Stabilized discontinuous Galerkin methods for solving hyperbolic conservation laws on grids with embedded objects

---

Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

Der Fakultät für Mathematik der

Technischen Universität Dortmund

vorgelegt von

Florian Streitbürger

im Mai 2023

Dissertation

Stabilized discontinuous Galerkin methods for solving hyperbolic conservation laws on grids with embedded objects

Fakultät für Mathematik
Technische Universität Dortmund

Mitglieder der Prüfungskommission:

Erstgutachterin:	Assoc. Prof. Dr. Sandra May
Zweitgutachter:	Prof. Dr. Stefan Turek
Drittgutachter:	Prof. Dr. Christian Engwer

Tag der mündlichen Prüfung: 28. August 2023

Abstract

This thesis covers a novel penalty stabilization for solving hyperbolic conservation laws using discontinuous Galerkin methods on grids with embedded objects. We consider cut cell grids, that are constructed by cutting the given object out of a Cartesian background grid. The resulting cut cells require special treatments, e.g., adding stabilization terms. In the context of hyperbolic conservation laws, one has to overcome the *small cell problem*: standard explicit time stepping becomes unstable on small cut cells when the time step is selected based on larger background cells.

This work will present the *Domain of Dependence* (DoD) stabilization in one and two dimensions. By transferring additional information between the small cut cell and its neighbors, the DoD stabilization restores the correct domains of dependence in the neighborhood of the cut cell. The stabilization is added as penalty terms to the semi-discrete scheme. When combined with a standard explicit time-stepping scheme, the stabilized scheme remains stable for a time-step length based on the Cartesian background cells. Thus, the small cell problem is solved.

In the first part of this work, we will consider one-dimensional hyperbolic conservation laws. We will start by explaining the ideas of the stabilization for linear scalar problems before moving to non-linear problems and systems of hyperbolic conservation laws. For scalar problems, we will show that the scheme ensures monotonicity when using its first-order version. Further, we will present an L^2 stability result. We will conclude this part with numerical results that confirm stability and good accuracy. These numerical results indicate that for both, linear and non-linear problems, the convergence order in L^∞ norm for smooth tests is $O(h^{p+1})$ when using polynomials of degree p.

In the second part, we will present first ideas for extending the DoD stabilization to two dimensions. We will consider different simplified model problems that occur when using two-dimensional cut cell meshes. An essential step for the extension to two dimensions will be the construction of weighting factors that indicate how we couple the multiple cut cell neighbors with each other. The monotonicity and L^2-stability of the stabilized system will be confirmed by transferring the ideas of the proof from one to two dimensions. We will conclude by presenting numerical results for advection along a ramp, demonstrating convergence orders of $O(h^{p+1/2})$ to $O(h^{p+1})$ for polynomials of degree p. Additionally, we present preliminary results for the two-dimensional Burgers and Euler equations on model meshes.

Acknowledgements

First and foremost, I express my deepest gratitude to my advisor, Professor Sandra May, for the opportunity to participate in this project and for trusting me to be her first Ph.D. student. Her guidance, attention to detail, and constructive feedback have been instrumental in shaping my research and helping me overcome various challenges throughout my academic journey. Sandra encouraged me and gave me the possibility to attend numerous conferences and workshops, which allowed me to present and discuss my research in a scientific environment, meet various interesting people, and grow professionally and personally. Above all, I cannot thank Sandra enough for her mentorship and her continuous support.

I also sincerely thank Professor Christian Engwer for his invaluable contributions to my scientific research and for collaborating with me over the past few years. His continuous support and expertise with DUNE have been a great help and have allowed me to overcome many challenges. Let me also express my gratitude to him for acting as a reviewer of this thesis. Additionally, I would like to recognize his student Gunnar Birke for his important contributions to this project.

Moreover, I would like to thank Professor Stefan Turek for not only laying the foundation of my scientific education in the field of numerical methods for partial differential equations but also for creating a truly remarkable working atmosphere at the Institute of Applied Mathematics at TU Dortmund University. I would also like to extend my appreciation to all members of LSIII. Without them, this excellent working atmosphere would not have been possible, and I feel genuinely privileged to have worked with such supportive colleagues.

A special thanks go to my office mates Hannes, Michael, Katharina, and Rida for all the entertainment and emotional support. Thank you for not just being coworkers, but friends, and for making the time in the office a memorable experience. Furthermore, I would like to thank Dr. Christoph Lohmann for always offering his help and proofreading this thesis.

In addition, financial support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 439956613 (Hypercut) is gratefully acknowledged.

Finally, I would like to thank my friends, especially Linus and Fabian, and my family, including my parents, Heike and Friedhelm, my two brothers, Bastian and Philipp, and my grandmother Gisela. Thank you for celebrating my achievements and milestones with me and for being a constant source of motivation and inspiration.

Lastly, I would like to thank Hanna for being my partner and best friend, for her endless love, and for enriching my life in countless ways.

Dortmund, May 2023

Florian Streitbürger

Statement of Authorship

This thesis contains sections that were completed in collaboration and have been published in journals. These include:

[25] C. Engwer, S. May, A. Nüßing, and F. Streitbürger. A stabilized dG cut cell method for discretizing the linear transport equation. *SIAM J. Sci. Comput.*, 42(6):A3677–A3703, 2020

[62] S. May and F. Streitbürger. DoD stabilization for non-linear hyperbolic conservation laws on cut cell meshes in one dimension. *Appl. Math. Comput.*, 419, 2022

[76] F. Streitbürger, G. Birke, C. Engwer, and S. May. DoD stabilization for higher-order advection in two dimensions. In Melenk et al. [63]

Chapter 3 is largely based on [62], while parts of Chapter 4 are based on [25] and [76].

List of Figures

List of Tables

Contents

1

Introduction

Motivation

In recent years, technology has advanced to the point where it is possible to simulate complex real-world applications as shown in Figure 1.1. Such complex geometries can make the meshing process a very challenging task. The standard way of constructing body-fitted meshes for these problems is to use an unstructured triangularization. Although unstructured meshes can be more flexible than structured meshes, they also have some challenges that must be considered. The construction of an unstructured mesh that matches the ge-

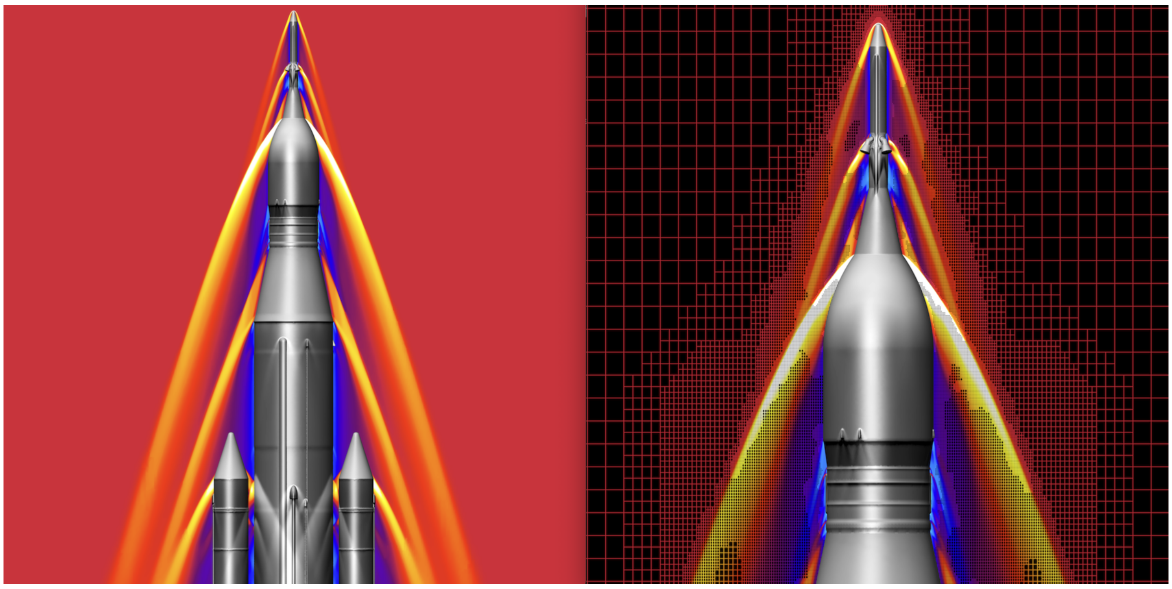


Figure 1.1: Left: Simulation of an SLS rocket from the NASA Artemis project. Right: Part of corresponding cut cell mesh for the rocket generated using the Cart3D software.[1]

[1]Picture taken from Cart3D website https://www.nas.nasa.gov/publications/software/docs/cart3d/

ometry of the problem domain is a challenging task and can therefore be difficult and time-consuming, especially for complex geometries. In addition, adapting the unstructured grid to changes in the problem domain can be challenging, requiring advanced algorithms and the manipulation of complex data structures.

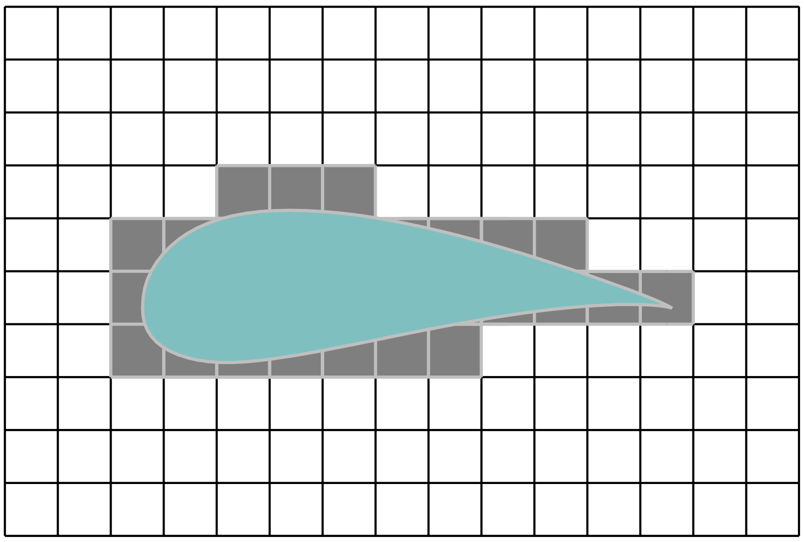


Figure 1.2: Example of a cut cell grid for an airfoil in two dimensions. The cut cells are highlighted in grey near the embedded boundary.[2]

An alternative way of constructing an accurate mesh for these complex problems is given by so-called *cut cell meshes* or *embedded boundary meshes*. These approaches can be a powerful tool to handle the complicated meshing process in numerical simulations. One of its most significant benefits is that the construction of cut cell meshes is fairly straightforward and does not require advanced computational costs. The details of how the embedded geometry is represented vary. We will focus on the following approach: We consider a Cartesian background mesh and cut out the given geometry of the problem resulting in a cut cell mesh, as illustrated in Figure 1.2. Where the object intersects the background cells, this results in so-called *cut cells*, which accurately represent the boundary of the given problem without requiring complicated meshing procedures. These cut cells are highlighted in gray in Figure 1.2. Away from the cut boundary the resulting cut cell mesh consists of Cartesian cells, which have valuable benefits. Cartesian grids usually improve accuracy due to the cancellation of error terms. In addition, Cartesian grids are more efficient in grid generation and more variable in the choice of applicable methods. We emphasize that Cartesian grids are generally more memory-efficient than unstructured meshes, and their regular structure makes them well-suited for parallel computing. Finally, the resulting meshes can easily be combined with adaptive mesh refinement (AMR) compared to unstructured meshes.

[2]Coordinates of airfoil taken from https://m-selig.ae.illinois.edu

All the benefits mentioned above make cut cell grids an excellent choice when solving complex problems in practice, but there are also some drawbacks that we want to address here. The problems that arise when using cut cell grids are caused by the cells at the cut boundary. The major downside about these approaches is that cut cells can become arbitrarily small. In addition, they will have cell sizes that differ by several magnitudes and have various shapes, resulting in stability issues that must be resolved. These stability issues typically differ depending on which equations are solved. As a result, different approaches have been developed to handle these problems.

Cell merging

One approach that deals with small cut cells in general and independent of the given equation are so-called *cell agglomeration* or *cell merging* methods [52, 66, 68, 71]. Cell merging solves the issues induced by the small cut cells geometrically. It combines small cut cells with adjacent cells to eliminate cells that are too small and their associated instabilities. However, as these methods reintroduce complexity to the mesh generation process and can become involved in higher spatial dimensions, they will not be considered here. We want to keep the cut cell mesh unchanged and solve the stability issues algebraically.

Elliptic and parabolic problems

In the context of algebraic solutions, we must distinguish between the different types of equations. When considering elliptic and parabolic partial differential equations (PDEs), the main issue that one has to deal with are ill-conditioned systems. One of the most prominent approaches in this setting is the ghost penalty stabilization [12, 13]. The idea is to add jump penalty terms to the formulation to regain the coercivity of the method on small cut cells. As a result, upper bounds for the condition number of the system can be found.

Small Cell Problem

This work will focus on hyperbolic PDEs, specifically time-dependent hyperbolic conservation laws. The arising problems differ compared to the ones for elliptic and parabolic PDEs. The biggest problem here is the *small cell problem*. The cause of this problem is that we want to use *explicit* time-stepping schemes for the temporal discretization. *Implicit* time-stepping schemes can be computationally too expensive for non-linear equations, which is why we avoid using them. To ensure stability, the time step size of the explicit time-stepping schemes must be chosen based on the smallest cell (in flow direction). This would result in an uncontrollable small time step size on a cut cell mesh. This is neither feasible nor what we want. Instead, we choose the time step based on the cell size of the background mesh. This requires to develop special schemes for small cut cells and their neighbors.

Hyperbolic approaches

Finite volume (FV) and discontinuous Galerkin (DG) methods are typically used to solve hyperbolic conservation laws. While using DG methods on cut cell meshes to solve hyperbolic conservation laws is relatively new, there is a long history of FV methods on cut cell meshes. In what follows, we give a summary of existing cut cell approaches that have been developed in the past decades.

One of the first approaches to tackle the small cell problem is given by the *flux redistribution* method [15, 18]. The idea is to redistribute the flux difference in the neighborhood of the small cut cell in a conservative way. This approach has been implemented in two and three dimensions for practical applications, but it is only first-order accurate on the stabilized cut cells.

Berger, Helzel, and LeVeque introduced in the early 2000s the so-called *h-box method* [7, 8, 42]. The idea of the method is that the fluxes at the edges of the small cut cells are computed by using the values from an artificially constructed cell of length h, which corresponds to the length of a Cartesian cell. As a result, the numerical method updates every cell in a physically acceptable way, leading to stability and good accuracy even in the presence of small cut cells.

Another approach is given by the *mixed explicit-implicit* method. Here, the Cartesian cells are treated with an explicit method, while an implicit method is used on the cut cells to guarantee better stability. An essential aspect of this method is the switch between the explicit and implicit time-stepping methods. May and Berger provide the *flux bounding* approach [61], which could be implemented in three dimensions.

More recently, novel and promising methods have been published. Berger and Giuliani proposed a new method called the *state redistribution* (SRD) method [6, 32], which is a post-processing technique that is applied after each time step to gain a stable scheme.

In another recent approach, Kerkmann and Helzel extended the active flux method to the setting of cut cell meshes [43, 48]. In contrast to other cut cell methods, the stability of the scheme is automatically achieved as the active flux method works with exact evolution by using the method of characteristics. At the moment, this approach is only able to solve the advection equation on cut cell meshes because it lacks an approximate evolution operator for non-linear equations.

Finally, Klein et al. propose a *dimensionally split* flux stabilization [34, 49]. The stabilization is obtained by using information of the local geometry and the wave speed.

While finite volume methods have been widely used in hyperbolic problems for decades, recent advancements in discontinuous Galerkin methods have shown promise for improving the accuracy and efficiency of simulations of hyperbolic conservation laws. In what follows, we will discuss the list of cut cell DG methods, which mainly consist of relatively novel approaches.

In the context of DG methods, different approaches rely on extending the ghost penalty stabilization from the elliptic to the hyperbolic setting [29, 30, 39, 74, 75]. Most of these methods focus on the conditioning problem of the system matrix, as in the elliptic case. Only Fu et al. [29, 30] extend the ghost penalty stabilization to address the small cell problem, which makes this approach more related to our case.

In a recent approach [47], Kaur and Hicken investigate how effective DG Difference (DGD) methods are on cut cell meshes, and they conclude that, unlike traditional methods, DGD methods do not face the typical conditioning problem for steady hyperbolic problems on cut cell meshes.

A very promising approach is extending the state redistribution method to the DG setting [31]. Like the FV approach, the SRD method is applied as a post-processing step. The SRD method in the context of DG works in two dimensions and shows good numerical results.

Goal of this thesis

In this work, we will present and discuss the so-called *Domain of Dependence* (DoD) stabilization, which is a novel penalty stabilization technique that solves the small cell problem and comes with valuable theoretical properties. Our approach to an algebraic solution is to add a jump penalty term to the semi-discrete formulation. We start with a standard DG discretization in space given by: Find $\mathbf{u}^h$ in a function space V_h^p such that

$$\left(d_t \mathbf{u}^h(t), \mathbf{w}^h\right)_{L^2(\Omega)} + a_h\left(\mathbf{u}^h(t), \mathbf{w}^h\right) = 0, \quad \forall\, \mathbf{w}^h \in V_h^p. \tag{1.1}$$

We then modify this formulation by adding a penalty term $J_h(\cdot,\cdot)$ to obtain the following *stabilized* semi-discrete formulation: Find $\mathbf{u}^h \in V_h^p$ such that

$$\left(d_t \mathbf{u}^h(t), \mathbf{w}^h\right)_{L^2(\Omega)} + a_h\left(\mathbf{u}^h(t), \mathbf{w}^h\right) + J_h\left(\mathbf{u}^h(t), \mathbf{w}^h\right) = 0, \quad \forall\, \mathbf{w}^h \in V_h^p. \tag{1.2}$$

By using the new stabilized semi-discrete formulation together with an explicit time-stepping scheme, we can choose a time step size based on regular-sized Cartesian cells and avoid the small cell problem. Furthermore, we will show that the stabilized scheme preserves several theoretical and numerical properties: For the first-order version, we obtain a monotone scheme independent of the size of the small cut cell. Additionally, we can show L^2 stability in the semi-discrete setting for arbitrary polynomial degrees p, and our numerical results demonstrate convergence orders of $p+1$ for smooth solutions and robust behavior for problems involving shocks.

Structure of thesis

In Chapter 2, we will start with summarizing general theory regarding hyperbolic conservation laws, which will be important throughout this work. Next, we will describe the numerical discretization approach used to solve the given problem and discuss the essential ingredients involved in the spatial and temporal approximation. At the end of this chapter, we will discuss cut cell grids and examine the small cell problem more closely.

In Chapter 3, we will introduce the DoD stabilization for the one-dimensional case. First, we will discuss the ideas of the DoD stabilization using the linear advection equation. After that, we will extend these ideas in the one-dimensional case to non-linear equations and systems of conservation laws, including the compressible Euler equations. We will conclude this chapter by investigating the stability and accuracy of the stabilized method with numerical tests in one dimension.

Finally, we will adapt the ideas we developed in one dimension to the two-dimensional case in Chapter 4. This involves some effort, as the cut cells differ significantly between one dimension and higher spatial dimensions. Additional features provided by higher spatial dimensions, e.g., an infinite number of propagation directions, need to be considered here. We will present initial numerical results for this extension to assess stability and the potential for high accuracy.

We conclude this thesis with an outlook and a discussion of potential areas for future research.

Notation

Symbol	Description
d	Spatial dimension
m	Number of equations
x	Spatial variable in 1 dimension
$\mathbf{x}$	Spatial variable in d dimensions
t	Temporal variable
T	Final time
f	Flux in 1d
$\mathbf{f}$	Flux in $\mathbb{R}^{d\times m}$
u	Conserved variable
$\mathbf{u}$	Vector of conserved variables
β	Velocity field
Ω	Domain
$\partial\Omega$	Boundary of domain
Γ_{in}	Inflow boundary
$\mathbf{n}$	Normal vector of boundary
g	Inflow boundary condition
$\mathbf{s}$	Source term
γ_x	Characteristic curve
U	Entropy function
$\mathbf{F}$	Entropy flux
$(U,\mathbf{F})$	Entropy pair
$\mathbf{A}$	Jacobian matrix of flux $\mathbf{f}$
λ_i	Eigenvalues of the Jacobian matrix and the hyperbolic problem
Λ	Diagonal matrix consisting of eigenvalues λ_i
$\mathbf{r}_i$	Right eigenvectors of Jacobian matrix
$\mathbf{R}$	Matrix containing right eigenvectors
$\mathbf{v}$	Characterstic variables
ν	CFL constant

Table 1.1: Symbols regarding the physical domain

Symbol	Description
E	Cell
e	Edge
$\mathbf{n}_e$	Normal vector of edge e
Δt	Time step size
t^n	n^{th} time step
Γ_h^{int}	Set of internal edges
Γ_h^{ext}	Set of boundary (external) edges
Γ	Set of all edges
V_h^p	Finite-Element function space of piecewise polynomial functions
p	Polynomial degree
w^h	Test function living in V_h^p
$[\![\cdot]\!]_e$	Jump on edge e
$\{\!\{\cdot\}\!\}_e$	Average on edge e
$\mathcal{H}(\mathbf{n}_e,\cdot,\cdot)$	Numerical flux function on edge e
$\mathcal{M}_h$	Triangulation / Cut cell mesh
$a_h(\cdot,\cdot)$	Standard semi-discrete operator
$\mathcal{S}_h(\cdot,\cdot)$	Semi-discrete operator regarding source term $\mathbf{s}$

Table 1.2: Symbols regarding the discretization of the problem

Symbol	Description
$C^1(\Omega)$	Space of continuously differentiable functions on Ω
$C_c^1(\Omega)$	Space of $C^1(\Omega)$ functions with compact support
$L^p(\Omega)$	Space of L^p Lebesgue-integrable functions on Ω
$\mathbf{a}\cdot\mathbf{b}$	Scalar product of two vectors $\mathbf{a},\mathbf{b}\in\mathbb{R}^d$
$(\cdot,\cdot)_{L^2(\Omega)}$	Scalar product on the space $L^2(\Omega)$
$\|\cdot\|_{L^p(\Omega)}$	Norm on the space $L^p(\Omega)$

Table 1.3: Sets and norms

Symbol	Description
E_{k_1}	Small cut cell in 1D model problem
E_{cut}	Small cut cell in 2D model problems
α	Volume fraction of cut cell volume and Cartesian volume
α_E	Capacity of a cut cell E in 2D
h	Cell length of Cartesian cell in 1D
I_{equi}	Index set of all cells of length h in 1D
I_{all}	Index set containing all cells in 1D
I_{Neigh}	Index set containing small cut cell and its neighbors in 1D
$J_h(\cdot,\cdot)$	DoD Stabilization terms
$J_h^0(\cdot,\cdot)$	DoD Stabilization edge terms
$J_h^1(\cdot,\cdot)$	DoD Stabilization volume terms
$\mathcal{L}_E^{\text{ext}}(\cdot)$	Extension operator from cell E to whole domain
η_{k_1}	Stabilization parameter for cell k_1
$\mathcal{H}_a(\mathbf{n}_e,\cdot,\cdot)$	Jacobian of numerical flux with respect to the first argument
$\mathcal{H}_b(\mathbf{n}_e,\cdot,\cdot)$	Jacobian of numerical flux with respect to the second argument
$\mathbf{K},\mathbf{L},\mathbf{R}$	Parameter matrices of DoD stabilization
$\mathbf{I}^m$	Identity matrix of size $m \times m$
γ	Angle between cut and x-axis in 2D model problem
x_0	Starting point of cut in 2D model problem
$\omega_{i,j}$	Weights coupling edge e_i and edge e_j in DoD stabilization term

Table 1.4: Symbols regarding the DoD stabilization

2 Theoretical and numerical aspects of hyperbolic conservation laws

2.1. Theory of hyperbolic conservation laws

Time-dependent hyperbolic conservation laws are systems of partial differential equations that describe processes in which physical quantities are neither created nor destroyed but conserved. In practice, they are pretty versatile and are used in numerous scientific fields. One of the most significant areas of application of conservation laws is aerodynamics or, more generally *computational fluid dynamics* (CFD), where they are used to model and predict the behavior of a fluid in complex geometries. In real-world applications, this can be the flow around an object such as an aircraft or the blades of a turbine. The conserved quantities in these settings are often the mass, the momentum, or the energy.

Another important application area of hyperbolic conservation laws is traffic modeling, like road traffic or pedestrian flows. Traffic planning committees can use these models to optimize the construction of roads and improve traffic light circuits. In addition, traffic modeling can be used to predict the behavior of groups of pedestrians to improve safety at large public events. For further information regarding the fields of application, we refer to LeVeque [56, 57].

Following [44], we can derive the classical PDE for hyperbolic conservation laws in the following way: We consider an arbitrary but fixed domain $\widetilde{\Omega} \subset \Omega \subset \mathbb{R}^d$, $d \in \{1,2\}$ and a conserved quantity Q which exists in Ω. This quantity Q at time t can be described with the help of a density u as

$$Q(t) = \int_{\widetilde{\Omega}} u(x,t)\mathrm{d}x.$$

The total amount of Q in the fixed domain $\widetilde{\Omega}$ is conserved and can only be changed by incoming or outgoing fluxes of Q across the boundary. This means that the rate of change of the quantity Q is equal to the total flow of Q across the boundary. In a more mathematical terminology using the *flux function* $\mathbf{f}$, the temporal derivative ∂_t and the unit outward normal vector $\mathbf{n}$, we can summarize this observation as:

$$\partial_t \int_{\widetilde{\Omega}} u(\mathbf{x},t)\mathrm{d}\mathbf{x} = -\int_{\partial\widetilde{\Omega}} \mathbf{f}(u)\cdot\mathbf{n}\,\mathrm{d}s. \tag{2.1}$$

Here, $\mathbf{a}\cdot\mathbf{b}$ denotes the scalar product of $\mathbf{a}$ and $\mathbf{b}$ in $\mathbb{R}^d$. We simplify equation (2.1) by using the Gauss divergence theorem and by changing the order of integration in space and differentiation in time to obtain:

$$\int_{\widetilde{\Omega}} (\partial_t u(x,t) + \nabla\cdot\mathbf{f}(u))\,\mathrm{d}\mathbf{x} = 0.$$

Since the fixed domain $\widetilde{\Omega}$ is an arbitrary subset of Ω, we can neglect the integration to receive the final equation (2.2) that holds true in Ω:

$$\partial_t u(\mathbf{x},t) + \nabla\cdot\mathbf{f}(u) = 0. \tag{2.2}$$

In general, one can classify hyperbolic conservation laws as linear or non-linear and whether they are scalar equations or a system of equations. In what follows, we will first introduce scalar conservation laws, which are often the first step when developing novel numerical methods. We will discuss essential concepts such as the methods of characteristics and different definitions of solutions. In the next section, we will cover systems of hyperbolic conservation laws.

2.1.1. Scalar equations

Scalar hyperbolic conservation laws are given by

$$\partial_t u(\mathbf{x},t) + \nabla\cdot\mathbf{f}(u(\mathbf{x},t)) = 0 \quad \text{in } \Omega\times(0,T) \tag{2.3}$$

with a conserved variable $u:\mathbb{R}^d\times(0,T)\to\mathbb{R}$, $d\in\{1,2\}$ and a flux function $\mathbf{f}:\mathbb{R}\to\mathbb{R}^d$. Furthermore, the spatial domain is given by $\Omega\subset\mathbb{R}^d$ and the final time is denoted with $T\in\mathbb{R}^+$. At this point, we emphasize the notation that we will use in the course of this work. From now on, we will use the non-bold notation for a variable z to indicate that this variable is clearly a scalar. In contrast to this, we will use the **bold** notation for a variable $\mathbf{z}$ that is a vector or for the case where the variable can be either a scalar or a vector.

We need to define initial data

$$u = u_0 \qquad \text{on } \Omega \times \{t = 0\}$$

and boundary conditions at the inlet

$$u = g \qquad \text{on } \Gamma^{in} \times (0, T)$$

to complete the formulation. Here, we denote by $\mathbf{n} \in \mathbb{R}^d$ the outer unit normal vector on $\partial\Omega$, and by

$$\Gamma^{\text{in}} := \{\mathbf{x} \in \partial\Omega : \partial_u \mathbf{f}(u(\mathbf{x}, t)) \cdot \mathbf{n}(\mathbf{x}) < 0\} \tag{2.4}$$

the inflow boundary of the domain.

For sufficiently smooth $\mathbf{f}$ and u we can rewrite equation (2.3) in the quasi-linear form

$$\partial_t u + \sum_{i=1}^{d} \partial_u f^i(u) \partial_{x^i} u = 0 \quad \text{in } \Omega \times (0, T) \tag{2.5}$$

with f^i being the ith, $i \in \{1, ..., d\}$ component of $\mathbf{f}$. Scalar equations are of significant importance in the development of new numerical methods. Most of them have been studied in the past extensively, and one can define exact solutions for them. Therefore, they can be seen as an intermediate step for more advanced problems.

In what follows, we introduce two of the most common examples of scalar hyperbolic conservation laws that we will use throughout this work.

Linear advection equation

The linear advection equation is one of the simplest examples of a linear scalar hyperbolic conservation law. It is given by

$$\partial_t u + \nabla \cdot (\beta u) = 0 \tag{2.6}$$

with the velocity field $\beta = \beta(\mathbf{x}, t)$. In all our cases, the velocity field must be solenoidal $\nabla \cdot \beta = 0$. In most cases, the velocity field will even be constant.

The linear advection equation is a good guide for developing new numerical methods. The equation is easy to solve and theoretically well understood, but at the same time, it shows essential properties of hyperbolic conservation laws. Therefore, the linear advection equation provides a simple example for studying the mathematical properties of hyperbolic conservation laws, such as the well-posedness and the stability of numerical methods.

In order to develop more complex models for practical applications, it is a convenient first step to understand this simple example because it can offer insights into important physical processes.

Burgers equation

The Burgers equation is a well-known non-linear scalar conservation law. The equation is given by

$$\partial_t u + \nabla \cdot \left(\frac{1}{2} u^2 \mathbf{1} \right) = 0$$

with $\mathbf{1} = (1, ..., 1)^T \in \mathbb{R}^d$. It is the fundamental model problem for non-linear hyperbolic equations that is well understood. Nevertheless, it still shows key features of conservation laws, like the generation of shocks and rarefaction waves, making it a valuable model for studying fundamental phenomena when constructing new numerical methods.

Method of characteristics

The *method of characteristics* is a powerful tool to find exact solutions to hyperbolic conservation laws. In this process, the scalar partial differential equation is rewritten into a system of ordinary differential equations. For simplicity and a better visual understanding, we will explain the method of characteristics using a scalar conservation law in 1D

$$\partial_t u + \partial_x(f(u)) = \partial_t u + \partial_u(f(u))\partial_x u = 0 \qquad \text{in } \Omega \times (0,T) \tag{2.7}$$

$$u(x,0) = u_0(x) \qquad \text{on } \Omega \times \{0\}. \tag{2.8}$$

The idea is to find the solution $u(x,t)$ at a given point $x \in \Omega$ and a time t using a specific curve γ_{x_0} that connects (x,t) with a starting point $(x_0, 0)$ for some $x_0 = x_0(x,t) \in \Omega$. These curves $\gamma_{x_0} : (0,T) \to \Omega$ are called characteristics if they satisfy the special property that

$$\partial_t \gamma_{x_0}(t) = \partial_u f(u(\gamma_{x_0}(t), t)). \tag{2.9}$$

This property will guarantee that the solution u of (2.7) is constant along the curve γ_{x_0}, as we can easily see:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} u(\gamma_{x_0}(t), t) &= \partial_x u(\gamma_{x_0}(t), t)\partial_t \gamma_{x_0}(t) + \partial_t u(\gamma_{x_0}(t), t) \\ &= \partial_x u(\gamma_{x_0}(t), t)\partial_u f(u(\gamma_{x_0}(t), t)) + \partial_t u(\gamma_{x_0}(t), t) \\ &\overset{(2.7)}{=} 0 \end{aligned}$$

This implies that if we know the solution at $(x_0, 0)$ (which is given by the initial data $u_0(x_0)$), we know the solution along the characteristic curve γ_{x_0} that starts at x_0. For a point (x,t) that is connected to $(x_0, 0)$ through γ_{x_0}, there holds

$$u(x,t) = u(x_0, 0) = u_0(x_0). \tag{2.10}$$

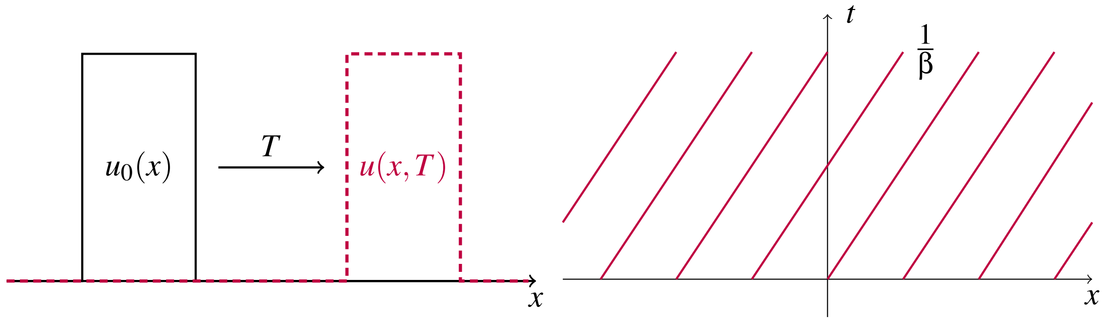


Figure 2.1: Left: Linear advection equation with discontinuous hat function as initial data (black) and solution at time T (red). Right: Corresponding characteristics in an x-t-diagram with slope $\frac{1}{\beta}$.

In general, we can use equations (2.9) and (2.10) to derive the following identity

$$\gamma_{x_0}(t) = \partial_u f(u_0(x_0))t + x_0. \tag{2.11}$$

This gives us a general classical solution to a scalar conservation law using $x = \gamma_{x_0}(t)$

$$u(x,t) = u_0(x - \partial_u f(u_0(x_0))t) = u_0(x_0).$$

The method of characteristic is useful to compute the exact solution for the two examples of scalar equations we have already presented. The characteristic curve for the linear advection equation

$$\partial_t u + \beta \partial_x u = 0$$

and a given point $x_0 \in \mathbb{R}$ can be described through the following ODE

$$\partial_t \gamma_{x_0}(t) = \beta,\ \gamma_{x_0}(0) = x_0,$$

which has the solution $\gamma(t) = \beta t + x_0$. Thus, the exact solution can be computed by

$$u(x,t) = u_0(x - \beta t).$$

In Figure 2.1, we show an example of transported mass with a positive velocity $\beta > 0$.

Analogously, we can find characteristic curves that describe the behavior of the one-dimensional Burgers equation, which is given in quasi-linear form by

$$\partial_t u + u \partial_x u = 0.$$

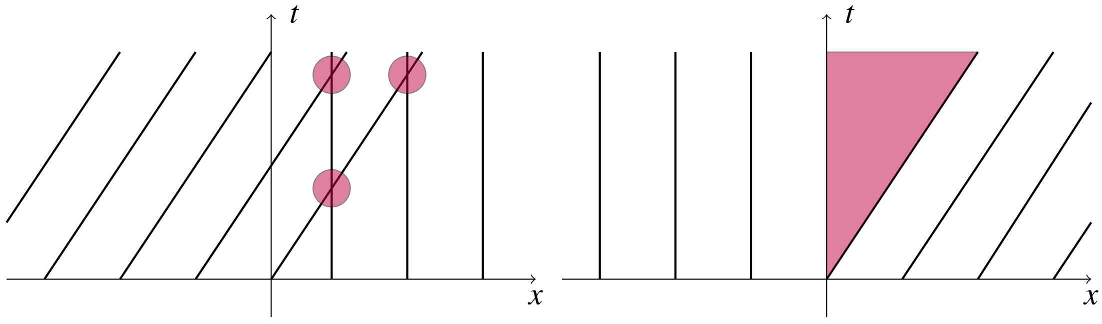


Figure 2.2: Problematic characteristic setups for the one-dimensional Riemann problem: Crossing characteristics (left) and diverging characteristics (right).

Once again, we consider a given point $x_0 \in \mathbb{R}$. For this equation, the characteristic curve $\gamma_{x_0}(t)$ satisfies the ODE

$$\partial_t \gamma_{x_0}(t) = u(\gamma(t), t),\ \gamma_{x_0}(0) = x_0.$$

This means that the characteristics are linear straight lines whose slope is determined by the initial data u_0. The fact that the initial data determine the slopes of the characteristics can lead to problems because the characteristics might cross or diverge. In Figure 2.2, two examples of Riemann problems with problematic characteristic setups are given. At these points, the solution is no longer well-defined. In the next section, we will discuss this in more detail.

When solving real-world problems, it might be challenging to define proper characteristics that lead us to the solution u. Nevertheless, the method of characteristics helps define exact solutions to toy problems and can help to design new schemes like, e.g., the Active Flux Method [27, 43, 48]. For further details about the method of characteristics, we refer to the works of Lax and Dafermos [20, 55].

Weak solutions

One special feature of non-linear hyperbolic conservation laws is that solutions can form shock waves in finite time even though the initial values are smooth. This is because characteristics may cross, resulting in faster waves overtaking slower ones. The solution $u(\mathbf{x}, t)$ has an infinite slope at this so-called *breakup point*. After this breakup point, a classical solution to the conservation law does not exist anymore; thus, a more general concept is needed. Therefore, we introduce the concept of *weak solutions*. Weak solutions must be defined so that the continuity requirements are much less restrictive. For this purpose, we define the functional space

$$C_c^1(\mathbb{R}^d \times \mathbb{R}^+) = \left\{ w \in C^1(\mathbb{R}^d \times \mathbb{R}^+) \mid w \text{ has a compact support in } \mathbb{R}^d \times \mathbb{R}^+ \right\}.$$

We multiply equation (2.3) by a test function $w \in C_c^1(\mathbb{R}^d \times \mathbb{R}^+)$ and integrate in space and time:

$$\int_0^\infty \int_{\mathbb{R}^d} [\partial_t u + \nabla \cdot \mathbf{f}(u)] \, w \mathrm{d}\mathbf{x}\mathrm{d}t = 0 \tag{2.12}$$

Furthermore, shifting the derivatives from the solution u and the flux function $\mathbf{f}$ to the test function w using integration by parts, results in

$$\int_0^\infty \int_{\mathbb{R}^d} [u\partial_t w + \mathbf{f}(u) \cdot \nabla w] \, \mathrm{d}\mathbf{x}\mathrm{d}t = -\int_{\mathbb{R}^d} u_0(\mathbf{x}) w(\mathbf{x},0)\mathrm{d}\mathbf{x}. \tag{2.13}$$

Here, we used the given initial data u_0 and the fact that w has compact support, and thus the boundary terms vanish. Next, we can define the term *weak solution*.

Definition 1 (Weak solution)**.** *[33, Definition 2.1] A function $u \in L^1 \cap L^\infty(\mathbb{R}^d \times \mathbb{R}^+)$ is called a weak solution of (2.3) if equation (2.13) holds for all test functions $w \in C_c^1(\mathbb{R}^d \times \mathbb{R}^+)$.*

Weak solutions offer a more general perspective on the concept of solutions than classical solutions that fulfill equation (2.3). This is because, for the existence of weak solutions, the flux function f and the solution u itself do not need to be differentiable in the classical sense. The downside of weak solutions is that they are not necessarily unique, and therefore we need to define additional requirements to obtain the concept of a unique and physical solution.

Entropy solutions

In this part, we roughly follow the book of Godlewski and Raviart [33].

The concept of additional diffusion is one way to address the problem of forming discontinuities and crossing characteristics. We introduce the *viscous regularization* of the scalar conservation law (2.3), which is given by the following PDE

$$\begin{aligned} \partial_t u + \nabla \cdot \mathbf{f}(u) &= \varepsilon \Delta u \qquad \text{in } \mathbb{R}^d \times \mathbb{R}_+, \\ u(\mathbf{x},0) &= u_0(\mathbf{x}) \qquad \text{in } \mathbb{R}^d \end{aligned} \tag{2.14}$$

for a positive constant $\varepsilon > 0$. As stated by Dafermos [20], it can be shown that problem (2.14) is well-posed for a sufficiently regular flux function $\mathbf{f}$ and initial value u_0. Consequently, a smooth solution u^ε exists for every $\varepsilon > 0$. If this sequence of smooth solutions $\{u^\varepsilon\}_{\varepsilon>0}$ converges with $\lim_{\varepsilon\to 0} u^\varepsilon = u$, we define u as the *vanishing viscosity solution*. One can show [33, Theorem 3.3] that the vanishing viscosity solution is a weak solution of the scalar conservation law

$$\begin{aligned} \partial_t u + \nabla \cdot \mathbf{f}(u) &= 0 \\ u(\mathbf{x},0) &= u_0(\mathbf{x}) \end{aligned}$$

in the sense of Definition 1.

Next, let us consider a strictly convex function $U = U(u)$. Additionally, we define the following function

$$\mathbf{F}(u) = \int_0^u U'(s)\mathbf{f}'(s)\mathrm{d}s \tag{2.15}$$

for which we can show that $\mathbf{F}'(u) = U'(u)\mathbf{f}'(u)$. These definitions provide the so-called *entropy pair* $(U, \mathbf{F})$ consisting of the *entropy function* U and the *entropy flux* $\mathbf{F}$. Furthermore, the expression $U'(u)$ is often referred to as the *entropy variable*. We take the entropy variable $U'(u)$ and multiply it with equation (2.14)

$$U'(u)\partial_t u + U'(u)\nabla\cdot\mathbf{f}(u) = \varepsilon U'(u)\Delta u.$$

We use the regularity of the solution u and the flux function $\mathbf{f}$ and obtain

$$U'(u)\partial_t u + U'(u)\sum_{i=1}^{d}\partial_u f^i(u)\partial_{x_i}u = \varepsilon U'(u)\Delta u$$

which can be rewritten as

$$\begin{aligned} & \partial_t(U(u)) + \sum_{i=1}^{d}\partial_u F^i(u)\partial_{x_i}u = \varepsilon U'(u)\Delta u \\ \Leftrightarrow \qquad & \partial_t(U(u)) + \nabla\cdot\mathbf{F}(u) = \varepsilon U'(u)\Delta u. \end{aligned}$$

Using

$$\Delta U(u) = U'(u)\Delta u + U''(u)|\nabla u|^2$$

we obtain

$$\partial_t(U(u)) + \nabla\cdot\mathbf{F}(u) = \varepsilon\Delta U(u) - \varepsilon U''(u)|\nabla u|^2.$$

Next, we build the integral over an arbitrary finite domain $\widetilde{\Omega} \subset \mathbb{R}^d$ and apply the Gauss divergence theorem to obtain

$$\int_{\widetilde{\Omega}} \partial_t(U(u)) + \nabla\cdot\mathbf{F}(u)\mathrm{d}\mathbf{x} = \int_{\partial\widetilde{\Omega}} \varepsilon U'(u)\nabla u\cdot\mathbf{n}\mathrm{d}s - \int_{\widetilde{\Omega}} \varepsilon U''(u)|\nabla u|^2\mathrm{d}\mathbf{x}. \tag{2.16}$$

For the limit $\varepsilon \to 0$ one can show that the first term on the right hand side in (2.16) vanishes. The second term is a bit more tricky, since it does not necessarily converge to zero for a discontinuous solution u as $\varepsilon \to 0$. In this case we can use the assumption that U is a convex function, which means that $U'' > 0$. This gives us in summary

$$\int_{\widetilde{\Omega}} \partial_t(U(u)) + \nabla\cdot\mathbf{F}(u)\mathrm{d}\mathbf{x} \leq 0. \tag{2.17}$$

Since equation (2.17) holds for an arbitrary $\widetilde{\Omega}$, we obtain the final *entropy inequality*

$$\partial_t(U(u)) + \nabla \cdot \mathbf{F}(u) \leq 0. \tag{2.18}$$

We can use these observations to define the concept of an *entropy solution*.

Definition 2 (Entropy solution)**.** *[33, Definition 3.2] A function u is an entropy solution of* (2.3) *for a given initial data $u_0 \in L^1 \cap L^\infty(\mathbb{R}^d)$ if*

1. *$u \in L^1 \cap L^\infty(\mathbb{R}^d \times \mathbb{R}_+)$*

2. *u is a weak solution in the sense of Definition 1*

3. *for all test functions $\phi \in C_c^1(\mathbb{R}^d \times \mathbb{R}_+))$ with $\phi \geq 0$ and for all entropy pairs $(U, \mathbf{F})$, u satisfies*
$$\int_{\mathbb{R}^d \times \mathbb{R}_+} (U(u)\partial_t\phi + \mathbf{F}(u) \cdot \nabla\phi)\, d\mathbf{x}dt \geq 0. \tag{2.19}$$

Note that inequality (2.19) must hold for *every* entropy pair. This can become an involved task since, for scalar conservation laws, every convex function is an entropy function. As a result, more convenient concepts have been developed to show the existence of an entropy solution like the *Kruzkov entropy pairs* [53].

For entropy solutions, one can prove several valuable properties. First of all, one can show that the entropy solution of a conservation law is unique [44]. Furthermore, we can use the entropy inequality to derive bounds on the entropy solution: We integrate the entropy inequality (2.18) over $\mathbb{R}^d \times (0,t)$ and assume compact support of the solution to obtain:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_0^t \int_{\mathbb{R}^d} U(u)\mathrm{d}\mathbf{x}\mathrm{d}\tau \leq 0.$$

The fundamental theorem of calculus gives

$$\int_{\mathbb{R}^d} U(u(\mathbf{x},t))\mathrm{d}\mathbf{x} \leq \int_{\mathbb{R}^d} U(u_0(\mathbf{x}))\mathrm{d}\mathbf{x}. \tag{2.20}$$

For the particular choice of $U(u) = \frac{1}{2}u^2$, we can derive from (2.20) an L^2 stability estimate for the entropy solution u.

Finally, one can show that the vanishing viscosity solution $u = \lim_{\varepsilon \to 0} u^\varepsilon$ from above is not only a weak solution but also an entropy solution [33].

2.1.2. Systems of equations

Although scalar equations are of great help when developing new methods, the goal of this work is to solve time-dependent systems of conservation laws in one and two space dimensions. These are given by

$$\partial_t \mathbf{u} + \sum_{i=1}^{d} \partial_{x_i} \mathbf{f}^i(\mathbf{u}) = \mathbf{0} \qquad \text{in } \Omega \times (0,T). \tag{2.21}$$

Here, $\mathbf{u} : \mathbb{R}^d \times (0,T) \to \mathbb{R}^m, d \in \{1,2\}, m \in \mathbb{N}$ is a vector of m conserved quantities and $\mathbf{f}^i : \mathbb{R}^m \to \mathbb{R}^m$, $i \in \{1,...,d\}$, are the flux functions. To complete the formulation, we need to define the initial data

$$\mathbf{u} = \mathbf{u}_0 \qquad \text{on } \Omega \times \{t = 0\}$$

and boundary conditions at the inlet boundary Γ^{in}

$$\mathbf{u} = \mathbf{g} \qquad \text{on } \Gamma^{in} \times (0,T).$$

Compared to the scalar case (2.4), it is more difficult to define the inlet boundary and the boundary conditions in general for systems of hyperbolic conservation laws. According to Dolejší and Feistauer [22], the theory of well-posed boundary conditions in multidimensional problems is still an open problem and current research. In this work, we discuss the choice of boundary conditions in the corresponding chapters of the numerical results.

For each flux function $\mathbf{f}^i$, $i \in \{1,...,d\}$, we can define the $m \times m$ Jacobian matrix

$$\mathbf{A}^i(\mathbf{u}) = \begin{pmatrix} \partial_{u_1} f_1^i(\mathbf{u}) & \cdots & \partial_{u_m} f_1^i(\mathbf{u}) \\ \vdots & \ddots & \vdots \\ \partial_{u_1} f_m^i(\mathbf{u}) & \cdots & \partial_{u_m} f_m^i(\mathbf{u}) \end{pmatrix}. \tag{2.22}$$

Under the assumption that $\mathbf{u}$ and $\mathbf{f}$ are differentiable, we can rewrite equation (2.21) by applying the chain rule to receive the quasi-linear form

$$\partial_t \mathbf{u} + \sum_{i=1}^{d} \mathbf{A}^i(\mathbf{u}) \partial_{x_i} \mathbf{u} = \mathbf{0}. \tag{2.23}$$

We use the quasi-linear form (2.23) to define the concept of *hyperbolicity*.

Definition 3 (Hyperbolicity)**.** *[57, Definition 18.1.] We call the system* (2.23) *hyperbolic if for all $n_1, \ldots, n_d \in \mathbb{R}$ the matrix $\mathbf{A}(\mathbf{u}) = \sum_{i=1}^{d} n_i \mathbf{A}^i(\mathbf{u})$ is diagonizable with only real eigenvalues $\lambda_j(\mathbf{u})$, $j \in \{1,...,m\}$.*

In this work, we will consider the Euler equations as a well-known example of a system of hyperbolic equations. As a first step towards the Euler equations, we will start with a discussion of linear systems of conservation laws.

Linear system

A linear system is the simplest form of a system of conservation laws. It is given by

$$\partial_t \mathbf{u} + \sum_{i=1}^{d} \mathbf{A}^i \partial_{x_i} \mathbf{u} = \mathbf{0},$$

with $\mathbf{A}^i$ being constant matrices. Let us take a closer look at the one-dimensional problem which is given by (neglecting the index $i = 1$)

$$\partial_t \mathbf{u} + \mathbf{A} \partial_x \mathbf{u} = \mathbf{0}. \tag{2.24}$$

Since we are considering hyperbolic problems, we know that there exist m real eigenvalues $\lambda_1, \ldots, \lambda_m$ with corresponding eigenvectors $\mathbf{r}_1, \ldots, \mathbf{r}_m$. Without loss of generality in this work, we will assume that $\lambda_1 \leq \ldots \leq \lambda_m$. Moreover, we define the diagonal matrix Λ, which diagonal elements are the eigenvalues of $\mathbf{A}$, and the matrix $\mathbf{R}$, which columns are the right eigenvectors of $\mathbf{A}$, as

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{bmatrix} \quad \text{and} \quad \mathbf{R} = \left[\mathbf{r}_1 | \ldots | \mathbf{r}_m \right]. \tag{2.25}$$

Thus, this gives us the following relation

$$\mathbf{AR} = \mathbf{R}\Lambda. \tag{2.26}$$

This relation leads to the decomposition of the Jacobian matrix

$$\mathbf{A} = \mathbf{R}\Lambda\mathbf{R}^{-1}, \tag{2.27}$$

which can be used to rewrite the given problem (2.24) (using that $\mathbf{R}$ is a constant matrix) into

$$\begin{aligned} & \partial_t \mathbf{u} + \mathbf{A} \partial_x \mathbf{u} = \mathbf{0} \\ \Leftrightarrow \quad & \partial_t \mathbf{u} + \mathbf{R}\Lambda\mathbf{R}^{-1} \partial_x \mathbf{u} = \mathbf{0} \\ \Leftrightarrow \quad & \partial_t (\mathbf{R}^{-1}\mathbf{u}) + \Lambda \partial_x (\mathbf{R}^{-1}\mathbf{u}) = \mathbf{0}. \end{aligned}$$

Introducing the *characteristic variables* $\mathbf{v} = \mathbf{R}^{-1}\mathbf{u}$ we can reformulate the linear system (2.24) in the characteristic form:

$$\partial_t \mathbf{v} + \Lambda \partial_x \mathbf{v} = \mathbf{0}. \tag{2.28}$$

This representation can be interpreted as a decoupled system of m scalar advection equations with different wave speeds λ_i. These linear advection equations can then be solved independently from each other. We solve these linear advection equations by using the method of characteristics. For the corresponding initial values, we can do the transformation

$$\mathbf{v}_0(x) = \mathbf{R}^{-1}\mathbf{u}_0(x).$$

Then the solutions of (2.28) are given by

$$v_i(x,t) = v_{i,0}(x - \lambda_i t), \quad i = 1,\dots,m. \tag{2.29}$$

If we use once again the definition of the characteristic variables, we specify the analytical solution of the initial problem (2.24):

$$\mathbf{u}(x,t) = \sum_{j=1}^{m} v_j(x,t)\mathbf{r}_j = \sum_{j=1}^{m} v_{j,0}(x - \lambda_j t)\mathbf{r}_j. \tag{2.30}$$

Therefore the final solution is a linear combination of m waves traveling at the characteristic speeds $\lambda_1,\dots,\lambda_m$.

Next, let us try to extend the above ideas to the two-dimensional case of a linear system, which can be formulated as

$$\partial_t \mathbf{u} + \mathbf{A}\partial_x \mathbf{u} + \mathbf{B}\partial_y \mathbf{u} = \mathbf{0}. \tag{2.31}$$

Note that we use the notation $\mathbf{A}^1 = \mathbf{A}$ and $\mathbf{A}^2 = \mathbf{B}$. In this setting, the equations can only be decoupled for the particular case of commutative matrices $\mathbf{AB} = \mathbf{BA}$, in which the matrices $\mathbf{A}$ and $\mathbf{B}$ share a common eigenvector basis $\mathbf{R}$. Then, we can decompose the two matrices

$$\mathbf{A} = \mathbf{R}\Lambda_x\mathbf{R}^{-1} \quad \text{and} \quad \mathbf{B} = \mathbf{R}\Lambda_y\mathbf{R}^{-1} \tag{2.32}$$

with the two corresponding diagonal matrices Λ_x and Λ_y consisting of the eigenvalues of $\mathbf{A}$ and $\mathbf{B}$. We can once again define the characteristic variables as $\mathbf{v} = \mathbf{R}^{-1}\mathbf{u}$ and rewrite the linear system (2.31) into the decoupled system

$$\partial_t \mathbf{v} + \Lambda_x \partial_x \mathbf{v} + \Lambda_y \partial_y \mathbf{v} = \mathbf{0}. \tag{2.33}$$

This system can be solved analogously to the one-dimensional case to obtain an analytical solution for the two-dimensional linear system.

Let us now consider the general case of non-commutating matrices $\mathbf{AB} \neq \mathbf{BA}$. Then the matrices $\mathbf{A}$ and $\mathbf{B}$ are not simultaneously diagonalizable, and we are only able to diagonalize them separately into

$$\mathbf{A} = \mathbf{R}_x \Lambda_x \mathbf{R}_x^{-1} \quad \text{and} \quad \mathbf{B} = \mathbf{R}_y \Lambda_y \mathbf{R}_y^{-1}. \tag{2.34}$$

Consequently, it is impossible to decouple these systems in the above way. The multidimensional acoustics equation is a well-known example of a system where the matrices $\mathbf{A}$ and $\mathbf{B}$ are not simultaneously diagonalizable. This leads to waves that can propagate information in infinitely many possible directions, which results in a significantly more complex solution process.

Compressible Euler equations

The Euler equations combine the conservation laws of mass, momentum, and energy. We will first look at the one-dimensional Euler equations and discuss their basic structure before getting to the two-dimensional case.

The one-dimensional Euler equations can be written as

$$\partial_t \mathbf{u} + \partial_x \mathbf{f}(\mathbf{u}) = \mathbf{0} \tag{2.35}$$

with the vector of conserved variables $\mathbf{u}$ and the flux function $\mathbf{f}$ given by

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \quad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}. \tag{2.36}$$

Here, ρ is the density, u is the velocity, E is the energy, and p is the pressure. To complete the system of Euler equations, we need to introduce the *equation of state*, which is given for an ideal gas by

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2 \tag{2.37}$$

with the *adiabatic gas constant* γ. In this work, we choose $\gamma = 1.4$. We can rewrite equation (2.37) to obtain a formula for the pressure

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho u^2\right). \tag{2.38}$$

Furthermore, we introduce the *speed of sound* a as

$$a = \sqrt{\frac{\gamma p}{\rho}} \tag{2.39}$$

and the *total specific enthalpy* as

$$H = \frac{E+p}{\rho}. \tag{2.40}$$

We can now rewrite the Euler equations in the quasi-linear form

$$\partial_t \mathbf{u} + \mathbf{A}(\mathbf{u})\partial_x \mathbf{u} = \mathbf{0} \tag{2.41}$$

with the Jacobian matrix

$$\mathbf{A}(\mathbf{u}) = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)\left(\frac{u_2}{u_1}\right)^2 & (3-\gamma)\left(\frac{u_2}{u_1}\right) & \gamma-1 \\ -\frac{\gamma u_2 u_3}{u_1^2}(\gamma-1)\left(\frac{u_2}{u_1}\right)^3 & \frac{\gamma u_3}{u_1} - \frac{3}{2}(\gamma-1)\left(\frac{u_2}{u_1}\right)^2 & \gamma\left(\frac{u_2}{u_1}\right) \end{bmatrix}. \tag{2.42}$$

The Jacobian matrix may be written in the more convenient form expressed in terms of the total specific enthalpy H and the particle velocity u

$$\mathbf{A}(\mathbf{u}) = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)u^2 & (3-\gamma)u & \gamma-1 \\ \frac{1}{2}(\gamma-1)u^3 - uH & H-(\gamma-1)u^2 & \gamma u \end{bmatrix}. \tag{2.43}$$

We emphasize that matrix $\mathbf{A}(\mathbf{u})$ now depends explicitly on the solution $\mathbf{u}$. Using basic calculus, one can show the following proposition.

Proposition 1. *[78, Proposition 3.5] The Jacobian matrix has real eigenvalues*

$$\lambda_1 = u-a, \quad \lambda_2 = u, \quad \lambda_3 = u+a \tag{2.44}$$

with the corresponding right eigenvectors

$$\mathbf{r}_1 = \begin{bmatrix} 1 \\ u-a \\ H-ua \end{bmatrix}, \quad \mathbf{r}_2 = \begin{bmatrix} 1 \\ u \\ \frac{1}{2}u^2 \end{bmatrix}, \quad \mathbf{r}_3 = \begin{bmatrix} 1 \\ u+a \\ H+ua \end{bmatrix}. \tag{2.45}$$

In addition to the conservative representation of the Euler equations given by (2.35) and (2.36), there are also non-conservative representations. We emphasize that the conservative and non-conservative formulations are equivalent for a smooth solution, whereas they are not for a solution containing shock waves. If the solution contains discontinuous shock waves, the non-conservative formulation returns incorrect solutions. Nevertheless, there are good reasons also to pay attention to the non-conservative schemes as they have advantages in some cases over their conservative counterpart.

By using the equation of state (2.37), we rewrite the Euler equations in its *primitive form*

$$\partial_t \mathbf{v} + \mathbf{A}(\mathbf{v})\partial_x \mathbf{v} = \mathbf{0}, \tag{2.46}$$

with

$$\mathbf{v} = \begin{bmatrix} \rho \\ u \\ p \end{bmatrix}, \quad \mathbf{A}(\mathbf{v}) = \begin{bmatrix} u & \rho & 0 \\ 0 & u & \frac{1}{\rho} \\ 0 & \rho a^2 & u \end{bmatrix}. \tag{2.47}$$

The primitive variables density ρ, velocity u, and pressure p are intuitively more informative. Thus, the numerical results of Euler equations are often presented in primitive variables, although the calculation was made initially in conservative variables. In some cases, the primitive formulation can provide a better insight into the theory of Euler equations, e.g., in analyzing the different waves of the solution of the Riemann problem [78]. Finally, there are situations where the numerical solution of a non-conservative scheme yields better results than its conservative counterpart [46]. In addition to the conservative and primitive variables, there are other approaches to represent the Euler equations, which can be found in Toro [78].

Next, we turn to the two-dimensional system of Euler equations that are given in the conservative form by

$$\partial_t \mathbf{u} + \partial_x \mathbf{f}^1(\mathbf{u}) + \partial_y \mathbf{f}^2(\mathbf{u}) = \mathbf{0} \tag{2.48}$$

with

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \quad \mathbf{f}^1(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{bmatrix}, \quad \mathbf{f}^2(\mathbf{u}) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}. \tag{2.49}$$

The equation of state in the two-dimensional case is given by

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2). \tag{2.50}$$

One crucial property of the two-dimensional Euler equations is its *rotational invariance*. To further explain this attribute, we need to express the unit outward vector $\mathbf{n} = (n_1, n_2)^\top$ of a two-dimensional surface $\mathbf{S}$ using the angle θ that is formed by the x-axis and the normal vector itself. In other words

$$\mathbf{n} = (\cos(\theta), \sin(\theta))^\top. \tag{2.51}$$

Proposition 2. *[78, Proposition 3.15] For all angles* $\theta \in [0, 2\pi]$ *and all vectors* $\mathbf{u} \in \mathbb{R}^4$ *one can show*

$$\cos(\theta)\mathbf{f}^1(\mathbf{u}) + \sin(\theta)\mathbf{f}^2(\mathbf{u}) = \mathbf{T}^{-1}\mathbf{f}^1(\mathbf{T}\mathbf{u}) \tag{2.52}$$

using the rotation matrix $\mathbf{T} = \mathbf{T}(\theta)$ *which is given by*

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.53}$$

For the proof, we refer to Toro [78]. Note that the term on the left side of equation (2.52) can be interpreted as the integrand of the surface integral

$$\int_{\mathbf{S}} (\mathbf{f}^1, \mathbf{f}^2)^T \cdot \mathbf{n} \, \mathrm{d}s = \int_{\mathbf{S}} cos(\theta)\mathbf{f}^1(\mathbf{u}) + \sin(\theta)\mathbf{f}^2(\mathbf{u}) \, \mathrm{d}s.$$

The rotational invariance can be helpful for computational purposes to deal with domains that are not aligned with the x-axis or y-axis. Furthermore, it is sufficient to only check the Jacobian matrix in x-direction regarding the hyperbolicity of the system. The Jacobian matrix $\mathbf{A}^1(\mathbf{u}) = \mathbf{A}(\mathbf{u})$ for the corresponding flux in x-direction $\mathbf{f}^1(\mathbf{u})$ is given by

$$\mathbf{A}(\mathbf{u}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ (\gamma-1)\frac{1}{2}\mathbf{V}^2 - u^2 & (3-\gamma)u & -(\gamma-1)v & \gamma-1 \\ -uv & v & u & 0 \\ u\left((\gamma-1)\frac{1}{2}\mathbf{V}^2 - H\right) & H-(\gamma-1)u^2 & -(\gamma-1)uv & \gamma u \end{bmatrix} \tag{2.54}$$

with the *specific kinetic energy* $\frac{1}{2}\mathbf{V}^2 = \frac{1}{2}\left(u^2 + v^2\right)$.

Once again, one can show that the Euler equations in two dimensions are hyperbolic by computing the eigenvalues and eigenvectors.

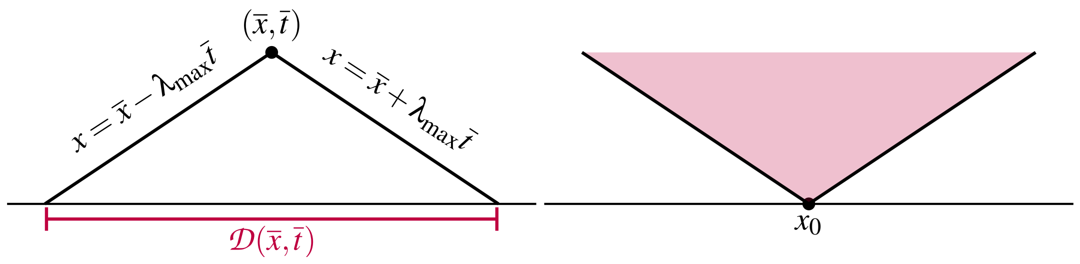
Proposition 3. *[78, Proposition 3.14] The Jacobian matrix* $\mathbf{A}(\mathbf{u})$ *has the real eigenvalues*

$$\lambda_1 = u - a, \quad \lambda_2 = \lambda_3 = u, \quad \lambda_4 = u + a \tag{2.55}$$

with the corresponding eigenvectors

$$\mathbf{r}_1 = \begin{bmatrix} 1 \\ u-a \\ v \\ H-ua \end{bmatrix}, \quad \mathbf{r}_2 = \begin{bmatrix} 1 \\ u \\ v \\ \frac{1}{2}\mathbf{V}^2 \end{bmatrix}, \quad \mathbf{r}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ v \end{bmatrix}, \quad \mathbf{r}_3 = \begin{bmatrix} 1 \\ u+a \\ v \\ H+ua \end{bmatrix}. \tag{2.56}$$

For further information regarding the Euler equations, we refer to Toro [78]. We conclude our theoretical discussion of hyperbolic conservation laws by introducing the concept of the *domain of dependence* (DoD) and *range of influence* (RoI).



$(\bar{x},\bar{t})$
$x=\bar{x}-\lambda_{\max}\bar{t}$
$x=\bar{x}+\lambda_{\max}\bar{t}$
$\mathcal{D}(\bar{x},\bar{t})$
x_0

Figure 2.3: [56] Domain of dependence and range of influence.

Domain of dependence and range of influence

Following Leveque [57], we can use the method of characteristics to introduce two important concepts: the domain of dependence and the range of influence. Thanks to the method of characteristics, we know that the solution at a given point $(\bar{x},\bar{t})$ is determined by the data in some finite set $\mathcal{D} \subset \mathbb{R}^d$ around the point $\bar{x}$. This set can be bounded by the maximum eigenvalue that denotes the maximum wave speed $\lambda_{\max}$, through $\mathcal{D} \subset \{x : |x-\bar{x}| \leq \lambda_{\max}\bar{t}\}$. We emphasize that the size of the set increases with progressive time. This set $\mathcal{D}$ is called the domain of dependence. Analogously we can define the range of influence for a point x_0 as the set of points whose domains of dependence overlap with x_0 [57]. See Figure 2.3 for an illustration. These definitions allow us to conclude that for hyperbolic equations, the maximum speed of propagating information is bounded by $\lambda_{\max}$ and, therefore, finite.

We note that this work can only treat the tip of the iceberg with respect to the theory of hyperbolic conservation laws. For a good overview and a deeper insight into the basic principles regarding hyperbolic conservation laws, we refer to [20, 56, 57, 78].

2.2. Numerical discretization of the problem

In what follows, we will introduce aspects of the discretization for numerically solving the given problem

$$\partial_t \mathbf{u} + \sum_{i=1}^{d} \partial_{x_i} \mathbf{f}^i(\mathbf{u}) = \mathbf{0} \qquad \text{in } \Omega \times (0,T). \tag{2.57}$$

In this scope, we will discuss both the spatial and temporal discretization in one and two dimensions. If a distinction between the one-dimensional and the two-dimensional case is necessary, we will highlight this explicitly.

2.2.1. Discretization of the domain

In this section, we will introduce the notation for the discretization of a domain in the case of a general unstructured grid. This is consistent with our plan to use cut cell grids later since cut cell grids can be interpreted as a special case of unstructured grids.

We consider an open, connected, polygonal domain $\Omega \subset \mathrm{R}^d$ with $d \in \{1,2\}$. For numerical computations, we need to discretize the domain Ω by using a triangulation $\mathcal{M}_h$. This triangulation is given by a (possibly unstructured) grid consisting of cells $E \in \mathcal{M}_h$.

One-dimensional case

In one dimension, we assume without loss of generality that the domain is given by $\Omega = (0,1)$. We divide $\overline{\Omega}$ in N cells $E_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$, $j \in \{1,...,N\}$ with cell lengths $\Delta x_j = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}$. For a more convenient and consistent setup between one and two dimensions, we define edges between two cells as

$$e_{j+\frac{1}{2}} = e_{E_j,E_{j+1}} = E_j \cap E_{j+1} = x_{j+\frac{1}{2}}.$$

Furthermore, we define the set of internal and external edges as

$$\Gamma_h^{\text{int}} = \left\{ e_{j+\frac{1}{2}} \,\middle|\, j \in \{1,...,N-1\} \right\}, \tag{2.58}$$

$$\Gamma_h^{\text{ext}} = \left\{ e_{\frac{1}{2}}, e_{N+\frac{1}{2}} \right\} = \{0,1\}. \tag{2.59}$$

We note that the elements of the set of external edges are defined by the intersection of a cell $E \in \mathcal{M}_h$ with the boundary $\partial\Omega$. We introduce this unusual and overkill notation for one-dimensional edges to allow later a common spatial and temporal discretization of the given problem for one and two dimensions. However, in 1D, we will use the more natural notation in numerical methods and refer to 'cell j' and associate faces with the coordinate '$x_{j\pm 1/2}$' instead of using 'cell E' or 'cell E_j' or faces '$e_{j\pm 1/2}$'. Despite the initial lack of clarity regarding the definition of an outer normal n_{E_j} in 1D, we still define it here. The outer normal vector in 1D is a scalar. At the left edge $x_{k-\frac{1}{2}}$ it has the value $n = -1$, and at the right edge $x_{k+\frac{1}{2}}$ the value $n = 1$.

Two-dimensional case

In the two-dimensional case, there is no natural order of cells, which means that we drop the index j here. The mesh cells E have different polygonal shapes, e.g. they are triangular or quadrilateral. We define the edge between two elements E and E' as

$$e_{E,E'} = E \cap E'. \tag{2.60}$$

The internal and external skeletons of the partitioning are given by

$$\Gamma_h^{\text{int}} = \left\{ e_{E_1,E_2} \;\middle|\; E_1, E_2 \in \mathcal{M}_h \text{ and } E_1 \neq E_2 \text{ and } |e_{E_1,E_2}| > 0 \right\}, \tag{2.61}$$

$$\Gamma_h^{\text{ext}} = \left\{ e_E = \partial E \cap \partial\Omega \;\middle|\; E \in \mathcal{M}_h \text{ and } |e_E| > 0 \right\}, \tag{2.62}$$

with $|e|$ denoting the length of an edge e. In addition, we define the set of all edges

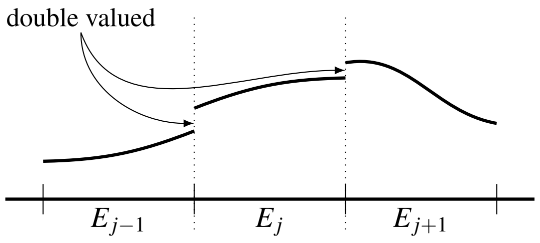
$$\Gamma = \Gamma_h^{\text{int}} \cup \Gamma_h^{\text{ext}}. \tag{2.63}$$

Finally, we define the unit outward vector pointing from cell E to cell E' as $\mathbf{n}_{E,E'}$.

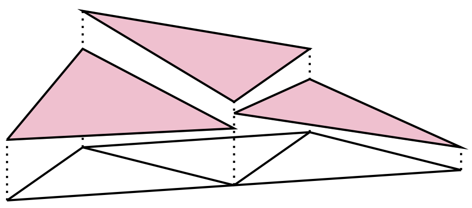
2.2.2. Discontinuous Galerkin method

We discretize the system of equations (2.57) using a Runge-Kutta discontinuous Galerkin (RKDG) approach. An RKDG method is based on the method of lines approach. This means that we first discretize in space using the discontinuous Galerkin methodology. Afterwards, we will apply a suitable explicit Runge-Kutta time-stepping scheme to solve the resulting system of ODEs. We start with the space discretization:

The idea of the DG approach is the same as for a classical Finite-Element-Method (FEM) approach: The numerical solution u^h of the equation is represented on different cells E as a linear combination of functions from a particular function space V_h. As the name already suggests, the functions do not have to fulfill continuity requirements between individual cells; see Figure 2.4 for a one-dimensional and a two-dimensional example.



(a) Discontinuous Galerkin in 1D



(b) Discontinuous Galerkin in 2D

Figure 2.4: Samples of discontinuous Galerkin ansatz functions in different spatial dimensions.

This leads us to the definition of the function space for the DG approach:

Definition 4 (Discrete Function Space)**.** *We define the discrete space $V_h^p \subset (L^2(\Omega))^m$ by*

$$V_h^p = \left\{ \mathbf{w}^h \in (L^2(\Omega))^m \,|\, w_l^h|_E \in P^p(E) \, \textit{for each component } l = 1,...,m, \ \forall E \in \mathcal{M}_h \right\},$$

where $P^p(E)$ denotes the space of all polynomials of degree p on cell E.

The evaluation of functions $\mathbf{w}^h \in V_h^p$ are not well-defined on cell edges. The lack of continuity requirements between cells leads to different limits $\mathbf{w}_E^h$ and $\mathbf{w}_{E'}^h$ on an edge $e_{E,E'}$. Therefore, we define the jump in one and two dimensions according to DiPietro and Ern [21].

Definition 5 (Jump & Average). *The jump in normal direction $\mathbf{n}_{E_1,E_2}$ over a face $e_{E_1,E_2} = \partial E_1 \cap \partial E_2$ between two elements E_1 and E_2 is defined by*

$$\left[\!\left[u^h \right]\!\right]_{e_{E_1,E_2}}(\mathbf{x}) := u^h|_{E_1}(\mathbf{x}) - u^h|_{E_2}(\mathbf{x}). \tag{2.64}$$

The (scalar-valued) average *on a face is defined by*

$$\left\{\!\!\left\{ u^h \right\}\!\!\right\}_{e_{E_1,E_2}}(\mathbf{x}) := \frac{1}{2}(u^h|_{E_1}(\mathbf{x}) + u^h|_{E_2}(\mathbf{x})).$$

Remark 6. *[21, Remark 1.19] The jump and average can also be defined for boundary faces $e \in \Gamma_h^{ext}$. We set for $e \in \Gamma_h^{ext}$ with $e = \partial E \cap \partial\Omega$:*

$$\left[\!\left[u^h \right]\!\right]_e(\mathbf{x}) = \left\{\!\!\left\{ u^h \right\}\!\!\right\}_e(\mathbf{x}) = u^h|_E(\mathbf{x})$$

When considering a vector-valued function $\mathbf{u}^h$, the jump and the average are applied to every component of the function. From now on, we will drop the subscript e_{E_1,E_2} and the variable $\mathbf{x}$, if it is clear from the context. Note that the jump and the average might vary over the face for V_h^p, $p > 0$ in two dimensions.

In order to obtain the semi-discrete DG scheme, we follow DiPietro and Ern [21] and test equation (2.57) on a cell E with some test function $\mathbf{w}^h \in V_h^p$. Note that, generally, one can choose different function spaces for the functions that represent the numerical solution and the test functions. In this work, we will choose the same function space for both. After testing equation (2.57), we do integration by parts in space and obtain

$$\int_\Omega \partial_t \mathbf{u} \cdot \mathbf{w}^h \mathrm{d}\mathbf{x} - \int_\Omega \mathbf{f}(\mathbf{u}) \cdot \nabla \mathbf{w}^h \mathrm{d}\mathbf{x} + \sum_{e \in \Gamma} \int_e \mathbf{f}_{\mathbf{n}_e}(\mathbf{u}) \cdot \left[\!\left[\mathbf{w}^h \right]\!\right] \mathrm{d}s = 0, \tag{2.65}$$

with the products being interpreted as the corresponding scalar product and the definition of the normal flux function, which is given by

$$\mathbf{f}_{\mathbf{n}_e} = \sum_{i=1}^{d} \mathbf{f}^i n_{e,i} \tag{2.66}$$

with $n_{e,i}$ being the ith component of the unit normal vector $\mathbf{n}_e$.

The numerical solution $\mathbf{u}^h$ is represented as a combination of functions from V_h^p and, therefore, discontinuous at the edges $e \in \Gamma_h^{\text{int}}$. Consequently, the evaluation of the numerical solution on an edge $e_{E,E'}$ is no longer unique since it has two different limits $\mathbf{u}_E^h$ and $\mathbf{u}_{E'}^h$. Thus, we replace the non-linear flux function $\mathbf{f}_{\mathbf{n}_e}(\cdot)$ with a numerical flux function $\mathcal{H}(\mathbf{n}_e, \cdot, \cdot)$ that depends on the two values at the corresponding point on the edge.

We obtain the final formulation of the semi-discrete problem: Find $\mathbf{u}^h \in V_h^p$ such that

$$\left(d_t \mathbf{u}^h(t), \mathbf{w}^h\right)_{L^2(\Omega)} + a_h\left(\mathbf{u}^h(t), \mathbf{w}^h\right) = 0 \quad \forall \mathbf{w}^h \in V_h^p, \tag{2.67}$$

with

$$a_h(\mathbf{u}^h, \mathbf{w}^h) = -\sum_{E \in \mathcal{M}_h} \int_E \mathbf{f}(\mathbf{u}^h) \cdot \nabla \mathbf{w}^h \mathrm{d}\mathbf{x} + \sum_{e \in \Gamma_h^{\text{int}}} \int_e \mathcal{H}(\mathbf{n}_e, \mathbf{u}_E, \mathbf{u}_{E'}) \cdot \left[\!\left[\mathbf{w}^h\right]\!\right] \mathrm{d}s$$
$$+ \sum_{e \in \Gamma_h^{\text{ext}}} \int_e \mathcal{H}(\mathbf{n}_e, \mathbf{u}_E, \widetilde{\mathbf{u}}) \cdot \mathbf{w}^h \mathrm{d}s.$$

Here, $(\cdot,\cdot)_{L^2(\Omega)}$ denotes the standard scalar product in $(L^2(\Omega))^m$. Furthermore, we incorporate boundary conditions by suitably defining the external Riemann data $\widetilde{\mathbf{u}}$ in $\mathcal{H}(\mathbf{n}_e, \mathbf{u}_E, \widetilde{\mathbf{u}})$ for every $e \in \Gamma_h^{\text{ext}}$. Later in this work, we will provide particular choices for $\widetilde{\mathbf{u}}$ in the numerical results part of each section.

2.2.3. Numerical flux function

The choice of the numerical flux function $\mathcal{H}(\mathbf{n}_e, \cdot, \cdot)$ is essential for developing numerical methods for hyperbolic conservation laws. In particular, if we are interested in proving theoretical results in the context of a scalar equation, we must assume some standard properties for this case. Therefore we follow Cockburn and Shu [16] and demand the following properties.

Prerequisite 2.2.1. *We consider the case of a scalar conservation law. Then, we request the numerical flux $\mathcal{H}(\mathbf{n}_e, \cdot, \cdot)$ to satisfy the following properties:*

1. *Consistency:* $\mathcal{H}(\mathbf{n}_e, u, u) = \mathbf{f}_{\mathbf{n}_e}(u)$.
2. *Conservation:* $\mathcal{H}(\mathbf{n}_e, u^-, u^+) = -\mathcal{H}(-\mathbf{n}_e, u^+, u^-)$
3. *Continuity:* $\mathcal{H}(\mathbf{n}_e, u^-, u^+)$ *is at least Lipschitz continuous with respect to both arguments* u^- *and* u^+.
4. *Monotonicity:* $\mathcal{H}(\mathbf{n}_e, u^-, u^+)$
   - *is a non-decreasing function of its first argument* u^-,
   - *is a non-increasing function of its second argument* u^+.

Then, the flux has the so-called *E-flux property* which was defined by Osher [67]: For all u with $u = (\tau u^- + (1-\tau)u^+), \tau \in [0,1]$ there holds

$$(\mathcal{H}(\mathbf{n}_e, u^-, u^+) - \mathbf{f}_{\mathbf{n}_e}(u))(u^+ - u^-) \leq 0. \tag{2.68}$$

Throughout this work, we will use different numerical fluxes for the problems we consider. For the linear advection equation (2.6) with the velocity β, we will consider the classical *upwind flux*, which can be written as

$$\mathcal{H}(\mathbf{n}_e, u^-, u^+) = \begin{cases} \beta u^+, & \text{if } \beta \cdot \mathbf{n}_e \geq 0 \\ \beta u^-, & \text{else.} \end{cases} \tag{2.69}$$

For the Burgers equation in one dimension, we work with the exact Gudonov flux, which is given by

$$\mathcal{H}(n_e, u^-, u^+) = \begin{cases} \min_{u^- \leq u \leq u^+} f_{n_e}(u), & \text{if } u^- \leq u^+, \\ \min_{u^+ \leq u \leq u^-} f_{n_e}(u), & \text{if } u^+ \leq u^-. \end{cases} \tag{2.70}$$

Additionally, we will consider the approximate Roe Riemann solver for the Euler equations in one dimension. The details regarding this numerical flux function can be found in [78].

For the two-dimensional tests, we will consider the local Lax-Friedrichs flux, which is given by

$$\mathcal{H}(\mathbf{n}_e, \mathbf{u}^-, \mathbf{u}^+) = \frac{1}{2}\left(\mathbf{f}_{\mathbf{n}_e}(\mathbf{u}^-) + \mathbf{f}_{\mathbf{n}_e}(\mathbf{u}^+) - \lambda_{\mathbf{n}_e}\left(\mathbf{u}^-, \mathbf{u}^+\right)\left(\mathbf{u}^+ - \mathbf{u}^-\right)\right) \tag{2.71}$$

with $\lambda_{\mathbf{n}_e}(\mathbf{u}^-, \mathbf{u}^+)$ being the largest absolute eigenvalue of $\partial_{\mathbf{u}}\mathbf{f}_{\mathbf{n}_e}(\mathbf{u}^-)$ and $\partial_{\mathbf{u}}\mathbf{f}_{\mathbf{n}_e}(\mathbf{u}^+)$. We emphasize that the local Lax-Friedrichs flux and the upwind flux are the same for the linear advection equation.

As a next step, we will combine the semi-discrete problem (2.67) with a suitable time-stepping scheme. This will result in a fully discrete problem that we can then solve later by iterating over the time steps.

2.2.4. Time-stepping schemes

For the full discretization of the system of the ordinary differential equations (ODEs), which is given by the semi-discrete problem (2.67), we use strong stability preserving Runge-Kutta (SSP-RK) time-stepping schemes, also formerly known as Total Variation Diminishing (TVD) RK time-stepping schemes. Later in this section, we will define and explain the concept of TVD schemes in more detail.

We consider a general ODE of the form

$$\mathrm{d}_t y = L(y), \tag{2.72}$$

discretize the time horizon $(0, T)$ in time intervals (t^n, t^{n+1}) using a time step size Δt, and define $t^n = n\Delta t$. The numerical approximation of the solution vector y at a certain time t^n will be abbreviated as $y^n \approx y(t^n)$. The idea of SSP-RK schemes is as follows: We start with a semi-discrete method-of-lines approach that guarantees strong stability in a certain norm if combined with the explicit Euler time-stepping scheme

$$\left\|y^{n+1}\right\| = \|y^n + \Delta t_E L(y^n)\| \leq \|y^n\| \tag{2.73}$$

under some time step restriction $\Delta t_E \geq 0$. Then the goal is to find a higher-order time discretization that preserves the strong stability of the explicit Euler scheme by constructing the new scheme as a convex combination of explicit Euler steps.

The key to finding such higher-order s-stage SSP Runge-Kutta methods is by writing them in the following form:

$$\begin{aligned} y^{(0)} &= y^n, \\ y^{(i)} &= \sum_{j=0}^{i-1} (\alpha_{i,j} y^{(j)} + \Delta t \beta_{i,j} L(y^{(j)})), \qquad i = 1, \ldots, s \\ y^{n+1} &= y^{(s)} \end{aligned} \tag{2.74}$$

This is the so-called Shu-Osher representation of a Runge-Kutta scheme [73]. For consistency we need that $\sum_{j=0}^{i-1} \alpha_{i,j} = 1$. It can be shown that the Shu-Osher representation is equivalent to the traditional way of writing Runge-Kutta methods, which are normally given by a Butcher-Tableau. By writing the Runge-Kutta scheme in the Shu-Osher representation, it is easy to see that each stage is a convex combination of y^n and explicit Euler updates with Δt being scaled by $\frac{\beta_{i,j}}{\alpha_{i,j}}$

$$\left\|y^{(i)}\right\| = \left\|\sum_{j=0}^{i-1} (\alpha_{i,j} y^{(j)} + \Delta t \beta_{i,j} L(y^{(j)}))\right\| = \left\|\sum_{j=0}^{i-1} (\alpha_{i,j} \left(y^{(j)} + \Delta t \frac{\beta_{i,j}}{\alpha_{i,j}} L(y^{(j)})\right)\right\| \leq \|y^n\|. \tag{2.75}$$

Here, the last inequality is a direct consequence of the strong stability given by the explicit Euler scheme (2.73) under some modified time step restriction

$$\Delta t \leq \min_{i,j} \frac{\alpha_{i,j}}{\beta_{i,j}} \Delta t_E. \tag{2.76}$$

Note that the coefficients $\alpha_{i,j}$ and $\beta_{i,j}$ must be non-negative. Furthermore, the $\alpha_{i,j}$ may only be zero if the corresponding $\beta_{i,j}$ are zero. The manipulations in (2.75) show a vital property to ensure that the given time-stepping schemes are strong stability preserving, which we will now summarize in the following Lemma.

Lemma 7 (SSP)**.** *[73] Let the explicit Euler update be strongly stable in the sense of* (2.73) *under the time step restriction $\Delta t \leq \Delta t_E$ for some $\Delta t_E \geq 0$. Then the RK time-stepping scheme* (2.74) *is strong stability preserving*

$$\|y^{n+1}\| \leq \|y^n\|$$

if the coefficients $\alpha_{i,j}, \beta_{i,j} \geq 0$ and the time step restriction (2.76) *is fulfilled.*

Very popular SSP RK methods are given by

- The first-order explicit Euler scheme

$$y^{n+1} = y^n + \Delta t L(y^n). \tag{2.77}$$

- The second-order Heun's method

$$\begin{aligned} y^{(1)} &= y^n + \Delta t L(y^n), \\ y^{n+1} &= \frac{1}{2}y^n + \frac{1}{2}y^{(1)} + \frac{1}{2}\Delta t L(y^{(1)}). \end{aligned} \tag{2.78}$$

- The third-order Shu-Osher scheme

$$\begin{aligned} y^{(1)} &= y^n + \Delta t L(y^n), \\ y^{(2)} &= \frac{3}{4}y^n + \frac{1}{4}y^{(1)} + \frac{1}{4}\Delta t L(y^{(1)}), \\ y^{n+1} &= \frac{1}{3}y^n + \frac{2}{3}y^{(2)} + \frac{2}{3}\Delta t L(y^{(2)}). \end{aligned} \tag{2.79}$$

In this work, we will use SSP-RK schemes up to order four. The corresponding coefficients for the Shu-Osher form and more information regarding SSP RK schemes can be found in the contributions [35–37, 51, 73].

CFL number

In the previous part, we have already seen that we are restricted in our choice of the time step size Δt. Especially when the ODE is given by the semi-discrete scheme of a hyperbolic PDE, the spatial discretization plays an important role in the upper bound of our time step. This can be summarized for the one-dimensional case in the following mathematical relation

$$\Delta t \leq \frac{\nu}{2p+1}\frac{\Delta x}{\lambda_{\max}}. \tag{2.80}$$

Here, Δx denotes the cell size, $\lambda_{\max}$ is the fastest wave speed, and $\nu \in (0,1]$ is a parameter called the CFL number. The factor $\frac{1}{2p+1}$ depends on the polynomial degree p of the function

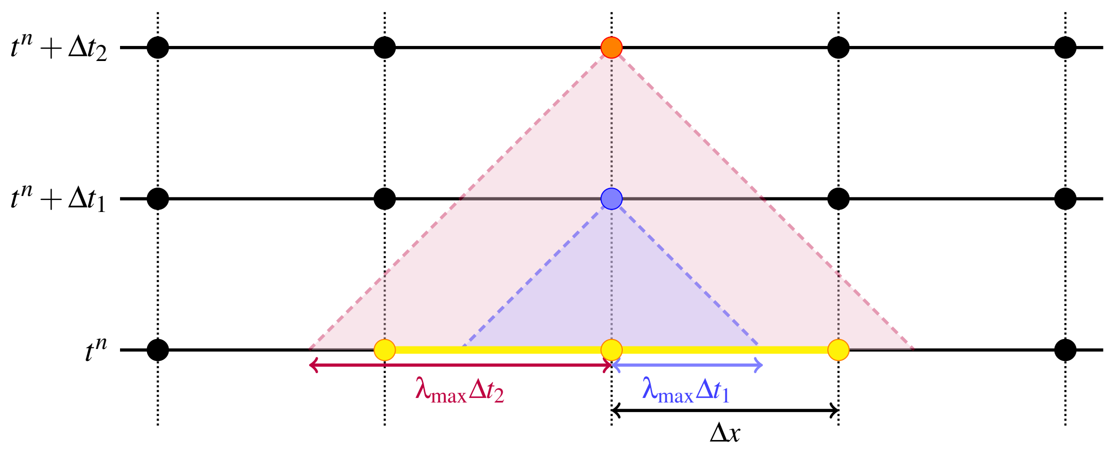


Figure 2.5: [56] Illustration of the CFL time step restriction: For the larger time step Δt_2, the physical domain of dependence (red) is not included in the numerical domain of dependence (yellow), whereas for the smaller time step Δt_1 (blue) this is the case. The CFL condition ensures that the physical domain of dependence is fully contained within the numerical one.

space V_h^p and will be discussed below. This connection between the size of the temporal and spatial discretization was discovered by Richard Courant, Kurt Friedrichs and Hans Lewy in [19], which is why it is called the CFL number.

The motivation for the CFL number can be easily seen if we consider the case of piecewise constant polynomials $p = 0$ and look at the domain of dependence for a given point concerning one time step. In Figure 2.5, we show the physical DoD for two different time step lengths Δt_1 (blue) and Δt_2 (red). Moreover, the numerical DoD of a classical RKDG scheme is marked in yellow on the lowest line. In order to obtain stability, the numerical DoD must enclose the physical DoD. In the first case (red) of Figure 2.5, the physical DoD is not included in the numerical DoD resulting in an unstable update since some important information is missing. In the second case (blue), the physical DoD is included in the numerical DoD, which gives us a stable solution update.

For higher-order RKDG methods with $p > 0$, one needs to decrease the time step size according to equation (2.80) by a factor of $\frac{1}{2p+1}$. This scaling is mandatory to ensure L^2 stability [17], but it should be understood more in the sense of a rule of thumb. Nevertheless, one could show that with the scaling, the time step restrictions are only $\leq 5\%$ smaller than many numerically-obtained time step restrictions.

When considering the two-dimensional case, an optimal choice for the time step size is more difficult to define, especially on unstructured grids. There is an approach [14] that works on robust CFL conditions for hyperbolic conservation laws on triangular meshes. The idea is to replace the cell size Δx with the minimal cell width along the characteristic flow direction. However, in this work, we will choose the time step in two dimensions according to equation (2.80) and replace the cell size Δx with the inscribed radii of the Cartesian mesh cells.

2.2.5. Desirable theoretical properties

In this work, we will develop new numerical methods. To evaluate the quality of these methods, we require specific criteria. Therefore, we will be presenting a set of desirable theoretical properties that we aim to validate for these novel methods in the following section. For most of these theoretical properties, we rely on solving scalar equations using piecewise constant polynomials from V_h^0. If not other stated, we always consider the one-dimensional setup. For this case, the degrees of freedom are u_j^n which denotes the mean value of the solution on cell E_j at time t^n.

We start by introducing *monotonicity*. Monotonicity is an essential property of a first-order scheme for hyperbolic conservation laws as it guarantees that under- and overshoots cannot occur. We recall the definition of monotonicity according to [78].

Definition 8 (Monotonicity)**.** *A method $u_i^{n+1} = H(u_{i-j_L}^n, u_{i-j_L+1}^n, \dots, u_{i+j_R}^n)$ is called* monotone*, if $\forall i$ there holds for every l with $-j_L \leq l \leq j_R$*

$$\frac{\partial H}{\partial u_{i+l}}(u_{i-j_L}, \dots, u_{i+j_R}) \geq 0. \tag{2.81}$$

Although it seems necessary to use a monotone scheme to ensure that the solution stays inside the physical admissible bounds, one can show that monotone schemes are at most first-order accurate [41, 56]. Thus, it will not be possible to maintain monotonicity when developing higher-order methods. Nevertheless, we want our schemes to be the best of both approaches: we want to have a high-order method that provides excellent accuracy, but at the same time, the solution should not develop an unphysical behavior and stay within the correct bounds. In the next section, we will introduce how to achieve this by using so-called *limiting techniques*.

For now, we continue with the definition of the *total variation* followed by the definition of a *total variation diminishing* method.

Definition 9 (Total variation (TV))**.** *The total variation of a discrete solution u^n is defined as*

$$TV(u^n) = \sum_i \left| u_{i+1}^n - u_i^n \right|. \tag{2.82}$$

Definition 10 (Total variation diminishing (TVD))**.** *We call a method total variation diminishing if there holds*

$$TV(u^{n+1}) \leq TV(u^n) \tag{2.83}$$

for all times t^n, $n \in \mathbb{N}$.

The standard way to verify that a scheme is TVD is to apply Harten's Lemma [40]. This Lemma provides an easy way to show TVD stability but relies on an equidistant grid. Therefore, we will not be able to use it in this work and will have to work with the definition itself.

If we are considering higher-order solutions that have been computed using V_h^p with $p > 0$, we want to study something similar to TVD stability. Therefore we introduce the definition of *total variation diminishing in the means* (TVDM).

Definition 11 ([17]). *A DG scheme is called total variation diminishing in the means (TVDM) if for all $n \in \mathbb{N}$*

$$TV(\overline{u}^{n+1}) \leq TV(\overline{u}^n) \quad \text{with } TV(\overline{u}^n) = \sum_i |\overline{u}_{i+1}^n - \overline{u}_i^n|. \tag{2.84}$$

Here, $\overline{u}_i^n$ denotes the mean of u on cell i at time t^n.

For piecewise constant polynomials, the means $\overline{u}_i^n$ correspond to the unknowns u_i^n, and TVDM coincides with the TVD property of Definition 10.

2.2.6. Limiter

Higher-order methods provide much better accuracy on smooth solutions than first-order methods but are also more sensitive to unphysical oscillations near discontinuities. In contrast, the advantage of using first-order methods is that they usually guarantee some kind of stability, e.g., monotonicity or TVD stability. In particular, when solving systems of hyperbolic equations like the Euler equations, the given scheme should guarantee that the quantities of interest remain within the range of physically acceptable values. This is essential since the numerical method will collapse while calculating the numerical flux if the pressure becomes negative.

The idea of a limiter is now to combine the best features of both worlds: when the numerical solution is smooth, we want to have the highest possible accuracy that is provided by the higher-order methods, but at the same time, we need the stability of the first-order method. This combination will result in a scheme that guarantees stability in the presence of shocks and a higher accuracy than the first-order method in smooth regions. There are various approaches of different limiters, all of which have advantages and disadvantages.

In Figure 2.6, we can see the results of two simulations using higher-order polynomials. The initial value of the simulation consists of two parts: a discontinuous hat function and a smooth bell curve. No limiter was used during the simulation in the left figure, whereas one was used in the right figure. The results in Figure 2.6 show precisely the behavior we explained above. In the unlimited case on the left side, we see that the smooth part is approximated accurately, while the discontinuous parts show unphysical oscillations and problematic under- and overshoots. In the limited case on the right side of Figure 2.6, there are no oscillations, and the physical bounds are preserved. When we take a closer look at the smooth part of the right figure, we notice a diffusive behavior at the peak induced by the limiter. This phenomenon is known as *peak clipping*.

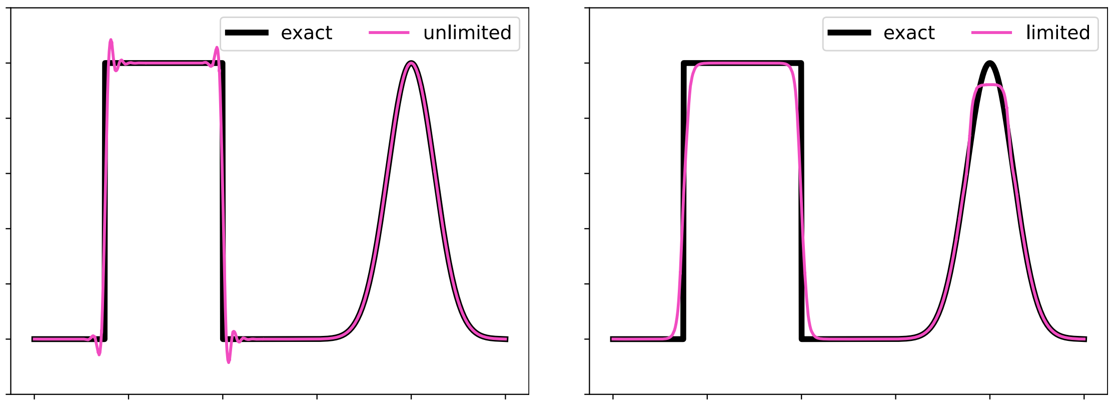


Figure 2.6: Comparison of the numerical solution of an advected discontinuous and a continuous initial profile with an unlimited and a limited numerical solution.

In one dimension, we will use the total variation diminishing in the means (TVDM) generalized slope limiter developed by Cockburn and Shu [16]. The standard scheme for limiting the discrete solution u_j on a cell E_j, (of a non-uniform mesh) can be summarized as follows:

1. Compute the limited extrapolated values $u_j^{\lim}(x_{j-\frac{1}{2}})$ and $u_j^{\lim}(x_{j+\frac{1}{2}})$:

$$\begin{aligned} u_j^{\lim}(x_{j-\frac{1}{2}}) &= \overline{u}_j - \tilde{m}(\overline{u}_j - u_j(x_{j-\frac{1}{2}}), \overline{u}_j - \overline{u}_{j-1}, \overline{u}_{j+1} - \overline{u}_j) \\ u_j^{\lim}(x_{j+\frac{1}{2}}) &= \overline{u}_j + \tilde{m}(u_j(x_{j+\frac{1}{2}}) - \overline{u}_j, \overline{u}_j - \overline{u}_{j-1}, \overline{u}_{j+1} - \overline{u}_j) \end{aligned}$$

   with $\overline{u}_j$ denoting the mass average of u_j over cell E_j and $\tilde{m}$ being the *minmod* function given by

$$\tilde{m}(a_1,\ldots,a_n) = \begin{cases} s \cdot \min_{1\le i\le n} |a_i| & \text{if } \operatorname{sign}(a_1) = \ldots = \operatorname{sign}(a_n) = s, \\ 0 & \text{otherwise.} \end{cases}$$

2. If the limited values $u_j^{\lim}(x_{j-\frac{1}{2}})$ and $u_j^{\lim}(x_{j+\frac{1}{2}})$ are equal to the unlimited values $u_j(x_{j-\frac{1}{2}})$ and $u_j(x_{j+\frac{1}{2}})$, set $u_j^{\lim} = u_j$. Otherwise, reduce u_j to P^1 by setting higher-order coefficients to zero. Then, limit the linear polynomial such that the edge evaluations of the limited polynomial do not exceed $u_j^{\lim}(x_{j-\frac{1}{2}})$ and $u_j^{\lim}(x_{j+\frac{1}{2}})$, respectively. Use the result as $u_j^{\lim}$.

These steps are applied as a postprocessing step to each intermediate solution of the stages in the SSP RK time-stepping scheme.

In two dimensions, we have run first numerical tests [25] using a Barth-Jespersen type limiter [1] that has been extended to the DG setting by exploiting the structure of the local Taylor basis [54]. This limits the gradient in such a way that the local solution of a cell E evaluated at each neighboring centroid does not exceed the maximum/minimum taken over

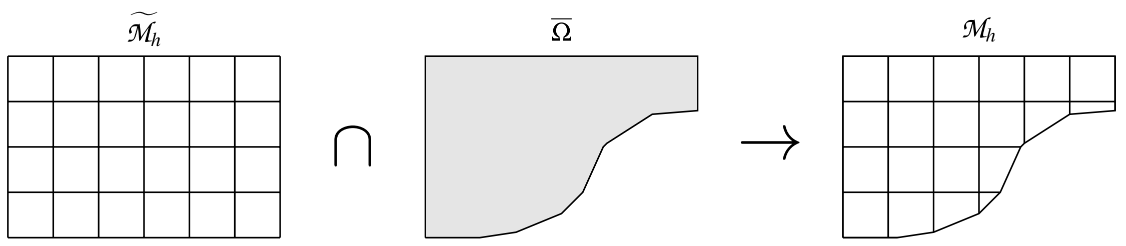


Figure 2.7: [25] Construction of a cut cell mesh $\mathcal{M}_h$: The Cartesian background mesh $\widetilde{\mathcal{M}_h}$ of a larger domain $\widetilde{\Omega}$ is intersected with the computational domain $\overline{\Omega}$, leading to cut cells $E = \widetilde{E} \cap \overline{\Omega}$, where $\widetilde{E} \in \widetilde{\mathcal{M}_h}$ is an element of the background mesh.

the cell's E average value and the average values of all of cell's E face neighbors. In this work, we will not show two-dimensional numerical results, that use a limiter.

We are now prepared and have the theoretical and numerical foundations for solving hyperbolic conservation equations. As a next step, we will dive into the topic of cut cell methods. We start with our approach for the construction of cut cell grids for a given domain Ω. Afterwards, we will continue with a deeper analysis of the small cell problem in one and two dimensions. In the remaining part of this work, we will present our approach to stabilize numerical methods for solving hyperbolic conservation laws on cut cell meshes.

2.3. The cut cell approach

Following [25], the construction of a cut cell grid $\mathcal{M}_h$ for a given open, connected domain $\Omega \subset \mathbb{R}^d$ is a trivial task. We start with a Cartesian background mesh $\widetilde{\mathcal{M}_h}$ consisting of equidistant sized cells $\widetilde{E}$ (i.e., $h_i = h,\ i \in \{1,...,d\}$), which can be interpreted as a discretization of a larger domain $\widetilde{\Omega}$ with $\Omega \subset \widetilde{\Omega}$. Intersecting $\overline{\Omega}$ and the background mesh induces the cut cell mesh

$$\mathcal{M}_h := \left\{ E := \widetilde{E} \cap \overline{\Omega} \;\middle|\; \widetilde{E} \in \widetilde{\mathcal{M}_h} \right\}.$$

Note that $\mathcal{M}_h$ is a partition of $\overline{\Omega}$ consisting of structured (Cartesian) cells and cut cells. The cut cells created at the intersected boundary have irregular shapes and can become arbitrarily small, leading to problems in the numerical simulation. Hyperbolic conservation laws can be particularly sensitive to arbitrarily small cut cells since they are typically discretized in time by explicit time-stepping schemes. The stability of these explicit schemes depends on the chosen time step size, which is directly correlated to the cell sizes. In what follows, we will first discuss the small cell problem and later present our approach to solving this problem. In order to gain a deeper insight into the small cell problem, we will study several examples in one and two dimensions. These examples will help us to better understand the small cell problem and show us key elements to consider when looking for a solution.

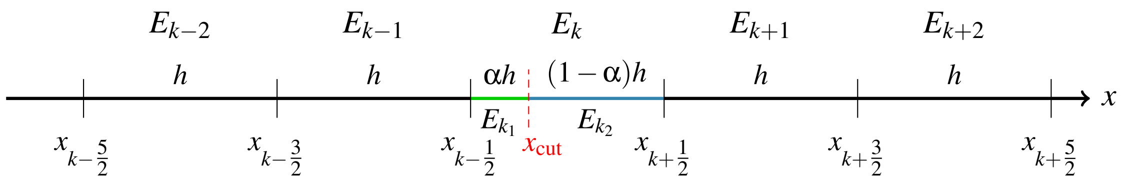


Figure 2.8: [25] Model problem **MP**: equidistant mesh with cell E_k split into two cells of lengths αh and $(1-\alpha)h$ with $\alpha \in (0, \frac{1}{2}]$.

2.3.1. Small cell problem in one dimension

We consider the linear advection equation in 1D. Without loss of generality, we consider the interval $\Omega = (0,1)$ and assume that the velocity $\beta > 0$ is constant. The PDE is given by

$$u_t(x,t) + \beta u_x(x,t) = 0 \text{ in } \Omega \times (0,T), \quad u(0,t) = g(t) \text{ for } t \in (0,T), \tag{2.85}$$

with initial data $u(x,0) = u_0(x)$.

In the 1D case, there are no true cut cell grids, but we can mimic the small cell problem. For this purpose, we introduce the model problem **MP** shown in Figure 2.8: we discretize the interval $\overline{\Omega}$ in N cells $E_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ of equidistant length $\Delta x = h$. Then we split one cell, the cell E_k, in two cells of lengths αh (referred to as cell E_{k_1}) and $(1-\alpha)h$ (referred to as cell E_{k_2}) with $\alpha \in (0, \frac{1}{2}]$.

Definition 12. *For the model problem $\mathcal{M}_h$, we define the following index sets*

$$I_{equi} = \{1 \leq j \leq N | j \neq k\}, \; I_{all} = I_{equi} \cup \{k_1, k_2\}, \; I_{Neigh} = \{k-1, k_1, k_2\}. \tag{2.86}$$

Here, I_{equi} contains the indices of all cells of length h, and I_{Neigh} contains the indices of the small cut cell E_{k_1} and its left and right neighbor.

The standard unstabilized fully-discrete update formula for a cell E_j using piecewise constant polynomials from V_h^0 and the explicit Euler method is given by:

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x_j} \left(\mathcal{H}(u_j^n, u_{j+1}^n) - \mathcal{H}(u_{j-1}^n, u_j^n) \right). \tag{2.87}$$

Inserting the upwind flux (2.69), the given cell lengths, and the time step size according to the Cartesian cells $\Delta t = \frac{\nu h}{\beta}$, the update in the neighborhood of the small cut cell reads as:

$$\begin{aligned}
u_{k-1}^{n+1} &= u_{k-1}^n - \nu \left(u_{k-1}^n - u_{k-2}^n \right), \\
u_{k_1}^{n+1} &= u_{k_1}^n - \frac{\nu}{\alpha} \left(u_{k_1}^n - u_{k-1}^n \right), \\
u_{k_2}^{n+1} &= u_{k_2}^n - \frac{\nu}{(1-\alpha)} \left(u_{k_2}^n - u_{k_1}^n \right).
\end{aligned} \tag{2.88}$$

These updates unveil the different issues that lead to the small cell problem. We follow [59] and discuss these issues in more detail:

Issue 1: Unstable update on small cut cell

First, we focus on the update of the small cut cell E_{k_1}. The standard update when using an RKDG approach with piecewise constant polynomials is given on all cells in the same form: Take the value of the old time step and modify it with a flux difference that is scaled by a fraction of the time step size and the grid size. When we focus on the update of the small cut cell E_{k_1}, we notice that the fraction in front of the flux difference is of size $O(\frac{1}{\alpha})$. In our case, we are mainly interested in how the update formula behaves when α tends to zero, which means that the cut cell becomes extremely small compared to a Cartesian cell.

Consequently, the fraction in front of the flux difference tends to infinity for $\alpha \to 0$. If this behavior holds for the whole update formula (2.88) of the small cut cell E_{k_1}, this implies that the update is unstable. This can be avoided if the flux difference is $O(\alpha h)$. In order to check this, we assume that the physical solution u is smooth. The numerical solution is computed using piecewise constant polynomials. Therefore, the constant value on each cell corresponds to the mean value of the solution on that cell. Combining this knowledge with the smoothness of the solution gives us the following behavior for the flux difference

$$u_{k_1}^n - u_{k-1}^n = O\left(\frac{(1+\alpha)h}{2}\right).$$

Multiplying the flux difference with the fraction results in

$$\frac{\nu}{\alpha}\left(u_{k_1}^n - u_{k-1}^n\right) = O\left(\frac{1}{\alpha}\right) \cdot \underbrace{O\left(\frac{(1+\alpha)h}{2}\right)}_{=O(h)} = O\left(\frac{h}{\alpha}\right).$$

Finally, we see that the update on the small cut cell E_{k_1} is unstable and expect a large under- or overshoot on E_{k_1} for a small value of α.

Issue 2: Missing information on outflow neighbor

We continue with the update of the direct neighbors of the small cut cell E_{k_1}. In the setting of our model problem **MP** and a positive velocity $\beta > 0$, the inflow neighbor is cell E_{k-1}, and the outflow neighbor is cell E_{k_2}. At this point, we emphasize that it is sufficient to focus on the outflow neighbor E_{k_2}. This is because the propagation of information is in the direction of the velocity. As a result, the inflow neighbor E_{k-1} only *sees* a Cartesian grid and will not receive any information from its small outflow neighbor.

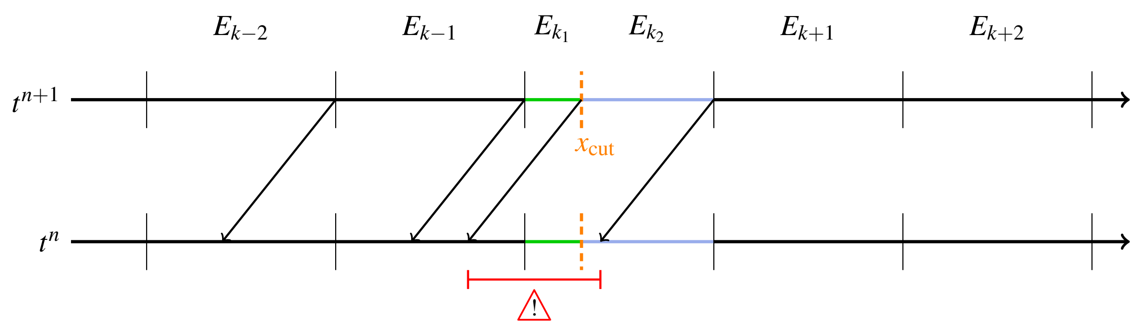


Figure 2.9: Characteristics of a constant advection on the mesh of the model problem **MP** between two time steps. The physical domain of dependence of the large outflow neighbor of the small cut cell is not included in the numerical domain of dependence of a standard update formula.

In Figure 2.9, we can see the grid of the model problem **MP** at two different time steps t^n and t^{n+1}. These time steps are connected through the characteristics shown by arrows. Moreover, the characteristics give us the domain of dependence for every cell. If we examine these DoDs more closely, we observe a mismatch between the physical DoD and the numerical standard update formula of cell E_{k_2}. The physical DoD of cell E_{k_2} (marked by the red interval and the danger sign in Figure 2.9) reaches back into cell E_{k-1}. This is a problem because cell E_{k-1} is not a direct neighbor of cell E_{k_2}, but the standard update formula (2.87) only includes information from direct neighbors. As a result, the outflow neighbors of small cut cells are missing important information, which needs to be addressed by solution approaches to achieve stability and high accuracy.

Numerical example of small cell problem

Next, we discuss a numerical example, which shows what happens if we do not fix the small cell problem. We consider the model problem **MP** and place the cell E_k in the middle of the domain such that $x_{k-\frac{1}{2}} = 0.5$. In addition, we choose the initial data

$$u_0(x) = \sin(2\pi x), \tag{2.89}$$

with the exact solution

$$u(x,t) = \sin(2\pi(x-t)). \tag{2.90}$$

We discretize the interval $\Omega = [0,1]$ in $N = 40$ cells and choose the velocity as $\beta = 1$ and the time step as $\Delta t = 0.4h$ with $h = \frac{1}{N}$.

In Figure 2.10, we show the numerical solution after one time step for different values of $\alpha = 10^{-i}$, $i \in \{1,2,3\}$. We use the unstabilized scheme for piecewise constant polynomials in combination with the explicit Euler method in time. For $\alpha = 10^{-1}$, we observe a value of approximately $u_{k_1} \approx 0.33$ on the small cut cell, leading to unphysical oscillations in the

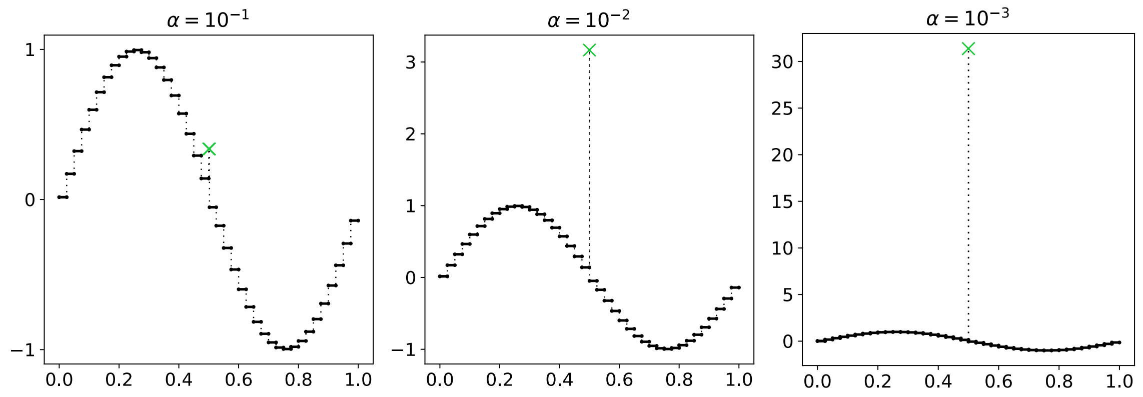


Figure 2.10: Numerical example for the small cell problem in one dimension: Solution of the unstabilized scheme after one time step for different values of $\alpha = 10^{-i}$, $i \in \{1,2,3\}$. The value of the solution on the small cut cell is illustrated with a green marker ('$\times$').

numerical solution. As we decrease to $\alpha = 10^{-2}$, a noticeable overshoot occurs on the small cut cell, approximately $u_{k_1} \approx 3.16$, which exceeds the physical bounds $[-1,1]$ of the exact solution. Further reducing α to $\alpha = 10^{-3}$ results in a substantial increase in the overshoot on the small cut cell to approximately $u_{k_1} \approx 31.38$. These results underline the influence of α on the solution update of the small cut cell k_1, as predicted in the discussion of the first issue above. As the value of α decreases, the solution on the small cut cell deteriorates, which is evident from the increasing magnitude of the overshoot. This overshoot causes the update of the neighboring outflow cells to become unstable and oscillatory in the progress of the simulation, which can further amplify the error in the solution. Consequently, this can lead to numerical instabilities and uncontrollable inaccuracies in the simulation. Therefore, it is essential to develop special schemes to ensure a stable and accurate update.

2.3.2. Small cell problem in two dimensions

As a next step, let us take a closer look at the two-dimensional case. In two dimensions, not every small cut cell is a cell that needs to be stabilized. Here, it is more important in which direction the cut cell is small. This can be observed well in the two test cases of Figure 2.11. In this setup, we have a constant advection in x-direction through a velocity field β and two different grid cases. Both grids consist primarily of equidistant square cells of size h^2 with an additional layer of small cells of sizes αh^2, $\alpha \ll 1$, which are marked in red. In the first case, the additional layer of small cells is added as a column, whereas in the second case, this layer is added as a row.

Let us examine the case of the first grid more closely. The length of the small cell in the direction of the advection is αh. Just like in the one-dimensional case, we can look at the

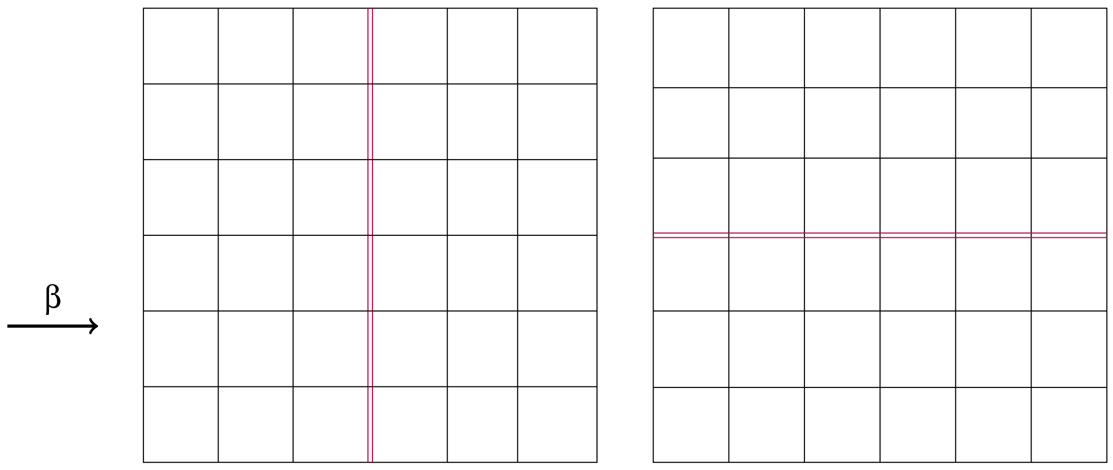


Figure 2.11: Different setups of two-dimensional anisotropic grids: For the left grid the anisotropy (red cells) is vertical to the velocity field β, whereas for the right grid the anisotropy is parallel to the velocity field.

update formula of a small cell using piecewise constant polynomials. In this case, the update formula for the two-dimensional advection equation is given by

$$\begin{aligned} u_{i,j}^{n+1} &= u_{i,j}^{n} - \frac{\Delta t}{\alpha h^2}\left(\int_{\partial E_{i,j}} H(n,u^-,u^+)\mathrm{d}s\right) \\ &= u_{i,j}^{n} - \frac{\Delta t}{\alpha h^2}\left(h\left(\beta\cdot\begin{bmatrix}1\\0\end{bmatrix}\right)u_{i-1,j}^{n} - h\left(\beta\cdot\begin{bmatrix}1\\0\end{bmatrix}\right)u_{i,j}^{n}\right) \\ &= u_{i,j}^{n} - \frac{\nu}{\alpha}(u_{i-1,j}^{n} - u_{i,j}^{n}). \end{aligned} \tag{2.91}$$

We see once again that the fraction in front of the flux difference is of size $O(\frac{1}{\alpha})$. This fraction can not be compensated with the flux difference, leading to an unstable update of these small cells. One can also trace back characteristics to get more insight into the stability of the scheme. These characteristics travel parallel to the direction of the velocity β. As a result, the outflow neighbors of the small cells miss information again. We now switch to the second case, which is given by the grid on the right side in Figure 2.11 and the same velocity field β in x-direction. We can check the update formula for a small cell in this case as well:

$$\begin{aligned} u_{i,j}^{n+1} &= u_{i,j}^{n} - \frac{\Delta t}{\alpha h^2}\left(\int_{\partial E_{i,j}} H(n,u^-,u^+)\mathrm{d}s\right) \\ &= u_{i,j}^{n} - \frac{\Delta t}{\alpha h^2}\left(\alpha h\left(\beta\cdot\begin{bmatrix}1\\0\end{bmatrix}\right)u_{i-1,j}^{n} - \alpha h\left(\beta\cdot\begin{bmatrix}1\\0\end{bmatrix}\right)u_{i,j}^{n}\right) \\ &= u_{i,j}^{n} - \nu(u_{i-1,j}^{n} - u_{i,j}^{n}). \end{aligned} \tag{2.92}$$

In this case, the fluxes are integrated over the short edges of lengths αh, which gives us an additional factor α in the update formula. Therefore, the α-dependency cancels out in the update, and consequently, the update on the small cell is stable for this setup. The same holds true when looking at the characteristics of each cell and the thereby given Domain of Dependencies. For each cell, the physical Domain of Dependence is now included in the numerical Domain of Dependence.

Another way of interpreting this behavior is when we look at an arbitrary one-dimensional hypersurface along the velocity field β for both grids. For the first grid on the left side of Figure 2.11, we will obtain a one-dimensional grid with one small cell of size αh. Therefore this grid is comparable to our one-dimensional model problem **MP** and comes with stability issues. The hypersurface for the right grid in Figure 2.11 is given as a one-dimensional equidistant grid without any small cells.

3 DoD Stabilization in one dimension

3.1. Formulation for linear scalar problems

In this section, we will consider the linear advection equation in one spatial dimension, which is given by

$$u_t(x,t) + \beta u_x(x,t) = 0 \text{ in } \Omega \times (0,T), \quad u(0,t) = g(t) \text{ for } t \in (0,T), \tag{3.1}$$

with initial data $u(x,0) = u_0(x)$. Without loss of generality, we will assume in this section that the velocity β is positive, and therefore the information propagates from left to right. We emphasize that in the event of a negative velocity $\beta < 0$, everything that follows will work similarly with a few minor adjustments. The details of negative velocity cases will become clear in the extension to a general non-linear conservation law.

If not stated otherwise, in this section, we will focus on the model problem **MP**, which is given by a mostly Cartesian grid including two cut cells of sizes αh and $(1-\alpha)h$ as shown in Figure 3.1.

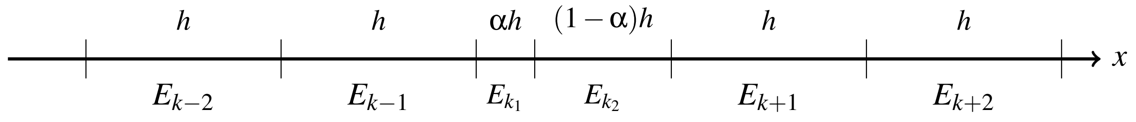


Figure 3.1: One-dimensional model problem **MP** consisting of two cut cells.

For numerical tests, we will use a modified version of the model problem **MP**. In this modified version, instead of having only one pair of cut cells, we will split every cell within a certain interval into a pair of two cut cells. Consequently, we will have multiple consecutive cut cell pairs, with one small cut cell and one larger cut cell. We believe that this modification is an excellent choice for a model problem in our numerical tests as it closely mimics cut cell grids observed in two dimensions: Small cut cells are often surrounded by larger cut cells. By incorporating this characteristic into our modified model problem, we can better capture the difficulties of the two-dimensional scenario.

In what follows, we will start with the case of piecewise constant polynomials, which was first discussed in the master's thesis [77] and later published in the collaboration [25]. We will then proceed to extend the stabilization to higher-order polynomials based on the collaborations [25, 62].

3.1.1. Piecewise constant polynomials

The general idea of the DoD stabilization is to add certain penalty terms, summarized in $J_h(\cdot,\cdot)$, to the semi-discrete formulation. The resulting DoD stabilized scheme for piecewise constant polynomials is then given by: Find $u^h \in V_h^0$ such that

$$(d_t u^h(t), w^h)_{L^2(\Omega)} + a_h(u^h(t), w^h) + J_h(u^h(t), w^h) = 0, \quad \forall w^h \in V_h^0. \tag{3.2}$$

There exist numerous penalty stabilizations for general grids in the form of adding suitable jump terms on edges to the standard DG discretization for elliptic problems. Well-known examples are the *symmetric interior penalty Galerkin (SIPG)* method [23, 82] and the *non-symmetric interior penalty Galerkin (NIPG)* [69] method. These stabilizations impose a weak continuity of the numerical solution. One can prove valuable properties for the stabilized methods, such as the existence and uniqueness of a solution and error estimates in different norms. Another example of a penalty stabilization for elliptic problems, which is more important in the area of cut cell methods, is the ghost penalty method [12, 13]. This penalty stabilization regains the coercivity of the method on small cut cells. All of the methods mentioned above have in common that the jumps in the penalty terms are evaluated on the same edge e, i.e., the stabilization terms are given in the form $J(u^h, w^h) \sim [\![u^h]\!]_e [\![w^h]\!]_e$.

When working on hyperbolic conservation laws, there is a natural direction in which information flows, which can be seen in the method of characteristics. Therefore it might be a good idea to consider this when developing new stabilization methods for hyperbolic conservation laws. In the previous chapter, we found out that the outflow neighbor of the small cut cell lacks essential information for a physically correct update. In consequence, we propose a stabilization term that is given in the following form:

$$J_h^{k_1}(u^h, w^h) = \beta\eta_{k_1} \left[\!\left[u^h\right]\!\right]_{k-\frac{1}{2}} \left[\!\left[w^h\right]\!\right]_{k_{\text{cut}}} \tag{3.3}$$

This stabilization term is given as a product of the jump $[\![\cdot]\!]$ of the numerical solution u^h and the test function w^h. The expression $[\![w^h]\!]_{k_{\text{cut}}}$ denotes the jump at the cut edge $x_{k_{\text{cut}}}$ (see Figure 2.8), i.e. $[\![w^h]\!]_{k_{\text{cut}}} = w^h|_{k_1}(x_{k_{\text{cut}}}) - w^h|_{k_2}(x_{k_{\text{cut}}})$. In addition, the stabilization term is scaled by the advection velocity β and a stabilization parameter $\eta_{k_1} \in [0,1)$, which we will discuss in detail soon. We emphasize the asymmetrical form of the stabilization, which is because we evaluate the jump of the solution u^h at the inflow face $k-\frac{1}{2}$ and the jump of the test function w^h at the outflow face k_{cut} of the small cut cell E_{k_1}. Through this asymmetric construction, we take the natural direction into account in which information flows.

As a first step, we return to the issues we identified in the last chapter and look at the effect of the stabilization. The stabilization is added to the update of the small cut cell and its outflow neighbor and does not influence the inflow neighbor. This is because the jump of the test function is evaluated at the outflow edge. Thus, the stabilized update formulas for using the explicit Euler method in time are given by

$$\begin{aligned}
u_{k-1}^{n+1} &= u_{k-1}^n - \frac{\Delta t}{h}\beta\left(u_{k-1}^n - u_{k-2}^n\right), \\
u_{k_1}^{n+1} &= u_{k_1}^n - \frac{\Delta t}{\alpha h}\beta\left(u_{k_1}^n - u_{k-1}^n\right) - \frac{\Delta t}{\alpha h}\beta\eta_{k_1}\left(u_{k-1}^n - u_{k_1}^n\right), \\
u_{k_2}^{n+1} &= u_{k_2}^n - \frac{\Delta t}{(1-\alpha)h}\beta\left(u_{k_2}^n - u_{k_1}^n\right) + \frac{\Delta t}{(1-\alpha)h}\beta\eta_{k_1}\left(u_{k-1}^n - u_{k_1}^n\right).
\end{aligned} \tag{3.4}$$

Issue 1: Unstable update on small cut cell

We recall that the first issue is an unstable update on the small cut cell, which is induced by an $O\left(\frac{1}{\alpha}\right)$ dependency in the unstabilized update formula

$$u_{k_1}^{n+1} = u_{k_1}^n - \frac{\nu}{\alpha}\left(u_{k_1}^n - u_{k-1}^n\right). \tag{3.5}$$

We compare this to the stabilized update formula given by (3.4). We simplify this to get

$$u_{k_1}^{n+1} = u_{k_1}^n - \frac{\nu}{\alpha}\left(1-\eta_{k_1}\right)\left(u_{k_1}^n - u_{k-1}^n\right). \tag{3.6}$$

Therefore, when analyzing the two update formulas, we notice that the only but essential difference is the additional scaling term $1-\eta_{k_1}$. A suitable choice for the stabilization parameter η_{k_1} is $\eta_{k_1} \sim 1-\alpha$. This results in $1-\eta_{k_1} \sim \alpha$ and therefore removes the $O\left(\frac{1}{\alpha}\right)$ dependency in the update formula. Consequently, the stabilized update formula is stable independent of the choice of α.

Issue 2: Missing information on outflow neighbor

The second issue, which arises when we study the physical and numerical domain of dependence of the outflow neighbor E_{k_2}, will be addressed next. When using the unstabilized

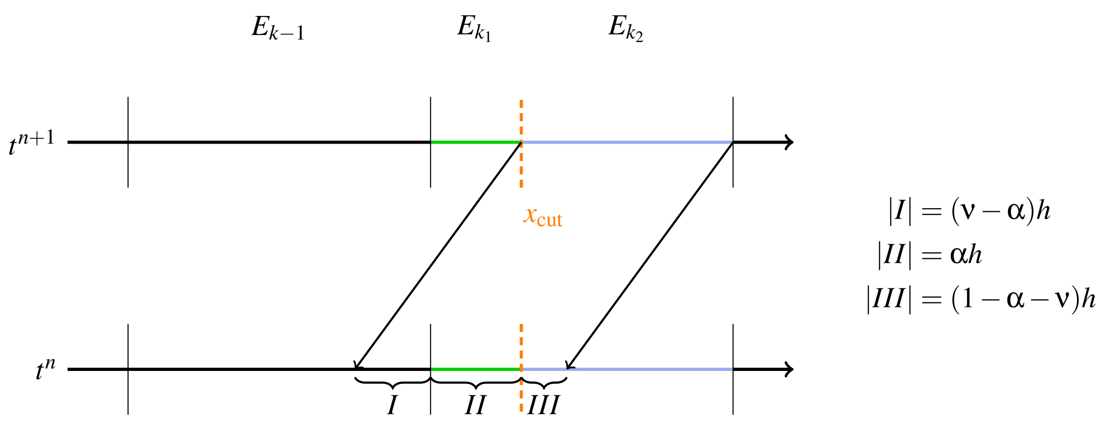


Figure 3.2: The size of the different subintervals of the domain of dependence of cell E_{k_2}.

update formula, we notice that the outflow neighbor of the small cut cell is missing important information. The stabilized update formula of E_{k_2} is given by

$$u_{k_2}^{n+1} = u_{k_2}^n - \frac{\nu}{1-\alpha}\left(u_{k_2}^n - (1-\eta_{k_1})u_{k_1}^n - \eta_{k_1}u_{k-1}^n\right). \tag{3.7}$$

We notice that the stabilized update formula of cell E_{k_2} now contains information from cell E_{k-1}. We assume once again for the stabilization parameter that $\eta_{k_1} \sim 1-\alpha$. Then, the influence of the small cut cell value u_{k_1} is scaled down by a factor $1-\eta_{k_1} \sim \alpha$, and this information is replaced by the physical admissible values from u_{k-1}.

In conclusion, the proposed stabilization term provides the right tools to tackle the two issues of the small cell problem. It remains to choose a suitable value for the stabilization parameter η_{k_1}, which we will discuss in the next section.

Theoretical properties

Choice of η_{k_1}

For the choice of the stabilization parameter η_{k_1}, we once again look at the domain of dependence of the outflow neighbor E_{k_2}, see Figure 3.2. By tracing back the characteristics, we can figure out how the solution at the new time t^{n+1} is composed of the given solutions at time t^n. This gives us the following physically exact update for the outflow neighbor of the small cell when starting with piecewise constant values at t^n

$$\begin{aligned} u_{k_2}^{n+1} &= \frac{1}{|E_{k_2}|}\left(|I|u_{k-1}^n + |II|u_{k_1}^n + |III|u_{k_2}^n\right) \\ &= \frac{\nu-\alpha}{1-\alpha}u_{k-1}^n + \frac{\alpha}{1-\alpha}u_{k_1}^n + \frac{1-\alpha-\nu}{1-\alpha}u_{k_2}^n. \end{aligned} \tag{3.8}$$

We compare the stabilized update formula in (3.7) with the composition mentioned in (3.8). Therefore, we reformulate (3.7) and sort it by the different cell averages

$$\begin{aligned} u_{k_2}^{n+1} &= u_{k_2}^n - \frac{\nu}{1-\alpha}\left(u_{k_2}^n - (1-\eta_{k_1})u_{k_1}^n - \eta_{k_1}u_{k-1}^n\right) \\ &= \frac{\nu\eta_{k_1}}{1-\alpha}u_{k-1}^n + \frac{\nu(1-\eta_{k_1})}{1-\alpha}u_{k_1}^n + \frac{1-\alpha-\nu}{1-\alpha}u_{k_2}^n. \end{aligned} \tag{3.9}$$

The comparison of equations (3.8) and (3.9) reveals that the stabilized scheme is exact for piecewise constant data at t^n, if the stabilization parameter η_{k_1} is chosen as

$$\eta_{k_1} = 1 - \frac{\alpha}{\nu}. \tag{3.10}$$

Furthermore, this choice of the stabilization parameter satisfies the condition $\eta_{k_1} \sim 1-\alpha$, which is mandatory to fix the first issue of an unstable update on cell E_{k_1}.

Finally, the stabilization term should vanish in the special case when the time step is small enough such that the characteristics of the cell E_{k_2} only reach back into its direct inflow neighbor E_{k_1}. This is the case for $\alpha > \nu$ and, hence, motivates the final definition of the stabilization parameter as

$$\eta_{k_1} = 1 - \min\left(\frac{\alpha}{\nu}, 1\right). \tag{3.11}$$

There is limited flexibility in the choice of the stabilization parameter to fix the two issues of the small cell problem. In the next section, we will define a range of suitable values based on theoretical foundations.

In what follows, we will discuss various properties of the DoD stabilization, which were presented first in other works [25, 62]. We will label the corresponding results adequately.

Monotonicity

One desirable property for a first-order scheme is to be monotone when solving hyperbolic conservation laws. This monotonicity implies a maximum principle for the numerical solution, which guarantees that unphysical overshoots and undershoots cannot occur. The major drawback of monotone schemes is that they are at most first-order accurate.

In the next steps, we study the monotonicity and TVD properties of the DoD stabilized scheme.

Theorem 13. *[77, Theorem 10] Consider the model problem **MP** 2.8 discretized in time by the explicit Euler scheme. Let the time step be given by $\Delta t = \frac{\nu h}{|\beta|}$ for $0 < \alpha < \nu < 1-\alpha$. Then, the method is monotone in the sense of Definition 8.*

Proof. The proof of this statement was originally presented in the master's thesis [77, Theorem 10], but for an updated version, we refer to [25].In addition, the monotonicity of the method is shown for the more general case of a non-linear scalar equation in Theorem 21 below. □

TVD stability

Next, we show that the stabilized scheme discretized in time using the *explicit Euler* scheme is TVD stable for the model problem 2.8 under a CFL condition, that is independent of the value of α.

Lemma 14. *[25, Lemma 4.8] Consider the model problem **MP** discretized in time using the explicit Euler scheme. Then, the stabilized scheme is TVD stable for* $\nu < 1-\alpha$.

Proof. We decompose

$$TV(u^{n+1}) = \sum_j |u_j^{n+1} - u_{j-1}^{n+1}| = \underbrace{\sum_{j\le k-1} |u_j^{n+1} - u_{j-1}^{n+1}|}_{T_1} + \underbrace{|u_{k_1}^{n+1} - u_{k-1}^{n+1}|}_{T_2} + \underbrace{|u_{k_2}^{n+1} - u_{k_1}^{n+1}|}_{T_3}$$
$$+ \underbrace{|u_{k+1}^{n+1} - u_{k_2}^{n+1}|}_{T_4} + \underbrace{\sum_{j\ge k+2} |u_j^{n+1} - u_{j-1}^{n+1}|}_{T_5}.$$

For the unstabilized parts of the scheme, we obtain

$$T_1 \le \sum_{j\le k-2} |u_j^n - u_{j-1}^n| + (1-\nu)|u_{k-1}^n - u_{k-2}^n|, \quad T_5 \le \sum_{j\ge k+2} |u_j^n - u_{j-1}^n| + \nu|u_{k+1}^n - u_{k_2}^n|.$$

Using the stabilized update formulas in (3.4), the direct substitution of the formulae results in

$$T_2 \le \nu|u_{k-1}^n - u_{k-2}^n|.$$

For T_3 and T_4, we reorder the terms resulting from the formulas (3.4) to get

$$T_3 \le \left(1 - \frac{\nu}{1-\alpha}\right)|u_{k_2}^n - u_{k_1}^n| + \left(1 - \frac{\nu-\alpha}{1-\alpha}\right)|u_{k_1}^n - u_{k-1}^n|,$$
$$T_4 \le (1-\nu)\,|u_{k+1}^n - u_{k_2}^n| + \frac{\nu}{1-\alpha}|u_{k_2}^n - u_{k_1}^n| + \frac{\nu-\alpha}{1-\alpha}|u_{k_1}^n - u_{k-1}^n|.$$

We emphasize that all terms are positive due to the assumptions made. Summing up the estimates for $T_1, \ldots, T_5$ implies the claim. □

This TVD stability guarantees that the DoD-stabilized scheme does not develop any spurious oscillations. Next, we compare the DoD stabilization to the well-known *h*-box method. This part is based on the work in [25].

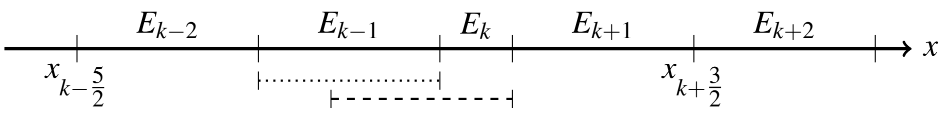


Figure 3.3: [25] One-dimensional FV model mesh and boxes used for the h-box stabilization.

Comparison to the h-box method

The idea behind the proposed stabilization is similar to that of the h-box method [7, 8, 42]. In this approach, one creates boxes of length h at the cut cell edges to reconstruct the appropriate domains of dependence for small cut cells and their outflow neighbors. These boxes are then used for the flux computation on cut cell edges.

We compare the FV-based h-box method to our DG stabilization for piecewise constant polynomials and explicit Euler method in time. We consider a 1D mesh (which is often used as a cut model for FV schemes) consisting of elements with mesh size h with one small cell of length αh, $\alpha \in (0,1]$, referred to as cell E_k in the interior of the domain (see Figure 3.3). We solve the advection equation (2.85) with $\beta = 1$ using upwind fluxes. Then, for both methods, the update formulae on cells $i \leq k-1$ and $i \geq k+2$ correspond to the unstabilized upwind scheme.

The h-box method on the cut cell k results in the update formula

$$u_k^{n+1} = u_k^n - \frac{\Delta t}{\alpha h}\left(u_{k+\frac{1}{2}}^{n,L} - u_{k-\frac{1}{2}}^{n,L}\right) \tag{3.12}$$

with upwind fluxes

$$u_{k-\frac{1}{2}}^{n,L} = u_{k-1}^n \qquad \text{and} \qquad u_{k+\frac{1}{2}}^{n,L} = \hat{\eta} u_k^n + (1-\hat{\eta}) u_{k-1}^n.$$

Here, $u_{k+\frac{1}{2}}^{n,L}$ is constructed using linear interpolation with a parameter $\hat{\eta}$. This results in

$$u_k^{n+1} = u_k^n - \frac{\Delta t}{\alpha h}\hat{\eta}\left(u_k^n - u_{k-1}^n\right). \tag{3.13}$$

The DoD stabilization (3.3) for this setting is given by

$$u_k^{n+1} = u_k^n - \frac{\Delta t}{\alpha h}(1-\eta)\left(u_k^n - u_{k-1}^n\right). \tag{3.14}$$

Substituting $\hat{\eta} = 1 - \eta$, equations (3.13) and (3.14) are identical. The same can be shown for the update of the cell $k+1$ and, thus, both methods coincide up to the choice of the stabilization parameter.

In [8] the following two values for η are proposed:

$$\eta_1 = 1 - \alpha \quad \text{and} \quad \eta_2 = 1 - \frac{2\alpha}{1+\alpha} \tag{3.15}$$

The choice η_1 is based on cell averaging, and the choice η_2 on optimizing the one-step error: it is the only choice that leads to a second-order one-step error (compared to first-order otherwise). We propose $\eta_k = 1 - \frac{\alpha}{\nu}$. The choice η_k results in the exact advection of a piecewise constant solution in the situation above.

The apparent benefit of η_2 regarding the one-step error does not impact the accuracy at time T. This phenomenon, where the global error behaves better than expected based on the local error, is called *supraconvergence*. For η_1, [8] shows global first-order convergence and a variant of the proof can be shown for our suggestion $\eta_k = 1 - \frac{\alpha}{\nu}$.

Proposition 4. *[25, Proposition 4.9] We consider the above mentioned situation (mesh shown in Figure 3.3, upwind scheme on all cells and stabilized by the DoD stabilization on cell E_k), which can be summarized as*

$$\begin{aligned}
u_j^{n+1} &= u_j^n - \frac{\Delta t}{h}\left(u_j^n - u_{j-1}^n\right), \quad j \in \{1,...,N\}/\{k,k+1\},\\
u_k^{n+1} &= u_k^n - \frac{\Delta t}{\alpha h}(1-\eta)\left(u_k^n - u_{k-1}^n\right),\\
u_{k+1}^{n+1} &= u_{k+1}^n - \frac{\Delta t}{h}\left(u_{k+1}^n - (1-\eta)u_k^n - \eta u_{k-1}^n\right).
\end{aligned}$$

Then, using $\eta = \eta_k$, the scheme is of first-order for sufficiently smooth solutions.

Proof. The proof is based on the ideas of Wendroff and White [80, 81] and is highly influenced by the proof of Berger et al. [8, Proposition 1]. The basic idea is to find an appropriate grid function r, which is a first-order approximation of the grid function of the exact solution u. At the same time, this grid function r needs to be chosen such that the truncation error for the grid of the model problem is of first-order. We define the grid function r with a parameter ζ as

$$r_i^n = u_i^n + \zeta_i h u_x(x_i, t^n). \tag{3.16}$$

On the Cartesian part of the grid away from the small cut cell, we choose the parameter $\zeta_i = 0$, $i \neq k$ because the truncation error is already of first-order here. In the neighborhood of the small cut cell, the parameter ζ_k must still be chosen. The truncation error of the update formula (3.14) is given by

$$\begin{aligned}
Lr =& \frac{r_k^{n+1} - r_k^n}{\Delta t} + \frac{1-\eta}{\alpha}\frac{r_k^n - r_{k-1}^n}{h} \\
=& \frac{u_k^{n+1} + \zeta_k h u_x(x_k, t^{n+1}) - u_k^n - \zeta_k h u_x(x_k, t^n)}{\Delta t} \\
&+ \frac{1-\eta}{\alpha} \cdot \frac{u_k^n + \hat{\eta}_k h u_x(x_k, t^n) - u_{k-1}^n}{h} + O(\Delta t, h) \\
=& \frac{u_k^n + \Delta t u_t(x_k, t^n) + \zeta_k h u_x(x_k, t^n) - u_k^n - \zeta_k h u_x(x_k, t^n)}{\Delta t} \\
&+ \frac{1-\eta}{\alpha} \cdot \frac{u_k^n + \zeta_k h u_x(x_k, t^n) - u_k^n + \frac{1}{2}(1+\alpha) h u_x(x_k, t^n)}{h} + O(\Delta t, h) \\
=& u_t(x_k, t^n) + \frac{(1-\eta)(\zeta_k + \frac{1}{2}(1+\alpha))}{\alpha} u_x(x_k, t^n) + O(\Delta t, h).
\end{aligned}$$

For the truncation error to be of first-order, we need to choose the parameter ζ_k such that the prefactor of $u_x(x_k, t^n)$ is equal to 1. This gives us the following condition

$$\begin{aligned}
1 &\overset{!}{=} \frac{(1-\eta)(\zeta_k + \frac{1}{2}(1+\alpha))}{\alpha} \\
\Leftrightarrow \quad \zeta_k &= \frac{\alpha}{1-\eta} - \frac{1}{2}(1+\alpha)
\end{aligned}$$

For our choice of the stabilization parameter $\eta_k = 1 - \frac{\alpha}{\nu}$ we obtain the grid function

$$r_k^n = u_k^n + \left(\nu - \frac{1}{2}(1+\alpha)\right) h u_x(x_k, t^n).$$

Since the truncation error of r_k^n is of order $O(\Delta t, h)$ and there holds $r = u + O(h)$, this completes the proof. $\square$

Remark 15. *The result of Proposition 4 can also be proven for the model mesh shown in Figure 2.8, using the additional modification of the grid functions on cells E_{k_1} and E_{k_2}:*

$$\begin{aligned}
r_{k_1}^n &= u_{k_1}^n + \left(\nu - \frac{1}{2}(1+\alpha)\right) h u_x(x_{k_1}, t^n), \\
r_{k_2}^n &= u_{k_2}^n - \frac{\alpha}{2} h u_x(x_{k_2}, t^n).
\end{aligned}$$

3.1.2. Higher-order polynomials

We now consider the extension of the stabilization to higher-order polynomials. The principles outlined in this section were initially introduced for piecewise linear polynomials in [25] and was later extended to higher-order polynomials in [62]. We keep the general idea

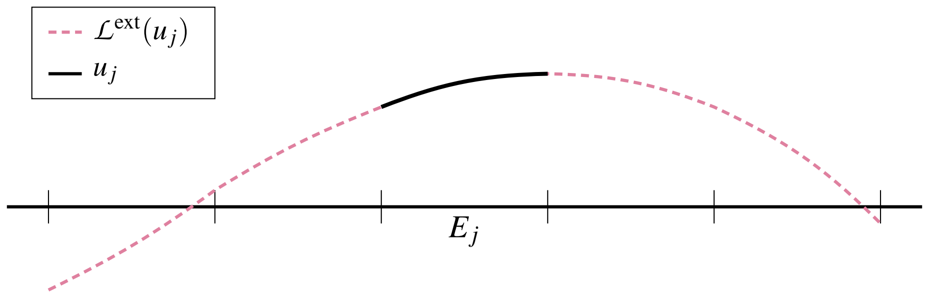


Figure 3.4: Sketch of the extended version $\mathcal{L}^{\text{ext}}(u_j)$ (purple dashed) of a piecewise polynomial solution u_j on cell E_j (black line).

that the DoD stabilization should transfer mass from the inflow neighbor directly to the outflow neighbor to ensure a proper mass distribution among the cut cell and its neighbors. Again we achieve this by adding an additional flux at the outflow edge of the small cut cell. Unlike in the case of piecewise constant functions, we must now carefully choose at which point we evaluate the functions. The stabilization is based on extending the influence of the polynomial solutions on cells E_{k-1} and E_{k_2} to the small cut cell E_{k_1}. We do this by means of an *extension operator* $\mathcal{L}^{\text{ext}}$.

Definition 16 (Extension operator)**.** *We introduce an extension operator $\mathcal{L}^{ext}_{E'}$ that extends a function $u^h \in V_h^k$ from a cell $E' \in \mathcal{M}_h$ to the whole domain Ω:*

$$\mathcal{L}^{ext}_{E'} : V_h^k(\mathcal{M}_h)|_{E'} \to P^k(\Omega) \qquad s.t.\ \mathcal{L}^{ext}_{E'}(u^h) \in P^k(\Omega)\ and\ \mathcal{L}^{ext}_{E'}(u^h)|_{E'} = u^h|_{E'}.$$

This extension is trivial as $u^h|_{E'} \in P^k(E')$ and polynomials can be evaluated outside of their original support.

In this work, we will use a simpler notation for an extended function as described in what follows.

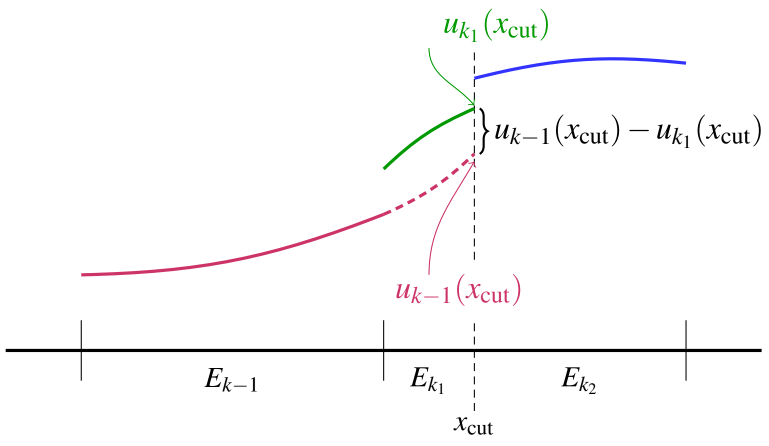
Notation 17. *We will drop the extension operator $\mathcal{L}^{ext}_{E_j}$ and use the notation*

$$u_{E_j}(x), \quad x \in \Omega,$$

to indicate that the discrete polynomial function u_{E_j} from cell E_j is evaluated at a point x, possibly outside of E_j. See Figure 3.4 for an illustration of the extension operator.

The DoD stabilization for higher-order polynomials is given by two terms $J_h^{0,k_1}(\cdot,\cdot)$ and $J_h^{1,k_1}(\cdot,\cdot)$. The first stabilization term $J_h^{0,k_1}(\cdot,\cdot)$ is given by:

$$J_h^{0,k_1}(u^h,w^h) = \beta\eta_{k_1}\left[u_{k-1}(x_{\text{cut}}) - u_{k_1}(x_{\text{cut}})\right]\left[\!\left[w^h\right]\!\right]_{\text{cut}}. \tag{3.17}$$



$u_{k_1}(x_{\text{cut}})$
$\}\, u_{k-1}(x_{\text{cut}}) - u_{k_1}(x_{\text{cut}})$
$u_{k-1}(x_{\text{cut}})$
E_{k-1}
E_{k_1}
E_{k_2}
x_{cut}

Figure 3.5: Sketch of the extended jump term between u_{k-1} and u_{k_1} that is evaluated at x_{cut}.

Note that we use the extended solution of the inflow neighbor to determine the size of the correction at the outflow edge. See Figure 3.5 for a visualization of the extended jump term.

After restoring a more physical mass distribution in the neighborhood of the small cut cell, we need to restore control over the gradients. Therefore we introduce a second stabilization term $J_h^{1,k_1}(\cdot,\cdot)$ that is given in the form of a penalty term, which is evaluated on cell volumes. It controls the mass distribution primarily *within* the small cut cell E_{k_1} and secondarily *within* its neighborhood. The stabilization accounts for how much mass has been moved into and out of the small cut cell E_{k_1} from and to its left and right neighbors through $a_h(\cdot,\cdot)$ and $J_h^{0,k_1}(\cdot,\cdot)$. The second stabilization term $J_h^{1,k_1}(\cdot,\cdot)$ is given by:

$$J_h^1(u^h, w^h) = \beta\eta_{k_1} \int_{k_1} [u_{k-1}(x) - u_{k_1}(x)] \, [\partial_x w_{k-1}(x) - \partial_x w_{k_1}(x)] \, \mathrm{d}x. \tag{3.18}$$

We apply the extension operator to both the discrete solution and the test function of the inflow neighbor E_{k-1}. The stabilization term $J_h^{1,k_1}(\cdot,\cdot)$ was initially introduced in a simplified version in [25] and subsequently expanded to its current form (3.18) in [62].

Finally, the full stabilization $J_h(\cdot,\cdot)$ for cell k_1 is then given by

$$J_h^{k_1}(u^h, w^h) = J_h^{0,k_1}(u^h, w^h) + J_h^{1,k_1}(u^h, w^h). \tag{3.19}$$

Theoretical properties

Next, we will present theoretical properties for piecewise polynomials of higher-order. We will start with a symbolic eigenvalue analysis for a small one-dimensional cut cell problem. This will illustrate, that the given method is absolute stable independent of the size of the volume fraction α. After that, we will examine TVDM stability and L^2 stability for the stabilized method.

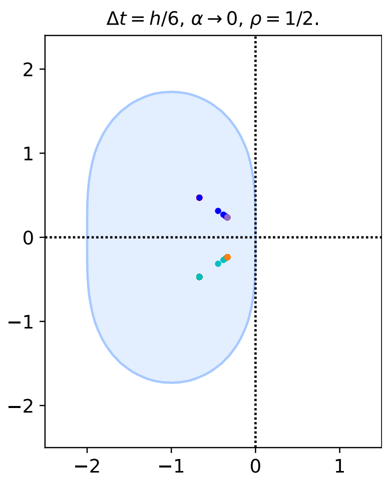


Figure 3.6: Eigenvalues of (3.22): The stability region of the second-order Heun's method is shown in blue. The colors refer to the different eigenvalues and show their evolution for decreasing α ($\alpha = 2^{-i}$, $i \in [1, \dots, 10]$).

Choice of η_{k_1}

We now illustrate the stability of our stabilized scheme in combination with the second-order explicit RK scheme (2.78) by means of an eigenvalue analysis using symbolic computations with `sympy`[64]. This part is based on the collaborated work [25]. We consider the model problem **MP** for the special case of $N = 5$, i.e., we start with four equidistant cells of length h and split the third cell into two cells of length αh and length $(1-\alpha)h$. We use Dirichlet boundary conditions and without loss of generality set $\beta = 1$.

We consider the stabilized scheme

$$(d_t u^h(t), w^h)_{L^2(\Omega)} + a_h(u^h(t), w^h) + J_h(u^h(t), w^h) = 0, \tag{3.20}$$

with the stabilization term $J_h(\cdot,\cdot)$ given by (3.19). For the construction of the stabilization term $J_h(\cdot,\cdot)$, we have made several design choices, e.g., the choice of η_{k_1} and the general structure of the terms. We rewrite the variational formulation (3.20) as a system of ODEs

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}(t) = -M^{-1}(A+J)\mathbf{u}(t) \tag{3.21}$$

with $\mathbf{u}(t)$ being the coefficient vector of the discrete solution $u^h(t)$, M being the mass matrix, A being the stiffness matrix incorporating $a_h(\cdot,\cdot)$, and J incorporating the corresponding parts of $J_h(\cdot,\cdot)$. We then symbolically set up the scaled operator

$$R = -\Delta t M^{-1}(A+J), \tag{3.22}$$

which is the stability function used in the ODE stability analysis.

Following [58], for the analysis of the stability of a scalar ordinary differential equation, we consider the simple test equation

$$\frac{\mathrm{d}}{\mathrm{d}t}y(t) = \lambda y(t) \tag{3.23}$$

with $\lambda \in \mathbb{C}$ being a complex number. Applying the second-order Heun's method to the scalar problem (3.23) gives

$$y^{n+1} = \left(1 + \Delta t\lambda + \tfrac{1}{2}(\Delta t\lambda)^2\right) y^n. \tag{3.24}$$

The weight $\omega(\Delta t\lambda) = 1 + \Delta t\lambda + \frac{1}{2}(\Delta t\lambda)^2$ is called *amplification factor* and we say the method is absolute table when $|\omega(\Delta t\lambda)| \leq 1$; otherwise it is unstable. We then define the *stability region* (SR) of the method, which is given by the following set

$$SR = \left\{z \in \mathbb{C} \text{ with } \left|1 + z + \tfrac{1}{2}z^2\right| \leq 1\right\}.$$

Consequently, we obtain the stability of the method, if we choose Δt such that $\Delta t\lambda \in SR$.

In the setting of systems of ordinary differential equations as given by (3.21) we can extend the ideas of the scalar case. Similar to the scalar case, we define the amplification matrix $\omega(R) = I + R + \frac{1}{2}R^2$ as the weight associated with the numerical method. The stability of the method is achieved when $|\omega(R)| \leq 1$, indicating that the amplification matrix for all eigenvalues of R remains within the desirable range. Conversely, if $|\omega(R)| > 1$ for any eigenvalue, the method is considered unstable.

Next, we check the distribution of the eigenvalues for our stabilized scheme. As $\alpha \leq \frac{1}{2}$, a lower bound for the size of the unstabilized cut cell E_{k_2} is $\frac{h}{2}$. The limiting CFL number ν for piecewise linear polynomials is then $\nu = \frac{1}{2}$. We therefore choose $\Delta t = \frac{h}{6}$ to compute the eigenvalues of R. For the limit $\alpha \to 0$, we find the following complex eigenvalues:

$$\begin{aligned}
\lambda_{1,2} &= -\frac{2 \pm \sqrt{2}i}{3}, \\
\lambda_{3,4,5,6} &= -\frac{2 + \sqrt{2}i}{6}, \\
\lambda_{7,8,9,10} &= -\frac{2 - \sqrt{2}i}{6}.
\end{aligned}$$

For a sequence of decreasing α, we visualize the eigenvalues of R and observe that for $\Delta t = \frac{h}{6}$ all eigenvalues stay within the stability region of the explicit second-order SSP RK scheme (2.78) (Figure 3.6), indicating the stability of the explicit time-stepping scheme.

TVDM stability

Next, we will continue with the natural extension of TVD stability to higher-order polynomials: TVDM stability. In general, we can only expect to obtain a TVDM stability result for higher-order polynomials if we apply a limiter. In Section 2.2.6, we discussed the TVDM generalized slope limiter [16], which we will use for the 1D examples. For the proof of the TVDM property, we need to modify the limiting process on the stabilized cells.

In the penalty term $J_h(\cdot,\cdot)$, we evaluate the solutions of cell E_{k-1} outside of its original support. We, therefore, extend the limiting on this cell by additionally enforcing

$$\min\left(\overline{u}^n_{k-1},\overline{u}^n_{k_1},\overline{u}^n_{k_2}\right) \le u_{h,k-1}(x_{\text{cut}}) \le \max\left(\overline{u}^n_{k-1},\overline{u}^n_{k_1},\overline{u}^n_{k_2}\right). \tag{3.25}$$

As for the standard cells, we first check whether or not it is necessary to change the high-order polynomial (see Step 1) and only adjust the solution if needed.

Lemma 18. *[25, Lemma 4.10.] Consider the model problem **MP** discretized with the explicit Euler method in time. Assume that the limiter has been modified on cell E_{k-1} to additionally enforce (3.25). Then the scheme is TVDM stable for $\nu < \frac{1}{4}$.*

Proof. The result can be shown similarly to the proof of Lemma 14 and can be found in [25] for a slightly different choice of the stabilization parameter η_{k_1}. □

Corollary 19. *[25, Corollary 4.11.] Consider **MP** discretized with the second-order SSP RK scheme (2.78) in time. Assume that the limiter has been modified on cell E_{k-1} to enforce (3.25). Then the scheme is TVDM stable for $\nu < \frac{1}{4}$.*

Proof. The result follows directly from the fact that SSP RK schemes are constructed to be convex combinations of explicit Euler steps [36]. □

L^2 stability

Next, we will discuss the L^2 stability of the semi-discrete scheme (3.20). We consider the semi-discrete problem for the linear advection equation in 1D, which is given by

$$\left(d_t u^h(t), w^h\right)_{L^2(\Omega)} + a_h\left(u^h(t), w^h\right) + J_h(u^h(t), w^h) = 0 \tag{3.26}$$

using the bilinear form

$$a_h(u^h, w^h) = -\sum_{j\in I_{\text{all}}} \int_j \beta u^h \partial_x w^h \mathrm{d}x + \sum_{j=0}^{N} \beta u^h(x^-_{j+1/2}) \left[\!\left[w^h \right]\!\right]_{j+1/2} + \beta u^h(x^-_{\text{cut}}) \left[\!\left[w^h \right]\!\right]_{\text{cut}} \tag{3.27}$$

and the stabilization term

$$\begin{aligned} J_h(u^h(t), w^h) = {} & \beta\eta_{k_1} \left[u_{k-1}(x_{\text{cut}}) - u_{k_1}(x_{\text{cut}})\right]\left[w_{k_1}(x_{\text{cut}}) - w_{k_2}(x_{\text{cut}})\right]_{\text{cut}} \\ & + \beta\eta_{k_1} \int_{k_1} \left[u_{k-1}(x) - u_{k_1}(x)\right]\left[\partial_x w_{k-1}(x) - \partial_x w_{k_1}(x)\right] \mathrm{d}x. \end{aligned} \tag{3.28}$$

For this scheme, we obtain the following result.

Theorem 20. *Let $u^h(t)$, with $u^h(t) \in V_h^p$ for any fixed t, be the solution to the semi-discrete problem (3.26) for the linear advection equation (2.3) with periodic boundary conditions. Furthermore, let the evaluation of the volume integrals in $a_h(\cdot,\cdot)$ and $J_h(\cdot,\cdot)$ be exact. Then, the solution satisfies for all $t \in (0,T)$*

$$\left\| u^h(t) \right\|_{L^2(\Omega)} \leq \left\| u^h(0) \right\|_{L^2(\Omega)}.$$

Proof. Choosing $w^h = u^h(t)$ in (3.26) results in

$$\left(d_t u^h(t), u^h(t) \right)_{L^2(\Omega)} + a_h \left(u^h(t), u^h(t) \right) + J_h(u^h(t), u^h(t)) = 0. \tag{3.29}$$

We integrate in time to achieve

$$\int_0^t \left(d_\tau u^h(\tau), u^h(\tau) \right) \mathrm{d}\tau = \int_0^t \frac{d}{d\tau} \frac{1}{2} \left\| u^h(\tau) \right\|_{L^2(\Omega)}^2 \mathrm{d}\tau = \frac{1}{2} \left\| u^h(t) \right\|_{L^2(\Omega)}^2 - \frac{1}{2} \left\| u^h(0) \right\|_{L^2(\Omega)}^2.$$

Therefore, it remains to show that for any fixed t

$$a_h \left(u^h(t), u^h(t) \right) + J_h(u^h(t), u^h(t)) \geq 0. \tag{3.30}$$

From now on, we will omit the explicit time dependence for brevity. We first start with the unstabilized case and consider only the bilinear form $a_h(\cdot,\cdot)$. Using direct calculations and reordering terms, equation (3.27) can be written as

$$a_h(u^h, u^h) = \sum_{j \in I_{\text{equi}}} \frac{1}{2} \beta \left[\!\left[u^h \right]\!\right]_{j-\frac{1}{2}}^2 + \frac{1}{2} \beta \left[\!\left[u^h \right]\!\right]_{k-\frac{1}{2}}^2 + \frac{1}{2} \beta \left[\!\left[u^h \right]\!\right]_{\text{cut}}^2 \tag{3.31}$$

due to the periodic boundary conditions. Note that we have used the periodic boundary conditions here. For obvious reasons, all terms in equation (3.31) are non-negative.

For the stabilization term, we find that

$$\begin{aligned}
J_h(u^h, u^h) =& \beta \eta_{k_1} \left[u_{k-1}(x_{\text{cut}}) - u_{k_1}(x_{\text{cut}}) \right] \left[u_{k_1}(x_{\text{cut}}) - u_{k_2}(x_{\text{cut}}) \right]_{\text{cut}} \\
& + \beta \eta_{k_1} \int_{k_1} \left[u_{k-1}(x) - u_{k_1}(x) \right] \left[\partial_x u_{k-1}(x) - \partial_x u_{k_1}(x) \right] \mathrm{d}x \\
=& \beta \eta_{k_1} \Big(u_{k-1}(x_{\text{cut}}) u_{k_1}(x_{\text{cut}}) - u_{k-1}(x_{\text{cut}}) u_{k_2}(x_{\text{cut}}) \\
& - u_{k_1}^2(x_{\text{cut}}) + u_{k_1}(x_{\text{cut}}) u_{k_2}(x_{\text{cut}}) \\
& + (u_{k-1} - u_{k_1})^2(x_{\text{cut}}) - (u_{k-1} - u_{k_1})^2(x_{k-\frac{1}{2}}) \Big) \\
=& \frac{1}{2} \beta \eta_{k_1} \left(- \left[\!\left[u^h \right]\!\right]_{k-\frac{1}{2}}^2 - \left[\!\left[u^h \right]\!\right]_{\text{cut}}^2 + \left[u_{k_2}(x_{\text{cut}}) - u_{k-1}(x_{\text{cut}}) \right]^2 \right),
\end{aligned} \tag{3.32}$$

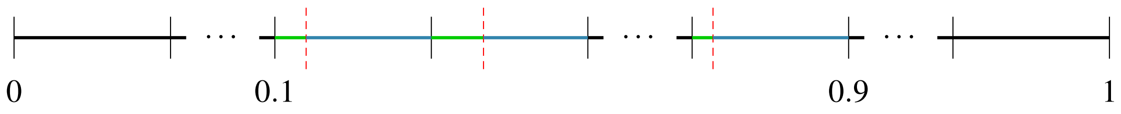


Figure 3.7: Modified model problem **MP** for numerical tests in one dimension: We consider the interval $[0,1]$ and between $x = 0.1$ and $x = 0.9$ we split every cell into a pair of two cut cells, which are marked in green and blue.

where the first two terms in the last line of equation (3.32) are non-positive and, hence, might lead to problems. Fortunately, when considering the sum $a_h(u^h, u^h) + J_h(u^h, u^h)$, we observe that the stabilization has the effect of replacing a specific amount, identified by η_{k_1}, of the 'standard' jumps at both edges $x_{k-\frac{1}{2}}$ and x_{cut} of the small cut cell E_{k_1} by an 'extended' jump between the inflow neighbor E_{k-1} and the outflow neighbor E_{k_2}, evaluated at x_{cut}. This finishes the proof. $\square$

3.1.3. Numerical results

For the numerical results in one dimension, we consider a modified version of the model problem **MP**: We take the closed interval $\overline{\Omega} = [0,1]$ and split every cell E_k between $x = 0.1$ and $x = 0.9$ in cut cell pairs (E_{k_1}, E_{k_2}) of lengths $\alpha_k h$ and $(1-\alpha_k)h$, where $\alpha_k \in (0, \frac{1}{2}]$ may vary for different k. Figure 3.7 illustrates this setup.

Afterwards, we distinguish between two cases:

- Case 1 ('$\alpha = 10^{-\square}$'): The cut cell fraction α_k is the same for all cut cell pairs, i.e. $\alpha_k \equiv \alpha$.

- Case 2 ('rand α'): The cut cell fraction α_k varies and is computed randomly as $\alpha_k = 10^{-2} X_k$ with X_k being a uniformly distributed random number in $(0,1)$.

For the convergence tests, we compare additionally a reference solution on a uniform mesh called ('equi'). On these uniform mesh the stabilization vanishes.

The DoD stabilization for a general cut cell mesh and, in particular, the modified model problem is given by

$$J_h(u^h, w^h) = \sum_{k \in I} J_h^{0,k}(u^h, w^h) + J_h^{1,k}(u^h, w^h). \tag{3.33}$$

The set I contains all cells that need to be stabilized. In our numerical tests, we use the following definition

$$I = \{E_k \mid \eta_k > 0\}$$

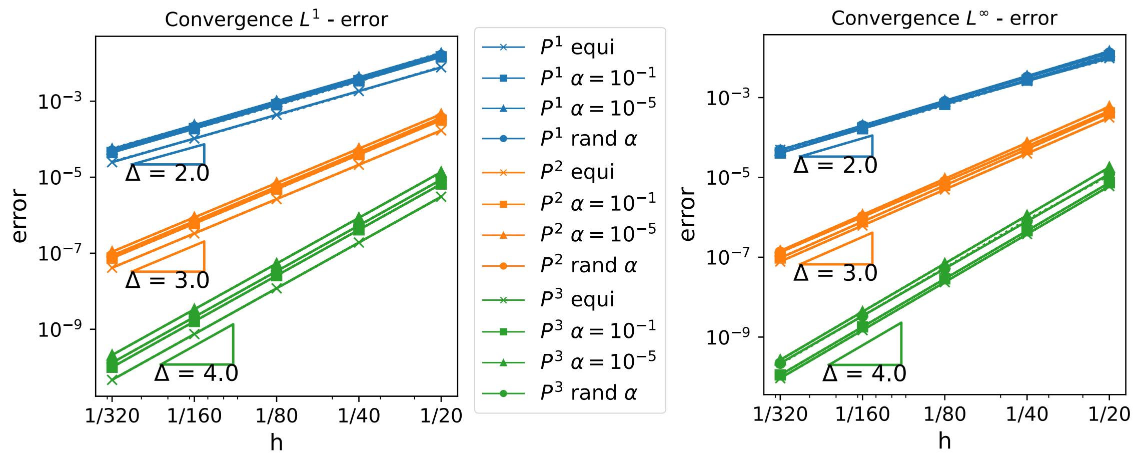


Figure 3.8: Convergence test for a smooth solution for the linear advection equation: Error in the L^1 and L^∞ norm.

with the stabilization parameter η_k defined according to equation (3.11). Note that we define the stabilization parameter on all cells by setting the volume fraction $\alpha = 1$ on Cartesian cells.

Convergence tests

We consider the linear advection equation (2.85) using the constant velocity $\beta = 1$. The smooth initial data $u_0(x) = \sin(2\pi x)$, and periodic boundary conditions, so that the smooth solution reads $u(x,t) = \sin(2\pi(x-t))$ in combination with periodic boundary conditions.

We consider three different cut cell setups: we test moderately *large* values of the cut cell fraction with constant $\alpha = 10^{-1}$, but also relatively small values with $\alpha = 10^{-5}$. In addition, we test the case of random cut cell fractions and compare them to the results on a uniform mesh.

In Figure 3.8, we show the L^1 error and the L^∞ error at time $T = 1$. The L^1 error and the L^∞ error norm are defined as

$$\left\| u(\cdot,T) - u^h(\cdot,T) \right\|_{L^1(\Omega)} = \int_\Omega \left| u(x,T) - u^h(x,T) \right| \mathrm{d}x$$

and

$$\left\| u(\cdot,T) - u^h(\cdot,T) \right\|_{L^\infty(\Omega)} = \max_{x\in\Omega} \left| u(x,T) - u^h(x,T) \right|.$$

For both error norms, we observe in all test scenarios convergence orders $p+1$ for polynomials of order p no matter whether or not cut cells are part of the mesh.

Stability tests

In Section 3.1.2, we presented the eigenvalue distribution for different choices of α and in the limit $\alpha \to 0$ using a grid of $N = 5$ cells and piecewise linear functions for the semi-discrete stabilized scheme. We continue the eigenvalue analysis for our stabilized method numerically for a larger number of grid cells and different polynomial degrees. Instead of the semi-discrete stabilized scheme, we now consider the fully discretized scheme. Since we are considering a linear test problem at the moment, we can rewrite our fully discretized scheme for every polynomial degree p in the matrix-vector form

$$\mathbf{U}^{n+1} = \mathbf{A}\mathbf{U}^{n},$$

where $\mathbf{A}$ is the global system matrix and $\mathbf{U}^n$ is the vector of all degrees of freedom at time t^n. The eigenvalues λ_i of this matrix $\mathbf{A}$ have important consequences for the stability and convergence of the numerical solution. This is because the eigenvalues of the global system matrix determine the behavior of the numerical solution over time [79]:

- $\underline{|\lambda_i| > 1}$: In general, if the magnitude of an eigenvalue λ_i is greater than 1, the corresponding mode of the numerical solution will grow exponentially in time, leading to instability and divergence of the solution.

- $\underline{|\lambda_i| < 1}$: Conversely, if the eigenvalue magnitude is less than 1, the mode will decay exponentially over time, leading to stability and convergence of the solution.

- $\underline{|\lambda_i| = 1}$: For the special case that the magnitude of the eigenvalue is equal to 1, the mode will oscillate without changing amplitude, leading to neutral stability.

By requiring that the eigenvalues of the global system matrix lie within the unit circle, i.e., $|\lambda_i| \leq 1 \; \forall i$, we ensure that all modes of the numerical solution will either decay or oscillate and will not grow in time. This will ensure that the numerical solution is stable. In practice, this condition is often used as a diagnostic tool to assess the stability and convergence of numerical methods for solving PDEs and to identify any sources of instability that may need to be addressed to obtain a stable and accurate numerical solution.

We consider the linear advection equation (2.85) with the velocity $\beta = 1$ for arbitrary initial data with periodic boundary conditions. In Figure 3.9, we show the eigenvalues for the different cut cell grids under investigation and polynomial degrees $p = 0, \ldots, 3$. For all setups, we observe that the eigenvalues stay inside the unit circle, which means that we see numerically $|\lambda| \leq 1$ and the stability of the given method numerically.

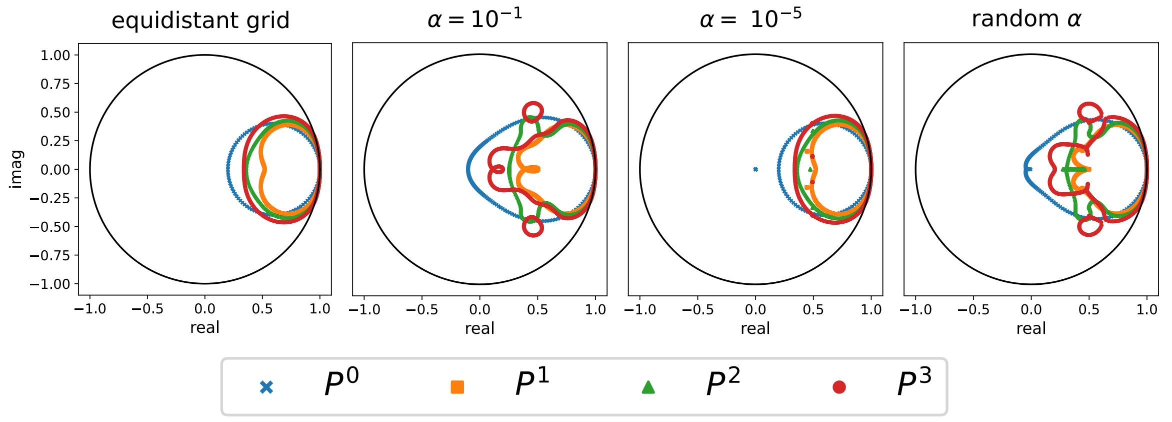


Figure 3.9: Eigenvalue distribution for different ansatz orders $p \in \{0,1,2,3\}$ and the different cut cell grids.

3.2. Extension to non-linear systems of conservation laws

The following part is based on [62]. As a next step, we extend the concept of the DoD stabilization to non-linear scalar equations and systems of hyperbolic conservation laws. These are given by the equations

$$\partial_t \mathbf{u} + \partial_x \mathbf{f}(\mathbf{u}) = \mathbf{0}$$

where $\mathbf{u} : \mathbb{R} \times (0,T) \to \mathbb{R}^m$ is the vector of conserved variables and $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ is the possibly non-linear flux function. Furthermore, the integer $m \geq 1$ denotes the number of equations under investigation. Note that the scalar case $m = 1$ is included in all considerations even though we sometimes use bold typesetting.

3.2.1. Piecewise constant polynomials

The stabilization for non-linear conservation laws in the setting of piecewise constant polynomials for the model problem **MP** is given by

$$J_h(\mathbf{u}^h, \mathbf{w}^h) = J_h^{0,k_1}(\mathbf{u}^h, \mathbf{w}^h)$$

with

$$\begin{aligned} J_h^{0,k_1}(\mathbf{u}^h, \mathbf{w}^h) =& \eta_{k_1} \left[\mathcal{H}(\mathbf{u}_{k-1}, \mathbf{u}_{k_2})(x_{k-\frac{1}{2}}) - \mathcal{H}(\mathbf{u}_{k-1}, \mathbf{u}_{k_1})(x_{k-\frac{1}{2}}) \right] \cdot \left[\!\left[\mathbf{w}^h \right]\!\right]_{k-\frac{1}{2}} \\ &+ \eta_{k_1} \left[\mathcal{H}(\mathbf{u}_{k-1}, \mathbf{u}_{k_2})(x_{\text{cut}}) - \mathcal{H}(\mathbf{u}_{k_1}, \mathbf{u}_{k_2})(x_{\text{cut}}) \right] \cdot \left[\!\left[\mathbf{w}^h \right]\!\right]_{\text{cut}}. \end{aligned} \tag{3.34}$$

Although it is not important at which locations the piecewise constant functions are evaluated, we include them here to emphasize the symmetric structure of the two terms in

$J_h^{0,k_1}(\cdot,\cdot)$. We add jump terms at both edges of E_{k_1}, accounting for the two possible flow directions in the non-linear case. Note that we make use of the extension operator here when $\mathbf{u}_{k-1}$ and $\mathbf{u}_{k_2}$ are evaluated at x_{cut} and $x_{k-\frac{1}{2}}$, respectively. We emphasize that the proposed stabilization (3.34) reduces to the already known stabilization (3.17) when considering a linear problem. Once again, we look at the issues induced by the small cut cell and discuss the effect of the stabilization. The stabilized update formulas in the non-linear case are given by

$$\begin{aligned}
\mathbf{u}_{k-1}^{n+1} &= \mathbf{u}_{k-1}^{n} - \frac{\Delta t}{h}\{(1-\eta_{k_1})\mathcal{H}(\mathbf{u}_{k-1}^{n},\mathbf{u}_{k_1}^{n}) - \mathcal{H}(\mathbf{u}_{k-2}^{n},\mathbf{u}_{k-1}^{n}) + \eta_{k_1}\mathcal{H}(\mathbf{u}_{k-1}^{n},\mathbf{u}_{k_2}^{n})\},\\
\mathbf{u}_{k_1}^{n+1} &= \mathbf{u}_{k_1}^{n} - \frac{\Delta t}{\alpha h}(1-\eta_{k_1})\{\mathcal{H}(\mathbf{u}_{k_1}^{n},\mathbf{u}_{k_2}^{n}) - \mathcal{H}(\mathbf{u}_{k-1}^{n},\mathbf{u}_{k_1}^{n})\},\\
\mathbf{u}_{k_2}^{n+1} &= \mathbf{u}_{k_2}^{n} - \frac{\Delta t}{(1-\alpha)h}\{\mathcal{H}(\mathbf{u}_{k_2}^{n},\mathbf{u}_{k+1}^{n}) - (1-\eta_{k_1})\mathcal{H}(\mathbf{u}_{k_1}^{n},\mathbf{u}_{k_2}^{n}) - \eta_{k_1}\mathcal{H}(\mathbf{u}_{k-1}^{n},\mathbf{u}_{k_2}^{n})\}.
\end{aligned} \tag{3.35}$$

Issue 1: Unstable update on small cut cell

The unstabilized update formula for the small cut cell in the non-linear case reads

$$\mathbf{u}_{k_1}^{n+1} = \mathbf{u}_{k_1}^{n} - \frac{\Delta t}{\alpha h}\{\mathcal{H}(\mathbf{u}_{k_1}^{n},\mathbf{u}_{k_2}^{n}) - \mathcal{H}(\mathbf{u}_{k-1}^{n},\mathbf{u}_{k_1}^{n})\}. \tag{3.36}$$

In the unstabilized case, there is again an $O(\frac{1}{\alpha})$ dependency when updating the small cut cell E_{k_1}. This leads to an unstable update on the small cut cell for a decreasing $\alpha \to 0$. When comparing the unstabilized with the stabilized update formula, we see that they again only differ by the factor $(1-\eta_{k_1})$. Therefore, the stabilization parameter η_{k_1} can be chosen in such a way that the $O(\frac{1}{\alpha})$ dependency cancels out, and the update on the small cut cell becomes stable again. In summary, the stabilization has a similar effect on the first issue as in the linear case above.

Issue 2: Missing information on outflow neighbor

Next, we will look at the second issue, which is caused by the requirement for correct physical information on the outflow neighbors of small cut cells. When considering the stabilized update formulas in the neighborhood of the small cut cell, we notice that both neighbors have been modified. This modification is constructed such that we scale down the flux between the small cut cell and its neighbor and add a greater amount of an additional flux between the two neighbors of the small cut cell. This allows cell E_{k-1} to obtain information from cell E_{k_2} and vice versa.

Monotonicity

As a next step, we extend the monotonicity statement that we have proved for the case of solving a linear scalar equation using piecewise constant polynomials. For this, we will again work with Definition 8. Furthermore, we use the definition $\mathcal{H}_a(\mathbf{u}^-,\mathbf{u}^+) \in \mathbb{R}^{m\times m}$ to denote the Jacobian of the numerical flux $\mathcal{H}(\mathbf{u}^-,\mathbf{u}^+)$ with respect to the first argument, i.e., $(\mathcal{H}_a(\mathbf{u}^-,\mathbf{u}^+))_{i,j=1}^m := \left(\frac{\partial}{\partial(\mathbf{u}^-)_j}\mathcal{H}(\mathbf{u}^-,\mathbf{u}^+)_i\right)_{i,j=1}^m$. Analogously, $\mathcal{H}_b(\mathbf{u}^-,\mathbf{u}^+)$ denotes the Jacobian concerning the second argument $\mathbf{u}^+$.

Theorem 21. *[62, Theorem 5] Consider the stabilized scheme (3.35) using elements of V_h^0 for the model problem **MP** with explicit Euler in time, applied to a scalar conservation law. Let the time step be given by $\Delta t = \frac{\nu h}{\lambda_{\max}}$ for $0 < \alpha < \nu < 1-\alpha$. Let the numerical flux $\mathcal{H}$ satisfy Prerequisite 2.2.1. In addition, we assume that*

$$\left|\mathcal{H}_a(u,v)\right| + \left|\mathcal{H}_b(w,u)\right| \le \frac{\nu h}{\Delta t} \quad \forall u,v,w \tag{3.37}$$

is satisfied. Then, the stabilized scheme is monotone.

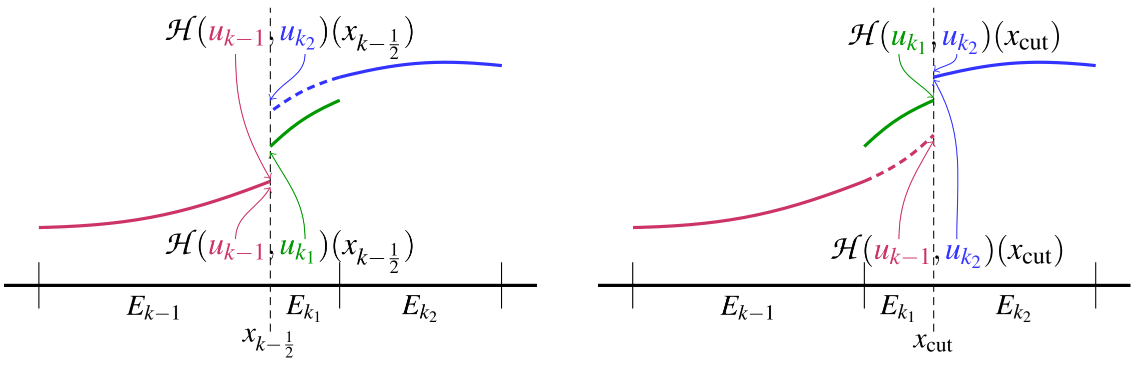
Remark 22. *Condition (3.37) is a common condition for monotonicity on regular meshes, see [65].*

Proof. Away from the two cut cells, we use a standard first-order DG scheme on a uniform mesh, which is monotone under the given assumptions. Therefore, it suffices to show property (2.81) for the three cells $E_j, j \in I_{Neigh}$, that are affected by our stabilization. The update formulas are given by (3.35). Due to the fact that $0 < \eta_{k_1} < 1$, the non-negativity of $\frac{\partial}{\partial u_i^n} u_j^{n+1}$ for $i \neq j$ follows directly from the monotonicity of the fluxes. It remains to investigate the sign of $\frac{\partial}{\partial u_j^n} u_j^{n+1}$ for $j \in I_{Neigh}$. We start with cell E_{k-1}:

$$\begin{aligned}\frac{\partial}{\partial u_{k-1}^n} u_{k-1}^{n+1} &= 1 - \frac{\Delta t}{h}\Big\{(1-\eta_{k_1})\mathcal{H}_a(u_{k-1}^n,u_{k_1}^n) + \eta_{k_1}\mathcal{H}_a(u_{k-1}^n,u_{k_2}^n) \\ &\qquad - (1-\eta_{k_1})\mathcal{H}_b(u_{k-2}^n,u_{k-1}^n) - \eta_{k_1}\mathcal{H}_b(u_{k-2}^n,u_{k-1}^n)\Big\} \\ &\ge 1 - \frac{\Delta t}{h}\left\{(1-\eta_{k_1})\frac{\nu h}{\Delta t} + \eta_{k_1}\frac{\nu h}{\Delta t}\right\} \ge 0.\end{aligned}$$

For the small cut cell E_{k_1} and $\eta_{k_1} = 1 - \frac{\alpha}{\nu}$, we have

$$\begin{aligned}\frac{\partial}{\partial u_{k_1}^n} u_{k_1}^{n+1} &= 1 - \frac{\Delta t}{\alpha h}(1-\eta_{k_1})\{\mathcal{H}_a(u_{k_1}^n,u_{k_2}^n) - \mathcal{H}_b(u_{k-1}^n,u_{k_1}^n)\} \\ &\ge 1 - \frac{\Delta t}{\alpha h}\frac{\alpha}{\nu}\frac{\nu h}{\Delta t} \ge 0.\end{aligned}$$



$\mathcal{H}(u_{k-1}, u_{k_2})(x_{k-\frac{1}{2}})$
$\mathcal{H}(u_{k-1}, u_{k_1})(x_{k-\frac{1}{2}})$
E_{k-1}
E_{k_1}
E_{k_2}
$x_{k-\frac{1}{2}}$
$\mathcal{H}(u_{k_1}, u_{k_2})(x_{\text{cut}})$
$\mathcal{H}(u_{k-1}, u_{k_2})(x_{\text{cut}})$
E_{k-1}
E_{k_1}
E_{k_2}
x_{cut}

Figure 3.10: [62] Sketch of the different jump terms that are evaluated for the stabilization term J_h^{0,k_1}.

Finally, for cell E_{k_2} we obtain

$$\begin{aligned}\frac{\partial}{\partial u_{k_2}^n} u_{k_2}^{n+1} =& 1 - \frac{\Delta t}{(1-\alpha)h}\Big\{(1-\eta_{k_1})\mathcal{H}_a(u_{k_2}^n, u_{k+1}^n) + \eta_{k_1}\mathcal{H}_a(u_{k_2}^n, u_{k+1}^n) \\ & \qquad - (1-\eta_{k_1})\mathcal{H}_b(u_{k_1}^n, u_{k_2}^n) - \eta_{k_1}\mathcal{H}_b(u_{k-1}^n, u_{k_2}^n)\Big\} \\ \geq & 1 - \frac{\Delta t}{(1-\alpha)h}\frac{\nu h}{\Delta t} \geq 0.\end{aligned}$$

This concludes the proof. □

3.2.2. Higher-order polynomials

In this section, we will discuss the extension to higher-order polynomials for systems of non-linear conservation laws. We have already seen in the previous chapter that the DoD stabilization for higher-order polynomials and linear equations consists of two parts with different tasks:

$$J_h(\mathbf{u}^h, \mathbf{w}^h) = J_h^{0,k_1}(\mathbf{u}^h, \mathbf{w}^h) + J_h^{1,k_1}(\mathbf{u}^h, \mathbf{w}^h).$$

The first term $J_h^{0,k_1}(\cdot,\cdot)$ was already defined in the previous section by

$$\begin{aligned}J_h^{0,k_1}(\mathbf{u}^h, \mathbf{w}^h) =& \eta_{k_1}\left[\mathcal{H}(\mathbf{u}_{k-1}, \mathbf{u}_{k_2})(x_{k-\frac{1}{2}}) - \mathcal{H}(\mathbf{u}_{k-1}, \mathbf{u}_{k_1})(x_{k-\frac{1}{2}})\right] \cdot \left[\!\left[\mathbf{w}^h\right]\!\right]_{k-\frac{1}{2}} \\ & + \eta_{k_1}\left[\mathcal{H}(\mathbf{u}_{k-1}, \mathbf{u}_{k_2})(x_{\text{cut}}) - \mathcal{H}(\mathbf{u}_{k_1}, \mathbf{u}_{k_2})(x_{\text{cut}})\right] \cdot \left[\!\left[\mathbf{w}^h\right]\!\right]_{\text{cut}}.\end{aligned}$$

Once again, we emphasize the fact that we use the extension operator to evaluate $\mathbf{u}_{k-1}$ and $\mathbf{u}_{k_2}$ outside of their original domain, see Figure 3.10 for a visualization of the symmetrized non-linear term.

The stabilization term $J_h^{1,k_1}(\cdot,\cdot)$ is given in the form of a penalty term that is evaluated on cell volumes. It controls the mass distribution primarily *within* the small cut cell E_{k_1} and secondarily *within* its neighbors E_{k-1} and E_{k_2}. The stabilization accounts for how much mass has been moved into and out of the small cut cell E_{k_1} from and to its left and right neighbors using $a_h(\cdot,\cdot)$ and $J_h^{0,k_1}(\cdot,\cdot)$. The terms are derived from the proof of the L^2 stability, see Theorem 23 below. This part is based on the findings in [62].

Analogously to the ansatz functions, we also extrapolate the test functions to be used within their direct neighbor but outside of their original support. The stabilization term $J_h^{1,k_1}(\cdot,\cdot)$ is given by

$$\begin{aligned} J_h^{1,k_1}(\mathbf{u}^h,\mathbf{w}^h) =&\eta_{k_1} \sum_{j\in I_{Neigh}} \mathbf{K}(j) \int_{k_1} \left(\mathcal{H}(\mathbf{u}_{k-1},\mathbf{u}_{k_2}) - \mathbf{f}(\mathbf{u}_j)\right)\cdot \partial_x \mathbf{w}_j \,\mathrm{d}x \\ &+\eta_{k_1} \sum_{j\in I_{Neigh}} \mathbf{K}(j) \int_{k_1} \left(\mathcal{H}_a(\mathbf{u}_{k-1},\mathbf{u}_{k_2})\mathbf{u}_j\right)\cdot \partial_x \mathbf{w}_{k-1} \,\mathrm{d}x \\ &+\eta_{k_1} \sum_{j\in I_{Neigh}} \mathbf{K}(j) \int_{k_1} \left(\mathcal{H}_b(\mathbf{u}_{k-1},\mathbf{u}_{k_2})\mathbf{u}_j\right)\cdot \partial_x \mathbf{w}_{k_2} \,\mathrm{d}x. \end{aligned} \tag{3.38}$$

Here, the matrices $\mathbf{K}(j) \in \mathbb{R}^{m\times m}, j \in I_{Neigh}$, incorporate information about the flow directions. They are defined using positive semi-definite matrices $\mathbf{L}_{k_1}, \mathbf{R}_{k_1} \in \mathbb{R}^{m\times m}$ and the identity matrix $\mathbf{I}^m \in \mathbb{R}^{m\times m}$. We set

$$\mathbf{K}(k-1) = \mathbf{L}_{k_1}, \quad \mathbf{K}(k_1) = -\mathbf{I}^m, \quad \text{and} \quad \mathbf{K}(k_2) = \mathbf{R}_{k_1}.$$

The choices of the matrices $\mathbf{L}_{k_1}$ and $\mathbf{R}_{k_1}$ will be discussed in the next paragraph based on the findings presented in [62].

3.2.3. Choice of parameters

The matrices $\mathbf{L}_{k_1}$ and $\mathbf{R}_{k_1}$ incorporate information about the flow direction.

We knew that for the linear advection equation (2.85) with positive velocity $\beta > 0$, we need to set $L_{k_1} = 1$ and $R_{k_1} = 0$, whereas for negative velocity $\beta < 0$, $L_{k_1} = 0$ and $R_{k_1} = 1$ is required. In other words: L_{k_1} and R_{k_1} contain information about the flow direction (but not about the amplitude of the flow, i.e., only the sign of β is important).

We use this knowledge to find proper parameter matrices for the case of a non-linear system of equations. As a first step, we consider the linear system

$$\partial_t \mathbf{u} + \mathbf{A}\partial_x \mathbf{u} = \mathbf{0} \tag{3.39}$$

and decompose the matrix $\mathbf{A}$ to get the flow information of interest. Thanks to the assumption of hyperbolicity, $\mathbf{A}$ is diagonalizable with real eigenvalues $\lambda_i, i = 1,\dots,m$. Therefore, we can rewrite $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{-1}$, where the columns of $\mathbf{Q}$ contain the right eigenvectors of $\mathbf{A}$ and Λ is a diagonal matrix containing the eigenvalues $(\lambda_i)_i$. One can reinterpret the resulting linear system as m independent linear advection equations with velocities $\lambda_i = \Lambda_{ii}$. Based on the sign of λ_i, we can now determine the flow directions and define the diagonal entries of matrices $\Lambda^+, \Lambda^- \in \mathbb{R}^{m\times m}$ as follows:

$$\Lambda^+_{ii} = \begin{cases} 1 & \text{if } \Lambda_{ii} > 0, \\ \frac{1}{2} & \text{if } \Lambda_{ii} = 0, \\ 0 & \text{if } \Lambda_{ii} < 0, \end{cases} \quad \text{and} \quad \Lambda^-_{ii} = \begin{cases} 0 & \text{if } \Lambda_{ii} > 0, \\ \frac{1}{2} & \text{if } \Lambda_{ii} = 0, \\ 1 & \text{if } \Lambda_{ii} < 0. \end{cases}$$

Then, we define $\mathbf{L}_{k_1}$ and $\mathbf{R}_{k_1}$ by transforming the diagonal matrices back using the matrices $\mathbf{Q}$ and $\mathbf{Q}^{-1}$:

$$\mathbf{L}_{k_1} = \mathbf{Q}\Lambda^+\mathbf{Q}^{-1} \quad \text{and} \quad \mathbf{R}_{k_1} = \mathbf{Q}\Lambda^-\mathbf{Q}^{-1}. \tag{3.40}$$

Note that $\mathbf{L}_{k_1}$ and $\mathbf{R}_{k_1}$ are positive semi-definite matrices and satisfy $\mathbf{L}_{k_1} + \mathbf{R}_{k_1} = \mathbf{I}^m$.

We can now transfer the linear system case to *non-linear* problems. For this we use the same approach but replace $\mathbf{A}$ by the (non-linear) Jacobian matrix $\mathbf{f_u}(\mathbf{u})$, evaluated at a suitable average $\hat{\mathbf{u}}$ of $\mathbf{u}_{k-1}(x_{k_1})$ and $\mathbf{u}_{k_2}(x_{k_1})$, with x_{k_1} denoting the cell centroid of cell E_{k_1}. For scalar problems, i.e., $m = 1$, we simply use the arithmetic average $\hat{u} = (u_{k-1}(x_{k_1}) + u_{k_2}(x_{k_1}))/2$ and set

$$L_{k_1} = I^+ = \begin{cases} 1 & \text{if } \partial_u f(\hat{u}) > 0, \\ \frac{1}{2} & \text{if } \partial_u f(\hat{u}) = 0, \\ 0 & \text{if } \partial_u f(\hat{u}) < 0, \end{cases} \quad \text{and} \quad R_{k_1} = I^- = \begin{cases} 0 & \text{if } \partial_u f(\hat{u}) > 0, \\ \frac{1}{2} & \text{if } \partial_u f(\hat{u}) = 0, \\ 1 & \text{if } \partial_u f(\hat{u}) < 0. \end{cases}$$

For solving the compressible Euler equations, we use the Roe average [70]

$$\hat{\mathbf{u}}(\mathbf{u}_{k-1}, \mathbf{u}_{k_2}) = \frac{1}{2}\begin{pmatrix} \sqrt{\rho_{k-1}} + \sqrt{\rho_{k_2}} \\ \sqrt{\rho_{k-1}}v_{k-1} + \sqrt{\rho_{k_2}}v_{k_2} \\ \sqrt{\rho_{k-1}}H_{k-1} + \sqrt{\rho_{k_2}}H_{k_2} \end{pmatrix}$$

with $H = \frac{E+p}{\rho}$, which are evaluated at x_{k_1}, and with dropping the evaluation point x_{k_1} for brevity. Then, we decompose $\mathbf{f_u}(\hat{\mathbf{u}})$ into $\mathbf{Q}\Lambda\mathbf{Q}^{-1}$ and use again definition (3.40).

L^2 stability

In this section, we prove for non-linear scalar equations that the stabilized semi-discrete scheme

$$(d_t u^h(t), w^h)_{L^2(\Omega)} + a_h(u^h(t), w^h) + J_h(u^h(t), w^h) = 0 \tag{3.41}$$

is L^2 stable for arbitrary polynomial degrees p and the model problem **MP**. This section is based on the work [62]. We note that the *un*stabilized semi-discrete scheme

$$(d_t u^h(t), w^h)_{L^2(\Omega)} + a_h(u^h(t), w^h) = 0 \tag{3.42}$$

is also L^2 stable in this setting, as shown in the proof below. However, when combined with an explicit time-stepping scheme, one would need to take tiny time steps to ensure stability for the fully discrete scheme. This is not the case for our stabilized scheme. The difficulty in designing the stabilization term $J_h(\cdot,\cdot)$ is to find a formulation that is both L^2 stable for the semi-discrete setting and solves the small cell problem for the fully discrete setting in a monotone way.

Theorem 23. *[62, Theorem 6] Let $u^h(t)$, with $u^h(t) \in V_h^p$ for any fixed t, be the solution to the semi-discrete problem (3.41) for the scalar equation (2.3) with periodic boundary conditions. Let the numerical flux function $\mathcal{H}(\cdot,\cdot)$ satisfy Prerequisite 2.2.1. Further, let the evaluation of the volume terms in $a_h(\cdot,\cdot)$ and $J_h(\cdot,\cdot)$ be exact. Then, the solution satisfies for all $t \in (0,T)$*

$$\left\|u^h(t)\right\|_{L^2(\Omega)} \le \left\|u^h(0)\right\|_{L^2(\Omega)}.$$

Proof. We choose $w^h = u^h(t)$ in (3.41) to get

$$(d_t u^h(t), u^h(t))_{L^2(\Omega)} + a_h(u^h(t), u^h(t)) + J_h(u^h(t), u^h(t)) = 0.$$

Integration in time for the first term results in

$$\int_0^t (d_\tau u^h(\tau), u^h(\tau))_{L^2(\Omega)}\,\mathrm{d}\tau = \int_0^t \frac{\mathrm{d}}{\mathrm{d}\tau}\frac{1}{2}\left\|u^h(\tau)\right\|^2_{L^2(\Omega)}\mathrm{d}\tau = \frac{1}{2}\left\|u^h(t)\right\|^2_{L^2(\Omega)} - \frac{1}{2}\left\|u^h(0)\right\|^2_{L^2(\Omega)}.$$

It remains to show that for any fixed t

$$a_h(u^h(t), u^h(t)) + J_h(u^h(t), u^h(t)) \ge 0.$$

For the remainder of the proof, we will omit the explicit time dependence of $u^h(t)$ for brevity.

Unstabilized case: We first prove L^2 stability for the unstabilized case, i.e., we show $a_h(u^h, u^h) \ge 0$. Here, we follow Jiang and Shu [45] for the particular case of the square entropy function. We define

$$g(u) = \int^u f(\hat{u})\mathrm{d}\hat{u}.$$

This implies $g'(u) = f(u)$. By the E-flux property (2.68) and the mean value theorem, we have

$$\mathcal{H}(u^-, u^+)(u^- - u^+) - (g(u^-) - g(u^+)) \ge 0. \tag{3.43}$$

Further, there holds for an arbitrary cell E_i and an arbitrary u_j

$$\int_{E_i} f(u_j)\,\partial_x u_j \mathrm{d}x = g(u_j(x_{i+\frac{1}{2}})) - g(u_j(x_{i-\frac{1}{2}})).$$

We define the flux

$$F_{i+\frac{1}{2}}(u^h) = \mathcal{H}(u_i,u_{i+1})(x_{i+\frac{1}{2}})\, u_i(x_{i+\frac{1}{2}}) - g(u_i(x_{i+\frac{1}{2}})).$$

Then we can rewrite the contribution of the bilinear form $a^h(\cdot,\cdot)$ for a single, arbitrary cell E_i as

$$\begin{aligned}
&-\int_{E_i} f(u_i(x))\partial_x u_i(x)\mathrm{d}x + \mathcal{H}(u_i,u_{i+1})(x_{i+\frac{1}{2}})\, u_i(x_{i+\frac{1}{2}}) - \mathcal{H}(u_{i-1},u_i)(x_{i-\frac{1}{2}})\, u_i(x_{i-\frac{1}{2}})\\
=&-g(u_i(x_{i+\frac{1}{2}})) + g(u_i(x_{i-\frac{1}{2}})) + \mathcal{H}(u_i,u_{i+1})(x_{i+\frac{1}{2}})\, u_i(x_{i+\frac{1}{2}}) - \mathcal{H}(u_{i-1},u_i)(x_{i-\frac{1}{2}})\, u_i(x_{i-\frac{1}{2}})\\
=&F_{i+\frac{1}{2}}(u) + g(u_i(x_{i-\frac{1}{2}})) - \mathcal{H}(u_{i-1},u_i)(x_{i-\frac{1}{2}})\, u_i(x_{i-\frac{1}{2}})\\
=&F_{i+\frac{1}{2}}(u) - F_{i-\frac{1}{2}}(u) - g(u_{i-1}(x_{i-\frac{1}{2}})) + g(u_i(x_{i-\frac{1}{2}})) + \mathcal{H}(u_{i-1},u_i)(x_{i-\frac{1}{2}}) \left[\!\left[u^h \right]\!\right]_{i-\frac{1}{2}}.
\end{aligned}$$

Using the notation $[\![g(u)]\!]_{i+\frac{1}{2}} = g(u_i(x_{i+\frac{1}{2}})) - g(u_{i+1}(x_{i+\frac{1}{2}}))$, we can summarize

$$\begin{aligned}
a_h(u^h,u^h) = &\sum_{j\in I_{\text{equi}}} \left(F_{j+\frac{1}{2}}(u^h) - F_{j-\frac{1}{2}}(u^h) + \mathcal{H}(u_{j-1},u_j)(x_{j-\frac{1}{2}}) \left[\!\left[u^h \right]\!\right]_{j-\frac{1}{2}} - [\![g(u)]\!]_{j-\frac{1}{2}} \right)\\
&+ \left(F_{\text{cut}}(u^h) - F_{k-\frac{1}{2}}(u^h) + \mathcal{H}(u_{k-1},u_{k_1})(x_{k-\frac{1}{2}}) \left[\!\left[u^h \right]\!\right]_{k-\frac{1}{2}} - [\![g(u)]\!]_{k-\frac{1}{2}} \right)\\
&+ \left(F_{k+\frac{1}{2}}(u^h) - F_{\text{cut}}(u^h) + \mathcal{H}(u_{k_1},u_{k_2})(x_{\text{cut}}) \left[\!\left[u^h \right]\!\right]_{\text{cut}} - [\![g(u)]\!]_{\text{cut}} \right).
\end{aligned}$$

Due to the fluxes F building a telescoping sum and the use of periodic boundary conditions, this implies

$$a_h(u^h,u^h) = \mathbb{T}_1 + \mathbb{T}_2$$

where

$$\begin{aligned}
\mathbb{T}_1 =& \sum_{j\in I_{\text{equi}}} \left(\mathcal{H}(u_{j-1},u_j)(x_{j-\frac{1}{2}}) \left[\!\left[u^h \right]\!\right]_{j-\frac{1}{2}} - [\![g(u)]\!]_{j-\frac{1}{2}} \right),\\
\mathbb{T}_2 =& \mathcal{H}(u_{k-1},u_{k_1})(x_{k-\frac{1}{2}}) \left[\!\left[u^h \right]\!\right]_{k-\frac{1}{2}} - [\![g(u)]\!]_{k-\frac{1}{2}} + \mathcal{H}(u_{k_1},u_{k_2})(x_{\text{cut}}) \left[\!\left[u^h \right]\!\right]_{\text{cut}} - [\![g(u)]\!]_{\text{cut}}.
\end{aligned}$$

are non-negative due to (3.43).

Contribution of stabilization: Now we consider the stabilization and show $a_h(u^h,u^h) + J_h(u^h,u^h) \geq 0$ instead of $J_h(u^h,u^h) \geq 0$. For the edge stabilization, we achieve

$$\begin{aligned}
\frac{1}{\eta_{k_1}} J_h^{0,k_1}(u^h,u^h) &= \Big[\mathcal{H}(u_{k-1},u_{k_2})(x_{k-\frac{1}{2}}) - \mathcal{H}(u_{k-1},u_{k_1})(x_{k-\frac{1}{2}})\Big] \left[\!\left[u^h\right]\!\right]_{k-\frac{1}{2}} \\
&\quad + \left[\mathcal{H}(u_{k-1},u_{k_2})(x_{\text{cut}}) - \mathcal{H}(u_{k_1},u_{k_2})(x_{\text{cut}})\right] \left[\!\left[u^h\right]\!\right]_{\text{cut}} \\
&= -\mathbb{T}_2 + \mathbb{T}_3
\end{aligned}$$

with

$$\mathbb{T}_3 = \mathcal{H}(u_{k-1},u_{k_2})(x_{k-\frac{1}{2}}) \left[\!\left[u^h\right]\!\right]_{k-\frac{1}{2}} - [\![g(u)]\!]_{k-\frac{1}{2}} + \mathcal{H}(u_{k-1},u_{k_2})(x_{\text{cut}}) \left[\!\left[u^h\right]\!\right]_{\text{cut}} - [\![g(u)]\!]_{\text{cut}}.$$

Since $\eta_{k_1} \in (0,1)$, we can later take care of the negative term $-\eta_{k_1}\mathbb{T}_2$ by adding the function $a_h(\cdot,\cdot)$ to get

$$a_h(u^h,u^h) - \eta_{k_1}\mathbb{T}_2 = \mathbb{T}_1 + (1-\eta_{k_1})\mathbb{T}_2 \geq 0.$$

It remains to examine $\mathbb{T}_3$ and the volume stabilization term J_h^{1,k_1}. Here, we make use of the assumption of the flux $\mathcal{H}$ being differentiable, a.e. to write

$$\frac{\mathrm{d}}{\mathrm{d}x}\mathcal{H}(u_{k-1},u_{k_2}) = \mathcal{H}_a(u_{k-1},u_{k_2})\partial_x u_{k-1} + \mathcal{H}_b(u_{k-1},u_{k_2})\partial_x u_{k_2}.$$

This implies

$$\begin{aligned}
\frac{1}{\eta_{k_1}} J_h^{1,k_1}(u^h,u^h) &= \sum_{j\in I_{Neigh}} K(j) \int_{k_1} \big(\mathcal{H}(u_{k-1},u_{k_2}) - f(u_j)\big)\,\partial_x u_j \mathrm{d}x \\
&\quad + \sum_{j\in I_{Neigh}} K(j) \int_{k_1} \mathcal{H}_a(u_{k-1},u_{k_2})\, u_j\, \partial_x u_{k-1}\mathrm{d}x + \sum_{j\in I_{Neigh}} K(j) \int_{k_1} \mathcal{H}_b(u_{k-1},u_{k_2})\, u_j\, \partial_x u_{k_2}\mathrm{d}x \\
&= \sum_{j\in I_{Neigh}} K(j) \int_{k_1} \mathcal{H}(u_{k-1},u_{k_2})\partial_x u_j \mathrm{d}x - \sum_{j\in I_{Neigh}} K(j) \Big(g(u_j(x_{\text{cut}})) - g(u_j(x_{k-\frac{1}{2}}))\Big) \\
&\quad + \sum_{j\in I_{Neigh}} K(j) \int_{k_1} \left(\frac{\mathrm{d}}{\mathrm{d}x}\mathcal{H}(u_{k-1},u_{k_2})\right) u_j\,\mathrm{d}x.
\end{aligned}$$

Using $\frac{\mathrm{d}}{\mathrm{d}x}\big(\mathcal{H}(u_{k-1},u_{k_2})u_j\big) = \mathcal{H}(u_{k-1},u_{k_2})\partial_x u_j + \frac{\mathrm{d}}{\mathrm{d}x}\mathcal{H}(u_{k-1},u_{k_2})u_j$, this results in

$$\begin{aligned}
\frac{1}{\eta_{k_1}} J_h^{1,k_1}(u^h,u^h) = \sum_{j\in I_{Neigh}} K(j)\big[&\big(\mathcal{H}(u_{k-1},u_{k_2})u_j\big)(x_{\text{cut}}) - \big(\mathcal{H}(u_{k-1},u_{k_2})u_j\big)(x_{k-\frac{1}{2}}) \\
&- g(u_j(x_{\text{cut}})) + g(u_j(x_{k-\frac{1}{2}}))\big].
\end{aligned}$$

Recall that

$$K(k-1) = L_{k_1}, \quad K(k_1) = -1, \quad K(k_2) = R_{k_1}$$

with $L_{k_1}, R_{k_1} \in [0,1]$ and $L_{k_1} + R_{k_1} = 1$. Then, skipping some tedious computations for brevity, we obtain

$$\begin{aligned}
\frac{1}{\eta_{k_1}} J_h^{1,k_1}(u^h,u^h) + \mathbb{T}_3 =& \mathcal{H}(u_{k-1},u_{k_2})(x_{k-\frac{1}{2}})u_{k-1}(x_{k-\frac{1}{2}}) - \mathcal{H}(u_{k-1},u_{k_2})(x_{\text{cut}})u_{k_2}(x_{\text{cut}}) \\
& - g(u_{k-1}(x_{k-\frac{1}{2}})) + g(u_{k_2}(x_{\text{cut}})) \\
& + L_{k_1} \Big[\mathcal{H}(u_{k-1},u_{k_2})(x_{\text{cut}})u_{k-1}(x_{\text{cut}}) - \mathcal{H}(u_{k-1},u_{k_2})(x_{k-\frac{1}{2}})u_{k-1}(x_{k-\frac{1}{2}}) \\
& - g(u_{k-1}(x_{\text{cut}})) + g(u_{k-1}(x_{k-\frac{1}{2}})) \Big] \\
& + R_{k_1} \Big[\mathcal{H}(u_{k-1},u_{k_2})(x_{\text{cut}})u_{k_2}(x_{\text{cut}}) - \mathcal{H}(u_{k-1},u_{k_2})(x_{k-\frac{1}{2}})u_{k_2}(x_{k-\frac{1}{2}}) \\
& - g(u_{k_2}(x_{\text{cut}})) + g(u_{k_2}(x_{k-\frac{1}{2}})) \Big] \\
=& \mathbb{T}_4 + \mathbb{T}_5
\end{aligned}$$

with

$$\begin{aligned}
\mathbb{T}_4 &= L_{k_1} \left[\mathcal{H}(u_{k-1},u_{k_2})(x_{\text{cut}}) \left(u_{k-1}(x_{\text{cut}}) - u_{k_2}(x_{\text{cut}}) \right) - g(u_{k-1}(x_{\text{cut}})) + g(u_{k_2}(x_{\text{cut}})) \right] \\
\mathbb{T}_5 &= R_{k_1} \left[\mathcal{H}(u_{k-1},u_{k_2})(x_{k-\frac{1}{2}}) \left(u_{k-1}(x_{k-\frac{1}{2}}) - u_{k_2}(x_{k-\frac{1}{2}}) \right) - g(u_{k-1}(x_{k-\frac{1}{2}})) + g(u_{k_2}(x_{k-\frac{1}{2}})) \right].
\end{aligned}$$

Note that we used $L_{k_1} + R_{k_1} = 1$ here. Again, $\mathbb{T}_4, \mathbb{T}_5 \geq 0$ due to (3.43). In total, the stabilization reads as

$$J_h^{0,k_1}(u^h,u^h) + J_h^{1,k_1}(u^h,u^h) = -\eta_{k_1}\mathbb{T}_2 + \eta_{k_1}\mathbb{T}_4 + \eta_{k_1}\mathbb{T}_5.$$

Together with the bilinear form a_h, this gives

$$a_h(u^h,u^h) + J_h(u^h,u^h) = \mathbb{T}_1 + (1-\eta_{k_1})\mathbb{T}_2 + \eta_{k_1}\mathbb{T}_4 + \eta_{k_1}\mathbb{T}_5.$$

As $\mathbb{T}_1, \mathbb{T}_2, \mathbb{T}_4, \mathbb{T}_5 \geq 0$ and all prefactors are non-negative due to $0 < \eta_{k_1} < 1$, this concludes the proof. □

3.2.4. Numerical results

In this section, we present numerical results for non-linear scalar conservation laws and non-linear systems of conservation laws. We will show results for piecewise constant polynomials in space as well as for higher-order polynomials to illustrate accuracy and stability of the proposed scheme.

For this purpose, we again use the modified model problem shown in Figure 3.7, which is given by the interval $\Omega = [0,1]$ with cut cell pairs between $x = 0.1$ and $x = 0.9$. These cut cell pairs might have different cut cell fractions α_k.

First of all, we test convergence properties for smooth solutions, which are non-trivial to construct, e.g. for the compressible Euler equations. Therefore, we use the concept of *manufactured solutions* for this purpose: We define a smooth function $\mathbf{u}(x,t)$ acting as the solution of our system. Then we insert $\mathbf{u}(x,t)$ in the corresponding equations of the system and compute the right-hand side $\mathbf{s}$ in such a way that the PDE is satisfied. This typically results in a non-zero source term $\mathbf{s}$ and instead of solving

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = \mathbf{0} \quad \text{in } \Omega \times (0,T), \tag{3.44}$$

we now solve the system

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = \mathbf{s} \quad \text{in } \Omega \times (0,T). \tag{3.45}$$

The semi-discrete problem is then given by: Find $\mathbf{u}^h \in V_h^p$ such that

$$\left(d_t \mathbf{u}^h(t), \mathbf{w}^h\right)_{L^2(\Omega)} + a_h\left(\mathbf{u}^h(t), \mathbf{w}^h\right) + J_h\left(\mathbf{u}^h(t), \mathbf{w}^h\right) = \mathcal{S}_h\left(\mathbf{s}, \mathbf{w}^h\right) \quad \forall \mathbf{w}^h \in V_h^p,$$

where

$$\mathcal{S}_h(\mathbf{s}, \mathbf{w}^h) = \sum_{j \in I_{\text{all}}} \int_j \mathbf{s} \cdot \mathbf{w}^h \mathrm{d}x.$$

Remark 24 (Limiter revisited)**.** *As in case of the linear stabilization, we need to evaluate the solution on the cells in the neighborhood of the small cut cell outside of their original support. For the one-dimensional model problem, this includes the cells E_{k-1} and E_{k_2}. Therefore, the solution has to satisfy the following local maximum principle in the neighborhood of the small cut cell:*

$$\begin{aligned}
\min\left(\overline{u}_{k-1}^n, \overline{u}_{k_1}^n, \overline{u}_{k_2}^n\right) &\le u_{k-1}(x_{cut}) \le \max\left(\overline{u}_{k-1}^n, \overline{u}_{k_1}^n, \overline{u}_{k_2}^n\right), \\
\min\left(\overline{u}_{k-1}^n, \overline{u}_{k_1}^n, \overline{u}_{k_2}^n\right) &\le u_{k_2}(x_{k-\frac{1}{2}}) \le \max\left(\overline{u}_{k-1}^n, \overline{u}_{k_1}^n, \overline{u}_{k_2}^n\right).
\end{aligned}$$

The validity of these inequality constraints is enforced using the limiting technique as mentioned in Section 2.2.6. We note that these changes in the limiter induce additional diffusion to a limited solution. However, this work focuses on the development of the stability term $J_h(\cdot,\cdot)$ and not on the limiting technique, which is a very challenging task in this setting. It combines the problems of not limiting higher-order polynomials at smooth extrema and difficulties caused by the cut cell geometry [60]. Finally, this additional restriction on the solution in the neighborhood of E_{k_1} scales with α and is, therefore, negligible for small cut cells.

Burgers equation

We start with two tests for the inviscid Burgers equation. In both cases, the initial solution is given by a sine curve. While the solution stays smooth in the first test by adding a suitable source term, shock and rarefaction waves develop in the second test,

Convergence test

We consider the manufactured solution

$$u(x,t) = \sin(4\pi(x-t))$$

with periodic boundary conditions and the source term

$$s(x,t) = 4\pi \cos(4\pi(x-t))\,(\sin(4\pi(x-t))-1)\,.$$

In Figure 3.11, we show the error, measured in the L^1 and in the L^∞ norm, for different

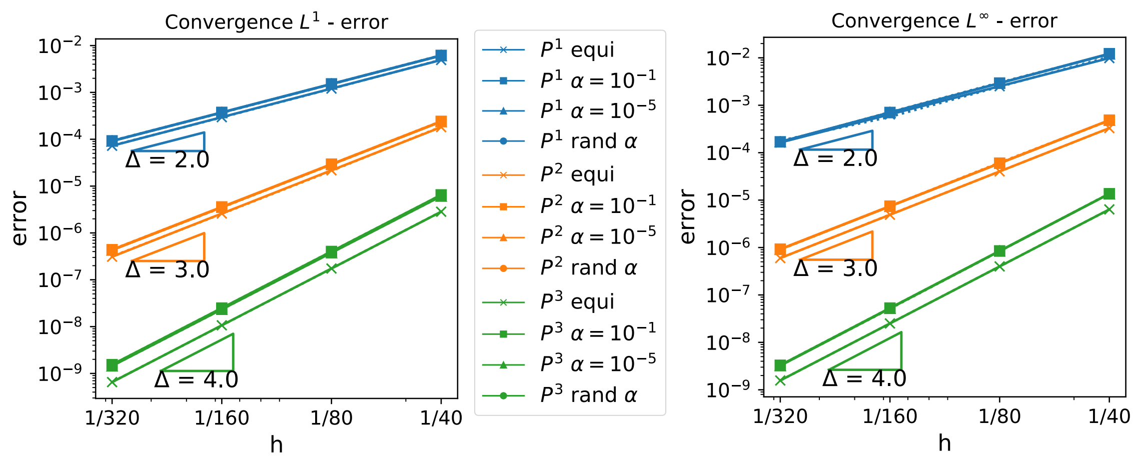


Figure 3.11: [62] Convergence test for manufactured solution for Burgers equation: Error in the L^1 and L^∞ norm.

values of the volume fractions α_k and different polynomial degrees at the final time $T = 1$. We test the same setups as for the linear advection equation in the previous chapter, including large cut cells ('$\alpha = 10^{-1}$'), small cut cells ('$\alpha = 10^{-5}$') and random cut cell sizes ('rand α'). Finally, we compare it to the case of a Cartesian grid ('equi').

As expected, we observe convergence of order $p+1$ for polynomial degree p for both the L^1 and the L^∞ norm. We also note that the error sizes for the different test cases involving varying values of α_k are close to the case without any cut cells.

Stability test

In the next step, we test the stability of the numerical solution on cut cells. In this investigation, we pay special attention to shock and rarefaction waves passing through a small cut cell. Therefore, we consider a non-smooth problem and choose the initial data

$$u_0(x) = \sin(4\pi(x+0.5))$$

with periodic boundary conditions and set the source term $s = 0$. As it is well-known, these initial data result in the development of shock waves in regions where the derivative of u_0 is negative.

Figure 3.12 shows the solution at the final time $T = 0.1$ for different polynomial degrees and α_k being chosen randomly as specified above. The cut cell mesh was created from a mesh with $N = 100$ equidistant cells, and hence, contains 180 cells. For piecewise constant polynomials, the computed solution does not create overshoot, according to the monotonicity result in Theorem 21. We also show the solution for V_h^3, with and without the use of the limiter. Without the limiter, the solution produces an overshoot near the shock. Nevertheless, the numerical tests are stable as in the case of a regular mesh and do not break, despite using small cut cells. With a limiter, the overshoot disappears.

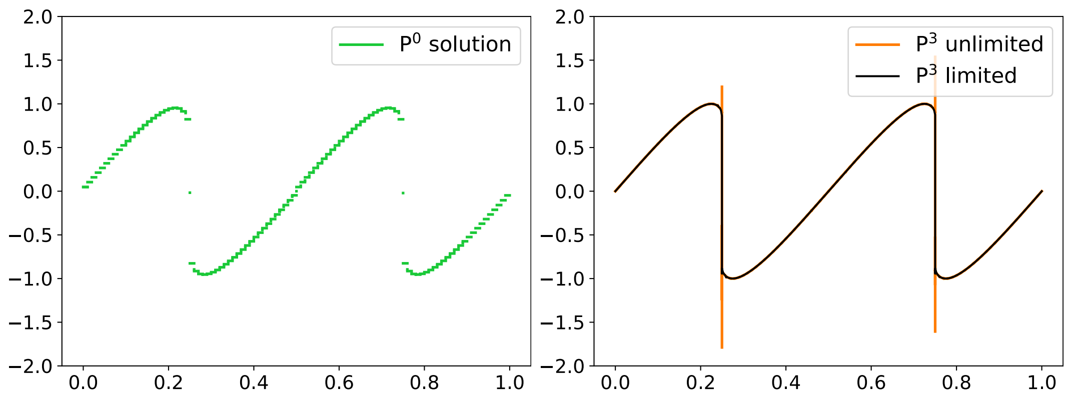


Figure 3.12: [62] Stability test for Burgers equation: Solution at the final time for piecewise constant polynomials (left) and piecewise cubic polynomials with and without a limiter (right).

Euler equations

For the Euler equations, we consider two tests: a test with a smooth manufactured solution and the Sod shock tube test.

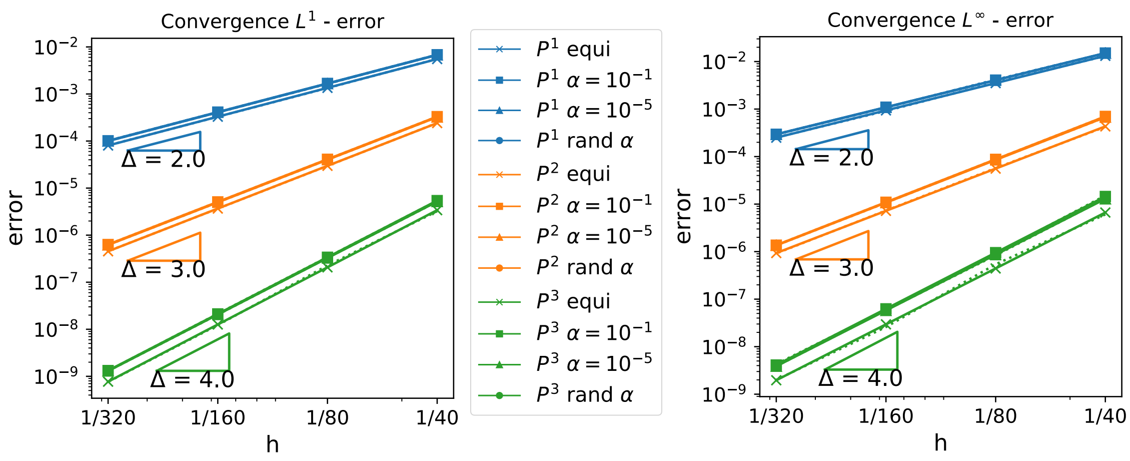


Figure 3.13: [62] Convergence test for manufactured solutions for Euler equations: Error in the L^1 and L^∞ norm.

Convergence test

We define the solution (in terms of primitive variables) by

$$\begin{pmatrix} \rho \\ v \\ p \end{pmatrix} = \begin{pmatrix} 2+\sin(2\pi(x-t)) \\ \sin(2\pi(x-t)) \\ 2+\cos(2\pi(x-t)) \end{pmatrix}$$

together with periodic boundary conditions. The source term $\mathbf{s}(x,t)$ can be calculated by inserting the vector of conserved variables $\mathbf{u}(x,t)$ into equation (3.45) (but it is not given here due to its length).

In Figure 3.13, we show the L^1 and the L^∞ error for the same test cases as in the scalar setting at time $T = 1$. In the general case of a system of m partial differential equations, we compute the L^1 and L^∞ error as

$$\|\mathbf{u}(\cdot,T)\|_1 = \sum_{l=1}^{m} \|\mathbf{u}_l(\cdot,T)\|_{L^1(\Omega)}, \quad \|\mathbf{u}(\cdot,T)\|_\infty = \max_{1\le l\le m} \|\mathbf{u}_l(\cdot,T)\|_{L^\infty(\Omega)}.$$

Again, we see the optimal order of convergence in the L^1 and the L^∞ norm for the different polynomial degrees.

Stability test

We conclude the numerical results with the well-known Sod shock tube test [78]. The following Riemann problem gives the initial data

$$(\rho, \rho v, E) = \begin{cases} (1, 0, 2.5) & \text{if } x < 0, \\ (0.125, 0, 0.25) & \text{else.} \end{cases}$$

We choose $\Omega = (-1, 1)$ for this test and use transmissive boundary conditions. We discretize Ω with $N = 100$ equidistant cells and split every cell in $[-0.9, 0.9]$ into a pair of two cut cells with the volume fraction α_k chosen randomly as described above. We set $T = 0.4$.

In Figure 3.14, we show the solution for the density and the velocity at the final time using piecewise constant polynomials. As expected for V_h^0, the solution is stable but is quite diffusive. Especially within the rarefaction wave for the velocity v, one can see that the solution values on small cut cells lie nicely between the values of their larger neighbors. (We have chosen a thick line width to make the solution visible; as a result, the small cut cells seem larger than they are.) Figure 3.15 shows the solution for piecewise linear, limited polynomials. We applied the limiter described in the section above to the components of the conserved variables and added a check to ensure that the pressure remains positive. Compared to the results for V_h^0, the results are significantly less diffusive and mostly free of oscillations.

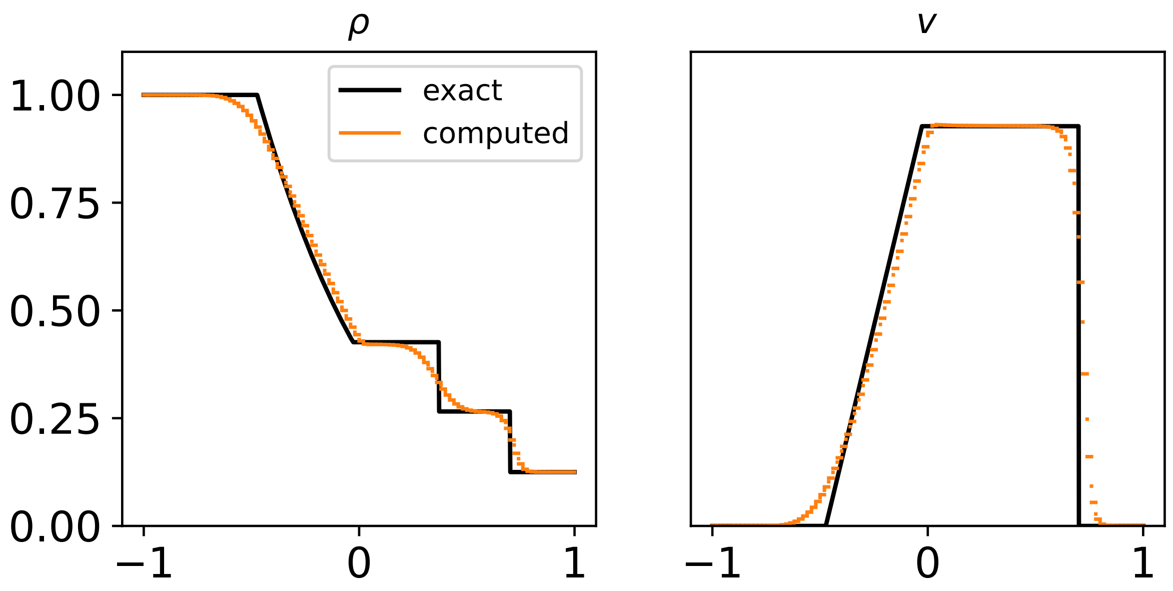


Figure 3.14: [62] Sod shock tube test: Numerical solution for density ρ and velocity v at final time using piecewise constant polynomials.

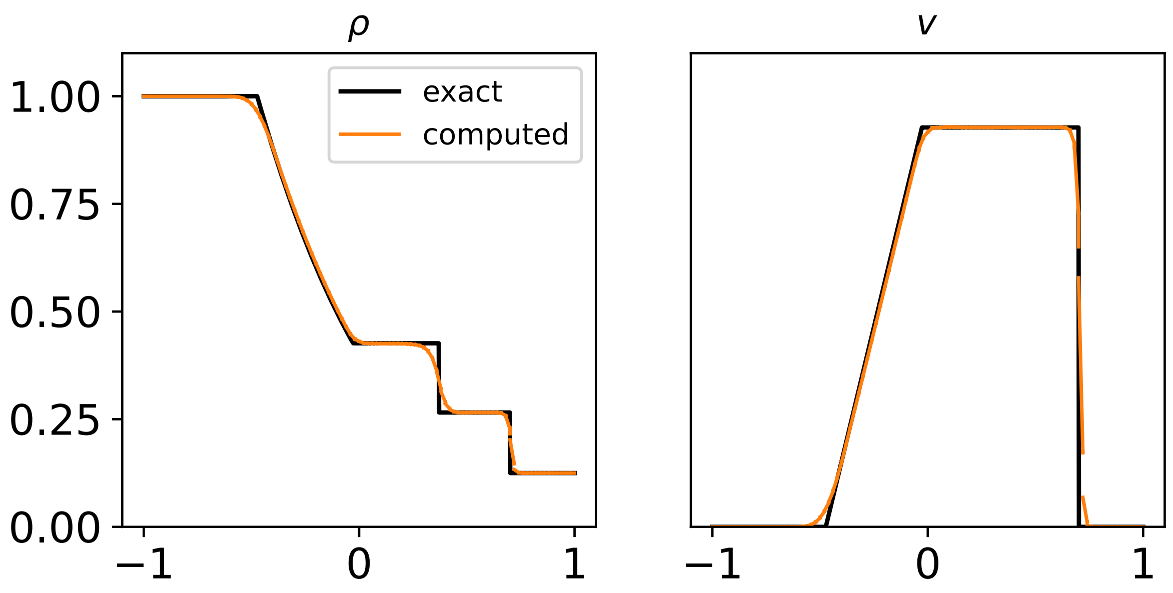


Figure 3.15: [62] Sod shock tube test: Numerical solution for density ρ and velocity v at the final time using piecewise linear polynomials with a limiter.

4

DoD Stabilization in two dimensions

In the previous chapter, we introduced the DoD stabilization in one dimension. We now extend the concept from one dimension to a two-dimensional setting, keeping the main idea behind the stabilization the same. We add a penalty stabilization to the semi-discrete scheme and obtain: Find $\mathbf{u}^h(t) \in V_h^p$ such that

$$\left(d_t \mathbf{u}^h(t), \mathbf{w}^h\right)_{L^2} + a_h\left(\mathbf{u}^h(t), \mathbf{w}^h\right) + J_h\left(\mathbf{u}^h(t), \mathbf{w}^h\right) = 0, \quad \forall \mathbf{w}^h \in V_h^p. \tag{4.1}$$

The stabilization term $J_h(\cdot,\cdot)$ keeps its general structure and is given by the two terms

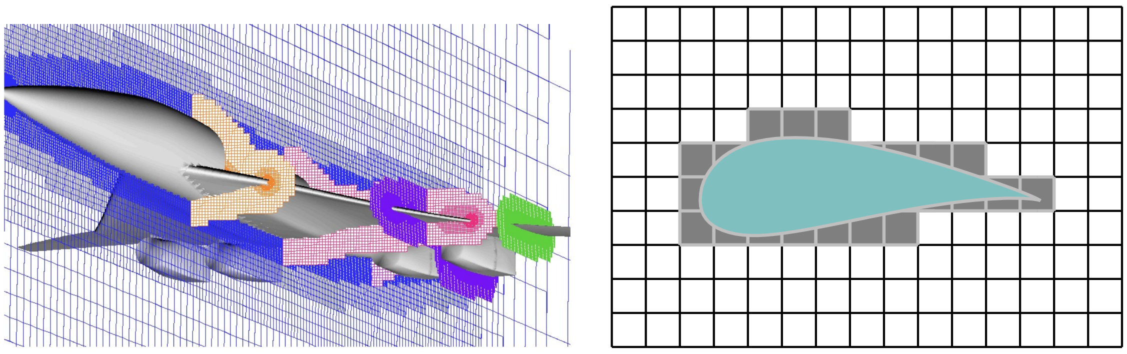
$$J_h(\mathbf{u}^h, \mathbf{w}^h) = J_h^0(\mathbf{u}^h, \mathbf{w}^h) + J_h^1(\mathbf{u}^h, \mathbf{w}^h) = \sum_{E \in I} \left(J_h^{0,E}(\mathbf{u}^h, \mathbf{w}^h) + J_h^{1,E}(\mathbf{u}^h, \mathbf{w}^h)\right). \tag{4.2}$$

The set I denotes the set of small cut cells that require stabilization and will be described later in the numerical results.

In general, the two terms $J_h^0(\cdot,\cdot)$ and $J_h^1(\cdot,\cdot)$ shall perform the same tasks as in the one-dimensional case:

- $J_h^0(\cdot,\cdot)$ is designed to ensure proper mass distribution *among* small cut cells and their neighbors. This is realized by means of terms that are evaluated at cell interfaces.
- $J_h^1(\cdot,\cdot)$ consists of volume terms that correct the mass distribution *within* cells in the neighborhood of small cut cells.

In what follows, we will discuss the particular definitions of $J_h^0(\cdot,\cdot)$ and $J_h^1(\cdot,\cdot)$ in more detail. We start with a short introduction regarding cut cells in two dimensions and the model problems we consider. Further, we will give an overview of the software that we will use in two dimensions. Afterwards, we will present the stabilization terms for different setups. We



(a) Three-dimensional cut cell grid of an airplane.[3] **(b)** Cut cell grid for an airfoil[4] consisting of curved and split cut cells.

Figure 4.1: Examples of complex cut cell configurations in different spatial dimensions.

will first start with the linear case and focus on the linear advection equation. After that, we will extend the stabilization to the case of non-linear systems in two dimensions. We will prove valuable theoretical properties in each section and show numerical results to support our findings.

4.1. Preparations for the cut cell approach in two dimensions

Two-dimensional cut cell grids differ significantly from those we discussed in one dimension. We start with general preparations for the DoD stabilization in two dimensions and discuss cut cell model problems and the software that we will use for our numerical tests.

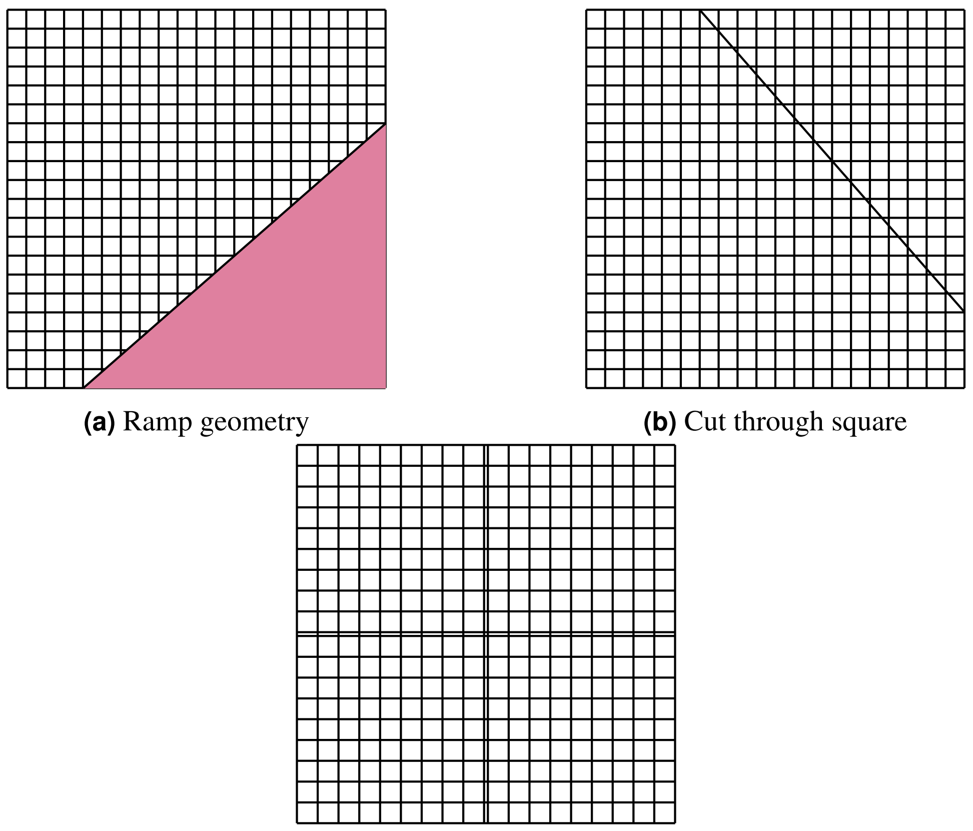
4.1.1. Cut cells in 2D

Embedded boundary meshes for complex geometries in higher spatial dimensions consist of different types of cut cells. Depending on how the given geometry intersects with the Cartesian grid, there can occur, e.g., *curved* and *split* cut cells, see Figure 4.1 for examples. Curved cut cells are created when a cut cell mesh is used to represent a curved boundary in a computational domain. Split cut cells emerge when the geometry intersects a Cartesian cell, such that the cell is split into multiple cut cells. These different cut cell types can become challenging to handle even without the need to stabilize them and will be analyzed in more detail in the future.

In this work, we focus on the development of new methods. Therefore, we take a step back and consider different model problems defined on the unit square $\Omega = [0,1]^2$. These model

[3] Picture taken from Cart3D website https://www.nas.nasa.gov/publications/software/docs/cart3d/

[4] Coordinates of airfoil taken from https://m-selig.ae.illinois.edu


(a) Ramp geometry **(b)** Cut through square

(c) Anisotropic grid

Figure 4.2: Three model problems in two dimensions, that we will discuss in this work.

problems are simpler than real-world applications. Nevertheless, they can cover various situations that occur when using cut-cell meshes for real-world applications.

The first model problem is the ramp geometry as illustrated in Figure 4.2a. Here, the linear advection parallel to the ramp is discussed for this geometry. This setup is a helpful initial step from the one-dimensional to the two-dimensional case. The reason is that for advection along the ramp, we can show that it is sufficient to stabilize triangular cells, which only have one inflow and one outflow neighbor. We will discuss this in more detail later.

For the second model problem, we consider a unit square geometry meshed with a Cartesian grid. Then, we construct cut cells by introducing a straight line that cuts through the domain. See Figure 4.2b for an illustration. This model problem differs from the first grid as we are now in the setting where we need to stabilize cells with possibly more than one inflow neighbor and outflow neighbor.

Finally, an anisotropic grid defines the third model problem. This model problem is a modified version of the anisotropic grid in the h-box paper [8]. The grid consists of one narrow column and one shallow row. We will use this model problem later to take a first step towards systems of non-linear conservation laws in two dimensions and to show preliminary results for the two-dimensional Euler equations.

4.1.2. DUNE software package

For the numerical simulations in the two-dimensional case, we use the *Distributed and Unified Numerics Environment* (DUNE) toolbox [2, 3]. DUNE is an open-source software framework that provides a modular and flexible platform for solving partial differential equations using finite elements, finite volumes, and finite differences. It is written in C++ and provides many data structures and algorithms for building and solving complex numerical simulations. Furthermore, its modular structure allows users to use only the modules needed for their specific applications. DUNE is widely used in the scientific and engineering communities for various applications, including fluid dynamics, solid mechanics, and electromagnetics.

Besides the set of DUNE core modules, which are used by most other DUNE packages, we rely for the cut cell simulations on the dune-pdelab [5], dune-udg [4, 24], and the TPMC [26] library. The module dune-pdelab provides an abstract interface for assembling and solving linear and non-linear systems of equations, arising from the discretization of PDEs. The dune-udg module allows easy implementation of unfitted discontinuous Galerkin (udg) methods on cut-cell grids. In addition, it offers a straightforward integration with dune-pdelab. The topology-preserving marching cubes (TPMC) method in the form of the dune-tpmc library will create cut cells and their corresponding quadrature rules.

4.2. Formulation for linear scalar problems

In this section, we consider the two-dimensional linear advection equation, which is given by

$$\begin{aligned} \partial_t u + \nabla \cdot (\beta u) &= 0 && \text{in } \Omega \times (0,T), \\ u &= g && \text{on } \Gamma^{\text{in}} \times (0,T), \\ u &= u_0 && \text{on } \Omega \times \{t=0\}. \end{aligned} \tag{4.3}$$

Here, $u : \Omega \times [0,T] \to \mathbb{R}$ is a scalar conserved variable, and $\beta : \mathbb{R}^2 \to \mathbb{R}^2$ is the velocity field, which we assume to be incompressible. The inflow boundary is defined as $\Gamma^{\text{in}} := \{\mathbf{x} \in \partial\Omega : (\beta(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x})) < 0\}$, with $\mathbf{n} \in \mathbb{R}^2$ is the outer unit normal vector on $\partial\Omega$. We start with the unstabilized semi-discrete problem, which we have already introduced in Section 2.2.2. We refer to Section 2.2 for the basic definitions regarding the two-dimensional setting. The unstabilized semi-discrete problem is given by: Find $u^h(t) \in V_h^p$ such that

$$\left(d_t u^h(t), w^h\right)_{L^2(\Omega)} + a_h\left(u^h(t), w^h\right) = 0, \quad \forall w^h \in V_h^p. \tag{4.4}$$

For the bilinear form $a_h(\cdot,\cdot)$, we will use the upwind flux, which can be formulated in the two-dimensional case as

$$\begin{aligned}\mathcal{H}(\mathbf{n}_e,a,b) &= (\beta\cdot\mathbf{n}_e)\frac{1}{2}(a+b)-\frac{1}{2}|\beta\cdot\mathbf{n}_e|(b-a)\\ &= (\beta\cdot\mathbf{n}_e)^{\oplus}a-(\beta\cdot\mathbf{n}_e)^{\ominus}b.\end{aligned} \tag{4.5}$$

The negative and positive components of a quantity $y\in\mathbb{R}$ are defined as $y^{\ominus}:=\frac{|y|-y}{2}$ and $y^{\oplus}:=\frac{|y|+y}{2}$. Note that $y^{\ominus},y^{\oplus}\geq 0$.

The bilinear form is then given by

$$\begin{aligned}a_h(u^h,w^h) =&-\sum_{E\in\mathcal{M}_h}\int_E u^h(\beta\cdot\nabla w^h)\mathrm{d}\mathbf{x}\\ &+\sum_{e\in\Gamma_h^{\text{int}}}\int_e\left((\beta\cdot\mathbf{n}_e)\left\{\!\!\left\{u^h\right\}\!\!\right\}\left[\!\left[w^h\right]\!\right]+\frac{1}{2}|\beta\cdot\mathbf{n}_e|\left[\!\left[u^h\right]\!\right]\left[\!\left[w^h\right]\!\right]\right)\mathrm{d}s\\ &+\sum_{e\in\Gamma_h^{\text{ext}}}\int_e\left((\beta\cdot\mathbf{n}_e)^{\oplus}u^h-(\beta\cdot\mathbf{n}_e)^{\ominus}g\right)w^h\mathrm{d}s\end{aligned}$$

with Γ_h^{int} and Γ_h^{ext} being the sets of internal and external edges, respectively. We now discuss different stabilization aspects using the model problems we defined above. We start with the ramp geometry (see Figure 4.2a above and 4.3 below) and consider advection along the ramp. The following section is mainly based on the findings in [25] and [76].

4.2.1. Ramp geometry

As we will see in the course of this section, advection along the ramp is an excellent first step when extending the DoD stabilization from one to two dimensions. The ramp geometry is entirely defined by the starting point $(x_0,0)$ and the angle γ relative to the x-axis, as shown in Figure 4.3. Note that the starting point $(x_0,0)$ does not need to align with the background mesh. After cutting out the ramp object, we obtain 3-sided, 4-sided, and 5-sided cut cells at the boundary, depending on the choice of x_0 and γ. We have already discussed in Section 2.3 that only some cut cells in two dimensions need stabilization. Whether or not a cut cell has to be stabilized depends on its size in the flow direction and the chosen time step Δt. Thus, we define the capacity of a cell in two dimensions.

Definition 25 (Capacity). *The capacity of a cut cell E for the linear advection equation with a velocity field $\beta\in\mathbb{R}^2$ is given by*

$$\alpha_E := \min\left(\frac{1}{2p+1}\,\frac{|E|}{\Delta t\int_{\partial E}(\beta\cdot\mathbf{n}_E)^{\ominus}ds},1\right) \tag{4.6}$$

with $|E|$ being the volume of cell E. The capacity measures the fraction of the inflow, that is allowed to flow into the cut cell E without producing an overshoot.

In order to ensure that large cut cells, that are almost the size of Cartesian cells, do not require stabilization, we choose the time step $\Delta t \leq \nu \frac{h}{\|\beta\|}$ with a slightly reduced CFL constant $\nu \in (0, \frac{1}{2}]$. Here, we define h as the edge length of the Cartesian cells. When considering advection along the ramp, one can show, that for this choice of Δt, only triangular cut cells require stabilization [25].

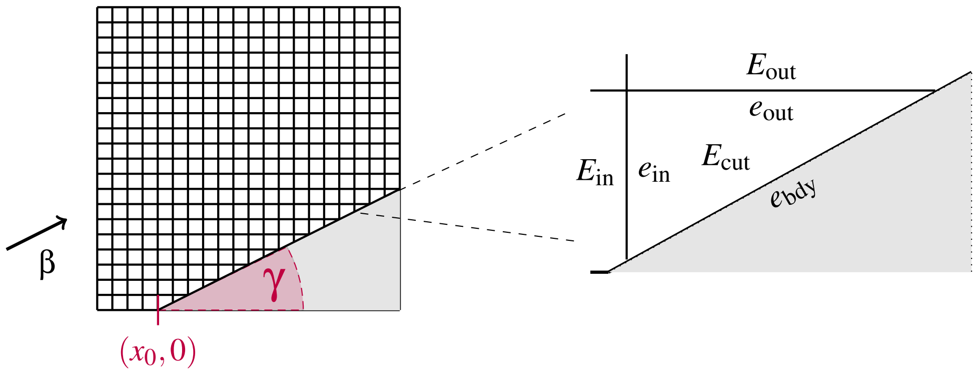


Figure 4.3: Left: Ramp geometry is entirely defined by the starting point $(x_0, 0)$ and the angle γ. Right: Zoom into a triangular cut cell.

This observation is helpful because triangular cut cells are unique in the ramp setting. For triangular cut cells, every edge of the cell has a different boundary condition. By definition of the model problem, the advection direction is parallel to the ramp. Without loss of generality, we assume that it is directed from the lower left corner to the upper right as indicated in Figure 4.3. For the zoomed-in triangular cut cell E_{cut} in Figure 4.3, we obtain the following boundary conditions:

- A no-flow boundary condition is imposed on the *boundary edge* e_{bdy} due to the flow being parallel to the ramp.
- One of the two remaining edges is identified as the *inflow edge* e_{in}, which is characterized by $(\beta \cdot \mathbf{n}_{E_{\text{cut}}}) < 0$.
- The other remaining edge is defined as the *outflow edge* e_{out}, which is characterized by $(\beta \cdot \mathbf{n}_{E_{\text{cut}}}) \geq 0$.

In our framework, the assumption of a penetration-free boundary condition at the intersection is a natural constraint to consider. The fact that we can uniquely define one inflow and one outflow edge for E_{cut} helps us transfer the one-dimensional ideas to the two-dimensional setting. In one dimension, every cell has two edges, which can be uniquely defined as an inflow or an outflow edge for the linear advection equation. As a result, we are now in a similar situation when comparing 1D and 2D. Thus, we copy the ideas from one dimension and transfer them to the two-dimensional setup.

In one dimension, the DoD stabilization for a cut cell E_{cut} is given as a sum of two stabilization terms:

$$J_h^{E_{\text{cut}}}(u^h, w^h) = J_h^{0,E_{\text{cut}}}(u^h, w^h) + J_h^{1,E_{\text{cut}}}(u^h, w^h) \tag{4.7}$$

The idea of the first stabilization term $J_h^{0,E_{\text{cut}}}(\cdot,\cdot)$ is to guarantee a proper mass distribution among the cut cell and its neighbors. This is achieved in 1D by adding an additional flux on the outflow edge, which directly transfers mass from the inflow neighbor to the outflow neighbor. The stabilization term $J_h^{0,E_{\text{cut}}}(\cdot,\cdot)$ for the one-dimensional linear advection equation is defined in (3.17). By taking this idea and transferring it to the two-dimensional setting, we obtain the first stabilization term

$$J_h^{0,E_{\text{cut}}}(u^h, w^h) = \eta_{E_{\text{cut}}} \int_{e_{\text{out}}} \left(\mathcal{L}_{E_{\text{in}}}^{\text{ext}}(u^h) - u_E\right)(\beta \cdot \mathbf{n}) \left[\!\left[w^h \right]\!\right] \mathrm{d}s. \tag{4.8}$$

Just as in 1D, we use the extension operator $\mathcal{L}_{E_{\text{in}}}^{\text{ext}}(\cdot)$, see Definition 16, to evaluate the solution of the inflow neighbor $u_{E_{\text{in}}}$ at the outflow edge e_{out}. Again, we scale this extended jump by a stabilization parameter $\eta_{E_{\text{cut}}}$. The choice of $\eta_{E_{\text{cut}}}$ offers again some flexibility, as we will see in the next section. We choose the stabilization parameter as

$$\eta_{E_{\text{cut}}} = 1 - \alpha_{E_{\text{cut}}} \tag{4.9}$$

with the capacity $\alpha_{E_{\text{cut}}} = \min\left(\frac{1}{2p+1}\ \frac{|E_{\text{cut}}|}{\Delta t \int_{\partial E_{\text{cut}}} (\beta \cdot \mathbf{n}_{E_{\text{cut}}})^{\ominus} \mathrm{d}s}, 1\right)$ according to Definition 25. In the context of the linear advection equation, one can show that the stabilization parameter in 1D is the one-dimensional equivalent of equation (4.9) [25].

The second stabilization term $J_h^{1,E_{\text{cut}}}(\cdot,\cdot)$ is given in one dimension as a volume term, see equation (3.18). The idea of this second term is to redistribute the mass within the cells in the neighborhood of the small cut cell E_{cut}. Once again, we take the ideas of 1D and create the two-dimensional version of $J_h^{1,E_{\text{cut}}}(\cdot,\cdot)$ by replacing the one-dimensional terms with their corresponding counterparts in 2D. Thus, we obtain

$$J_h^{1,E}(u^h, w^h) = \eta_E \int_E \left(\mathcal{L}_{E_{\text{in}}}^{\text{ext}}(u^h) - u_E\right)\left(\beta \cdot \left(\mathcal{L}_{E_{\text{in}}}^{\text{ext}}(\nabla w^h) - \nabla w_E\right)\right) \mathrm{d}\mathbf{x}. \tag{4.10}$$

Here, we apply the extension operator $\mathcal{L}_{E_{\text{in}}}^{\text{ext}}(\cdot,\cdot)$ to both the discrete solution and the test function of the inflow neighbor E_{in}. Comparing the stabilization in one and two dimensions reveals the resemblance of the terms and shows that the underlying ideas remain unchanged, despite the difference in dimensionality. The final DoD stabilized semi-discrete scheme is then given by: Find $u^h(t) \in V_h^p$ such that

$$\left(d_t u^h(t), w^h\right)_{L^2} + a_h\left(u^h(t), w^h\right) + J_h\left(u^h(t), w^h\right) = 0, \quad \forall w^h \in V_h^p, \tag{4.11}$$

with the stabilization term

$$J_h(u^h, w^h) = J_h^0(u^h, w^h) + J_h^1(u^h, w^h) = \sum_{E \in I} \left(J_h^{0,E}(u^h, w^h) + J_h^{1,E}(u^h, w^h)\right).$$

It is worth noting that the formulation of the stabilization terms becomes more advanced when dealing with cells with multiple in- or outflow edges. We will discuss this case later and focus now on theoretical and numerical results for the ramp geometry.

Theoretical results

The similarity between the stabilization in different spatial dimensions allows us to transfer theoretical results from one dimension into higher spatial dimensions.

Theorem 26 (Monotonicity)**.** *Consider the stabilized scheme for the ramp geometry using the functional space V_h^0 in space and explicit Euler in time applied to the linear advection equation (4.3). Additionally, choose the time step $\Delta t \leq \nu \frac{h}{\|\beta\|}$ with $0 < \nu \leq 0.5$. Then, the stabilized update on a small cut cell E_{cut} is monotone for $\eta_{E_{cut}} \in [1 - \alpha_{E_{cut}}, 1]$.*

Proof. We consider a triangular cut cell from the ramp geometry with the notation of Figure 4.3. The unstabilized update is given by

$$u_{E_{\text{cut}}}^{n+1} = u_{E_{\text{cut}}}^n - \frac{\Delta t}{|E_{\text{cut}}|}\left(\int_{e_{\text{in}}} -(\beta \cdot \mathbf{n}_{e_{\text{in}}})^{\ominus} u_{E_{\text{in}}} \mathrm{d}s + \int_{e_{\text{out}}} (\beta \cdot \mathbf{n}_{e_{\text{out}}})^{\oplus} u_{E_{\text{cut}}} \mathrm{d}s\right). \tag{4.12}$$

We emphasize that the update formula is missing a flux on edge e_{bdy}, because a no-flow boundary condition is imposed there. Adding $J_h^{0,E_{\text{cut}}}(\cdot,\cdot)$ given by (4.8) to the update formula (4.12), we obtain

$$\begin{aligned} u_{E_{\text{cut}}}^{n+1} = u_{E_{\text{cut}}}^n &- \frac{\Delta t}{|E_{\text{cut}}|}\left(\int_{e_{\text{in}}} -(\beta \cdot \mathbf{n}_{e_{\text{in}}})^{\ominus} u_{E_{\text{in}}} \mathrm{d}s + \int_{e_{\text{out}}} (\beta \cdot \mathbf{n}_{e_{\text{out}}})^{\oplus} u_{E_{\text{cut}}} \mathrm{d}s\right) \\ &- \frac{\Delta t}{|E_{\text{cut}}|}\eta_{E_{\text{cut}}} \int_{e_{\text{out}}} (\beta \cdot \mathbf{n}_{e_{\text{out}}})^{\oplus} (u_{E_{\text{in}}} - u_{E_{\text{cut}}}) \mathrm{d}s. \end{aligned}$$

Using that $\nabla \cdot \beta = 0$, we have

$$\begin{aligned} 0 = \int_{E_{\text{cut}}} (\nabla \cdot \beta) \mathrm{d}\mathbf{x} &= \int_{\partial E_{\text{cut}}} (\beta \cdot \mathbf{n}) \mathrm{d}s \\ &= \int_{e_{\text{out}}} (\beta \cdot \mathbf{n}_{e_{\text{out}}})^{\oplus} \mathrm{d}s - \int_{e_{\text{in}}} (\beta \cdot \mathbf{n}_{e_{\text{in}}})^{\ominus} \mathrm{d}s \\ \Rightarrow \quad \int_{e_{\text{out}}} (\beta \cdot \mathbf{n}_{e_{\text{out}}})^{\oplus} \mathrm{d}s &= \int_{e_{\text{in}}} (\beta \cdot \mathbf{n}_{e_{\text{in}}})^{\ominus} \mathrm{d}s. \end{aligned} \tag{4.13}$$

Since $u_{E_{\text{in}}}$ is constant, the stabilized update on the cut cell reads as

$$u_{E_{\text{cut}}}^{n+1} = u_{E_{\text{cut}}}^{n} - \frac{\Delta t}{|E_{\text{cut}}|} (1 - \eta_{E_{\text{cut}}}) \left(\int_{e_{\text{in}}} -(\beta \cdot \mathbf{n}_{e_{\text{in}}})^{\ominus} u_{E_{\text{in}}} \mathrm{d}s + \int_{e_{\text{out}}} (\beta \cdot \mathbf{n}_{e_{\text{out}}})^{\oplus} u_{E_{\text{cut}}} \mathrm{d}s \right).$$

Next, we will check the partial derivatives of the update formula according to Definition 8. For the partial derivative with respect to the inflow neighbor, we obtain

$$\frac{\partial}{\partial u_{E_{\text{in}}}} u_{E_{\text{cut}}}^{n+1} = -\frac{\Delta t}{|E_{\text{cut}}|} (1 - \eta_{E_{\text{cut}}}) \int_{e_{\text{in}}} -(\beta \cdot \mathbf{n}_{e_{\text{in}}})^{\ominus} \mathrm{d}s \geq 0.$$

Finally, for the partial derivative with respect to $u_{E_{\text{cut}}}$, the following condition holds:

$$\begin{aligned} \frac{\partial}{\partial u_{E_{\text{cut}}}} u_{E_{\text{cut}}}^{n+1} &= 1 - \frac{\Delta t}{|E_{\text{cut}}|} (1 - \eta_{E_{\text{cut}}}) \int_{e_{\text{out}}} (\beta \cdot \mathbf{n}_{e_{\text{out}}})^{\oplus} \mathrm{d}s \overset{!}{\geq} 0 \\ \Leftrightarrow \qquad 1 - \frac{|E_{\text{cut}}|}{\Delta t \int_{e_{\text{out}}} (\beta \cdot \mathbf{n}_{e_{\text{out}}})^{\oplus} \mathrm{d}s} &= 1 - \alpha_{E_{\text{cut}}} \leq \eta_{E_{\text{cut}}} \end{aligned}$$

Note that the equality in the last line holds due to the calculation in (4.13). This concludes the proof. □

Next, we prove L^2 stability for the stabilized semi-discrete scheme for an arbitrary polynomial degree p. In general, L^2 stability is affected by the inflow and outflow through $\partial\Omega^{\text{in}}$ and $\partial\Omega^{\text{out}}$ during $(0,T)$, but in the ramp setup only Cartesian faces are in $\partial\Omega^{\text{in}} \cup \partial\Omega^{\text{out}}$. Our aim is to show L^2 stability for the stabilized scheme *with cut cells*, not to analyze inflow/outflow influence on L^2 stability. To simplify, we assume the solution has compact support inside Ω during $(0,T)$ and does not intersect the Cartesian boundary ($\text{supp}(u) \cap (\partial\Omega^{\text{in}} \cup \partial\Omega^{\text{out}}) = \emptyset$).

Theorem 27. *[63, Theorem 1] Consider the advection equation (4.3) for the setup of a ramp with incompressible velocity field $\beta = (\beta_1, \beta_2)^T$ parallel to the ramp. Let the solution u have compact support for $t \in (0,T)$ and $\text{supp}(u) \cap (\Gamma^{in} \cup \Gamma^{out}) = \emptyset$. Let $u^h(t) \in V_h^p$ be the solution to the stabilized semi-discrete problem (4.11). Under the assumption of exact integration, the semi-discrete solution then satisfies*

$$\left\| u^h(t) \right\|_{L^2(\Omega)} \leq \left\| u^h(0) \right\|_{L^2(\Omega)} \qquad \forall t \in (0,T).$$

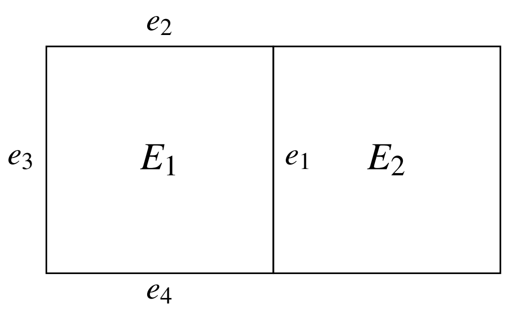


Figure 4.4: [76] Setup for Cartesian cells.

Proof. Setting $w^h = u^h(t)$ in (4.11) and ignoring boundary contributions with respect to Γ^{in}, we achieve

$$\left(d_t u^h(t), u^h(t)\right)_{L^2(\Omega)} + a_h\left(u^h(t), u^h(t)\right) + J_h\left(u^h(t), u^h(t)\right) = 0.$$

Integration of the first term in time yields

$$\begin{aligned}\int_0^t \left(d_\tau u^h(\tau), u^h(\tau)\right)_{L^2(\Omega)} \mathrm{d}\tau &= \int_0^t \frac{d}{d\tau}\frac{1}{2}\left\|u^h(\tau)\right\|^2_{L^2(\Omega)} \mathrm{d}\tau \\ &= \frac{1}{2}\left\|u^h(t)\right\|^2_{L^2(\Omega)} - \frac{1}{2}\left\|u^h(0)\right\|^2_{L^2(\Omega)},\end{aligned}$$

and it remains to show that for any fixed t

$$a_h(u^h(t), u^h(t)) + J_h(u^h(t), u^h(t)) \geq 0.$$

We will first discuss $a_h(\cdot,\cdot)$ and then $J_h(\cdot,\cdot)$. (We omit the explicit time dependence from now on for brevity.)

By definition of $a_h(\cdot,\cdot)$ and ignoring outflow across Γ^{out}, the bilinear form satisfies

$$\begin{aligned}a_h(u^h, u^h) = &- \sum_{E \in \mathcal{M}_h} \int_E u^h\left(\beta \cdot \nabla u^h\right) \mathrm{d}x \\ &+ \sum_{e \in \Gamma_h^{\text{int}}} \int_e \left((\beta \cdot \mathbf{n}_e) \left\{\!\!\left\{u^h\right\}\!\!\right\} \left[\!\left[u^h\right]\!\right] + \frac{1}{2}\left|(\beta \cdot \mathbf{n}_e)\right| \left[\!\left[u^h\right]\!\right]^2 \right) \mathrm{d}s.\end{aligned}$$

For the integral term, we rewrite (exploiting $\nabla \cdot \beta = 0$)

$$-\int_E u^h\left(\beta \cdot \nabla u^h\right) \mathrm{d}\mathbf{x} = -\int_E \nabla \cdot \left(\frac{1}{2}\beta (u^h)^2\right) \mathrm{d}\mathbf{x} = -\int_{\partial E} \left(\frac{1}{2}\beta (u^h)^2\right) \cdot \mathbf{n}\, \mathrm{d}s.$$

Let us first consider a standard Cartesian cell E_1 with edges as shown in Figure 4.4. Then, for $\beta = (\beta_1, \beta_2)^T$ and assuming without loss of generality that $\beta_1, \beta_2 \geq 0$, we have

$$-\int_{E_1} u_h \left(\beta \cdot \nabla u^h\right) \mathrm{d}\mathbf{x} = -\int_{e_1} \frac{1}{2}\beta_1 (u^h)^2 \,\mathrm{d}s - \int_{e_2} \frac{1}{2}\beta_2 (u^h)^2 \,\mathrm{d}s + \int_{e_3} \frac{1}{2}\beta_1 (u^h)^2 \,\mathrm{d}s + \int_{e_4} \frac{1}{2}\beta_2 (u^h)^2 \,\mathrm{d}s.$$

For the edge terms in $a_h(\cdot,\cdot)$, let us consider an internal edge e_1 connecting E_1 and E_2. Then, (using from now on the notation $u_{E'}$ to indicate that we evaluate the discrete solution from cell E', potentially outside of its original support)

$$\begin{aligned}
&\int_{e_1} \left((\beta \cdot \mathbf{n}_{e_1}) \left\{\!\!\left\{ u^h \right\}\!\!\right\} \left[\!\left[u^h \right]\!\right] + \frac{1}{2} |(\beta \cdot \mathbf{n}_{e_1})| \left[\!\left[u^h \right]\!\right]^2 \right) \mathrm{d}s \\
&\quad = \int_{e_1} \left(\frac{1}{2}\beta_1 (u_{E_1} + u_{E_2})(u_{E_1} - u_{E_2}) + \frac{1}{2}\beta_1 (u_{E_1} - u_{E_2})^2 \right) \mathrm{d}s \\
&\quad = \int_{e_1} \beta_1 \left((u_{E_1})^2 - u_{E_1} u_{E_2} \right) \mathrm{d}s.
\end{aligned}$$

Combining this with the corresponding contributions for edge e_1 from the volume terms from cells E_1 and E_2, we get

$$\begin{aligned}
-\int_{e_1} \frac{1}{2}\beta_1 (u_{E_1})^2 \,\mathrm{d}s + &\int_{e_1} \frac{1}{2}\beta_1 (u_{E_2})^2 \,\mathrm{d}s \\
&+ \int_{e_1} \left((\beta \cdot \mathbf{n}_{e_1}) \left\{\!\!\left\{ u^h \right\}\!\!\right\} \left[\!\left[u^h \right]\!\right] + \frac{1}{2} |(\beta \cdot \mathbf{n}_{e_1})| \left[\!\left[u^h \right]\!\right]^2 \right) \mathrm{d}s \\
&= \int_{e_1} \left(\frac{1}{2}\beta_1 (u_{E_1})^2 - \beta_1 u_{E_1} u_{E_2} + \frac{1}{2}\beta_1 (u_{E_2})^2 \right) \mathrm{d}s \\
&= \int_{e_1} \frac{1}{2}\beta_1 \left(u_{E_1} - u_{E_2}\right)^2 \mathrm{d}s.
\end{aligned}$$

Let us now add the cut cells. For the small triangular cut cell E_{cut} with the notation from Figure 4.3, we obtain with $\beta = (\beta_1, \beta_1)^T$ and assuming that $\beta_1, \beta_2 \geq 0$

$$-\int_{E_{\text{cut}}} u^h (\beta \cdot \nabla u_h) \,\mathrm{d}\mathbf{x} = -\int_{\partial E_{\text{cut}}} \left(\frac{1}{2}\beta (u^h)^2 \right) \cdot \mathbf{n} \,\mathrm{d}s = -\int_{e_{\text{out}}} \frac{1}{2}\beta_2 (u^h)^2 \,\mathrm{d}s + \int_{e_{\text{in}}} \frac{1}{2}\beta_1 (u^h)^2 \,\mathrm{d}s.$$

Therefore, taking the boundary term in $a_h(\cdot,\cdot)$ into account as well as the contribution from the volume term of cell E_{in}, we get for the edge e_{in}

$$\begin{aligned}
-\int_{e_{\text{in}}} \frac{1}{2}\beta_1 (u_{E_{\text{in}}})^2 \,\mathrm{d}s + \int_{e_{\text{in}}} \frac{1}{2}\beta_1 (u_{E_{\text{cut}}})^2 \,\mathrm{d}s \\
+ \int_{e_{\text{in}}} \left((\beta\cdot\mathbf{n}_{e_{\text{in}}}) \left\{\!\!\left\{u^h\right\}\!\!\right\} \left[\!\left[u^h\right]\!\right] + \frac{1}{2} |(\beta\cdot\mathbf{n}_{e_{\text{in}}})| \left[\!\left[u^h\right]\!\right]^2 \right) \mathrm{d}s \\
= \int_{e_{\text{in}}} \left(\frac{1}{2}\beta_1 (u_{E_{\text{in}}})^2 - \beta_1 u_{E_{\text{in}}} u_{E_{\text{cut}}} + \frac{1}{2}\beta_1 (u_{E_{\text{cut}}})^2 \right) \mathrm{d}s \\
= \int_{e_{\text{in}}} \frac{1}{2}\beta_1 \left(u_{E_{\text{in}}} - u_{E_{\text{cut}}}\right)^2 \mathrm{d}s.
\end{aligned}$$

We obtain a similar term for edge e_{out}, involving solutions from cells E_{cut} and E_{out}. Thus, ignoring boundary contributions across $\partial\Omega^{\text{in}} \cup \partial\Omega^{\text{out}}$ due to the assumption of compact support, there holds

$$a_h(u^h,u^h) = \sum_{e\in\Gamma_h^{\text{int}}} \frac{1}{2} \int_e |(\beta\cdot\mathbf{n}_e)| \left[\!\left[u^h\right]\!\right]^2 \mathrm{d}s. \tag{4.14}$$

Finally, without the stabilization term $J_h(\cdot,\cdot)$, there holds L^2 stability.

Let us now add the stabilization term

$$J_h(u^h,u^h) = \sum_{E\in I} J_h^{0,E}(u^h,u^h) + J_h^{1,E}(u^h,u^h).$$

We only stabilize small triangular cells of type E_{cut}. There holds

$$\begin{aligned}
J_h^{0,E_{\text{cut}}}(u^h,u^h) &= \eta_{E_{\text{cut}}} \int_{e_{\text{out}}} (u_{E_{\text{in}}} - u_{E_{\text{cut}}}) (\beta\cdot\mathbf{n}_{e_{\text{out}}}) \left[\!\left[u^h\right]\!\right] \mathrm{d}s \\
&= \eta_{E_{\text{cut}}} \int_{e_{\text{out}}} \beta_2 (u_{E_{\text{in}}} - u_{E_{\text{cut}}})(u_{E_{\text{cut}}} - u_{E_{\text{out}}}) \mathrm{d}s \\
&= \eta_{E_{\text{cut}}} \int_{e_{\text{out}}} \beta_2 \left(u_{E_{\text{in}}} u_{E_{\text{cut}}} - u_{E_{\text{in}}} u_{E_{\text{out}}} - (u_{E_{\text{cut}}})^2 + u_{E_{\text{cut}}} u_{E_{\text{out}}}\right) \mathrm{d}s.
\end{aligned}$$

We now consider $J_h^{1,E_{\text{cut}}}(\cdot,\cdot)$ given by

$$J_h^{1,E_{\text{cut}}}(u^h,u^h) = \eta_{E_{\text{cut}}} \int_{E_{\text{cut}}} (u_{E_{\text{in}}} - u_{E_{\text{cut}}}) (\beta\cdot(\nabla u_{E_{\text{in}}} - \nabla u_{E_{\text{cut}}}))\, \mathrm{d}\mathbf{x}.$$

With $\beta = (\beta_1, \beta_2)^T$ and $\nabla \cdot \beta = 0$, there holds

$$\begin{aligned}
J_h^{1,E_{\text{cut}}}(u^h, u^h) &= \eta_{E_{\text{cut}}} \int_{E_{\text{cut}}} \nabla \cdot \left(\frac{1}{2} \beta (u_{E_{\text{in}}} - u_{E_{\text{cut}}})^2 \right) \mathrm{d}\mathbf{x} \\
&= \eta_{E_{\text{cut}}} \int_{\partial E_{\text{cut}}} \left(\frac{1}{2} \beta (u_{E_{\text{in}}} - u_{E_{\text{cut}}})^2 \right) \cdot \mathbf{n} \, \mathrm{d}s \\
&= \eta_{E_{\text{cut}}} \int_{e_{\text{out}}} \left(\frac{1}{2} \beta_2 (u_{E_{\text{in}}} - u_{E_{\text{cut}}})^2 \right) \mathrm{d}s - \eta_{E_{\text{cut}}} \int_{e_{\text{in}}} \left(\frac{1}{2} \beta_1 (u_{E_{\text{in}}} - u_{E_{\text{cut}}})^2 \right) \mathrm{d}s.
\end{aligned}$$

As $0 \leq \eta_{E_{\text{cut}}} \leq 1$, the negative term over the edge e_{in} can be compensated with the edge term $\int_{e_{\text{in}}} \beta_1 \left(\frac{1}{2} (u_{E_{\text{in}}} - u_{E_{\text{cut}}})^2 \right) \mathrm{d}s$ from $a_h(\cdot,\cdot)$ in (4.14). For the edge e_{out}, we collect all terms from $J_h^{0,E_{\text{cut}}}(\cdot,\cdot)$ and $J_h^{1,E_{\text{cut}}}(\cdot,\cdot)$ leading to

$$\begin{aligned}
&\eta_{E_{\text{cut}}} \int_{e_{\text{out}}} \beta_2 \left(u_{E_{\text{in}}} u_{E_{\text{cut}}} - u_{E_{\text{in}}} u_{E_{\text{out}}} - (u_{E_{\text{cut}}})^2 + u_{E_{\text{cut}}} u_{E_{\text{out}}} + \frac{1}{2} (u_{E_{\text{in}}} - u_{E_{\text{cut}}})^2 \right) \mathrm{d}s \\
&= \eta_{E_{\text{cut}}} \int_{e_{\text{out}}} \beta_2 \left(\frac{1}{2} (u_{E_{\text{in}}})^2 - \frac{1}{2} (u_{E_{\text{cut}}})^2 - u_{E_{\text{in}}} u_{E_{\text{out}}} + u_{E_{\text{cut}}} u_{E_{\text{out}}} \right) \mathrm{d}s \\
&= \eta_{E_{\text{cut}}} \int_{e_{\text{out}}} \frac{1}{2} \beta_2 (u_{E_{\text{in}}} - u_{E_{\text{out}}})^2 \, \mathrm{d}s - \eta_{E_{\text{cut}}} \int_{e_{\text{out}}} \frac{1}{2} \beta_2 (u_{E_{\text{cut}}} - u_{E_{\text{out}}})^2 \, \mathrm{d}s.
\end{aligned}$$

The right term in the last line involves the standard jump over edge e_{out} and (same as for edge e_{in}) can be compensated with its positive counterpart in the sum in (4.14). The first term in the last line consists of a new extended jump involving the difference of the solution of cell E_{in} and the solution of cell E_{out}, both evaluated on the outflow edge e_{out}. This concludes the proof. □

Numerical results

In this subsection, we present numerical results for the linear advection equation in 2D using higher-order polynomials for the ramp setup. For the definition of the initial data, we will use a rotated and shifted coordinate system $(\hat{x}, \hat{y})$, which is defined in such a way that the $\hat{x}$-direction is parallel and the $\hat{y}$-direction is orthogonal to the ramp. The introduced coordinate system can be derived from the standard Cartesian coordinate system (x, y) using the angle γ and the starting point of the ramp $(x_0, 0)$. We obtain the shifted and rotated coordinate system by the following formula

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \cos\gamma & \sin\gamma \\ -\sin\gamma & \cos\gamma \end{pmatrix} \cdot \begin{pmatrix} x - x_0 \\ y \end{pmatrix}. \tag{4.15}$$

Using this coordinate system, we consider the following test setup [25, 76]: We start the

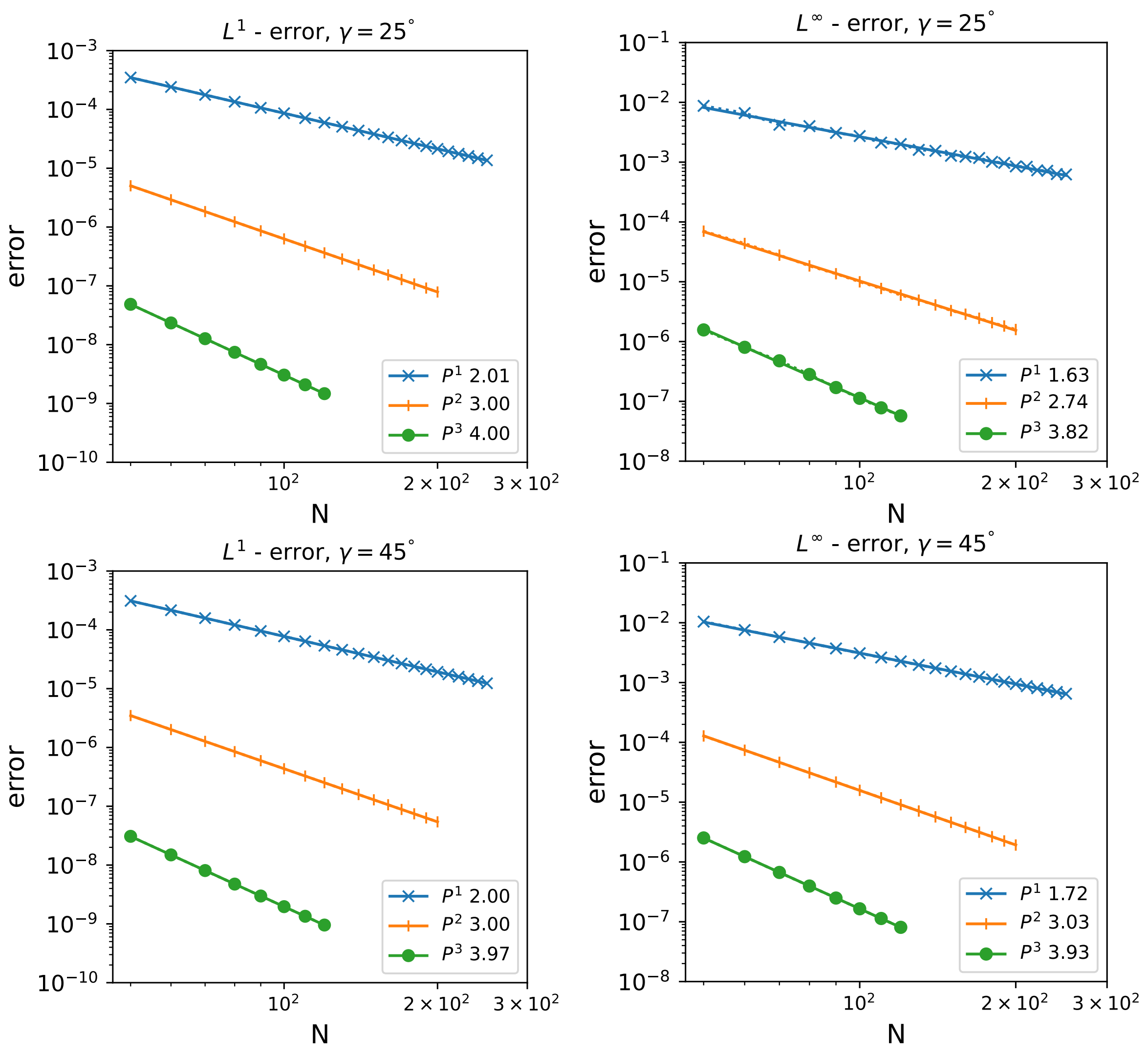


Figure 4.5: [76] Convergence orders in L^1 and L^∞ norm for the error at time $T = 0.3$ for a ramp geometry with $\gamma = 25^\circ$ (top) and $\gamma = 45^\circ$ (bottom) and different polynomial degrees $p = 1, 2, 3$.

ramp at $x_0 = 0.2001$ and run convergence tests for different angles γ. The velocity field $\beta \in \mathbb{R}^2$ is given by

$$\beta(\hat{x}, \hat{y}) = (2 - \hat{y}) \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The solenoidal velocity field β is parallel to the ramp, while its magnitude decreases for an increasing distance to the ramp. We choose the smooth initial data

$$u_0(\hat{x}, \hat{y}) = \sin\left(\frac{\sqrt{2}\pi\hat{x}}{1 - 0.2001} \right),$$

which has the exact solution

$$u(\hat{x}, \hat{y}, t) = u_0(\hat{x} - t\beta_1, \hat{y} - t\beta_2).$$

We derive the inflow conditions on Γ^{in} from the exact solution. In addition, we compute the discrete solution at time $T = 0.3$ using piecewise polynomials of degrees $p = 1, 2, 3$. In time we use an SSP RK scheme of the same order as the space discretization. The time step size Δt is given by

$$\Delta t \leq 0.4 \frac{1}{2p+1} \frac{h}{\|\beta\|}$$

with h being the edge length of the Cartesian cells. We choose the set of stabilized cells $I = \{E \,|\, E \text{ is triangle and } |E| < 0.4h^2\}$.

The convergence results for ramp angles of $\gamma = 25^\circ$ and $\gamma = 45^\circ$ are presented in Figure 4.5 for both L^1 and L^∞ norms. The convergence orders observed for polynomials of degree p in the L^1 norm are approximately $p+1$ for both angles. In the L^∞ norm, the results are between $p + \frac{1}{2}$ and $p+1$. It is common to observe a phenomenon of reduced convergence order of the L^∞ error at the cut boundary. Other cut cell approaches observed and reported this behavior in the past [6–8, 31, 43, 48]. Achieving full order of accuracy in the L^∞ norm on cut cell meshes in higher spatial dimensions is generally challenging. One possible explanation for this behavior is that the one-step error on cut cell meshes is typically one order lower than that on structured meshes. While in one dimension, it is often observed that these additional error sources do not accumulate, compare Proposition 4, the error accumulation in two dimensions is not studied well and needs to be investigated in more detail in the future.

4.2.2. Cut cells with multiple inflow and outflow edges

Next, we will discuss the case of multiple inflow and outflow edges for the linear advection equation. This causes more interactions between the neighbors of small cut cells that need to be considered. Due to the additional complexity, we will focus on the function space V_h^0, which consists of piecewise constant polynomials. We consider the second model problem shown in Figure 4.2b. The computational domain consists of an equidistantly meshed unit square $\Omega = [0, 1]^2$ with one artificial cut through the domain. The Cartesian cells divided by that cut can then be reinterpreted as two cut cells. In contrast to the case of advection along the ramp, as shown in Figure 4.3 above, we will now consider a velocity field, which is not necessarily parallel to any of the cut cells' edges.

The content of this section is partly based on the collaboration in [10]. In addition, certain ideas that will be presented in the following have previously been published in the master's thesis of G. Birke [9]. During our collaboration, we worked simultaneously on the linear advection equation for multiple inflow and outflow edges but with different goals: While the

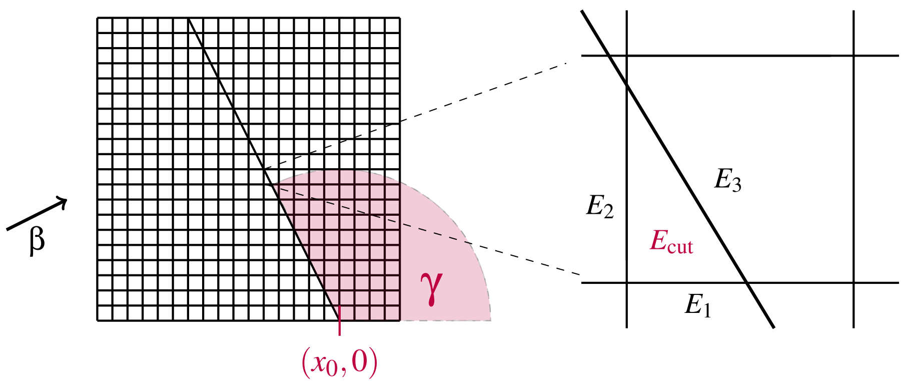


Figure 4.6: Second model problem given by an artificial cut through a square with a zoom into an example of cut cell E_{cut}.

work in [9] focuses on extending the stabilization to linear systems (in particular the acoustic equation), we aim to extend to non-linear problems. The formulation that we will present in the following for the linear advection, which is suitable for extension to Burgers equation, is different than the formulation in [9], but it can be shown, using algebraic manipulations, that they coincide for linear advection.

The fundamental idea of the DoD stabilization is to redistribute mass between inflow and outflow neighbors of small cut cells. For the new set of cells with multiple inflow or outflow neighbors, we have several pairwise interactions to consider. Since we will focus on the case of piecewise constant polynomials, the stabilization term $J_h(\cdot,\cdot)$ only consists of the edge stabilization term $J_h^0(\cdot,\cdot)$

$$J_h(u^h,w^h) = J_h^0(u^h,w^h) = \sum_{E\in I} J_h^{0,E}(u^h,w^h).$$

We consider the example of the cut cell E_{cut}, shown in Figures 4.6 and 4.7. For this particular cell and the given velocity field β we have $\beta\cdot\mathbf{n}_{e_1}\le 0$, $\beta\cdot\mathbf{n}_{e_2}\le 0$ and $\beta\cdot\mathbf{n}_{e_3}\ge 0$. Consequently, the cells E_1 and E_2 are inflow neighbors, and E_3 is an outflow neighbor. In this setting, the DoD stabilization should move mass from E_1 and E_2 directly to cell E_3. This will extend the domain of dependence of cell E_3 to include cells E_1 and E_2. Thus, the stabilization should add an additional flux on the outflow edge e_3 containing extended jumps from the two inflow edges e_1 and e_2. Consequently, we suggest the following stabilization term for the particular cell E_{cut}:

$$J_h^{0,E_{\text{cut}}}(u^h,w^h) = \eta_{E_{\text{cut}}}\int_{e_3}\left(\omega_{1,3}\mathcal{L}_{E_1}^{\text{ext}}(u^h)+\omega_{2,3}\mathcal{L}_{E_2}^{\text{ext}}(u^h)-u_{E_{\text{cut}}}\right)(\beta\cdot\mathbf{n}_{e_3})\left[\!\left[w^h\right]\!\right]\mathrm{d}s \tag{4.16}$$

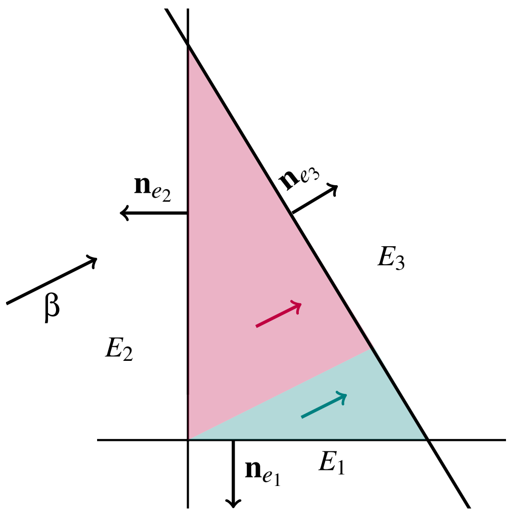


Figure 4.7: Example of triangular cut cell with two inflow neighbors E_1 and E_2 and one outflow neighbor E_3.

Here, we introduce *weights* $\omega_{1,3}, \omega_{2,3} \in [0,1]$ that provide information about the relation between the inflow cells E_i, $i \in \{1,2\}$, and the outflow cell E_3. In order to obtain a consistent stabilized method, we demand that the weights build a convex combination

$$\omega_{1,3} + \omega_{2,3} = 1. \tag{4.17}$$

Therefore, we can rewrite equation (4.16) to obtain

$$\begin{aligned} J_h^{0,E_{\text{cut}}}(u^h, w^h) =& \eta_{E_{\text{cut}}} \int_{e_3} \Big(\omega_{1,3} \left(\mathcal{L}_{E_1}^{\text{ext}}(u^h) - u_{E_{\text{cut}}} \right) \\ &+ \omega_{2,3} \left(\mathcal{L}_{E_2}^{\text{ext}}(u^h) - u_{E_{\text{cut}}} \right) \Big) (\beta \cdot \mathbf{n}_{e_3}) \left[\!\left[w^h \right]\!\right] \mathrm{d}s. \end{aligned}$$

Next, we explain how we compute the weights for the case of linear advection in two dimensions. In general, the weights specify the *direction* in which stabilization is applied, while the difference of the extended fluxes determines the *amount* of the stabilization. We define the weights $\omega_{i,j}$ such that they measure the proportion of the inflow coming through edge e_i to the total inflow of cell E_{cut} for a constant solution u_c. For the computation of the weights, we introduce the following set containing all edges of a cell E:

$$F_h^E = \{ e \in \Gamma_h^{\text{int}} \cup \Gamma_h^{\text{ext}} |\ e \subset \partial E \}$$

Since we apply the stabilization for linear advection with an upwind flux only on outflow edges, we set

$$\omega_{i,j} = 0, \quad \forall\, j \text{ with } (\beta \cdot \mathbf{n}_{e_j}) \leq 0. \tag{4.18}$$

For the final formulation of the weights, we use the Jacobians of the upwind flux function, which is given by

$$\mathcal{H}(\mathbf{n}_e, a, b) = (\beta \cdot \mathbf{n}_e)^{\oplus} a - (\beta \cdot \mathbf{n}_e)^{\ominus} b. \tag{4.19}$$

The Jacobians of the numerical flux then read as

$$\mathcal{H}_a(\mathbf{n}_e, a, b) = (\beta \cdot \mathbf{n}_e)^{\oplus} \quad \text{and} \quad \mathcal{H}_b(\mathbf{n}_e, a, b) = -(\beta \cdot \mathbf{n}_e)^{\ominus}. \tag{4.20}$$

Finally, we suggest the following formula for the weights using an arbitrary constant value $u_c \in \mathbb{R}^+$

$$\omega_{i,j} = \frac{\int_{e_i} \mathcal{H}_b(\mathbf{n}_{e_i}, u_c, u_c) \mathrm{d}s}{\sum_{e_k \in F_h^E} \int_{e_k} \mathcal{H}_b(\mathbf{n}_{e_k}, u_c, u_c) \mathrm{d}s} \chi_j^+ = \frac{\int_{e_i} (\beta \cdot \mathbf{n}_{e_i})^{\ominus} \mathrm{d}s}{\sum_{e_k \in F_h^E} \int_{e_k} (\beta \cdot \mathbf{n}_{e_k})^{\ominus} \mathrm{d}s} \chi_j^+. \tag{4.21}$$

The factor χ_j^+ is an indicator that should enforce the requirement (4.18). Thus, we choose

$$\chi_j^+ = \begin{cases} 1, & \text{if } (\beta \cdot \mathbf{n}_{e_j}) > 0, \\ 0, & \text{else.} \end{cases} \tag{4.22}$$

We note that we use the Jacobians $\mathcal{H}_b(\mathbf{n}_e, a, b)$ with respect to the second argument b in formula (4.21). This is because we want to track the inflow coming through edge e. According to our definition, the normal $\mathbf{n}_e$ is pointing from a to b; see Figure 4.7. Thus, the Jacobian $\mathcal{H}_b(\mathbf{n}_e, a, b)$ with respect to b represents the inflow part on edge e. In addition, in formula (4.21), we divide by the total sum of the Jacobians to maintain the consistency of the scheme and fulfill a more general version of (4.17): For every outflow edge e_j, we obtain

$$\sum_{e_i \in F_h^E} \omega_{i,j} = 1. \tag{4.23}$$

Furthermore, we emphasize that the Jacobians $\mathcal{H}_b(\mathbf{n}_e, a, b)$ are evaluated using an arbitrary constant solution $a = b = u_c$. In the linear case, the choice of values to evaluate the Jacobian of the numerical flux function is not important, as it is constant in both arguments. However, we introduce the notation here to establish consistency with the non-linear formulation and to provide better comparability between the two formulations.

Finally, the stabilization term on a general cut cell E for the linear advection equation reads as

$$J_h^{0,E}(u^h, w^h) = \eta_E \sum_{e_j \in F_h^E} \int_{e_j} \sum_{e_i \in F_h^E} \omega_{i,j} (\mathcal{L}_{E_i}^{\text{ext}}(u^h) - u_E) \left(\beta \cdot \mathbf{n}_{e_j}\right) \left[\!\left[w^h \right]\!\right] \mathrm{d}s. \tag{4.24}$$

Next, we check if the general stabilization term given by equation (4.24) is consistent with the stabilization term, that we presented for the ramp geometry.

Remark 28. *We consider a small triangular cut cell of the ramp geometry for the case of advection along the ramp. We have one inflow edge* (e_1)*, one no-flow edge* (e_2) *and one outflow edge* (e_3)*, which results in the final weights*

$$\begin{array}{lll} \omega_{1,1} = 0 & \omega_{1,2} = 0 & \omega_{1,3} = 1 \\ \omega_{2,1} = 0 & \omega_{2,2} = 0 & \omega_{2,3} = 0 \\ \omega_{3,1} = 0 & \omega_{3,2} = 0 & \omega_{3,3} = 0. \end{array}$$

Thus, the stabilization term is given by

$$J_h^{0,E_{cut}}(u^h,w^h) = \eta_{E_{cut}} \int_{e_3} (\mathcal{L}_{E_1}^{ext}(u^h) - u_{E_{cut}})\,(\beta\cdot\mathbf{n})\,\left[\!\left[w^h\right]\!\right] ds,$$

which is consistent with the formulation we presented in the previous chapter.

Next, we will discuss the particular choice of weights for two examples with multiple inflow or outflow edges.

Example 1. *We start with the case of a triangular cell, which has one inflow* (e_1) *and two outflow* (e_2, e_3) *edges. According to the prerequisite we made in equation (4.18), we know that* $\omega_{i,1} = 0$*, for* $i = 1,2,3$*. Since* e_2 *and* e_3 *are outflow edges, we know that* $(\beta\cdot\mathbf{n}_{e_2/e_3})^\ominus = 0$*. Subsequently, following equation (4.21) we obtain the weights*

$$\begin{array}{lll} \omega_{1,1} = 0 & \omega_{1,2} = 1 & \omega_{1,3} = 1 \\ \omega_{2,1} = 0 & \omega_{2,2} = 0 & \omega_{2,3} = 0 \\ \omega_{3,1} = 0 & \omega_{3,2} = 0 & \omega_{3,3} = 0. \end{array}$$

Example 2. *Next, we consider the case of a triangular cell with two inflow* (e_1, e_2) *and one outflow edge* (e_3)*, as shown in Figure 4.7. The velocity field is given by* $\beta = (\beta_1, \beta_2)^T$ *with* $\beta_1, \beta_2 \geq 0$*. Furthermore, we assume that* $|e_1| = \alpha h$*,* $|e_2| = \delta h$ *and* $|e_3| = \sqrt{\alpha^2+\delta^2}h$*. The inflow portion from edges* e_1*,* e_2 *are given by* $\int_{e_1}(\beta\cdot\mathbf{n}_{e_1}) = \alpha h\beta_2$ *and* $\int_{e_2}(\beta\cdot\mathbf{n}_{e_2}) = \delta h\beta_1$*. Once again, equation (4.18) results in* $\omega_{i,1/2} = 0$ *for* $i \in \{1,2,3\}$*. The final weights are given by*

$$\begin{array}{lll} \omega_{1,1} = 0 & \omega_{1,2} = 0 & \omega_{1,3} = \dfrac{\alpha\beta_2}{\alpha\beta_2+\delta\beta_1} \\ \omega_{2,1} = 0 & \omega_{2,2} = 0 & \omega_{2,3} = \dfrac{\delta\beta_1}{\alpha\beta_2+\delta\beta_1} \\ \omega_{3,1} = 0 & \omega_{3,2} = 0 & \omega_{3,3} = 0. \end{array}$$

Theoretical results

For the artificial cut model problem and the proposed stabilization, there exist theoretical results for using piecewise constant polynomials.

Theorem 29 (Monotonicity)**.** *Consider the stabilized scheme for the artificial cut geometry using piecewise constant polynomials and explicit Euler in time for the linear advection equation 4.3. Additionally, choose the time step $\Delta t \leq \nu \frac{h}{\|\beta\|}$ with $0 < \nu \leq 0.5$. Then, the stabilized update on a small cut cell E_{cut} is monotone for $\eta_{E_{cut}} \in [1-\alpha_{cut}, 1]$.*

Proof. The idea of this proof is comparable to the idea of the proof for the two-dimensional Burgers equation. Since the setting is similar as we consider the second model problem in both cases, we will skip the proof here and refer to the proof of Theorem 31 later in this work. □

Besides monotonicity, we can also show L^2 stability.

Theorem 30. *[10, Theorem 3.1] Consider (4.3) with homogeneous boundary conditions. Assume that the discrete solution $u^h(t)$ vanishes on the boundary $\partial\Omega$ for all $t \in (0,T)$. Let $u_h(t) \in V_h^0$ be the solution to the semi-discrete problem (4.11). Then the solution is L^2 stable and satisfies*

$$||u^h(t)||_{L^2(\Omega)} \leq ||u^h(0)||_{L^2(\Omega)} \qquad \forall\, t \in (0,T).$$

Proof. The idea of the proof is similar to the one of Theorem 27 and can be found in detail in the collaborated work [10]. The proof in [10] is based on the fact that the weights $\omega_{i,j}$ fulfill the two conditions

$$\sum_{e_i \in F_h^E} \omega_{i,j} = 1 \tag{4.25}$$

and

$$\sum_{e_j \in F_h^E} \int_{e_j} \omega_{i,j} (\beta \cdot \mathbf{n}_e)^{\oplus} \mathrm{d}s = \int_{e_i} (\beta \cdot \mathbf{n}_e)^{\ominus} \mathrm{d}s. \tag{4.26}$$

While the first condition (4.25) is given by construction, the second condition (4.26) is a direct consequence of the divergence-free velocity field:

$$0 = \int_E \nabla \cdot \beta \mathrm{d}\mathbf{x} = \int_{\partial E} \mathbf{n} \cdot \beta \mathrm{d}s = \sum_{e_i F_h^E} \int_{e_i} (\mathbf{n}_{e_i} \cdot \beta)^{\oplus} - (\mathbf{n}_{e_i} \cdot \beta)^{\ominus} \mathrm{d}s$$

Thus, we obtain

$$\sum_{e_j \in F_h^E} \int_{e_j} \omega_{i,j} (\beta \cdot \mathbf{n}_e)^{\oplus} \mathrm{d}s = \sum_{e_j \in F_h^E} \int_{e_j} \frac{\int_{e_i} (\beta \cdot \mathbf{n}_{e_i})^{\ominus} \mathrm{d}s}{\sum_{e_k \in F_h^E} \int_{e_k} (\beta \cdot \mathbf{n}_{e_k})^{\ominus} \mathrm{d}s} \chi_j^{+} (\beta \cdot \mathbf{n}_e)^{\oplus} \mathrm{d}s$$
$$= \int_{e_i} (\beta \cdot \mathbf{n}_{e_i})^{\ominus} \mathrm{d}s \frac{\sum_{e_j \in F_h^E} \int_{e_j} (\beta \cdot \mathbf{n}_e)^{\oplus} \mathrm{d}s}{\sum_{e_k \in F_h^E} \int_{e_k} (\beta \cdot \mathbf{n}_{e_k})^{\ominus} \mathrm{d}s}$$
$$= \int_{e_i} (\beta \cdot \mathbf{n}_{e_i})^{\ominus} \mathrm{d}s.$$

Since we can show the two conditions for the choice of our weights, the remaining part of the proof is equivalent to that in [10]. □

4.3. Extension to non-linear systems of conservation laws

In this section, we present the first step towards extending the DoD stabilization to non-linear systems of conservation laws. First, we focus on Burgers equation for the artificial cut geometry, followed by preliminary results for the two-dimensional Euler equations on an anisotropic grid.

4.3.1. Scalar case for artificial cut grid

In the scalar case, we will show how to extend the DoD stabilization for the example of the inviscid, isotropic Burgers equation in two dimensions. Once again, we will consider the function space of piecewise constant polynomials V_h^0 and discuss the choice of the weights using the second model problem, which is given by an equidistant meshed unit square with one artificial cut through the domain. The model problem is shown in Figure 4.2b and for the case of linear advection in Figures 4.6 and 4.7. The inviscid, isotropic Burgers equation is given by

$$\begin{aligned} \partial_t u + \nabla \cdot \left(\frac{\mathbf{1} u^2}{2} \right) &= 0 && \text{in } \Omega \times (0,T), \\ u &= g && \text{on } \Gamma^{\text{in}} \times (0,T), \\ u &= u_0 && \text{on } \Omega \times \{t = 0\}, \end{aligned} \tag{4.27}$$

with $\mathbf{1} = (1,1)^\top \in \mathbb{R}^2$. The flux function is of the following form:

$$\mathbf{f}(u) = \left(\frac{1}{2} u^2, \frac{1}{2} u^2 \right)^\top$$

For a smooth solution u, we can rewrite equation (4.27) as

$$\partial_t u + u\partial_x u + u\partial_y u = 0. \tag{4.28}$$

The stabilized semi-discrete form is given by: Find $u^h \in V_h^0(\Omega)$ such that

$$\left(\mathrm{d}_t u^h(t), w^h\right)_{L^2(\Omega)} + a_h\left(u^h(t), w^h\right) + J_h\left(u^h(t), w^h\right) = 0 \quad \forall w^h \in V_h^0(\Omega) \tag{4.29}$$

with

$$a_h\left(u^h, w^h\right) = -\sum_{E\in\mathcal{M}_h}\int_E \mathbf{f}(u^h)\cdot w^h \mathrm{d}\mathbf{x} + \sum_{e\in\Gamma_h^{\text{int}}} \mathcal{H}(\mathbf{n}_e, u_E, u_{E'}) \left[\!\left[w^h\right]\!\right] \mathrm{d}s$$
$$+ \sum_{e\in\Gamma_h^{\text{ext}}}\int_e \mathcal{H}(\mathbf{n}_e, u_E, \widetilde{u}) w^h \mathrm{d}s.$$

For internal edges $e \in \Gamma_h^{\text{int}}$, connecting cells E and E', the unit normal vector $\mathbf{n}_e$ is pointing from cell E to cell E'. On a boundary edge $e \in \Gamma_h^{\text{ext}}$ that belongs to cell $E \in \mathcal{M}_h$, we define the boundary value $\widetilde{u}$ as

$$\widetilde{u}(\mathbf{x}) = \begin{cases} g(\mathbf{x}) & \forall\, \mathbf{x} \in \Gamma^{\text{in}}, \\ u_E(\mathbf{x}) & \text{else.} \end{cases}$$

The numerical flux will be computed using the local Lax-Friedrichs flux, which is given by

$$\mathcal{H}(\mathbf{n}_e, a, b) = \frac{1}{2}(\mathbf{1}\cdot\mathbf{n}_e)\left(\frac{1}{2}a^2 + \frac{1}{2}b^2\right) - \frac{1}{2}\left|\mathbf{1}\cdot\mathbf{n}_e\right|\lambda_{\max}(b-a) \tag{4.30}$$

and which can be rewritten as

$$\begin{aligned}\mathcal{H}(\mathbf{n}_e, a, b) =& \frac{1}{2}\left((\mathbf{1}\cdot\mathbf{n}_e)^{\oplus} - (\mathbf{1}\cdot\mathbf{n}_e)^{\ominus}\right)\left(\frac{1}{2}a^2 + \frac{1}{2}b^2\right) \\ & - \frac{1}{2}\left((\mathbf{1}\cdot\mathbf{n}_e)^{\oplus} + (\mathbf{1}\cdot\mathbf{n}_e)^{\ominus}\right)\lambda_{\max}(b-a)\end{aligned} \tag{4.31}$$

with $\lambda_{\max} = \max(|a|, |b|)$ and the negative and positive components as defined above.

Once again, since we consider piecewise constant polynomials, the stabilization term is given by $J_h(\cdot,\cdot) = J_h^0(\cdot,\cdot) = \sum_{E\in I} J_h^{0,E}(\cdot,\cdot)$. We choose the edge stabilization $J_h^{0,E}(\cdot,\cdot)$ for a cell E as follows:

$$J_h^{0,E}(u^h, w^h) = \eta_E \sum_{e_j\in F_h^E}\int_{e_j}\sum_{e_i\in F_h^E}\omega_{i,j}\left(\mathcal{H}(\mathbf{n}_{e_j}, \mathcal{L}_{E_i}^{\text{ext}}(u^h), u_{E_j}) - \mathcal{H}(\mathbf{n}_{e_j}, u_E, u_{E_j})\right)\left[\!\left[w^h\right]\!\right] \mathrm{d}s \tag{4.32}$$

The stabilization parameter η_E for the Burgers equation is chosen as

$$\eta_E = 1 - \min\left(\frac{1}{2p+1}\frac{|E|}{\nu h \sum_{\partial E}(\mathbf{1}\cdot\mathbf{n}_E)^{\ominus}\,\mathrm{d}s}, 1\right) \tag{4.33}$$

with ν being the CFL parameter and h being the length of a Cartesian cell.

Next, we want to define the weights for stabilization similar to the linear advection equation. Since we will use an arbitrary but constant solution u_c for the computation of the weights, we have made the following simplification: Instead of the exact Jacobians, we will use approximated Jacobians of the numerical flux, which are built assuming a constant maximum wave speed $\lambda_{\max}$. Thus, the approximated Jacobians are given by

$$\widetilde{\mathcal{H}}_a(\mathbf{n}_e, a, b) = \frac{1}{2}\left((\mathbf{1}\cdot\mathbf{n}_e)^{\oplus} - (\mathbf{1}\cdot\mathbf{n}_e)^{\ominus}\right)a + \frac{1}{2}\left((\mathbf{1}\cdot\mathbf{n}_e)^{\oplus} + (\mathbf{1}\cdot\mathbf{n}_e)^{\ominus}\right)\lambda_{\max} \tag{4.34}$$

and

$$\widetilde{\mathcal{H}}_b(\mathbf{n}_e, a, b) = \frac{1}{2}\left((\mathbf{1}\cdot\mathbf{n}_e)^{\oplus} - (\mathbf{1}\cdot\mathbf{n}_e)^{\ominus}\right)b - \frac{1}{2}\left((\mathbf{1}\cdot\mathbf{n}_e)^{\oplus} + (\mathbf{1}\cdot\mathbf{n}_e)^{\ominus}\right)\lambda_{\max}. \tag{4.35}$$

In the setting of non-linear scalar equations, we suggest the following formula for the weights with $u_c \in \mathbb{R}^+$ being again an arbitrary but constant value:

$$\omega_{i,j} = \frac{\int_{e_i}\widetilde{\mathcal{H}}_a(\mathbf{n}_{e_i}, u_c, u_c)\mathrm{d}s}{\sum_{e_k\in F_h^E}\int_{e_k}\widetilde{\mathcal{H}}_a(\mathbf{n}_{e_k}, u_c, u_c)\mathrm{d}s}\chi_j^- + \frac{\int_{e_i}\widetilde{\mathcal{H}}_b(\mathbf{n}_{e_i}, u_c, u_c)\mathrm{d}s}{\sum_{e_k\in F_h^E}\int_{e_k}\widetilde{\mathcal{H}}_b(\mathbf{n}_{e_k}, u_c, u_c)\mathrm{d}s}\chi_j^+ \tag{4.36}$$

The two indicator functions for the case of the two-dimensional Burgers equation are defined by

$$\chi_j^+ = \begin{cases} 1, & \text{if } (\mathbf{1}\cdot\mathbf{n}_{e_j}) > 0 \\ 0, & \text{else,} \end{cases} \quad \text{and} \quad \chi_j^- = \begin{cases} 1, & \text{if } (\mathbf{1}\cdot\mathbf{n}_{e_j}) < 0 \\ 0, & \text{else.} \end{cases} \tag{4.37}$$

For an arbitrary constant value $u_c \in \mathbb{R}^+$ the Jacobians reduce to

$$\widetilde{\mathcal{H}}_a(\mathbf{n}_e, u_c, u_c) = (\mathbf{1}\cdot\mathbf{n}_e)^{\oplus}u_c \quad \text{and} \quad \widetilde{\mathcal{H}}_b(\mathbf{n}_e, u_c, u_c) = -(\mathbf{1}\cdot\mathbf{n}_e)^{\ominus}u_c. \tag{4.38}$$

Thus, we rewrite (4.36) as

$$\omega_{i,j} = \frac{|e_i|(\mathbf{1}\cdot\mathbf{n}_{e_i})^{\oplus}}{\sum_{e_k\in F_h^E}|e_k|(\mathbf{1}\cdot\mathbf{n}_{e_k})^{\oplus}}\chi_j^- + \frac{|e_i|(\mathbf{1}\cdot\mathbf{n}_{e_i})^{\ominus}}{\sum_{e_k\in F_h^E}|e_k|(\mathbf{1}\cdot\mathbf{n}_{e_k})^{\ominus}}\chi_j^+. \tag{4.39}$$

We note that the weights given by (4.39) do not depend on the constant value u_c. This is because the constant value appears in both the numerator and denominator and can thus be factored out and eliminated. As we intended, this results in weights that only consider the direction of the fluxes and do not additionally scale the stabilization.

Comparing the weights for the Burgers equation (4.39) with the weights for the linear advection equation (4.21) reveals that for the Burgers equation, we now include all edges. This is because we do not know a priori which edge is an inflow or outflow edge. The decision regarding the edges depends on the solution and is decided at each time step. Furthermore, we are now considering the Lax-Friedrichs flux, which is a central flux. As a result, the numerical flux might consist of a part that flows in the upwind direction and another part that flows in the downwind direction. Thus, similar to one dimension, we need to introduce a more general formulation for the stabilization in the non-linear case. We propose a symmetric structure that considers all edges for the formula of the weights. This leads to adding a second term in equation (4.39), which measures the proportion of the outflow coming through edge e_i to the total outflow of cell E.

Finally, because of $\nabla \cdot \mathbf{1} = 0$, we have

$$0 = \int_E \nabla \cdot \mathbf{1}\, \mathrm{d}\mathbf{x} = \int_{\partial E} \mathbf{1} \cdot \mathbf{n}\, \mathrm{d}s = \int_{\partial E} (\mathbf{1} \cdot \mathbf{n})^{\oplus} - (\mathbf{1} \cdot \mathbf{n})^{\ominus} \mathrm{d}s = \sum_i |e_i| \left[(\mathbf{1} \cdot \mathbf{n}_{e_i})^{\oplus} - (\mathbf{1} \cdot \mathbf{n}_{e_i})^{\ominus} \right].$$

Thus, we can define $\mathcal{W} = \sum_i |e_i| (\mathbf{1} \cdot \mathbf{n}_{e_i})^{\oplus} = \sum_i |e_i| (\mathbf{1} \cdot \mathbf{n}_{e_i})^{\ominus}$ and rewrite the formula of the weights into

$$\omega_{i,j} = \frac{|e_i| (\mathbf{1} \cdot \mathbf{n}_{e_i})^{\ominus} \chi_j^{+} + |e_i| (\mathbf{1} \cdot \mathbf{n}_{e_i})^{\oplus} \chi_j^{-}}{\mathcal{W}}. \tag{4.40}$$

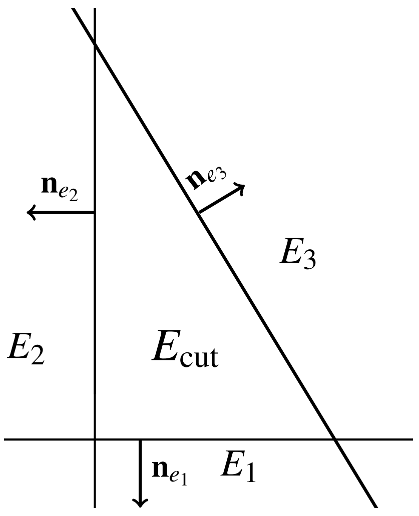


Figure 4.8: Example of a triangular cut cell for the artificial cut geometry.

Theorem 31 (Monotonicity). *We consider the artificial cut geometry given in Figure 4.2b for the two-dimensional Burgers equation using piecewise constant polynomials V_h^0. In addition, we consider the stabilized update using the DoD stabilization (4.32) with the explicit Euler method in time. As the numerical flux function, we use the local Lax-Friedrichs flux* *given by (4.31) and choose the time step according to $\Delta t \leq \nu \frac{h}{\lambda_{\max}}$ with $0 < \nu \leq 0.5$. Then, the stabilized update on a small triangular cut cell E_{cut} is monotone for $\eta_{E_{cut}} \in \left[1 - \frac{|E_{cut}|}{\nu h \sum_i |e_i|}, 1\right]$ with e_i being the edges of E_{cut}.*

Proof. The unstabilized update for a triangular cut cell E_{cut} with the notation of Figure 4.8 is given by

$$u^{n+1}_{E_{\text{cut}}} = u^n_{E_{\text{cut}}} - \frac{\Delta t}{|E_{\text{cut}}|}\left(\int_{e_1} \mathcal{H}(\mathbf{n}_{e_1}, u_{E_{\text{cut}}}, u_{E_1})\mathrm{d}s + \int_{e_2} \mathcal{H}(\mathbf{n}_2, u_{E_{\text{cut}}}, u_{E_2})\mathrm{d}s + \int_{e_3} \mathcal{H}(\mathbf{n}_3, u_{E_{\text{cut}}}, u_{E_3})\mathrm{d}s\right). \tag{4.41}$$

By adding the stabilization term $J_h^{0,E_{\text{cut}}}(\cdot,\cdot)$ given by (4.32), we obtain the following stabilized update formula

$$\begin{aligned} u^{n+1}_{E_{\text{cut}}} = u^n_{E_{\text{cut}}} - \frac{\Delta t}{|E_{\text{cut}}|}(1-\eta_{E_{\text{cut}}})\Bigg(\int_{e_1} \mathcal{H}(\mathbf{n}_{e_1}, u_{E_{\text{cut}}}, u_{E_1})\mathrm{d}s \\ + \int_{e_2} \mathcal{H}(\mathbf{n}_2, u_{E_{\text{cut}}}, u_{E_2})\mathrm{d}s + \int_{e_3} \mathcal{H}(\mathbf{n}_3, u_{E_{\text{cut}}}, u_{E_3})\mathrm{d}s\Bigg) + \Theta \end{aligned} \tag{4.42}$$

with

$$\begin{aligned} \Theta = -\frac{\Delta t}{|E_{\text{cut}}|}\eta_{E_{\text{cut}}}\Bigg(&\int_{e_1} \omega_{2,1}\mathcal{H}(\mathbf{n}_{e_1}, u_{E_2}, u_{E_1}) + \omega_{3,1}\mathcal{H}(\mathbf{n}_{e_1}, u_{E_3}, u_{E_1})\mathrm{d}s \\ &+ \int_{e_2} \omega_{1,2}\mathcal{H}(\mathbf{n}_{e_2}, u_{E_1}, u_{E_2}) + \omega_{3,2}\mathcal{H}(\mathbf{n}_{e_2}, u_{E_3}, u_{E_2})\mathrm{d}s \\ &+ \int_{e_3} \omega_{1,3}\mathcal{H}(\mathbf{n}_{e_3}, u_{E_1}, u_{E_3}) + \omega_{2,3}\mathcal{H}(\mathbf{n}_{e_3}, u_{E_2}, u_{E_3})\mathrm{d}s\Bigg). \end{aligned} \tag{4.43}$$

In equation (4.42), we have used that $\sum_{e_i \in F_h^{E_{\text{cut}}}} \omega_{i,j} = 1,\ \forall e_j \in \partial E_{\text{cut}}$. Next, we show that Θ vanishes for our choice of weights. For two neighbors of the cut cell E_i and E_j, there holds $\omega_{i,j} = \omega_{j,i} = 0$ if $(\mathbf{1}\cdot\mathbf{n}_{e_i})$ and $(\mathbf{1}\cdot\mathbf{n}_{e_j})$ have the same sign. If they have mixed signs and we assume without loss of generality that $(\mathbf{1}\cdot\mathbf{n}_{e_i}) > 0$ and $(\mathbf{1}\cdot\mathbf{n}_{e_j}) < 0$, we can show the following identity

$$\begin{aligned} &\int_{e_j} \omega_{i,j}\mathcal{H}(\mathbf{n}_{e_j}, u_{E_i}, u_{E_j})\mathrm{d}s \\ &\quad = |e_j|\frac{|e_i|(\mathbf{1}\cdot\mathbf{n}_{e_i})^{\oplus}}{\mathcal{W}}\frac{1}{2}(\mathbf{1}\cdot\mathbf{n}_{e_j})^{\ominus}\left(-\left(\frac{1}{2}(u_{E_i})^2 + \frac{1}{2}(u_{E_j})^2\right) - \lambda_{\max}(u_{E_j} - u_{E_i})\right) \\ &\quad = -|e_i|\frac{|e_j|(\mathbf{1}\cdot\mathbf{n}_{e_j})^{\ominus}}{\mathcal{W}}\frac{1}{2}(\mathbf{1}\cdot\mathbf{n}_{e_i})^{\oplus}\left(\left(\frac{1}{2}(u_{E_i})^2 + \frac{1}{2}(u_{E_j})^2\right) - \lambda_{\max}(u_{E_i} - u_{E_j})\right) \\ &\quad = -\int_{e_i} \omega_{j,i}\mathcal{H}(\mathbf{n}_{e_i}, u_{E_j}, u_{E_i})\mathrm{d}s. \end{aligned} \tag{4.44}$$

As a result, the weights in equation (4.43) are either zero or chosen such that the terms cancel out. Consequently, we have $\Theta = 0$, and the stabilized update formula is given by the standard update with a scaled flux difference

$$\begin{aligned} u_{E_{\text{cut}}}^{n+1} = u_{E_{\text{cut}}}^{n} - \frac{\Delta t}{|E_{\text{cut}}|}(1-\eta_{E_{\text{cut}}})\bigg(\int_{e_1} \mathcal{H}(\mathbf{n}_{e_1}, u_{E_{\text{cut}}}, u_{E_1})\mathrm{d}s \\ + \int_{e_2} \mathcal{H}(\mathbf{n}_{e_2}, u_{E_{\text{cut}}}, u_{E_2})\mathrm{d}s + \int_{e_3} \mathcal{H}(\mathbf{n}_{e_3}, u_{E_{\text{cut}}}, u_{E_3})\mathrm{d}s\bigg). \end{aligned} \tag{4.45}$$

Finally, we will check the partial derivatives of the update formula according to Definition 8. For the partial derivatives, that depend on the neighbors, we obtain

$$\frac{\partial}{\partial u_{E_i}} u_{E_{\text{cut}}}^{n+1} = -\frac{\Delta t}{|E_{\text{cut}}|}(1-\eta_{E_{\text{cut}}})\int_{e_i} \mathcal{H}_b(\mathbf{n}_{e_i}, u_{E_{\text{cut}}}, u_{E_i})\mathrm{d}s \geq 0,$$

due to the fact that $\mathcal{H}(\mathbf{n}_{e_i}, \cdot, \cdot)$ is a non-increasing function of its second argument. For the partial derivative with respect to $u_{E_{\text{cut}}}$, there holds

$$\begin{aligned} \frac{\partial}{\partial u_{E_{\text{cut}}}} u_{E_{\text{cut}}}^{n+1} &= 1 - \frac{\Delta t}{|E_{\text{cut}}|}(1-\eta_{E_{\text{cut}}})\sum_i \int_{e_i} \mathcal{H}_a(\mathbf{n}_{e_i}, u_{E_{\text{cut}}}, u_{E_i})\mathrm{d}s \\ &\geq 1 - \frac{\Delta t}{|E_{\text{cut}}|}(1-\eta_{E_{\text{cut}}})\sum_i |e_i|\lambda_{\max} \\ &\geq 0. \end{aligned}$$

The last inequality holds due to the choice of the stabilization parameter $\eta_{E_{\text{cut}}}$. □

Numerical results

In this section, we perform numerical tests for the isotropic Burgers equation in two dimensions, given by (4.27). We present a convergence test for a smooth function and a stability test for a two-dimensional Riemann problem. For both tests, we consider the second model problem (Figure 4.2b) and use piecewise constant polynomials in space in combination with the explicit Euler scheme in time.

Convergence test

For the convergence test, we consider a smooth Gaussian curve, with the initial data being given by

$$u_0(x,y) = \exp(-20((x-x_c)^2 + (y-y_c)^2)). \tag{4.46}$$

As stated above, the computational domain is $\Omega = [0,1]^2$. At the boundary, we choose transmissive boundary conditions. Similar to the one-dimensional sine test case for the Burgers equation, we find that the Gaussian curve undergoes the process of wave breaking

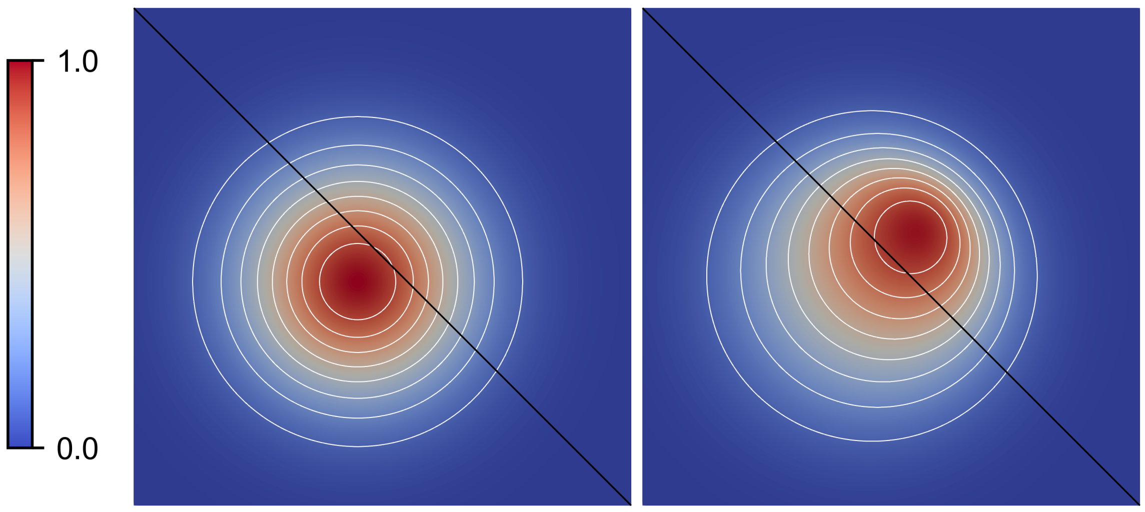


Figure 4.9: Setup of the smooth Gaussian curve used for the convergence test at the initial time $T = 0$ (left) and the final time $T = 0.1$ (right). The black line highlights the artificial cut, and the white circles indicate isolines of the numerical solution.

and shock formation. As time evolves, steep gradients form and, finally, a discontinuity in the form of a shock occurs. Since we are interested in the convergence behavior of the stabilized method, we compare the numerical solution with the exact solution when the curve is still smooth. For this particular test case, we choose the final time $T = 0.1$, which we estimated experimentally. In the resulting time interval $[0,T]$, we use Newton's method to numerically solve the non-linear equation $u = u_0(x-ut, y-ut)$ to examine the exact solution u. In Figure 4.9, we show the initial data and the numerical solution at the final time $T = 0.1$ for a grid consisting of 200×200 cells. The artificial cut is defined by the angle $\gamma = 135°$ and the starting point $x_0 = 0.9999$. We choose the time step according to $\Delta t \leq \nu \frac{h}{\lambda_{\max}}$ with h being the length of a Cartesian cell and $\nu = 0.4$. Here, we define the set of stabilized cells as $I = \{E \mid |E| < 0.4h^2\}$. We note that the gradient at the final time is much steeper in the upper right part of the domain, compared to the gradient of the initial data. However, the numerical solution is smooth and no shock is formed so far.

The convergence study for the Gaussian curve is shown in Table 4.1. We show the L^1 and L^∞ errors for different setups of the artificial cut. These different setups result in both cut cells with multiple inflow edges and cut cells with multiple outflow edges. We choose the starting point x_0 and the angle γ such that the artificial cuts lie in the region of the steep gradient ($\gamma = 125°$) and in the region of the moderate gradient ($\gamma = 145°$). A third setup ($\gamma = 135°$) is chosen such that the peak of the Gaussian curve passes through the cut. For all cases, we see optimal convergence in both error norms.

Setup x_0 / γ	Background grid	L^1 error	order	L^∞ error	order
0.9999 / 125°	40×40	9.31e-03	–	1.75e-01	–
	80×80	4.64e-03	1.00	8.96e-02	0.96
	160×160	2.33e-03	0.99	4.69e-02	0.93
	320×320	1.16e-03	1.00	2.36e-02	0.98
	640×640	5.84e-04	0.99	1.19e-02	0.98
0.9999 / 135°	40×40	9.37e-03	–	1.74e-01	–
	80×80	4.66e-03	1.00	8.98e-02	0.96
	160×160	2.33e-03	0.99	4.70e-02	0.93
	320×320	1.16e-03	1.00	2.37e-02	0.98
	640×640	5.84e-04	0.99	1.19e-02	0.98
0.9999 / 145°	40×40	9.34e-03	–	1.74e-01	–
	80×80	4.65e-03	1.00	8.94e-02	0.95
	160×160	2.33e-03	0.99	4.70e-02	0.92
	320×320	1.16e-03	1.00	2.36e-02	0.98
	640×640	5.84e-04	0.99	1.19e-02	0.98

Table 4.1: Convergence tests for different cut setups

Stability test

For the stability test, we use the following two-dimensional Riemann problem [38] as initial data:

$$u_0(x,y) = \begin{cases} 0.5, & \text{if } x < 0.5 \text{ and } y < 0.5, \\ 0.8, & \text{if } x > 0.5 \text{ and } y < 0.5, \\ -0.2, & \text{if } x < 0.5 \text{ and } y > 0.5, \\ -1, & \text{if } x > 0.5 \text{ and } y > 0.5 \end{cases} \tag{4.47}$$

The values at the boundary are defined using the exact solution to this Riemann problem, which can be found in [38].

In Figure 4.10, we show the solution on a 200×200 grid at the final time $T = 0.5$ for two different cut setups. The artificial cut in the left plot is defined by the angle $\gamma = 135°$ and the starting point $x_0 = 0.9999$, which results in a cut from the lower right to the upper left corner. The resulting cut cells that need to be stabilized in this setup are all of the same sizes and have a volume fraction of $\frac{\text{Vol}_{\text{cut}}}{\text{Vol}_{\text{cart}}} \sim 2 \cdot 10^{-4}$ when compared to the Cartesian cells.

In the right plot of Figure 4.10, the artificial cut is constructed via the angle $\gamma = 155°$ and the starting point $x_0 = 0.9999$. In this setup, the cut is less steep, and we observe a range of different cut cells that need to be stabilized. The cell volumes in this test vary strongly and are in the interval $|E| \in [4.89 \cdot 10^{-12}, 2.5 \cdot 10^{-5}]$.

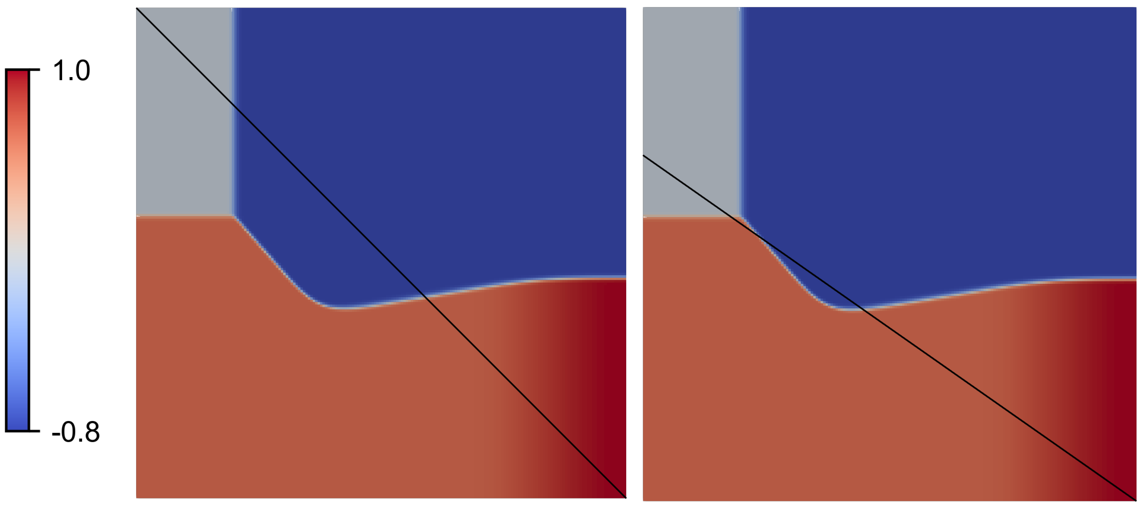


Figure 4.10: Stability test: Two-dimensional Riemann problem at final time $T = 0.5$ for different cut setups. The black line highlights the artificial cut.

In all our numerical tests, we observe stable behavior and have never detected over- or undershoots, even in the presence of shock waves. These results support our theoretical findings.

4.3.2. Preliminary results for system case on an anisotropic grid

Finally, we are interested in extending the DoD stabilization to non-linear systems of hyperbolic conservation laws in two dimensions. In this section, we will consider the two-dimensional Euler equations given by

$$\partial_t \mathbf{u} + \partial_x \mathbf{f}^1(\mathbf{u}) + \partial_y \mathbf{f}^2(\mathbf{u}) = \mathbf{0} \tag{4.48}$$

with

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \quad \mathbf{f}^1(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ u(E+p) \end{bmatrix}, \quad \mathbf{f}^2(\mathbf{u}) = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}. \tag{4.49}$$

For further details regarding the Euler equations, see section 2.1.2.

We consider the third model problem, which is given by the anisotropic grid shown in Figure 4.11. For systems of hyperbolic conservation laws, the search for adequate weights $\omega_{i,j}$ is more advanced compared to the scalar case. In a recent work [9, 11], the weights for two-dimensional linear systems were presented. In contrast to the scalar case, the weights for systems of equations are $m \times m$ matrices, with m being the number of equations. The definition of adequate weights for the two-dimensional Euler equations is an active field of research and has not been solved yet.

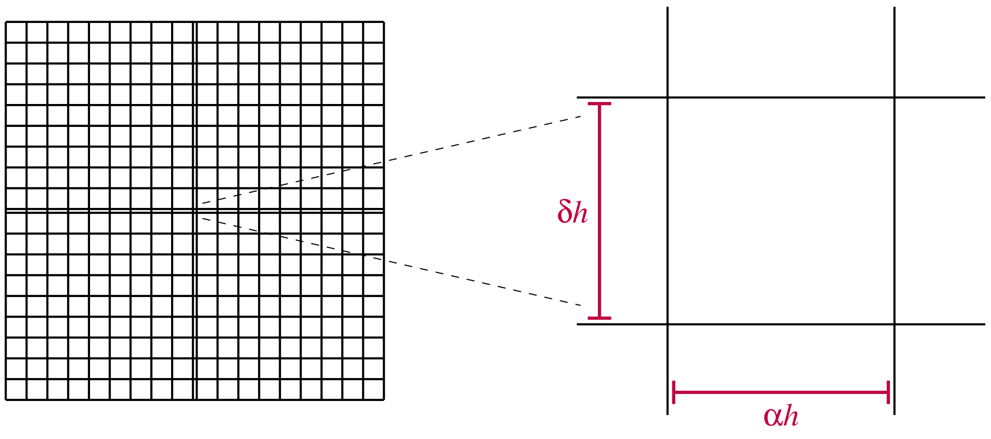


Figure 4.11: Left: Third model problem given by an anisotropic grid. Right: Zoom into small cell.

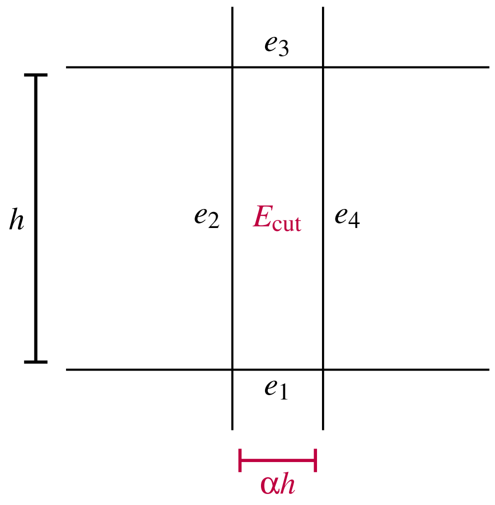


Figure 4.12: Zoom into example of small cell E_{cut} which is of size $\alpha h \times h$.

For the given model problem, the edges of each cell are parallel either to the x-axis or the y-axis. We exploit this geometry and apply the DoD stabilization using a dimensionally splitting approach. For each dimension, we use a variant of the one-dimensional stabilization. The resulting scheme only couples edges in x-direction with edges in x-direction and edges in y-direction with edges in y-direction, respectively. In other words, we couple edges that are parallel to each other. In addition, we only couple edges if the distance between these edges is smaller than a certain threshold. For the anisotropic grid, this implies that small cells of the column with a length of αh are stabilized only in x-direction, while cells of the row with height δh are stabilized only in y-direction. The only exception is the small cut cell in the center of the grid, which is small in both directions and hence is stabilized in both directions separately.

For the definition of the stabilization terms, we focus on one particular example, which is part of the small column and is shown in Figure 4.12. The anisotropic cell E_{cut} in Figure 4.12

is of size $\alpha h \times h$. Thus, we apply the stabilization in x-direction and couple the edges e_2 and e_4, but do not stabilize in y-direction. For this particular setup, the stabilization is given by

$$J_h^{E_{\text{cut}}}(\mathbf{u}^h, \mathbf{w}^h) = J_h^{0,E_{\text{cut}}}(\mathbf{u}^h, \mathbf{w}^h) + J_h^{1,E_{\text{cut}}}(\mathbf{u}^h, \mathbf{w}^h)$$

with the edge stabilization

$$\begin{aligned} J_h^{0,E_{\text{cut}}}(\mathbf{u}^h, \mathbf{w}^h) = \eta_{E_{\text{cut}}} \int_{e_4} \Big(\mathcal{H}(\mathbf{n}_x, \mathcal{L}_{E_2}^{\text{ext}}(\mathbf{u}^h), \mathbf{u}_{E_4}) - \mathcal{H}(\mathbf{n}_x, \mathbf{u}_{E_{\text{cut}}}, \mathbf{u}_{E_4}) \Big) \cdot \Big[\!\!\Big[\mathbf{w}^h \Big]\!\!\Big] \, \mathrm{d}s \\ + \eta_{E_{\text{cut}}} \int_{e_2} \Big(\mathcal{H}(-\mathbf{n}_x, \mathcal{L}_{E_4}^{\text{ext}}(\mathbf{u}^h), \mathbf{u}_{E_2}) - \mathcal{H}(-\mathbf{n}_x, \mathbf{u}_{E_{\text{cut}}}, \mathbf{u}_{E_2}) \Big) \cdot \Big[\!\!\Big[\mathbf{w}^h \Big]\!\!\Big] \, \mathrm{d}s \end{aligned}$$

and the volume stabilization using the normal vector $\mathbf{n}_x = (1,0)^T$

$$\begin{aligned} J_h^{1,E_{\text{cut}}} = & \eta_{E_{\text{cut}}} \sum_{j \in \{2,\text{cut},4\}} \mathbf{K}(j) \int_{E_{\text{cut}}} \Big(\mathcal{H}(\mathbf{n}_x, \mathcal{L}_{E_2}^{\text{ext}}(\mathbf{u}^h), \mathcal{L}_{E_4}^{\text{ext}}(\mathbf{u}^h)) - \mathbf{f}(\mathcal{L}_{E_j}^{\text{ext}}(\mathbf{u}^h)) \Big) \cdot \mathcal{L}_{E_j}^{\text{ext}}(\nabla \mathbf{w}^h) \, \mathrm{d}\mathbf{x} \\ & + \eta_{E_{\text{cut}}} \sum_{j \in \{2,\text{cut},4\}} \mathbf{K}(j) \int_{E_{\text{cut}}} \Big(\mathcal{H}_a(\mathbf{n}_x, \mathcal{L}_{E_2}^{\text{ext}}(\mathbf{u}^h), \mathcal{L}_{E_4}^{\text{ext}}(\mathbf{u}^h)) \mathcal{L}_{E_j}^{\text{ext}}(\mathbf{u}^h) \Big) \cdot \mathcal{L}_{E_2}^{\text{ext}}(\nabla \mathbf{w}^h) \, \mathrm{d}\mathbf{x} \\ & + \eta_{E_{\text{cut}}} \sum_{j \in \{2,\text{cut},4\}} \mathbf{K}(j) \int_{E_{\text{cut}}} \Big(\mathcal{H}_b(\mathbf{n}_x, \mathcal{L}_{E_2}^{\text{ext}}(\mathbf{u}^h), \mathcal{L}_{E_4}^{\text{ext}}(\mathbf{u}^h)) \mathcal{L}_{E_j}^{\text{ext}}(\mathbf{u}^h) \Big) \cdot \mathcal{L}_{E_4}^{\text{ext}}(\nabla \mathbf{w}^h) \, \mathrm{d}\mathbf{x}. \end{aligned}$$

We emphasize that the stabilization terms $J_h^{0,E_{\text{cut}}}$ and $J_h^{1,E_{\text{cut}}}$ are the straightforward two-dimensional extensions of the one-dimensional stabilization terms defined in (3.34) and (3.38). Thus, we use here the one-dimensional stabilization parameter $\eta_{E_{\text{cut}}} = 1 - \frac{\alpha}{\nu}$. Furthermore, we refer to Subsection 3.2.3 for the one-dimensional definitions of the matrices $\mathbf{K}(\cdot)$. The two-dimensional equivalents of $\mathbf{K}(\cdot)$ can be calculated in a straightforward way using a suitable average of $\mathcal{L}_{E_2}^{\text{ext}}(\mathbf{u}^h)$ and $\mathcal{L}_{E_4}^{\text{ext}}(\mathbf{u}^h)$. The formulation for the DoD stabilization presented in this case is explicitly given for the case of a small cell in the x-direction. However, it is important to note that this formulation can be extended analogously to the case of a small cell in the y-direction. In addition, the rotational invariance of the Euler equations (see 2.52) allows for the use of the same stabilization formulation regardless of the orientation of the small cell.

The presented approach runs stable during the simulation and shows promising results for the current geometry. To demonstrate the effectiveness of this method, we present some preliminary numerical results in the following section.

Numerical results

We present a convergence study for a smooth vortex for the two-dimensional Euler equations, which will be rotated and advected diagonally in x- and y-directions [28, 72]. We consider the computational domain $\Omega = [0,10]^2$ with Dirichlet boundary conditions. The

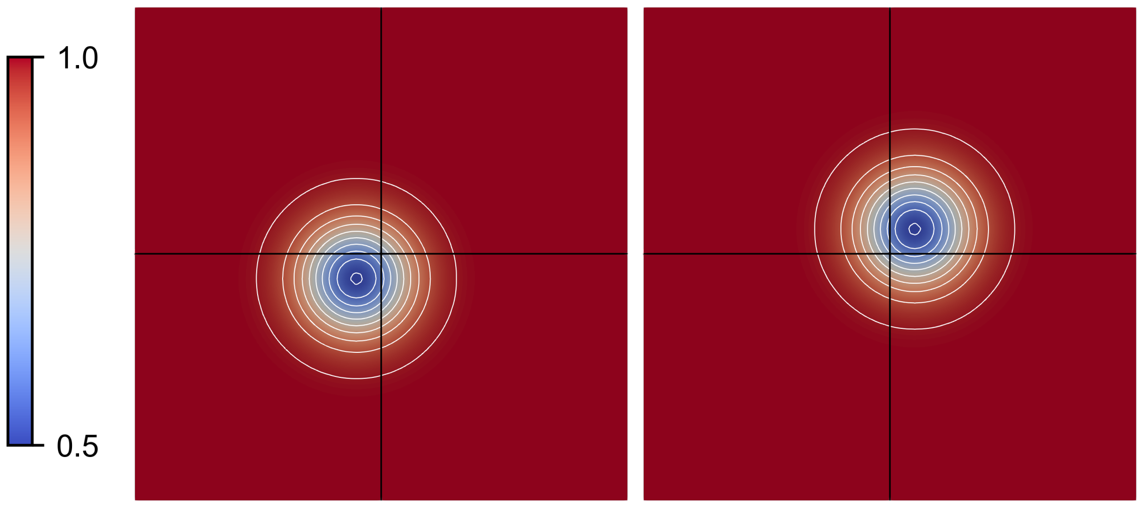


Figure 4.13: Setup of two-dimensional Euler equations for the vortex advection problem on an anisotropic grid. Solution of the density ρ at the initial time $T = 0$ (left) and the final time $T = 1$ (right). The black lines highlight the anisotropic row and column and the white circles indicate isolines of the numerical solution

center of the initial vortex is given by $(x_c, y_c) = (4.5, 4.5)$, and the radius is set to $r_c = 1$. The initial condition is defined as

$$u = 1 - (y - y_c)\phi(r), \quad v = 1 + (x - x_c)\phi(r), \quad \theta = 1 - \frac{\gamma - 1}{2\gamma}\phi(r)^2, \quad s = 0, \tag{4.50}$$

where $r = \sqrt{(x-x_c)^2 + (y-y_c)^2}$, $\phi(r) = \varepsilon \exp\left(\alpha\left(1 - \left(\frac{r}{r_c}\right)^2\right)\right)$, $\varepsilon = \frac{5}{2\pi}$, $\alpha = 12$, $\theta = \frac{p}{\rho}$ and $s = \log(p) - \gamma\log(\rho)$. The exact solution is known in this setup and can be explicitly computed by

$$\mathbf{u}(x,y,t) = \mathbf{u}(x-t, y-t, 0) = \mathbf{u}_0(x-t, y-t). \tag{4.51}$$

In Figure 4.13, we show the initial data and the numerical solution at the final time $T = 1$ using piecewise quadratic polynomials. The computational domain in Figure 4.13 is given by a 100×100 grid with one column of width αh and one row of height δh. We choose $\alpha = 10^{-2}$ and $\delta = 10^{-2}$, which results in volume fractions $\frac{\text{Vol}_{\text{cut}}}{\text{Vol}_{\text{cart}}}$ between 10^{-2} and 10^{-4}.

In Figure 4.14, we present the convergence results for the given setup in both L^1 and L^∞ norms for the function space V_h^p, $p = 0, 1, 2, 3$. In both error norms, we observe for polynomials of degree p convergence orders close to $p+1$.

The numerical results presented here are only preliminary and should be interpreted with caution. While they provide initial insights into the stabilization and look promising, further research is needed to investigate these findings in more detail. We have also run tests for smaller values of α and δ and noted convergence issues. Although the tests ran stable and we solved the small cell problem, we observed that the convergence results did not improve beyond a certain point for smaller values of α and δ. We currently believe that

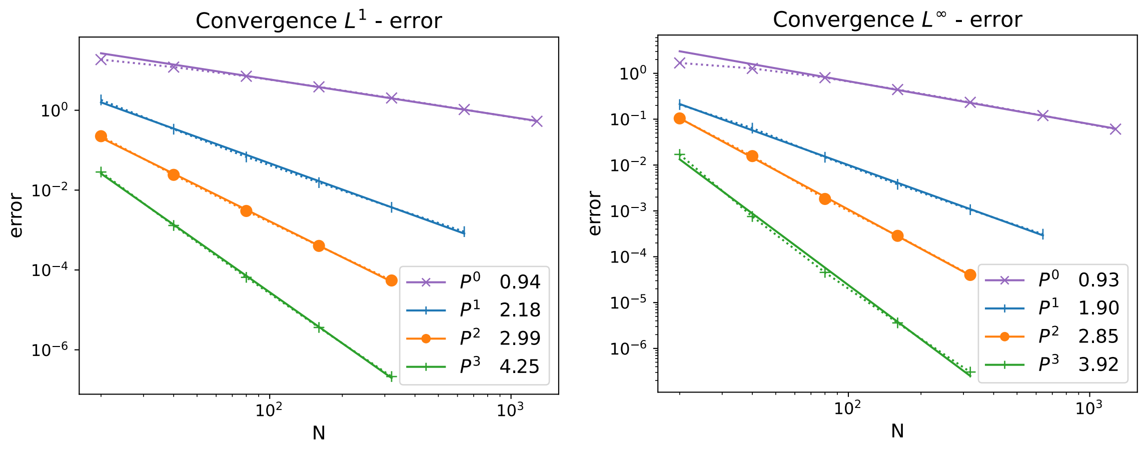


Figure 4.14: Convergence orders in L^1 and L^∞ norm for the error at time $T = 1.0$ for the anisotropic grid and different polynomial degrees $p = 0,1,2,3$.

this is due to certain implementation choices, but this must be further investigated. Our initial investigations suggest that these issues may be caused by round-off errors. Therefore, these results should be considered as a starting point for future studies rather than definitive conclusions.

5 Conclusions and Outlook

In this work, we have presented a novel penalty stabilization technique to solve hyperbolic conservation laws numerically on grids with embedded objects. The *Domain of Dependence* (DoD) stabilization is added to the semi-discrete scheme in the neighborhood of small cut cells. If combined with an explicit time-stepping scheme, the resulting method allows choosing the time step based on the size of the Cartesian background cells.

After introducing the mathematical basics for the discretization of hyperbolic conservation laws, we analyzed in Chapter 2 the small cell problem in more detail. We have seen that the unstabilized scheme on small cut cells leads to an uncontrollable update of the physical quantity, and the outflow neighbors of small cut cells lack information on their update. Based on these observations, we presented an algebraic solution to the small cell problem in the following chapters.

In Chapter 3, we presented the DoD stabilization for the one-dimensional case. We started with linear problems for the case of piecewise constant polynomials. The DoD stabilization redistributes the mass in the neighborhood of small cut cells in a *monotone* way. We compared the stabilization to the well-known h-box method and noticed that the two methods coincide in the case of piecewise constant polynomials in one dimension. Consequently, we could adopt a result from the h-box method showing that the method is first-order accurate for piecewise constant polynomials.

We extended the DoD stabilization to higher-order polynomials and non-linear systems of hyperbolic conservation laws. For the extension to higher-order polynomials, we introduced a volume stabilization term and an extension operator that evaluates functions from the function space V_h^p outside of their original domain. For the extension to non-linear problems, we needed to consider changing flow directions. Thus, we made use of Riemann solvers in the formulation of the stabilization terms. The semi-discrete stabilized formulation is L^2

stable for arbitrary polynomial degrees p. Furthermore, our numerical results show the same order of accuracy as standard RKDG schemes on equidistant meshes and are robust in the presence of shocks if a limiter is added.

In Chapter 4, we extended the DoD stabilization to the two-dimensional setting for different model problems. We started with the case of advection along a ramp and presented the DoD stabilization for arbitrary polynomial degrees. The ideas behind the stabilization for this case are quite comparable to the one-dimensional case. In consequence, we were able to extend the theoretical results regarding monotonicity and L^2 stability to two dimensions as well.

For the more general setting of a cut cell with multiple in- and outflow edges, we introduced weights to physically match in- and outflow edges. Furthermore, we presented weights for an example of a non-linear scalar equation, the two-dimensional Burgers equation. Both methods result in provable monotone updates on the small cut cells, and we showed stable numerical results and convergence studies. Finally, we presented preliminary results for the two-dimensional Euler equations on an anisotropic grid using a dimensionally splitting approach.

There are several interesting topics to consider for future research work. While the one-dimensional case has been thoroughly studied, there are still many open questions in two dimensions.

The first step will be the definition of weights for solving linear problems on general cut cells with multiple in- and outflow edges using higher-order polynomials. The L^2 stability proof will serve again as a helpful guideline in finding these weights.

Another goal for future work will be the search for suitable weights for the two-dimensional Euler equations on general cut cell grids. This task will be challenging, but we are confident that an essential first step has been taken by finding weights for the acoustic wave equation. Furthermore, it would be interesting to show that the DoD stabilization leads not only to an L^2 stable scheme but also to an entropy stable scheme in the presence of cut cells, particularly for the Euler equations. However, entropy stability has yet to be investigated in the context of cut cells. An excellent blueprint to show this could first deal with the one-dimensional case and subsequently address higher dimensions.

The development of an accurate limiter for higher-order polynomials on cut cell meshes remains an active research topic and an open problem. For the development of an effective limiting method on complex geometries using cut cell grids, it is necessary to address the challenge of limiting higher-order polynomials on irregularly shaped cut cells.

Bibliography

[1] T. J. Barth and D. Jespersen. The design and application of upwind schemes on unstructured meshes. *AIAA*, 1989.

[2] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn , M. Ohlberger, and O. Sander. A generic grid interface for parallel and adaptive scientific computing. Part I: Abstract framework. *Computing*, 82(2–3):103–119, 2008.

[3] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn , R. Kornhuber, M. Ohlberger, and O. Sander. A generic grid interface for parallel and adaptive scientific computing. Part II: Implementation and tests in DUNE. *Computing*, 82(2–3):121–138, 2008.

[4] P. Bastian and C. Engwer. An unfitted finite element method using discontinuous Galerkin. *Internat. J. Numer. Methods Engrg.*, 79:1557–1576, 2009.

[5] P. Bastian, F. Heimann, and S. Marnach. Generic implementation of finite element methods in the distributed and unified numerics environment (DUNE). *Kybernetika*, 46(2):294–315, 2010.

[6] M. Berger and A. Giuliani. A state redistribution algorithm for finite volume schemes on cut cell meshes. *J. Comput. Phys.*, 428, March 2021.

[7] M. Berger and C. Helzel. A simplified h-box method for embedded boundary grids. *SIAM J. Sci. Comput.*, 34(2):A861–A888, 2012.

[8] M. Berger, C. Helzel, and R. J. LeVeque. H-Box method for the approximation of hyperbolic conservation laws on irregular grids. *SIAM J. Numer. Anal.*, 41(3):893–918, 2003.

[9] G. Birke. Stabilization of hyperbolic transport and acoustic equations on general cut-cell meshes. Master's thesis, University of Münster, 2023.

[10] G. Birke, C. Engwer, S. May, and F. Streitbürger. DoD stabilization of linear hyperbolic pdes on general cut-cell meshes. In Ch. Böhm, K. Mang, B. Markert, S. Reese, M. Schmidtchen, J. Waimann, and M. Kaliske, editors, *Special Issue: 92nd Annual Meeting of the International Association of Applied Mathematics and Mechanics (GAMM)*. Wiley-VCH, 2023.

[11] G. Birke, C. Engwer, S. May, and F. Streitbürger. Domain of Dependence stabilization for the acoustic wave equation on 2D cut-cell meshes. arXiv:2304.04323, 2023.

[12] E. Burman. Ghost penalty. *C. R. Math.*, 348(21):1217 – 1220, 2010.

[13] E. Burman, S. Claus, P. Hansbo, M. G. Larson, and A. Massing. CutFEM: Discretizing geometry and partial differential equations. *Intern. J. Numer. Methods Engrg.*, 104:472–501, 2015.

[14] N. Chalmers and L. Krivodonova. A robust CFL condition for the discontinuous Galerkin method on triangular meshes. *J. Comput. Phys.*, 403:109095, 2020.

[15] I.-L. Chern and P. Colella. A conservative front tracking method for hyperbolic conservation laws. Technical report, Lawrence Livermore National Laboratory, Livermore, CA, 1987. Preprint UCRL-97200.

[16] B. Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework. *Math. Comp.*, 52(186):411–435, 1989.

[17] B. Cockburn and C.-W. Shu. Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. *SIAM J. Sci. Comput.*, 16(3):173–261, 2001.

[18] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano. A Cartesian grid embedded boundary method for hyperbolic conservation laws. *J. Comput. Phys.*, 211(1):347–366, 2006.

[19] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Math. Ann.*, 100(1):32–74, 1928.

[20] C. M. Dafermos. *Hyperbolic conservation laws in continuum physics*. Springer, Heidelberg; New York, 2010.

[21] D. Di Pietro and A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*. Springer New York, 2012.

[22] V. Dolejší and M. Feistauer. *Discontinuous Galerkin Method*. Springer Cham, 2015.

[23] J. Douglas and T. Dupont. Interior penalty procedures for elliptic and parabolic Galerkin methods. In R. Glowinski and J. L. Lions, editors, *Computing Methods in Applied Sciences*, pages 207–216, Berlin, Heidelberg, 1976. Springer Berlin Heidelberg.

[24] C. Engwer and F. Heimann. Dune-udg: a cut-cell framework for unfitted discontinuous Galerkin methods. In *Advances in DUNE*, pages 89–100. Springer, 2012.

[25] C. Engwer, S. May, A. Nüßing, and F. Streitbürger. A stabilized dG cut cell method for discretizing the linear transport equation. *SIAM J. Sci. Comput.*, 42(6):A3677–A3703, 2020.

[26] C. Engwer and A. Nüßing. Geometric reconstruction of implicitly defined surfaces and domains with topological guarantees. *ACM Trans. Math. Software (TOMS)*, 44(2):14, 2017.

[27] T. Eymann and P. Roe. Active flux schemes. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. AIAA, 2011.

[28] U. S. Fjordholm, S. Mishra, and E. Tadmor. Arbitrarily high-order accurate entropy stable essentially nonoscillatory schemes for systems of conservation laws. *SIAM J. Numer. Anal.*, 50(2):544–573, 2012.

[29] P. Fu, T. Frachon, G. Kreiss, and S. Zahedi. High order discontinuous cut finite element methods for linear hyperbolic conservation laws with an interface. *J. Sci. Comput.*, 90(3):84, 2022.

[30] P. Fu and G. Kreiss. High order cut discontinuous Galerkin methods for hyperbolic conservation laws in one space dimension. *SIAM J. Sci. Comput.*, 43(4):A2404–A2424, 2021.

[31] A. Giuliani. A two-dimensional stabilized discontinuous Galerkin method on curvilinear embedded boundary grids. *SIAM J. Sci. Comput.*, 44(1):A389–A415, 2022.

[32] A. Giuliani, A. S. Almgren, J. B. Bell, M. Berger, M. T. Henry de Frahan, and D. Rangarajan. A weighted state redistribution algorithm for embedded boundary grids. *J. Comput. Phys.*, 464:111305, 2022.

[33] E. Godlewski and P. A. Raviart. *Numerical Approximation of Hyperbolic Systems of Conservation Laws*. Number Nr. 118 in Appl. Math. Sci. Springer, 1996.

[34] N. Gokhale, N. Nikiforakis, and R. Klein. A dimensionally split Cartesian cut cell method for hyperbolic conservation laws. *J. Comput. Phys.*, 364:186–208, 2018.

[35] S. Gottlieb, D. Ketcheson, and C.-W. Shu. *Strong Stability Preserving Runge-Kutta and Multistep Time Discretizations.* World Scientific Publishing, 2011.

[36] S. Gottlieb and C.-W. Shu. Total-variation-diminishing Runge-Kutta schemes. *Math. Comp.*, 67:73–85, 1998.

[37] S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Rev.*, 43(1):89–112, 2001.

[38] J.-L. Guermond and M. Nazarov. A maximum-principle preserving C0 finite element method for scalar conservation equations. *Comput. Methods Appl. Mech. Engrg.*, 272:198–213, 2014.

[39] C. Gürkan, S. Sticko, and A. Massing. Stabilized cut discontinuous Galerkin methods for advection-reaction problems. *SIAM J. Sci. Comput.*, 42(5):A2620–A2654, 2020.

[40] A. Harten. High resolution schemes for hyperbolic conservation laws. *J. Comput. Phys.*, 49(3):357–393, 1983.

[41] A. Harten, J. M. Hyman, and P. D. Lax. On finite-difference approximations and entropy conditions for shocks. *Comm. Pure Appl. Math.*, 29:297–319, May 1976.

[42] C. Helzel, M. Berger, and R. J. LeVeque. A high-resolution rotated grid method for conservation laws with embedded geometries. *SIAM J. Sci. Comput.*, 26(3):785–809, 2005.

[43] C. Helzel and D. Kerkmann. An active flux method for cut cell grids. In Klöfkorn et al. [50], pages 507–515.

[44] H. Holden and N. H. Risebro. *Front Tracking for Hyperbolic Conservation Laws.* Springer New York, 2002.

[45] G. Jiang and C.-W. Shu. On a cell entropy inequality for discontinuous Galerkin methods. *Math. Comp.*, 62(206):531–538, 1994.

[46] S. Karni. Multicomponent flow calculations by a consistent primitive algorithm. *J. Comput. Phys.*, 112(1):31–43, 1994.

[47] S. Kaur and J. E. Hicken. High-order discontinuous Galerkin Difference cut-cell discretization. In *AIAA Scitech 2021 Forum*, 2021.

[48] D. Kerkmann. *Active Flux Methods for Conservation Laws on Complex Geometries.* PhD thesis, Heinrich Heine University Düsseldorf, 2021.

[49] R. Klein, K. R. Nordin-Bates, and N. Nikiforakis. Well-balanced compressible cut-cell simulation of atmospheric flow. *Philos. Trans. Roy. Soc. A*, 367:4559–4575, 2009.

[50] R. Klöfkorn, E. Keilegavlen, A.F. Radu, and J. Fuhrmann, editors. Springer International Publishing, 2020.

[51] J. Kraaijevanger. Contractivity of Runge-Kutta methods. *BIT*, 31:482–528, 1991.

[52] L. Krivodonova and R. Qin. A discontinuous Galerkin method for solutions of the Euler equations on Cartesian grids with embedded geometries. *J. Comput. Sci.*, 4(1–2):24–35, 2013.

[53] S. N. Kruzkov. First order quasilinear equations in several independent variables. *Mat. Sb.*, 10:217–243, 1970.

[54] D. Kuzmin. A vertex-based hierarchical slope limiter for p-adaptive discontinuous Galerkin methods. *J. Comput. Appl. Math.*, 233(12):3077–3085, 2010.

[55] P. D. Lax. *Hyperbolic Partial Differential Equations.* Courant Lect. Notes Math. Courant Institute of Mathematical Sciences, 2006.

[56] R. J. LeVeque. *Numerical Methods for Conservation Laws.* EMS Ser. Lect. Math. Zürich. Springer Basel AG, 1992.

[57] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts Appl. Math. Cambridge University Press, 2002.

[58] R.J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems.* Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics SIAM, 2007.

[59] S. May. Time-dependent conservation laws on cut cell meshes and the small cell problem. In Klöfkorn et al. [50].

[60] S. May and M. Berger. Two-dimensional slope limiters for finite volume schemes on non-coordinate-aligned meshes. *SIAM J. Sci. Comput.*, 35(5):A2163–A2187, 2013.

[61] S. May and M. J. Berger. An explicit implicit scheme for cut cells in embedded boundary meshes. *J. Sci. Comput.*, 71:919–943, 2017.

[62] S. May and F. Streitbürger. DoD stabilization for non-linear hyperbolic conservation laws on cut cell meshes in one dimension. *Appl. Math. Comput.*, 419, 2022.

[63] J. M. Melenk, I. Perugia, J. Schöberl, and C. Schwab, editors. *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2020+1*. Springer Cham, 2021.

[64] A. Meurer et. al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

[65] S. Mishra, U. Fjordholm, and R. Abgrall. Numerical methods for conservation laws and related equations, February 2019. lecture notes.

[66] B. Müller, S. Krämer-Eis, F. Kummer, and M. Oberlack. A high-order discontinuous Galerkin method for compressible flows with immersed boundaries. *Internat. J. Numer. Methods Engrg.*, 110(1):3–30, 2016.

[67] S. Osher. Riemann solvers, the entropy condition, and difference approximations. *SIAM J. Numer. Anal.*, 21(2):217–235, 1984.

[68] J. J. Quirk. An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies. *Comput. & Fluids*, 23(1):125–142, 1994.

[69] B. Rivière, M. F. Wheeler, and V. Girault. A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems. *SIAM J. Numer. Anal.*, 39(3):902–931, 2002.

[70] Philip Roe. Approximate riemann solvers, parameter vector, and difference schemes. *J. Comput. Phys.*, 43:357–372, 10 1981.

[71] S. Schoeder, S. Sticko, G. Kreiss, and M. Kronbichler. High-order cut discontinuous Galerkin methods with local time stepping for acoustics. *Internat. J. Numer. Methods Engrg.*, 121(13):2979–3003, 2020.

[72] C.-W. Shu. High order eno and weno schemes for computational fluid dynamics. In Timothy J. Barth and Herman Deconinck, editors, *High-Order Methods for Computational Physics*, pages 439–582, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[73] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comput. Phys.*, 77(2):439–471, 1988.

[74] S. Sticko and G. Kreiss. Higher order cut finite elements for the wave equation. *J. Sci. Comput.*, 80(3):1867–1887, 2019.

[75] S. Sticko, G. Ludvigsson, and G. Kreiss. High-order cut finite elements for the elastic wave equation. *Adv. Comput. Math.*, 46(45), 2020.

[76] F. Streitbürger, G. Birke, C. Engwer, and S. May. DoD stabilization for higher-order advection in two dimensions. In Melenk et al. [63].

[77] F. Streitbürger. Stabilisierungstechniken für DG-Verfahren zur Lösung von hyperbolischen Erhaltungsgleichungen auf Gittern mit eingebetteten Objekten. Master's thesis, TU Dortmund University, 2018.

[78] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer, 3rd edition, 2009.

[79] L. N. Trefethen and M. Embree. *Spectra and Pseudospectra.* Princeton Univ. Press, Princeton, 2005.

[80] B. Wendroff and A. B. White. Some supraconvergent schemes for hyperbolic equations on irregular grids. In Josef Ballmann and Rolf Jeltsch, editors, *Nonlinear Hyperbolic Equations — Theory, Computation Methods, and Applications: Proceedings of the Second International Conference on Nonlinear Hyperbolic Problems, Aachen, FRG, March 14 to 18, 1988*, pages 671–677. Vieweg+Teubner Verlag, Wiesbaden, 1989.

[81] B. Wendroff and A. B. White. A supraconvergent scheme for nonlinear hyperbolic systems. *Comput. Math Appl.*, 18(8):761 – 767, 1989.

[82] M. F. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM J. Numer. Anal.*, 15(1):152–161, 1978.