaces defined in 2.39 with finite-dimension spaces with bases formed by continuous piecewise polynomial functions. These attempts struggled with a common problem of spurious oscillations appearing in the solution - the origin of which is the lack of "stabilization", provided by the second-order terms in elliptic equations. Solution to these problems is the application of stabilization techniques, that usually introduce some sort of artificial diffusion (the second-order term), all of which are non-physical, and generally involve "magical" numbers - constants that are of pure computational nature (not a part of the physical description) or even worse are problem-specific.

3.2.1 Overview of the DG method

Due to this reason, there was an effort to develop methods which would not need such stabilization techniques, and would still offer reasonable resolution of shock-waves, boundary and interior layers, and steep gradients without exhibiting spurious oscillations in the approximate solutions. The approach taken here is based on the idea to combine finite volume and Finite element methods leading to the so-called *discontinuous Galerkin finite element method (DGFEM, DG)*. Here we shall derive and analyze DG for our equations. Let T_h be a triangulation of Ω. For each $K \in T_h$

we introduce the notation

$$\partial K^- = \{x \in \partial K; \beta(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) < 0\}, \tag{3.4}$$
$$\partial K^+ = \{x \in \partial K; \beta(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) \geq 0\}. \tag{3.5}$$

By $H^1(\Omega, T_h)$ we denote the so-called *broken Sobolev space*:

$$H^1(\Omega, T_h) = \left\{v \in L^2(\Omega);\ v|_K \in H^1(K)\, \forall K \in T_h\right\}. \tag{3.6}$$

This space is an approximation of the space defined in 2.38, but it contains functions that are discontinuous on element interfaces $\Gamma_i j$.

For $u \in H^1(\Omega, T_h)$ we set

$$u_K^+ = \text{trace of } u|_K \text{ on } \partial K \tag{3.7}$$

(i.e. the interior trace of u on ∂K). For each face $E \subset \partial K \backslash \Gamma$ of K, there exists $K' \neq K,\ K' \in T_h$, adjacent to E from the opposite side than K. Then we put

$$u_K^- = \text{trace of } u|_{K'} \text{ on } E. \tag{3.8}$$

In this way we obtain the exterior trace u_K^- of u on $\partial K \backslash \Gamma$ and define the jump of u on $\partial K \backslash \Gamma$:

$$[u]_K = u_K^+ - u_K^-. \tag{3.9}$$

Approximation of the broken Sobolev space

Let the domain Ω be covered with a mesh $T_h = \{K_1,\ K_2, \ldots, K_M\}$ where each element K_m carries an arbitrary polynomial degree $1 \leq p_m, \forall m = 1, 2, \ldots, M$. The broken Sobolev space $H^1(\Omega, T_h)$ will be approximated by a finite-dimensional space of picewise-polynomial functions

$$V_h = \{v \in L^2(\Omega);\ v|_{K_m} \in P^{p_m}(K_m) \text{ for all } 1 \leq m \leq M\} \tag{3.10}$$

where P^p is defined as

$$P^p = \text{span}\{\sum_{\substack{0 \leq i,j,k \leq p \\ i+j+k \leq p}} \alpha_i\ x_1^i\ x_2^j\ x_3^k,\quad \alpha_i \in \mathbb{R}\}.$$

3.2.2 DG formulation of MHD equations

Although the resulting system will look very similar to the weak formulation 2.37, the derivation makes more sense to be done starting with the 2.30.

As stated in 3.1, at this point we will discretize the problem in space, and leave the time-derivative untouched. The approximate solution will be sought at each time instant t as an element of the finite-dimensional space

$$[V_h]^8, \tag{3.11}$$

where V_h is defined in 3.10. Functions

$$\mathbf{v}_h \in [V_h]^8 \approx \left[H^1(\Omega, T_h)\right]^8, \tag{3.12}$$

where $H^1(\Omega, T_h)$ is defined in 3.6, are in general discontinuous on interfaces Γ_{ij}. By $\mathbf{v}_h|_{ij}$ and $\mathbf{v}_h|_{ji}$ we denote the values of $\mathbf{v}_h$ on Γ_{ij} considered from the interior and the exterior of K_i, respectively. The symbols

$$\langle \mathbf{v}_h \rangle_{ij} = \frac{1}{2}\left(\mathbf{v}_h|_{ij} + \mathbf{v}_h|_{ji}\right), \ [\mathbf{v}_h]_{ij} = \mathbf{v}_h|_{ij} - \mathbf{v}_h|_{ji}$$

denote the average and jump of a function $\mathbf{v}_h$ on Γ_{ij}. In order to derive the discrete problem, we multiply 2.30 by a test function $\mathbf{v}_h \in [V_h]^8$ in a component-wise fashion, integrate over any element $K_i \in T_h$, apply Green's theorem and sum over all $i \in I$, where I is defined in 3.1:

$$\int_{\Omega_t} \frac{\partial \mathbf{\Psi}_h}{\partial t}\mathbf{v}_h - \sum_{K_i \in T_h} \int_{K_i} \mathbf{F}(\mathbf{\Psi}_h)(\nabla \cdot \mathbf{v}_h) + \sum_{K_i \in T_h} \sum_{j \in s_i} \int_{\Gamma_{ij}} \left(\mathbf{F}(\mathbf{\Psi}_h) \cdot \boldsymbol{n}_{ij}\right)\mathbf{v}_h = \int_{\Omega_t} \mathbf{S}\mathbf{v}_h, \tag{3.13}$$

where $\boldsymbol{n}_{ij}$ is the unit outer normal to Γ_{ij}. Now, the term

$$\int_{\Gamma_{ij}} \mathbf{F}(\mathbf{\Psi}_h) \cdot \boldsymbol{n}_{ij}\mathbf{v}_h \tag{3.14}$$

is problematic, because the value of $\mathbf{\Psi}_h$ on Γ_{ij} is not unique - we have two values:

- $\mathbf{\Psi}_h|_{ij}$ - which is the value of $\mathbf{\Psi}_h$ on Γ_{ij} considered from the element K_i,
- $\mathbf{\Psi}_h|_{ji}$ - which is the value of $\mathbf{\Psi}_h$ on Γ_{ij} considered from the element K_j.

Note: This corresponds to the notation set in 3.7, 3.8 - if we take K_i as the element at hand, we have

$$\mathbf{\Psi}_h|_{ij} = \mathbf{\Psi}_{h\,K_i}^{+}, \quad \mathbf{\Psi}_h|_{ji} = \mathbf{\Psi}_{h\,K_i}^{-}$$

Now, because of this non-uniqueness of the values, we replace the term 3.14 with the so-called *numerical flux* $\mathbf{H} = \mathbf{H}(\mathbf{v}, \mathbf{w}, \boldsymbol{n})$ in the following fashion:

$$\left(\mathbf{F}(\mathbf{\Psi}_h) \cdot \boldsymbol{n}_{ij}\right)\mathbf{v}_h \approx \mathbf{H}(\mathbf{\Psi}_h|_{ij}, \mathbf{\Psi}_h|_{ji}, \boldsymbol{n}_{ij})\,\mathbf{v}_h. \tag{3.15}$$

We impose the following requirements on the numerical flux:

A) $\mathbf{H}(\mathbf{v}, \mathbf{w}, \boldsymbol{n})$ is defined and continuous on $\mathcal{D} \times \mathcal{D} \times \mathcal{S}_1$, where $\mathcal{D}$ is the domain of definition of the flux $\mathbf{F}$ and $\mathcal{S}_1$ is the unit sphere in $\mathbb{R}^3$.

B) $\mathbf{H}$ is *consistent*:

$$\mathbf{H}(\mathbf{v}, \mathbf{v}, \boldsymbol{n}) = \mathbf{F}(\mathbf{v})\,\boldsymbol{n},\ \mathbf{v} \in \mathcal{D},\ \boldsymbol{n} \in \mathcal{S}_1. \tag{3.16}$$

C) $\mathbf{H}$ is *conservative*:

$$\mathbf{H}(\mathbf{v}, \mathbf{w}, \boldsymbol{n}) = -\mathbf{H}(\mathbf{w}, \mathbf{v}, -\boldsymbol{n}),\ \mathbf{v}, \mathbf{w} \in \mathcal{D},\ \boldsymbol{n} \in \mathcal{S}_1. \tag{3.17}$$

And using these properties of the numerical flux, we can rewrite 3.13 as:

$$\begin{aligned} \int_{\Omega_t} \frac{\partial \boldsymbol{\Psi}_h}{\partial t} \mathbf{v}_h \quad &- \quad \sum_{K_i \in T_h} \int_{K_i} \mathbf{F}(\boldsymbol{\Psi}_h)\,(\nabla \cdot \mathbf{v}_h) \\ &+ \quad \sum_{\Gamma_{ij} \in \Gamma_I} \int_{\Gamma_{ij}} \mathbf{H}(\boldsymbol{\Psi}_h|_{ij}, \boldsymbol{\Psi}_h|_{ji}, \boldsymbol{n}_{ij})\,\mathbf{v}_h = \int_{\Omega_t} \mathbf{S}\mathbf{v}_h, \end{aligned} \tag{3.18}$$

where we used the definition of 3.2 on the page 13.

3.2.3 Numerical flux

Generally, the numerical flux function can be a non-differentiable (or even discontinuous) function. That is challenging from the perspective of the usage of Newton's method to solve the resulting nonlinear problem arising when using implicit time-discretization.
Another complication arising from evaluation of numerical fluxes on element interfaces exists in distributed solver, where we need to make sure that all processors have relevant data (e.g. previous solution values) from all cells that neighbor any cells assembled on the processor at hand. This issue gets worse when local mesh refinement (there are more neighbor elements of the current cell across the interface at hand), as well as if periodic boundary conditions are used (the neighbor graph is more complex).

Lax-Friedrichs numerical flux

This is the most straightforward numerical flux satisfying 3.16, and 3.17 and is defined as follows:

$$\mathbf{H}_{LF}(\mathbf{v}, \mathbf{w}, \boldsymbol{n}) = \frac{1}{2}\left[\mathbf{F}(\mathbf{v}) + \mathbf{F}(\mathbf{w})\right] - \frac{\alpha}{2}(\mathbf{w} - \mathbf{v}), \tag{3.19}$$

where the parameter α is the so-called *stabilization parameter*, which in order for the solution to be stable needs to fulfill $\alpha < \frac{\Delta t}{\Delta x}$, where $\Delta t, \Delta x$ need to satisfy the CFL condition 3.35. Now, this numerical flux is very sensitive to the choice of α

- for larger values, it tends to be very diffusive, but for lower values, it is unstable and produces oscillations in the vicinity of shocks [**?**]. In this work it is primarily used for implementation verification purposes, as due to its simplicity, the risk of errors in its implementation is rather negligible.

Riemann problem for MHD

For the class of numerical fluxes, known as the *Godunov type fluxes*, the numerical flux across the element boundary is constructed as the flux $\mathbf{F}$ of a solution of the following Riemann problem (3.20) at $x = 0$: *Riemann problem*

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial F}{\partial x} = 0, \tag{3.20}$$

where

$$\mathbf{U} = \begin{pmatrix} \rho \\ \pi_1 \\ \pi_2 \\ \pi_3 \\ U \\ B_2 \\ B_3 \end{pmatrix}, \ \mathbf{F} = \begin{pmatrix} \pi_1 \\ \frac{\pi_1^2}{\rho} - B_1^2 + \frac{1}{2}(p + U_m) \\ \frac{\pi_2 \pi_1}{\rho} - B_1 B_2 \\ \frac{\pi_3 \pi_1}{\rho} - B_1 B_3 \\ \frac{\pi_1}{\rho}\left(\frac{\gamma}{\gamma-1} p + U_k\right) + \frac{2}{\rho}(\pi_k B_1 - \pi_1 B_k) B_1 \\ \frac{\pi_1 B_2 - \pi_2 B_1}{\rho} \\ \frac{\pi_1 B_3 - \pi_3 B_1}{\rho} \end{pmatrix}, \tag{3.21}$$

and where due to the divergence free condition $\nabla \cdot \boldsymbol{B} = 0$ of the magnetic field), B_1 is given as constant.

These equations (3.20) have seven eigenvalues which correspond to two Alfve'n waves ($\lambda_{2,6}$), two slow magneto-acoustic waves ($\lambda_{3,5}$), and two fast magneto-acoustic waves ($\lambda_{1,7}$), and one entropy wave (λ_4):

$$\lambda_{2,6} = \frac{\pi_1}{\rho} \mp c_a, \tag{3.22}$$

$$\lambda_{3,5} = \frac{\pi_1}{\rho} \mp c_s, \tag{3.23}$$

$$\lambda_{1,7} = \frac{\pi_1}{\rho} \mp c_f, \tag{3.24}$$

$$\lambda_4 = \frac{\pi_1}{\rho}, \tag{3.25}$$

where $c_a = \sqrt{\frac{B_1^2}{rho}}$, $c_{s,f} = \left\{ \frac{\gamma p + |B|^2 \mp \sqrt{(\gamma p + |B|^2)^{\frac{1}{2}} - 4\gamma p B_1^2}}{2\rho} \right\}^{\frac{1}{2}}$.

For MHD equations, there is no exact solver of the Riemann problem across the element boundary, and therefore, approximate solvers are used. One instance of the derived numerical flux based on the solution at $x = 0$ of 3.20 is described in the next section.

HLLD numerical flux

The abbreviation **HLLD** stands for Harten-Lax-van Leer (HLL) approximate Riemann solver, and **D** stands for Discontinuities. This particular numerical flux has been introduced in [7] and has been shown to be very suitable for the studied problems. TODO: doplnit

Numerical fluxes comparison

For the comparison, a simple version of the benchmark described in 5.1.1 was used. In the next figures, there are snapshots of density solution component at several time steps for the Lax-Friedrichs flux for two values of $\alpha = 0.5$, $\alpha = 1.5$, and for the HLLD flux. Please note that the results here are calculated with piecewise-constant functions.

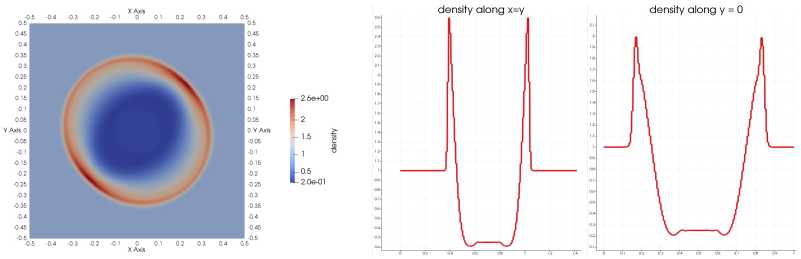


Figure 3.1: Density at time $t = 0.1$, Lax-Friedrichs flux with $\alpha = 0.5$

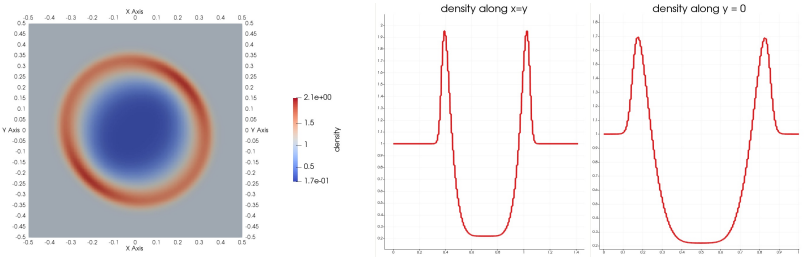


Figure 3.2: Density at time $t = 0.1$, Lax-Friedrichs flux with $\alpha = 1.5$

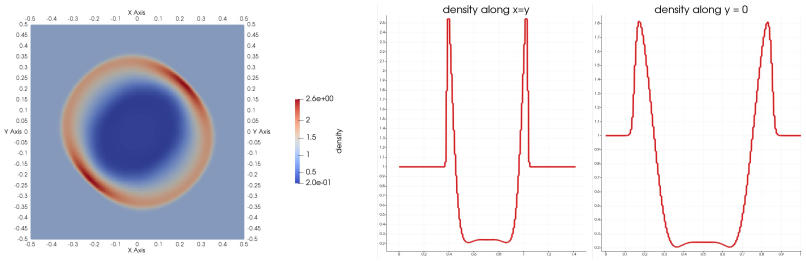


Figure 3.3: Density at time $t = 0.1$, HLLD flux

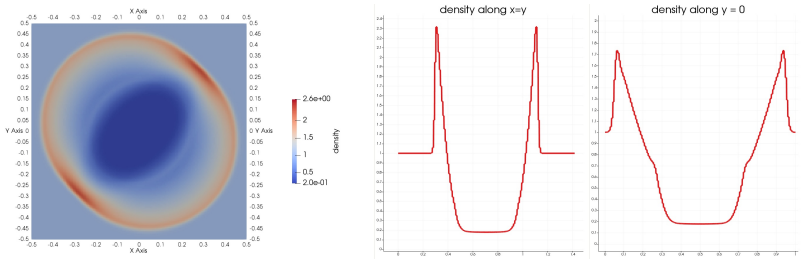


Figure 3.4: Density at time $t = 0.16$, Lax-Friedrichs flux with $\alpha = 0.5$

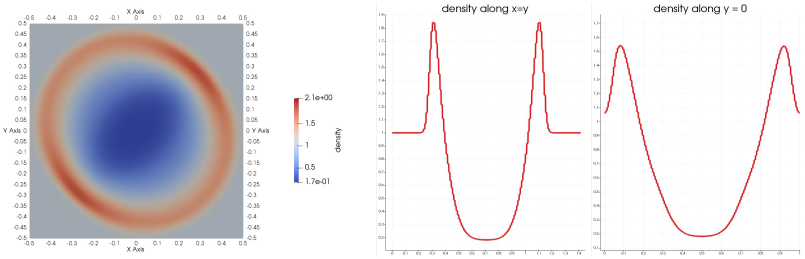


Figure 3.5: Density at time $t = 0.16$, Lax-Friedrichs flux with $\alpha = 1.5$

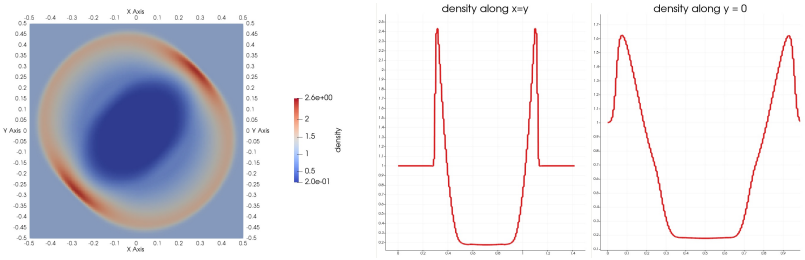


Figure 3.6: Density at time $t = 0.16$, HLLD flux

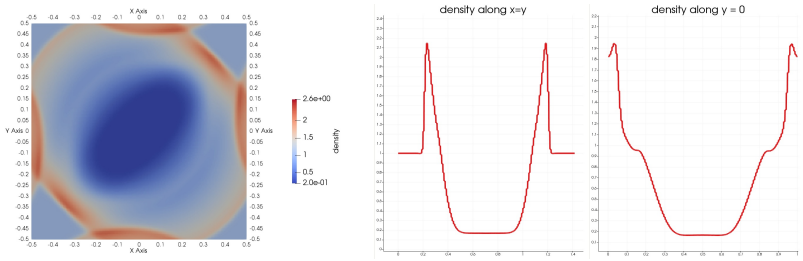


Figure 3.7: Density at time $t = 0.23$, Lax-Friedrichs flux with $\alpha = 0.5$

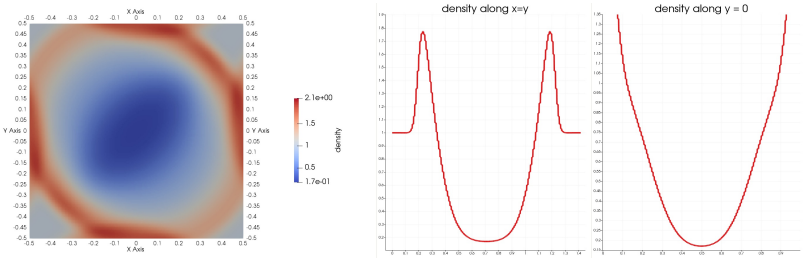


Figure 3.8: Density at time $t = 0.23$, Lax-Friedrichs flux with $\alpha = 1.5$

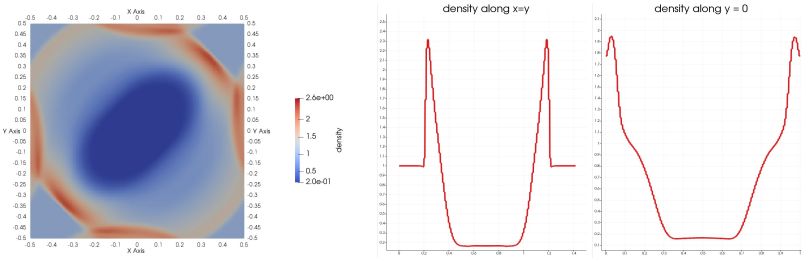


Figure 3.9: Density at time $t = 0.23$, HLLD flux

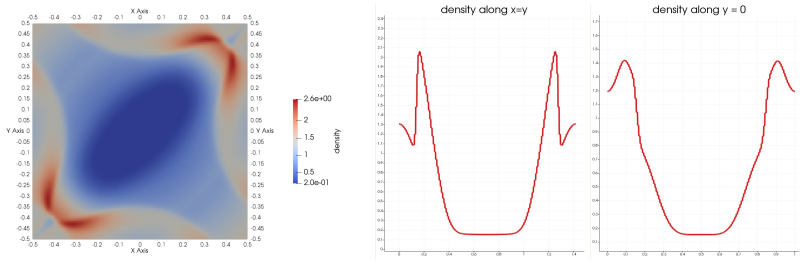


Figure 3.10: Density at time $t = 0.3$, Lax-Friedrichs flux with $\alpha = 0.5$

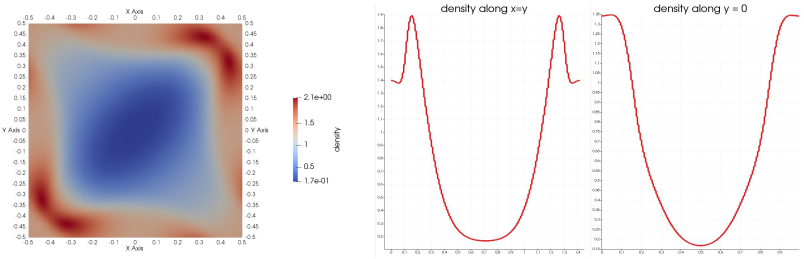


Figure 3.11: Density at time $t = 0.3$, Lax-Friedrichs flux with $\alpha = 1.5$

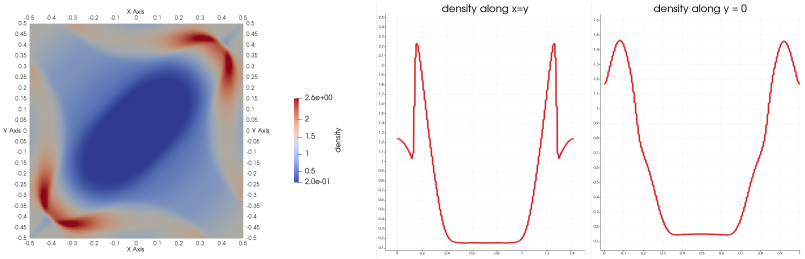


Figure 3.12: Density at time $t = 0.3$, HLLD flux

There are a few observations from all the triplets of Figures:

- The Lax-Friedrichs flux is very sensitive to the choice of α, for $\alpha = 1.5$, the solution is too diffusive,
- The HLLD flux is quantitatively and qualitatively very close to the 'ideal' Lax-Friedrichs flux where $\alpha = 0.5$.

The second observation could lead us to the conclusion that the Lax-Friedrichs flux, being simple to implement, is superior to HLLD which is quite complex. Unfortunately, as shown e.g. in [**?**], the Lax-Friedrichs flux is not stable enough to be used for arbitrary problem setup, also when handling solution with discontinuities, it is very dissipative [**?**].

Second reason why for practical usage of the developed code, the HLLD flux is strongly recommended is, that it is parameter-free, as opposed to Lax-Friedrichs flux having the stabilization parameter α.

3.2.4 Numerical handling of boundary conditions

In what follows, we are only interested in using flux-induced inflow and outflow boundary conditions (see Section 2.3). To account for these boundary conditions, we need to investigate the term

$$\int_{\Gamma_{ij}} \mathbf{H}\left(\boldsymbol{\Psi}_h|_{ij}, \boldsymbol{\Psi}_h|_{ji}, \boldsymbol{n}_{ij}\right) \mathbf{v}_h$$

for $\Gamma_{ij} \in \Gamma_B$ (see 3.3 on page 13). This term is used in 3.18 for faces in Γ_I which are internal and always have 2 values connected to them - $\boldsymbol{\Psi}_h|_{ij}, \boldsymbol{\Psi}_h|_{ji}$ - which induces the notation. On a boundary face, the corresponding value to $\boldsymbol{\Psi}_h|_{ij}$ can be defined correspondingly as in the case of Γ_I, but $\boldsymbol{\Psi}_h|_{ji}$ needs to be defined.

Inflow boundary condition

First, if we want to prescribe an inflow boundary condition (i.e. we know what values should the state vector $\boldsymbol{\Psi}_h$ have on $\Gamma_{ij} \in \Gamma_B$), we define

$$\overline{\boldsymbol{\Psi}_h|_{ji}} \tag{3.26}$$

to be the prescribed value.

Outflow boundary condition

If we want to model an outflow boundary condition (i.e. do nothing condition), we may use the *consistency* of the numerical flux $\mathbf{H}$ defined in 3.16, and define

$$\overline{\boldsymbol{\Psi}_h|_{ji}} = \boldsymbol{\Psi}_h|_{ij}, \tag{3.27}$$

which is a suitable definition for the outflow boundary condition. It is important to mention, that setting the inflow boundary condition does not imply that solution values on this boundary equal to these prescribed value. This follows from the definition of broken Sobolev space (3.6). Moreover the values of the solution on the boundary also depend on the numerical flux used, as the values on the boundary are merely one of the input parameters for the flux (See 3.15).

Periodic boundary conditions

Periodic boundary conditions come in the form described in 2.41, for pair of points on the boundary (2.40). The point mapping is the main complication encountered, namely in the distributed computations, and moreover if AMR is used in distributed computations (see 4.2). Otherwise, from the integral evaluation perspective, an edge on a periodic boundary is approached in the very same way as internal edges (we have values from both sides, and we evaluate the numerical flux in quadrature points).

DG full problem statement

Now, taking 3.26 and 3.27, we can enhance 3.18 with an additional term, that will add the boundary conditions into the equation:

$$\sum_{\Gamma_{ij}\in\Gamma_B}\int_{\Gamma_{ij}}\mathbf{H}\left(\boldsymbol{\Psi}_h|_{ij},\overline{\boldsymbol{\Psi}_h|_{ji}},\boldsymbol{n}_{ij}\right)\mathbf{v}_h,$$

so that the complete semi-discrete problem reads:

$$\begin{aligned}\int_{\Omega_t}\frac{\partial\boldsymbol{\Psi}_h}{\partial t}\mathbf{v}_h \quad &- \quad \sum_{K_i\in T_h}\int_{K_i}\mathbf{F}\left(\boldsymbol{\Psi}_h\right)\left(\nabla\cdot\mathbf{v}_h\right) \\ &+ \quad \sum_{\Gamma_{ij}\in\Gamma_I}\int_{\Gamma_{ij}}\mathbf{H}\left(\boldsymbol{\Psi}_h|_{ij},\boldsymbol{\Psi}_h|_{ji},\boldsymbol{n}_{ij}\right)\mathbf{v}_h \\ &+ \quad \sum_{\Gamma_{ij}\in\Gamma_B}\int_{\Gamma_{ij}}\mathbf{H}\left(\boldsymbol{\Psi}_h|_{ij},\overline{\boldsymbol{\Psi}_h|_{ji}},\boldsymbol{n}_{ij}\right)\mathbf{v}_h \\ &= \quad \int_{\Omega_t}\mathbf{S}\mathbf{v}_h.\end{aligned} \tag{3.28}$$

Now we can formulate the definition of the *semi-discrete solution* $\boldsymbol{\Psi}_h=\boldsymbol{\Psi}_h\left((t,\boldsymbol{x})\right)$ *of MHD equations 2.30* as

A) $\boldsymbol{\Psi}_h\in C^1\left((0,T),[V_h]^8\right)$,

B) 3.28 holds for all $t\in(0,T)$, and all $\mathbf{v}\in[V_h]^8$,

C) $\boldsymbol{\Psi}_h\left(0,\boldsymbol{x}\right)=\Pi_h\boldsymbol{\Psi}^0\left(\boldsymbol{x}\right)$,

where Π_h is a projection of the initial condition $\boldsymbol{\Psi}^0$ onto $[V_h]^8$.

3.3 Slope limiting

It is well known [**?**] that the Discontinuous Galerkin method exhibits nonphysical spurious oscillations in the vicinity of sharp discontinuities. Noteworthy is the fact, that with continuous Finite Element spaces, the situation is even worse, as the oscillations tend to propagate through the computational domain. With the DG method, the problem is localized to a single layer of elements bordering any sharp front. This behavior is not acceptable, and measures must be taken to eliminate such oscillations - methods aiming at solving this are usually labeled as flux limiters, slope limiters, or shock capturing schemes.

These methods can be categorized according to multiple aspects. Out of these, two are important from the perspective of this work. First categorization is whether the approach changes the equations by introducing additional term that 'smoothes' the solution near the sharp front (this may be understood as a form of *artificial viscosity / resistivity*) - such approach is proposed e.g. in [**?**].

In this work, such an approach is not preferred, as we aim at implementing a generally usable solver, where extensive analysis of the impact of a change in the governing equations for the particular problem is not possible.

Second categorization worth mentioning is whether the particular approach is suitable for multi-scale phenomena, where the solution can exhibit large jumps in all possible configurations with respect to the (non-uniformly refined) mesh. From a pragmatic perspective, a rather simple slope and robust limiting technique that does not change the governing equations is the Barth-Jespersen limiter [**?**]. The Barth-Jespersen limiter considers the piecewise-linear solution in the form

$$u_h(x) = u_c + \alpha_e (\nabla u)_c \cdot (x - x_c), \; 0 \leq \alpha_e \leq 1, \tag{3.29}$$

where u_c is the cell average on the element e, and $(\nabla u)_c$ is the gradient of the solution on the element e.

The sought parameter α_e is the parameter (correction factor) that determines the maximum admissible slope and is defined as:

$$\alpha_e = \min_i \begin{cases} \min\left\{1, \frac{u_e^{max} - u_c}{u_i - u_c}\right\}, & if \;\; u_i - u_c > 0, \\ 1, & if \; u_i - u_c = 0, \\ \min\left\{1, \frac{u_e^{min} - u_c}{u_i - u_c}\right\}, & if \;\; u_i - u_c < 0, \end{cases} \tag{3.30}$$

where $u_i = u_c + (\nabla u)_c \cdot (x_i - x_c)$ is the unconstrained solution value at the vertex x_i, and u_e^{min}, u_e^{max} are the minimum and maximum cell averages of all elements sharing a common edge with the element e.

However, using this limiting technique, there are two suboptimal behavior features - the bounds for the limited solution are set on one hand too tight - solution values from elements meeting at a vertex but having no common edge/face are not taken into account (they might extend the interval for the admissible correction factor α_e, and on the other hand too loose - solution values on elements that share a common edge may extend the admissible interval for α_e based on the value at a vertex that does not belong to that particular common edge.

Both these two problems are solved in the slope limiting technique chosen for this work

3.3.1 Vertex-based limiter

Introduced by D. Kuzmin in [**?**], the Vertex-based limiter aims at being an improvement over the Barth-Jespersen limiter. It also considers the solution in the form 3.29, but the definition of the correction factor α_e differs:

$$\alpha_e = \min_i \begin{cases} \min\left\{1, \frac{u_i^{max}-u_c}{u_i-u_c}\right\}, & if \quad u_i - u_c > 0, \\ 1, & if \; u_i - u_c = 0, \\ \min\left\{1, \frac{u_i^{min}-u_c}{u_i-u_c}\right\}, & if \quad u_i - u_c < 0, \end{cases} \tag{3.31}$$

where in this case u_i^{min}, u_i^{min} are defined in such a way that for each of the vertices they are initialized with a small and a large constant, respectively, and then in the loop over all elements that contain the $i-$th vertex, the values are updated as:

$$u_i^{max} = \max\left\{u_c, u_i^{max}\right\}, \quad u_i^{min} = \min\left\{u_c, u_i^{min}\right\}. \tag{3.32}$$

This slope limiting technique proves to have all the required attributes from the perspective of this work. The algorithm implementing this technique looks is presented

in 1 TODO algoritmus

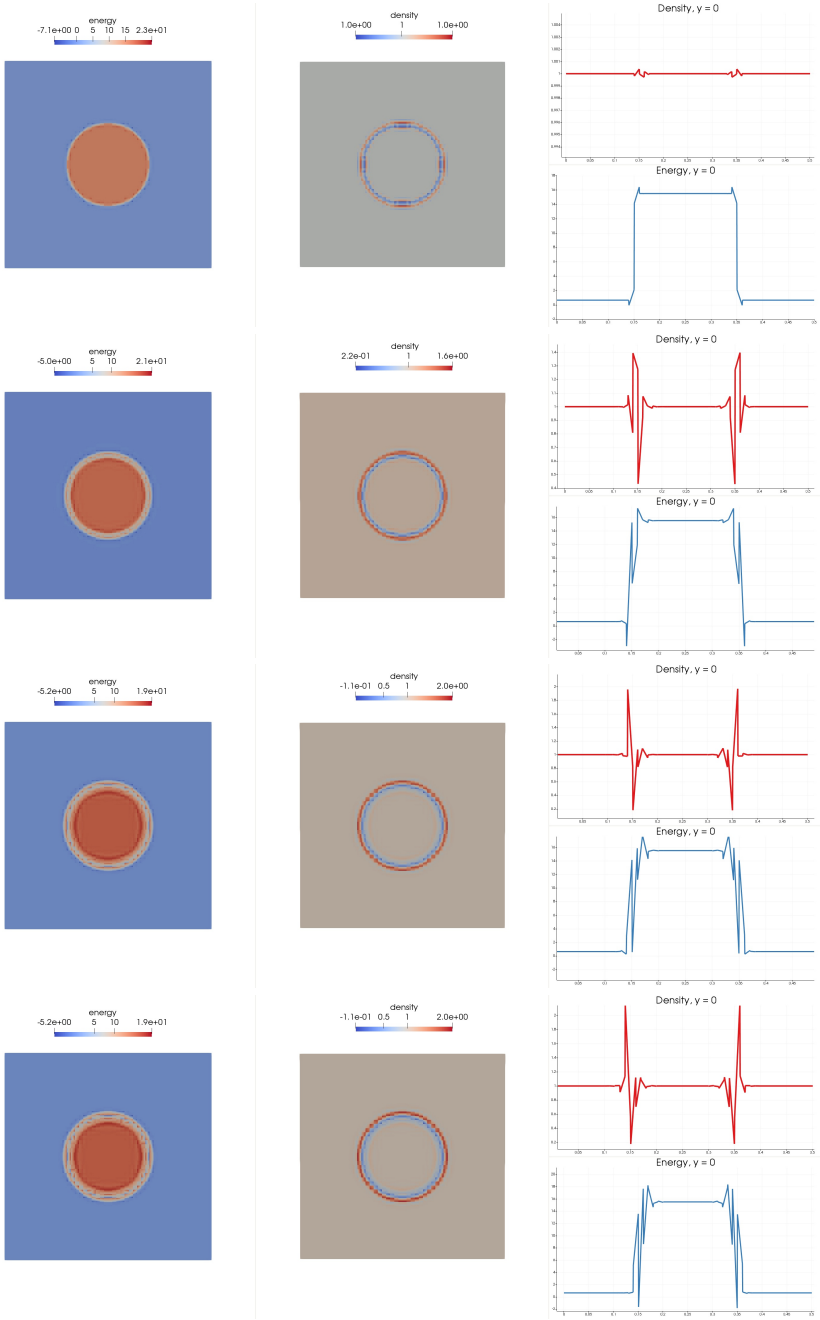
Algorithm 1: Assembling of the algebraic problem 3.46

```
# 1 - Loop over elements
foreach K_i ∈ T_h do
    Data: Quadrature points {x_1^i, ..., x_n^i}
    Data: Jacobian of the mapping J_{K_i} mapping the reference element
          (unit cube) to the actual element
    Data: Quadrature weights {w_1, ..., w_n}
    # Loop over quadrature points
    foreach j ∈ {1, ..., n} do
        Set: (JxW)_j = J_{K_i} × w_j
        # Loop over test functions
        foreach v ∈ v_h(K_i) do
            Data: l - index of v in the global system, i.e. row in 3.43 - 3.45
            # Loop over basis functions
            foreach u ∈ v_h(K_i) do
                Data: m - index of u in the global system, i.e. column in 3.43
                a_lm += (JxW)_j a_lmij
            b_l += (JxW)_j b_lij
# 2 - Loop over edges
foreach Γ_ij ∈ T_h do
    Data: Quadrature points {x_1^ij, ..., x_{n_f}^ij}
    Data: Jacobian of the mapping J_{K+} = J_{K−} mapping the reference face
          (unit square) to the actual face, where K+, K− are elements
          adjacent to Γ_ij if this is an internal edge, or K+ = K− if this is a
          boundary edge.
    Data: Quadrature weights {w_1, ..., w_{n_f}}
    # Loop over quadrature points
    foreach j_f ∈ {1, ..., n_f} do
        Set: (JxW)_{j_f} = J_K+ × w_{j_f}   # Here it does not matter if we
         choose J_{K+} or J_{K−}
        # Loop over test functions
        foreach v ∈ v_h(K+) do
            Data: l - index of v in the global system, i.e. row in 3.43 - 3.45
            b_l += (JxW)_{j_f} b'_{lij j_f}
```

Comparison of limited and unlimited solution

For illustration, the same problem as in 3.2.3 is considered, which is suitable for illustrating the undershoots and overshoots, as the problem contains a sharp front where this behavior is clearly visible. In the next Figures, first several snapshots of an unlimited solution, and then several snapshot of a limited solution are presented.

Note that the unlimited solution can't progress beyond a certain point, as the undershoots and overshoots get so large that the density values get to be negative, therefore e.g. calculating its square root needed for the speed of sound evaluation is impossible. Even before that, we get to non-physical regime, and e.g. the energy values start growing beyond limit.



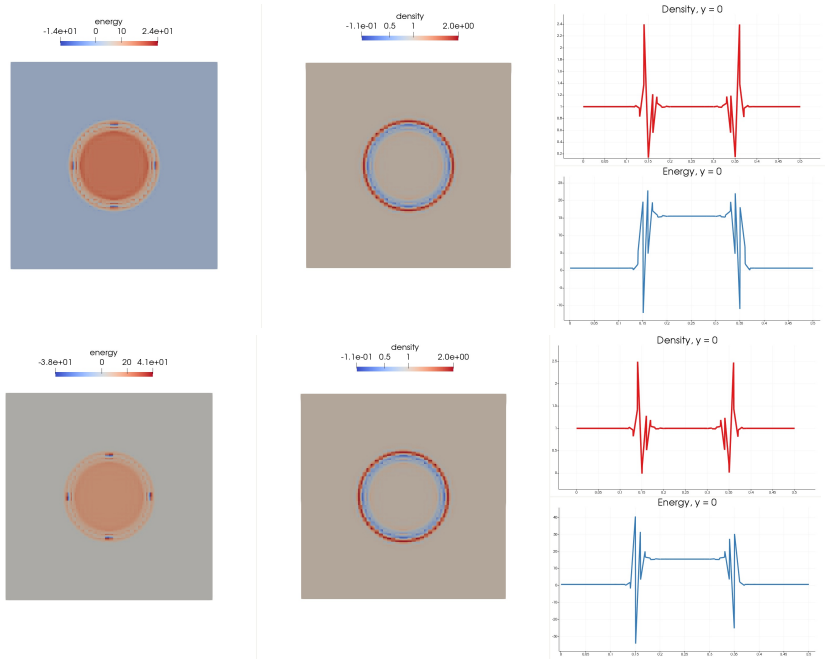
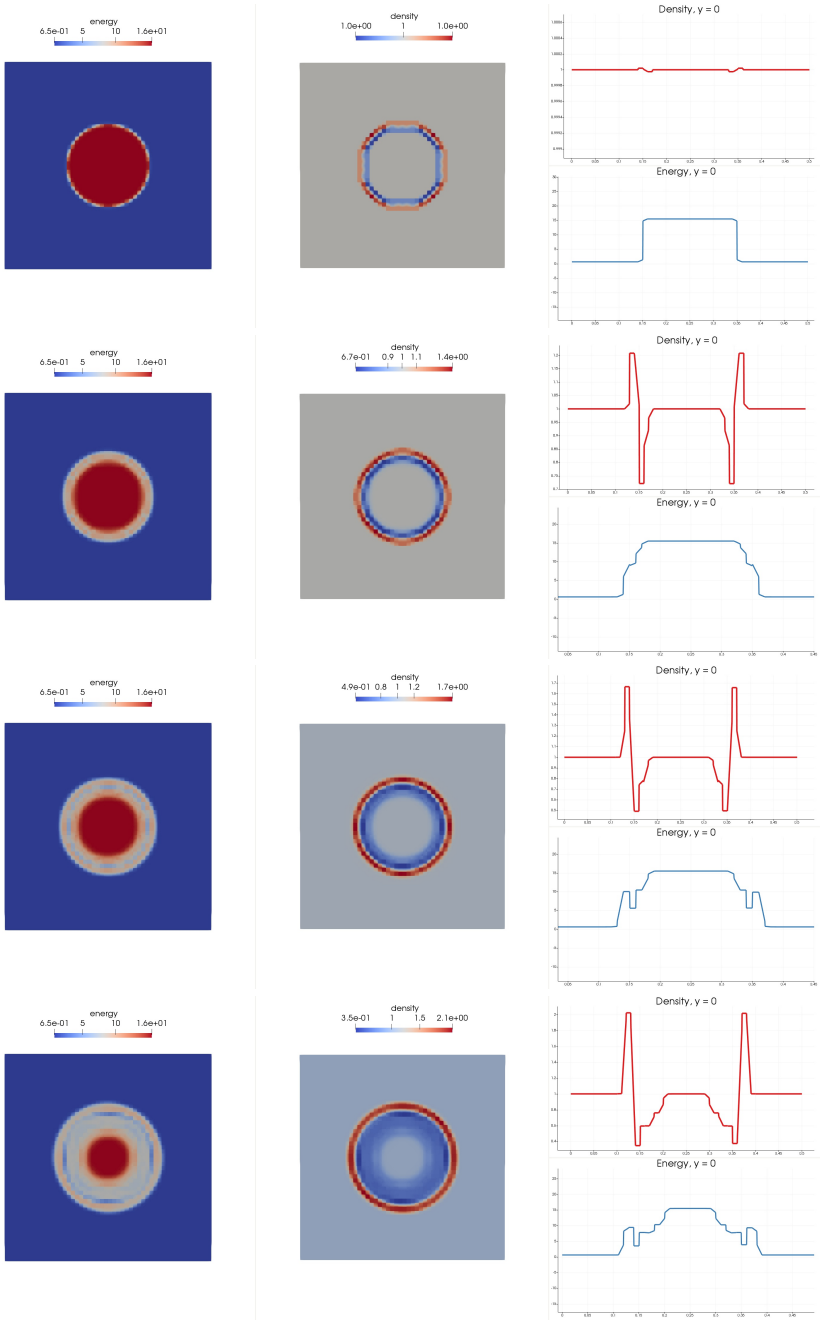


Figure 3.13: Unlimited solution - Energy, density, and their values over line y = 0, the solution cannot progress beyond the last snapshot, as the oscillations are orders of magnitude larger than the physical solution

Slope limiting



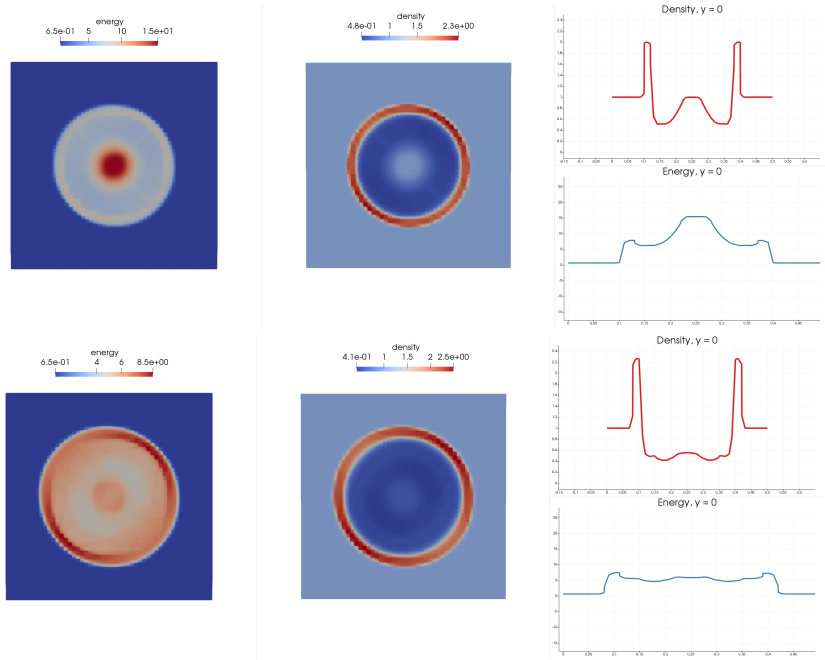


Figure 3.14: Solution limited with Vertex-based limiter - Energy, density, and their values over line y = 0

3.4 Divergence-free FE space

The divergence-free constraint of the magnetic field, $\boldsymbol{B} = 0$ (Gauss's law) is not enforced by the solution definition 2.2. Therefore, we need to perform additional work to be sure that we do not have a non-physical solution in the sense that the constraint is not satisfied.

There are two often used approaches to handle this problem - the Constraint-Transport (CT) method, and divergence cleaning. The first one is not suitable for this work, as it constraints the triangulation in such a way, that implementing Adaptive Mesh Refinement would be very complicated, if possible at all. The second approach, the divergence cleaning methods need additional postprocessing step which may be omitted for the sake of calculation efficiency. The approach taken in this work is to replace the standard FE space 3.11 with basis functions 3.12 for the magnetic field part (B) with a vector-valued (3-dimensional) space V_h^B of functions that have exactly

$$\nabla \cdot \mathbf{v}_h^B = 0, \quad \mathbf{v}_h^B \in V_h^B, \tag{3.33}$$

where these functions are as before discontinuous on interfaces Γ_{ij}. The basis of space V_h^B for piecewise-linear functions can be selected in several ways, in this work, the following basis was selected:

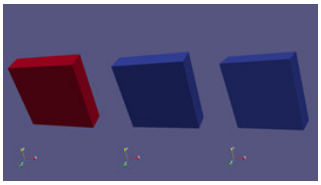
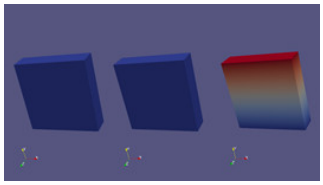
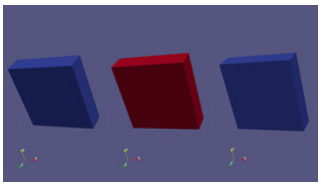
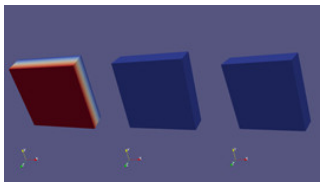
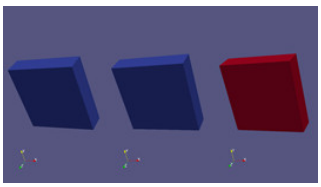
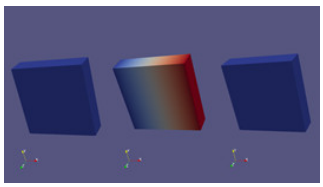
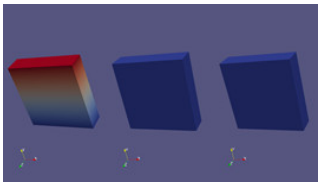
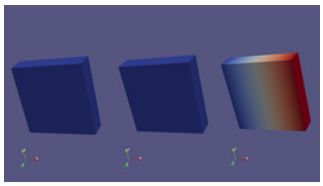
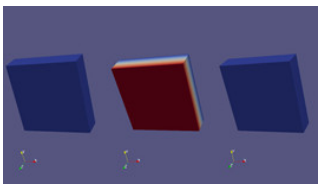
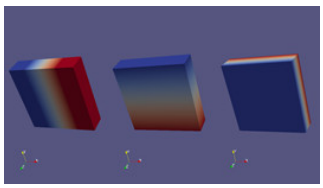
$\begin{pmatrix} B_x(x,y,z) \\ B_y(x,y,z) \\ B_z(x,y,z) \end{pmatrix}$	Visualization	$\begin{pmatrix} B_x(x,y,z) \\ B_y(x,y,z) \\ B_z(x,y,z) \end{pmatrix}$	Visualization
$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$		$\begin{pmatrix} 0 \\ 0 \\ y \end{pmatrix}$	
$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$		$\begin{pmatrix} z \\ 0 \\ 0 \end{pmatrix}$	
$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$		$\begin{pmatrix} 0 \\ z \\ 0 \end{pmatrix}$	
$\begin{pmatrix} y \\ 0 \\ 0 \end{pmatrix}$		$\begin{pmatrix} 0 \\ 0 \\ x \end{pmatrix}$	
$\begin{pmatrix} 0 \\ x \\ 0 \end{pmatrix}$		$\begin{pmatrix} 2x \\ -y \\ -z \end{pmatrix}$	

Table 3.1: Divergence-free space basis

In 3.1, it is obvious that only one function actually has more nonzero components.

In what follows, the notation $[V_h]^8$ used before will be used for the space where the last three components are replaced by V_h^B. Note that there are some technicalities with respect to the usage of V_h^B needed to be handled in computation, e.g. in 3.31,

or later in 3.49, 3.53 that we do not explicitly attend to.

3.5 Discretization in time

Relations 3.18 represent a system of ordinary differential equations which can be solved by a suitable numerical method. Since we are interested in applying the Rothe's method (as opposed to the method of lines, which switches the order of discretization in time and space), we now want to discretize the time derivative. In order to do so, we consider a partition $0 = t_0 < t_1 < t_2 < ...$ of the time interval $(0, T)$ and set $\tau_k = t_{k+1} - t_k$. We use the notation $\boldsymbol{w}_h^k$ for the approximation of $\boldsymbol{w}_h(t_k)$.

3.5.1 Discrete problem

Then we apply a time discretization scheme, for example, the simple explicit *Euler method* and our *fully discrete problem* reads: for each $k > 0$ find $\boldsymbol{w}_h^{k+1}$ such that

A) ${\boldsymbol{\Psi}_h}^{k+1} \in [V_h]^8$,

B) For all test functions $\mathbf{v}_h \in [V_h]^8$:

$$\begin{aligned} \int_{\Omega_t} \frac{{\boldsymbol{\Psi}_h}^{k+1} - {\boldsymbol{\Psi}_h}^{k}}{\tau} \mathbf{v}_h \quad &- \quad \sum_{K_i \in T_h} \int_{K_i} \mathbf{F}\left(\boldsymbol{\Psi}_h^k\right) (\nabla \cdot \mathbf{v}_h) \\ &+ \quad \sum_{\Gamma_{ij} \in \Gamma_I} \int_{\Gamma_{ij}} \mathbf{H}\left(\boldsymbol{\Psi}_h^k|_{ij}, \boldsymbol{\Psi}_h^k|_{ji}, \boldsymbol{n}_{ij}\right) \mathbf{v}_h \\ &+ \quad \sum_{\Gamma_{ij} \in \Gamma_B} \int_{\Gamma_{ij}} \mathbf{H}\left(\boldsymbol{\Psi}_h^k|_{ij}, \overline{\boldsymbol{\Psi}_h^k|_{ji}}, \boldsymbol{n}_{ij}\right) \mathbf{v}_h \\ &= \quad \int_{\Omega_t} \mathbf{S}\mathbf{v}_h, \end{aligned} \tag{3.34}$$

C) ${\boldsymbol{\Psi}_h}^0(\boldsymbol{x}) = \Pi_h \boldsymbol{\Psi}^0(\boldsymbol{x})$,

where Π_h is a projection of the initial condition $\boldsymbol{\Psi}^0$ onto $[V_h]^8$.

3.5.2 Time step length

Time step length is an important attribute of the discretization. If it is too small, the calculation might be taking too long to finish, with unnecessary precision with respect to time. If it is too large, we may end up with unstable calculation and obtain results with nonphysical oscillations, or without a solution whatsoever. That is why we need to take extra care to derive the proper value. From the stability perspective,

we have a condition for the upper bound of the time step - this condition is called the *Courant-Friedrichs-Lewy* condition. This condition is of the following form:

$$\tau_{max} = \min\left\{\frac{\Delta_{xmin}}{c_{max}}, \frac{\Delta_{xmin}^2}{2\eta_{max}}\right\}, \tag{3.35}$$

where Δ_{xmin} is the smallest dimension of any element, η_{max} highest resistivity in the domain, and v_{max} highest velocity in the domain, where the following velocities are taken into account:

$$c_s = \sqrt{\frac{\gamma(\gamma-1)}{\rho}(U - \rho v^2 - U_B)}, \tag{3.36}$$

$$c_a = \sqrt{\frac{B^2}{\rho}}, \tag{3.37}$$

$$u, \tag{3.38}$$

where c_s is the speed of sound, c_a is the Alfvén speed, and u is the speed of plasma. We then take

$$c_{max} = \max\{c_s, c_a, u\}.$$

3.6 Algebraic formulation

Last step in the DG method discretization is to transform the system of equations **??** into a system of linear algebraic equations at every time step t_k and obtain the solution at this time step as the solution of this linear algebraic system.

First, we rearrange the system in the following manner:

$$\begin{aligned}\sum_{K_i\in T_h}\int_{K_i}\mathbf{v}_h{\boldsymbol{\Psi}_h}^{k+1} &= \sum_{K_i\in T_h}\int_{K_i}\left[{\boldsymbol{\Psi}_h}^{k} + \tau\mathbf{S} + \tau\mathbf{A}\left(\boldsymbol{\Psi}_h^k\right)(\nabla\cdot\mathbf{v}_h)\right]\mathbf{v}_h \\ &- \sum_{\Gamma_{ij}\in\Gamma_I}\int_{\Gamma_{ij}}\mathbf{H}\left(\boldsymbol{\Psi}_h^k|_{ij}, \boldsymbol{\Psi}_h^k|_{ji}, \boldsymbol{n}_{ij}\right)\mathbf{v}_h \\ &- \sum_{\Gamma_{ij}\in\Gamma_B}\int_{\Gamma_{ij}}\mathbf{H}\left(\boldsymbol{\Psi}_h^k|_{ij}, \overline{\boldsymbol{\Psi}_h^k|_{ji}}, \boldsymbol{n}_{ij}\right)\mathbf{v}_h.\end{aligned} \tag{3.39}$$

We can see that the left hand side does not depend on the previous solution values, so there is no need to recalculate the matrix entries in every time step (unless we employ AMR, in which case the mesh and therefore the set of basis functions

changes). Now

$$\boldsymbol{\Psi}_h{}^{k+1} = \sum_{l=0}^{l=L} y_l \mathbf{v}_{hl}, L = \dim\left([V_h]^8\right) \tag{3.40}$$

for some (obviously finite) basis $\{v_{h1}, ..., v_{hL}\}$ of $[V_h]^8$. Next, since $\boldsymbol{\Psi}_h^k$, τ, S, $\mathbf{A}$ (and the basis) are all known, we can define

$$\begin{aligned}
a_{lm} &= \sum_{K_i \in T_h} \int_{K_i} \mathbf{v}_{hl}\mathbf{v}_{hm}, && (3.41)\\
b_l &= \sum_{K_i \in T_h} \int_{K_i} \left[\boldsymbol{\Psi}_h{}^k + \tau\mathbf{S} + \tau\mathbf{A}\left(\boldsymbol{\Psi}_h^k\right)(\nabla \cdot \mathbf{v}_{hl})\right]\mathbf{v}_{hl} && (3.42)\\
&- \sum_{\Gamma_{ij} \in \Gamma_I} \int_{\Gamma_{ij}} \mathbf{H}\left(\boldsymbol{\Psi}_h^k|_{ij}, \boldsymbol{\Psi}_h^k|_{ji}, \boldsymbol{n}_{ij}\right)\mathbf{v}_{hl}\\
&- \sum_{\Gamma_{ij} \in \Gamma_B} \int_{\Gamma_{ij}} \mathbf{H}\left(\boldsymbol{\Psi}_h^k|_{ij}, \overline{\boldsymbol{\Psi}_h^k|_{ji}}, \boldsymbol{n}_{ij}\right)\mathbf{v}_{hl},\\
A &= \{a_{lm}\}_{l,m=1}^{l,m=L}, && (3.43)\\
b &= \{b_l\}_{l=1}^{l=L}, && (3.44)\\
y &= \{y_l\}_{l=1}^{l=L}, && (3.45)
\end{aligned}$$

and rewriting 3.39 using 3.41 - 3.45, we come to the *fully discrete algebraic problem at time instance* t_{k+1}:

$$Ay = b, \tag{3.46}$$

whose well-posedness, and other attributes that allow for a successful solution of this system, come from the properties of the DG method. Now if we solve the system 3.46, and obtain the solution vector y, we are able to reconstruct the discrete solution $\boldsymbol{\Psi}_h{}^{k+1} \in [V_h]^8$ using the relation 3.40.

In the implementation, we take the elements $K \in T_h$, of the triangulation T_h to be rectangular hexahedra (rectangular parallelepipeds).

3.7 Numerical integration

Evaluation of the integral values in 3.41, 3.42 is performed using the *Gaussian numerical quadrature*. A quadrature rule approximates the integral values by replacing the integral as a weighted sum of integrand values at specified points in the domain of integration. The Gaussian quadrature is constructed so that the approximation is exact for polynomials of degree $2n$ - 1 (and less). This is acceptable, as our space V_h is constructed using polynomials - see section 3.2.1 on the page 15. We only need to take the value n to be corresponding to the value of p_m for the element K_m. The rule for both a 2-dimensional element face Γ, and a 3-dimensional cube

K is derived from a one-dimensional approximation (where the interval $[-1, 1]$ is a convention):

$$\int_{-1}^{1} f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i),$$

where the numbers $w_i > 0$ are the *quadrature weights*, and the points (numbers in this case) x_i are the *quadrature points*, in the following way:

$$\int_{\Gamma} f(\boldsymbol{x})\ dx = \int_{-1}^{1}\int_{-1}^{1} f(x_1, x_2)\ dx \approx \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j f(x_{1i}, x_{2j}),$$

$$\int_{K} f(\boldsymbol{x})\ dx = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(x_1, x_2, x_3)\ dx \approx \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} w_i w_j w_k f(x_{1i}, x_{2j}, x_{3k}),$$

and transformation to a generic rectangular hexahedron (rectangular parallelepiped) is performed using the transformation in one dimension:

$$\int_{a}^{b} f(x)\,dx \approx \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}x + \frac{a+b}{2}\right)\,dx.$$

Applying the Gaussian quadrature rule then results in the following one-dimensional approximation:

$$\int_{a}^{b} f(x)\,dx \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right).$$

And the transformations in higher dimensions follow naturally. For $\Gamma = [a_1, b_1] \times [a_2, b_2]$ we have:

$$\int_{\Gamma} f(x)\,d\boldsymbol{x} \approx \frac{b_2-a_2}{2}\frac{b_1-a_1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j f\left(\frac{b_1-a_1}{2}x_i + \frac{a_1+b_1}{2}, \frac{b_2-a_2}{2}x_j + \frac{a_2+b_2}{2}\right).$$

Taking now e.g. 3.41, and notation for quadrature points and weights, we can write (omitting the operand $\boldsymbol{x} = (x_1, x_2, x_3)$):

$$a_{lm} = \sum_{K_i \in T_h}\int_{K_i} \mathbf{v}_{hl}\mathbf{v}_{hm}, \tag{3.47}$$

$$a_{lm} := \sum_{K_i \in T_h}\int_{K_i} f(\mathbf{v}_{hl}, \mathbf{v}_{hm}), \tag{3.48}$$

$$a_{lm} \approx \sum_{K_i \in T_h}\sum_{\boldsymbol{j}=\overrightarrow{1}}^{\overrightarrow{n}} f\left(\mathbf{v}_{hl}\left(\boldsymbol{x}^i_{\boldsymbol{j}}\right), \mathbf{v}_{hm}\left(\boldsymbol{x}^i_{\boldsymbol{j}}\right)\right)\ w_{\boldsymbol{j}}, \tag{3.49}$$

where $\boldsymbol{j}$ is a multi-index used in sum over (volumetric) quadrature points $\boldsymbol{x}_{\boldsymbol{j}}^{i} \in K_i$. Similarly for the right-hand side (3.42):

$$
\begin{aligned}
b_l &= \sum_{K_i \in T_h} \int_{K_i} \left[{\boldsymbol{\Psi}_h}^k + \tau \mathbf{S} + \tau \mathbf{A} \left(\boldsymbol{\Psi}_h^k \right) (\nabla \cdot \mathbf{v}_{hl}) \right] \mathbf{v}_{hl} && (3.50)\\
&- \sum_{\Gamma_{ij} \in \Gamma_I} \int_{\Gamma_{ij}} \mathbf{H} \left(\boldsymbol{\Psi}_h^k|_{ij}, \boldsymbol{\Psi}_h^k|_{ji}, \boldsymbol{n}_{ij} \right) \mathbf{v}_{hl} \\
&- \sum_{\Gamma_{ij} \in \Gamma_B} \int_{\Gamma_{ij}} \mathbf{H} \left(\boldsymbol{\Psi}_h^k|_{ij}, \overline{\boldsymbol{\Psi}_h^k|_{ji}}, \boldsymbol{n}_{ij} \right) \mathbf{v}_{hl}, \\
b_l &:= \sum_{K_i \in T_h} \int_{K_i} g\left(\mathbf{v}_{hl}\right) - \sum_{\Gamma_{ij} \in \Gamma_I} \int_{\Gamma_{ij}} g^{'}\left(\mathbf{v}_{hl}\right) - \sum_{\Gamma_{ij} \in \Gamma_B} \int_{\Gamma_{ij}} g^{''}\left(\mathbf{v}_{hl}\right) && (3.51)\\
b_l &\approx \sum_{K_i \in T_h} \sum_{\boldsymbol{j}=\overrightarrow{1}}^{\overrightarrow{n}} g\left(\mathbf{v}_{hl}\left(\boldsymbol{x}_{\boldsymbol{j}}^{i}\right)\right) w_{\boldsymbol{j}} && (3.52)\\
&- \sum_{\Gamma_{ij} \in \Gamma_I} \sum_{\boldsymbol{j}_f=\overrightarrow{1}}^{\overrightarrow{n_f}} g^{'}\left(\mathbf{v}_{hl}\left(\boldsymbol{x}_{\boldsymbol{j}_f}^{ij}\right)\right) w_{\boldsymbol{j}_f} \\
&- \sum_{\Gamma_{ij} \in \Gamma_B} \sum_{\boldsymbol{j}_f=\overrightarrow{1}}^{\overrightarrow{n_f}} g^{''}\left(\mathbf{v}_{hl}\left(\boldsymbol{x}_{\boldsymbol{j}_f}^{ij}\right)\right) w_{\boldsymbol{j}_f},
\end{aligned}
$$

where in addition to $\boldsymbol{j}$ as explained before, $\boldsymbol{j}_f$ is a multi-index used sum over face quadrature points $\boldsymbol{x}_{\boldsymbol{j}_f}^{ij} \in \Gamma_{ij}$. Based on this, we can define

$$
\begin{aligned}
a_{lmi\boldsymbol{j}} &= f\left(\mathbf{v}_{hl}\left(\boldsymbol{x}_{\boldsymbol{j}}^{i}\right), \mathbf{v}_{hm}\left(\boldsymbol{x}_{\boldsymbol{j}}^{i}\right)\right) w_{\boldsymbol{j}}, && (3.53)\\
b_{li\boldsymbol{j}} &= g\left(\mathbf{v}_{hl}\left(\boldsymbol{x}_{\boldsymbol{j}}^{i}\right)\right) w_{\boldsymbol{j}}, && (3.54)\\
b^{'}_{lij\boldsymbol{j}_f} &= \begin{cases} g^{'}\left(\mathbf{v}_{hl}\left(\boldsymbol{x}_{\boldsymbol{j}_f}^{ij}\right)\right) w_{\boldsymbol{j}} & \text{if } \Gamma_{ij} \in \Gamma_I \\ g^{''}\left(\mathbf{v}_{hl}\left(\boldsymbol{x}_{\boldsymbol{j}_f}^{ij}\right)\right) w_{\boldsymbol{j}} & \text{if } \Gamma_{ij} \in \Gamma_B \end{cases}. && (3.55)
\end{aligned}
$$

3.8 Assembling the algebraic problem

Now we have a clear expression how to evaluate the integral values 3.41, 3.42 using 3.53, 3.54, 3.55, but we need to construct the matrix A (3.43), and the right-hand-side vector b (3.44) in an effective manner. This is generally achieved through a *element-wise* assembling of these structures. Key to this is to create a data structure that identifies for a particular element K_i all the test functions $\mathbf{v}_{hl}$ that make sense

to be evaluated (have non-empty support) on K_i, i.e. we are looking for the set

$$\mathbf{v}_h\left(K_i\right) = \left\{\mathbf{v}_h \in V_h : supp\left(\mathbf{v}_h\right) \cap K_i \neq \emptyset\right\}, \tag{3.56}$$

and do the same for the faces Γ_i (both boundary, and internal):

$$\mathbf{v}_h\left(\Gamma_i\right) = \left\{\mathbf{v}_h \in V_h : supp\left(\mathbf{v}_h\right) \cap \Gamma_i \neq \emptyset\right\}. \tag{3.57}$$

Now the assembling procedure looks like this:

Algorithm 2: Assembling of the algebraic problem 3.46

```
# 1 - Loop over elements
foreach K_i ∈ T_h do
    Data: Quadrature points {x_1^i, ..., x_n^i}
    Data: Jacobian of the mapping J_{K_i} mapping the reference element
          (unit cube) to the actual element
    Data: Quadrature weights {w_1, ..., w_n}
    # Loop over quadrature points
    foreach j ∈ {1, ..., n} do
        Set: (JxW)_j = J_{K_i} × w_j
        # Loop over test functions
        foreach v ∈ v_h(K_i) do
            Data: l - index of v in the global system, i.e. row in 3.43 - 3.45
            # Loop over basis functions
            foreach u ∈ v_h(K_i) do
                Data: m - index of u in the global system, i.e. column in 3.43
                a_lm  +=  (JxW)_j  a_lmij
            b_l  +=  (JxW)_j  b_lij
# 2 - Loop over edges
foreach Γ_ij ∈ T_h do
    Data: Quadrature points {x_1^ij, ..., x_{n_f}^ij}
    Data: Jacobian of the mapping J_{K+} = J_{K-} mapping the reference face
          (unit square) to the actual face, where K+, K- are elements
          adjacent to Γ_ij if this is an internal edge, or K+ = K- if this is a
          boundary edge.
    Data: Quadrature weights {w_1, ..., w_{n_f}}
    # Loop over quadrature points
    foreach j_f ∈ {1, ..., n_f} do
        Set: (JxW)_{j_f} = J_K+ × w_{j_f}   # Here it does not matter if we
         choose J_{K+} or J_{K-}
        # Loop over test functions
        foreach v ∈ v_h(K+) do
            Data: l - index of v in the global system, i.e. row in 3.43 - 3.45
            b_l  +=  (JxW)_{j_f}  b'_{lij j_f}
```

An important remark needs to be added here, with respect to the fact, that we aim for a distributed computation. As stated before, the domain decomposition approach leads to each of the processors $P_i \in \{P_0, ..., P_N\}$ of the total number of processors employed for the computation owning only a subset of elements $\{K \in T\}_i$. If the processor P_i did not have any data about other elements, we would not be able to perform the evaluation involving b' defined in 3.55, or more precisely g' defined in 3.53 as the values $\boldsymbol{\Psi}_h^k|_{ji}$ for edges Γ_{ij} of elements from $\{K \in T\}_i$ will not

always be there.

To amend this situation, in the distributed triangulation, each processor holds not only the mesh element data it owns, but also data for mesh elements it needs - in Discontinuous Galerkin method, for exactly the purposes of internal face integral evaluation described in the previous paragraph. This data is called *ghost elements*, or *ghost cells*, as these are only copies provided by other processors which own the particular elements.

TODO obrazek domain decomposition s ghost cell layer

Note that in the described algorithm, and also with respect to the remarks in the previous paragraphs, periodic boundary conditions are not specifically handled, as they are only a technically more complex case of internal edges (both from the perspective of necessity of utilizing ghost cells, and from the perspective of evaluating b_l in 2.

3.9 Time-stepping and linearization

3.9.1 Time-stepping

The simple time discretization scheme that we use in 3.5.1 on the page 34 allows us to simply implement the time-stepping in the following fashion:

Algorithm 3: Time-stepping procedure

Set: $y_0 =$ (initial solution)
Set: $ts = 1$ # initial time step
Set: $t = 0.00$ # initial time
# Loop over time steps
for ; $t < T$; $t = t + \tau$, $ts = ts + 1$ **do**
  Data: Solution from the previous time step y^{ts-1}
  1.Call procedure 2 to obtain $A, b\left(y^{ts-1}\right)$
  2.Solve the problem $Ay^{ts} = b\left(y^{ts-1}\right)$ # See note below
  3.(If necessary) postprocess y^{ts}
  4.Calculate updated value of τ

Note The step 2 (solving the algebraic problem) is of course a key point in the overall process. Because of its importance, the aim of this work is not to describe, or even implement an algebraic solver for this purpose. Many scientific teams have spent many years on publicly available open-source solvers that are usable by the software we develop for the purpose of solving the MHD phenomena. We use the existing solvers.

3.10 Performance considerations

3.10.1 Parallelization

To be able to perform any large scale calculations, we need to be able to utilize the performance of hardware at maximum. Being able to use modern, multi-core computers is an absolute must to achieve good performance, as the execution time when using parallel execution can decrease by a factor of corresponding to the number of cores - and modern machines have tens of cores available.

The parallelization is possible at several places in the overall algorithm 2. But it makes most sense to parallelize the outer-most loop over elements, and over edges.

Another point for parallelization is the algebraic solver. As explained in 3.9.1, we rely on existing software packages for finding the solution of the algebraic system 3.46 - all the used solvers support and heavily utilize parallelization.

3.10.2 Vectorization

Similarly as in section 3.10.1, the goal here is to be able to solve the discretized problem in the most efficient (fastest) way possible. One of the features that (modern) hardware offers is to employ vectorization instructions - i.e. unary or binary instructions that operate on $N, N > 1$ values (in case of unary instructions), or $N, N > 1$ pairs of values (in case of binary instructions) at the same time - the number N depends on the capabilities of the CPU, and on the precision (single or double). On the hardware that was available to perform calculations for this thesis (and based on always-used double precision), $N = \{4, 8\}$, using such instructions requires both using them in the code (one has to specify that these instructions shall be generated explicitly), and having the compiler aware of these, and able to utilize them in the generated machine code - both compilers used for work on this thesis (GNU gcc, and Microsoft Visual Studio) support vectorization instructions (SSE, SSE2, AVX, AVX2).

Vectorization helps heavily with respect to CPU time spent on calculation. In the algorithm 2, vectorized instructions are used in:

- evaluation of the expressions $a_{lm} += JxW_{\boldsymbol{j}}\ a_{lmK\boldsymbol{j}}$
- evaluation of expressions $b_l += JxW_{\boldsymbol{j}}\ b_{lK\boldsymbol{j}}|_{edge|volumetric}$
- calculation of $JxW_{\boldsymbol{j}}$

3.10.3 Distribution

- AZ DO PLNE PRACE

4 Adaptive Mesh Refinement

As the Adaptive Mesh Refinement (AMR) is a very important algorithm in the overall numerical solution, handling the multi-scale aspect of the studied problems, in this chapter, a description of what needs to be taken care of for the DG method, in the distributed settings, and what concrete decisions were made during this work, and justification of these in light of the requirements on the numerical solution.

Note that in what follows, the symbol T or $T_{i\in\{0,1,2,\dots\}}$ shall denote a general computational mesh and a particular mesh in the series of refined meshes respectively. We shall drop the subscript h from T_h referring to the dimension (maximum of diameters of all elements) of the mesh where possible.

4.1 Overview of the AMR

The general schema of any Adaptive Mesh Refinement algorithm is described here:

Algorithm 4: Generic AMR algorithm

Data: Mesh T_0
Result: A mesh T_n and a solution $\boldsymbol{y}_n$ on this mesh satisfying the solution acceptance criteria

```
i = 0
while true do
    obtain solution y_i on T_i
    evaluate solution y_i acceptance criteria
    if solution acceptance criteria satisfied then
        return
    else
        identify subset T_i^r of all elements K ∈ T_i to be refined, T_i^r ⊆ T_i
        obtain T_{i+1} by refining (at least) all K ∈ T_i^r
        i = i + 1
```

In 4, *solution acceptance criteria* is usually either spatial error estimate threshold, or number of elements, etc. In the same description, the note that there might be other elements $K \notin T_i^r$ refined in order to maintain some desired attributes of the mesh, such as 1-irregularity (the refinement level difference of two elements sharing a common edge / face is at most one), smoothness of mesh (there is e.g. no unrefined elements for which all, or a majority of neighboring elements would be refined), etc.

Overview of the AMR

Since we are dealing with evolution equations, it is necessary to specify how the AMR algorithm relates with the non-AMR solution algorithm 3. There are several points we need to take into considerations:

- slope limiting as a postprocessing step after the solution must not be omitted in case of higher-order (e.g. linear) basis functions
- the solution needs to be transferred to the refined mesh in order to be able to assemble the matrix and the right-hand side on the refined mesh in the next adaptivity iteration
- since the solution evolves and the refinements that contribute to error reduction at time step n do not contribute to error reduction at time step $n+m$ (the solution was e.g. oscillating or potentially discontinuous at time step n, but is smooth at time step $n+m$), we also want to revert such refinements as the simulation time progresses, we call this process *coarsening* of elements. For this we shall define a set T_i^c of all elements to be coarsened.

The algorithm looks like this:

Algorithm 5: AMR for time-discretized problems

Set: $y_0 =$ (initial solution)
Set: $ts = 1$ # initial time step
Set: $t = 0.00$ # initial time
# Loop over time steps
for ; $t < T$; $t = t + \tau$, $ts = ts + 1$ **do**
  Data: Solution from the previous time step y^{ts-1} expressed on the mesh T_0^{ts}
  i = 0
  while *true* **do**
    call procedure 2 to obtain $A_i, b_i\left(y_i^{ts-1}\right)$ on the mesh T_i^{ts}
    solve the problem $A_i y_i^{ts} = b_i\left(y_i^{ts-1}\right)$
    postprocess the solution y_i^{ts}
    evaluate solution y_i^{ts} acceptance criteria
    if *solution acceptance criteria satisfied* **then**
      $y^{ts} = y_i^{ts}$
      break While loop
    else
      identify subset T_i^r of all elements $K \in T_i^{ts}$ to be refined, $T_i^r \subseteq T_i^{ts}$
      identify subset T_i^c of all elements $K \in T_i^{ts}$ to be coarsened, $T_i^c \subseteq T_i^{ts}$
      obtain T_{i+1}^{ts} by refining (at least) all $K \in T_i^r$ and coarsening a subset of T_i^c
      transfer the solution y_i^{ts} onto T_{i+1}^{ts}
      i = i + 1
  calculate updated value of τ

The critical points of the algorithm are

- solution acceptance criteria evaluation,
- identification of subset T_i^r,
- identification of subset T_i^c.

For the last two points, we shall consider a function r

$$r: \ T_i^{ts} \to [0, +\infty), \tag{4.1}$$

which shall be called *refinement indicator*, and the set $T_i^r = \{K^r\}$ shall be then defined as a set of all such elements for which one of these criteria are satisfied:

- $r\left(K^r\right) > \alpha \cdot \max\left\{r\left(K\right) \ | \ K \in T_i^{ts}\right\}$, or
- $r\left(K^r\right) > \beta \cdot \sum\left\{r\left(K\right) \ | \ K \in T_i^{ts}\right\}$.

The set $T_i^c = \{K^c\}$ shall be defined similarly as

- $r\left(K^c\right) < \gamma \cdot \max\left\{r\left(K\right) \ | \ K \in T_i^{ts}\right\}$, or
- $r\left(K^c\right) < \delta \cdot \sum\left\{r\left(K\right) \ | \ K \in T_i^{ts}\right\}$.

Note that the parameters $0 < \gamma \leq \alpha < 1, \ 0 < \delta \leq \beta < 1$ are artificial, and do not affect the overall solution quality (as the solution acceptance criteria has to be met independently of choices of their values). Nevertheless, the choices may affect performance - e.g. for a high α, β, many steps are needed in order to satisfy the solution acceptance criteria, and on the other hand, if values are too low, there is an unnecessary high number of degrees of freedom that do not contribute substantially to the reduction of the solution error.

Please also note, that description of edge cases, and limitations are not given here for brevity. These cases include e.g. what happens if an element is both selected for coarsening and refinement, or how we maintain a 'minimal' refinement level so that we do not coarsen beyond a rational limit.

4.2 Adaptive-mesh refinement and DG

With the Discontinuous Galerkin method, using AMR brings additional complexity into the evaluation of face integrals described in 3.53, 3.55. To describe the process in detail, definition of the numerical flux 3.15 needs to be taken into account. What is evaluated during the assembling procedure 2 is the term 3.55. In order to evaluate

TODO obrazky moznych pripadu sousedicich elementu, jak zassemblovat

TODO zminka o peridickych podminkach, ze to je zase o tom samem, pridat nekam

4.2.1 Relationship with slope limiters

TODO tady hlavne ze ten relativne drahy napocet sousedu ve vrcholu (je drahy proto, ze na to klasicke struktury nejsou delane) musim tedy delat rozumne, pak ze samozrejme musim respektovat refinement a zajimaji me jen aktivni elementy

4.3 Reference solution approach

As the aim of this work is to prepare a universally usable solver for MHD problems, the refinement indicator 4.1 must ideally not be dependent on any attributes of the solved problem data (initial condition, boundary conditions, physical quantities such as γ, etc.).

5 Results

In this section, results from the computation using the implemented software are presented. There are two classical benchmarks for 3-dimensional MHD equations, namely the MHD Blast [?], [?], and the Orszag-Tang vortex [10]. And then the main section of this work contains results from the flux rope eruption model on and above the Sun's surface.

5.1 Benchmarks

The benchmarks presented in this Section do not have an exact analytical solution, but the formation of waves and discontinuities is well studied, and benchmarking is usually performed on the basis of comparing the structure and presence of non-physical attributes.

5.1.1 MHD Blast

5.1.2 Orszag-Tang vortex

TODO - ze clanku papers- Londrillo.pdf vzit stranu 30 a napocitat to stejny

5.2 Flux rope eruption model

This model is based on the original Titov-Demoulin model from [?], as used in [?].

The model parameters are as follows. Note that k_B is the Boltzmann constant $k_B = 1.38064852 \times 10^{-23} \frac{\mathrm{J}}{\mathrm{K}}$, m_p is the plasma mass, and g gravitational acceleration.

$$\beta = 0.05 \quad \text{... Plasma beta} \tag{5.1}$$

$$L_G = 2.0\ k_B \frac{T_{ext}}{(m_p g)} = 1.2 \times 10^8\ [\text{m}] \tag{5.2}$$

$$L_G = 20.0 \quad \text{... Coronal height scale in dimension-less units} \tag{5.3}$$

$$\tag{5.4}$$

$$\text{!!!} \quad \text{TODO - Proc je v kodu} L_G = 0.0\ ? \tag{5.5}$$

$$\tag{5.6}$$

$$N_t = -3.0 \quad \text{... Torus winding number} \tag{5.7}$$

$$R = 4.0 \quad \text{... Torus major radius} \tag{5.8}$$

$$2L = 4.0 \quad \text{... Magnetic charge separation distance} \tag{5.9}$$

$$d = 2.0 \quad \text{... Geometrical factor} \tag{5.10}$$

$$q_{mag} = \left| \frac{\ln\left(8.0 e^{-5/4} R\right)}{4} N_t \left(\frac{L}{R}\right)^2 \left[1 + \left(\frac{R}{L}\right)^2\right]^{3/2} \right| \tag{5.11}$$

$$q_{mag} \approx \text{Normalised magnetic charge corresponding to global equilibrium} \tag{5.12}$$

$$q_{mag} = \frac{\ln(8R) - \frac{5}{4}}{4} |N_t| \frac{\left(1 + \left(\frac{L}{R}\right)^2\right) \sqrt{1 + \left(\frac{L}{R}\right)^2}}{\frac{L}{R}} \tag{5.13}$$

$$\tag{5.14}$$

$$\text{!!!} \quad \text{TODO - Neni toto spatne nakodene?} \tag{5.15}$$

$$\tag{5.16}$$

$$H = 2.0\ \frac{N_t^2}{R^2} \quad \text{... "Helicity" factor inside tho loop} \tag{5.17}$$

$$\frac{T_{ext}}{T_{in}} = 10.0 \quad \text{... Coronal/prominence temperature ratio} \tag{5.18}$$

$$\tag{5.19}$$

$$\text{!!!} \quad \text{TODO - Proc je v kodu Tc2Tp = 1.0?} \tag{5.20}$$

$$\tag{5.21}$$

$$\tag{5.22}$$

6 Conclusion, outlook

Further work will focus on real-world industrial and astrophysical problems, mainly study of the magnetic field reconnection - [2], and other phenomena occurring both in solar physics and in industrial applications of plasma flow. Before the implemented software package is ready to be used for solution of such problems, it needs to be improved in several ways - first, the benchmark shown in this work clearly showed, that a more stable numerical scheme needs to be implemented. Second, the performance needs to be tweaked by the use of distributed computations. This will need to employ some of the following techniques:

- Second-order scheme for the discretization of the time derivative
- Adaptive algorithm for the discretization of the time derivative
- Shock-capturing for high-order DG scheme to prevent non-physical oscillations
- Adaptive algorithm for the discretization of the space derivatives communication
- A specific shapeset of basis and test functions based on Taylor expansions.

After the implementation is stable for the benchmark presented here, and as well as additional benchmarks (such as Orszag-Tang Vortex - [10]), next steps are distributed calculation of larger real-world problems and further work on performance improvements. Performance improvements will include some of the following:

- Caching of values that are necessary in multiple spaces of the algorithm (utilizing the RAM)
- Further use of vectorization for evaluation of integral quantities
- Distributed computation using the Message Passing Interface
- Optimal selection of both solver, and preconditioner for the solution of the resulting algebraic problems

Step after that is adding of additional physics into the implementation, mainly in the form of magnetic reconnection models. From the implementation point of view, additional physics usually implies extending the matrix assembly routines by evaluating more physical quantities, possibly in a different setup (e.g. point values, particle-based quantities, etc.), or either post-processing or iterating the results in order to add some new rules and relations that must hold for the solution.

Plan of work:

Quarter	Activities
Q2/2016	Stabilize the algorithm, achieve both stable, reliable, and fast calculation of 2 - 3 benchmarks
Q3/2016	Implementing more numerical fluxes, performance improvements
Q4/2016	Adding physics (e.g. magnetic reconnection), start with real-world problems
Q1+Q2/2017	Distributed computations, adaptive algorithms

Bibliography

[1] Bangerth, W., D. Davydov, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and D. Wells (????), "The `deal.II` library, version 8.4." *preprint*.

[2] Bárta, Miroslav, Jörg Büchner, Marian Karlický, and Jan Skála (2011), "Spontaneous current-layer fragmentation and cascading reconnection in solar flares. i. model and analysis." *The Astrophysical Journal*, 737, 24, URL `http://stacks.iop.org/0004-637X/737/i=1/a=24`.

[3] Davis, T. (2006), *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, URL `http://epubs.siam.org/doi/abs/10.1137/1.9780898718881`.

[4] Kestener, P. et al. (1992), "Ramses-gpu." URL `http://www.maisondelasimulation.fr/projects/RAMSES-GPU/html/`.

[5] Korous, Lukas and Pavel Solin (2012), "An adaptive $$hp$$ -dg method with dynamically-changing meshes for non-stationary compressible euler equations." *Computing*, 95, 425–444, URL `http://dx.doi.org/10.1007/s00607-012-0257-1`.

[6] Ma, Zhonghua, Lukas Korous, and Erick Santiago (2012), "Solving a suite of {NIST} benchmark problems for adaptive {FEM} with the hermes library." *Journal of Computational and Applied Mathematics*, 236, 4846 – 4861, URL `http://www.sciencedirect.com/science/article/pii/S0377042712000611`. {FEMTEC} 2011: 3rd International Conference on Computational Methods in Engineering and Science, May 9–13, 2011.

[7] Miyoshi, T. and K. Kusano (2005), "A multi-state HLL approximate Riemann solver for ideal magnetohydrodynamics." *Journal of Computational Physics*, 208, 315–344.

[8] Mocz, P., M. Vogelsberger, D. Sijacki, R. Pakmor, and L. Hernquist (2014), "A discontinuous Galerkin method for solving the fluid and magnetohydrodynamic equations in astrophysical simulations." *mnras*, 437, 397–414.

[9] Norman, M., J. Wilson, et al. (1992), "Zeus." URL `http://www.astro.princeton.edu/~jstone/zeus.html`.

[10] Orszag, Steven A. and Cha-Mei Tang (1979), "Small-scale structure of two-dimensional magnetohydrodynamic turbulence." *Journal of Fluid Mechanics*, 90, 129–143, URL http://journals.cambridge.org/article_S002211207900210X.

[11] Rossmanith, J. A. (2013), "High-Order Discontinuous Galerkin Finite Element Methods with Globally Divergence-Free Constrained Transport for Ideal MHD." *ArXiv e-prints*.

[12] Skala, J. and M. Barta (2012), "LSFEM implementation of MHD numerical solver." *ArXiv e-prints*.

[13] Skála, J., Baruffa, F., Büchner, J., and Rampp, M. (2015), "The 3d mhd code goemhd3 for astrophysical plasmas with large reynolds numbers." *A&A*, 580, A48, URL http://dx.doi.org/10.1051/0004-6361/201425274.

[14] Solin, Pavel, Lukas Korous, and Pavel Kus (2014), "Hermes2d, a c++ library for rapid development of adaptive -fem and -dg solvers." *Journal of Computational and Applied Mathematics*, 270, 152 – 165, URL http://www.sciencedirect.com/science/article/pii/S0377042714000831. Fourth International Conference on Finite Element Methods in Engineering and Sciences (FEMTEC 2013).

[15] Stone, J., T. Gardiner, P. Teuben, et al. (2008), "Athena." URL https://trac.princeton.edu/Athena/.