2018 | Faculty of Sciences

UHASSELT
KNOWLEDGE IN ACTION

MATH CCES Center for Computational Engineering Science Mathematics Division | RWTH AACHEN UNIVERSITY

Doctoral dissertation submitted to obtain the degrees of
- Doctor in the Science: Mathematics | UHasselt
- Doktor der Naturwissenschaften (Dr. rer. nat.) | RWTH Aachen University

Alexander Egon Jaust

DOCTORAL DISSERTATION

Novel implicit unconditionally stable time-stepping for DG-type methods and related topics

Promoters: Prof. Dr Jochen Schütz | UHasselt
Prof. Dr Manuel Torrilhon | RWTH Aachen University

D/2018/2451/74

Fakultät für
Mathematik, Informatik
und Naturwissenschaften

RWTH AACHEN UNIVERSITY

Novel implicit unconditionally stable time-stepping for DG-type methods and related topics

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Alexander Egon Jaust M.Sc. RWTH

aus

Recklinghausen

Berichter:

Prof. Dr. Jochen Schütz

Prof. Dr. Manuel Torrilhon

Tag der mündlichen Prüfung:

29.10.2018

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek verfügbar.

Abstract (English)

The thesis is concerned with the improvement and evaluation of hybridized discontinuous Galerkin (HDG) methods for problems from computational fluid dynamics (CFD). HDG methods and high order methods in general promise to improve the quality of simulations used in research and development due to their high accuracy. However, due to their relatively young age there are open question regarding their stability, efficiency and applicability to a wide range of different problems. This thesis focuses on tackling some of these points.

The main concern of this thesis are time-dependent, viscous flow problems that are often described by Euler or Navier-Stokes equations. Therefore, we investigate different types of implicit time integration methods of high order that have not been applied earlier to the HDG methods. This includes implicit general linear methods and multiderivative time integrators. The multiderivative time integrators introduce additional time derivatives that need to be approximated. We find that the approach used for explicit time integrators fails to deliver a uniformly stable discretization. Thus, we devise a new approach for implicit multiderivative integrators that remedies the stability issues.

The efficient implementation of the numerical method is important to limit the required run-time of simulations. During this thesis we have developed several implementations which led to the contribution of an implementation of an HDG method to FESTUNG, an open source framework for MATLAB / GNU Octave. Besides efficiency we focus on a code that is well documented, extendable and easy to use.

Many equations used for CFD are nonlinear and thus their solutions are prone to develop shocks. In order to handle shocks properly with the HDG method, a shock capturing method must be employed. We identify a suitable shock capturing method and justify our decision by comparing the method to other approaches. Additionally, we show that our choice allows to approximate several demanding test cases that contain shocks.

The approximation of a solution to the equations we study requires to solve linear systems of equations. The solving process is a very time and memory consuming part of a simulation. Thus, it is necessary to have an efficient solution strategy. We discuss a newly introduced linear solver that makes use of the special structure of discontinuous Galerkin methods. We apply the new solver to nonlinear problems, which has not been done before, and compare it to an established linear solver to find good agreements in the results of both solvers.

Abstract (Dutch)

In dit proefschrift hebben we het over het onderzoek en de verbetering van een hybridized discontinuous Galerkin (HDG) methode, toegepast op problemen uit de computationele vloeistofmechanica. De HDG methode is een methode van hoge consistentie-orde, die zeer nauwkeurige simulaties in onderzoek en ontwikkeling belooft. De methode is vrij recent, en er zijn dus nog heel wat open vragen rond stabiliteit, efficiëntie en dergelijke. In dit proefschrift zullen we antwoord geven op enkele van die vragen.

Qua toepassingen is dit proefschrift gericht op tijdsafhankelijke viscose stromingen, vaak beschreven door de Euler of Navier-Stokes vergelijkingen. Er worden verschillende impliciete tijdstapmethoden van hoge orde onderzocht, die tot nog toe nooit op de HDG methode toegepast werden. In het bijzonder zullen we het hebben over impliciete algemene lineaire methoden, en impliciete multiderivative tijdstapmethoden. De stabiliteit van de laatstgenoemde integratoren is bij een eerste aanpak, gebruikmakend van de standaardprocedure voor expliciete methoden, slechter dan verwacht. Er wordt dus een nieuwe methode voorgesteld die een onvoorwaardelijke stabiliteit garandeert.

Uiteraard is een efficiënte implementatie van numerieke methoden belangrijk om snel goede resultaten te verkrijgen. Er wordt met verschillende implementaties gewerkt, in het bijzonder hebben we een HDG implementatie bijgedragen aan FESTUNG, een open source framework voor MATLAB / GNU Octave. Naast het verkrijgen van een efficiënt stuk software was het zeer belangrijk om een goed gedocumenteerd en gebruikersvriendelijk programma aan te maken.

De vergelijkingen die vloeistofmechanica beschrijven zijn normaliter niet-lineair. Dit kan tot discontinuïteiten leiden. Om dit met de HDG methode te behandelen is dus een betrouwbare shock capturing methode nodig. In deze thesis identificeren we een geschikte shock capturing methode en motiveren onze keuze door een vergelijking met andere gebruikelijke methoden te maken. We laten zien dat verschillende moeilijke problemen met shocks opgelost kunnen worden.

Tenslotte maakt de benadering het onvermijdelijk om lineaire stelsels van vergelijkingen op te lossen, wat een groot deel van de tijd en van het geheugen in beslag neemt. Bijgevolg is een efficiënte strategie voor de benadering van deze (lineaire) stelsels broodnodig. We bespreken een nieuwe methode die de speciale hiërarchische structuur van de basisfuncties gebruikt, en passen deze ook op een niet-lineaire vergelijking toe.

Abstract (German)

Die vorliegende Dissertation beschäftigt sich mit der Verbesserung und Untersuchung von hybridisierten diskontinuierlichen Galerkin-Verfahren (HDG-Verfahren) für Probleme der numerischen Strömungsmechanik. HDG-Verfahren und im Allgemeinen Verfahren hoher Ordnung, zeichnen sich durch eine hohe Genauigkeit aus. Die hohe Genauigket verspricht die Qualität von Simulationen aus Forschung und Enwicklung zu verbessern. Es sind jedoch Fragen zur Stabilität, Effizienz und Anwendbarkeit auf eine große Anzahl verschiedener Probleme offen.

Das Hauptaugenmerk dieser Arbeit sind zeitabhängige Strömungen, die häufig durch die Euler- oder Navier-Stokes-Gleichungen beschrieben. Aus diesem Grund untersuchen wir verschiedene Arten von impliziten Zeitintegratoren hoher Ordnung, die noch nicht mit HDG-Verfahren angewandt wurden. Die Untersuchung beinhaltet implizite allgemeine lineare Verfahren und implizite multiderivative Zeitintegratoren. Die Stabilität der impliziten multiderivative Zeitintegratoren ist geringer als erwartet, wenn das typische Vorgehen für explizite Verfahren dieser Art verwendet wird. Darum entwickeln wir einen neuen Ansatz für implizite Verfahren, der die Stabilitätsprobleme löst.

Die effiziente Implementierung des numerischen Verfahrens ist sehr wichtig für einen geringen Zeitaufwand von Simulationen. Im Rahmen der Entwicklung verschiedener Implementierungen wurde eine Version des HDG-Verfahrens zu FESTUNG, einem quelloffenen Softwarepaket für MATLAB / GNU Octave, beigesteuert. Neben guter Effizienz zielen wir auf gute Dokumentation, einfache Nutzbarkeit und Erweiterbarkeit ab.

Viele Gleichungen aus der Strömungsmechanik sind nichtlinear, was häufig zu Unstetigkeiten in der Lösung führt. Um diese Unstetigkeiten vernünftig mit einem Verfahren hoher Ordnung wie dem HDG-Verfahren zu approximieren, muss ein shock-capturing Verfahren verwendet werden. Wir identifizieren ein geeignetes Verfahren und begründen unsere Entscheidung durch den Vergleich mit anderen solcher Verfahren. Des Weiteren zeigen wir, dass unsere Wahl des Verfahrens die Approximation verschiedener komplizierter Testfälle erlaubt.

Das Lösen der betrachteten Probleme erfordert die Lösung von linearen Gleichungssystemen. Dieser Schritt macht einen Großteil der Zeit- und Speicheranforderungen der Simulation aus. Deshalb ist es wichtig einen effizienten linearen Löser zu verwenden. Wir diskutieren einen neuen linearen Löser, der die spezielle Struktur von diskontinuierlichen Galerkin-Verfahren ausnutzt. Wir wenden den Löser erstmals auf nichtlineare Probleme an und beobachten eine gute Übereinstimmung der Resultate mit denen eines etablierten Lösers.

Acknowledgements

First of all I would like to thank Prof. Dr. Jochen Schütz and Prof. Dr. Manuel Torrilhon for being the supervisors of my thesis. Especially I am grateful for all the support and discussions with Prof. Dr. Jochen Schütz, who introduced me to the topic of hybridized discontinuous Galerkin methods. I want to thank Prof. Dr. Manuel Torrilhon for his belief in my research and his support that allowed me to start this PhD.

Furthermore, I want to thank Prof. Sebastian Noelle, PhD, who was also a big support of my research, especially during the time I spent in Aachen. I further thank Prof. Dr. Sorin Pop, Prof. Dr. Peter De Maesschalck and Prof. Martin Grepl, PhD, for being part of the committee. Special thanks goes to Prof. David C. Seal, PhD, for the fruitful collaboration over the course of my PhD studies and for being part of the PhD defense committee.

I am thankful for everybody who made it possible that this PhD became a joint PhD between Hasselt University and RWTH Aachen University. This includes Prof. Dr. Manuel Torrilhon and Prof. Dr. Sebastian Noelle who also supported my research financially during my time in Aachen, and Hasselt University who granted funding for the second half of my PhD studies (special research fund BOF16DOC02). Also I am happy for the help of the members of the administrations who where involved in the negotiations of the contract between the universities.

Moreover, I want to thank the members of the Doctoral School of Sciences & Technology in Hasselt. They were always helpful when I had any questions about the administrative process. The same holds for the members of the Promotionsbüro in Aachen. I am grateful for the funding provided by the FWO (grant K200517N) that allowed me to have a research stay at University of Erlangen-Nuremberg in 2017, where I could work with Dr. Vadym Aizinger. It was a great experience and I was lucky to have a fruitful collaboration also with Balthasar Reuter.

Thanks go to all the amazing people that I met during my time at both universities. This includes colleagues, students and professors. I am especially thankful for the help of Dr. Carina Bringedal who gave extensive feedback on the writing of the thesis.

Last but not least I would like to thank my friends, my parents and my brother who supported me through the time I spent working on this thesis.

Contents

1. Introduction

Research and development in the field of fluid dynamics are more and more supported by simulations. Real world experiments, e.g. in wind channels, are still an important pillar when developing new technical devices. However, facilities that allow for running such experiments are usually expensive. Moreover, the manufacturing of prototypes and adjusting of experiment parameters might be expensive and time consuming, maybe even impossible. One way to augment or replace some of these real world experiments are simulations.

In this thesis we are interested in the development of a numerical method for flow problems stemming from classical fluid dynamics. Typical applications can be the flow around an obstacle like around aircrafts, cars, buildings, or the inside of technical devices such as an engine. An example is given in Fig. 1.1. These flows can be described by mathematical models involving partial differential equations (PDEs). The PDEs used in fluid dynamics are based on the physical principles that mass, momentum and energy are conserved. Famous examples in the field of fluid dynamics are the Euler and Navier-Stokes equations. The description by PDEs allows to solve the problems approximately with numerical methods. The field of solving flow problems numerically is usually referred to as computational fluid dynamics (CFD).

CFD became important after the introduction of finite volume (FV) methods because these respect the conservative property of the PDEs to be solved. Due to the popularity of FV methods a lot of research has been done on these. Thus, there is a solid understanding of their behavior and trust in their results both in science and industry. However, ongoing research on numerical methods has led to the development of several new schemes in order to improve or succeed finite volume methods [234]. These schemes are usually referred to as high order methods as they promise higher accuracy (in space) than established (low order) classical FV methods. The high accuracy is highly desirable as it promises to lead to more efficient numerical discretizations. Moreover, high spatial accuracy extends the applicability of CFD simulations to further problems that require the resolution of large and very fine scales such as aeroacoustics. Classical finite volume methods are either too inaccurate to resolve such features or require too much computational work. Moreover, the extension of these methods to high order on unstructured meshes with elements of arbitrary shape is cumbersome. As high order methods are relatively new the understanding of these methods is not yet as solid as for the FV methods. Therefore, a lot of research is done to evaluate and

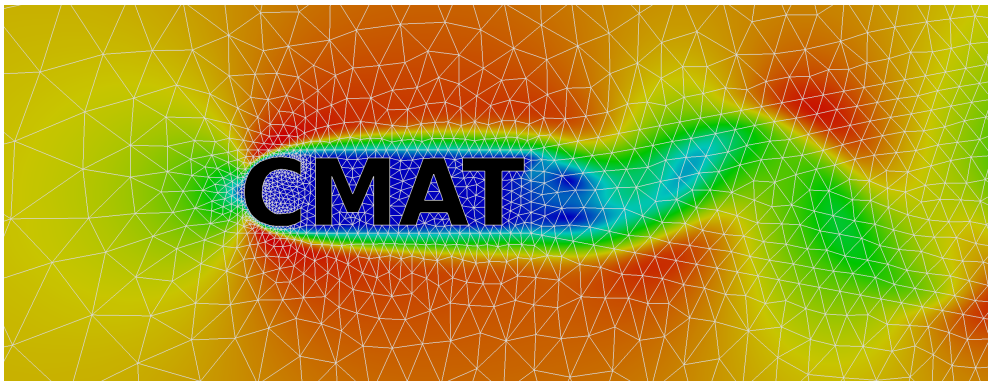
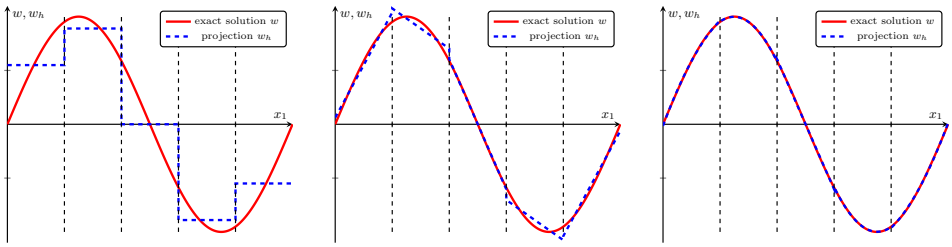


Figure 1.1.: A flow around the logo of the computational mathematics (CMAT) group of Hasselt university. The flow comes from the left and a wake develops downstream of the letters.

understand these methods that focuses on making the high order methods stable, efficient and suitable for a wide range of flow problems. This is necessary for the further promotion of CFD in terms of reliability and applicability to real world problems of practical significance [221].

A class of methods of particular interest for problems from computational fluid dynamics, because they respect the conservation property, are discontinuous Galerkin (DG) methods. These methods combine ideas from established finite volume methods with ideas from finite element methods. The domain of interest is split into intervals, also called elements, (of length Δx). Then, the solution is approximated on each interval by a polynomial of degree P being typical for finite element method. The approximation by a polynomial allows to increase the local accuracy by increasing the polynomial degree. The term "discontinuous" is part of the name of DG methods as these methods do not enforce continuity of the approximated solution between the intervals. In Fig. 1.2, we present the influence of the polynomial degree P on the approximation for a simple example. The quality of the approximation increases rapidly with increasing polynomial degree for smooth solutions. For the lower degree approximations it is also possible to observe the discontinuity in the solution between elements.

Discontinuous Galerkin methods inherit many desirable features from finite volume methods while leading to a higher order approximation of space. A drawback specific to this class of methods is the large number of degrees of freedom that arise in comparison to other methods. This becomes especially penalizing when using implicit solution techniques such as implicit time discretizations and steady state solvers. In this case, a (series of) linear system of equations has to be solved where the unknowns become globally coupled. This leads to



(a) $P = 0$ (b) $P = 1$ (c) $P = 2$

Figure 1.2.: The projection of $\sin(2\pi x_1)$ on $[0, 1]$ onto polynomials of degree $P = 0, 1, 2$. The domain $[0, 1]$ has been split into five intervals of length $\Delta x = 0.2$ that are indicated by the dashed vertical lines. The projection becomes rapidly better when the polynomial degree is increased.

intense memory and run-time requirements.

When the problem is solved in time one usually computes the solution for time steps of size Δt. The size of the time steps cannot be chosen freely, but it depends on the desired accuracy and stability requirements. The discretization of PDEs such as the Euler and Navier-Stokes equations has to respect an upper bound of the CFL-condition in order to be stable if *explicit* time integrators are used. The CFL-condition links the time step size Δt to the size of the elements Δx with the speed information propagates. For the Euler equations the CFL number behaves like $C \sim \frac{\Delta t}{\Delta x}$ while for the Navier-Stokes equations the CFL number behaves like $C \sim \frac{\Delta t}{\Delta x^2}$. In the case of large information propagation speeds or very small elements, the time step size has to be adapted. This can usually not be avoided since the speed of information propagation is dictated by the problem at hand. At the same time the size of the intervals Δx also depends on the problem and features one wants to observe, hence the mesh size can become very small. One can circumvent the strict CFL-condition of explicit time integration methods by using certain *implicit* time integration methods at the cost that (nonlinear) systems of equations have to be solved in each time step.

In order to reduce the number of globally coupled unknowns *hybridized* discontinuous Galerkin (HDG) methods have been introduced. These methods introduce an additional unknown that exists only on element traces, i.e. the element boundaries. The additional unknown allows to rewrite the arising global system of equations. The resulting globally coupled system of equations contains only degrees of freedom of the newly introduced *hybrid* unknown. As the trace of an element has lower dimensionality than the element, this approach can lead to immense reductions in the number of globally coupled unknowns. The unknowns on elements still exist and have to be computed in an element-local fashion which works very nicely with nowadays parallel computer architectures.

The HDG methods have mainly been introduced for time-independent problems, but have been extended to time-dependent problems as well. In this setting these methods are mainly used with implicit time integration schemes. The special structure of the HDG methods is in particular beneficial if linear systems of equations have to be solved. This is the case when implicit time integrators are used or steady state problems are solved. Still, most of the work has focused on the application of HDG methods to steady state problems.

This thesis focuses on further extending the applicability of HDG methods to time-dependent problems. We believe that the schemes can be advantageous for problems from CFD that greatly benefit from implicit time integration techniques. Thus, we work on the following topics related to this goal in this thesis:

- We implement and discuss the HDG methods in an open source framework for MATLAB / GNU Octave that allows for rapid prototyping. The main goals is to provide on an efficient implementation that is well documented such other researcher can benefit from this implementation as well.

- We consider two classes of time integration methods new to HDG methods: Implicit general linear methods and multiderivative time integrators. These can offer great advantages over classical time integrators in terms of stability, run-time and accuracy. We discuss two ways to incorporate multiderivative time integrators and examine their stability. For this we devise a new way to approximate time derivatives that leads to a stable time discretization as the standard approach for explicit schemes fails to be uniformly stable for implicit schemes.

- We identify and investigate a suitable shock capturing method with a simple mesh adaptation procedure. This allows us to apply the HDG method to time-dependent problems with discontinuous solutions. We show the solution of a number of demanding test cases in order to present the versatility of the approach.

- We investigate a newly introduced solver for the linear systems that uses the special structure of discontinuous Galerkin methods. Our experiments include the application of this solver to nonlinear problems.

Each of these points are critical in order to apply the HDG method to a wide range of flow problems in an efficient and stable manner.

Structure of the thesis

This thesis is structured as followed. In Ch. 2, we introduce the governing equations studied in this thesis. This includes a review of classical PDEs from fluid dynamics such as the Euler and Navier-Stokes equations. Ch. 3 introduces discontinuous Galerkin methods. We

first introduce a 'classical' local discontinuous Galerkin (LDG) method. On this basis we describe the hybridized discontinuous Galerkin methods. The HDG methods are our main concern in this thesis. We also highlight the special features of the methods compared to other discontinuous Galerkin methods and shortly discuss their implementation. As we are focusing mainly on time-dependent problems, we review some already established *implicit* time integration schemes for the HDG methods in Ch. 4. The chapter covers backwards differentiation formulae and diagonally implicit Runge-Kutta methods. These methods are used in the following chapters to approximate the solution of new test cases or to compare against new types of time integrators. In Ch. 5, we introduce implicit general linear methods as a first new type of time integrators. These methods are a generalization of the time integrators introduced in the chapter before. Afterwards, we focus extensively on implicit multiderivative time integrators in Ch. 6. We present and analyze two different ways to incorporate the additional time derivatives arising from these kind of time integrators. In Ch. 7, we discuss the efficient implementation of hybridized discontinuous Galerkin methods in MATLAB / GNU Octave. The implementation has been done in the open source framework called FESTUNG and includes an extensive documentation of the implementation. Afterwards, we discuss the application of the HDG method to time-dependent problems with discontinuous solutions in Ch. 8. The approach includes the careful choice of a shock capturing method and the investigation of a mesh adaptation procedure. In Ch. 9, we discuss a special solver for the arising linear systems that takes advantage of the structure of discontinuous Galerkin methods. We end the thesis with a conclusion of the work and outlining present and possible future work in Ch. 10.

Publications

During the preparation of this thesis the following works have been published or accepted for publication:

Journal publications

[124] A. Jaust, B. Reuter, V. Aizinger, J. Schütz, and P. Knabner. "FESTUNG: A MATLAB / GNU Octave toolbox for the discontinuous Galerkin method. Part III: Hybridized discontinuous Galerkin (HDG) formulation". In: *Computers and Mathematics with Applications* 2018.12 (2018), pp. 4505–4533.

[126] A. Jaust, J. Schütz, and D. C. Seal. "Implicit multistage two-derivative discontinuous Galerkin schemes for viscous conservation laws". In: *Journal of Scientific Computing* 69.2 (2016), pp. 866–891.

[212] J. Schütz, D. C. Seal, and A. Jaust. "Implicit Multiderivative Collocation Solvers for Linear Partial Differential Equations with Discontinuous Galerkin Spatial Discretizations". In: *Journal of Scientific Computing* 73.2 (2017), pp. 1145–1163.

Other publications

[120] A. Jaust and J. Schütz. "General linear methods for time-dependent PDEs". In: *Theory, Numerics and Applications of Hyperbolic Problems II*. Ed. by C. Klingenberg and M. Westdickenberg. in press.

[121] A. Jaust, J. Schütz, and D. C. Seal. "Multiderivative time-integrators for the hybridized discontinuous Galerkin method". In: *Conference Proceedings of the YIC GACM 2015*. Ed. by S. Elgeti and J.-W. Simon. Publication Server of RWTH Aachen University, 2015. URL: https://publications.rwth-aachen.de/record/480970/files/ProceedingsYIC-GACM-ACCES.pdf.

[122] A. Jaust, J. Schütz, and M. Woopen. "A Hybridized discontinuous Galerkin Method for Unsteady Flows with Shock-Capturing". In: *AIAA Paper 2014-2781* (2014).

[125] A. Jaust, J. Schütz, and V. Aizinger. "An efficient linear solver for the hybridized discontinuous Galerkin method". In: *PAMM* 16.1 (2016), pp. 845–846.

[127] A. Jaust, J. Schütz, and M. Woopen. "An HDG Method for Unsteady Compressible Flows". English. In: *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014*. Ed. by R. M. Kirby, M. Berzins, and J. S. Hesthaven. Vol. 106. Springer International Publishing, 2015, pp. 267–274.

This thesis discusses parts of the results of the publications above, but also presents further results.

2. Governing equations

The goal of this thesis is to investigate a numerical method for accurately solving time-dependent partial differential equations (PDEs) arising from fluid dynamics. Therefore, the equations to be solved are usually the Euler or Navier-Stokes equations [7, 153, 185, 207, 230, 236]. There exist other ways of describing flows problems, e.g. using the Boltzmann equation [224, 239], but we do not focus on these kinds of models here. While developing and implementing the methods it is usually a good choice to start with simpler or at least scalar equations such as the convection, convection-diffusion or Burgers' equation [143, 145, 229]. Therefore, we introduce these equations as well.

The chapter is structured as follows. First, we define some notation to compactly represent the equations. After that, the special structure all of the equations considered in this thesis is laid out. Finally, the Euler and Navier-Stokes equations, including a dimensional analysis and boundary conditions, are introduced. Most parts of this section are described in more detail in standard text books about fluid dynamics and aerodynamics [7, 153, 185, 207, 230, 236] and publications about the analysis and numerical approximation of conservation laws [91, 136–138, 143, 145, 229].

2.1. Notation

We consider equations in d space dimensions with coordinate $x_i,\ i = 1, ..., d$. In this work we focus on PDEs in one or two space dimensions. However, the PDEs considered extend to three space dimensions, too. Vectors $v \in \mathbb{R}^n$ are denoted by italic letters and matrices $\mathrm{A} \in \mathbb{R}^{m \times n}$ by capitals using roman letters. Besides standard notation for derivatives we introduce the dyadic product for two vectors $a, b \in \mathbb{R}^d$ defined as

$$a \otimes b = ab^{\mathrm{T}} = \begin{pmatrix} a_1 b_1 & \dots & a_1 b_d \\ \vdots & \ddots & \vdots \\ a_d b_1 & \dots & a_d b_d \end{pmatrix}. \tag{2.1}$$

The product is also defined for vectors of different lengths and dimensions larger than d, but we only use it for vectors having the same length as the space dimension d. Additionally, we

define the divergence of a matrix $\mathrm{A} \in \mathbb{R}^{m \times d}$ as

$$\nabla \cdot \mathrm{A} = \nabla \cdot \begin{pmatrix} a_{11} & \dots & a_{1d} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{md} \end{pmatrix} \coloneqq \begin{pmatrix} \nabla \cdot (a_{11}, \dots, a_{1d})^{\mathrm{T}} \\ \vdots \\ \nabla \cdot (a_{m1}, \dots, a_{d})^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{d} \frac{\partial a_{1i}}{\partial x_i} \\ \vdots \\ \sum_{i=1}^{d} \frac{\partial a_{mi}}{\partial x_i} \end{pmatrix}. \tag{2.2}$$

We solve the PDEs for an unknown w that is a function of time and space, i.e. $w = w(t, x)$. In order to avoid unnecessary clutter in the equations we do not always state this explicitly. Additionally, we use the Kronecker delta δ_{ij} that is defined as

$$\delta_{ij} \coloneqq \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$$

2.2. Problems in flux formulation

All PDEs in this work can be reformulated as

$$\begin{aligned} \partial_t w + \nabla \cdot (f_c(w) - f_v(w, \nabla w)) &= h(w, \nabla w) \quad \text{on} \quad (t, x) \in (0, t_{\text{end}}] \times \Omega \\ w(0, x) &= w_0(x) \end{aligned} \tag{2.3}$$

where the unknown $w(t, x)$ may be vector valued with m components. The space domain is denoted as Ω and the time interval as $(0, t_{\text{end}}]$. We might also refer to the time-space domain as $\Omega_T \coloneqq (0, t_{\text{end}}] \times \Omega$. The functions f_c, f_v and h are given and may depend on the unknown w and the latter two also on the gradient ∇w. Additionally the functions might depend explicitly on time t and space x, but we suppress this to keep the notation compact. We refer to f_c as convective, to f_v as viscous flux function and to h as source term. Therefore, we refer to the equation (2.3) as the PDE in flux formulation. The fluxes are given as

$$f_c = (f_{c,1}, \dots, f_{c,d}), \quad f_v = (f_{v,1}, \dots, f_{v,d})$$

such that we can make use of the divergence operator for matrices (2.2). The flux in x_i-direction is given by $f_{c,i}$ and $f_{v,i}$ respectively. Usually, both fluxes are continuously differentiable, i.e. $f_{c,i} \in C^l(\mathbb{R}^m, \mathbb{R}^m)$ and $f_{v,i} \in C^l(\mathbb{R}^m \times \mathbb{R}^{m \times d}, \mathbb{R}^m)$ with $l \geq 1$.

Depending on the type of equation analyzed, not all terms must be present. If $\partial_t w$ is not present, the PDE reads

$$\nabla \cdot (f_c(w) - f_v(w, \nabla w)) = h(w, \nabla w)$$

and the PDE describes a steady state problem, that is $w = w(x)$. In the absence of a viscous flux and source term

$$\partial_t w + \nabla \cdot f_c(w) = 0$$

we have a first order partial differential equation. In case, a viscous flux f_v is present, the PDE is a second order partial differential equation because f_v depends on ∇w.

2.3. Hyperbolic equation

Let us for the moment focus on the hyperbolic PDE

$$\partial_t w + \nabla \cdot f_c(w) = 0 \tag{2.4}$$

already mentioned in the previous section. Equations that fall into this framework have been extensively studied in the field of conservation laws. For a detailed analysis of these equations and numerical methods the reader is referred to text books like [91, 136–138, 143, 145, 229].

The equation is hyperbolic if the eigenvalues of a linear combination of the Jacobians $f'_{c,i}(w), i = 1, \ldots, d$ with

$$\mathrm{A}(w, v) \coloneqq \sum_{i=1}^{d} \frac{\partial}{\partial w} f_{c,i}(w) \cdot v_i \tag{2.5}$$

are real for any $v \in \mathbb{R}^d \setminus \{0\}$ and a full set of linear independent eigenvectors exists. In this case, A can be written as

$$\mathrm{A}(w, v) = \mathrm{R}(w, v)\Lambda(w, v)\mathrm{R}^{-1}(w, v) \tag{2.6}$$

with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_m) \in \mathbb{R}^{m \times m}$ being the matrix having the eigenvalues on its diagonal and $\mathrm{R} = (r_1, \ldots, r_m),\ r_i \in \mathbb{R}^m$, is the matrix with the associated eigenvectors as columns. The real eigenvalues dictate an important property of this type of equations; they describe wave propagation velocities. From this follows the speed and the direction in which information propagates along so called characteristic lines.

In 1D, the solution being transported along characteristic lines is an important property and allows, in principle, to solve the PDE exactly. However, for nonlinear problems the characteristic lines tend to intersect after finite time resulting in a discontinuity, a shock, in the solution. Therefore, the solution is not continuous nor differentiable anymore and therefore does not fulfill the equation (2.4) in the classical sense. However, discontinuous solutions should be admissible because the physical problems one aims to model show (nearly) discontinuous properties as well.

In multiple space dimensions the equations show the same behavior like the development of non-smooth solutions. However, the analysis is much more involved and cannot be easily taken over from the 1D case. Nevertheless, the results from 1D can guide the development of numerical schemes [91, 111, 143, 229]. Especially boundary conditions can make use of the theory in 1D as the problem can be interpreted as a 1D problem in direction normal to the boundary, see Sec. 2.6.2.

A number often discussed for hyperbolic problems is the so-called CFL (Courant-Friedrichs-Lewy) number. The CFL number relates the speed of which information propagates in the mesh to the time step size of a numerical simulation. In this thesis we use a hyperbolic CFL

number defined as

$$\mathrm{CFL} \coloneqq |\lambda_{\max}| \frac{\Delta t}{\Delta x_{\min}} \tag{2.7}$$

based on the largest eigenvalue $\lambda_{\max}$ present in the computational domain and $\Delta x_{\min}$ being the smallest edge of all elements in the mesh. We use this definition of the CFL number also for problems that are not purely hyperbolic. In these cases the CFL number is based on the eigenvalue of the hyperbolic part of the equation.

Remark 1 *Flows in reality do not have truly discontinuous jumps in their quantities due to the present viscosity. Instead the flow quantities change strongly over a very short distance leading to steep gradients. The width or thickness of a shock depends on the viscosity μ and heat conductivity κ which themselves depend on the free mean path l of the fluid [248]. The free mean path is the average distance of molecules in the gas traveling before they collide. Depending on the assumptions on the viscosity and the conductivity with regard to their dependence on temperature and pressure and the used equations, Navier-Stokes or Boltzmann, the shock width is often derived to be a multiple of the free mean path l [156, 187, 192]. The flow features, such as vortices, observed in typical CFD applications are much larger than the free mean path and therefore than a shock. In this case it can be a valid modeling assumption for shocks to be discontinuous [145].*

2.4. Convection-diffusion equation

A common model problem is the convection-diffusion equation which can be written in flux formulation (2.3) with fluxes

$$f_c(w) = uw, \quad f_v(w, \nabla w) = \varepsilon \nabla w \tag{2.8}$$

and possibly a given source term h. The convection velocity $u = u(t, x)$ may be a function of time and space and describes the transport of the unknown along the flow field $u = (u_1, \ldots, u_d)$. In the case of a multicomponent vector $m > 1$ the velocity u might become a matrix. The viscous term with diffusion constant $\varepsilon > 0$ accounts for diffusive effects. This leads to a smoothing of the solution over time. In case of vanishing diffusivity $\varepsilon \equiv 0$, the problem is purely convective meaning the initial data is transported along u.

This equation models the typical flow properties present in real world flows, namely convective transport and diffusive transport. Therefore, the convection-diffusion equation is very suitable for model development as one can easily focus on single effects by setting $u \equiv 0$ or $\varepsilon \equiv 0$. Moreover, it is comparably easy to construct problems that have an analytical solution.

Remark 2 *In the literature, the convection speed u is sometimes also referred to as advection speed when it is constant. Consequently, the equation might be referred to as advection-diffusion equation in this case.*

2.5. Boundary conditions

We mainly use two kinds of boundary conditions for hyperbolic equations and the convection-diffusion equations.

Dirichlet boundaries If incoming characteristics or the diffusivity of a problem requires a boundary value we can set the value on the boundary as

$$w_{\mathrm{BC}} = w_\infty$$

where the ∞-index refers to a known boundary state.

Periodic boundaries It is often advantageous to use a very simple setting in terms of domain and boundary conditions for verification and testing. A common choice is to consider a periodic domain with periodic length l_i in the ith space dimension. This refers to setting

$$w(t, x_i) = w(t, x_i + l_i)$$

and extends the domain to infinity in the respective space dimension.

2.6. Euler equations

A famous set of equations used in CFD are the Euler equations [7, 91, 153, 207, 230, 236]. They describe inviscid flow. The equations in d space dimensions are given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0, \tag{2.9a}$$

$$\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u + p\mathrm{I}) = 0, \tag{2.9b}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + p)u) = 0, \tag{2.9c}$$

and describe the conservation of mass (2.9a), conservation of momentum in every spatial direction i (2.9b), and conservation of energy (2.9c). For this given set of Euler equations it is assumed that the effect of gravitational forces can be neglected. Otherwise, an additional term ρg would appear on the right hand side of the momentum conservation equation.

There are $d + 2$ equations, but $d + 3$ unknowns (density ρ $[\frac{\mathrm{kg}}{\mathrm{m}^3}]$, velocity in x_i-direction u_i $[\frac{\mathrm{m}}{\mathrm{s}}]$, total energy per unit volume E $[\frac{\mathrm{J}}{\mathrm{m}^3}]$ and pressure p $[\frac{\mathrm{N}}{\mathrm{m}^2}]$). Therefore, an additional equation is needed. A common closure is the ideal gas law [153, 236]

$$p = \rho R T \tag{2.10}$$

with R $[\frac{\mathrm{J}}{\mathrm{kg} \cdot \mathrm{K}}]$ being the specific gas constant and T $[K]$ the temperature. Under the assumption that the gas is calorically perfect, the specific internal energy is given as $e = c_v T$ $[\frac{\mathrm{J}}{\mathrm{kg}}]$ with

c_v $[\frac{\mathrm{J}}{\mathrm{kg}\cdot\mathrm{K}}]$ being the specific heat capacity at constant volume. The specific heat capacity at constant pressure is referred to as c_p $[\frac{\mathrm{J}}{\mathrm{kg}\cdot\mathrm{K}}]$. Specific heats and the specific gas constant are parameters depending on the studied gas. For air under normal conditions a ratio of specific heats $\gamma = \frac{c_p}{c_v}$ is usually assumed to be $\gamma \equiv 1.4$ [7, 153, 236]. Moreover, the relations

$$c_p - c_v = R, \quad c_p = \frac{\gamma R}{\gamma - 1}, \quad c_v = \frac{R}{\gamma - 1},$$

hold and allow us to rewrite the internal energy e as

$$e = c_v T = \frac{RT}{\gamma - 1} \Leftrightarrow RT = (\gamma - 1)e. \tag{2.11}$$

Additionally, the total energy E can be expressed as

$$E = \rho \left(e + \frac{1}{2} \|u\|_2^2 \right)$$

making it clear that the total energy consists of two parts, the inner energy ρe the fluid holds as heat and as kinetic energy $\frac{1}{2}\rho\|u\|_2^2$. Using this equation together with the equations (2.11) for the specific internal energy the pressure is given as

$$p = \rho RT = (\gamma - 1)\rho e = (\gamma - 1)\left(E - \frac{1}{2}\rho \|u\|_2^2 \right). \tag{2.12}$$

Another important quantity is the speed of sound of the fluid

$$c = \sqrt{\gamma RT} = \sqrt{\frac{\gamma p}{\rho}}. \tag{2.13}$$

In particular when one wants to compare two different flow conditions one often refers to the Mach number

$$\mathrm{Ma} = \frac{u_0}{c} \tag{2.14}$$

that relates a characteristic flow velocity u_0 to the speed of sound. The Mach number is a *nondimensional* number that can be obtained by dimensional analysis [7, 153, 207, 230, 236] as described in the next section. The Mach number often serves as a measure of compressibility and information propagation direction. For low Mach numbers $\mathrm{Ma} < 0.3$ the density variations are usually less than 5%. In this case, the flow is sometimes considered incompressible, i.e. $\rho \equiv$ const. In the case $\mathrm{Ma} > 1$ the flow is supersonic which affects the direction of information propagation. The eigenvalues are all positive in this case hence information can only move in flow direction and not upstream anymore. Flows around obstacles like an airfoil often locally exhibit Mach numbers in the supersonic regime even if the free stream is subsonic ($\mathrm{Ma} < 1$), but close to the supersonic regime, i.e., $\mathrm{Ma} > 0.8$. This has to be considered because supersonic flows do usually *not* return to subsonic conditions by undergoing a smooth transition. Instead, a shock — a discontinuity in flow quantities —

develops. It is a common feature of supersonic flows that is also observed in the real world. This behavior results from the nonlinearity in the Euler equations and can pose a severe problem for numerical discretizations.

As mentioned in the previous section, the Euler equations can be rewritten in flux formulation (2.3). Since the Euler equations are a system of first order equations no viscous flux is present. In this work we focus on the Euler equations in two space dimensions. In this case the vector of unknowns and the convective flux $f_c = (f_{c,1}, f_{c,2})$ are given as

$$w = \begin{pmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ E \end{pmatrix}, \quad f_{c,1}(w) = \begin{pmatrix} \rho u_1 \\ p + \rho u_1^2 \\ \rho u_1 u_2 \\ u_1(E+p) \end{pmatrix}, \quad f_{c,2}(w) = \begin{pmatrix} \rho u_2 \\ \rho u_1 u_2 \\ p + \rho u_2^2 \\ u_2(E+p) \end{pmatrix}. \tag{2.15}$$

2.6.1. Dimensional analysis

The Euler equations (2.9) are given in 'dimensional' form. In order to simplify the comparison of different experiments related to fluid dynamics one can bring the equations to dimensionless form by eliminating all units of the unknowns and parameters present in the equations [7, 153, 207, 230, 236]. Rewriting the equations in nondimensional form leads to a set of dimensionless parameters that allow for a better understanding of the behavior and comparison of different problems. Consider two flows with largely differing flow velocities. Although one flow is 'fast' while the other is 'slow' they can behave identical when the remaining flow parameters are adapted accordingly. The dimensional analysis normalizes the equations such that these cases can be easily identified.

In this work the dimensional analysis is done by defining suitable reference parameters and rescaling the equations to remove all units. Another way to do so is using Buckingham's Π-theorem [7, 153, 207] for which all units of parameters and unknowns in the equations are identified. These are then used to compute the dimensionless numbers. The number of dimensionless parameters is independent of the used reference parameters and which approach for the nondimensionalization is used. However, this may affect the definition of the dimensional parameters. We choose the reference parameters such that we get dimensionless numbers that are commonly used in fluid dynamics.

First, the reference parameters have to be defined. A reference density ρ_0, length L, velocity u_0 and pressure p_0 are chosen. In contrast to the other sections, the unknowns and parameters with units are denoted by $\cdot'$ in the following. Unknowns and parameters without this indicator are dimensionless. Then, the dimensionless parameters can be expressed as

$$\rho = \frac{\rho'}{\rho_0}, \ \frac{\partial}{\partial x_i} = L\frac{\partial}{\partial x_i{}'}, \ u_i = \frac{u_i'}{u_0}, \ t = \frac{t'}{(\frac{L}{u_0})}, \ p = \frac{p'}{p_0}. \tag{2.16}$$

This choice follows [7] and is not the only suitable choice as explained earlier. One could choose a reference time, for example. However, the reference parameters should be known quantities. While it is usually not easy to define an obvious reference time for flow problems it is easy to define a meaningful length due to the investigated geometry (size of an obstacle in flow field, height of a channel) and velocity like the velocity at the inflow or at free flow conditions. In the following we briefly present the process of scaling the mass, momentum and energy equations.

Conservation of mass The scalings in (2.16) are plugged into the mass conservation (2.9a) equation. This leads to

$$\begin{aligned} \frac{\partial \rho'}{\partial t'} + \nabla' \cdot \left(\rho' u'\right) &= 0 \\ \Leftrightarrow \left(\frac{u_0 \rho_0}{L}\right) \frac{\partial \rho}{\partial t} + \left(\frac{u_0 \rho_0}{L}\right) \nabla \cdot (\rho u) &= 0 \\ \Leftrightarrow \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) &= 0 \end{aligned}$$

and as all reference quantities cancel, the dimensional and dimensionless form of the equation are identical.

Conservation of momentum After rescaling the momentum equation (2.9b)

$$\begin{aligned} \frac{\partial(\rho' u')}{\partial t'} + \nabla' \cdot \left(\rho' u' \otimes u'\right) &= -\nabla' \cdot (p' \mathrm{I}) \\ \left(\frac{\rho_0 u_0^2}{L}\right) \frac{\partial(\rho u)}{\partial t} + \left(\frac{\rho_0 u_0^2}{L}\right) \nabla \cdot (\rho u \otimes u) &= -\left(\frac{p_0}{L}\right) \nabla \cdot (p \mathrm{I}) \\ \frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u) &= -\frac{p_0}{\rho_0 u_0^2} \nabla \cdot (p \mathrm{I}) \end{aligned}$$

a factor of scaling parameters is left in front of the pressure gradient. This is a dimensionless number for the Euler equations. When multiplied by the dimensionless ratio of specific heat capacities γ

$$\frac{p_0}{\rho_0 u_0^2} = \frac{c^2}{\gamma u_0^2} = \frac{1}{\gamma \mathrm{Ma}^2}$$

it can be reformulated using the speed of sound (2.13) to reveal the Mach number (2.14). The dimensionless form of the momentum equation therefore reads

$$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u) = -\frac{1}{\gamma \mathrm{Ma}^2} \nabla \cdot (p \mathrm{I}).$$

The ratio of specific heats is a dimensionless number depending on the fluid/gas studied while the Mach number depends solely on the flow conditions.

Conservation of energy As for the mass and momentum equations the energy equation (2.9c) is rescaled

$$\frac{\partial E'}{\partial t'} + \nabla' \cdot \left(E'u'\right) = -\nabla' \cdot \left(p'u'\right)$$
$$\left(\frac{\rho_0 u_0^3}{L}\right) \frac{\partial E}{\partial t} + \left(\frac{\rho_0 u_0^3}{L}\right) \nabla \cdot (Eu) = -\left(\frac{p_0 u_0}{L}\right) \nabla \cdot (pu)$$
$$\frac{\partial E}{\partial t} + \nabla \cdot (Eu) = -\frac{p_0}{\rho_0 u_0^2} \nabla \cdot (pu)$$

and the same term of reference parameters is obtained as for the momentum equation, i.e. no new dimensionless parameter is found. The energy equation in dimensionless form is given as

$$\frac{\partial E}{\partial t} + \nabla \cdot (Eu) = -\frac{1}{\gamma \mathrm{Ma}^2} \nabla \cdot (pu)\,.$$

2.6.2. Boundary conditions

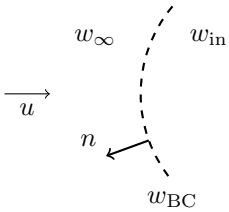
In order to solve the Euler equations boundary conditions w_{BC} must be prescribed. Special care has to be taken on boundaries at which the fluid enters the domain (inflow boundaries, see Fig. 2.1a), i.e. $u \cdot n < 0$, and leaves the domain (outflow boundaries, see Fig. 2.1b), i.e. $u \cdot n > 0$. The normal vector n points out of the computational domain and u is the flow velocity vector. The boundary conditions may depend on the state inside the computational domain w_{in} and the free flow state outside the computational domain w_∞ on these boundaries. See Fig. 2.1 for a brief sketch of the situation of both boundary situations.

The Euler equations form a hyperbolic equation with real eigenvalues [91, 111, 143, 229]

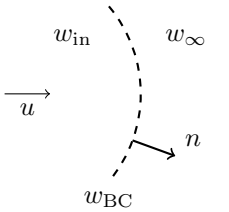
$$\lambda_1 = u \cdot n + c,\ \lambda_2 = u \cdot n - c,\ \lambda_j = u \cdot n,\ j = \{3, \ldots, m\} \tag{2.17}$$

that dictate how boundary conditions have to be prescribed. This follows from (2.6) by setting v to the normalized outward pointing normal vector n with $\|n\|_2 = 1$ on the domain boundary. The eigenvalues refer to the characteristic velocities in the domain, the normal velocity $u \cdot n$ of the fluid and the speed of sound c in the domain. Thus, whenever $\lambda_i < 0$ on a boundary holds, a characteristic quantity has to be prescribed by a known boundary state and for $\lambda_i > 0$ on a boundary the interior state can be used. This directly relates to the Mach number (2.14). For subsonic flows, i.e. $\mathrm{Ma} < 1$, the absolute value of the flow velocity including the absolute value of the velocity in the normal direction $u \cdot n$ are guaranteed to be smaller than the speed of sound. This means that information mainly propagates in the normal flow direction, but part of the information may also propagate in the opposite direction as $\lambda_2 < 0$. Therefore, $m - 1$ quantities must be prescribed on inflow boundaries while one quantity must be given on outflow boundaries where $\mathrm{Ma} < 1$ holds.

The situation changes for supersonic flows, i.e. $\mathrm{Ma} > 1$, as flow information propagates only downstream. In this case the sign of all eigenvalues at the inflow boundary $u \cdot n < 0$



(a) Inflow boundary.



(b) Outflow boundary.

Figure 2.1.: Sketch of the inflow and outflow boundary situation. The flow direction is from left to right, w_∞ indicates the free flow conditions outside the domain and w_{in} indicates the state inside the domain. A compatible boundary state w_{BC} has to be determined on the domain boundary (dashed line).

is negative and at the outflow boundary $u \cdot n > 0$ is positive. Therefore, a full set of m quantities must be given at the inflow boundary while no quantities are to be prescribed on outflow boundary.

Although the number of conditions that has to be set is known due to the characteristics, the characteristics do not state how values on the boundary have to be determined. In the literature, see e.g. [40, 77, 111, 143, 229], different ways to define boundary conditions for numerical simulations are described. In the following, we limit the description to the ones used in this work.

Slip wall boundary condition Solid walls, such as the walls of a channel or the surface of an airfoil cannot be penetrated by the fluid. Thus, the boundary condition must be chosen such that the velocity in normal direction is zero, i.e. $u \cdot n = 0$, on walls. This means the flow shall not go through the boundary, but rather follow its tangential direction. This can be achieved by imposing a boundary condition that mirrors the velocity in the normal direction. The velocity in the tangential direction, the density and the energy are unchanged. This boundary state w_{BC} in 2D reads

$$w_{\text{BC}}(w_{\text{in}}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-n_1^2 & -n_1 n_2 & 0 \\ 0 & -n_1 n_2 & 1-n_2^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} w_{\text{in}}$$

where w_{in} is the state on the inside of the domain boundary and $n_i,\ i = 1, 2$ are the components of the normal vector.

Characteristic far field boundary condition Due to the Euler equations being hyperbolic the flux Jacobian using the normal vector n can be written as

$$\left(\frac{\partial}{\partial w} f_c(w)\right) \cdot n = \mathrm{R}(w,n)\Lambda(w,n)\mathrm{R}^{-1}(w,n)$$

as stated in (2.6). This allows to set the boundary conditions at the far field using the characteristic representation $w_{\mathrm{C}} = \mathrm{R}^{-1}(w,n)w$. This representation is advantageous as the boundary values can be set depending on the sign of the local eigenvalues (2.17). An eigenvalue $\lambda_i < 0$ describes a characteristic entering the domain. The corresponding characteristic unknown on the boundary $w_{\mathrm{BC,C},i}$ (indicated by the additional BC subscript) is set from the given boundary state that is usually *not* given in the characteristic representation. In this case the boundary state is transformed to the characteristic representation by $w_{\mathrm{BC,C},i} = (\mathrm{R}^{-1}w_{\mathrm{BC}})_i$. For all other eigenvalues the values from the interior of the domain can be used. After setting all entries, the vector is transformed back into conservative form w_{BC}. The corresponding matrices R and R^{-1} can be found in text books as [91, 111, 143]. Note that the order of eigenvalues used in the diagonal matrix $\Lambda(w,n)$ may differ depending on the source.

Riemann invariant boundary condition Another way to obtain boundary conditions is based on so called *Riemann invariants* [40, 111, 143, 229]. On the boundary we can interpret the Euler equations as 1D problem when considering the problem in the normal direction, see Fig. 2.1 for a sketch of the situation at inflow and outflow boundaries. In this case, one can observe that certain quantities, the Riemann invariants, are constant on characteristic curves that are related to the eigenvalues $u, u+c$ and $u-c$ (2.17) for *smooth* flows.

Using the 1D Euler equations (in non-conservative form) and its eigenvector basis (2.3) one can derive the quantities that are constant along the characteristic lines [111, 143, 229]. The entropy s is such a quantity which fulfills the convection equation

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

and therefore is constant along streamlines. Two additional invariants $R^{\pm} = u \pm \frac{2c}{\gamma-1}$ that are constant can be derived on the characteristic lines given by the eigenvalues $u \pm c$.

The correct boundary state w_{BC} has to be derived from the two states adjacent to the boundary: The state in the interior of the domain w_{in} and the state outside of the domain given by the free flow quantities w_∞. The density and pressure on the boundary can be determined by

$$\rho_{\mathrm{BC}} = \left(\frac{c_{\mathrm{BC}}^2}{\gamma s_{\mathrm{BC}}}\right)^{\frac{1}{\gamma-1}}, \quad p_{\mathrm{BC}} = \frac{\rho_{\mathrm{BC}} c_{\mathrm{BC}}^2}{\gamma} \tag{2.18}$$

once the entropy $s = \frac{c^2}{\gamma\rho^{\gamma-1}}$ is known. On an inflow boundary the flow is entering the domain. Thus, the entropy is constant along the streamline entering the domain and it can

be determined from the free flow state. In this case, we have

$$s_{\mathrm{BC}} = \frac{c_\infty^2}{\gamma \rho_\infty^{\gamma-1}} \tag{2.19}$$

on inflow boundaries.

An expression for the speed of sound c_{BC} on the boundary is still missing, but necessary to determine the density and pressure (2.18). Moreover, the velocity on the boundary u_{BC} is needed to complete the boundary state w_{BC}. Both can be determined using the Riemann invariants $R^{\pm}$. At an inflow boundary, the Riemann invariants referring to outgoing waves R^+ and incoming waves R^-

$$R^+ = u_{\mathrm{in}} \cdot n + \frac{2c_{\mathrm{in}}}{\gamma - 1}, \quad R^- = u_\infty \cdot n - \frac{2c_\infty}{\gamma - 1},$$

depend on the interior state and free flow. We know that the invariants must be constant along their characteristic lines. Therefore, we can follow the characteristic lines to the boundary state and obtain

$$u_{\mathrm{BC}} \cdot n + \frac{2c_{\mathrm{BC}}}{\gamma - 1} = u_{\mathrm{in}} \cdot n + \frac{2c_{\mathrm{in}}}{\gamma - 1}, \tag{2.20a}$$

$$u_{\mathrm{BC}} \cdot n - \frac{2c_{\mathrm{BC}}}{\gamma - 1} = u_\infty \cdot n - \frac{2c_\infty}{\gamma - 1}. \tag{2.20b}$$

After adding and subtracting the two equations we obtain

$$u_{\mathrm{BC}} \cdot n = \frac{1}{2}\left(R^+ + R^-\right), \quad c_{\mathrm{BC}} = \frac{\gamma - 1}{4}\left(R^+ - R^-\right)$$

for the velocity in the normal direction and the speed of sound on the boundary. Now, the entropy (2.19) and therefore the density and pressure (2.18) can be determined. The velocity has to be projected back onto the x_1- and x_2-direction as the whole process based on Riemann invariants uses the 1D problem in the direction normal to the domain boundary. Then, the velocity vector is given by

$$u_{\mathrm{BC}} = u_\infty + \left(u_{\mathrm{BC}} \cdot n - u_\infty \cdot n\right) n. \tag{2.21}$$

Outflow boundary conditions follow the same procedure, but the roles of interior and free flow quantities are interchanged.

Supersonic boundary conditions As described before, at supersonic flow conditions we need to describe all flow quantities at the inflow and no boundary conditions at the outflow. Therefore, we set

$$w_{\mathrm{BC}}(w_{\mathrm{in}}) = w_{\mathrm{BC}} = \begin{pmatrix} \rho_\infty \\ \rho_\infty u_\infty \\ E_\infty \end{pmatrix}$$

at the inflow and $w_{\mathrm{BC}}(w_{\mathrm{in}}) = w_{\mathrm{in}}$ on outflow boundaries. The quantities given at free flow conditions are denoted by $\cdot_\infty$. This is the extension of the Dirichlet boundary conditions to the Euler equations as described previously, see Sec. 2.5.

2.7. Navier-Stokes equations

The second set of famous equations in CFD discussed in this work are the Navier-Stokes equations [7, 153, 185, 207, 230, 236]. These extend the Euler equations by modeling viscous effects. This means that momentum and energy are exchanged within the fluid by inner friction and heat transport is accounted for. The incorporation of viscosity also introduces flow features like boundary layers and turbulence, e.g., that are not described by the Euler equations. As one can see from the Navier-Stokes equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0 \tag{2.22a}$$

$$\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u + p\mathrm{I}) = \nabla \cdot \mathrm{S} \tag{2.22b}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E+p)\rho u) = \nabla \cdot (\mathrm{S}u - \nabla q) \tag{2.22c}$$

the viscous effects are described by additional terms compared to the Euler equations (2.9). The vector of unknowns w is the same as for the Euler equations. The additional terms are the stress tensor S and the heat flux q $[\frac{\mathrm{W}}{\mathrm{m}^2}]$. The components of the stress tensor are defined as

$$S_{ij} := \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\delta_{ij} \nabla \cdot u \right).$$

The stress tensor introduces a viscosity μ $[\frac{\mathrm{kg}}{\mathrm{ms}}]$ that depends on the type of fluid and the temperature. A common closure for the viscosity is Sutherland's law [225] given as

$$\mu = \frac{C_1 T^{\frac{3}{2}}}{T + C_2}$$

where C_1 and C_2 are model constants depending on the fluid. A common choice of the coefficients for air modeled as an ideal gas are

$$C_1 = 1.461 \cdot 10^{-6} \frac{\mathrm{kg}}{\mathrm{m} \cdot \mathrm{s} \cdot \sqrt{\mathrm{K}}}, \quad C_2 = 110.3\mathrm{K}.$$

A new dimensionless number tightly linked to the viscosity is the Reynolds number

$$\mathrm{Re} = \frac{\rho_0 u_0 L}{\mu_0} \tag{2.23}$$

which relates the inertial to viscous forces. For small Reynolds numbers the viscous forces dominate. The Reynolds number is also often used as an indicator whether a turbulent or a

laminar flow is present. What 'small' (or 'large') means in this setting highly depends on the problem analyzed.

The heat flux q is defined as

$$q = -\kappa \nabla T$$

with conductivity $\kappa > 0$ $[\frac{\mathrm{W}}{\mathrm{m}\cdot\mathrm{K}}]$. The negative sign results from heat being transferred from regions of high to regions of low temperature. The conductivity depends on the fluid type and can be derived from the Prandtl number, another dimensionless number:

$$\mathrm{Pr} = \frac{\mu c_p}{\kappa},$$

that is usually known. For air at moderate temperatures a value of $\mathrm{Pr} = 0.72$ is commonly assumed. The temperature T is now needed explicitly in the equation and can be computed using the definition of the internal energy (2.11) and the ideal gas law (2.10) to give

$$T = \frac{1}{c_v}\left(\frac{E}{\rho} - \frac{1}{2}\|u\|_2^2\right) = \frac{1}{c_v(\gamma-1)\rho}\left(E - \frac{1}{2}\rho\|u\|_2^2\right) = \frac{1}{(\gamma-1)c_v}\frac{p}{\rho}.$$

The pressure p is given by the ideal gas law (2.10).

As for the Euler equations, the Navier-Stokes equations can be written in flux formulation (2.3) as well. The convective flux f_c is the same as for the Euler equations (2.15) and the viscous flux

$$f_{v,1}(w) = \begin{pmatrix} 0 \\ \mathrm{S}_{11} \\ \mathrm{S}_{21} \\ \mathrm{S}_{11}u_1 + \mathrm{S}_{12}u_2 + \kappa\frac{\partial T}{\partial x_1} \end{pmatrix}, \quad f_{v,2}(w) = \begin{pmatrix} 0 \\ \mathrm{S}_{12} \\ \mathrm{S}_{22} \\ \mathrm{S}_{21}u_1 + \mathrm{S}_{22}u_2 + \kappa\frac{\partial T}{\partial x_2} \end{pmatrix} \tag{2.24}$$

has to be added. The equations include space derivatives up to second order and therefore the PDE is of second order. The problems are usually convection dominated, i.e. the viscous effects are small compared to convective effects.

2.7.1. Dimensional analysis

The dimensional analysis follows the same procedure as for the Euler equations in Sec. 2.6.1. First, reference values of the parameters are fixed to express the dimensionless parameters as

$$\begin{aligned} &\rho = \frac{\rho'}{\rho_0},\ \frac{\partial}{\partial x_i} = L\frac{\partial}{\partial x_{i'}},\ u_i = \frac{u_i'}{u_0},\ t = \frac{t'u_0}{L},\ p = \frac{p'}{p_0}, \\ &e = \frac{e'}{c_v T_0},\ \kappa = \frac{\kappa'}{\kappa_0},\ \mu = \frac{\mu'}{\mu_0}. \end{aligned} \tag{2.25}$$

These contain the same parameters as for the Euler equation (2.16), but has been extended by a reference internal energy $e_0 = c_v T_0$, a reference heat conductivity κ_0, and a reference viscosity μ_0.

Mass equation Using the dimensionless parameters as defined in (2.25) and plugging them into the mass equation (2.22b) one obtains the same representation as for the Euler equations, see Sec. 2.6.1. This is no surprise as mass conservation has to be fulfilled for both equations.

Momentum equation The analysis of the momentum equation is also similar to the analysis for the Euler equations where the added stress tensor makes the only difference. Therefore, the rescaling of the momentum equation

$$\begin{aligned}
&\frac{\partial(\rho' u')}{\partial t'} + \nabla' \cdot \left(\rho' u' \otimes u' + p' \mathrm{I}\right) = \nabla' \cdot \mathrm{S}' \\
\Leftrightarrow\ &\frac{\rho_0 u_0^2}{L} \frac{\partial(\rho u)}{\partial t} + \frac{\rho_0 u_0^2}{L} \nabla \cdot (\rho u \otimes u) + \frac{p_0}{L} \nabla \cdot (p\mathrm{I}) = \frac{\mu_0 u_0}{L^2} \nabla \cdot \mathrm{S} \\
\Leftrightarrow\ &\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u) + \frac{p_0}{\rho_0 u_0^2} \nabla \cdot (p\mathrm{I}) = \frac{\mu_0}{\rho_0 u_0 L} \nabla \cdot \mathrm{S} \\
\Leftrightarrow\ &\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u) + \frac{1}{\gamma \mathrm{Ma}^2} \nabla \cdot (p\mathrm{I}) = \frac{1}{\mathrm{Re}} \nabla \cdot \mathrm{S}
\end{aligned}$$

only reveals one additional dimensionless number that has not been present in the Euler equations, namely the Reynolds number (2.23).

Energy equation The dimensional analysis of the energy equation (2.22c) differs clearly from the dimensional analysis of the Euler equation due to the presence of the stress tensor and heat conduction. Moreover, the definition of a reference energy e_0 allows for a different dimensionless form. The energy equation is rescaled using the defined parameters

$$\begin{aligned}
&\frac{\partial E'}{\partial t'} + \nabla' \cdot \left((E' + p')u'\right) = \nabla' \cdot (\mathrm{S}' u') + \nabla' \cdot (\kappa' \nabla' T') \\
\Leftrightarrow\ &\frac{\rho_0 u_0 c_v T_0}{L} \frac{\partial E}{\partial t} + \frac{\rho_0 u_0 c_v T_0}{L} \nabla \cdot (Eu) + \frac{p_0 u_0}{L} \nabla \cdot (pu) = \frac{\mu_0 u_0^2}{L^2} \nabla \cdot (\mathrm{S}u) + \frac{\kappa_0 T_0}{L^2} \nabla \cdot (\kappa \nabla T)
\end{aligned}$$

such that all dimensions vanish. After eliminating the parameters in front of the time derivative

$$\frac{\partial E}{\partial t} + \nabla \cdot (Eu) + \frac{p_0}{\rho_0 c_v T_0} \nabla \cdot (pu) = \frac{\mu_0 u_0}{\rho_0 L c_v T_0} \nabla \cdot (\mathrm{S}u) + \frac{\kappa_0}{L \rho_0 u_0 c_v} \nabla \cdot (\kappa \nabla T)$$

three factors are obtained that can be expressed as dimensionless parameters. The first factor

$$\frac{p_0}{\rho_0 c_v T_0} = \frac{\rho_0 R T_0}{\rho_0 c_v T_0} = \frac{R}{c_v} = \gamma - 1$$

depends only on the dimensionless ratio of specific heats. The second factor reveals

$$\frac{\mu_0 u_0}{\rho_0 L c_v T_0} = \frac{\mu_0}{\rho_0 L u_0} \cdot \frac{u_0^2}{c_v T_0} = \frac{\gamma R}{c_v} \frac{\mu_0}{\rho_0 L u_0} \cdot \frac{u_0^2}{\gamma R T_0} = \gamma(\gamma - 1)\mathrm{Re}^{-1}\mathrm{Ma}^{-2},$$

which is the ratio of specific heats, the Reynolds number and the Mach number. The last factor

$$\frac{\kappa_0}{L\rho_0 u_0 c_v} = \frac{\mu_0}{L\rho_0 u_0}\frac{\kappa_0}{c_v \mu_0} = \frac{\mu_0}{L\rho_0 u_0}\frac{\kappa_0 \gamma}{c_p \mu_0} = \mathrm{Re}^{-1}\mathrm{Pr}^{-1}$$

introduces the Reynolds and the Prandtl number. Using these dimensionless numbers, the energy equation can be written as

$$\frac{\partial E}{\partial t} + \nabla \cdot (Eu) + (\gamma - 1)\nabla \cdot (pu) = \frac{\gamma(\gamma-1)}{\mathrm{Re} \cdot \mathrm{Ma}^2}\nabla \cdot (\mathrm{S}u) + \frac{1}{\mathrm{Re} \cdot \mathrm{Pr}}\nabla \cdot (\kappa \nabla T).$$

We have found four dimensionless numbers for the Navier-Stokes equations. This means that knowledge of the Mach number, Reynolds number, Prandtl number and the ratio of specific heats are sufficient to uniquely define the flow for the Navier-Stokes equations.

2.7.2. Boundary conditions

Most boundary conditions described for the Euler equations, see Section 2.6.2, hold also for the Navier-Stokes equations. The only exception is the wall boundary condition. The fluid still cannot penetrate the wall, but due to the viscous effects the wall influences the flow velocity in tangential direction. This is accounted for by using no-slip wall boundary conditions which enforce $u = 0$. Therefore, the boundary state is given by

$$w_{\mathrm{BC}}(w_{\mathrm{in}}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} w_{\mathrm{in}}.$$

On no slip walls an additional condition for the temperature has to be imposed. We choose adiabatic condition $\nabla T \cdot n = 0$, i.e. no heat flux through the wall. This is achieved by setting the first and last row of the viscous flow (2.24) to zero

$$f_v(w_{\mathrm{BC}})_{\mathrm{wall}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} f_v(w_{\mathrm{BC}}).$$

3. Space discretization

In this work we focus on the presented governing equations from the previous section that are discretized in space by a discontinuous Galerkin (DG) method. The first work published on discontinuous Galerkin method dates back to 1973 [193]. The method was picked up quickly and led to the development of interior penalty methods [10, 66], but was rarely used in CFD. It took almost twenty years until the DG method received major recognition in the CFD community. This was mainly after a paper series of Cockburn, Shu and co-authors [45, 47, 52, 57, 58] on solving hyperbolic problems with these schemes. They combined a discontinuous Galerkin space discretization with Runge-Kutta time stepping and approaches from finite volume methods for flux definition and stabilization. These schemes are often referred to as Runge-Kutta discontinuous Galerkin (RKDG) methods.

The approach has been generalized by Cockburn and Shu leading to the so called *local* discontinuous Galerkin (LDG) method [56, 247]. This method uses a mixed formulation [26] of the original problem to generate an explicit representation of higher space derivatives. Peraire and Persson introduced the *compact* discontinuous Galerkin (CDG) method by [179] which is closely related to LDG methods. Many other authors developed discontinuous Galerkin methods for CFD as well: Famous methods are the ones by Bassi and Rebay [20, 21], Douglas and Dupont [66] or Baumann and Oden [22, 23]. Hartmann and Houston showed that discontinuous Galerkin methods have nice properties for adaptive space discretizations [102, 105, 106].

Further information on discontinuous Galerkin methods can be found in the books of Hesthaven and Warburton [109], Di Pietro and Ern [62], the lecture notes of Hartmann [104], or in the review articles of Cockburn and co-authors [54, 55] and the references therein.

In this section, the space discretization is described for the considered problems in flux formulation

$$\begin{aligned} \partial_t w + \nabla \cdot (f_c(w) - f_v(w, \nabla w)) &= h(w, \nabla w) \quad \text{on} \quad (t,x) \in \Omega_T \\ w(0,x) &= w_0(x). \end{aligned} \tag{2.3 revisited}$$

After the introduction of some additional notation, we describe the general idea of discontinuous Galerkin methods. Then, we go further to the hybridized discontinuous Galerkin (HDG) methods that are the main concern of this thesis. For the HDG discretization, we explain the linearization and static condensation process in more detail in order to emphasize

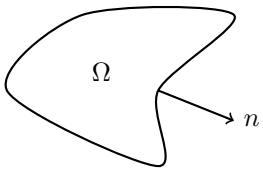


Figure 3.1.: An example domain Ω with outward pointing normal vector n.

the special properties of this scheme. We conclude this section with some remarks on the implementation of the method.

3.1. Notation

We introduce some additional notation for the description of the numerical method. These follow standard notation commonly used for discontinuous Galerkin methods as it can be found in articles and textbooks. For a domain Ω as depicted in Fig. 3.1 an outward pointing normal vector n is defined for every point of the domain boundary $\partial\Omega$. We assume that all normal vectors are normalized, i.e. $\|n\|_2 = 1$. The definition of the normal vector allows to define boundary values stemming from the inner and the outer of the domain for an arbitrary function v as

$$v^{\pm}(x) \coloneqq \lim_{\epsilon \to 0} v(x \pm \epsilon n), \quad x \in \partial\Omega.$$

Thus, the values v^- are referred to as inner values and v^+ as outer values. This is important as discontinuous Galerkin discretizations usually lead to two distinctive values of the approximated solution on (element) boundaries.

Based on this definition the average operator $\{\cdot\}$ and jump operator $[\![\cdot]\!]$ for a scalar $v \in \mathbb{R}$ or vector quantity $v \in \mathbb{R}^d$ are defined. The average operator $\{\cdot\}$ is given as

$$\{v\} = \frac{1}{2}(v^+ + v^-) \tag{3.1}$$

and acts on each component if $v \in \mathbb{R}^d$. The jump operator $[\![\cdot]\!]$ distinguishes more clearly between a scalar or a vector quantity. In the scalar case $v \in \mathbb{R}$, the operator is given as

$$v \in \mathbb{R}: \quad [\![v]\!] \coloneqq n^+ v^+ + n^- v^- = (v^- - v^+) n^- \in \mathbb{R}^d. \tag{3.2a}$$

In the case of a vector $v \in \mathbb{R}^d$ the jump operator is defined as

$$v \in \mathbb{R}^d: \quad [\![v]\!] \coloneqq n^+ \cdot v^+ + n^- \cdot v^- = \left(v^- - v^+\right) \cdot n^- \in \mathbb{R}. \tag{3.2b}$$

Note that the jump operator applied to a scalar quantity gives a vector quantity as result while applying the jump operator to a vector quantity results in a scalar quantity. We slightly

(a) Square domain Ω. (b) Domain divided into 32 subdomains T_k.

Figure 3.2.: A square domain Ω is partitioned into subdomains $T_1, \ldots, T_{32}$.

abuse the notation of the jump operator for multicomponent unknowns, i.e. $m > 1$. If the vector v has m components, the jump operator is defined as

$$v \in \mathbb{R}^m : \quad [\![v]\!] \coloneqq \begin{pmatrix} [\![v_1]\!]^{\mathrm{T}} \\ \vdots \\ [\![v_m]\!]^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{m \times d}. \tag{3.2c}$$

The definition refers to the component-wise application of the jump operator for scalars (3.2a). If the jump operator is applied to a quantity $v \in \mathbb{R}^{m \times d}$ that, for example, results from evaluating the flux functions f_c and f_v, see Sec. 2.2, it is defined as

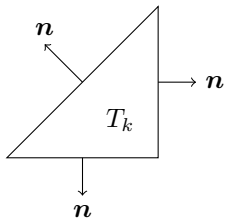
$$v \in \mathbb{R}^{m \times d} : \quad [\![v]\!] \coloneqq \begin{pmatrix} [\![(v_{11}, \ldots, v_{1d})^{\mathrm{T}}]\!] \\ \vdots \\ [\![(v_{m1}, \ldots, v_{md})^{\mathrm{T}}]\!] \end{pmatrix} \in \mathbb{R}^{m}. \tag{3.2d}$$

It means that the jump operator is applied to each row interpreted as vector.

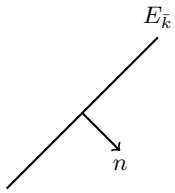
In general, we do not consider the domain as a whole, but a partitioning of the domain into K non-overlapping subdomains or elements T_k such that

$$\Omega = \bigcup_{k=1}^{K} T_k \tag{3.3}$$

holds, see Fig. 3.2. Such a partitioning is also called a mesh $\mathcal{T}_h = \cup_k T_k$ and contains all elements T_k. A mesh in 2D using triangles to subdivide a square domain is given in Fig. 3.2b. The characteristic size of the elements is given by the mesh size Δx which resembles the shortest side of all the triangular elements in the mesh. The boundary of an element T_k is referred to as ∂T_k and contains the surrounding edges. We also have edges $E_{\bar{k}}$ stemming from the subdivision. The set of all edges or faces is called $\mathcal{E}_h = \cup_{\bar{k}} E_{\bar{k}}$ and it contains $\overline{K} \coloneqq |\mathcal{E}_h|$ edges. This set of edges is sometimes also referred to as the skeleton of the mesh. Note that there is a difference when one refers to edges from the sets $\mathcal{E}_h$ and ∂T_k. In the set $\mathcal{E}_h$ each edge of the triangulation appears once. If one considers every element T_k and considers all



(a) Normal vectors on element edges/faces.



(b) Normal vector of an edge.

Figure 3.3.: A triangular domain with the corresponding normal vectors of each edge.

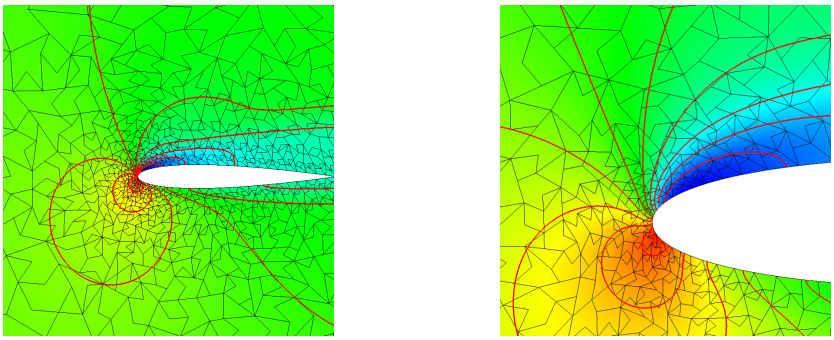
its edges in ∂T_k one visits inner edges of the skeleton twice because inner edges are shared by neighboring elements, see Fig. 3.2b.

For an element, we have an outward pointing normal vector defined for each side, see Fig. 3.3a. Additionally, we have a normal vector defined for each edge $E_{\bar{k}} \in \mathcal{E}_h$, see Fig. 3.3b. These figures show simplices as we deal with these type of elements if nothing else is stated. However, one reason for the popularity of discontinuous Galerkin methods is their ability to work on elements of (almost) arbitrary shape and it is also possible to combine different types of shapes in one mesh. A good example for this are DG methods working on agglomerated meshes [8, 18, 19, 37]. In Fig. 3.4 an example of a flow simulation on a mesh consisting of arbitrary polygons is given.

Agglomerated meshes can be easily generated by connecting neighboring elements of a 'standard' mesh consisting of simplices or quadrilaterals, for example, as generated by common meshing tools. This allows to solve a problem on a much coarser mesh which may lead to less unknowns. This applies also to the HDG method, but depends on the shape of polygons. The reduction of degrees of freedom depends on the ratio of the number of edges $\overline{K}$ to number of elements K, see Fig. 3.6. The HDG method reduces the number of unknowns most (compared to standard DG methods) if one has the least amount of edges per element. This is the case for simplices and gets worse for arbitrary polygons with increasing number of edges.

3.2. Discontinuous Galerkin methods

The discontinuous Galerkin method we present falls into the class of the local discontinuous Galerkin (LDG) methods [56, 216, 247]. In general, the problem (2.3) is a second order PDE due to the viscous terms. The problem is rewritten as a system of first order PDEs. For this, an auxiliary unknown $\sigma \coloneqq \nabla w$ is introduced. The auxiliary unknown allows to rewrite the



(a) Density distribution around a NACA0012 airfoil.

(b) Close up of the density distribution at the tip of a NACA0012 airfoil.

Figure 3.4.: Plot of the density solution of a flow around a NACA0012 airfoil (left) and a close up on the tip of the airfoil (right) computed with the HDG method. The mesh consists of arbitrarily shaped polygons. Free stream boundary conditions with Mach number $\mathrm{Ma} = 0.2$, Reynolds number $\mathrm{Re} = 500$ and an angle of attack of 10° are used. The red lines indicate isolines of the density and the black lines indicate the outline of the elements.

original problem as

$$\begin{aligned} \sigma - \nabla w &= 0 \\ \partial_t w + \nabla \cdot (f_c(w) - f_v(w, \sigma)) &= h(w, \sigma) \end{aligned} \tag{3.4}$$

and is often referred to as a problem in mixed formulation [26, 252]. Now, one seeks for a solution to this problem. The name of the discontinuous Galerkin method already indicates that one uses a Galerkin ansatz in a discontinuous setting. We assume that the domain Ω has been subdivided such that a suitable mesh $\mathcal{T}_h$ is available. Then, we define the finite dimensional function spaces

$$H_h := \{v \in [L^2(\Omega)]^{m \cdot d} \mid v_{|T_k} \in [\mathbb{P}_P(T_k)]^{m \cdot d},\ T_k \in \mathcal{T}_h\} \tag{3.5a}$$

$$V_h := \{v \in [L^2(\Omega)]^{m} \mid v_{|T_k} \in [\mathbb{P}_P(T_k)]^{m},\ T_k \in \mathcal{T}_h\} \tag{3.5b}$$

for the approximation of the unknowns $\sigma \approx \sigma_h \in H_h$ and $w \approx w_h \in V_h$. We use the index $\cdot_h$ to indicate discrete quantities. We choose polynomial functions spaces with $\mathbb{P}_P(T_k)$ being the polynomial space of degree P on the simplex T_k. The degree of the polynomial space refers to the *total* degree in this thesis. For a polynomial space in two space dimensions $d = 2$ a possible choice of the basis is $x_1^i x_2^j$. The polynomial space of total degree P contains all basis functions that fulfill $i + j \leq P$. The function spaces restrict the polynomial space $\mathbb{P}_P$ to the subdomains T_k of the mesh. Thus, continuity of the solution is only assumed on an element

as the solution is represented by a polynomial. Jumps in the solution between neighboring elements are allowed. We use the function spaces to seek for a solution of the problem in weak formulation which is common for Galerkin methods [26, 252].

In general, any polynomial basis can be used. However, the basis should be chosen carefully since in practice the choice can have a serious impact on the performance and accuracy of the scheme [62, 109]. In this work, we use Legendre polynomials as basis functions. The approximations to the solutions w_h and σ_h shall lie within these spaces. In the scalar case, i.e. m=1, one looks for polynomial representations of the unknown expressed as

$$\begin{aligned} \sigma_h^k(t,x) &= \sum_{i=1}^{N^\sigma} \Sigma_i^k(t)\tau_{h,i}^k(x), \quad x \in T_k, \\ w_h^k(t,x) &= \sum_{i=1}^{N} W_i^k(t)\varphi_{h,i}^k(x), \quad x \in T_k. \end{aligned} \tag{3.6}$$

The index k refers to the representation of the solution on the kth element T_k and the solution is reconstructed using a given set of N basis functions $\varphi_{h,i}^k$ defined for the element and a set of coefficients $W_i^k(t)$ that have to be determined. The equivalent holds for the auxiliary unknown and its N^σ basis functions $\tau_{h,i}^k$ and coefficient vector $\Sigma_i^k(t)$. This easily extends to vector quantities using the respective basis functions and introducing additional coefficients for each quantity. The representation (3.6) may differ slightly depending on the chosen polynomial space.

The unknowns approximating the solution are functions in time t and space x and the test functions are functions in space x. This does not change in the following and therefore the explicit dependence is not stated in order to keep the notation short.

Let us first restrict to the problem on a single element T_k. We seek a solution $(\sigma_h, w_h) \in H_h \times V_h$ to the problem (3.4). By multiplying the equation by test functions $(\tau_h, \varphi_h) \in H_h \times V_h$ and integrating over the element we obtain

$$\begin{aligned} \int_{T_k} \sigma_h \tau_h dx - \int_{T_k} \nabla w_h \tau_h dx &= 0 \\ \int_{T_k} \partial_t w_h \varphi_h dx + \int_{T_k} \nabla \cdot (f_c(w_h) - f_v(w_h, \sigma_h))\varphi_h dx &= \int_{T_k} h(w_h, \sigma_h)\varphi_h dx. \end{aligned}$$

Subsequently, we integrate by parts to remove the requirement of the solution to be differen-

tiable. Thus, we obtain the weak formulation

$$\int_{T_k} \sigma_h \tau_h dx + \int_{T_k} w_h \nabla \cdot \tau_h dx - \int_{\partial T_k} \widehat{w} \tau_h^- \cdot n ds - \int_{\partial T_k \cap \partial\Omega} w_{\mathrm{BC}} \tau_h^- \cdot n ds = 0 \tag{3.8a}$$

$$\int_{T_k} \partial_t w_h \varphi_h dx - \int_{T_k} (f_c - f_v) \cdot \nabla \varphi_h dx + \int_{\partial T_k} (\widehat{f_c} - \widehat{f_v}) \cdot n \varphi_h^- ds + \int_{\partial T_k \cap \partial\Omega} (\widehat{f_c^{\mathrm{BC}}} - \widehat{f_v^{\mathrm{BC}}}) \cdot \varphi_h^- n ds = \int_{T_k} h \varphi_h dx \tag{3.8b}$$

that has to hold for all test functions $(\tau_h, \varphi_h) \in H_h \times V_h$. As there is a finite number of test functions this creates a finite set of equations that can be solved. We do not mention the explicit dependency of the fluxes and source term on σ_h and w_h in (3.8) to keep the representation compact.

The weak formulation contains two different kinds of integrals on element boundaries ∂T_k: Integrals over edges that do not intersect the domain boundary $\partial T_k \setminus \partial\Omega$ and such integrals where the element boundary intersects with the domain boundary $\partial T_k \cap \partial\Omega$. The integrals inside the domain Ω couple neighboring elements while the integrals on the the domain boundary incorporate boundary conditions specified by the problem studied, see Sec. 2. The global solution is given by putting all local solutions together by summing over all elements T_k. Further explanations of weak formulations and finite element methods, especially discontinuous Galerkin methods, can be found in text books and lecture notes [26, 62, 104, 109, 252] and in references therein. Note that there are no continuity restrictions on the solution between elements and jumps in the solution between two neighboring elements T_k and $T_{k'}$ are admissible even if the true solution of the problem is continuous. Thus, discontinuous Galerkin methods fall into the class of non-conforming methods. The evaluation of functions on boundary integrals requires special treatment and is denoted by $\widehat{\cdot}$.

The coupling between elements is achieved solely through integrals over element boundaries ∂T_k where the convective flux $\widehat{f_c}$, viscous flux $\widehat{f_v}$ and $\widehat{w}$ have to be evaluated. In order to define these in a manner that leads to a stable, consistent, and conservative approximation one uses numerical flux functions denoted by $\widehat{\cdot}$. This idea is closely related to finite volume methods where the solution is also allowed to be discontinuous between elements. In the setting of the local discontinuous Galerkin methods the numerical flux $\widehat{f_c}$ has to obey requirements known from finite volume methods; namely the flux must be consistent, conservative, and monotone [143, 145, 229]. A possible choice is a (local/global) Lax-Friedrichs/Rusanov flux [55, 138, 143, 200, 229]

$$\widehat{f_c}(w_h^+, w_h^-) \coloneqq \{f_c(w_h)\} - \frac{\alpha_c}{2} [\![w_h]\!] \tag{3.9}$$

for the convective flux. The stabilization parameter α_c has to be chosen carefully to obtain a stable scheme and should be at least of the size of maximum absolute value of the local eigenvalue of $\left(\frac{\partial}{\partial w} f_c(w^{\pm})\right) \cdot n$ on the edge.

The remaining flux functions have to be chosen for the viscous flux and the auxiliary equation. Flux functions of convective problems usually stem from hyperbolic problems where one has some knowledge about the propagation direction. Thus, the fluxes are constructed to respect this direction by *upwinding*. Diffusive (elliptic) problems do not have a dominating direction of information propagation. Therefore, local discontinuous Galerkin methods define the numerical fluxes as

$$\begin{aligned} \widehat{w}(w_h^+, w_h^-, \sigma_h^+, \sigma_h^-) &:= \{w_h\} - C_{12}[\![w_h]\!] + C_{22}[\![f_v(w_h, \sigma_h)]\!] \\ \widehat{f}_v(w_h^+, w_h^-, \sigma_h^+, \sigma_h^-) &:= \{f_v(w_h, \sigma_h)\} + C_{11}[\![w_h]\!] - C_{12}[\![f_v(w_h, \sigma_h)]\!] \end{aligned} \tag{3.10}$$

with $C_{11} > 0$ and $C_{22} \geq 0$ being scalars and C_{12} being a vector [53, 56]. The choice of these parameters determines the stability and convergence of the scheme. In the literature, a number of different choices are discussed, see e.g. [53, 55, 181, 247]. In general, the idea is to fix a direction of propagation and invert the direction between the unknown and auxiliary unknowns.

In order to introduce the global problem we use the following abbreviations

$$\begin{aligned} (g_1, g_2)_{\mathcal{T}_h} &:= \sum_{k=1}^{K} \int_{T_k} g_1 \cdot g_2 dx \\ (g_1, g_2)_{\partial\mathcal{T}_h^{\mathrm{in}}} &:= \sum_{k=1}^{K} \int_{\partial T_k \setminus \partial\Omega} g_1 \cdot g_2 ds, \quad (g_1, g_2)_{\partial\mathcal{T}_h^{\mathrm{BC}}} := \sum_{k=1}^{K} \int_{\partial T_k \cap \partial\Omega} g_1 \cdot g_2 ds \end{aligned} \tag{3.11}$$

for the summation of inner products. This allows for a more compact notation. Then, the global version of the semi-discrete problem (3.8) is to find a solution $(\sigma_h, w_h) \in H_h \times V_h$ such that for all test functions $(\tau_h, \varphi_h) \in H_h \times V_h$

$$\begin{aligned} (\sigma_h, \tau_h)_{\mathcal{T}_h} + (w_h, \nabla \cdot \tau_h)_{\mathcal{T}_h} - (\widehat{w}, \tau_h^- \cdot n)_{\partial\mathcal{T}_h^{\mathrm{in}}} \\ -(w_{\mathrm{BC}}, \tau_h^- \cdot n)_{\partial\mathcal{T}_h^{\mathrm{BC}}} = 0 \\ (\partial_t w_h, \varphi_h)_{\mathcal{T}_h} - ((f_c - f_v), \nabla\varphi_h)_{\mathcal{T}_h} + ((\widehat{f_c} - \widehat{f_v}) \cdot n, \varphi_h^-)_{\partial\mathcal{T}_h^{\mathrm{in}}} \\ +((\widehat{f_c^{\mathrm{BC}}} - \widehat{f_v^{\mathrm{BC}}}) \cdot n, \varphi_h^-)_{\partial\mathcal{T}_h^{\mathrm{BC}}} = (h, \varphi_h)_{\mathcal{T}_h} \end{aligned}$$

holds with suitable numerical fluxes defined as in (3.9) and (3.10).

Remark 3 *The number of total unknowns is greatly increased for large m and d due to the auxiliary unknowns. These can be removed by applying local lifting operators [179] which in turn increase the cost of the computations.*

3.3. Hybridized discontinuous Galerkin methods

The HDG methods have been introduced in 2009 [50, 167, 168] while the main ideas, namely the discontinuous Galerkin method [193] and the ideas to reduce the number of globally coupled unknowns [97, 251] were already developed in the 1970's and 1960's. The methods have been quickly adopted by the community and extended to linear and nonlinear problems such as from fluid dynamics [72, 163, 165, 167–169, 211, 237]. Nowadays, HDG discretizations also exist for many different problems outside of classical CFD applications such as a higher order (in terms of space derivatives) PDEs like the Korteweg-de Vries equation [65, 203], porous media [70, 83, 84, 202], magnetohydrodynamics [44, 146, 170, 232, 250], elasticity [129, 219, 235, 245], or wave equations [89, 90, 93, 206, 222], for example. Moreover, researchers are concerned with space-time HDG methods [197, 198], efficient implementation [124, 199, 202], HDG-DG methods for implicit-explicit (IMEX) discretizations [130], and its efficiency compared to other numerical discretizations [112, 124, 132, 246]. Additional information can be found, e.g., in the introductory paper of Huerta et al. [215] or the review paper of Cockburn [48] and the respective references therein.

The HDG methods have been introduced because the class of 'classical' discontinuous Galerkin methods such as the one introduced in the previous section have one major drawback. The local stencil of the solution without continuity requirements introduces a large number of unknown coefficients to be determined when compared to other numerical methods. Then, if an implicit solution technique is chosen, the degrees of freedom become globally coupled leading to a matrix of considerable size that must be inverted. This may limit the application of DG schemes to problems of engineering size due to memory and runtime requirements. However, one often wants to use an implicit solution techniques. A common example are steady state problems where implicit schemes converge quickly to the steady state solution while explicit schemes would not. Time-dependent problems can also benefit from implicit time-stepping if the problem is stiff. This is usually the case for problems in CFD due to the diffusive nature of the Navier-Stokes equations, diffusive stabilization techniques for the Euler equations or anisotropic meshes.

The core idea of *hybridized* or *hybridizable* discontinuous Galerkin methods [50, 69, 165, 167–169, 211] is the introduction of an additional *hybrid unknown* λ that only exists on the element boundaries. Although this gives rise to more degrees of freedom in total, the hybrid unknown makes the resulting system of equations amenable to static condensation. After applying static condensation the system of equations is only globally coupled in this hybrid unknown λ. A brief sketch of this process is given in Fig. 3.5. Static condensation and its implications for the method are discussed in greater detail in Section 3.3.2. The current section focuses solely on the introduction of the numerical scheme.

We can reuse many of the terms introduced for the LDG method in the previous section,

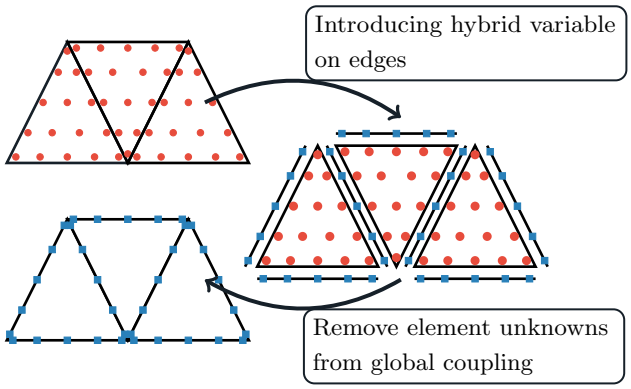


Figure 3.5.: The (simplified) process of removing the globally coupled unknowns stemming from the DG discretization (•) by introducing hybrid unknowns (▪).

see Sec. 3.2. For the hybrid unknown λ_h we define an additional function space

$$M_h := \{f \in [L^2(\mathcal{E}_h)]^m \mid f_{|E_{\bar{k}}} \in [\mathbb{P}_P(E_{\bar{k}})]^m,\ E_{\bar{k}} \in \mathcal{E}_h\} \tag{3.13}$$

that defines the piecewise polynomial functions on edges $E_{\bar{k}}$. The hybrid unknown is then expressed in terms of basis functions $\mu_h \in M_h$

$$\lambda_h^{\bar{k}}(t,x) = \sum_{i=1}^{\overline{N}} \Lambda_i^{\bar{k}} \mu_{h,i}^{\bar{k}}(x), \quad x \in E_{\bar{k}}. \tag{3.14}$$

similar to the auxiliary unknown σ_h and unknown w_h in (3.6). The only difference, is that a solution is approximated on the edge $E_{\bar{k}}$.

Again, this is a discontinuous approximation, i.e. there is no continuity requirement of the hybrid unknown. This means, that at vertices where several edges touch, the values of λ_h from different edges may differ. A special class of methods that enforce a continuous or a mixed continuous/discontinuous representation of the hybrid unknown are the so called embedded discontinuous Galerkin methods (EDG) and interior embedded discontinuous Galerkin methods (IEDG) [51, 72, 98, 166, 178]. However, these methods are out of the scope of this work.

The discretization of the problem (2.3) follows the same procedure as in the previous section, see Sec. 3.2. The equation is rewritten as a system of first order PDEs by introducing the auxiliary unknown σ_h. Then, the equations are multiplied by test functions $(\tau_h, \varphi_h, \mu_h) \in$

$H_h \times V_h \times M_h$, integrated by parts and summed up over all elements and edges to obtain

$$(\sigma_h, \tau_h)_{\mathcal{T}_h} + (w_h, \nabla \cdot \tau_h)_{\mathcal{T}_h} - (\widehat{w}, \tau_h^- \cdot n)_{\partial \mathcal{T}_h^{\text{in}}} - (w_{\text{BC}}, \tau_h^- \cdot n)_{\partial \mathcal{T}_h^{\text{BC}}} = 0 \tag{3.15a}$$

$$\begin{aligned}(\partial_t w_h, \varphi_h)_{\mathcal{T}_h} - ((f_c - f_v), \nabla \varphi_h)_{\mathcal{T}_h} + ((\widehat{f_c} - \widehat{f_v}), n\varphi_h^-)_{\partial \mathcal{T}_h^{\text{in}}} \\ + ((\widehat{f_c^{\text{BC}}} - \widehat{f_v^{\text{BC}}}), n\varphi_h^-)_{\partial \mathcal{T}_h^{\text{BC}}} = (h, \varphi_h)_{\mathcal{T}_h}\end{aligned} \tag{3.15b}$$

$$\begin{aligned}([\![\widehat{f_c} - \widehat{f_v}]\!], \mu_h)_{\mathcal{E}_h^{\text{in}}} + \alpha(\lambda_h - w_{\text{BC}}, \mu_h)_{\mathcal{E}_h^{\text{BC}}} \\ + ((f_v - f_v^{\text{BC}}) \cdot n, \mu_h)_{\mathcal{E}_h^{\text{BC}}} = 0.\end{aligned} \tag{3.15c}$$

The first two equations look in fact almost identical to (3.8). However, there are two main differences. Firstly, different numerical flux functions are used. As numerical flux function in the auxiliary equation (3.15a) we use

$$\widehat{w} \coloneqq \lambda_h. \tag{3.16}$$

The remaining flux functions are modified (global) Lax-Friedrichs/Rusanov [165] fluxes

$$\widehat{f_c}(\lambda_h, w_h^-; n) = f_c(\lambda_h) + \alpha_c(\lambda_h - w_h^-)n^{\text{T}} \tag{3.17a}$$

$$\widehat{f_v}(\lambda_h, w_h^-, \sigma_h^-; n) = f_v(\lambda_h, \sigma_h^-) - \alpha_v(\lambda_h - w_h^-)n^{\text{T}} \tag{3.17b}$$

where $\alpha_c > 0$ and $\alpha_v > 0$ are stabilization parameters. The fluxes have been modified to use the hybrid unknown λ_h and data local (σ_h^-, w_h^-) to an element T_k. Thus, the fluxes over an edge of this element can be evaluated without the knowledge about the solution (σ_h^+, w_h^+) on the neighboring element. The convective stabilization coefficient α_c has to be set (at least) as large as the largest eigenvalue of the matrix described in (2.5). This is identical to the Lax-Friedrichs flux (3.9). For the global Lax-Friedrichs/Rusanov flux we choose the largest eigenvalue in the computational domain $\mathcal{T}_h$.

The viscous stability parameter follows from the diffusive processes. For the Navier-Stokes equations the stability parameter can be set to $\alpha_v = \frac{\gamma}{\text{Pr} \cdot \text{Re}}$ as suggested in [165]. The non-dimensional numbers refer to the dominance of diffusive processes. The authors also argue that α_v is usually very small for compressible flows as the Reynolds number tends to be large. As the stabilization term of the numerical fluxes only differ in the stabilization parameter we can write the difference of the numerical fluxes as

$$\widehat{f_c}(\lambda_h, w_h^-) - \widehat{f_v}(\lambda_h, w_h^-, \sigma_h^-) = f_c(\lambda_h) - f_v(\lambda_h, \sigma_h^-) + \alpha(\lambda_h - w_h^-)n^{\text{T}}$$

with a combined stability coefficient $\alpha \coloneqq \alpha_c + \alpha_v$.

The second difference is the third equation (3.15c) that has been added. This equation is needed to uniquely define the hybrid unknown and is chosen such that fluxes on edges are conservative in the normal direction. For a compact notation of the HDG method (3.15)

we use the shorthand notation for inner products introduced in Eq. (3.11) and additionally define additional shorthand notations of inner produces on element edges

$$(g_1, g_2)_{\mathcal{E}_h^{\text{in}}} \coloneqq \sum_{\bar{k}=1}^{\overline{K}} \int_{E_{\bar{k}} \setminus \partial\Omega} g_1 \cdot g_2 ds, \quad (g_1, g_2)_{\mathcal{E}_h^{\text{BC}}} \coloneqq \sum_{\bar{k}=1}^{\overline{K}} \int_{E_{\bar{k}} \cap \partial\Omega} g_1 \cdot g_2 ds. \tag{3.18}$$

In order to allow for an even more compact notation the some abbreviations are defined. We group the function spaces as $\mathbb{X}_h \coloneqq H_h \times V_h \times M_h$. In a similar fashion, the abbreviated vector of test functions $\mathbf{x}_h \in \mathbb{X}_h$ is defined as $\mathbf{x}_h(x) \coloneqq (\tau_h(x), \varphi_h(x), \mu_h(x))$ and the abbreviated solution vector $\mathbf{w}_h(t,x) \coloneqq (\sigma_h(t,x), w_h(t,x), \lambda_h(t,x))$ are defined. Similar to the previous section, see Sec. 3.2, the combined approximated solutions $\mathbf{w}_h$ and the combined test functions $\mathbf{x}_h$ are functions in time and space, respectively. Again, we do not state the dependence in time t and space x explicitly to keep the notation short. Then, we write the global semi-discrete problem (3.15) simply as

$$\partial_t \mathcal{T}(\mathbf{w}_h, \mathbf{x}_h) = \mathcal{N}(\mathbf{w}_h, \mathbf{x}_h). \tag{3.19}$$

The term $\mathcal{T}(\mathbf{w}_h, \mathbf{x}_h)$ refers to terms incorporating the time derivative of the solution

$$\partial_t \mathcal{T}(\mathbf{w}_h, \mathbf{x}_h) = (0, (\partial_t w_h, \varphi_h)_{\mathcal{T}_h}, 0)^{\mathrm{T}} \tag{3.20}$$

and $\mathcal{N}(\mathbf{w}_h, \mathbf{x}_h)$ collects all remaining, possibly nonlinear, terms. This notation shows that the discretized system of equations is a system of ordinary differential equations (ODEs). The system of equations can now be discretized in time by a suitable time integration scheme. Particular choices and the reasoning are presented in the chapters Ch. 4–6. The absence of time derivatives of the auxiliary unknown σ_h and the hybrid unknown λ_h leads to a differential algebraic equation (DAE). The differential algebraic structure requires a careful choice of the time integrator [101].

Remark 4 *In case a first order PDE is discretized, i.e. $f_v \equiv 0$, the auxiliary variable σ_h is not needed. This means equation* (3.15a) *is removed and the remaining equations* (3.15b) *and* (3.15c) *simplify accordingly.*

Remark 5 *We expect that the method has optimal convergence in $\|\cdot\|_{L^2}$-norm [165, 211] for smooth problems. This means the method should converge with order $P+1$ in space for the unknown w_h including and auxiliary unknown σ_h. In the setting of time dependent problems, the observed convergence rate also depends on the chosen time step size Δt and the order of convergence of the chosen time integrator.*

3.3.1. Linearization of nonlinear problems

The system of equations (3.19) is nonlinear when discretizing Euler or Navier-Stokes equations. Therefore, the arising system of equations cannot be solved directly. In this work, we solve the nonlinear problem by applying a (damped) Newton-Raphson method [61], often also called Newton's method. We assume that the semi-discrete problem has been discretized in time, e.g. by an implicit Euler method. Then, the system of equations can be written as

$$\mathcal{T}(\mathbf{w}_h^{n+1}, \mathbf{x}_h) - \mathcal{T}(\mathbf{w}_h^{n}, \mathbf{x}_h) = \Delta t \, \mathcal{N}(\mathbf{w}_h^{n+1}, \mathbf{x}_h)$$

with n denoting the current time level, $n+1$ being the time level where a new solution has to be computed for and Δt being the time step size $\Delta t = t^{n+1} - t^n$. We can rewrite this system of equations as $\mathbf{N}$

$$\mathbf{N}(\mathbf{w}_h^{n+1}) = 0 \tag{3.21}$$

such that one has to find a root of the system of equations. Newton's method solves this kind of problem by a series of linear problems. For a given start value $\mathbf{w}_h^{n+1,0}$ one obtains an approximation of the root by solving a sequence of equations

$$\mathbf{N}'(\mathbf{w}_h^{n+1,l}) s^l = -\mathbf{N}(\mathbf{w}_h^{n+1,l}) \tag{3.22}$$

in order to update the approximation to the root

$$\mathbf{w}_h^{n+1,l+1} = \mathbf{w}_h^{n+1,l} + s^l.$$

This procedure is repeated until an abort criteria is reached. A common criteria is a user-specified tolerance tol that specifies when the the update s^l is small $\|s^l\|_2 < \text{tol}$ such that the Newton method is stopped. In practice, one also defines an upper $k_{\max}$ for l at which the computations are aborted.

The construction of the matrix $\mathbf{N}'$, also called the Jacobian, requires to differentiate $\mathbf{N}$ with respect to the unknown $\mathbf{w}_h$. This can be done either exactly or by a suitable approximation that does not tamper the accuracy and convergence of Newton's method.

For high-order methods such as the discontinuous Galerkin method the matrices are usually large, but have only few nonzero entries per row. Such matrices are called *sparse*. The direct inversion of these matrices is undesirable in most cases. Although the matrix is sparse its inverse is usually a dense matrix with many nonzeros. Thus, the inversion would be memory and time consuming. A common approach is to solve this system by an iterative method from the class of Krylov methods that solves the system approximately [36, 152]. In this case, the method is also called a Newton-Krylov method. If we do not specify anything different, the system is solved using a restarted (F)GMRES method [201, 220].

The fact that the linear system of equations is solved approximately emphasizes also the need to specify a tolerance tol for the update s^l. One cannot expect that the method

converges with $s^l = 0$ on a computer with finite precision where the Jacobian may have been computed approximately and the resulting linear system has been solved with finite accuracy.

A practical problem may be that Newton's method only converges to a root if the starting value is already close enough to the root and the Jacobian is accurate and invertible. If this is fulfilled, the method converges with second order to the root. Moreover, for time-dependent problems the differences of the solution between time steps are often small and thus the solution at the old time level $\mathbf{w}_h^n$ is usually a good starting value for the Newton method.

However, in cases where the method does not properly converge or where one wants to speed up the nonlinear solver we introduce a dual time-stepping method as introduced by Jameson [116, 117]. The solution of (3.21) is interpreted as a steady state problem that is solved by

$$\frac{\partial \mathbf{w}_h^{n+1}}{\partial \tau} + \mathbf{N}(\mathbf{w}_h^{n+1}) = 0 \tag{3.23}$$

with a pseudo time τ. The dual-time stepping approach has the advantage that one can use techniques from steady-state frameworks to increase the convergence speed. Applying the implicit Euler and Newton's scheme to this equation gives

$$\left(\mathbf{N}'(\mathbf{w}_h^{n+1,l}) + \frac{1}{\Delta\tau^l}\mathrm{I}\right) s^l = -\mathbf{N}(\mathbf{w}_h^{n+1,l}) \tag{3.24}$$

with $\Delta\tau^l$ being the pseudo time step that does not relate to the physical time step Δt. The method can also be referred to as a damped Newton's method. The pseudo time step should depend on the solution $\mathbf{w}_h^{n+1,l}$ and $\Delta\tau^l \to \infty$ for $l \to \infty$ such that solution converges towards the solution of the initial method (3.22). In this work, we choose $\Delta\tau^l \coloneqq \mathrm{C}^l \cdot \Delta x$ with

$$\mathrm{C}^l = \begin{cases} c_0 \left[-2\left(\frac{l}{l_0}\right)^3 + 3\left(\frac{l}{l_0}\right)^2\right], & l \le l_0, \|\mathbf{N}(\mathbf{w}_h^{n+1,l-1})\|_2 > \mathrm{tol_C} \\ \mathrm{C}^{l-1}\left[1 + c_1 \max\left(0, \log\left(\frac{\|\mathbf{N}(\mathbf{w}_h^{n+1,l-1})\|_2}{\|\mathbf{N}(\mathbf{w}_h^{n+1,l-1})\|_2}\right)\right)\right], & \text{otherwise} \end{cases}$$

where Δx is the element size, l_0 is a user-defined iteration index and $\mathrm{tol_C}$ is a user-defined tolerance for which C should be ramped. After l_0 is exceeded or the residual is small enough, i.e. one is close to the solution, C is increased quickly in order to converge to the solution of the initial problem (3.22). The ramping is inspired by the approach discussed by May et al. [152]. Further information about the ramping procedure used in this thesis, the user-defined parameters c_0 and c_1, and its influence on steady state solution can be found in [209].

Note that the dual-time stepping scheme may harm the time accuracy if the inner iterations are not fully converged. Thus, we solve the arising linear system of equations with an iterative scheme, but with small tolerance. If a steady problem is solved, we can also use the approach to introduce a pseudo time. In this case no physical meaningful time t is present, but the discretized problem can still be expressed as in (3.21).

3.3.2. Static condensation

After the system of equations (3.15) has been discretized in time one obtains the fully discrete set of equations. Independent of whether one chooses the dual time-stepping approach (3.24) or the standard Newton's method (3.22) the resulting system can be written in matrix vector notation as

$$\begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} & \mathrm{A}_{\sigma\lambda} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} & \mathrm{A}_{w\lambda} \\ \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} & \mathrm{A}_{\lambda\lambda} \end{pmatrix} \begin{pmatrix} \Sigma \\ W \\ \Lambda \end{pmatrix} = \begin{pmatrix} R_\sigma \\ R_w \\ R_\lambda \end{pmatrix}. \tag{3.25}$$

The system matrix consists of blocks that refer to the equations of the auxiliary unknown $\mathrm{A}_{\sigma(\cdot)}$ (3.15a), the unknown $\mathrm{A}_{w(\cdot)}$ (3.15b) and the hybrid unknown $\mathrm{A}_{\lambda(\cdot)}$ (3.15c). The second index shows to which unknown the block is referring, so $\mathrm{A}_{\sigma w}$ refers to the matrix that stems from terms in the equation of the auxiliary unknown that include the unknown w_h. The vector $R_{(\cdot)}$ refers to the residual vectors of the corresponding equations and the vectors Σ, W, and Λ hold the coefficients needed to represent the approximated solutions. If Newton's method is used these vectors contain updates of the coefficients instead of the full coefficients of the solutions. The system of equations is considerably larger than for standard DG methods where all matrices and vectors referring to the hybrid unknown would vanish. However, the structure of the system of equations allows to solve the system in an efficient way. This process is often referred to as static condensation and the idea has been introduced already in 1965 for continuous Galerkin discretizations [97]. In the same year de Veubeke demonstrated how a continuous Galerkin method could be hybridized [251]. His discretization loosened the continuity restrictions of the unknowns and re-enforced the restriction through a hybrid unknown leading to a linear system of equations which is amenable to static condensation. This method has been further analyzed by several authors [9, 25, 49] who used the solution of the hybrid unknown for a post-processing procedure to increase the accuracy of the other unknowns. In the work of Cockburn et al. [51] a short overview over the history of hybridization of finite element methods and discontinuous Galerkin methods is given. In the same publication the authors introduced a unified framework of this concept for different numerical discretizations. This included the discontinuous Galerkin methods and thus leading to *hybridized* or *hybridizable* discontinuous Galerkin methods [50, 165, 167–169, 211]. A broad overview over the hybridization of numerical methods and static condensation can be found in a recent review paper of Cockburn [48]. As mentioned in this paper, the static condensation process is closely related to the Schur complement [249].

Now, to condense the system of equations (3.25) it can be interpreted as two systems. One system

$$\underbrace{\begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} \end{pmatrix}}_{=:\mathrm{A}_{\mathrm{loc}}} \begin{pmatrix} \Sigma \\ W \end{pmatrix} + \begin{pmatrix} \mathrm{A}_{\sigma\lambda} \\ \mathrm{A}_{w\lambda} \end{pmatrix} \Lambda = \begin{pmatrix} R_\sigma \\ R_w \end{pmatrix} \tag{3.26}$$

that stems from the original problem and the equation

$$\begin{pmatrix} \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} \end{pmatrix} \begin{pmatrix} \Sigma \\ W \end{pmatrix} + \mathrm{A}_{\lambda\lambda}\Lambda = R_\lambda \tag{3.27}$$

stemming from the hybrid equation. The first system (3.26) can be solved:

$$\begin{pmatrix} \Sigma \\ W \end{pmatrix} = \underbrace{\begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} \end{pmatrix}^{-1} \begin{pmatrix} R_\sigma \\ R_w \end{pmatrix}}_{=:L_1} - \underbrace{\begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} \end{pmatrix}^{-1} \begin{pmatrix} \mathrm{A}_{\sigma\lambda} \\ \mathrm{A}_{w\lambda} \end{pmatrix}}_{=:\mathrm{L}_2} \Lambda, \tag{3.28}$$

with the vector L_1 and matrix L_2. These are often referred to as *local solves* or *local solvers*. By reordering the coefficients of the vector $(\Sigma, W)^{\mathrm{T}}$ such that coefficients of elements are grouped, i.e. $(\Sigma_{T_1}, W_{T_1}, \ldots, \Sigma_{T_K}, W_{T_K})^{\mathrm{T}}$, it shows that the matrix $\mathrm{A}_{\mathrm{loc}}$ is block diagonal. This means the the system matrix consists of square matrices on its diagonal. Moreover, the residuals and matrices assembled in (3.26) can be assembled locally and also $\mathrm{A}_{\mathrm{loc}}$ can be inverted locally. This leads to many small systems of equations that need only little memory and the systems of equations can be solved independently. Therefore, the matrix blocks can be inverted in parallel.

Now, the local solves can be used to eliminate the dependency on $(\Sigma, W)^{\mathrm{T}}$ in the second equation (3.27) as

$$\begin{pmatrix} \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} \end{pmatrix} L_1 - \begin{pmatrix} \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} \end{pmatrix} \mathrm{L}_2\Lambda + \mathrm{A}_{\lambda\lambda}\Lambda = R_\lambda$$

which gives

$$\underbrace{\left(-\begin{pmatrix} \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} \end{pmatrix} \mathrm{L}_2 + \mathrm{A}_{\lambda\lambda}\right)}_{=:\mathrm{G}_{\mathrm{HDG}}} \Lambda = R_\lambda - \begin{pmatrix} \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} \end{pmatrix} L_1 \tag{3.29}$$

as globally coupled system. Therefore, assembling and solving the globally coupled system (3.25) is done in three steps that avoid constructing the matrix as a whole:

1. Compute the local solves using (3.26).
2. Solve the globally coupled system (3.29).
3. Update the local solutions using the local solves and the updated hybrid solution using (3.26).

Depending on the memory requirements one can either save the local solves L_1 and L_2 or recompute them when the solution is reconstructed.

We want to emphasize that the number of hybrid unknowns λ_h does not depend on the introduction of the auxiliary unknown σ_h. Thus, it is much less tempting to remove the auxiliary unknowns by lifting operators. The auxiliary unknown increases the assembly and

solving cost of the local solves and the globally coupled system of equations since additional terms have to be incorporated. On the other hand the lifting operator would also introduce a large number of smaller problems to be solved that requires computational work. Another way to avoid the additional unknowns would be the use of a different flux like an interior penalty formulation.

3.3.3. Consequences of the static condensation process

We want to briefly present the benefits possible by using a hybridized discontinuous Galerkin method compared to 'classical' discontinuous Galerkin methods. Consider a simple square domain as shown in Fig. 3.2a that is represented by a uniform mesh of triangles. Such a triangular mesh is shown in Fig. 3.2b. The mesh can be viewed as consisting of K_{x_1} square elements in the x_1-direction and K_{x_2} square elements in the x_2-direction. These square elements are then divided into two triangles. For uniform meshes of a square we have $K_{x_1} = K_{x_2}$ and thus the mesh contains $K = 2K_{x_1}K_{x_2} = 2 \cdot (K_{x_1})^2$ triangles. In this case, the mesh contains

$$\overline{K}(K) = \underbrace{4\sqrt{\frac{K}{2}}}_{\text{edges on } \partial\Omega} + \underbrace{2\left(\sqrt{\frac{K}{2}} - 1\right)\left(\sqrt{\frac{K}{2}}\right)}_{\text{vertical and horizontal edges}} + \underbrace{\frac{K}{2}}_{\text{diagonal edges}} = 2\sqrt{\frac{K}{2}} + \frac{3}{2}K$$

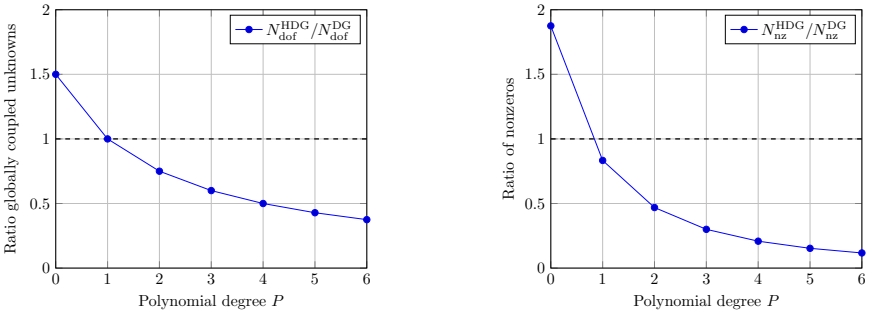
unique edges and is a function of K. The asymptotic ratio between the number of elements to edges in the mesh is

$$\lim_{K\to\infty} \frac{K}{\overline{K}(K)} = \lim_{K\to\infty} \frac{K}{2\sqrt{\frac{K}{2}} + \frac{3}{2}K} = \frac{2}{3}.$$

It is clearly visible that a triangular mesh as choses as example has always more edges than elements. In the setting of 'classical' DG methods the number of globally coupled unknowns stem from the degrees of freedom on elements while for the hybridized DG method the globally coupled unknowns stem from the degrees of freedom on edges. Even though the number of edges is typically larger than the number of elements the total number of globally coupled unknowns can be lower for hybridized DG methods because the number of degrees of freedom increase slower for increasing polynomial degree P. The number of unknowns N on a simplex is given by

$$N(d, P) = \Pi_{i=1}^{d} \frac{(P+i)}{i} \tag{3.30}$$

and depends on the space dimension d and polynomial degree P of the chosen polynomial representation [62]. On a triangle ($d = 2$) one has $N(2, P) = \frac{(P+1)(P+2)}{2}$ while on a edges ($d = 1$) one has $N(1, P) = (P + 1)$. Thus, the local number of degrees per freedom grows much faster for the 2D element than on a edge.



(a) Relative number of globally coupled degrees of freedom for DG and HDG methods.

(b) Relative number of nonzero entries for DG and HDG methods.

Figure 3.6.: Comparison of globally coupled unknowns and nonzero entries of DG and HDG methods. The mesh is assumed to have a ratio of number of elements to edges of $\frac{K}{\overline{K}} = \frac{2}{3}$. The data is presented for $d = 2$ space dimensions.

In Fig. 3.6 we show how the number of globally coupled unknowns and the number of nonzero entries in the resulting system of equations compare. It is assumed that the ratio between elements and edges is $\frac{K}{\overline{K}} = \frac{2}{3}$. Furthermore, possible auxiliary unknowns are neglected for the DG method. In the images we plot the ratio of globally coupled unknowns and number of nonzero entries of the HDG method compared to the DG method for different polynomial degrees $P \in \{0, 1, \dots, 6\}$. This means that whenever the ratio is below one, the HDG method is advantageous compared to the LDG method.

Fig. 3.6a shows the number of globally coupled unknowns. It is clear from the figure that the HDG method has the same number of globally coupled unknowns already for $P = 1$. The ratio of globally coupled unknowns of the HDG method compared to the DG method becomes increasingly advantageous for the HDG method as P increases.

Fig. 3.6b shows the number of nonzero entries in the resulting matrix. We made the simplifying assumption that all elements and edges are coupled to the same number of elements and edges. In practice, this does not hold for boundary elements and edges. The number of local degrees of freedom is even more important for the number of nonzeros as one has quadratic blocks of size $N(d, P)^2$ and therefore a quadratic dependency. This is also observable in the figure as the HDG method has less nonzero entries already for $P = 1$. It is assumed that an element or edge only couples to its nearest neighbors. Each triangular element creates four blocks of size $\left(\frac{(P+1)(P+2)}{2}\right)^2$ as it couples with itself and its three neighbors, see Fig. 3.7a. For edges of triangles five blocks of size $(P + 1)^2$ are contributed by each edge as a edge couples to itself and the four edges of the two elements sharing the edge,

(a) For a standard DG method the unknowns of the center triangle (red dots) are globally coupled to the degrees of freedoms of its the three neighboring elements (blue vertical lines).

(b) For an HDG method the unknowns of the center edge shared (red solid) by two elements are coupled to the unknowns of the edges of the adjacent triangles (blue dashes).

Figure 3.7.: Schematic of unknowns coupled between elements for the DG (left) and the HDG (right) method. In both methods the unknowns of the considered triangle or edge (red) is coupled to itself and its related neighbor counterparts (blue).

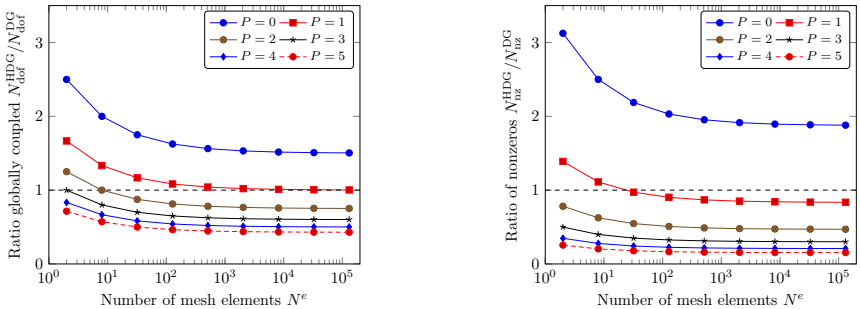
see Fig. 3.7b.

The results only hold for meshes that are infinitely often refined such that $\frac{K}{\overline{K}} = \frac{2}{3}$. Considering the square domain and the uniform triangular mesh in Fig. 3.2, we can compute the globally coupled degrees of freedom and nonzero entries for the DG and HDG method in a more realistic setting. This has been done in Fig. 3.8.

The coarsest mesh consists of $K = 2$ elements and is refined eight times leading to $K = 131072$ elements in the finest mesh. Fig. 3.8a shows the ratio of globally coupled unknowns of the HDG method compared to the number of globally coupled unknowns of a standard DG method. It is clear from the figure that the theoretical values are approached swiftly. If polynomials of degree $P > 1$ are used, the HDG method has less globally coupled unknowns than a standard DG method after 2 refinements which translates to a coarse mesh with only $K = 32$ elements. This mesh is the one presented in Fig. 3.2b. The number of nonzero entries in the matrix is already lower on the coarsest mesh for $P > 1$. Therefore, the HDG method offers benefits for globally coupled systems already on rather coarse meshes and moderate polynomial degrees.

For a mesh with $K = 128$ elements and $P = 3$ we plot the nonzero structure of the matrix G that represents the globally coupled system, see Fig. 3.9a. The matrix stems from an upwind DG and our HDG implementation as described in [124]. One clearly sees that the matrix $\mathrm{G}^{\mathrm{HDG}}$ of the HDG scheme is much smaller and has less than 50% of the number of nonzero entries of the DG method, see Fig. 3.9b.

It is obvious that the construction of the globally coupled system of equations for the HDG method requires more work than for standard DG schemes. The static condensation process requires the assembly of small, local systems that have to be solved. Due to the locality of the problem the assembly and solving of the system of linear system of equations can be done in parallel by default. A problem that can be run in parallel without further intervention,



(a) Development of globally coupled degrees of freedom under uniform refinement.

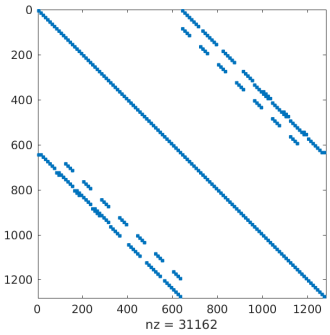
(b) Development of nonzero entries under uniform refinement.

Figure 3.8.: Comparison of globally coupled unknowns and nonzero entries of DG and HDG methods. A regular mesh of a square domain is generated and uniformly refined, see Fig. 3.2b for an example. Data for meshes with $K = 2, 8, 32, \ldots, 131072$ is displayed.

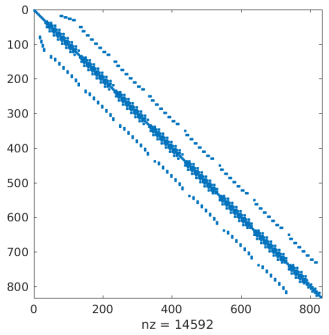
such as the assembly and solving of the local problems, is often called an *embarassangily parallel* problem [108]. When the assembly and solving of the linear system of equations is implemented the assembly can be parallelized without additional effort. Therefore, the assembly is well-suited for current (super-)computer architectures that heavily rely on parallel execution of code.

The potential savings in runtime and memory requirements due to the static condensation process do not only hold in theory but also in practice. In Fig. 3.10, we present the evolution of globally coupled unknowns and nonzero entries under the same assumptions as before. However, we consider a mesh as described for test case C2.1 of the first and second "International Workshop on High-Order CFD Methods" [234]. The coarsest mesh as shown in Fig. 3.11 is much closer to the meshes considered in engineering applications. The mesh contains two NACA0012 airfoils at different angle of attack in a circular domain that has a radius of 100 times the chord length of an airfoil. Again, the ratios of globally coupled unknowns and nonzeros of the HDG method compared to the DG method reach values close to the one for infinite refinements. However, this time the coarsest mesh is already benefiting the HDG method.

The second example, see Tab. 3.1, compares the errors (in L^2-norm) and the runtimes of a standard upwind DG method and an HDG method as presented in Sec. 3.3. The HDG method provides significant savings in runtime over the DG method especially for large P while obtaining similar errors. The results stem also from our MATLAB implementation described in [124]. In this paper there is also a more detailed analysis of the runtime.



(a) The nonzero structure of an upwind DG discretization. The matrix $G^{DG} \in \mathbb{R}^{1280\times1280}$ has 31162 nonzero entries.



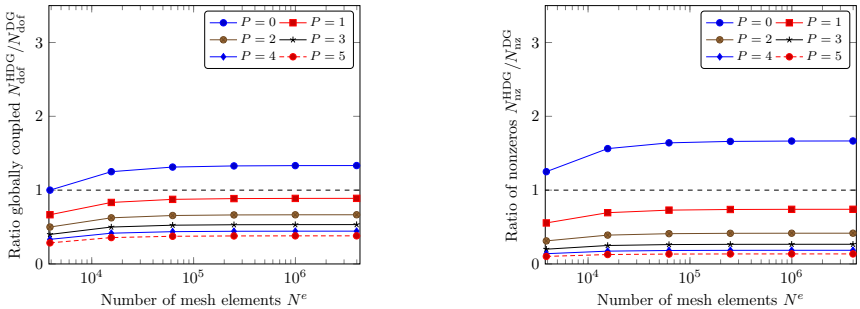
(b) The nonzero structure of an HDG discretization as described in this work. The matrix $G^{HDG} \in \mathbb{R}^{832\times832}$ has 14592 nonzero entries.

Figure 3.9.: The nonzero pattern of the global matrix of an upwind DG scheme (left) and an HDG scheme (right) as used in this work. A linear advection equation is solved using the MATLAB code described in [124]. A square mesh with $K = 128$ elements and polynomials of degree $P = 3$ are used.

Moreover, the implementation may be further optimized for the HDG method since most of the time is not spend on solving the equations, but rather on evaluating certain integrals of (3.15b). Additionally, the implementation does not exploit parallelization explicitly. However, MATLAB usually solves linear systems of equations in parallel in some way. Further comparisons of the HDG method to other numerical discretizations like DG [27, 241] and continuous Galerkin (CG) methods [112, 132, 246] can be found in the literature.

3.4. Implementation

The results presented in this work stem from various implementations of the HDG method. If nothing else is explicitly stated, the implementation is written in C++ and Netgen and NGSolve [204, 205] are used for mesh generation, mesh handling, polynomials and integration formulae. The local problems are solved using LAPACK [6] and the global system is solved using PETSc [12–14]. The code has been parallelized using MPI [158] such that it supports distributed memory system inspired by the thesis of Niklas Fischer [76]. For parallel computations the mesh is partitioned using METIS [131]. We do not show any results concerned with the parallel performance of the code used in this doctoral thesis as the parallel performance of the used HDG implementation was not part of the conducted research. Results for a parallel HDG implementation for shared memory systems can be



(a) Development of globally coupled degrees of freedom under uniform refinement.

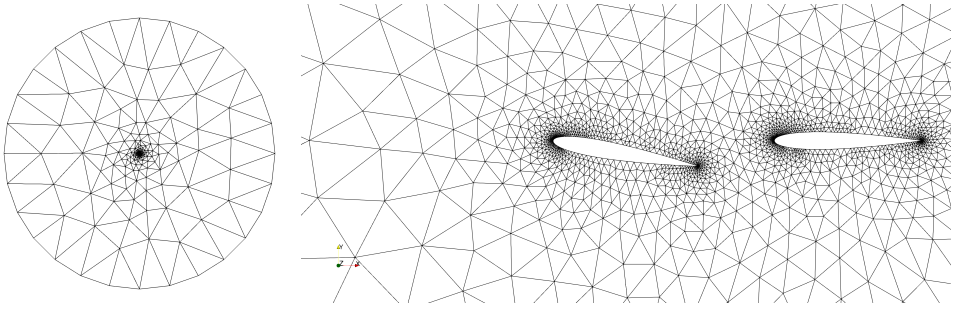
(b) Development of nonzero entries under uniform refinement.

Figure 3.10.: The ratio of globally coupled unknowns and nonzero entries in the global matrix. The graphs compare the HDG method to a standard DG method on a uniformly refined of a circular mesh containing two airfoils as described in [234] (NACA airfoil tandem). Data for meshes with $K = 3893, \ldots, 3986432$ elements is displayed.

found in the works of Roca et al. [199] and for distributed memory systems in the work of Samii et al. [202], for example. A major problem that makes it hard to have an efficient parallel implementation is the globally coupled system. The assembly and solving of the globally coupled system require the parallel processes to communicate. This communication is usually rather slow compared to other operations and may lead to other processes having to wait for completion of the communication. This is not unique to the HDG method, but all numerical methods that construct and solve a globally coupled linear system of equations.

In Fig. 3.12, a brief overview over the main building blocks of the code are presented. The object oriented features and templates available in C++ allow for a good abstraction. The choice of the programming language allows to easily switch between different time integrators, governing equations or file formats for output files. Other big building blocks are the wrapper that obtains the finite elements from NGSolve and the Newton solver that calls suitable linear solvers. A more elaborate explanation of relevant parts is given in the following sections.

Furthermore, an open source implementation in MATLAB for 2D linear advection equations is available [124]. We have contributed the implementation to the FESTUNG project [79, 80, 124, 195] during the preparation of this thesis. The project offers a high-performance MATLAB/GNU Octave framework for discontinuous Galerkin methods. A further implementation for developing and testing in 1D based on MATLAB, that follows the ideas of FESTUNG closely, has been used.



(a) Full mesh.

(b) Zoomed in mesh centered at the origin.

Figure 3.11.: A mesh containing a tandem of NACA airfoils. The generated mesh has $K = 3893$ elements and is centered around the center point of the first airfoil. On the left hand side the whole domain is shown and on the right hand side a close-up of the airfoils is presented.

	Error $\|\cdot\|_{L^2}$		runtime $[s]$	
P	DG	HDG	DG	HDG
0	1.87e-01	2.35e-01	27.1	39.8
1	7.25e-02	7.78e-02	157	218
2	5.53e-02	5.44e-02	797	717
3	4.02e-02	4.13e-02	3980	3166
4	4.16e-02	4.18e-02	10996	7199

Table 3.1.: Comparison of L^2-errors and runtimes for DG and HDG solvers as presented in [124]. A linear convection equation has been solved.

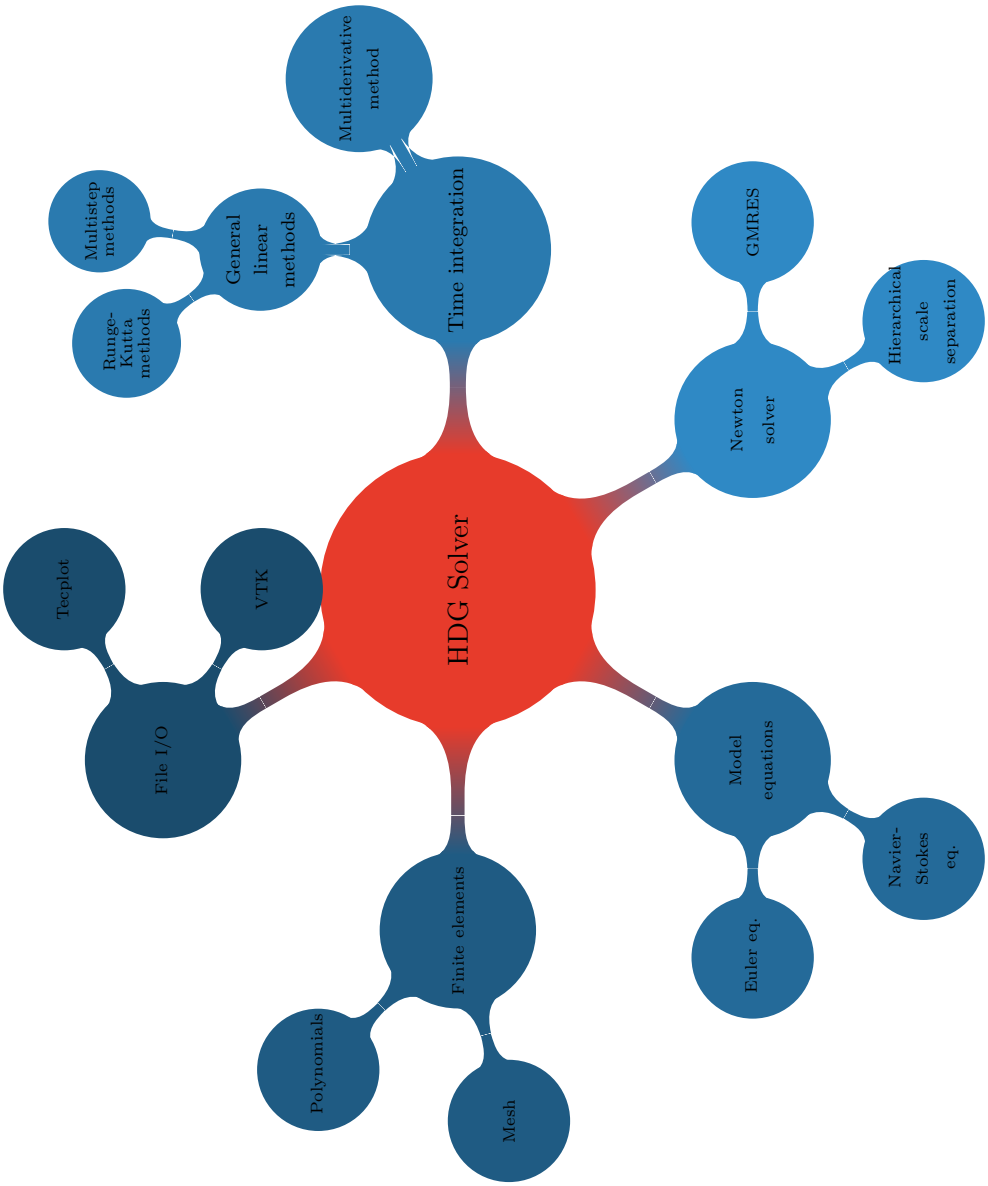


Figure 3.12.: Simplified structure of the implementation. The HDG solver consists of five major parts that are related to the finite element discretization in space, different time discretizations, different model equations that can be solved, an implementation of Newton's method with related linear solvers and file in- and output.

4. Review of time discretizations for the HDG method

In Sec. 3.3, the PDE (2.3) was discretized in space and one obtained a semi-discrete problem. In order to numerically solve time-dependent problems we need to discretize the problem also in time in order to receive the fully discrete system. It has also been mentioned in Sec. 3.3 that with

$$\partial_t \mathcal{T}(\mathbf{w}_h, \mathbf{x}_h) = \mathcal{N}(\mathbf{w}_h, \mathbf{x}_h) \qquad \text{(3.19 revisited)}$$

the semi-discrete system is a differential algebraic equation. This is common among many numerical space discretizations and is not unique to (hybridized) discontinuous Galerkin methods. A common approach to discretize the semi-discrete system is to use well-known time-stepping methods for ODEs. Popular choices are multistep schemes and multistage methods. For a broad overview of different methods we refer to the extensive text books of Hairer, Wanner and Nørsett [99], its second part by Hairer and Wanner [101] or Butcher [28].

In the previous chapter, it has also been mentioned that the semi-discrete system of the HDG method is a differential algebraic equation of order one [119, 165] due to the lack of time derivatives of the auxiliary unknown σ_h and hybrid unknown λ_h. The index of the DAE refers to how often the equations have to be differentiated with respect to time such that one can recover an ordinary ODE [101]. This is a tedious task and might not always be possible due the requirements of the differentiability of the solutions.

The time integrators should be at least A-stable [99, 101, 223] as the studied problems are stiff. The reason we prefer methods with good stability properties is that these methods can be applied directly to DAEs and keep the order of convergence for all quantities of the DAEs. Under certain conditions no order reduction is observed [101, 223]. This means that the time integrator keeps its order of convergence in time t when applied to a DAE. This is not automatically the case for other time integrators [101].

It was mentioned in the previous chapter that implicit methods create one or more globally coupled systems per time step that have to be solved. Therefore, the HDG method is in particular interesting due to the application of static condensation, see Sec. 3.3.2. However, the construction and solving of the linear system is still the most expensive part of the method. Therefore, we investigate several different classes of time integrators in order to identify stable and efficient time integrators for the HDG method.

This chapter focuses on introducing time integrators already established for the HDG method [119, 165, 168, 169, 208]. In this thesis, these integrators are used for either new

applications or to compare them with new time integrators. The chapter is structured as follows: First, we introduce additional notation for the time discretization of the semi-discrete problem (3.20). Subsequently, we discuss the stability requirements of our space discretization. Then, we present multistep methods, such as backwards differentiation formulae (BDF), and multistage methods, such as Runge-Kutta (RK) schemes. Each description of the numerical integrators is concluded by numerical experiments.

4.1. Notation and definitions

The time integrators used in this thesis are usually introduced in the setting of an ordinary differential equation

$$\begin{aligned} w'(t) &= g(t, w(t)), \quad t \in (0, t_{\text{final}}], \\ w(0) &= w_0, \end{aligned} \tag{4.1}$$

with unknown w, a given right hand side g and given initial data w_0. We do so, as well, to keep a consistent notation with the literature, but we also state the system of equations for the fully discretized problem for a discretization using the HDG method. This can be done quite easily, because, as mentioned in Sec. 3.3, the semi-discrete system (3.19)

$$\partial_t \mathcal{T}(\mathbf{w}_h, \mathbf{x}_h) = \mathcal{N}(\mathbf{w}_h, \mathbf{x}_h) \tag{3.19 revisited}$$

is a DAE. The notation has been chosen to 'hide' the differential algebraic equation such that the shape of the problem is very close to the problem of solving an ODE as shown in (4.1). Using this notation it is possible to apply the time integrators for ODEs to the semi-discrete PDE with very similar notation.

For the time discretization, the time slab is divided into a set of $N^t + 1$ discrete times

$$0 = t^0 < t^1 < \ldots < t^{N^t} = t_{\text{final}}$$

where the time difference between consecutive times

$$\Delta t^n \coloneqq t^{n+1} - t^n$$

is referred to as time step or time step size. In general, these may vary during the simulation, i.e. $\Delta t^n \neq \Delta t^{n+1}$. If equidistant time steps are used, i.e. $\forall n,\ \Delta t^n = \Delta t^{n+1}$, then we may drop the time index n and refer to the time step simply as $\Delta t = \frac{t_{\text{final}}}{N^t}$. In this case, the time is given as $t^n = n \cdot \Delta t$. Furthermore, we use the shorthand notations $w^n = w(t^n)$ and $g^n = g(t^n, w(t^n))$ when the exact solution and exact right hand side are evaluated at time t^n. In the discrete setting we use $w_h^n \approx w(t^n)$ and $g_h^n = g(t^n, w_h^n)$ in order to refer to the solution obtained from the time integrator. The approximated solution and right hand side evaluated using the approximated solution are indicated by $\cdot_h$. We use the same short hand notation for unknowns stemming from the discretization of the PDE.

The theory of differential equations is not the concern of the thesis. Therefore, we limit the introduction to a short explanation of the required stability properties and the description of the time integration methods to a minimum. Further information on these methods can be found in various text books [28, 99, 101, 223].

If multiderivative time integrators are used, we need more than just the first time derivative. In order to allow for a compact notation we define

$$\partial_t^i \coloneqq \frac{\partial^i}{\partial t^i}, \quad \partial_x^i \coloneqq \frac{\partial^i}{\partial x^i}$$

as shorthand notation for time and space derivatives in one space dimension.

4.1.1. Stability requirements

In order to underline the need for implicit solvers that are at least A-stable, we analyze the spectrum of the HDG-operator for the linear convection equation in 1D

$$\partial_t w + u\partial_x w = 0, \quad \text{on } x \in [0,1] \tag{4.2}$$

with periodic boundary conditions. After discretizing the system in space using the HDG method, one obtains the following system of equations

$$\begin{pmatrix} \mathrm{M}_\varphi \partial_t W \\ 0 \end{pmatrix} + \begin{pmatrix} \mathrm{A}_{ww} & \mathrm{A}_{w\lambda} \\ \mathrm{A}_{\lambda w} & \mathrm{A}_{\lambda\lambda} \end{pmatrix} \begin{pmatrix} W \\ \Lambda \end{pmatrix} = 0$$

where the naming of the matrices follows Sec. 3.3.2. The matrix M_φ denotes a mass matrix. This has to be reformulated to get the semi-discrete system that only depends on W. The hybrid unknown Λ can be eliminated from the system in a similar fashion to the static condensation procedure, see Sec. 3.3.2. The equation can be rewritten as

$$\mathrm{M}_\varphi \partial_t W + \mathrm{A}_{ww} W + \mathrm{A}_{w\lambda}\Lambda = 0 \tag{4.3a}$$

$$\mathrm{A}_{\lambda w} W + \mathrm{A}_{\lambda\lambda}\Lambda = 0 \tag{4.3b}$$

where the first line refers to the equations stemming from the initial PDE and the second line refers to the hybrid equation introduced for the HDG method, see Sec. 3.3. The set of hybrid equations can be used

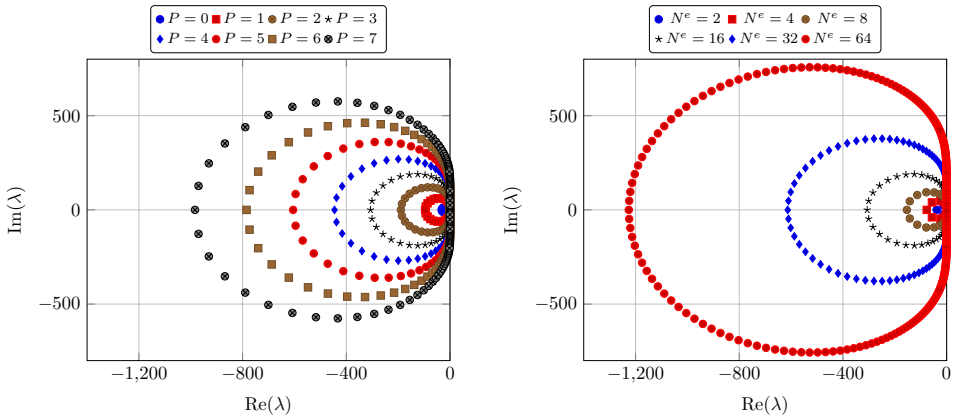
$$\mathrm{A}_{\lambda w} W + \mathrm{A}_{\lambda\lambda}\Lambda = 0 \Leftrightarrow \Lambda = -\mathrm{A}_{\lambda\lambda}^{-1}\mathrm{A}_{\lambda w} W$$

to express the hybrid unknown Λ in dependence of W. After plugging this into (4.3a) we obtain

$$\mathrm{M}_\varphi \partial_t W + \left(\mathrm{A}_{ww} - \mathrm{A}_{w\lambda}\mathrm{A}_{\lambda\lambda}^{-1}\mathrm{A}_{\lambda w}\right) W = 0.$$

Inverting the mass matrix leads to

$$\partial_t W = \tilde{\mathrm{A}} W \tag{4.4}$$



(a) Eigenvalues of the HDG method for increasing polynomial degree on a mesh with $K = 16$ elements.

(b) Eigenvalues of the HDG method under uniform refinement of the mesh with polynomials degree $P = 3$.

Figure 4.1.: The development of the spectrum of HDG operator for the linear convection equation. The linear convection equation has been discretized in 1D on $\Omega = [0, 1]$ and with periodic boundary conditions.

with $\tilde{\mathrm{A}} \coloneqq -\mathrm{M}_{\varphi}^{-1} \left(\mathrm{A}_{ww} - \mathrm{A}_{w\lambda}\mathrm{A}_{\lambda\lambda}^{-1}\mathrm{A}_{\lambda w}\right)$.

We compute the eigenvalues λ of $\tilde{\mathrm{A}}$ for $u = 1$. Thus, the stabilization coefficient α of the numerical flux functions, see (3.17a), is set to 1. The discretization uses Legendre polynomials as basis functions (without normalization) and the eigenvalues are computed by MATLAB's `eig` function for eigenvalues of full matrices. The resulting spectra are presented in Fig. 4.1 for increasing polynomial degree P on a fixed mesh with $K = 16$, see Fig. 4.1a, and under uniform mesh refinement for fixed $P = 3$.

All eigenvalues lie in the left half-plane of the complex plane and the spectrum depends on the mesh size Δx and P. In both cases, the spectrum grows rapidly when the spatial accuracy increases. Even for rather coarse meshes and low polynomial degrees the absolute values of the largest eigenvalues are already several orders of magnitude larger than the smallest eigenvalues.

In order to better understand the significance of the eigenvalues of the HDG-operator for the choice of a suitable time integrator, we briefly discuss *stiff* problems. The term stiff is commonly used in the theory of differential equations like ordinary differential equations (4.1) [28, 99, 101, 223]. Stiff problems usually behave badly when solved using explicit time integrators. These integrators require very small time step sizes Δt while convergence against the correct solution is slow and the numerical solution may show strong oscillations. In [101],

the authors call an equation stiff if explicit methods do not work properly and give a large number of examples for such problems.

The reason why a differential equation is stiff can have different causes. In the setting of a classical ODEs stiffness can refer to processes that happen concurrently, but at different speeds. A common example is the kinetics of chemical reactions where some reactions happen much faster than others.

The solution process of PDEs using a spatial discretization usually leads to a system of differential equations. This may result in a system of stiff ODEs depending on the PDE discretized and the numerical method used for the discretization in space. An example for such PDEs are the ones having a diffusion term. This terms often leads to stiff behavior of the system of differential equations [101]. An example for a stiff system caused by the space discretization is in fact the HDG method as the method leads to a system of differential algebraic equations that also introduces stiffness.

The analysis of the eigenvalues of a system of differential equations can help identifying stiff systems. If the system of differential equations is nonlinear, it has to be linearized. Then, the eigenvalues $\lambda_i \in \mathbb{C}$ of the resulting linear system can be computed. The problem is usually referred to as stiff if the real part of the eigenvalue is negative and the absolute values of the (linearized) problem differ greatly

$$\mathrm{Re}(\lambda) < 0, \ \max_{i,j} \frac{|\lambda_i|}{|\lambda_j|} \gg 1.$$

When plotting the spectrum of the HDG-operator, see Fig. 4.1, i.e. the eigenvalues of the system of differential equations (4.4), one sees that the absolute values of the eigenvalues differ greatly. Thus, the HDG discretization leads to a stiff system of differential equations as already claimed. This may seem like a drawback, but the HDG method is especially beneficial for problems when one has to use an implicit time integrator anyway. Then, one can take advantage of the static condensation process, see Sec. 3.3.2, that reduces the size of the globally coupled system one has to solve.

In order to tackle stiff problems time integration methods with special stability properties have been derived. A famous problem to test the stability of a scheme is Dahlquist's problem

$$w'(t) = \lambda w, \ w_0 = 1 \tag{4.5}$$

with $\lambda \in \mathbb{C}$ and solution $w(t) = e^{\lambda t}$. After applying a time discretization to Dahlquist's problem one obtains a procedure to update the solution in each time step for which the behavior depends on the scaled complex number $z \coloneqq \lambda \Delta t$. The stability region S contains all complex values z for which the time discretization is stable. More details on the update procedure for different kind of methods, such as multistep or multistage methods, for example, can be found in the literature [99, 101, 223].

As we have seen, the largest eigenvalues tend to become larger in magnitude when the discretization in space becomes more accurate, so one obvious choice for a preferable stability region is

$$S \supset \{z \in \mathbb{C} \mid \mathrm{Re}(z) \leq 0\},$$

where the complete left half plane of the complex numbers is included. In literature this is called *A-stability*. In practice one may not need a method that is stable for the complete half plane. As shown for the HDG method, see Fig. 4.1, one may not need stability for values close to the imaginary axis with large complex part. Still, A-stability would ensure stability in this region and therefore exclude numerical schemes stable for a large range of the left half plane, but not the whole. Thus, a (possibly) weaker sort of stability given as

$$S \supset \{z \in \mathbb{C} \mid |\arg(z) - \pi| < \alpha\},$$

exists. This type of stability is called *A(α)-stability* with α referring to the angle between the real axis and the upper/lower bound of the stability region. Thus, the inner angle of the axis-symmetric domain is given by 2α. For $\alpha = \frac{\pi}{2}$ the stability region is again the left half plane and thus its equivalent to A-stability. Depending on the angle α the eigenvalues may be outside of the stability region. In Fig. 4.1, we can see that some eigenvalues with large imaginary but small real parts that are clustered close to the imaginary axis and thus are require a large angle α to be within the stability region of the time integrator.

Although A- and A(α)-stability are a good measure whether a method is suitable for stiff problems we mention that "A-stability is not the whole answer to the problem of stiff equations" (R. Alexander [4]). Even A-stable methods would not necessarily lead to stable discretizations and the order of convergence of the method might deviate. This has led to the introduction of stricter stability conditions in order to ensure the stability of the method in such cases. For one-step methods one observes for problematic schemes that they do not damp oscillations in the solution sufficiently for $\mathrm{Re}(z) \to -\infty$. A one-step method that guarantees to damp oscillations even for very small z is called L-stable [101]. Therefore, L-stable methods can be beneficial for stiff problems as the ones we aim to solve in this thesis.

4.2. Multistep methods

In order to compute an approximation of the solution at a new time t^{n+1}, several solutions $w_h^{n+1-i},\ i = 0, 1, \ldots, r-1$ and evaluations of the right hand side $g_h^{n+1-i},\ i = 0, 1, \ldots, r$ of the r previous time steps are used. Then, the formula to update w_h^{n+1} given as

$$\sum_{i=0}^{r} a_i w_h^{n+1-i} = \Delta t^n \sum_{i=0}^{r} b_i g_h^{n+1-i} \tag{4.6}$$

	BDF1	BDF2	BDF3	BDF4
α	90.00°	90.00°	86.03°	73.35°

Source: Hairer and Wanner [101]

Table 4.1.: Angle α of $A(\alpha)$-stable BDF-methods up to order 4.

has to be solved. A method with such structure is called a multistep method. The coefficients $a_i, b_i \in \mathbb{R}$ and the number of steps $r \in \mathbb{N}$ have to be chosen such that the method is of high order in time and has good stability properties. In our case we use an implicit method, i.e. $a_0 \neq 0,\ b_0 \neq 0$, as only such a method can be at least A-stable.

For the discretization of the semi-discrete system we choose backwards differentiation formulae (BDF) that have been first introduced by Curtiss and Hirschfelder [60]. The method gained major recognition after the book of Gear [85]. Such a BDF method uses r known approximations $w_h^{n-i}, i = 0, \dots, r-1$, the unknown approximation w_h^{n+1} and evaluates the right hand side of the ODE only at t^{n+1}. Thus, the method is given by

$$\sum_{i=0}^{r} a_i w_h^{n+1-i} = \Delta t^n g_h^{n+1-i} \tag{4.7}$$

with coefficients $a_i \in \mathbb{R}$. The coefficients of the BDF scheme up to fourth order are given in the appendix, see Tab. A.1.

The BDF scheme up to $r \leq 6$ is zero-stable and of Qth order convergent in time, i.e. $\max_n \|w_h^{t^n} - w(t^n)\| \leq C\Delta t^Q$ with some positive constant C. For BDF methods it holds that the order in time is equal to the number of known approximation used, i.e. $Q = r$ holds. This property holds also when applied to differential algebraic equations of index 1 as created by the HDG method. In both cases, ODE and DAE of index one, the approximations of $w_h^{n-i},\ i = 0, \dots, r-1$ must fulfill $\|w_h^{n-i} - w(t^{n-i})\| = \mathcal{O}(\Delta t^Q),\ i = 1, \dots, r$. Consecutive time steps computed by the BDF method fulfill this automatically. However, special attention has to be paid to the first r approximations $w_h^1, \dots, w_h^r$ that have to be computed numerically when the scheme is started from the initial conditions. Note that $w_h^0 = w(0)$ is given by the initial condition and is therefore considered exact. Although BDF methods can be derived for variable time step sizes Δt^n [99] we restrict ourselves to constant time step sizes Δt. The schemes are also often named BDFr to indicate the order of convergence in time and the number of time steps used in the scheme.

The BDF schemes have the advantage that they lead to only one system of equations that has to be solved per time step. During the construction of the system only known data from previous time steps is needed, coming with the cost of some memory and a slightly increased assembly time for increasing r. However, the method is only A(α)-stable for $Q > 2$ with

α decreasing rapidly for increasing r, see Tab. 4.1. In fact, the famous 'second Dahlquist barrier' states that there cannot be a linear multistep method of order $Q > 2$ that is A-stable [101]. Therefore, we limit ourselves to the use of the BDF method up to $r = 3$. In practice, other implementations avoid these problem by using an automatic time step size and order adaptation [85].

BDF applied to the HDG method Following the definition of the BDF method for ODEs (4.7) applied to the semi-discrete HDG formulation (3.19) leads to the discrete formulation

$$\sum_{i=0}^{r} a_i \mathcal{T}(\mathbf{w}_h^{n+1-i}, \mathbf{x}_h) = \Delta t^n \mathcal{N}(\mathbf{w}_h^{n+1}, \mathbf{x}_h), \quad \forall \mathbf{x}_h \in \mathbb{X}_h.$$

which has to be solved for each time step $n = 1, \ldots, N^t$.

For methods of order $Q > 1$ a startup procedure is needed. The BDF2 method is started using the BDF1 method, which in fact is the implicit Euler method, because it is second order accurate for the first time step. The BDF3 method is started using the BDF2 method with an time step size $\Delta\tilde{t}$ that ensures the accuracy of the solution. Depending on the problem at hand this may lead to a large number of BDF2 steps that have to be carried out during the startup phase.

The BDF methods have been applied to the HDG method already in [119, 167, 168, 208]. We use the BDF methods for verification of test cases in which the time integrator is not the main concern.

4.2.1. Numerical results

It has been mentioned in Sec. 3.3 that we can expect the space discretization to be of order $P+1$ accurate. We solve time-dependent problems with an implicit time integrator. Therefore, the order of accuracy of the used time integrator is important for the overall order of accuracy of the scheme. One could also use lower order time integrators and reduce the time step size Δt accordingly to not tamper with the accuracy given by the space discretization. However, this would also lead to a CFL-like condition that we want to avoid. Thus, in combination with the time integrator the total error depends on the spatial and temporal discretization, Hence, we expect $\max(Q, P + 1)$ as total order of accuracy for smooth solutions.

In the following we present numerical results for two different test cases. The first one solves a linear advection-diffusion equation in order to verify the mixed approach to represent the second space derivatives. The second test case solves the Euler equations in order to verify that the implementation handles nonlinear systems of equations correctly.

Linear advection-diffusion equation

In this test case, the linear advection-diffusion equation with

$$f_c(w) = wu^{\mathrm{T}}, \quad f_v(w, \nabla w) = \varepsilon \Delta w$$

is solved on the domain $\Omega = [-0.5, 0.5]^2$ up to $t_{\text{final}} = \frac{\pi}{4}$ and constant diffusivity $\varepsilon = 10^{-3}$. The velocity vector is still constant in time, but depends on the space coordinate $u(x) = (-4x_2, 4x_1)^{\mathrm{T}}$. The initial condition is given by a Gaussian

$$w(0, x) = \exp\left(-\frac{(x_1 - x_{1,c})^2 + (x_1 - x_{2,c})^2}{2s^2}\right)$$

that travels in circular, counter-clockwise direction while being damped due to the diffusive process. The exact solution is given by

$$\begin{aligned} w(t, x) &= \frac{2s^2}{2s^2 + 4\varepsilon t} \exp\left(-\frac{(x_1' - x_{1,c})^2 + (x_2' - x_{2,c})^2}{2s^2 + 4\varepsilon t}\right) \\ x_1' &= x_1 \cos(4t) + x_2 \sin(4t) \\ x_2' &= -x_1 \sin(4t) + x_2 \cos(4t) \end{aligned}$$

with standard deviation s set to 0.1 and the point around which the initial Gaussian is centered is $x_c = (-0.2, 0)^{\mathrm{T}}$. We prescribe the exact solution as boundary condition on all domain boundaries. This is the same test case that has been used for verifying the HDG method in previous publications [119, 167, 208].

We set the stability parameter to $\alpha = 2$ and solve the system of equations using a restarted GMRES until the residual drops below 10^{-12}. The initial time step size on the coarsest mesh with $K = 2$ is $\Delta t = \frac{\pi}{32}$. The mesh is uniformly refined, thus with each refinement the number of elements increases by a factor of 4 and the time step is halved.

In Fig. 4.2, we show numerical results obtained with the BDF methods. All BDF methods reach the expected order of convergence. The solution already improves after few refinements. Additionally, increasing the polynomial degree P to Q reduces the total error, but the order of accuracy (slope) stays the same. This indicates that the global error on the coarser meshes is dominated by the spatial error.

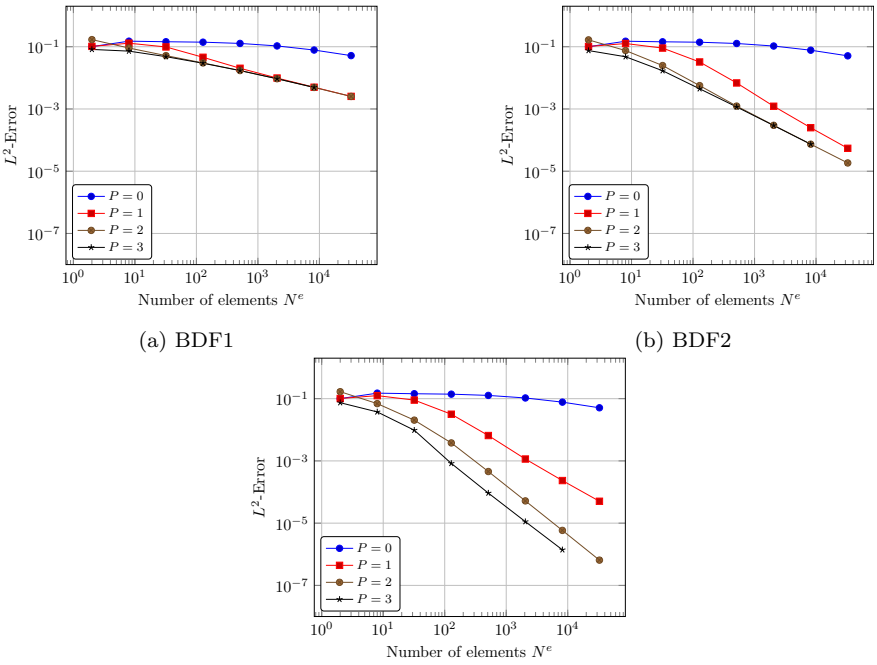
Euler equations

The Euler equations (2.9), as described in Sec. 2.6, are solved on $\Omega = [0, 2]^2$. The initial conditions are given by

$$\rho = 1 + 0.2 \sin(\pi \cdot (x_1 + x_2)),\ u_1 = 0.7,\ u_2 = 0.3,\ p = 1.0$$

and periodic boundary conditions are employed. The exact solution is given by

$$\rho(t, x) = 1.0 + 0.2 \sin(\pi \cdot ((x_1 + x_2) - t(u_1 + u_2)))$$

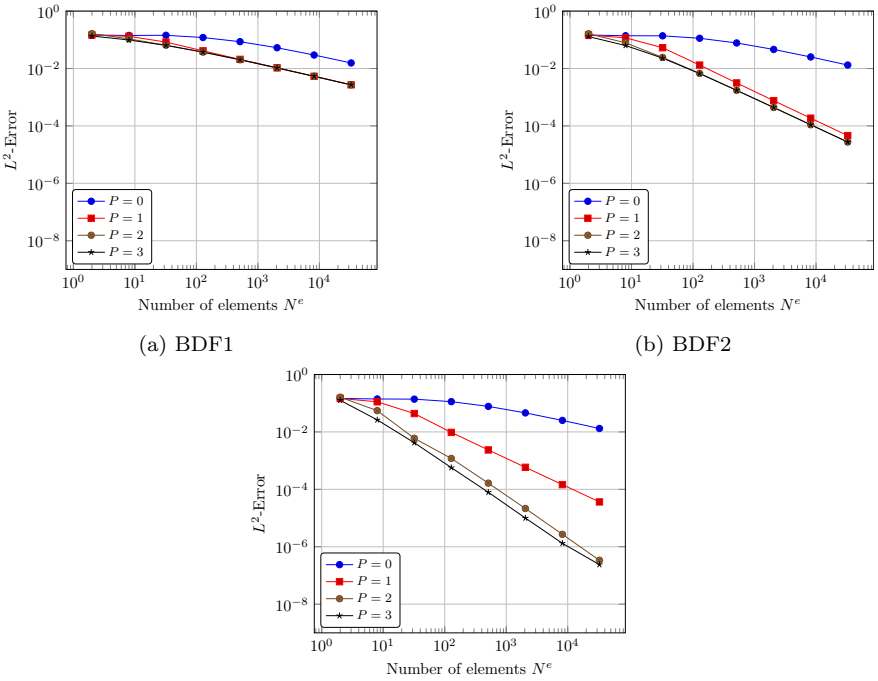


(a) BDF1

(b) BDF2

(c) BDF3

Figure 4.2.: Errors of the rotating Gaussian (linear convection-diffusion) test case solved with HDG and BDFr methods. The problem is solved on the domain $\Omega = [-0.5, 0.5]^2$ up to $t_{\text{final}} = \frac{\pi}{4}$ and constant diffusivity $\varepsilon = 10^{-3}$. The velocity vector is given as $u(x) = (-4x_2, 4x_1)^{\mathrm{T}}$ and the CFL-number is CFL $= \frac{\sqrt{2}\pi}{16} \approx 0.278$. After sufficient refinements the methods reach the expected convergence rate $\max(P + 1, Q)$.



(a) BDF1

(b) BDF2

(c) BDF3

Figure 4.3.: Errors of the nonlinear test case (Euler equations) solved with HDG and BDFr methods. The problem is solved on $\Omega = [0, 2]^2$ up to $t_{\text{final}} = 1$. The CFL-number is CFL ≈ 1.042. After sufficient refinements the methods reach the expected convergence rate $\max(P + 1, Q)$.

with the velocities and pressure being constant [128]. We run the simulation up to a final time $t_{\text{final}} = 1$.

The test case is rather simple as many quantities stay constant. However, it is one of the few cases where the solution stays smooth and where one has an analytical expression of the solution without adding an artificial source term to the equations. Moreover, it is testing the ability of the numerical method and its implementation to solve a system of equations correctly when the transport velocity is anisotropic.

The settings of the solver are very similar to the previous test cases. We set the stability parameter to $\alpha_c = 1$ and solve the system of equations using a restarted GMRES until the residual drops below 10^{-12}. The Newton solver iterates until the absolute or relative residual in L^2-norm drops below 10^{-12}. The initial time step size on the coarsest mesh with $K = 2$ is

$\Delta t = 0.5$. The mesh is uniformly refined, thus with each refinement the number of elements increases by a factor of 4 and the time step is halved.

In Fig. 4.3, we present the results of the BDF methods applied to this problem. The method reaches the expected order of convernce after few refinements. Similar to the linear advection-diffusion equation increasing the polynomial degree P to Q reduces the error slightly without changing the slope of the error plot.

4.3. Multistage methods

Another common class of time integrators are multistage methods. These fall in the class of one-step methods. Thus, no history of previous time steps is needed, but only the known solution at t^n. High order accuracy in time and stability are achieved by computing s intermediate solutions $w_h^{n,i}$ by solving

$$w_h^{n,i} = w_h^n + \Delta t^n \sum_{j=1}^{i} a_{ij} g_h^{n,i}, \quad i = 1, \dots, s. \tag{4.8a}$$

These intermediate steps are also called *stages*. The solutions of the stages are then combined

$$w_h^{n+1} = w_h^n + \Delta t^n \sum_{i=1}^{s} b_i g_h^{n,i} \tag{4.8b}$$

to update the solution. The coefficients a_{ij}, b_i and c_i are chosen in such a way that the approximation is of high order in time and the method has good stability properties. The time $t^{n,i}$ is given by the combination of the coefficient c_i and the time step size Δt as $t^{n,i} := t^n + c_i \Delta t^n$.

In order to construct methods that are at least A-stable, one has to choose an implicit method. Unlike classical multistep methods one can construct methods that are A- and L-stable also for orders higher than two. However, this comes at the cost of additional stages that lead to more computational work.

A common way to represent the coefficients of the method is the use of a Butcher tableau, see Tab. 4.2a. If for a stage i at least one coefficient $a_{ij},\ j \geq i$ is nonzero, the stage is implicit as it couples to itself or another unknown stage.

As not all stages need to be implicit this has led to a variety of terms indicating the implicit nature of the scheme. We focus on so called diagonally implicit Runge-Kutta (DIRK) methods [4]. For these methods it holds that $a_{ii} \neq 0$ and $a_{ij} = 0,\ j > i$, see Tab. 4.2b. Thus, a stage is coupled only to itself and previously computed stages and the method leads to s systems of equations that have to be solved consecutively.

In many applications, especially in semi-discretized PDEs, the initial system of equations is already very big. The application of fully implicit Runge-Kutta methods would lead to

c_1	a_{11}	$\dots$	a_{1s}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	$\dots$	a_{ss}
	b_1	$\dots$	b_s

(a) Butcher tableau of a Runge-Kutta method.

c_1	a_{11}		
$\vdots$	$\vdots$	$\ddots$	
c_s	a_{s1}	$\dots$	a_{ss}
	b_1	$\dots$	b_s

(b) Butcher tableau of an (embedded) diagonally implicit Runge-Kutta method. Left out coefficients in the upper right are zero.

Table 4.2.: Butcher tableau of Runge-Kutta methods and diagonally implicit Runge-Kutta methods.

systems of equations costly to assemble or invert. Thus, we do not consider these methods in this thesis. However, we want to mention that fully implicit Runge-Kutta methods have recently attracted interest in the CFD community [117, 118, 177]. One main advantage of these methods is that they can achieve higher order while having fewer stages than DIRK methods.

In many publications, DIRK methods with a constant diagonal term $a_{ii} = C$ are used [4, 41, 191]. Hairer and Wanner are referring to these methods also as singly diagonally implicit Runge-Kutta (SDIRK) method [101]. These methods can also be A- and L-stable with the restriction that such a method with s stages is at most of order $Q = s$ accurate in time.

We mainly use the second and third order method with two and three stages by Alexander [4] that are second and third order accurate in time and the fourth order method with five stages by Hairer and Wanner [101]. A second option of a four stage fourth order method is the scheme of Al-Rabeh [191]. However, the method of Al-Rabeh is not L-stable and thus the method might fail for very stiff problems. The methods are named DIRKsQ to refer to the DIRK method with s stages and order Q, so DIRK54 is Hairer's and Wanner's method with five stages that is fourth order accurate in time. Besides the mentioned methods we use also a first order DIRK scheme DIRK11 that in fact refers to the implicit Euler method. Note that the implicit Euler method also coincides with the BDF1 method. The coefficients of the methods can also found in the appendix A.

Remark 6 *All methods, except the one by Al-Rabeh, are stiffly accurate, i.e.* $a_{sj} = b_j$*. This also implies that the methods are L-stable [101].*

Time step adaptation One step methods like the presented DIRK methods are well-suited for coupling with automatic time step adaptation [41, 101, 191]. They only depend on the solution of the known time level which makes it easy to change the time step size Δt^n during the simulation without the need of any rescaling of previous solutions or similar. Moreover,

c_1	a_{11}		
$\vdots$	$\vdots$	$\ddots$	
c_s	a_{s1}	$\dots$	a_{ss}
	b_1	$\dots$	b_s
	$\hat{b}_1$	$\dots$	$\hat{b}_s$

Table 4.3.: Butcher tableau of an diagonally implicit Runge-Kutta method with an embedded formular for error estimation. Left out coefficients in the upper right are zero.

for a reasonable time step adaptation an estimator $\tilde{\delta}_{n,\Delta t^n}$ of the local error is necessary. The local error at time level $n+1$, which we define as

$$\tilde{\delta}_{n,\Delta t^n} \coloneqq w(t^{n+1}; t^n, w_h^n) - w_h(t^{n+1}; t^n, w_h^n),$$

is the difference between the exact solution w and the approximated solution w_h. Both use the approximated solution w_h^n at the known time level n as initial data for one time step. The time step adaptation should modify the time step such that at the end of the simulation the error in the solution is bounded

$$\|w(t^{N^t}) - w_h(t^{N^t})\| \leq \text{tol} \cdot (t^{N^t} - t^0)$$

where tol is a user defined constant. In this setting it is useful that

$$\|w(t^{N^t}) - w_h(t^{N^t})\| \leq \sum_n \|\tilde{\delta}_{n,\Delta t^n}\|$$

holds. By controlling the local error in every time step we can control the global error in time. For this, an estimator $s_{n,\Delta t^n} \approx \tilde{\delta}_{n,\Delta t^n}$ is needed. In the setting of DIRK methods a simple error estimator can be introduced.

It is possible to integrate a lower or higher order DIRK method within an existing DIRK method [41, 101, 191]. The Butcher tableau, see Tab. 4.2b, can be extended by additional coefficients for that, see Tab. 4.3. The coefficients $\hat{b}_i,\ i = 1, \dots, s$ are used to compute an additional approximation

$$\hat{w}_h^{n+1} = w_h^n + \Delta t^n \sum_{i=1}^{s} b_i g_h^{n,i}$$

to the solution. The estimator for the local error is then given as

$$s_{n,\Delta t^n} \coloneqq \|w_h^{n+1} - \hat{w}_h^{n+1}\|.$$

The time step adaptation depends strongly on the ratio

$$r_{n,\Delta t^n} \coloneqq \frac{s_{n,\Delta t^n}}{\text{tol} \cdot \Delta t^n}$$

as it relates the (estimated) local error of the executed time step to the maximum allowed error. If $r_{n,\Delta t^n} > 1$, the local error of the current time step has to be rejected and the time step must be recomputed with a smaller time step size. If $r_{n,\Delta t^n} \leq 1$, the local error was smaller than the maximum allowable error and the time step is accepted. The ratio helps to determine the time step size Δt^{n+1} since it should not be greater than one, but values close to one are preferred as this minimizes the number of time step sizes. At the same time the adaptation strategy should not be too optimistic in order to minimize the number of rejected time steps.

We aim to solve nonlinear problems and thus we can use additional information from the Newton method, see Sec. 3.3.1, to guide the time step adaptation [101]. If the solution varies only mildly between two time steps the Newton method is likely to converge within few iterations. This is taken into account by an additional safety factor

$$\alpha(k) \coloneqq 0.9 \cdot \frac{2k_{\max} + 1}{2k_{\max} + k}$$

with k being the number of Newton steps and $k_{\max}$ the maximum number of Newton steps allowed. If the solution is accepted, we set the new time step size to

$$\Delta t^{n+1} \coloneqq \alpha(k) r_{n,\Delta t^n}^{-\frac{1}{Q}} \Delta t^n.$$

It is sometimes necessary to enforce a minimum and maximum allowable time step. Therefore, we can set parameters $\Delta t_{\min}$ and $\Delta t_{\max}$ that ensure that $\Delta t_{\min} \leq \Delta t^n \leq \Delta t_{\max}$. More information on the time adaptation process can be found in the books of Hairer and Wanner [101] and Dahmen and Reusken [61] or the books and papers in which DIRK methods are used [41, 101, 191]. The employment of a time step adaptation strategy to an HDG space discretization has been discussed in [119].

DIRK methods applied to the HDG method As for the BDF method the DIRK methods for ODEs (4.8a) and (4.8b) can be applied directly to the semi-discrete problem (3.19). Thus, for every stage one has to solve

$$\mathcal{T}(\mathbf{w}_h^{n,i}, \mathbf{x}_h) = \mathcal{T}(\mathbf{w}_h^n, \mathbf{x}_h) + \Delta t^n \sum_{j=1}^{i} a_{ij} \mathcal{N}(\mathbf{w}_h^{n,i}, \mathbf{x}_h), \quad \forall \mathbf{x}_h \in \mathbb{X}_h,$$

and the solution at the time level t^{n+1} is obtained by

$$\mathcal{T}(\mathbf{w}_h^{n+1}, \mathbf{x}_h) = \mathcal{T}(\mathbf{w}_h^n, \mathbf{x}_h) + \Delta t^n \sum_{i=1}^{r} b_i \mathcal{N}(\mathbf{w}_h^{n,i}, \mathbf{x}_h), \quad \forall \mathbf{x}_h \in \mathbb{X}_h.$$

DIRK methods have been applied to the HDG method already in [119, 123, 165, 169]. Again, we use the schemes for verification or test cases in which the time integrator is not the main concern.

4.3.1. Numerical results

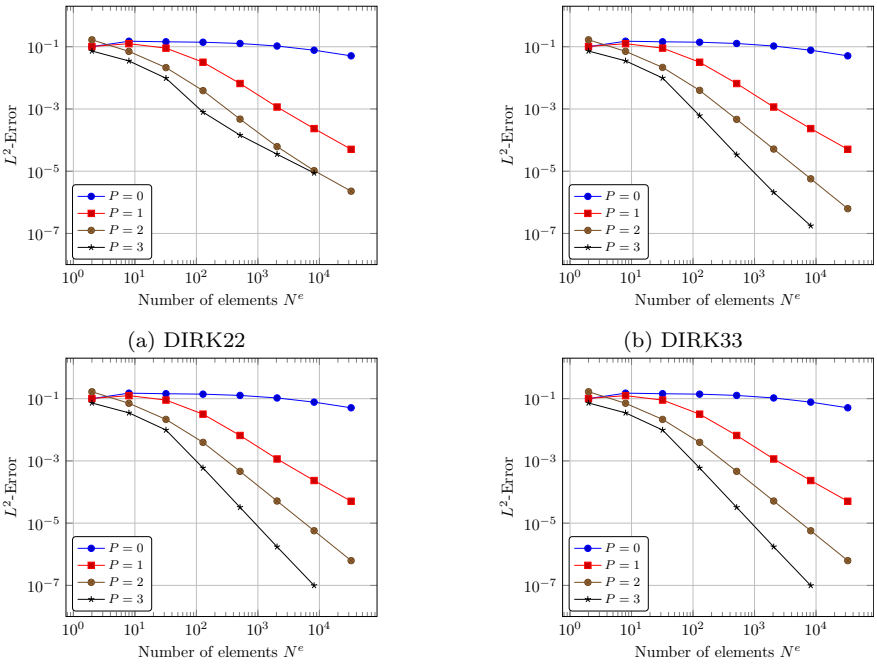
We verify the implementation using the same test cases as for the multistep methods, see Sec. 4.2.1. The settings of the problems, linear solver and Newton's method are the same as earlier. We do not present results for the first order DIRK method (=implicit Euler) which would be identical with BDF1 scheme. Instead, we present results for the second and third order method by Alexander [4], the fourth order method of Al-Rabeh [191] and the fourth order method of Hairer and Wanner [101].

Linear advection-diffusion equation

In Fig. 4.4, we present the solutions for the linear advection-diffusion test case, see Sec. 4.2.1. We see that the third and fourth order methods recover the correct order of convergence after few refinement steps. The second order DIRK scheme, see Fig. 4.4a, shows increasingly lower error for increasing polynomial degrees P which is in particular visible for $P = 2$ and $P = 3$.

Euler equations

In Fig. 4.5, we present results of the nonlinear test case, see Sec. 4.2.1, solved with DIRK methods. In all cases, the correct overall order of convergence $\max(Q, P+1)$ is observable for suitable polynomial degrees P and sufficiently refined meshes. As for the nonlinear test case, see Sec. 4.3.1, the methods with $Q < 4$ using polynomials of degree $P-1 > Q$ does not change the effective order of convergence, but decreases the overall error if it was dominated by the spatial error. Similar to the linear test case, the DIRK methods reach slightly lower errors than the BDF methods of the same order, see Fig. 4.3. Both fourth order time integrators show the lowest errors.



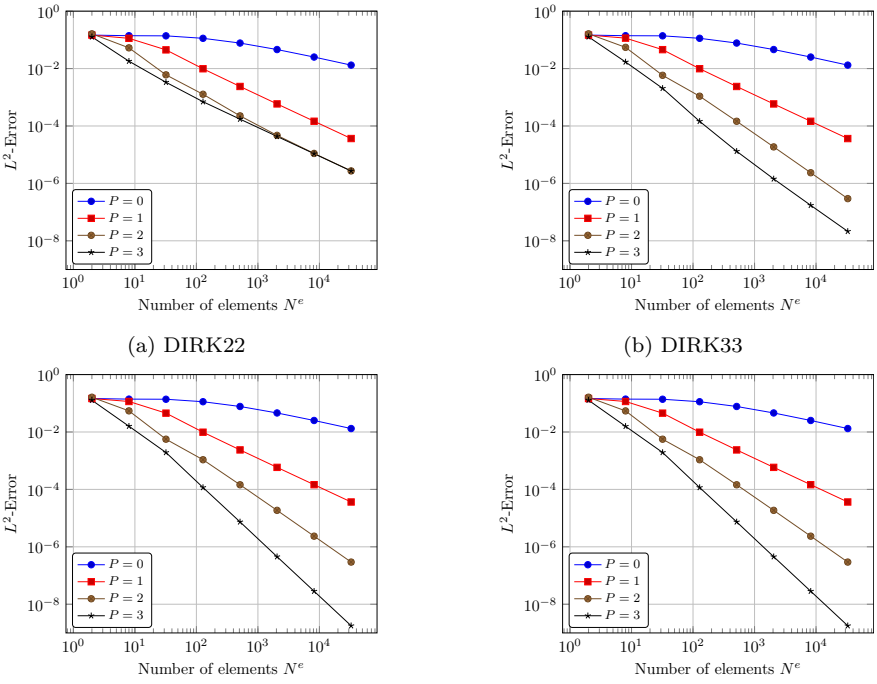
(a) DIRK22

(b) DIRK33

(c) DIRK34

(d) DIRK54

Figure 4.4.: Errors of the rotating Gaussian (linear convection-diffusion) test case solved with HDG and DIRKsQ methods. The problem is solved on the domain $\Omega = [-0.5, 0.5]^2$ up to $t_{\text{final}} = \frac{\pi}{4}$ and constant diffusivity $\varepsilon = 10^{-3}$. The velocity vector is given as $u(x) = (-4x_2, 4x_1)^{\mathrm{T}}$ and the CFL-number is CFL $= \frac{\sqrt{2}\pi}{16} \approx 0.278$. After sufficient refinements the methods reach the expected convergence rate $\max(P + 1, Q)$.



(a) DIRK22

(b) DIRK33

(c) DIRK34

(d) DIRK54

Figure 4.5.: Errors of the nonlinear test case (Euler equations) solved with HDG and DIRKsQ methods. The problem is solved on $\Omega = [0, 2]^2$ up to $t_{\text{final}} = 1$. The CFL-number is CFL ≈ 1.042. After sufficient refinements the methods reach the expected convergence rate $\max(P + 1, Q)$.

5. General linear methods for time integration

The previous chapter, see Ch. 4, presents two classes of time integrators that have been used for HDG space discretizations. In this chapter, we discuss general linear methods (GLM) [28–30, 35, 113, 114] that we have newly applied to the HDG method. These time integrators are a generalization of multistep and multistage methods. These methods combine the approach of the previous two classes of time integrators and allow to construct new schemes with advantageous properties. The results show that the methods are straight-forward to apply and also cover the time integrators of the previous chapter. Thus, the framework of general linear methods is also interesting from the implementation point of view. Some parts the results discussed in this chapter have been published in [120].

5.1. Introduction

The previously introduced time integrators can both be viewed as special cases of general linear methods (GLM) [28, 31, 32, 39, 114]. A big advantage over classical time integrators such as BDF and DIRK methods is that one can obtain methods with high accuracy that are L- and A-stable while having higher stage order $Q_i > 1, i = 1, \dots, s$. This is particularly important for very stiff problems [81]. Moreover, there are techniques available to adapt the time step size and the order of the scheme [34, 113, 114] and an extension to implicit explicit (IMEX) methods exists [233] as well. In this section, we study the coupling of an HDG method to a diagonally implicit multistage integration method (DIMSIM) in Nordsieck representation [30, 33].

Multistage methods, see Sec. 4.3, rely on only $r = 1$ external approximation – the solution at the previous time step – but compute $s \geq 1$ internal approximations during stages. Multistep methods, see Sec. 4.2, rely on $r \geq 1$ external approximations that are passed from one time step to another, but have only $s = 1$ internal approximation that equals the solution at the new time step. General linear methods allow the usage of several internal approximations $s \geq 1$ and external approximations $s \geq 1$.

Diagonally implicit Runge-Kutta methods may suffer from order reduction for very stiff problems [81]. This is caused by the rather low stage order. The stage order Q_i of the ith stage of a multistage order is given by

$$\|w_h^{n,i} - w(t^{n,i})\| = \mathcal{O}(\Delta t^{Q_i+1}).$$

For common DIRK methods with constant diagonal term a_{ii} the stage order is at most one [101].

In order to avoid the order reduction one can use methods of higher stage order. Implicit Runge-Kutta methods can be constructed with higher stage order if they are not only diagonally implicit [101, 177]. Another class of methods that can be used are general linear methods, like the diagonally implicit multistep integration methods (DIMSIMs) [30] as they can be constructed to have high order, high stage order and good stability properties [35]. While BDF methods suffer from the decreasing stability region for higher order, the DIRK schemes suffer from the increasing number of stages necessary to construct a method that is A- and L-stable. A DIRK method of order four that is A- and L-stable has at least five stages [101] while it is possible to construct a fourth order DIMSIM with the same stability properties that has only $s = 4$ internal approximations [243].

The update formulae of the DIMSIMs rely on four quantities. There are s internal approximations Y_i and the same number of right hand side evaluations of the ODE $G_j \coloneqq g(Y_j)$ that use the internal approximations. In the context of DIMSIMs G_j are also often referred to as stage derivatives. Additionally, r external approximations $y_j^{[n]}$ at the known time level t^n are used. The quantities are used to compute r of external approximations at the new time level t^{n+1}. In the literature, these quantities are often written as single vectors consisting of the data

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_s \end{bmatrix}, \quad G = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_s \end{bmatrix}, \quad y^{[n]} = \begin{bmatrix} y_1^{[n]} \\ y_2^{[n]} \\ \vdots \\ y_r^{[n]} \end{bmatrix}, \quad y^{[n+1]} = \begin{bmatrix} y_1^{[n+1]} \\ y_2^{[n+1]} \\ \vdots \\ y_r^{[n+1]} \end{bmatrix}. \tag{5.1}$$

The shape of the method depends heavily on the choice of values to be stored in $y^{[n+1]}$ and $y^{[n]}$ and the way it is represented in this notation [28, 33]. Pure multistep methods may store solutions at previous times $y^{n-r}, y^{n-r+1}, \ldots, y^{n-1}$, the corresponding derivatives $f(w^{n-r}), f(w^{n-r+1}), \ldots, f(w^{n-1})$ or both. Pure multistage methods only need to store the solution of the previous time w^{n-1}.

The shape of general linear methods is quite similar to the one of multistep methods (4.6) and multistage methods (4.8). Applying it to an ODE (4.1) leads to s internal approximations that have to be computed

$$Y_i = \sum_{j=1}^{s} a_{ij} \Delta t G_j + \sum_{j=1}^{r} u_{ij} y_j^{[n]}, \quad i = 1, \ldots, s. \tag{5.2a}$$

These intermediate solutions Y_i are then used to update the r external approximations

$$y_i^{[n+1]} = \sum_{j=1}^{s} b_{ij} \Delta t G_j + \sum_{j=1}^{r} v_{ij} y_j^{[n]}, \quad i = 1, \ldots, r, \tag{5.2b}$$

which includes the solution at the new time step [32]. The internal approximations are similar to the stages of a Runge-Kutta method while the external approximations are similar to the previous solutions used in multistep methods.

The order and stability of the method depends on the careful choice of real coefficients a_{ij}, u_{ij}, b_{ij} and v_{ij}. The coefficients of the method can be compactly written as matrices

$$\begin{array}{c|c} \mathrm{A} & \mathrm{U} \\ \hline \mathrm{B} & \mathrm{V} \end{array}, \quad \mathrm{A} \in \mathbb{R}^{s\times s}, \mathrm{B} \in \mathbb{R}^{r\times s}, \mathrm{U} \in \mathbb{R}^{s\times r}, \mathrm{V} \in \mathbb{R}^{r\times r} \tag{5.3}$$

and an abscissa vector $c \in \mathbb{R}^s$ that indicates the intermediate times $t^n + c_i\Delta t$ at which the stages are evaluated. This is the same as for multistage methods, see Sec. 4.3.

In this work, we focus on DIMSIMs in Nordsieck formulation [30, 33, 114]. These are closely related to (singly) diagonally implicit Runge-Kutta methods in the sense that the matrix A has the same structure as the diagonally implicit Runge-Kutta methods, see Tab. 4.2b. Thus, it is a lower triangular matrix with nonzero entries on the diagonal. Similar to the case of DIRK methods, it leads to s systems of equations that have to be solved for each internal approximation. Furthermore, these methods can be A- and L-stable just like Runge-Kutta methods. We use DIMSIMs of order $Q = 1$ to $Q = 3$ that were presented in [114]. The chosen DIMSIMs have stage order $Q_i = Q$. Therefore, these methods are also potentially suitable for problems where DIRK methods may suffer from order reduction.

We refer to the methods as DIMSIM1, DIMSIM2 and DIMSIM3 to distinguish between the schemes of different order. Each method has $s = Q$ internal and $r = Q + 1$ external approximations. In general, it is possible to construct the methods with only $r = Q$ external approximations. However, this formulation makes adaptation of the time step Δt more complicated. Therefore, the methods have been introduced in Nordsieck formulation [33, 174] as well. This means, instead of carrying data from several old time steps, $y^{[n]}$ is the Nordsieck vector

$$y^{[n]} = \begin{bmatrix} y(t^n) \\ \Delta t y'(t^n) \\ \Delta t^2 y^{(2)}(t^n) \\ \vdots \\ \Delta t^r y^{(r)}(t^n) \end{bmatrix} \tag{5.4}$$

that stores the derivatives of w. Using the specific form (5.4) has the advantage that time step adaptation can be easily incorporated since it only requires rescaling of the Nordsieck vector at the cost of an additional entry in $y^{[n]}$. This has been successfully applied in [29, 34, 113, 114] in the setting of ODEs. In this work, we do not pursue time step adaptation for DIMSIMs.

DIMSIMs in Nordsieck formulation are usually constructed from a DIMSIM in 'classical' formulation by transforming it. In order to differentiate between the classical formulation

and Nordsieck formulation the transformed matrices of the initial matrices (5.3) are often referred to as

$$\begin{array}{c|c} \mathrm{A} & \mathrm{P} \\ \hline \mathrm{G} & \mathrm{Q} \end{array}, \quad \mathrm{A} \in \mathbb{R}^{s\times s}, \mathrm{G} \in \mathbb{R}^{r\times s}, \mathrm{P} \in \mathbb{R}^{s\times r}, \mathrm{Q} \in \mathbb{R}^{r\times r}. \tag{5.5}$$

The naming also indicates, that the matrix A is not affected by the transformation. Therefore, the method keeps its diagonally implicit structure.

In practice, most users do not compute higher derivatves of the ODE due to the additional challenge. Therefore, one usually uses an approximation to the Nordsieck vector [33]. In the case of the first order method with $r = 2$ it is self-starting since the Nordsieck vector is given by

$$y^{[0]} = \begin{pmatrix} y_0 \\ \Delta t f(y_0) \end{pmatrix}.$$

Higher order methods require a different approach. In [244], the author constructed special Runge-Kutta schemes that compute an approximation to the Nordsieck vector at $t = 0$. In [113, 114], the author describes an approach where the higher order DIMSIMs are started from lower order DIMSIMs using suitable error estimators. In this work, we use an approach similar to the starting procedure of backwards differentiation formulae. We use an already available DIRK scheme of suitable order and compute r equidistant approximations to the solution at times $t^i = i \cdot \Delta t,\ i = 1, \ldots, r$. These values are used together with the given initial data to construct an approximation to the Nordsieck vector using interpolation. We interpolate the vector at t^r meaning that the first r are computed with the according DIRK method for schemes with $Q > 1$.

Backward differentiation formulae as general linear method For testing the implementation we choose to rewrite the already investigated BDF methods described in Sec. 4.2 as general linear method. In this case, the vector of external approximations contains the values at the r previous time steps

$$y^{[n]} = \left(w^n, \ldots, w^{n-s+1}\right)^{\mathrm{T}}.$$

Therefore, the methods have r external approximations and $s = 1$ internal approximations. The coefficients of the BDF3 method as general linear method are given in Tab. 5.1a.

Diagonally implicit Runge-Kutta methods as general linear method Similar to the BDF methods we can also rewrite DIRK methods as DIMSIMs. We do so in order to verify our implementation. As for the previous test cases we use test problems where an exact solution is available. Moreover, we can compare the numerical results of DIRK methods in the framework of general linear methods to the previous implementation of DIRK methods from the previous section. The coefficients of the DIRK22 method formulated as general linear method are given in Tab. 5.1b.

$\frac{6}{11}$	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$
$\frac{6}{11}$	$\frac{18}{11}$	$-\frac{9}{11}$	$\frac{2}{11}$
0	1	0	0
0	0	1	0

(a) BDF3 method formulated as general linear method.

$1-\frac{1}{\sqrt{2}}$	0	1
$\frac{1}{\sqrt{2}}$	$1-\frac{1}{\sqrt{2}}$	1
$\frac{1}{\sqrt{2}}$	$1-\frac{1}{\sqrt{2}}$	1

(b) DIRK22 method formulated as general linear method.

Table 5.1.: Coefficients of the BDF3 and DIRK22 method as used in the general linear method formulation.

Applying DIMSIMs to the HDG method In (3.19), the semi-discrete form of the equations is already in the shape of an ODE. Therefore, we have to solve (5.2a) and (5.2b) with slightly modified notation. In each stage i of the method we compute an internal approximation by solving

$$\mathcal{T}(\mathbf{w}_h^{n,i}, \mathbf{x}_h) = -\Delta t \sum_{j=1}^{i} a_{ij} \mathcal{N}(\mathbf{w}_h^{n,j}, \mathbf{x}_h) + \sum_{j=1}^{r} u_{ij} \mathcal{T}(y^{[n]}, \mathbf{x}_h), \quad \forall \mathbf{x}_h \in \mathbb{X}_h.$$

Once all stage values $w_h^{n,i}$ are known we obtain the updated solution from

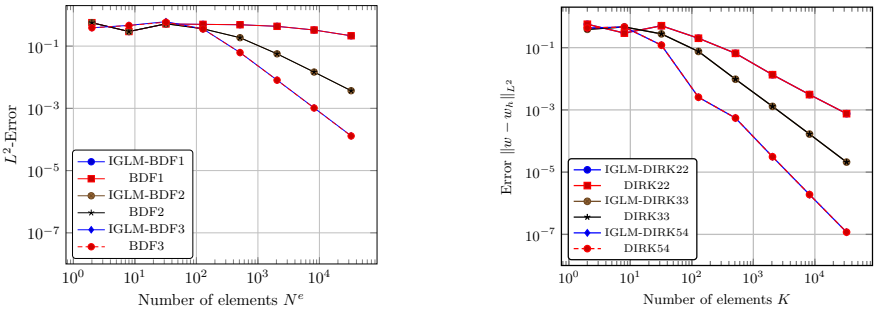
$$\mathcal{T}(y_i^{[n+1]}, \varphi_h) = -\sum_{j=1}^{s} b_{ij} \Delta t \mathcal{N}(\mathbf{w}_h^{n,j}, \mathbf{x}_h) + \sum_{j=1}^{r} v_{ij} \mathcal{T}(y_j^{[n]}, \mathbf{x}_h)$$

which only requires the local inversion of a mass matrix on each element. The term $y^{[n+1]}$ stores an approximation to the Nordsieck vector

$$y^{[n-1]} = \begin{bmatrix} w_h^n \\ \Delta t \partial_t w_h^n \\ \vdots \\ \Delta t^r \partial_t^r w_h^n \end{bmatrix} + \mathcal{O}(\Delta t^{Q+1}). \tag{5.6}$$

5.2. Numerical results

In this section we present numerical results obtained from the HDG discretization with DIMSIM time integrators in order to verify application of DIMSIMs to the HDG method. Besides the test cases shown in the previous Sections 4.2 and 4.3 we present additional results for the linear advection equation and of viscous flow around a circular obstacle for which vortices shed (Navier-Stokes equations). These results have been partly presented in [120].



(a) IGLM-BDF vs BDF

(b) IGLM-DIRK vs DIRK

Figure 5.1.: Errors of the linear convection test case solved with HDG and the BDF and DIRK methods from the previous sections and their respective formulations as general linear method. The problem is solved on the domain $\Omega = [0, 2]^2$ up to $t_{\text{final}} = 1$ and the constant velocity is $u = (1, 1)^{\mathrm{T}}$. The CFL-number is $\text{CFL} = \frac{1}{\sqrt{2}} \approx 0.707$. There is virtually no difference in the obtained errors.

Linear advection equation

The linear advection equation with constant velocity vector $u = (1, 1)^{\mathrm{T}}$ is solved on the domain $\Omega = [0, 2]^2$ up to $t_{\text{final}} = 1$. The viscous flux and source term are set to zero. The initial condition is given by

$$w(0, x) = \sin(2\pi x_1) \sin(2\pi x_2)$$

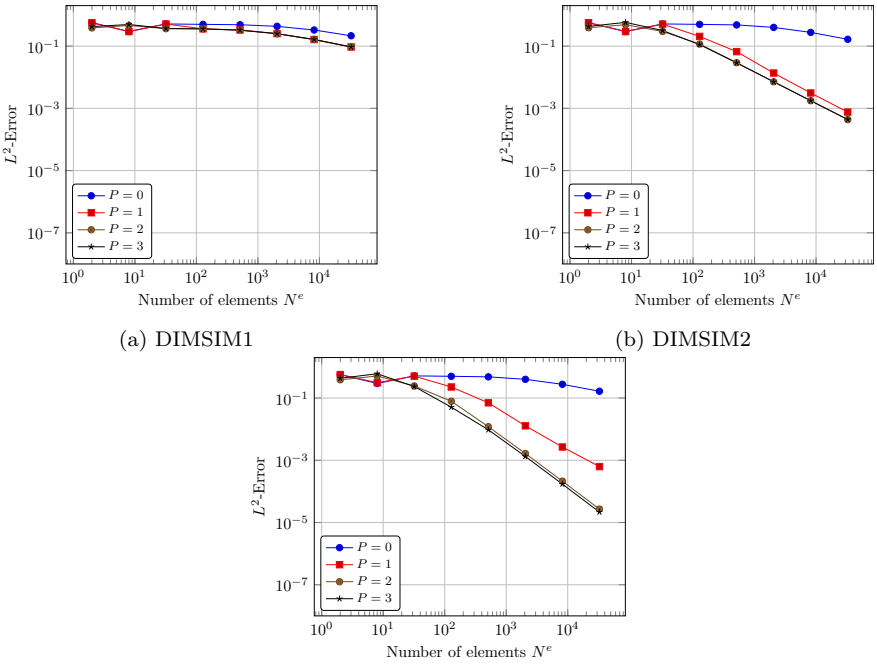
and periodic boundary conditions are used such that the exact solution at every time is given by

$$w(t, x) = \sin(2\pi x_1 - t) \sin(2\pi x_2 - t).$$

The stability parameter is set to $\alpha_c = \sqrt{2}$ and the system of equations is solved using a restarted GMRES until the residual drops below 10^{-12}. This is all carried out within the framework of the nonlinear solver. However, as it is a linear problem, only one Newton step is needed. The initial time step size on the coarsest mesh with $K = 2$ is $\Delta t = 0.5$. The mesh is uniformly refined, thus with each refinement the number of elements increases by a factor of 4 and the time step is halved.

In Fig. 5.1, we show numerical results of the BDF and DIRK methods and their respective formulations as general linear method in order to verify the implementation. One sees no differences in the errors of the two formulations. It indicates the correctness of implementation for multistage and multistep methods in GLM formulation.

In Fig. 5.2, the results obtained from thie DIMSIMs for different polynomial degrees are presented. All methods obtain the expected order of convergence once the spatial resolution



(a) DIMSIM1

(b) DIMSIM2

(c) DIMSIM3

Figure 5.2.: Errors of the linear convection test case solved with HDG and DIMSIMQ methods. The problem is solved on the domain $\Omega = [0, 2]^2$ up to $t_{\text{final}} = 1$ and the constant velocity is $u = (1, 1)^{\mathrm{T}}$. The CFL-number is $\text{CFL} = \frac{1}{\sqrt{2}} \approx 0.707$. After sufficient refinements the methods reach the expected convergence rate $\max(P + 1, Q)$.

is high enough. Further increasing the polynomial degree to $P > Q + 1$ decreases the error level without changing the slope. Therefore, the error is clearly dominated by the temporal error. This indicates the correctness of the implementation and the validity of our start-up procedure for the DIMSIMs in Nordsieck formulation.

Linear convection-diffusion equation

The second test problem is the linear advection-diffusion test case also used in the previous sections with exactly the same settings, see Sec. 4.2.1 for a detailed description. In Fig. 5.3 we present the results obtained with the DIMSIMs of order $Q = 1$ up to $Q = 3$. The behavior of the methods is identical to the behavior of the DIRK methods and the BDF1 method, see Fig. 4.4 and 4.2. This indicates that the DIMSIMs obtain the same accuracy in time as the

	Sr	c_D
Gopinath [92]	1.3406	0.1866
Henderson [107]	1.336	—
Williamson [238]	—	0.1919

Table 5.2.: Values of the Strouhal number Sr and the drag coefficient from the literature.

DIRK methods and the higher stage order does not have significant influence on the solution.

Euler equation

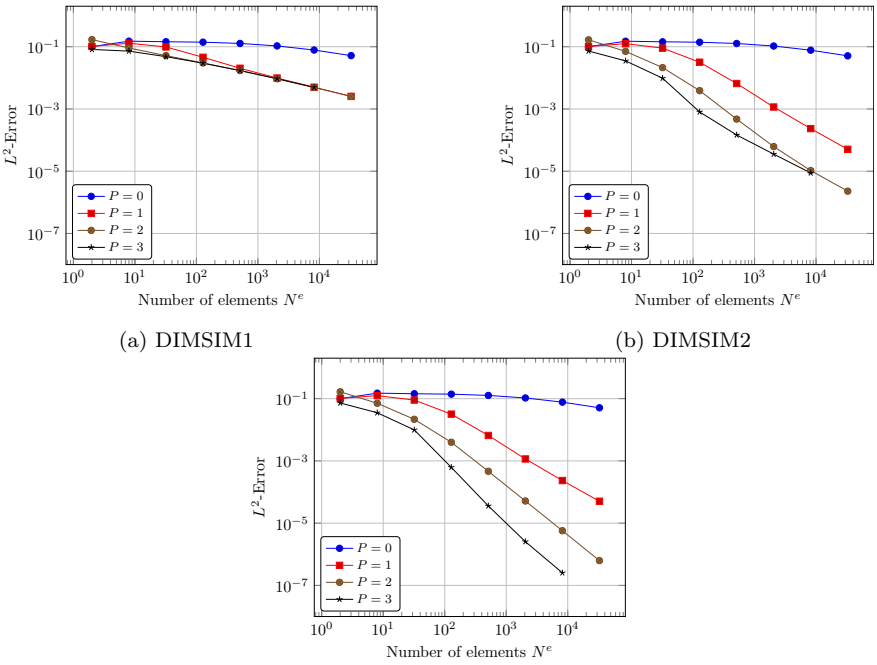
The correctness of the nonlinear solver is verified using the nonlinear problem posed by Euler equation with the settings described in Section 4.2.1. In Fig. 5.4, we present the numerical results. Again, the behavior is similar to the results of the DIRK methods of the respective order and the BDF1 method, see Fig. 4.5 and 4.3. The methods show the correct order of convergence $\max(P+1, Q)$. Increasing the spatial resolution to $P > Q - 1$ does not change the slope, and thus not the total order of convergence, but may reduce the error. The latter is mainly observed for the third order integrator, see Fig. 5.4c.

Navier-Stokes equations

Besides the Euler equations (2.9) a common equation that has to be solved in CFD are the Navier-Stokes equations (2.22). Thus, we present results for the compressible, viscous flow described by these equations. We consider the flow around a cylinder at Reynolds number $\mathrm{Re} = 180$ and Mach number $\mathrm{Ma} = 0.2$. At these flow conditions vortices shed from the cylinder which is known as a Kármán vortex street [7, 207].

We compute the solution on a mesh that extends to 20 diameters around the cylinder. The mesh has $K = 2916$ elements and it is the same that has been used in previous publications [119, 242]. We use polynomials of degree $P = 1, 2, 3$. The flow field is initialized with free stream conditions. At $t \approx 750$ the vortex street develops. We look at the fully evolved vortex street in the interval $t \in [10^3; 10^4]$ and compute the mean drag coefficient c_D and the Strouhal number Sr. These can be compared to data from the literature given in Tab. 5.2. The system of equations is solved using Newton's method until the residual drops below 10^{-10}. The arising linear system is then solved with a restarted GMRES until the relative residual drops below 10^{-10}.

The data obtained from the computations using the DIMSIMs are reported in Tab. 5.3. In general, the results are very similar to the results from the literature and our previous publications [119, 242].

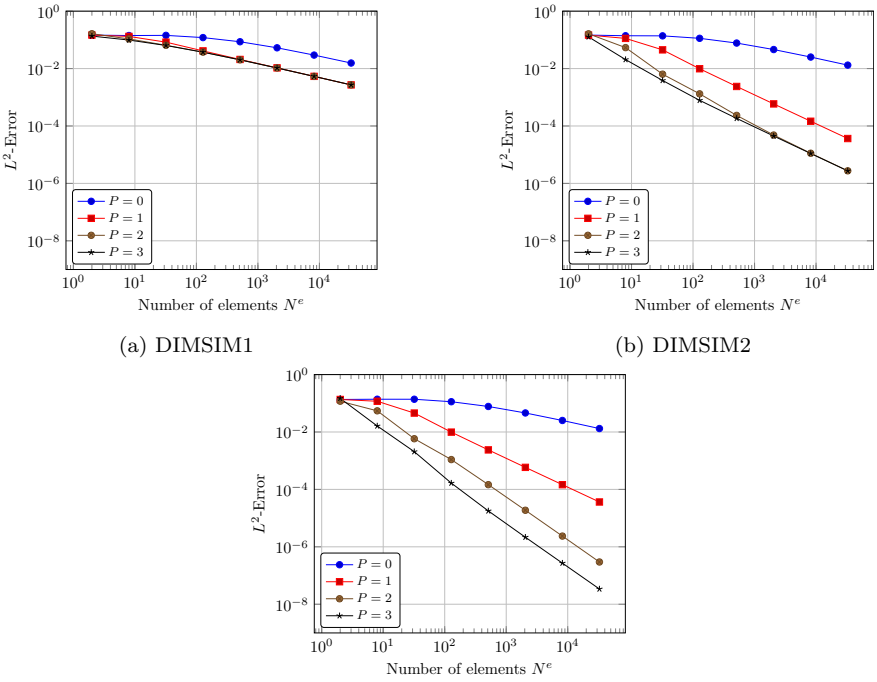


(a) DIMSIM1

(b) DIMSIM2

(c) DIMSIM3

Figure 5.3.: Errors of the rotating Gaussian (linear convection-diffusion) test case solved with HDG and DIMSIMs. The problem is solved on the domain $\Omega = [-0.5, 0.5]^2$ up to $t_{\text{final}} = \frac{\pi}{4}$ and constant diffusivity $\varepsilon = 10^{-3}$. The velocity vector is given as $u(x) = (-4x_2, 4x_1)^{\mathrm{T}}$ and the CFL-number is CFL $= \frac{\sqrt{2}\pi}{16} \approx 0.278$. After sufficient refinements the methods reach the expected convergence rate $\max(P + 1, Q)$.



(a) DIMSIM1

(b) DIMSIM2

(c) DIMSIM3

Figure 5.4.: Errors of the nonlinear test case (Euler equations) solved with HDG and DIMSIMs. The problem is solved on $\Omega = [0, 2]^2$ up to $t_{\text{final}} = 1$. The CFL-number is CFL ≈ 1.042. After sufficient refinements the methods reach the expected convergence rate $\max(P + 1, Q)$.

	$P=1$		$P=2$		$P=3$	
Δt	Sr	c_D	Sr	c_D	Sr	c_D
1	0.1733	1.2110	0.1733	1.2164	0.1733	1.2171
5	0.0000	0.9723	0.1222	0.9542	0.1238	0.9492
10	0.0000	0.9723	0.0000	0.9228	0.0000	0.9181

(a) Results obtained for DIMSIM1.

	$P=1$		$P=2$		$P=3$	
Δt	Sr	c_D	Sr	c_D	Sr	c_D
1	0.1898	1.3455	0.1898	1.3649	0.1898	1.3651
5	0.1849	1.3449	0.1882	1.3714	0.1882	1.3727
10	0.1733	1.3317	0.1774	1.3401	0.1774	1.3405

(b) Results obtained for DIMSIM2.

	$P=1$		$P=2$		$P=3$	
Δt	Sr	c_D	Sr	c_D	Sr	c_D
1	0.1898	1.3448	0.1898	1.3645	0.1898	1.3649
5	0.1832	1.3216	0.1882	1.3377	0.1882	1.3385
10	0.1667	1.2803	0.1708	1.2840	0.1708	1.2845

(c) Results obtained for DIMSIM3.

Table 5.3.: Values of Strouhal number Sr and drag coefficient c_D for viscous flow around a cylinder at Reynolds number $\mathrm{Re} = 180$ and Mach number $\mathrm{Ma} = 0.2$. The CFL-numbers are $\mathrm{CFL} \approx 15.085$ ($\Delta t = 1$), $\mathrm{CFL} \approx 75.425$ ($\Delta t = 5$) and $\mathrm{CFL} \approx 150.85$ ($\Delta t = 10$). The HDG method with varying polynomial degree P and a mesh with $K = 2916$ elements and DIMSIMs of order 1 to 3 have been used to solve the problem.

When a coarse resolution in time is used, i.e. $\Delta t = 10$, with the first order integrator, see Tab. 5.3a, the mean drag coefficient is far off the values from the literature. Moreover, the simulation does *not* capture the time-dependent behavior of the test case as one can see from the Strouhal number being zero. Only for $\Delta t = 1$ the first order method recovers the time-dependency for all polynomial degrees P. Although the values are still out of the range of the values from literature, they increase for decreasing Δt and thus tend to approach the right values. We have observed the failure to catch the time-dependency of this test case for time steps $\Delta t > 10$ even for higher order time integrators before [119] and thus is not unexpected.

The second and third order DIMSIMs, see Tab. 5.3b and 5.3c, both capture the time-dependent behavior in all cases. Even for $\Delta t = 10$ the Strouhal number is already close to the correct range of values and for smaller time steps the Strouhal is in between the values reported in the literature, see Tab. 5.2. Moreover, the approximation of the Strouhal number only differs slightly for increasing P while it changes much more for smaller time step size Δt. Thus, the Strouhal number profits much more from smaller time step sizes than from an improved spatial resolution.

This trend can also be observed for the mean drag coefficient. In most cases, the mean drag coefficient grows slowly for decreasing time step sizes. However, the second order DIMSIM overpredicts the mean drag coefficient for $P > 1$ and $\Delta t = 5$, but then the coefficient decreases again for $\Delta t = 1$. For the smallest time step size $\Delta t = 1$ the mean drag coefficients of both time integrators are very close to each other and slightly above the values from the literature.

6. Multiderivative time integrators for the hybridized discontinuous Galerkin method

The previous chapters introduced 'classical' multistep and multistage methods, see Ch. 4, and their generalization in the framework of general linear methods, see Ch. 5. In the current chapter, we introduce multiderivative time integrators [28, 67, 101, 139, 190, 214, 231] for the HDG method. As the name suggests, these methods incorporate additional time derivatives in order to achieve a high order time accuracy.

The chapter consists of two parts that discuss two different ways to discretize the higher order time derivatives. The first one deals with the discretization of multiderivative methods using a Cauchy-Kowalevski procedure. In the field of PDEs it is also referred to as Lax-Wendroff procedure. It uses the PDE to express time derivatives as space derivatives and has been successfully applied for the discretization of explicit multiderivative methods [214, 231].

In the second part, we present a new approach that introduces special auxiliary unknowns to express additional time derivatives. This approach is straight forward to apply to linear problems. It has also the advantage that the number of additional unknowns to be introduced is independent of the number of space dimensions d in contrast to the Cauchy-Kowalevski approach. We also discuss how the new approach can be extended to nonlinear problems.

This chapter extends the work on time integrators for the HDG method in several ways. To the knowledge of the author, multiderivative methods have not been used for HDG methods before. Moreover, the application of *implicit* multiderivative methods for the solution of PDEs seem to have gotten less attention than explicit methods. The second approach for discretizing the additional time derivatives has not been discussed in literature before, to our knowledge. The results of this chapter have been partially presented in [121, 126, 212].

6.1. Introduction

The time integrators presented in the previous sections obtained a high-order discretization by using a time history of the solution or by introducing intermediate steps. A third possible way directly follows from the Taylor series

$$w(t+\Delta t) = w(t^n) + \Delta t \partial_t w(t) + \frac{\Delta t^2}{2}\partial_t^2 w(t) + \ldots + \frac{\Delta t^M}{M!}\partial_t^M w(t) + \mathcal{O}(\Delta t^{M+1}). \tag{6.1}$$

These time integration methods are also referred to as Taylor (series) methods [28]. The approach presents a straightforward approach on how to construct a method of arbitrary order by incorporating additional time derivatives. It is no surprise that it is also possible to combine the previous approaches, multistep and multistep methods, with additional time derivatives [86, 87, 100]. In this case, we refer to the method as a multiderivative method while the Taylor methods refer to a method directly obtained by a Taylor expansion in time.

Lax and Wendroff used multiderivative methods in 1960 [139]. Various authors have been working on discontinuous Galerkin method with Lax-Wendroff type time integration [96, 188, 189] and called these methods Lax-Wendroff discontinuous Galerkin (LWDG) methods. The method has also been applied to weighted essentially non-oscillatory (WENO) methods [190, 214]. However, the methods have just recently (reattracted) interest for application to problems in CFD. Seal et al. introduced a framework for multiderivative multistage methods [214] for DG and WENO schemes applied to hyperbolic PDEs. Further work on two-derivative time integrators has been carried out by different authors [42, 231]. Recent publications use multiderivative time integrators to solve Navier-Stokes [173] and Euler equations [175]. All of the publications use explicit multiderivative methods.

In this work, we focus on *implicit* one-step methods with multiple time derivatives. The time integrators should be at least A-stable and therefore cannot be explicit. An additional challenge in this setting is the proper approximation of the additional time derivatives. This has to be done in a stable, accurate, and efficient manner. Moreover, it should be done in a straightforward way that allows for a simple and efficient implementation for its practical use. In the explicit case it is usually unnecessary to approximate the time derivatives with the full accuracy of the space discretization as the time derivatives are scaled by powers of the time step size (6.1). Then, the CFL condition of explicit methods enforces that $\Delta x \approx \Delta t$. We cannot expect this to hold for implicit methods where we want to use large time steps.

In the following, we discuss two different approaches for the approximation of the higher time derivatives. The first one follows a Cauchy-Kowalevski procedure to replace time by space derivatives. This has been used for explicit time integrators, but we also introduce additional unknowns for the approximation of space derivatives introduced. This is necessary in the implicit case where the time step size is usually much larger than the mesh size.

The second approach introduces auxiliary variables that directly approximate the needed time derivatives. We show that it has straightforward to use and has several advantages over the Cauchy-Kowalevski approach for implicit time integration.

In our first publication on multiderivative publications we have solved a simple linear problem and have obtained promising results [121]. Further, investigation has showed the instable behavior when the Cauch-Kowalevski procedure is used [126]. We solve this problem by discretizing the additional time derivatives in a new way in our most recent publication [212].

Order	(k,l)	α_1	α_2	α_3	β_1	β_2	β_3
3	$(1,2)$	$1/3$	0	0	$-2/3$	$1/6$	0
4	$(2,2)$	$1/2$	$1/12$	0	$-1/2$	$1/12$	0
5	$(2,3)$	$2/5$	$1/20$	0	$-3/5$	$3/20$	$-1/60$
6	$(3,3)$	$1/2$	$1/10$	$1/120$	$-1/2$	$1/10$	$-1/120$

Table 6.1.: Coefficients of two-point collocation schemes.

6.2. Lax-Wendroff/Cauchy-Kowalevski approach

The multiderivative time integrators considered in this section are two-point collocation methods. That means that the update is computed by approximating the solution only at the current time level t^n and the new time level t^{n+1} while using several derivatives of the unknown. The shape of such a method is given by

$$\sum_{j=0}^{l+k} \Delta t^j (\partial_t^j w^{n+1}) \mathcal{P}^{(l+k-j)}(0) = \sum_{j=0}^{l+k} \Delta t^j (\partial_t^j w^n) \mathcal{P}^{(l+k-j)}(1) \tag{6.2}$$

with polynomial $\mathcal{P}(t) = \frac{t^k (t-1)^l}{(k+l)!}$. We can write these methods more compactly as

$$w^{n+1} = w^n + \sum_{m=1}^{M} \Delta t^m (\alpha_m \partial_t^m w^n - \beta_m \partial_t^m w^{n+1}) \tag{6.3}$$

with coefficients α_m, $\beta_m \in \mathbb{R}$ being derived from (6.2). In this section, we focus on methods with two time derivatives, i.e. $0 < k \leq M$, $l = M$ and $M = 2$. In this case, the method can be written as

$$w^{n+1} = w^n + \Delta t \left(\alpha_1 \partial_t w^n - \beta_1 \partial_t^1 w^{n+1}\right) + \Delta t^2 \left(\alpha_2 \partial_t^2 w^n - \beta_2 \partial_t^2 w^{n+1}\right). \tag{6.4}$$

The stability properties and accuracy of the method depend on the choice of the coefficients which are determined from (6.2). In our case, we focus on implicit two-point collocation methods. Coefficients of methods of order three to six are given in Tab. 6.1. Note that in the current section we use only the third order and fourth order method. The fourth order method is A-stable while the third order method is A- and L-stable [126].

For simplicity, we want to apply the method (6.4) to a hyperbolic equation

$$\partial_t w + \nabla \cdot f_c(w) = 0$$

as introduced in Sec. 2.3. Applying the two-derivative method to the hyperbolic equation, the method reads

$$w^{n+1} = w^n + \Delta t \left(\alpha_1 \partial_t w^n - \beta_1 \partial_t w^{n+1}\right) + \Delta t^2 \left(\alpha_2 \partial_t^2 w^n - \beta_2 \partial_t^2 w^{n+1}\right) \tag{6.5}$$

where we have to approximate the time derivatives. For this, we use the Cauchy-Kowalevski procedure

$$\begin{aligned} \partial_t w &= -\nabla \cdot f_c(w) \\ \partial_t^2 w &= -\nabla \cdot \partial_t f_c(w) = -\nabla \cdot \left(f_c'(w) \partial_t w \right) = \nabla \cdot \left(f_c'(w) \nabla \cdot f_c(w) \right) \\ &\vdots \end{aligned} \tag{6.6}$$

where the PDE is used to replace time derivatives by space-derivatives. In the context of PDEs, this procedure for deriving higher order time discretizations is often referred to as a Lax-Wendroff procedure due to the work of Lax and Wendroff [139]. In theory, this approach can be continued up to arbitrary time derivatives. We want to emphasize that this approach is also applicable to viscous PDEs as done in our publication [126] in 1D. In our case, the discretization of a hyperbolic equation leads to a convection-diffusion type equation after discretizing in time. The approximation of higher order space derivatives using an HDG discretization is also discussed in [43, 46, 64] where additional auxiliary variables are introduced in a similar fashion as presented in this thesis.

For the application of the implicit two-derivative methods we rely on an accurate representation of the derivatives as we aim to use large time steps Δt. For this we can reuse already implemented codes by using the LDG formulation introduced in Sec. 3.2. In the case of a hyperbolic equation we can interpret $\nabla \cdot (f_c'(w) \nabla \cdot f_c(w))$ as an additional viscous flux. The hybridized discontinuous Galerkin method is especially beneficial when using the auxiliary unknown σ compared to the unhybridized LDG method. As described in Sec. 3.3 the number of unknowns only increases locally, while the globally coupled systems is still only coupled by the hybrid unknown λ independent of the auxiliary variable σ.

An HDG discretization of two-derivative time integrators

In order to give a compact representation we introduce

$$\begin{aligned} \partial_t w &= -\nabla \cdot f_c(w) =: \mathcal{R}(w) \\ \partial_t^2 w &= \nabla \cdot (f_c'(w) \nabla \cdot f_c(w)) =: \mathcal{R}^2(w) \end{aligned} \tag{6.7}$$

for the first and second time derivative. Then, we can express (6.5) as

$$w^{n+1} = w^n + \Delta t \left(\alpha_1 \mathcal{R}(w^n) + \alpha_2 \mathcal{R}(w^{n+1}) \right) - \Delta t^2 \left(\beta_1 \mathcal{R}^2(w^n) + \beta_2 \mathcal{R}^2(w^{n+1}) \right).$$

For the time discretization we use the shorthand notation

$$\partial_t \mathcal{T}(\mathbf{w}_h, \mathbf{x}_h) = \mathcal{N}(\mathbf{w}_h, \mathbf{x}_h). \tag{3.19 revisited}$$

Then, the system of equations to be solved is

$$\frac{1}{\Delta t}\left(\mathcal{T}(\mathbf{w}_h^{n+1}, \mathbf{x}_h) - \mathcal{T}(\mathbf{w}_h^{n}, \mathbf{x}_h)\right) = \sum_{i=1}^{2} \mathcal{N}_i(\mathbf{w}_h^{n+i-1}, \mathbf{x}_h)$$

with

$$\mathcal{N}_i(\mathbf{w}_h^{n+i-1}, \mathbf{x}_h) = \begin{pmatrix} \mathcal{N}_i^{\text{aux}}(\mathbf{w}_h^{n+i-1}, \tau_h) \\ \mathcal{N}_i^{\text{eq}}(\mathbf{w}_h^{n+i-1}, \varphi_h) \\ \mathcal{N}_i^{\text{hyb}}(\mathbf{w}_h^{n+i-1}, \mu_h) \end{pmatrix}.$$

The first entry, $\mathcal{N}_i^{\text{aux}}$, refers to the equation of auxiliary unknown σ (3.15a) that is introduced to approximate the second spatial derivatives. We focus on problems on domains with periodic boundaries such that the integrals incorporating boundary conditions vanishes. The other entries, $\mathcal{N}_i^{\text{eq}}$, refers to the equation stemming from the discretization of the initial PDE (3.15b), and $\mathcal{N}_i^{\text{hyb}}$ refers to the equation stemming from the hybrid approach to ensure that the flux in the normal direction is conservative (3.15c). These two equations are slightly modified to incorporate the space derivatives of second order and the coefficients of the time integrator. The fluxes f_c and f_v are replaced by

$$\begin{aligned} f_c(w^{n+i-1}) &= \alpha_i f_c(w^{n+i-1}), \\ f_v(w^{n+i-1}, \nabla w^{n+i-1}) &= -\beta_i \Delta t f_c'(w^{n+i-1})(\nabla \cdot f_c(w^{n+i-1})) \end{aligned} \tag{6.8}$$

to obtain the two-derivative formulation without changing the initial HDG formulation (3.15). Note that the ∇w is needed in order to express $\nabla \cdot f_c(w)$. The same numerical fluxes (3.17) are used with the fluxes (6.8).

This approach is, in principle, straightforward as the Jacobian $f_c'(w)$ is available in the code already. However, for the application of Newton's method, see Sec. 3.3.1, the Jacobian of our newly introduced viscous flux f_v is needed. This adds additional complexity as it introduces the derivative of the Jacobian and the derivative of $\nabla \cdot f_c(w)$.

6.3. Numerical results

In this section we show two dimensional numerical results for systems of equations. The results can also be found in our publication [126]. We solve the (nonlinear) system of equations using Newton's method. The resulting linear system is solved using GMRES with block Jacobi preconditioning until the relative residual drops below 10^{-12}. Newton's method is carried out until the L_2-norm of the residual drops below 10^{-10}.

Linear convection equation

We first examine a system of two uncoupled linear convection equations and solve it on $\Omega = [0, 2]^2$ up to final time $t_{\text{final}} = 0.1$. This test case is used to verify that the implementation

is correct and that the approach is valid. The flux is chosen to be $f_c(w) = (f_{c,1}, f_{c,2})$ with

$$f_{c,1}(w) = \mathrm{A}_1 w, \quad f_{c,2}(w) = \mathrm{A}_2 w. \tag{6.9}$$

The vector of unknowns is $w = (w_1, w_2)^{\mathrm{T}}$. The matrices for this linear system are given by

$$\mathrm{A}_1 = \begin{pmatrix} \frac{1}{3} & \frac{8}{3} \\ \frac{16}{3} & -\frac{7}{3} \end{pmatrix}, \ \mathrm{A}_2 = \begin{pmatrix} -\frac{7}{3} & -\frac{5}{3} \\ -\frac{10}{3} & -\frac{2}{3} \end{pmatrix}. \tag{6.10}$$

These matrices have the same eigenvector basis, which means we can express these as $\mathrm{A}_1 = \mathrm{S}\mathrm{D}_{\mathrm{A}_1}\mathrm{S}^{-1}$ and $\mathrm{A}_2 = \mathrm{S}\mathrm{D}_{\mathrm{A}_2}\mathrm{S}^{-1}$ with

$$\mathrm{D}_{\mathrm{A}_1} = \begin{pmatrix} -5 & 0 \\ 0 & 3 \end{pmatrix}, \mathrm{D}_{\mathrm{A}_2} = \begin{pmatrix} 1 & 0 \\ 0 & -4 \end{pmatrix}, \mathrm{S} = \begin{pmatrix} -\frac{1}{2} & 1 \\ 1 & 1 \end{pmatrix}, \mathrm{S}^{-1} = \begin{pmatrix} -\frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix}. \tag{6.11}$$

We choose the initial conditions to be

$$w_0(x) = \begin{pmatrix} \sin(\pi(x+y)) \\ \sin(\pi(x+y)) \end{pmatrix}. \tag{6.12}$$

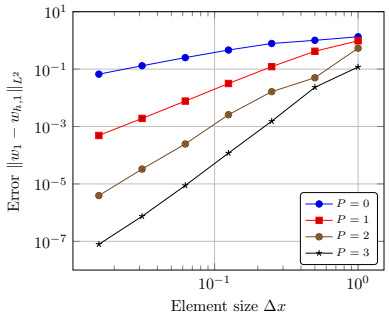
In this case, the exact solution is given by

$$w(t,x) = \begin{pmatrix} \sin(\pi(x+y+t)) \\ \sin(\pi(x+y+t)) \end{pmatrix} \tag{6.13}$$

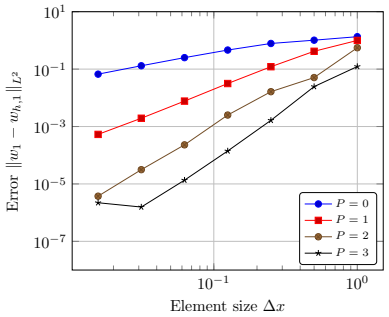
if periodic boundary conditions are employed. Results are presented in Fig. 6.1 for the ratio $\frac{\Delta t}{\Delta x} = 0.025$. The errors for w_1 and w_2 are perfectly identical which indicates the correctness of the implementation for systems of equations. The third order integrator reaches the expected order of convergence in all cases. For $P = 3$, the method is still third-order accurate, but it has a lower error than in the case of $P = 2$. The fourth-order integrator, however, does not achieve fourth-order in time. In the case $P < 3$, the method gets close to the expected order of $P + 1$ while for $P = 3$ the order deteriorates during the refinements. After the sixth refinement it seems not to converge any further. Most likely, this behavior is observed due to stability issues of the fourth-order integrator. Additionally the condition $\frac{\Delta t}{\Delta x} = 0.025$ is not desirable, but similar behavior has been also observed for the convection equation in one space dimension [126].

Euler equations

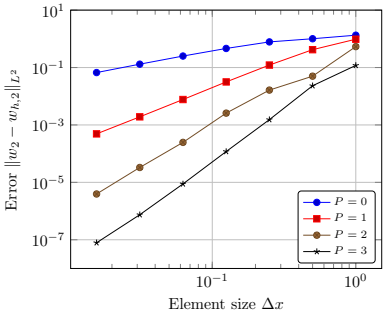
The second test case is the nonlinear problem described in Sec. 4.2.1. Thus, we solve the Euler equations with periodic boundary conditions. However, we solve it up to final time $t_{\text{final}} = 0.5$ due to the rather strict time step requirement also observed for this test case and we use $\frac{\Delta t}{\Delta x} = 0.05$. This is again a very strict restriction we do not want for an implicit method.



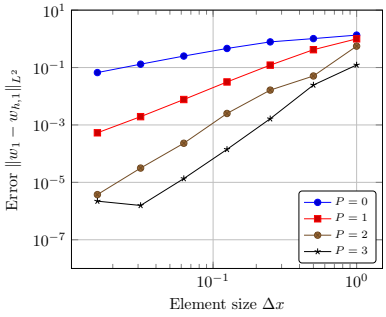
(a) Third-order integrator, error of w_1.



(b) Fourth-order integrator, error of w_1.

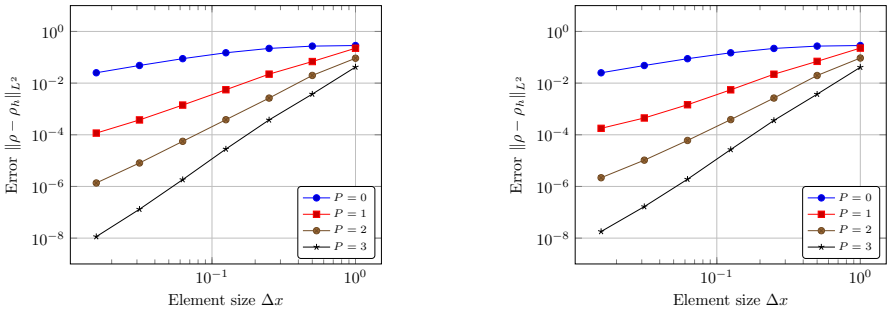


(c) Third-order integrator, error of w_2.



(d) Fourth-order integrator, error of w_2.

Figure 6.1.: Numerical results for the linear coupled convection equation obtained by the HDG method. The problem is solved on $\Omega = [0, 2]^2$ up to final time $t_{\text{final}} = 0.1$. The CFL-number is CFL ≈ 0.124. Temporal integration is performed with the third-order integrator and the fourth-order integrator. We show the results for both components w_1 and w_2.



(a) Third-order method.

(b) Fourth-order method.

Figure 6.2.: Numerical results for the Euler equations obtained by the HDG method. The problem is solved on $\Omega = [0, 2]^2$ up to final time $t_{\text{final}} = 0.5$. The CFL-number is CFL ≈ 0.104. Temporal integration is done via the third-order integrator and the fourth-order integrator. We show the error in the density ρ.

Both integrators produce similar errors with the third-order integrator having slightly lower errors. The conclusion is that the higher-order integrator does not exhibit any serious advantage over the lower-order integrator for this test case. For this problem, we find that increasing the polynomial order always increases the rate of convergence, which is in contrast to the previous cases. However, the method does not achieve optimal convergence. The third order integrator shows convergence rates larger than 3 for $P = 3$, indicating that the spatial error is dominating. In the linear test case, going from $P = 2$ to $P = 3$ decreased the error level, whereas the slope of the error graph stayed almost constant (cf. Fig. 6.1). This, in fact, *increases* the slope. Nevertheless, both integrators show a slight decrease of the convergence rate during refinements. We attribute this again to the poor stability of the scheme.

6.4. Analysis of the unstable behavior

We have seen that the multiderivative time integrators behave unexpected in the sense that they underlie certain restrictions on the time step size Δt depending on the mesh size Δx. Moreover, in the nonlinear test case no clear statement about the order of accuracy in time can be made.

Overall there are several disadvantages of the approach. The strict time step restriction prevents the efficient the application of the implicit multiderivative time integrators as the restriction seems to be even stricter than for explicit methods. Additionally, computing a single, implicit time step is much more time consuming due to the system of equations that has to be constructed and solved. The presented approach also has an additional auxiliary

variable and a rather complex viscous flux — especially for the Euler equations — compared to standard methods presented in the previous section, see Sec. 4.

In order to understand the issues of the method better, we carry out a von Neumann stability analysis. We discretize the linear convection equation using finite difference space discretizations in one space dimension on an infinite domain

$$\partial_t w + u\partial_x w = 0, \quad u > 0 \tag{6.14}$$

with constant, positive convection velocity u. We follow the Cauchy-Kovalevski procedure (6.6) to replace the time by space derivatives. This gives

$$\partial_t w = -u\partial_x w, \quad \partial_t^2 w = \partial_t \left(-u\partial_x w\right) = -u\partial_x \left(\partial_t w\right) = -u\partial_x \left(-u\partial_x w\right) = u^2\partial_x^2 w \tag{6.15}$$

for the derivatives. For the finite difference discretization we assume a constant mesh size $\Delta x = x_j - x_{j-1}$. The derivatives are discretized by an upwind difference for the convective term to respect the direction of information propagation and a central difference for the second space derivative

$$\partial_x w(x_j) \approx \frac{w_j - w_{j-1}}{\Delta x}, \quad \partial_x^2 w(x_j) \approx \frac{w_{j-1} - 2w_j + w_{j+1}}{\Delta x^2}. \tag{6.16}$$

This follows the 'standard' approach for convection-diffusion equations. The stability analysis of the fourth order method is presented first and subsequently the third order method is analyzed. A more detailed description of the analysis can be found in the appendix, see Sec. B.

Fourth order method

The fourth order time integrator is given by the formula

$$w^{n+1} = w^n + \frac{\Delta t}{2}\left(\partial_t w^n + \partial_t w^{n+1}\right) + \frac{\Delta t^2}{12}\left(\partial_t^2 w^n - \partial_t^2 w^{n+1}\right)$$

The method is applied to the convection equation (6.14) letting the time derivatives be replaced by space derivatives (6.15) such that we obtain

$$w^{n+1} = w^n - \frac{u\Delta t}{2}\left(\partial_x w^n + \partial_x w^{n+1}\right) + \frac{u^2\Delta t^2}{12}\left(\partial_x^2 w^n - \partial_x^2 w^{n+1}\right). \tag{6.17}$$

Then, the space derivatives are discretized by the finite differences (6.16) and we introduce $\nu := \frac{u\Delta t}{\Delta x}$ to simplify the notation. The term ν refers to the CFL number. The method should be stable for all choices of ν. This leads to

$$\begin{aligned} w_j^{n+1} = w_j^n &- \frac{\nu}{2}\left(w_j^n - w_{j-1}^n + w_j^{n+1} - w_{j-1}^{n+1}\right) \\ &+ \frac{\nu^2}{12}\left(w_{j-1}^n - 2w_j^n + w_{j+1}^n - w_{j-1}^{n+1} + 2w_j^{n+1} - w_{j+1}^{n+1}\right). \end{aligned}$$

The terms are reordered such that the unknowns at the new time level $n+1$ are on the left hand side while the unknowns at the old time level n are on the right hand side

$$\begin{aligned} w_j^{n+1} + \frac{\nu}{2}\left(w_j^{n+1} - w_{j-1}^{n+1}\right) + \frac{\nu^2}{12}\left(w_{j-1}^{n+1} - 2w_j^{n+1} + w_{j+1}^{n+1}\right) \\ = w_j^n - \frac{\nu}{2}\left(w_j^n - w_{j-1}^n\right) + \frac{\nu^2}{12}\left(w_{j-1}^n - 2w_j^n + w_{j+1}^n\right). \end{aligned} \tag{6.18}$$

A von Neumann stability analysis considers data that can be represented by a Fourier mode. The solution w_j at the points x_j of the finite difference discretization can then be expressed as

$$w_j^n = e^{\mathrm{i}kx_j}, \quad w_{j+1}^n = e^{\mathrm{i}kx_j}e^{\mathrm{i}k\Delta x}, \quad w_{j-1}^n = e^{\mathrm{i}kx_j}e^{-\mathrm{i}k\Delta x} \tag{6.19}$$

with i being the imaginary unit and k being the wave number. The updated solution is given by

$$w_j^{n+1} = r(k, \Delta x, \Delta t) w_j^n$$

using the amplification factor $r(k, \Delta x, \Delta t)$. We insert the Fourier mode representation of the unknowns in (6.18) and divide by $e^{\mathrm{i}kx_j}$ to obtain

$$\begin{aligned} r(k, \Delta x, \Delta t)\,(1 \; &+ \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)\Big) \\ &= 1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right). \end{aligned}$$

The amplification factor can be expressed as

$$r(k, \Delta x, \Delta t) = \frac{1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)}{1 + \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)}.$$

The discretization is stable if the Fourier mode is not amplified, i.e. $|r(k, \Delta x, \Delta t)| \leq 1$. Therefore, we investigate the numerator

$$f(k, \Delta x, \Delta t) = 1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)$$

and the denominator

$$g(k, \Delta x, \Delta t) = 1 + \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)$$

with the amplification factor $r(k, \Delta x, \Delta t) \coloneqq \frac{f(k, \Delta x, \Delta t)}{g(k, \Delta x, \Delta t)}$. Evaluating $|r(k, \Delta x, \Delta t)|$ requires the absolute values of the complex numbers given by the numerator and denominator. In order to avoid handling the square root resulting from the complex number we analyze $|r(k, \Delta x, \Delta t)|^2$. First, we replace the Fourier mode by its representation in trigonometric form $e^{\mathrm{i}kx_j} = \cos(kx_j) + \mathrm{i}\sin(kx_j)$. Then, the numerator and denominator are given by

$$\begin{aligned} f(k, \Delta x, \Delta t) = &1 - \frac{\nu}{2}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12}\left(2\cos(k\Delta x) - 2\right) \end{aligned}$$

and

$$\begin{aligned} g(k, \Delta x, \Delta t) =& 1 + \frac{\nu}{2} \left(1 - \cos(k\Delta x) + \mathrm{i} \sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12} \left(2\cos(k\Delta x) - 2\right). \end{aligned}$$

Now, we introduce the following substitution

$$y := \cos\left(\frac{k\Delta x}{2}\right) \Rightarrow \cos(k\Delta x) = 2y - 1,\ \sin(k\Delta x) = \pm 2\sqrt{y(1-y)} \tag{6.20}$$

with $y \in [0, 1]$. It allows to write the numerator as

$$f(k, \Delta x, \Delta t) = 1 - \nu\left(1 - y \pm \mathrm{i}\sqrt{y(1-y)}\right) + \frac{\nu^2}{3}(y-1)$$

and the denominator as

$$g(k, \Delta x, \Delta t) = 1 + \nu\left(1 - y \pm \mathrm{i}\sqrt{y(1-y)}\right) + \frac{\nu^2}{3}(y-1).$$

Now, $|r(k, \Delta x, \Delta t)|^2 \leq 1$ is only fulfilled if $|f(k, \Delta x, \Delta t)|^2 \leq |g(k, \Delta x, \Delta t)|^2$. Thus, we check

$$\begin{aligned} &|f(k, \Delta x, \Delta t)|^2 \leq |g(k, \Delta x, \Delta t)|^2 \\ \Leftrightarrow &\left(1 - \nu(1-y) + \frac{\nu^2}{3}(y-1)\right)^2 \leq \left(1 + \nu(1-y) + \frac{\nu^2}{3}(y-1)\right)^2. \end{aligned}$$

After expanding and canceling terms it gives

$$-\nu(1-y) + \frac{1}{3}\nu^3(y-1)^2 \leq \nu(1-y) - \frac{1}{3}\nu^3(y-1)^2$$

and can be further simplified to

$$\frac{2}{3}\nu^3(1-y)^2 \leq 2\nu(1-y).$$

This obviously holds for $y = 1$. Under the assumption that $y \neq 1$ the inequality can be simplified to

$$\begin{aligned} \frac{2}{3}\nu^3(1-y)^2 &\leq 2\nu(1-y) \\ \Leftrightarrow \nu^2 &\leq \frac{3}{1-y} \end{aligned}$$

and therefore $\nu \leq \sqrt{\frac{3}{1-y}},\ y \in [0, 1)$, has to hold since we consider only positive values for ν. From this follows that the method is unstable for all $y \in [0, 1)$ if no restrictions are imposed on the CFL number ν. However, one would expect a stable method if a A-stable time integrator is used. In the appendix, see Sec. B.1.1, a more detailed description of the computations is presented.

Third order method

The von Neumann stability analysis of the third order method follows the same procedure as for the fourth order method in the previous section. We start with the formula of the third order method

$$w_h^{n+1} = w_h^n + \frac{\Delta t}{3}\left(\partial_t w^n + 2\partial_t w^{n+1}\right) - \frac{\Delta t^2}{6}\partial_t^2 w^n.$$

Again, the method is applied to the convection equation (6.14), where time derivatives are replaced by space derivatives (6.15), leading to

$$w^{n+1} = w^n - \frac{u\Delta t}{3}\left(\partial_x w^n + 2\partial_x w^{n+1}\right) - \frac{u^2\Delta t^2}{6}\partial_x^2 w^{n+1}. \tag{6.21}$$

The resulting terms are discretized using the finite differences (6.16) and the CFL number $\nu \coloneqq \frac{u\Delta t}{\Delta x}$ is introduced. The unknowns are reordered such that the unknowns at the new time level $n+1$ are on the left hand side while the unknowns of the old time level n are on the right hand side. This gives

$$\begin{aligned} w_j^{n+1} + \frac{\nu}{3}\left(2w_j^{n+1} - 2w_{j-1}^{n+1}\right) + \frac{\nu^2}{6}\left(w_{j-1}^{n+1} - 2w_j^{n+1} + w_{j+1}^{n+1}\right) \\ = w_j^n - \frac{\nu}{3}\left(w_j^n - w_{j-1}^n\right). \end{aligned}$$

The solution is expressed using the Fourier modes (6.19) and $e^{\mathrm{i}kx_j}$ is eliminated which leads to

$$\begin{aligned} r(k,\Delta x,\Delta t)\left(1 + \frac{\nu}{3}\left(2 - 2e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{6}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)\right) \\ = 1 - \frac{\nu}{3}\left(1 - e^{-\mathrm{i}k\Delta x}\right). \end{aligned}$$

We define the amplification factor again as $r(k,\Delta x,\Delta t) \coloneqq \frac{f(k,\Delta x,\Delta t)}{g(k,\Delta x,\Delta t)}$ with

$$\begin{aligned} f(k,\Delta x,\Delta t) =& 1 - \frac{\nu}{3}\left(1 - e^{-\mathrm{i}k\Delta x}\right) \\ g(k,\Delta x,\Delta t) =& 1 + \frac{\nu}{3}\left(2 - 2e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{6}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right). \end{aligned}$$

Using the trigonometric form of the complex numbers gives

$$f(k,\Delta x,\Delta t) = 1 - \frac{\nu}{3}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right)$$

and

$$g(k,\Delta x,\Delta t) = 1 + \frac{2\nu}{3}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) + \frac{\nu^2}{6}\left(2\cos(k\Delta x) - 2\right).$$

We use substitution (6.20) and obtain

$$f(k,\Delta x,\Delta t) = 1 - \frac{2\nu}{3}\left(1 - y \pm \mathrm{i}\sqrt{y(1-y)}\right)$$

for the numerator and

$$g(k, \Delta x, \Delta t) = 1 + \frac{4\nu}{3}\left(1 - y \pm \mathrm{i}\sqrt{y(1-y)}\right) + \frac{2\nu^2}{3}(1y - 1)$$

for the denominator. As in the previous section we check whether $|f(k, \Delta x, \Delta t)|^2 \leq |g(k, \Delta x, \Delta t)|^2$ holds. We obtain

$$\begin{aligned}&\left(1 - \frac{2\nu}{3}(1-y)\right)^2 + \left(\frac{2\nu}{3}\sqrt{y(1-y)}\right)^2 \\ \leq &\left(1 + \frac{4\nu}{3}(1-y) + \frac{2\nu^2}{3}(1y-1)\right)^2 + \left(\frac{4\nu}{3}\sqrt{y(1-y)}\right)^2\end{aligned}$$

which can be simplified to

$$0 \leq (1-y)\left(3 + \frac{1}{3}\nu^3(1-y) - \frac{4}{3}\nu^2(1-y)\right).$$

Now, the method is stable if the inequality holds for all $y \in [0, 1]$. For an implicit method, that is A-stable, we expect this inequality to hold for all ν as well. The special case $y = 1$ is simple as the right hand side vanishes. For $y \neq 0$, it is sufficient to check

$$0 \leq \underbrace{3 + \left(\frac{1}{3}\nu^3 - \frac{4}{3}\nu^2\right)(1-y)}_{=:h(\nu)}. \tag{6.22}$$

This inequality is not fulfilled for arbitrary choices of $\nu > 0$ and $y \in [0, 1)$. For $y = 0$, we have plotted $h(\nu)$ in Fig. 6.3 and one clearly sees that the function becomes negative approximately for $\nu \in (2.30278, 3)$. The bounds have been determined numerically and rounded to five decimal places. Thus, the method is subject to a CFL-like condition for which it is stable. A more detailed version of the analysis can be found in the appendix, see Sec. B.1.2.

6.5. A new stable approach

The previous approach to implicit multiderivative time integrators has been shown to be unfeasible due to stability issues. However, multiderivative schemes are still an interesting alternative to the more standard schemes due to the availability of time integrators with very few stages and high accuracy.

6.5.1. Devising a stable discretization

The standard approach to discretize a convection-diffusion equation using upwind finite differences for the convective term and central differences for the diffusion term leads to a discretization that is only conditionally stable. We show now that it is possible to generate a unconditionally stable discretization with only small changes.

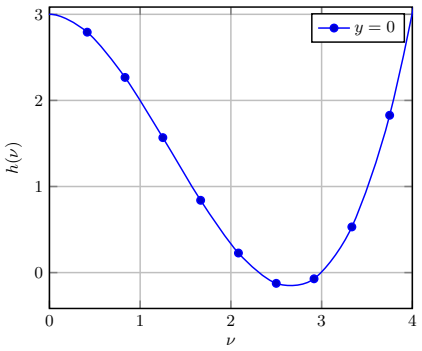


Figure 6.3.: Plot of the squared absolute value of the stability function $h(\nu)$ (6.22) for different CFL numbers ν over the domain $\nu \in [0, 4]$. The stability function is displayed for $y = 0$. The method is stable for this wave number provided $h(\nu) \geq 0$. In order to be stable for all wave numbers, we need $h(\nu) \geq 0$ for all $y \in [0, 1)$ where y is defined in (6.20).

The diffusion term stems from the convection term. Thus, we use the an upwind discretization for the second order term that has the same upwind discretization as the discretization of the convective term. More precisely, we use

$$\partial_x w(x_j) \approx \frac{w_j - w_{j-1}}{\Delta x}, \quad \partial_x^2 w(x_j) \approx \frac{w_j - 2w_{j-1} + w_{j-2}}{\Delta x^2}. \tag{6.23}$$

for the approximation of the space derivatives for $u > 0$. The same assumptions regarding the mesh, domain etc. are the same as for the analysis of the Cauchy-Kovalewski approach, see Sec. 6.4.

Fourth order method We follow the same procedure for the von Neumann stability analysis as earlier. After applying the fourth order integrator to the convection equation (6.14) an replacing time by space derivatives we get

$$w^{n+1} = w^n - \frac{u\Delta t}{2}\left(\partial_x w^n + \partial_x w^{n+1}\right) + \frac{u^2\Delta t^2}{12}\left(\partial_x^2 w^n - \partial_x^2 w^{n+1}\right). \tag{6.17 revisited}$$

Then, we discretize the derivatives using the finite differences (6.23) and use substitute $\nu := \frac{u\Delta t}{\Delta x} \geq 0$ to obtain

$$\begin{aligned} w^{n+1} = w^n &- \frac{\nu}{2}\left(w_j^n - w_{j-1}^n + w_j^{n+1} - w_{j-1}^{n+1}\right) \\ &+ \frac{\nu^2}{12}\left(w_j^n - 2w_{j-1}^n + w_{j-2}^n - w_j^{n+1} + 2w_{j-1}^{n+1} - w_{j-2}^{n+1}\right). \end{aligned}$$

We reorder the terms such that unknowns at the new time level $n+1$ are on the left hand side and the terms of the known time level n are on the right hand side

$$\begin{aligned} &w^{n+1} + \frac{\nu}{2}\left(w_j^{n+1} - w_{j-1}^{n+1}\right) + \frac{\nu^2}{12}\left(w_j^{n+1} - 2w_{j-1}^{n+1} + w_{j-2}^{n+1}\right) \\ =&w^n - \frac{\nu}{2}\left(w_j^n - w_{j-1}^n\right) + \frac{\nu^2}{12}\left(w_j^n - 2w_{j-1}^n + w_{j-2}^n\right). \end{aligned}$$

The solution is again represented by a Fourier mode (6.19). We obtain a new term $w_{j-2}^n = e^{\mathrm{i}kx_j}e^{-2\mathrm{i}k\Delta x}$ due to the upwind discretization of the second space derivative. After inserting the Fourier mode representation and dividing by $e^{\mathrm{i}kx_j}$ we get

$$\begin{aligned} &r(k,\Delta x,\Delta t)\left[1 + \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(1 - 2e^{-\mathrm{i}k\Delta x} + e^{-2\mathrm{i}k\Delta x}\right)\right] \\ =&1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(1 - 2e^{-\mathrm{i}k\Delta x} + e^{-2\mathrm{i}k\Delta x}\right) \end{aligned}$$

with amplification factor $r(k,\Delta x,\Delta t)$. Again, we define the amplification factor as $r(k,\Delta x,\Delta t) := \frac{f(k,\Delta x,\Delta t)}{g(k,\Delta x,\Delta t)}$. We have

$$\begin{aligned} f(k,\Delta x,\Delta t) =&1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(1 - 2e^{-\mathrm{i}k\Delta x} + e^{-2\mathrm{i}k\Delta x}\right) \\ g(k,\Delta x,\Delta t) =&1 + \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(1 - 2e^{-\mathrm{i}k\Delta x} + e^{-2\mathrm{i}k\Delta x}\right). \end{aligned}$$

Using the trigonometric form of the complex numbers gives

$$\begin{aligned} f(k,\Delta x,\Delta t) = 1 &- \frac{\nu}{2}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12}\left(1 - 2\cos(k\Delta x) + 2\mathrm{i}\sin(k\Delta x) + \cos(2k\Delta x) - \mathrm{i}\sin(2k\Delta x)\right) \end{aligned}$$

and

$$\begin{aligned} g(k,\Delta x,\Delta t) = 1 &+ \frac{\nu}{2}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12}\left(1 - 2\cos(k\Delta x) + 2\mathrm{i}\sin(k\Delta x) + \cos(2k\Delta x) - \mathrm{i}\sin(2k\Delta x)\right). \end{aligned}$$

For further simplification we use the double angle formulae

$$\begin{aligned} \sin(2\theta) &= 2\sin(\theta)\cos(\theta) \\ \cos(2\theta) &= \cos^2(\theta) - \sin^2(\theta) = 1 - 2\sin^2(\theta) = 2\cos^2(\theta) - 1 \end{aligned} \tag{6.24}$$

to express the numerator as

$$\begin{aligned} f(k,\Delta x,\Delta t) = 1 &- \frac{\nu}{2}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12}\left(1 - 2\cos(k\Delta x) + 2\mathrm{i}\sin(k\Delta x) + 2\cos^2(k\Delta x) - 1 - 2\mathrm{i}\sin(k\Delta x)\cos(k\Delta x)\right) \end{aligned}$$

and the denominator as

$$g(k,\Delta x,\Delta t) = 1 + \frac{\nu}{2}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right) + \frac{\nu^2}{12}\left(1-2\cos(k\Delta x)+2\mathrm{i}\sin(k\Delta x)+2\cos^2(k\Delta x)-1-2\mathrm{i}\sin(k\Delta x)\cos(k\Delta x)\right).$$

Again, we use the substitution (6.20) to express the terms as

$$f(k,\Delta x,\Delta t) = 1-\nu(1-y)\mp\nu\mathrm{i}\sqrt{y(1-y)} + \frac{\nu^2}{3}\left(1-3y+2y^2\pm\mathrm{i}\sqrt{y(1-y)}\mp\mathrm{i}(2y-1)\sqrt{y(1-y)}\right)$$

and

$$g(k,\Delta x,\Delta t) = 1+\nu(1-y)\pm\nu\mathrm{i}\sqrt{y(1-y)} + \frac{\nu^2}{3}\left(1-3y+2y^2\pm\mathrm{i}\sqrt{y(1-y)}\mp\mathrm{i}(2y-1)\sqrt{y(1-y)}\right).$$

The discretization is stable if $|r(k,\Delta x,\Delta t)|^2 \leq 1$ holds. Thus, we check if $|f(k,\Delta x,\Delta t)|^2 \leq |g(k,\Delta x,\Delta t)|^2$ holds. Hence, if

$$\left(1-\nu(1-y)+\frac{\nu^2}{3}\left(1-3y+2y^2\right)\right)^2 + y(1-y)\left(\mp\nu+\frac{\nu^2}{3}\left(\pm2\mp2y\right)\right)^2 \leq \left(1+\nu(1-y)+\frac{\nu^2}{3}(1-3y+2y^2)\right)^2 + y(1-y)\left(\pm\nu+\frac{\nu^2}{3}(\pm2\mp2y)\right)^2.$$

After bringing all terms to the right hand side and simplifying the expression we obtain

$$0 \leq 4\nu(1-y)+\frac{2\nu^3}{3}(1-y)^2.$$

This inequality is fulfilled for all y and ν as we require $\nu \geq 0$ and $y \in [0,1]$. It follows that the scheme is stable for all time step sizes. A more detailed version of the von Neumann stability analysis can be found in the appendix, see Sec. B.2.1.

Third order method The analysis for the third order method follows the same procedure as for the fourth order method. The method is applied to the convection equation (6.14) and the time derivatives are replaced by space derivatives. This gives

$$w^{n+1} = w^n - \frac{u\Delta t}{3}\left(\partial_x w^n + 2\partial_x w^{n+1}\right) - \frac{u^2\Delta t^2}{6}\partial_x^2 w^{n+1} \qquad \text{(6.21 revisited)}$$

and the derivatives are discretized using the finite differences (6.16) and $\nu \coloneqq \frac{u\Delta t}{\Delta x}$ is introduced to obtain

$$w_j^{n+1} = w_j^n - \frac{\nu}{3}\left(w_j^n - w_{j-1}^n + 2w_j^{n+1} - 2w_{j-1}^{n+1}\right) - \frac{\nu^2}{6}\left(w_j^{n+1} - 2w_{j-1}^{n+1} + w_{j-2}^{n+1}\right).$$

Again, we reorder the unknowns such that all unknowns at the new time level are on the left hand side, express the solution using Fourier modes (6.19) and eliminate $e^{\mathrm{i}kx_j}$. This gives

$$r(k,\Delta x,\Delta t)\left(1+\frac{2}{3}\nu\left(1-e^{-\mathrm{i}k\Delta x}\right)+\frac{\nu^2}{6}\left(1-2e^{-\mathrm{i}k\Delta x}+e^{-2\mathrm{i}k\Delta x}\right)\right) = 1-\frac{\nu}{3}\left(1-e^{-\mathrm{i}k\Delta x}\right).$$

Again, we define the amplification factor as $r(k,\Delta x,\Delta t) \coloneqq \frac{f(k,\Delta x,\Delta t)}{g(k,\Delta x,\Delta t)}$ with

$$\begin{aligned} f(k,\Delta x,\Delta t) =& 1-\frac{\nu}{3}\left(1-e^{-\mathrm{i}k\Delta x}\right) \\ g(k,\Delta x,\Delta t) =& 1+\frac{2\nu}{3}\left(1-e^{-\mathrm{i}k\Delta x}\right)+\frac{\nu^2}{6}\left(1-2e^{-\mathrm{i}k\Delta x}+e^{-2\mathrm{i}k\Delta x}\right) \end{aligned} \tag{6.25}$$

in this case. Using the trigonometric form of the complex numbers we obtain

$$f(k,\Delta x,\Delta t) = 1-\frac{\nu}{3}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right) \tag{6.26}$$

and

$$\begin{aligned} g(k,\Delta x,\Delta t) =& 1+\frac{2\nu}{3}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right) \\ &+\frac{\nu^2}{6}\left(1-2\cos(k\Delta x)+2\mathrm{i}\sin(k\Delta x)+\cos^2(k\Delta x)-1-2\mathrm{i}\sin(k\Delta x)\cos(k\Delta x)\right) \end{aligned} \tag{6.27}$$

and after using the substitution (6.20) one obtains

$$f(k,\Delta x,\Delta t) = 1-\frac{2\nu}{3}\left(1-y\pm\mathrm{i}\sqrt{y(1-y)}\right)$$

and

$$\begin{aligned} g(k,\Delta x,\Delta t) = 1&+\frac{4\nu}{3}\left(1-y\pm\mathrm{i}\sqrt{y(1-y)}\right) \\ &+\frac{2\nu^2}{3}\left(1-3y+2y^2\pm\mathrm{i}\sqrt{y(1-y)}\mp\mathrm{i}\sqrt{y(1-y)}(2y-1)\right). \end{aligned}$$

Now, we have to check whether $|f(k,\Delta x,\Delta t)|^2 \leq |g(k,\Delta x,\Delta t)|^2$ holds. We get

$$\begin{aligned} &\left(1-\frac{2\nu}{3}(1-y)\right)^2+\left(\frac{2\nu}{3}\sqrt{y(1-y)}\right)^2 \\ \leq&\left(1+\frac{4\nu}{3}(1-y)+\frac{2\nu^2}{3}\left(1-3y+2y^2\right)\right)^2 \\ &+y(1-y)\left(\pm\frac{4\nu}{3}+\frac{2\nu^2}{3}(\mp 2y\pm 2)\right)^2 \end{aligned}$$

which can be simplified to

$$\begin{aligned} 0 \leq & \frac{4}{3}\nu^2(1-y)^2 + \frac{4}{9}\nu^4(1-3y+2y^2)^2 \\ & + 4\nu(1-y) + \frac{4}{3}\nu^2(1-y)^2 + \frac{16}{9}\nu^3(1-y)^2 \\ & + \frac{4}{9}\nu^4 y(1-y)(2y-2)^2. \end{aligned}$$

This inequality is fulfilled for all y and ν as we require $\nu \geq 0$ and $y \in [0,1]$. It follows that the scheme is stable for all time step sizes. A more detailed version of the von Neumann stability analysis can be found in the appendix, see Sec. B.2.2.

6.5.2. Sketch of the new approach

In order to outline how a stable discretization can be obtained for arbitrary space discretizations we focus on $d = 1$ space dimensions in order to keep the notation simple. However, the idea directly extends to higher d as shown in [212]. The main idea is that for a linear PDE like a linear convection-diffusion equation, see Sec. 2.4,

$$\partial_t w + u\partial_x w - \varepsilon\partial_x^2 w = h, \quad \varepsilon > 0$$

one obtains a system of equations

$$\partial_t W = \mathrm{A}_{\mathrm{DG}} W + b_{\mathrm{DG}} \tag{6.28}$$

after discretizing the method in space with a 'standard' discontinuous Galerkin method that does not use any auxiliary unknowns. In our publication [212] a symmetric interior penalty method [10] has been used as space discretization, but it is also applicable to the hybridized discontinuous Galerkin method with minor modifications. The matrix A_{DG} stems from the space discretization of the space derivatives, the vector W contains the coefficients to express the solution (3.6) using the polynomial basis and the vector b_{DG} stems from the source term h.

The idea is to introduce auxiliary variables of the time derivative of the unknown instead of expressing the time derivative by space derivatives. For this we introduce the following indexing

$$\begin{aligned} W_1 &\coloneqq W \\ W_i &\coloneqq \partial_t W_{i-1},\ i > 1 \end{aligned} \tag{6.29}$$

where W_1 refers to the coefficients stemming from the unknown quantity w expressed by the polynomial basis. The coefficients $W_i,\ i > 1$ refer to the ith time derivative of the unknown w or their approximation, respectively.

This substitution allows to write the linear equation (6.28) as

$$\partial_t W_1 = \mathrm{A}_{\mathrm{DG}} W_1 + b_{\mathrm{DG}} \tag{6.30}$$

where we have done nothing but using our new nomenclature. Now, for a two-derivative method as the ones used in the previous section, see Sec. 6.2, the second time derivative is used. Diffentiating equation (6.30) with respect to time gives

$$\partial_t^2 W_1 = \mathrm{A}_{\mathrm{DG}} \partial_t W_1 + \partial_t b_{\mathrm{DG}}$$

where $\partial_t W_1$ can be expressed by (6.30) leading to

$$\partial_t^2 W_1 = \mathrm{A}_{\mathrm{DG}} \left(\mathrm{A}_{\mathrm{DG}} W_1 + b_{\mathrm{DG}} \right) + \partial_t b_{\mathrm{DG}}. \tag{6.31}$$

It is advantageous to express time derivatives by additional unknowns. In this case, we introduce the unknown

$$W_2 = \mathrm{A}_{\mathrm{DG}} W_1 + b_{\mathrm{DG}} \tag{6.32}$$

which allows us to express (6.31) compactly as

$$\begin{aligned} \partial_t^2 W_1 &= \mathrm{A}_{\mathrm{DG}} W_2 + \partial_t b_{\mathrm{DG}} \\ \Leftrightarrow \partial_t W_2 &= \mathrm{A}_{\mathrm{DG}} W_2 + \partial_t b_{\mathrm{DG}}. \end{aligned} \tag{6.33}$$

This approach can be extended straightforwardly to higher time derivatives. This also means that the second time derivative $\partial_t^2 W_1 = \partial_t W_2$ can be recovered from W_1 by differentiating the source term h to obtain $\partial_t b_{\mathrm{DG}}$.

The third time derivative would require the differentiation of (6.31) once again. As we have an explicit expression of the first time derivative (6.32) by W_2 and the second time derivative $W_3 = \partial_t W_2$ through (6.33) we can instead differentiate (6.33) to obtain

$$\begin{aligned} \partial_t^2 W_2 = \partial_t W_3 &= \mathrm{A}_{\mathrm{DG}} \partial_t W_2 + \partial_t^2 b_{\mathrm{DG}} \\ \Leftrightarrow \qquad \partial_t W_3 &= \mathrm{A}_{\mathrm{DG}} W_3 + \partial_t^2 b_{\mathrm{DG}}. \end{aligned}$$

Thus, an approximation to the third time derivative is available by differentiating the source term h once more to compute $\partial_t^2 b_{\mathrm{DG}}$.

The two-point collocation schemes with up to three time derivatives, as introduced in Sec. 6.2, then can be expressed as

$$\begin{aligned} \frac{W_1^{n+1} - W_1^n}{\Delta t} = \; & \mathrm{A}_{\mathrm{DG}} \left(\alpha_1 W_1^n - \beta_1 W_1^{n+1} \right) + \Delta t \mathrm{A}_{\mathrm{DG}} \left(\alpha_2 W_2^n - \beta_2 W_2^{n+1} \right) \\ & + \Delta t^2 \mathrm{A}_{\mathrm{DG}} \left(\alpha_3 W_3^n - \beta_3 W_3^{n+1} \right) \end{aligned} \tag{6.34}$$

using the additionally introduced auxiliary variables. Note that (6.34) presents the formulation without source term, i.e. $b_{\mathrm{DG}} = 0$, to allow for a compact notation.

In [212] we have shown that the new approach leads to an unconditionally stable time discretization. Moreover, an extensive number of numerical experiments in two space dimensions has been carried out. All investigated multiderivative integration methods have shown optimal order of convergence and usually the error is lower than for the DIRK methods presented in Sec. 4.3.

6.5.3. Application to hybridized discontinuous Galerkin methods

After the introduction of the new approach to discretize additional time derivatives we extend the approach to the hybridized discontinuous Galerkin method. This includes the application to DG discretizations like LDG that use an auxiliary unknown σ for space derivatives. The interior penalty DG method used in [212] does not use any auxiliary unknowns for space derivatives. Moreover, we present how to apply the approach to the HDG method.

We follow the approach of the previous section closely. In order to have all unknowns present, we consider the discretization of the 1D convection-diffusion equation

$$\partial_t w + u\partial_x w - \varepsilon\partial_x^2 w = h.$$

Using similar notation of the matrices as in the section about the static condensation process, see Sec. 3.3.2, we can write the resulting system of equations as

$$\begin{pmatrix} 0 \\ \mathrm{M}_\varphi \partial_t W_1 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} & \mathrm{A}_{\sigma\lambda} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} & \mathrm{A}_{w\lambda} \\ \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} & \mathrm{A}_{\lambda\lambda} \end{pmatrix} \begin{pmatrix} \Sigma_1 \\ W_1 \\ \Lambda_1 \end{pmatrix} + \begin{pmatrix} 0 \\ R_w \\ 0 \end{pmatrix} \tag{6.35}$$

which follows the notation we have introduced in (6.28). However, we obtain a larger matrix due to the auxiliary and hybrid unknown than in the previous section and the static condensation process has not yet been applied. Now, we can differentiate the equation with respect to time an obtain

$$\begin{pmatrix} 0 \\ \mathrm{M}_\varphi \partial_t^2 W_1 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} & \mathrm{A}_{\sigma\lambda} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} & \mathrm{A}_{w\lambda} \\ \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} & \mathrm{A}_{\lambda\lambda} \end{pmatrix} \begin{pmatrix} \partial_t\Sigma_1 \\ \partial_t W_1 \\ \partial_t\Lambda_1 \end{pmatrix} + \begin{pmatrix} 0 \\ \partial_t R_w \\ 0 \end{pmatrix}.$$

We use the substitution for time derivatives defined in (6.29) and apply it also for the auxiliary and hybrid unknown. This gives

$$\begin{pmatrix} 0 \\ \mathrm{M}_\varphi \partial_t W_2 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} & \mathrm{A}_{\sigma\lambda} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} & \mathrm{A}_{w\lambda} \\ \mathrm{A}_{\lambda\sigma} & \mathrm{A}_{\lambda w} & \mathrm{A}_{\lambda\lambda} \end{pmatrix} \begin{pmatrix} \Sigma_2 \\ W_2 \\ \Lambda_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \partial_t R_w \\ 0 \end{pmatrix}. \tag{6.36}$$

which is the HDG version of (6.33). For the definition of $\partial_t W_2$ the time derivatives of the auxiliary $\Sigma_2 = \partial_t\Sigma_1$ unknown, hybrid unknown $\Lambda_2 = \partial_t\Lambda_1$ and possible source terms $\partial_t R_w$ are needed. Thus, one also has to introduce additional auxiliary unknowns to approximate time derivatives that were not present in the initial system of equations (6.35). However, the globally coupled system is then, once again, only coupled in the hybrid unknown Λ_1 and Λ_2. The other unknowns W_1, W_2, Σ_1 and Σ_2 can be eliminated by the static condensation procedure, see Sec. 3.3.2.

All methods given in Tab. 6.1 are implemented. These are the same as in the paper [212]. In contrast to Sec. 6.2 this also includes methods of fifth and sixth order that use three time derivatives.

c	$A^{(1)}$	$\cdots$	$A^{(M)}$
	$(b^{(1)})^{\mathrm{T}}$	$\cdots$	$(b^{(M)})^{\mathrm{T}}$

(a) General shape of an extended Butcher tableau for multiderivative Runge-Kutta methods.

0	0	0	0	0	0	0
0.5	$\frac{101}{480}$	$\frac{8}{30}$	$\frac{55}{2400}$	$\frac{65}{4800}$	$-\frac{25}{600}$	$-\frac{25}{8000}$
1.0	$\frac{7}{30}$	$\frac{16}{30}$	$\frac{7}{30}$	$\frac{5}{300}$	0	$-\frac{5}{300}$
	$\frac{7}{30}$	$\frac{16}{30}$	$\frac{7}{30}$	$\frac{5}{300}$	0	$-\frac{5}{300}$

(b) Butcher tableau of the multiderivative Runge-Kutta method used in this work with $M = 2$.

Table 6.2.: General form of Butcher tableaux of multiderivative Runge-Kutta methods and coefficients of the two-derivative three-stage multiderivative Runge-Kutta method used in this work. The method is sixth order accurate in time.

Fully implicit multiderivative collocation methods In addition to the two-point collocation method, we implement the multiderivative collocation method presented in [212]. It follows a similar procedure as the standard Runge-Kutta method, see Sec. 4.3, where stage values

$$w_h^{n,i} = w_h^n + \sum_{m=1}^{M} \Delta t^m \sum_{j=1}^{s} a_{ij}^{(m)} \partial_t^m w_h^{n,j}, \quad i = 1, \dots, s, \tag{6.37a}$$

are approximated. After all stage values have been obtained the solution can be updated as

$$w_h^{n+1} = w_h^n + \sum_{m=1}^{M} \Delta t^m \sum_{i=1}^{s} b_i^{(m)} \partial_t^m w_h^{n,j}. \tag{6.37b}$$

The update formulae of Runge-Kutta methods are simply extended for M time derivatives of the unknown w_h instead of only using one. The coefficients can be noted in an extended Butcher tableau, see Tab. 6.2a. As an example, the coefficients of a method using three stages and two time derivatives, are given in Tab. 6.2b.

6.6. Numerical results

The new approach for discretizing implicit multiderivative time integrators is verified using the linear convection and convection-diffusion equation in one space dimension. As we did not show any results in the 1D case for any 'classical' time integrators introduced in Sec. 4, we first show results using different DIRK methods. Subsequently the results obtained by two-point collocation schemes and the fully implicit multiderivative collocation methods are presented. The solver used for that has been implemented in MATLAB.

In order to present the applicability of the approach in *two* space dimensions we present further results afterwards. They have been obtained by a symmetric interior penalty discontinuous Galerkin (SIP-DG) method. We have presented parts of the results in our publication [212]. This SIP-DG method is not hybridized. At the point of time this thesis has been

written the implementation of a HDG method using the new approach for multiderivative time integrators has been work in progress. Further details about the interior penalty DG method used can found in our publication and about interior penalty DG methods in [10, 66], for example. We follow the same order to present results as in the one dimensional case. The approach in two space dimensions is verified using the same PDEs as in one space dimension. For each PDE we first present results for DIRK methods since we haven't shown any results of the SIP-DG implementation before. Afterwards we present the results obtained by two-point collocation schemes and the fully implicit multiderivative collocation method. The solver used has been implemented in C++ based on Netgen and NGSolve using PETSc for solving the linear system.

6.6.1. Results in 1D

We present results using standard and multiderivative time integrators in one space dimension. The hybridized discontinuous Galerkin method has been used as space discretization.

Linear convection equation The linear convection equation

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

is solved on a domain $\Omega = [0, 1]$ in the time for $t_{\text{final}} = 1$ with constant velocity vector $u = 1$ and periodic boundary conditions. The initial condition is

$$w(0, x) = \sin(2\pi x)$$

such that the solution is given by $w(t, x) = \sin(2\pi x - ut)$. The initial time step size is $\Delta t = 0.5$ and mesh size is $\Delta x = 1.0$. Both are refined uniformly.

In Fig. 6.4, we present the results using DIRK methods of order $Q = 1$ to $Q = 4$ with polynomials of degree $P = Q - 1$. We see that all methods achieve the expected order of convergence which indicates the correctness of the implementation. Note that the scaling of the axes is chosen such that it is easily comparable with the numerical results of the multiderivative time integrators.

In Fig. 6.5, the results using the two-point collocation methods from third up to sixth order for different polynomial degrees are shown. One clearly sees that the order of covergence is either limited by the spatial resolution or the order of the time integrator. For all time integrators, the expected order of convergence in time is reached once the spatial resolution is high enough, i.e. $P = Q - 1$. Increasing the polynomial degree to $P = Q$ has no influence on the slope and thus no influence on the overall order of convergence of the method.

In Fig. 6.6, we present the results of the fully implicit multiderivative Runge-Kutta method for different polynomial degrees. Again, the correct order in time and space is recovered. If

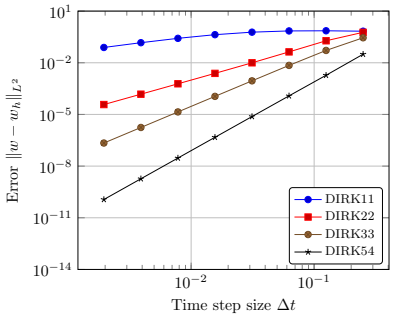


Figure 6.4.: Numerical results for the convection equation with parameters $u = 1$ and $t_{\text{final}} = 1.0$ using HDG for space discretization and DIRK schemes for time integration. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is CFL $= 0.5$.

the polynomial degree is high enough such that the overall error is not dominated anymore by the spatial discretization, i.e. $P = 5$, the errors are slightly lower than for the two-point collocation method of sixth order, see Fig. 6.5d.

All tested time integrators work as expected. In contrast to the Lax-Wendroff approach, see Sec. 6.2, no stability issues nor order degradation have been observed.

Linear convection-diffusion equation As second test case we consider the linear convection-diffusion equation

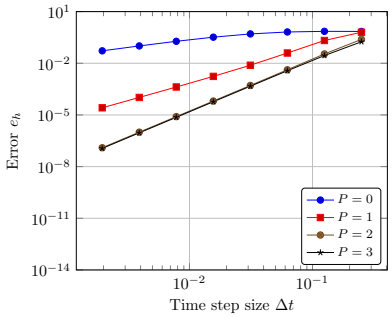
$$\partial_t w + u\partial_x w - \varepsilon\partial_x^2 w = h$$

on $\Omega = [0, 1]$ with periodic boundary conditions. The end time is $t_{\text{final}} = 1$, the convection velocity is $u = 1.0$ and the diffusion constant is $\varepsilon = 0.1$. In comparison to the pure convection equation in the previous section, the mixed formulation incorporating the auxiliary unknown σ is used, see Sec. 3.3. Thus, the new approach for the approximation of the time derivatives introduces additional unknowns for the time derivatives of σ (and their approximation). Moreover, we prescribe a source term h such that the solution is given by

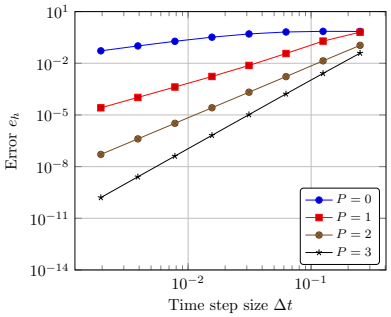
$$w(t, x) = e^{-t} \sin(2\pi(x - ut)).$$

The numerical experiments use the initial time step $\Delta t = 0.5$ and mesh size $\Delta x = 1.0$. Both are uniformly refined for the convergence study.

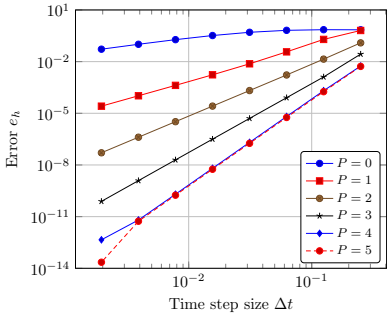
In Fig. 6.7 we present the results using DIRK methods of order $Q = 1$ to $Q = 4$ with polynomials of degree $P = Q - 1$. As for the convection case all methods achieve the expected order of convergence. This indicates the correctness of the implementation.



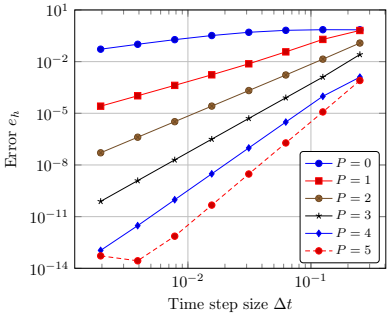
(a) Third order method ($k = 1, l = 2$).



(b) Fourth order method ($k = 2, l = 2$).



(c) Fifth order method ($k = 2, l = 3$).



(d) Sixth order method ($k = 3, l = 3$).

Figure 6.5.: Numerical results for the convection equation with parameters $u = 1$ and $t_{\text{final}} = 1.0$ using HDG for space discretization and two-point collocation schemes for time integration. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is CFL $= 0.5$.

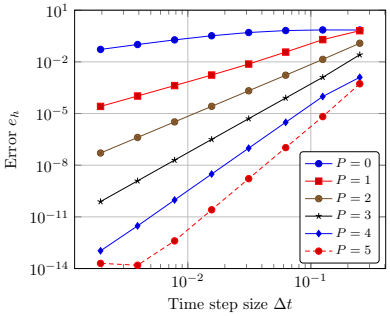


Figure 6.6.: Numerical results for the convection equation with parameters $u = 1$ and $t_{\text{final}} = 1.0$ using HDG for space discretization and a multiderivative Runge-Kutta method for time integration. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is $\text{CFL} = 0.5$.

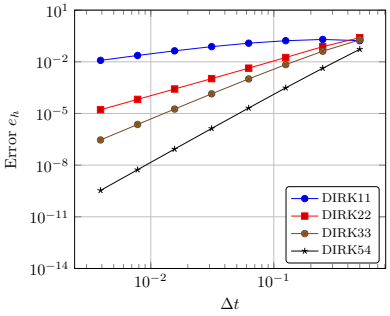


Figure 6.7.: Numerical results for the convection-diffusion equation with parameters $u = 1$, $t_{\text{final}} = 1.0$ and $\varepsilon = 0.1$ using HDG for space discretization and DIRK schemes for time integration. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is $\text{CFL} = 0.5$.

In Fig. 6.8, the results using the two-point collocation methods from third up to sixth order for different polynomial degrees are shown. All methods achieve the expected order of convergence in time once the spatial resolution is high enough. In contrast to the purely convective test case, the error is rather limited by the temporal error than the spatial error. For the third and fifth order methods, see Figs. 6.8a and 6.8c, we present also the results for $P = Q$. Besides the order of convergence being limited by the temporal order one barely sees any difference in the errors for $P = Q$ and $P = Q - 1$. The only obvious differences occur for the fifth order method for very small error levels where also the accuracy of the linear solver of MATLAB plays an important role.

In Fig. 6.9 we present the results of the fully implicit multiderivative Runge-Kutta method for different polynomial degrees. As for the two-point collocation methods the order of convergence is either limited by the spatial or temporal error and recovers sixth order for $P = 5$. The errors in the case $P \geq 4$ do not further decrease after a few refinements. We attribute this to the implementation of the method and the linear solver of MATLAB. The implementation does not assume that the multiderivative Runge-Kutta has an explicit first step, but allows for fully implicit multiderivative Runge-Kutta methods. Moreover, the local solvers are assembled for all elements at once, in order to allow for an easy and efficient implementation. However, without a reordering of the unknowns the local solves cannot be solved truly local. Instead, the arising system of equations is solved with MATLAB's linear solver (`mldivide`).

All tested time integrators work as expected for the linear convection-diffusion problem. Again, no stability issues nor order degradation have been observed.

6.6.2. Results in 2D using an interior penalty DG method

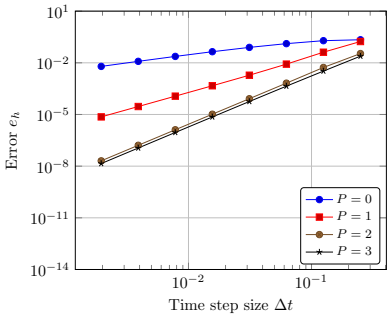
We present results using standard and multiderivative time integrators in two space dimensions. A symmetric interior penalty discontinuous Galerkin (SIP-DG) method has been used as space discretization. The results stem from our publication [212] that introduces the new approach for discretizing implicit multiderivative time integrators. Further information about this kind of discontinuous Galerkin methods can be found in our publication and the work of Arnold et al. [10], for example. If nothing else is mentioned, the linear system is solved using a GMRES solver with ILU(2) preconditioner until the relative residual drops below 10^{-10}.

Linear convection equation The linear convection equation

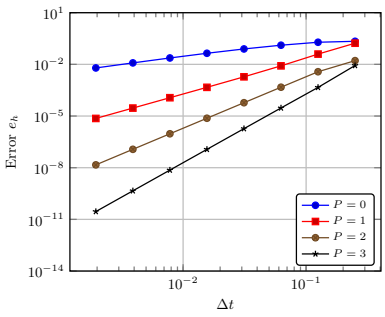
$$\partial_t w + \nabla \cdot (uw) = 0$$

is solved on a domain $\Omega = [0, 1]^2$ in the time for $t_{\text{final}} = 1$ with constant velocity vector $u = (1, 1)^{\mathrm{T}}$ and periodic boundary conditions. The initial condition

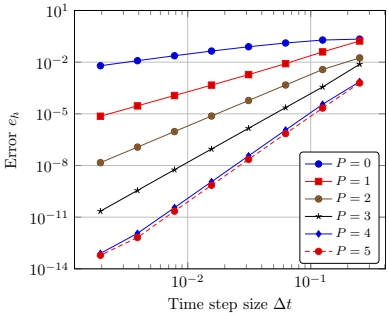
$$w(0, x) = \sin(2\pi x_1) \sin(2\pi x_2)$$



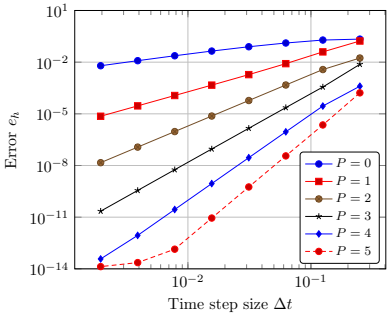
(a) Third order method ($k = 1, l = 2$).



(b) Fourth order method ($k = 2, l = 2$).



(c) Fifth order method ($k = 2, l = 3$).



(d) Sixth order method ($k = 3, l = 3$).

Figure 6.8.: Numerical results for the convection-diffusion equation with parameters $u = 1$, $t_{\text{final}} = 1.0$ and $\varepsilon = 0.1$ using HDG for space discretization and two-point collocation schemes for time integration. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is CFL $= 0.5$.

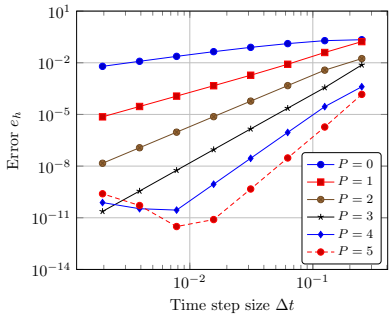


Figure 6.9.: Numerical results for the convection-diffusion equation with parameters $u = 1$, $t_{\text{final}} = 1.0$ and $\varepsilon = 0.1$ using HDG for space discretization and a multiderivative Runge-Kutta method for time integration. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is CFL $= 0.5$.

is used such that the solution is given by $w(t, x) = \sin(2\pi x_1 - u_1 t)\sin(2\pi x_2 - u_2 t)$. The initial time step size is $\Delta t = 0.25$ and the coarsest mesh consists of two triangular elements, i.e. $\Delta x = 1.0$. Both are refined uniformly.

In Fig. 6.10 we present the result using DIRK methods of order $Q = 1$ to $Q = 4$ with polynomials of degree $P = Q - 1$. We see that all methods achieve the expected order of convergence. This indicates the correctness of the implementation. Note that the scaling of the axes is chosen such that it is easily comparable with the numerical results of the multiderivative time integrators in two space dimensions.

In Fig. 6.11, the results using the two-point collocation methods from third up to sixth order for different polynomial degrees are shown. One clearly sees that the order of covergence is either limited by the spatial resolution or the order of the time integrator. For all time integrators, the expected order of convergence in time is reached once the spatial resolution is good enough, i.e. $P = Q - 1$. Increasing the polynomial degree to $P = Q$ has no influence on the slope and thus no influence on the overall order of convergence of the method. This results are also in good agreement with the results in one space dimension when the HDG method is used for space discretization, see Fig. 6.5.

In Fig. 6.12 we present the results of the fully implicit multiderivative Runge-Kutta method for different polynomial degrees. Again, the correct order in time and space is recovered. In contrast to the results in one space dimension, there is visible difference in the error between this time integrator and the two-point collocation scheme of sixth order, see Fig. 6.11d.

All tested time integrators work as expected. In contrast to the Lax-Wendroff approach,

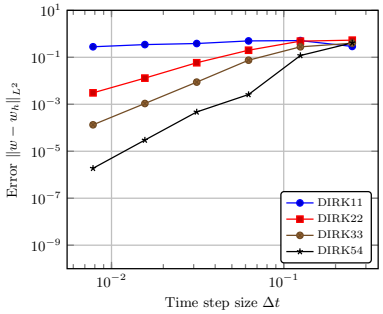


Figure 6.10.: Numerical results for the convection equation in two space dimensions with parameters $u = (1,1)^{\mathrm{T}}$, $t_{\mathrm{final}} = 1.0$. DIRK schemes have been used for time integration and a symmetric interior penalty discontinuous Galerkin method has been used for space discretization. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.25$. The CFL-number is $\mathrm{CFL} = \sqrt{2} \cdot 0.25 \approx 0.354$.

see Sec. 6.2, no stability issues nor order degradation have been observed. The results are in good agreement with the one obtained in one space dimension using the HDG method for space discretization.

Linear convection-diffusion equation As second test case we consider the linear convection-diffusion equation

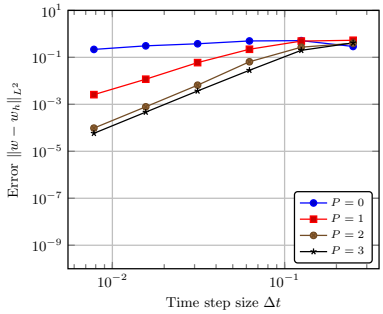
$$\partial_t w + \nabla \cdot (uw - \varepsilon \nabla w) = h$$

on $\Omega = [0,1]^2$ with periodic boundary conditions. The end time is $t_{\mathrm{final}} = 1$, the convection velocity is $u = (1,1)^{\mathrm{T}}$ and the diffusion constant is $\varepsilon = 0.1$. Moreover, we prescribe a source term h such that the solution is given by

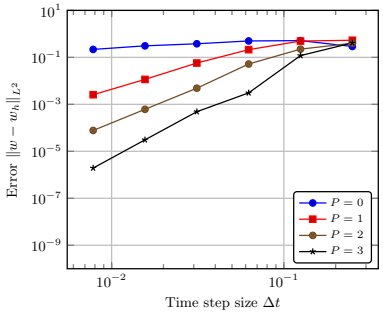
$$w(t,x) = e^{-t} \sin(2\pi(x_1 - u_1 t)) \sin(2\pi(x_2 - u_2 t)).$$

The numerical experiments use the initial time step $\Delta t = 0.5$ and mesh size $\Delta x = 1.0$. Both are uniformly refined for the convergence study. In comparison to the HDG discretization for this PDE in 1D the SIP-DG formulation does not require any auxiliary unknowns. However, the SIP-DG formulation is not meaningful for polynomials of degree $P = 0$. Thus, we present results only for $P > 0$.

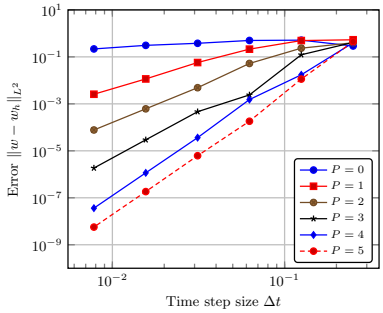
In Fig. 6.13, we present the result using DIRK methods of order $Q = 2$ to $Q = 4$ with polynomials of degree $P = Q - 1$. As for the convection case all methods achieve the expected order of convergence. This indicates the correctness of the implementation.



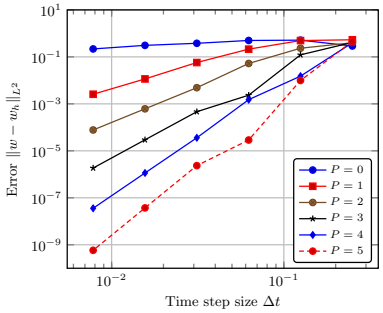
(a) Third order method ($k = 1, l = 2$).



(b) Fourth order method ($k = 2, l = 2$).



(c) Fifth order method ($k = 2, l = 3$).



(d) Sixth order method ($k = 3, l = 3$).

Figure 6.11.: Numerical results for the convection equation in two space dimensions with parameters $u = (1, 1)^{\mathrm{T}}$, $t_{\mathrm{final}} = 1.0$. Two-point collocation schemes have been used for time integration and a symmetric interior penalty discontinuous Galerkin method has been used for space discretization. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.25$. The CFL-number is CFL $= \sqrt{2} \cdot 0.25 \approx 0.354$.

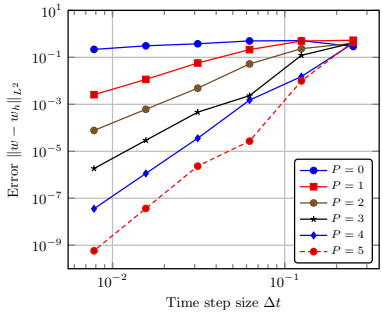


Figure 6.12.: Numerical results for the convection equation in two space dimensions with parameters $u = (1, 1)^{\mathrm{T}}$, $t_{\mathrm{final}} = 1.0$. Multiderivative Runge-Kutta method have been used for time integration and a symmetric interior penalty discontinuous Galerkin method has been used for space discretization. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.25$. The CFL-number is CFL $= \sqrt{2} \cdot 0.25 \approx 0.354$.

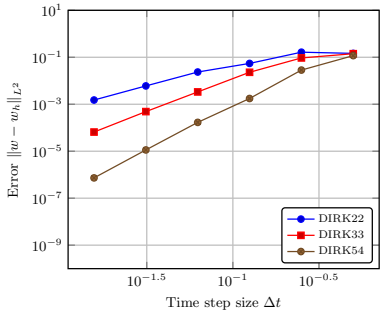
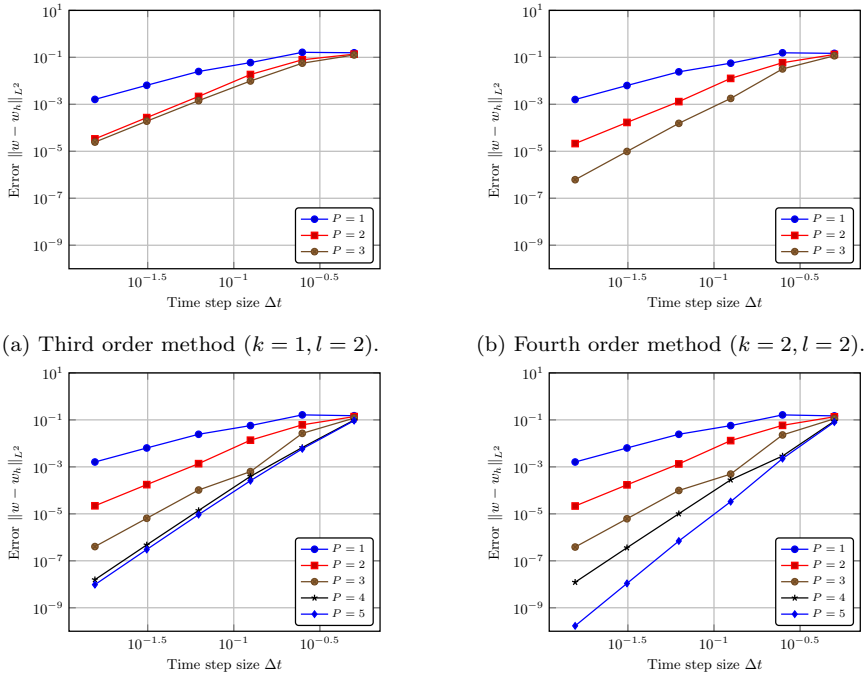


Figure 6.13.: Numerical results for the convection-diffusion equation in two space dimensions with parameters $t_{\mathrm{final}} = 1.0$ and $\varepsilon = 0.1$. DIRK schemes have been used for time integration and a symmetric interior penalty discontinuous Galerkin method has been used for space discretization. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is CFL $= \sqrt{2} \cdot 0.5 \approx 0.707$.



(a) Third order method ($k = 1, l = 2$).

(b) Fourth order method ($k = 2, l = 2$).

(c) Fifth order method ($k = 2, l = 3$).

(d) Sixth order method ($k = 3, l = 3$).

Figure 6.14.: Numerical results for the convection-diffusion equation in two space dimensions with parameters $u = 1$, $t_{\text{final}} = 1.0$ and $\varepsilon = 0.1$. Two-point collocation schemes have been used for time integration and a symmetric interior penalty discontinuous Galerkin method has been used for space discretization. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is CFL $= \sqrt{2} \cdot 0.5 \approx 0.707$.

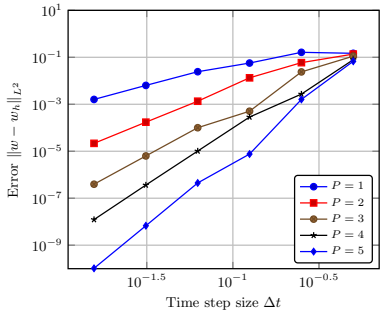


Figure 6.15.: Numerical results for the convection-diffusion equation in two space dimensions with parameters $u = 1$, $t_{\text{final}} = 1.0$ and $\varepsilon = 0.1$. A multiderivative Runge-Kutta method has been used for time integration and a symmetric interior penalty discontinuous Galerkin method has been used for space discretization. The time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. The CFL-number is $\text{CFL} = \sqrt{2} \cdot 0.5 \approx 0.707$.

In Fig. 6.14, the results using the two-point collocation methods from third up to sixth order for different polynomial degrees is shown. All methods achieve the expected order of convergence in time once the spatial resolution is high enough. For the third and fifth order methods, see Figs. 6.14a and 6.14c, we present also the results for $P = Q$. Besides the order of convergence being limited by the temporal order one barely sees any difference in the errors for $P = Q$ and $P = Q - 1$. For the simulations with $P = 5$ we observed that the linear solver might not converge. Thus, we use a direct solver. We observe this behavior also for the DIRK methods of all orders for $P = 5$. Thus, it's most likely related to the condition number of the linear system becoming increasingly worse for increasing polynomial degree P. This is not surprising, but also necessitates another solution strategy as direct solvers are usually not feasible. In our publication [212], we suggested that one could initialize the GMRES solver with the solution of a lower order method.

In Fig. 6.15, we present the results of the fully implicit multiderivative Runge-Kutta method for different polynomial degrees. As for the two-point collocation methods the order of convergence is either limited by the spatial or temporal error and recovers sixth order for $P = 5$. Again, we use a direct solver in order to solve the linear system for $P = 5$ instead of the GMRES method.

In contrast to the HDG implementation the sixth order time integrators show no deviation of the errors on the finest levels. The SIP-DG formulation does not depend on any local solves and thus is independent of the accuracy of these. Moreover, for the SIP-DG implementation

we present less refinements (six instead of eight in the one dimensional case). Thus, the errors for the most accurate methods are still greater than 10^{10} while in the one dimensional case the implementation reached much smaller error levels. We restrict ourselves to fewer refinements due to the increased computational work needed in two space dimensions which also increase the run-times of the implementation. All tested time integrators work as expected for the linear convection-diffusion problem. Again, no stability issues nor order degradation have been observed.

7. Efficient implementation of hybridized discontinuous Galerkin methods in MATLAB / GNU Octave

An important point in the development of numerical schemes is rapid prototyping. Suitable numerical libraries allow the quick implementation and testing of new ideas. Some popular libraries in the context of finite element methods are Netgen/NGSolve [204, 205], FEniCS [5, 147], deal.II [15] or Nektar++ [38], for example. These libraries provide highly optimized implementations that usually offer an interface to programming languages like C/C++ or scripting languages like python. In order to accompany such libraries by a tool that allows rapid prototyping in the context of discontinuous Galerkin methods with MATLAB / GNU Octave, FESTUNG (*F*inite *E*lement *S*imulation *T*oolbox for *UN*structured *G*rids) has been introduced in [80].

FESTUNG is an open source MATLAB [226] / GNU Octave [68] framework for the easy implementation of discontinuous Galerkin methods in two space dimensions. Besides focusing on the completeness of necessary tools and data structures for implementing DG methods easily, the framework focuses on an implementation with high computational efficiency and full compatibility with the open source software GNU Octave.

In [80], the following goals have been formulated that should be accomplished with FESTUNG:

1. "Design a general-purpose software package using the DG method for a range of standard applications" and provide this toolbox as a research and learning tool in the open source format (cf. [79])."

2. "Supply a well-documented, intuitive user-interface to ease adoption by a wider community of application and engineering professionals."

3. "Relying on the vectorization capabilities of MATLAB / GNU Octave optimize the computational performance of the toolbox components and demonstrate these software development strategies."

4. "Maintain throughout full compatibility with GNU Octave to support users of open source software."

The goals have been formulated to guide the development of a framework that fills a gap in the frameworks already available. A more detailed discussion of this can be found in the first FESTUNG paper [80].

The framework's source code is documented with Doxygen compatible comments. It is possible to generate the documentation of the code in various formats directly from its comments in the source code using Doxygen. Additionally, new releases are accompanied by publications that come with examples and detailed explanations of new features and changes.

The initial publication [80] introduced the FESTUNG framework and described the main goals and motivation of the project. Moreover, in this publication the basic concepts and implementations needed for a DG discretization like polynomials and handling of meshes are described. How to use the framework and its functions is explained by discretizing a time-dependent diffusion equation using the local discontinuous Galerkin method [56, 247], see Sec. 3.2. The implicit Euler method is used as time integrator. The second part of the paper series [195] deals with the discretization of a time-dependent convection equation using an upwind DG method with slope limiting. In this publication, explicit strong stability preserving (SSP) Runge-Kutta time integrators are used. The third part of the paper series [124] has been written during the preparation of this thesis. This work extends the framework and describes the implementation of a hybridized discontinuous Galerkin methods as used in this thesis including the introduction of DIRK methods for time integration. The implementation also uses a new program structure that will be described in the fourth part [194] that is currently in preparation. The third FESTUNG paper [124] is covered in this thesis.

We first introduce some additional notation that slightly deviates from the one introduced previously. This is done in order to stay aligned with the notation used in the FESTUNG framework. The notation also refers better to the implementation that heavily relies on vector and matrix operations in MATLAB / GNU Octave than the previously used notation but is harder to read. Afterwards, we briefly discuss the problem discretized and introduce the nomenclature for the arising terms as used in our publication. The assembly of matrices and vectors is shortly highlighted. The description of the assembly includes the implementation of the assembly in MATLAB / GNU Octave. This description focuses on a small choice of assembly routines in order to keep this section compact. An extensive description can be found in the paper series [80, 124, 195] and the documentation of the code [196]. We end the section with the discussion of some numerical results and compare the HDG method with an upwind DG discretization from a previous FESTUNG publication [195].

To the knowledge of the author of this thesis, no HDG implementation in MATLAB / GNU Octave has been available that fulfills all the goals formulated by the FESTUNG project. In [82], a MATLAB implementation of the HDG method in three space dimensions is discussed. It also makes extensive use of the vectorization in MATLAB / GNU Octave in order to allow for an efficient discretization. A comparison to other available MATLAB /

GNU Octave implementation can be found in the first part of the FESTUNG paper series [80].

7.1. Notation

The notation used in this chapter is a combination of the notation used in our publication [124] and the notation used in this thesis as introduced in Sec. 3. The notation in this chapter is refined as we use indices extensively to distinguish between and enumerate elements, faces, degrees of freedom and so on. Therefore, a slight deviation from the previously introduced notation cannot be avoided.

Recall that the domain Ω is partitioned into a mesh with K triangular elements T. The set $\mathcal{E}_h$ contains all edges $E_{\bar{k}}$ of the triangulation. The set of all edges can also be expressed as $\mathcal{E}_h = \mathcal{E}_h^{\text{in}} \cup \mathcal{E}_h^{\text{BC}}$ as it consists of all edges $E_{\bar{k}} \in \mathcal{E}_h^{\text{in}}$ that are at the interior of the domain and boundary edges $E_{\bar{k}} \in \mathcal{E}_h^{\text{BC}}$ that represent the domain boundary. Boundary conditions might be necessary on the boundary edges. Thus we split the set of boundary edges further into $\mathcal{E}_h^{\text{BC}} = \mathcal{E}_h^{\text{BC,in}} \cup \mathcal{E}_h^{\text{BC,out}}$ as it consists of the edges $E_{\bar{k}} \in \mathcal{E}_h^{\text{BC,in}}$ where inflow boundary conditions have to be prescribed and edges $E_{\bar{k}} \in \mathcal{E}_h^{\text{BC,out}}$ where outflow/no boundary conditions have to be prescribed.

On edge $E_{\bar{k}} \in \mathcal{E}_h$ a normal vector $n_{\bar{k}}$ (with unit length) is defined. It is chosen such that it points outwards of element $T_{\bar{k}-}$. The element can be chosen arbitrarily but stays fixed after choosing it. Boundary edges only have one adjoining element thus the vector $n_{\bar{k}}$ always points out of the domain boundary. From this definition it also follows that the element $T_{\bar{k}-}$ adjoining an edge always exists while the element $T_{\bar{k}+}$ only exists for edges $E_{\bar{k}} \in \mathcal{E}_h^{\text{in}}$. In order to reduce the number of indices, $T_{\bar{k}-}$ is referred to as $T_{\bar{k}}$ if no confusion with $T_{\bar{k}+}$ is possible.

In order to stay aligned with the implementation and the notation in the FESTUNG framework we denote the number of elements by K and the number of edges by $\overline{K}$. We refer to elements using indices k, and add bars $\bar{\cdot}$ (e.g., $\bar{k}$ or $\overline{K}$) whenever we refer to edges and edge related quantities. Additionally, we make use of the notation from the previous publications [80, 195] where we referred to edges of an element T_k as $E_{kn}, n \in \{1, 2, 3\}$. Note the difference between this element-local edge numbering ("E_{kn} is the nth edge of the kth element") and global edge numbering ("$E_{\bar{k}}$ is the $\bar{k}$th edge in the mesh").

For the description of the method we need mappings allowing us to switch back and forth between element-local and global indices. For that reason, we introduce a mapping $\rho(k, n)$ that relates the nth edge E_{kn} of element T_k to its global index in the set of edges $\mathcal{E}_h$,

$$\rho : \{1, \dots, K\} \times \{1, 2, 3\} \to \{1, \dots, \overline{K}\}, \quad (k, n) \mapsto \bar{k}\,. \tag{7.1}$$

This mapping is not injective since for each interior edge there exists a pair of index tuples

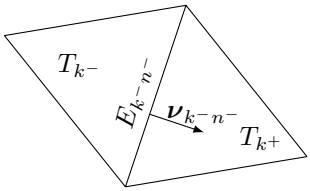


Figure 7.1.: Two triangles adjacent to edge $E_{\bar{k}}$. It holds $E_{\bar{k}} = E_{k^-n^-} = E_{k^+n^+}$ and $\nu_{k^-n^-} = -\nu_{k^+n^+}$.

(k^-, n^-) and (k^+, n^+) that map to the same edge index $\bar{k}$, i. e., $E_{\bar{k}} = E_{k^-n^-} = E_{k^+n^+}$ (see Fig. 7.1). We define a second mapping $\kappa(\bar{k}, l)$ to identify elements T_{k^-}, T_{k^+} adjacent to an edge $E_{\bar{k}}$. In this, argument $l \in \{1, 2\}$ denotes the edge-local index of the adjacent elements; it is constructed so that $l = 1$ refers to the "inner" element T_{k^-} (that always exists) while $l = 2$ refers to the "outer" element $T_{\bar{k}^+}$ that exists only for interior edges $E_{\bar{k}} \in \mathcal{E}_h^{\text{in}}$,

$$\kappa : \{1, \dots, \overline{K}\} \times \{1, 2\} \to \{0, \dots, K\}, \quad (\bar{k}, l) \mapsto \begin{Bmatrix} k^- \in \{1, \dots, K\}, & \text{if } l = 1 \\ k^+ \in \{0, \dots, K\}, & \text{if } l = 2 \end{Bmatrix}. \tag{7.2}$$

As element indices start counting from 1, we set $\kappa(\bar{k}, 2) = 0\,, \forall E_{\bar{k}} \in \mathcal{E}_h^{\text{BC}}$ to mark the absence of the "outer" element.

In order to refer to the implementation we use a monospace font. The function "assembleVecEdgePhiIntVal" is referred to as `assembleVecEdgePhiIntVal`, for example. Functions/syntax that stems form MATLAB / GNU Octave use the monospace font and are colored as in the respective MATLAB / GNU Octave editor such as `for` when indicating a `for`-loop.

7.2. Discretization

Let $\Omega_T := (0, t_{\text{end}}] \times \Omega$ be the time-space domain as introduced in Sec. 2.2. Then, we solve the linear convection equation

$$\partial_t w + \nabla \cdot f_c(w) = h(w), \quad \text{on} \quad (t, x) \in \Omega_T$$

with a source term in the flux formulation as introduced in Sec. 2.2. The convective flux is given as

$$f_c(w) = uw$$

with known velocity u. For further information, see Sec. 2.4. The problem is equipped with suitable initial and boundary data. As before, all functions and the unknown may depend on time t and space x coordinates, but we do not state this explicitly to maintain better readability.

The problem considered is the same as in the second part of the FESTUNG paper series [195].

7.2.1. Semi-discrete form

For discretizing the PDE in space we follow the procedure as described for the HDG method in Sec. 3.3. Then, we obtain the following weak formulation

$$\int_{T_k} \partial_t w_h \, \varphi_h \mathrm{d}x - \int_{T_k} f_c \cdot \nabla \varphi_h \mathrm{d}x + \int_{\partial T_k} \widehat{f}_c \cdot n_{T_k} \, \varphi_h \mathrm{d}s \, = \, \int_{T_k} h \, \varphi_h \, \mathrm{d}x \,, \tag{7.3a}$$

on a *single* element. On a *single* edge we obtain

$$\int_{E_{\bar{k}}} \mu_h \left\{ \begin{array}{cc} \alpha \left(2\lambda_h - w_h^- - w_h^+ \right) & \text{on } E_{\bar{k}} \in \mathcal{E}_h^{\text{in}} \\ \lambda_h - w_{\partial\Omega}(w_h^-) & \text{on } E_{\bar{k}} \in \mathcal{E}_h^{\text{BC}} \end{array} \right\} \mathrm{d}s \, = \, 0. \tag{7.3b}$$

As before one has to find solutions $(w_h, \lambda_h) \in V_h \times M_h$ that fulfill the equations for all test functions $(\varphi_h, \mu_h) \in V_h \times M_h$. This slightly differs from the description in Sec. 3.3 where we directly presented the global formulation. We choose the local description in order to properly describe the assembly process of the local matrices. The global formulation (3.15) is then given again by summing over all elements and edges.

The term $w_{\partial\Omega}$ denotes the value of the unknown on the domain boundary. In this chapter we discuss only a linear convection equation. Thus, we have only outflow and inflow boundary conditions and $w_{\partial\Omega}$ is given by

$$w_{\partial\Omega}(w_h^-) = \begin{cases} w_h^-, & \text{on outflow boundary } \partial\Omega_{\text{out}} \,, \\ w_{\text{D}}, & \text{on inflow boundary } \partial\Omega_{\text{in}} \,. \end{cases}$$

On inflow boundaries the known value w_{D} is prescribed and on outflow boundaries no values needs to be prescribed, see also Ch. 2. The local discrete solutions w_h on $T_k \in \mathcal{T}_h$ and λ_h on $E_{\bar{k}} \in \mathcal{E}_h$ are represented using local basis functions on elements and edges

$$w_h(t,x)\big|_{T_k} = \sum_{j=1}^{N} W_{kj} \, \varphi_{kj}(x) \,, \qquad \lambda_h(t,x)\big|_{E_{\bar{k}}} = \sum_{j=1}^{\overline{N}} \Lambda_{\bar{k}j} \, \mu_{\bar{k}j}(x) \,.$$

7.2.2. System of equations

The equation (7.3a) is tested with $\varphi_h = \varphi_{ki}, i = 1, \dots, N$ yielding a time-dependent system of equations. The element T_k contributes

$$\begin{aligned}
&\underbrace{\partial_t \sum_{j=1}^{K} W_{kj}(t) \int_{T_k} \varphi_{kj}\, \varphi_{ki}\, \mathrm{d}x}_{I\ (\mathrm{M}_\varphi)} - \underbrace{\sum_{j=1}^{K} W_{kj}(t) \sum_{m=1}^{2} \int_{T_k} u_m(t,x)\, \varphi_{kj}\, \partial_{x_m} \varphi_{ki}\, \mathrm{d}x}_{II\ (-\mathrm{G}^1 \text{ and } -\mathrm{G}^2)} \\
&+ \underbrace{\sum_{j=1}^{\overline{K}} \Lambda_{kj} \int_{\partial T_k \setminus \partial\Omega} (u(t,x)\cdot n)\ \mu_{\bar{k}j}\, \varphi_{ki}\, \mathrm{d}s}_{III\ (\mathrm{S})} \\
&- \underbrace{\alpha \sum_{j=1}^{\overline{K}} \Lambda_{kj} \int_{\partial T_k \setminus \partial\Omega} \mu_{\bar{k}j}\, \varphi_{ki}\, \mathrm{d}s}_{IV\ (\alpha \mathrm{R}_\mu)} + \underbrace{\alpha \sum_{j=1}^{K} W_{kj} \int_{\partial T_k \setminus \partial\Omega} \varphi_{kj}\, \varphi_{ki}\, \mathrm{d}s}_{V\ (\alpha \mathrm{R}_\varphi)} \\
&+ \underbrace{\int_{\partial T_k \cap \partial\Omega} \varphi_{ki}\, (u(t,x)\cdot n) \begin{cases} \sum_{j=1}^{\overline{K}} \Lambda_{kj}\, \mu_{\bar{k}j} & \text{on } \partial\Omega_{\mathrm{out}} \\ w_{\mathrm{D}}(t,x) & \text{on } \partial\Omega_{\mathrm{in}} \end{cases} \mathrm{d}s}_{VI\ (\mathrm{S}_{\mathrm{out}} \text{ and } F_{\varphi,D})} = \underbrace{\int_{T_k} h\, \varphi_{ki}\, \mathrm{d}x}_{VII\ (H)}
\end{aligned} \tag{7.4a}$$

to the system of equations. In the same manner (7.3b) is tested against $\mu_h = \mu_{\bar{k}i}, i = 1, \dots, \overline{N}$ such that the semi-discrete equation is given by

$$\begin{aligned}
&\underbrace{\alpha \sum_{j=1}^{\overline{K}} \Lambda_{kj} \int_{\partial T_k \setminus \partial\Omega} \mu_{\bar{k}j}\, \mu_{\bar{k}i}\, \mathrm{d}s}_{VIII\ (\alpha \bar{\mathrm{M}}_\mu)} - \underbrace{\alpha \sum_{j=1}^{K} W_{kj} \int_{\partial T_k \setminus \partial\Omega} \varphi_{kj}\, \mu_{\bar{k}i}\, \mathrm{d}s}_{IX\ (-\alpha \mathrm{T})} \\
&+ \underbrace{\sum_{j=1}^{\overline{K}} \Lambda_{kj} \int_{\partial T_k \cap \partial\Omega} \mu_{\bar{k}j}\, \mu_{\bar{k}i}\, \mathrm{d}s}_{X\ (\tilde{\mathrm{M}}_\mu)} - \underbrace{\int_{\partial T_k \cap \partial\Omega} \mu_{\bar{k}i} \begin{cases} \sum_{j=1}^{K} W_{kj}\, \varphi_{kj} & \text{on } \partial\Omega_{\mathrm{out}} \\ w_{\mathrm{D}}(t,x) & \text{on } \partial\Omega_{\mathrm{in}} \end{cases} \mathrm{d}s}_{XI\ (-\mathrm{K}_{\mu,\mathrm{out}} \text{ and } K_{\mu,D})} = 0.
\end{aligned} \tag{7.4b}$$

We have enumerated the different terms using Roman numerals and named the resulting matrices and vectors following the naming scheme of the previous FESTUNG publications [80, 195].

The global system of equations of (7.4) in matrix form is then given by

$$\mathrm{M}_\varphi \partial_t W + \underbrace{\left(-\mathrm{G}^1 - \mathrm{G}^2 + \alpha \mathrm{R}_\varphi\right)}_{=:\bar{\mathrm{L}}} W + \underbrace{\left(\mathrm{S} + \mathrm{S}_{\mathrm{out}} - \alpha \mathrm{R}_\mu\right)}_{=:\bar{\mathrm{M}}} \Lambda = \underbrace{H - F_{\varphi,D}}_{=:B_\varphi}, \tag{7.5a}$$

$$\underbrace{\left(-\alpha \mathrm{T} - \mathrm{K}_{\mu,\mathrm{out}}\right)}_{=:N} W + \underbrace{\left(\alpha \bar{\mathrm{M}}_\mu + \tilde{\mathrm{M}}_\mu\right)}_{=:P} \Lambda = \underbrace{-K_{\mu,D}(t)}_{=:B_\mu(t)} \tag{7.5b}$$

with representation vectors

$$W(t) \coloneqq \Big[W_{11}(t)\cdots W_{1N}(t)\cdots\cdots W_{K1}(t)\cdots W_{KN}(t)\Big]^{\mathrm{T}}$$
$$\Lambda(t) \coloneqq \Big[\Lambda_{11}(t)\cdots \Lambda_{1\overline{N}}(t)\cdots\cdots \Lambda_{\overline{K}1}(t)\cdots \Lambda_{\overline{K}\overline{N}}(t)\Big]^{\mathrm{T}}.$$

Contributions from volume terms I, II and VII

All matrices in system (7.5) have sparse block structure, and we define the non-zero entries in the remainder of this section. For all remaining entries we assume zero fill-in.

The integral of term I gives the standard mass matrix $\mathrm{M}_\varphi \in \mathbb{R}^{KN\times KN}$ with components defined as

$$[\mathrm{M}_\varphi]_{(k-1)N+i,(k-1)N+j} = \int_{T_k} \varphi_{kj}\,\varphi_{ki}\,\mathrm{d}x\,.$$

This leads to a block diagonal matrix because basis and test functions φ_{ki}, $i \in \{1,\dots,N\}$ are only supported on element T_k. Therefore

$$\mathrm{M}_\varphi = \begin{bmatrix} \mathrm{M}_{\varphi,T_1} & & \\ & \ddots & \\ & & \mathrm{M}_{\varphi,T_N} \end{bmatrix}$$

where M_{φ,T_k} is the *local* mass matrix

$$\mathrm{M}_{\varphi,T_k} = \int_{T_k} \begin{bmatrix} \varphi_{k1}\varphi_{k1} & \cdots & \varphi_{k1}\varphi_{kN} \\ \vdots & \ddots & \vdots \\ \varphi_{kN}\varphi_{k1} & \cdots & \varphi_{kN}\varphi_{kN} \end{bmatrix} \mathrm{d}x \in \mathbb{R}^{N\times N}\,.$$

We abbreviate $\mathrm{M}_\varphi = \mathrm{diag}(\mathrm{M}_{\varphi,T_1},\dots,\mathrm{M}_{\varphi,T_N})$.

The block matrices $\mathrm{G}^m \in \mathbb{R}^{KN\times KN}$, $m \in \{1,2\}$ are given component-wise by

$$[\mathrm{G}^m]_{(k-1)N+i,(k-1)N+j} = \int_{T_k} u_m\,\varphi_{kj}\,\partial_{x_m}\varphi_{ki}\,\mathrm{d}x \tag{7.7a}$$

again leading to a block-diagonal matrix $\mathrm{G}^m = \mathrm{diag}(\mathrm{G}^m_{T_1},\dots\mathrm{G}^m_{T_K})$, where each block is given by

$$\mathrm{G}^m_{T_K} = \int_{T_k} u_m \begin{bmatrix} \varphi_{k1}\partial_{x_m}\varphi_{k1} & \cdots & \varphi_{kN}\,\partial_{x_m}\varphi_{k1} \\ \vdots & \ddots & \vdots \\ \varphi_{k1}\,\partial_{x_m}\varphi_{kN} & \cdots & \varphi_{kN}\,\partial_{x_m}\varphi_{kN} \end{bmatrix} \mathrm{d}x\,. \tag{7.7b}$$

The source term enters the discretization as an additional term $H \in \mathbb{R}^{KN}$ on the right-hand side of the equation. The entries are given as

$$[H]_{(k-1)N+i} = \int_{T_k} h\,\varphi_{ki}\,\mathrm{d}x$$

such that the full vector is easily assembled as

$$H = \begin{bmatrix} H_{T_1} \\ \vdots \\ H_{T_K} \end{bmatrix} \qquad \text{with} \quad H_{T_k} = \int_{T_k} h \begin{bmatrix} \varphi_{k1} \\ \vdots \\ \varphi_{kN} \end{bmatrix} \mathrm{d}x.$$

Contribution of edge terms III, IV, V, VI— first equation

Compared to the DG discretization used in the previous FESTUNG publications [80, 195], the number of edge integrals has increased significantly. This is caused by the following factors:

1. Edge integrals are split into integrals over interior edges and over edges on the domain boundaries.

2. The numerical flux function (3.17) contains three terms compared to only one for the upwind flux used in [195].

3. An additional unknown is introduced that is only defined on edges resulting in an additional equation (7.3b).

To improve readability, we split the presentation of edge integrals into two sections: first, edge terms in the original equation for w_h (7.3a) and then the edge terms in the hybrid equation (7.3b). Throughout the assembly description and within the implementation we use the *element-based view*, i.e., all edge terms are presented in a form allowing for the assembly as nested loops over elements T_k, $k \in \{1, \dots, K\}$ and then edges E_{kn}, $n \in \{1, 2, 3\}$ of each element. This is different from the *edge based view* which would allow to assemble the terms in a single loop over all edges $E_{\bar{k}}$, $\bar{k} \in \{1, \dots, \overline{K}\}$. We make this choice since the data structures in FESTUNG favor the element-based view. For that reason, from now on, we always consider $\mu_{knj} = \mu_{\bar{k}j}$ using the mapping $\rho : (k, n) \mapsto \bar{k}$ specified in (7.1).

Term III contributes to matrix $\mathrm{S} \in \mathbb{R}^{KN \times \overline{K}\,\overline{N}}$ as

$$[\mathrm{S}]_{(k-1)N+i,(\bar{k}-1)\overline{N}+j} = \sum_{E_{kn} \in \partial T_k \cap \mathcal{E}_h^{\mathrm{in}}} \int_{E_{kn}} (u \cdot n_{kn})\, \mu_{knj}\, \varphi_{ki}\, \mathrm{d}s\,. \tag{7.9a}$$

The entries are structured into $N \times \overline{N}$-blocks with contributions from each edge on every element given as

$$\mathrm{S}_{E_{kn}} = \int_{E_{kn}} (u \cdot n_{kn}) \begin{bmatrix} \mu_{kn1}\, \varphi_{k1} & \cdots & \mu_{kn\overline{N}}\, \varphi_{k1} \\ \vdots & \ddots & \vdots \\ \mu_{kn1}\, \varphi_{kN} & \cdots & \mu_{kn\overline{N}}\, \varphi_{kN} \end{bmatrix} \mathrm{d}s\,. \tag{7.9b}$$

Note that this is the contribution of a single interior edge $E_{kn} \in \partial T_k \cap \mathcal{E}_h^{\text{in}}$ of triangle T_k. As this integral is also evaluated on the neighboring element, this block will produce *two* contributions to matrix S.

The matrix from term IV is a block matrix $\mathrm{R}_\mu \in \mathbb{R}^{KN \times \overline{K}\overline{N}}$ with blocks of size $N \times \overline{N}$ given by

$$[\mathrm{R}_\mu]_{(k-1)N+i,(\bar{k}-1)\overline{N}+j} = \sum_{E_{kn} \in \partial T_k \cap \mathcal{E}_h^{\text{in}}} \int_{E_{kn}} \mu_{knj}\, \varphi_{ki}\, \mathrm{d}s$$

with $i \in \{1, \dots, N\}$ and $j \in \{1, \dots, \overline{N}\}$. The local matrix of a single edge reads

$$\mathrm{R}_{\mu,E_{kn}} = \int_{E_{kn}} \begin{bmatrix} \mu_{kn1}\, \varphi_{k1} & \dots & \mu_{kn\overline{N}}\, \varphi_{k1} \\ \vdots & \ddots & \vdots \\ \mu_{kn1}\, \varphi_{kN} & \dots & \mu_{kn\overline{N}}\, \varphi_{kN} \end{bmatrix} \mathrm{d}s\,. \tag{7.10}$$

Term V gives another block diagonal contribution $\mathrm{R}_\varphi \in \mathbb{R}^{KN \times KN}$, where each entry is given by

$$[\mathrm{R}_\varphi]_{(k-1)N+i,(k-1)N+j} = \sum_{E_{kn} \in \partial T_k \cap \mathcal{E}_h^{\text{in}}} \int_{E_{kn}} \varphi_{kj}\, \varphi_{ki}\, \mathrm{d}s.$$

The element-local matrix is then

$$\mathrm{R}_{\varphi,T_k} = \sum_{E_{kn} \in \partial T_k \cap \mathcal{E}_h^{\text{in}}} \int_{E_{kn}} \begin{bmatrix} \varphi_{k1}\, \varphi_{k1} & \dots & \varphi_{kN}\, \varphi_{k1} \\ \vdots & \ddots & \vdots \\ \varphi_{k1}\, \varphi_{kN} & \dots & \varphi_{kN}\, \varphi_{kN} \end{bmatrix} \mathrm{d}s\,,$$

where each edge E_{kn} of each triangle T_k is visited exactly once, and $\mathrm{R}_\varphi = \mathrm{diag}(\mathrm{R}_{\varphi,T_1}, \dots, \mathrm{R}_{\varphi,T_K})$.

Term VI incorporates the boundary conditions on boundary edges $E_{\bar{k}} \in \mathcal{E}_h^{\text{BC}}$. In the case of an inflow boundary condition, this contributes to the right-hand side. Each entry of the vector $F_{\varphi,D} \in \mathbb{R}^{KN}$ is given as

$$[F_{\varphi,D}]_{(k-1)N+i} = \sum_{E_{kn} \in \partial T_k \cap \mathcal{E}_h^{\text{BC,in}}} \int_{E_{kn}} (u \cdot n_{kn})\, w_{\text{D}}\, \varphi_{ki}\, \mathrm{d}s\,.$$

Outflow boundary conditions depend on λ_h and therefore on the solution. This gives us an additional contribution $\mathrm{S}_{\text{out}} \in \mathbb{R}^{KN \times \overline{K}\overline{N}}$ to the left hand side, where each entry is given by

$$[\mathrm{S}_{\text{out}}]_{(k-1)N+i,(\bar{k}-1)\overline{N}+j} = \sum_{E_{kn} \in \partial T_k \cap \mathcal{E}_h^{\text{BC,out}}} \int_{E_{kn}} (u \cdot n_{kn})\, \mu_{knj}\, \varphi_{ki}\, \mathrm{d}s\,.$$

This is almost identical to (7.9a) with the only difference being the set of edges considered, and thus the sub-blocks take the same form as in equation (7.9b). In the implementation, S and S_{out} are assembled together.

Contribution of edge terms VIII, IX, X, and XI — hybrid equation

The first term — term VIII — is very similar to an edge mass matrix with the only differences being the parameter α that has to be respected and the fact that every edge is visited twice because it is an integral over interior edges $E_{\bar{k}} \in \mathcal{E}_h^{\text{in}}$. The matrix $\bar{\mathrm{M}}_\mu \in \mathbb{R}^{\overline{K}\overline{N}\times\overline{K}\overline{N}}$ is given by

$$[\bar{\mathrm{M}}_\mu]_{(\bar{k}-1)\overline{N}+i,(\bar{k}-1)\overline{N}+j} = \sum_{E_{kn}\in\partial T_k\cap\mathcal{E}_h^{\text{in}}} \int_{E_{kn}} \mu_{knj}\,\mu_{kni}\,\mathrm{d}s \tag{7.12}$$

for $i,j \in \{1,2,\dots,\overline{N}\}$ and $\bar{k} \in \{1,2,\dots,\overline{K}\}$. This leads to a block diagonal matrix because the ansatz and test functions $\mu_{kni} = \mu_{\bar{k}i}, i \in \{1,2,\dots,\overline{N}\}$ have support only on the corresponding edge $E_{kn} = E_{\bar{k}}$. Term IX is very similar to term IV, where each entry of the resulting matrix $\mathrm{T} \in \mathbb{R}^{\overline{K}\overline{N}\times KN}$ is given as

$$[\mathrm{T}]_{(\bar{k}-1)\overline{N}+i,(k-1)N+j} = \sum_{E_{kn}\in\partial T_k\cap\mathcal{E}_h^{\text{in}}} \int_{E_{kn}} \varphi_{kj}\,\mu_{kni}\,\mathrm{d}s \tag{7.13}$$

with $i \in \{1,\dots,\overline{N}\}$, $j \in \{1,\dots,N\}$, and $\bar{k}$ given by the mapping in (7.1). The contribution of a single edge is

$$\mathrm{T}_{E_{kn}} = \int_{E_{kn}} \begin{bmatrix} \varphi_{k1}\,\mu_{kn1} & \dots & \varphi_{kN}\,\mu_{kn1} \\ \vdots & \ddots & \vdots \\ \varphi_{k1}\,\mu_{kn1} & \dots & \varphi_{kN}\,\mu_{kn\overline{N}} \end{bmatrix} \mathrm{d}s. \tag{7.14}$$

In fact, we have $\mathrm{T} = \mathrm{R}_\mu^{\mathrm{T}}$ from (7.10) such that it does not have to be assembled separately. Term X gives us the edge mass matrix on boundary edges

$$[\tilde{\mathrm{M}}_\mu]_{(\bar{k}-1)\overline{N}+i,(\bar{k}-1)\overline{N}+j} = \sum_{E_{kn}\in\partial T_k\cap\mathcal{E}_h^{\text{BC}}} \int_{E_{kn}} \mu_{knj}\,\mu_{kni}\,\mathrm{d}s \tag{7.15}$$

for $i,j \in \{1,2,\dots,\overline{N}\}$, $\bar{k} \in \{1,2,\dots,\overline{K}\}$ and the matrix entries given in (7.12). These integrals are over edges on the domain boundary, so that each integral is only evaluated once.

The last term — term XI — incorporates boundary data into the hybrid equation. For the inflow boundary edges, we obtain a contribution to the right-hand side $K_{\mu,D} \in \mathbb{R}^{\overline{K}\overline{N}}$ with components

$$[K_{\mu,D}]_{(\bar{k}-1)\overline{N}+i} = \sum_{E_{kn}\in\partial T_k\cap\mathcal{E}_h^{\text{BC,in}}} \int_{E_{kn}} w_{\mathrm{D}}\,\mu_{kni}\,\mathrm{d}s\,,$$

and outflow boundaries add a contribution to the matrix $\mathrm{K}_{\mu,\text{out}} \in \mathbb{R}^{\overline{K}\overline{N}\times KN}$ where components are given as

$$[\mathrm{K}_{\mu,\text{out}}]_{(\bar{k}-1)\overline{N}+i,(k-1)N+j} = \sum_{E_{kn}\in\partial T_k\cap\mathcal{E}_h^{\text{BC,out}}} \int_{E_{kn}} \varphi_{kj}\,\mu_{kni}\,\mathrm{d}s,$$

meaning that we obtain a block matrix similar to T (see (7.14)).

7.2.3. Time discretization

In the FESTUNG framework we employ diagonally implicit Runge-Kutta (DIRK) methods as introduced in Sec. 4.3. Therefore, we do not repeat the details about the time integrators here, but briefly introduce the time discretization using the nomenclature of the FESTUNG framework. The system of equations in (7.5) in semi-discrete form can be reformulated in matrix notation as

$$\begin{aligned} \mathrm{M}_\varphi \partial_t W(t) &= B_\varphi(t) - \bar{\mathrm{L}}(t)\, W(t) - \overline{\mathrm{M}}(t)\, \Lambda(t) \;=:\; R_{\mathrm{RK}}(t, W(t), \Lambda(t))\,, \\ 0 &= B_\mu(t) - \mathrm{N}(t) W(t) - \mathrm{P}\Lambda(t), \end{aligned}$$

with matrices as defined in (7.5) This is a first order differential algebraic equation as described in Sec. 3.3. In the FESTUNG framework we use the DIRK schemes of orders 1 to 4 by Alexander [4] and Hairer and Wanner [101] without time step adaptation, see also Sec. 4.3. Time step adaptation, however, would be easy to incorporate as described in Sec. 4.3.

For the employed DIRK scheme with s stages, the update at t^{n+1} is obtained by solving

$$\begin{aligned} \mathrm{M}_\varphi W^{(i)} &= \mathrm{M}_\varphi W^n + \Delta t \sum_{j=1}^{i} a_{ij}\, R_{\mathrm{RK}}\left(t^{(j)}, W^{(j)}, \Lambda^{(j)}\right) \\ 0 &= B_\mu^{(i)} - \mathrm{N}^{(i)} W^{(i)} - \mathrm{P}\Lambda^{(i)} \end{aligned}$$

for $i = 1 \ldots, s$ and setting

$$W^{n+1} = W^{(s)}$$

with $t^{(i)} = t^n + c_i \Delta t^n$ and coefficients a_{ij}, b_j, c_i are defined in the routine `rungeKuttaImplicit` (see Butcher tableau in Tab. 4.2b).

Remark 7 *All of the employed DIRK schemes in the FESTUNG framework are A- and L-stable [101].*

Remark 8 *The DIRK schemes used in the FESTUNG framework are stiffly accurate, i.e.*

$$b_j = a_{sj}, \quad j = 1, \ldots, s\,.$$

This means in particular that the last update $W^{(s)}$ is the updated solution at the new time t^{n+1}.

7.3. Implementation

In the first publication [80] the following implementation conventions have been formulated:

1. "*Compute every piece of information only once.* In particular, this means that stationary parts of the linear system to be solved in a time step should be kept in the memory and not repeatedly assembled and that the evaluation of functions at quadrature points should be carried out only once."

2. "*Avoid long* `for` *loops.* With "long" loops we mean loops that scale with the mesh size, e. g., loops over the triangles $T_k \in \mathcal{T}_h$ or edges $E_{\bar{k}} \in \mathcal{E}_h$. Use *vectorization* instead."

3. "*Avoid changing the nonzero pattern of sparse matrices.* Assemble global block matrices with the command `sparse( , , , , )`, `kron`, or comparable commands."

In our implementation we follow these conventions as close as possible.

In this section, we present some of the general ideas used in FESTUNG. We define operators used for an efficient implementation of the HDG method and briefly present the structure of the program. Afterwards we present the construction of a selection of matrices needed to assemble the system of equations (7.5). We also discuss the implementation of these routines in MATLAB / GNU Octave.

We cover only a small choice of matrices in order to keep this section compact. Namely, we discuss the construction of the matrices G^m, S, $\mathrm{S}_{\mathrm{out}}$, $\bar{\mathrm{M}}_\mu$ and $\tilde{\mathrm{M}}_\mu$. The matrices G^m differ significantly from their counterpart in the previous publication [195] because the convection velocity u is evaluated at each integration point. The matrices S and $\mathrm{S}_{\mathrm{out}}$ differ from their counterparts in the previous publication since they incorporate the hybrid basis functions μ_h as the matrices stem from an edge integral that evaluates the hybrid unknown λ_h. The last matrices, $\bar{\mathrm{M}}_\mu$ and $\tilde{\mathrm{M}}_\mu$, that we describe, are matrices resulting from the hybrid equation (7.3b) and thus are absent in the previous parts of the paper series. The full documentation and explanation of further assembly routines can be found in the publications [80, 124, 195] and in the documented implementation on Github [196].

7.3.1. Backtransformation to reference element and reference interval

The integrals over edges $E_{\bar{k}}$ and elements T_k are not evaluated in the physical space. Instead the computations are done on a reference triangle $\hat{T} = \{[0,0]^{\mathrm{T}}, [1,0]^{\mathrm{T}}, [0,1]^{\mathrm{T}}\}$ and the reference interval $[0,1]$. Quantities evaluated on the reference triangle and interval are denoted by $\hat{\cdot}$. The result is transferred from the reference triangle to any triangle $T_k = \{x_{k1}, x_{k2}, x_{k3}\} \in \mathcal{T}_h$ using an affine mapping F_k. Besides the affine mapping F_k we also have its inverse F_k^{-1} that maps from the element in physical space back to the reference element $\hat{T}$. We briefly describe the transformation of integrals and discuss the newly introduced mappings used for the HDG method. A more extensive description of the mappings and the transformation of integrals can be found in the first FESTUNG publication [80].

For functions $w : T_k \to \mathbb{R}$ and $\hat{w} : \hat{T} \to \mathbb{R}$, we imply $\hat{w} = w \circ F_k$, i. e., $w(x) = \hat{w}(\hat{x})$. The gradient is transformed using the chain rule:

$$\nabla \;=\; \left(\hat{\nabla} F_k\right)^{-\mathrm{T}} \hat{\nabla}, \tag{7.17}$$

where we abbreviated $\hat{\nabla} = [\partial_{\hat{x}^1}, \partial_{\hat{x}^2}]^{\mathrm{T}}$. This results in transformation formulae for integrals

over an element T_k or an edge $E_{kn} \subset T_k$ for a function $w : \Omega \to \mathbb{R}$

$$\int_{T_k} w(x)\,\mathrm{d}x \;=\; \frac{|T_k|}{|\hat{T}|}\int_{\hat{T}} w\circ F_k(\hat{x})\,\mathrm{d}\hat{x} \;=\; 2|T_k|\int_{\hat{T}} \hat{w}(\hat{x})\,\mathrm{d}\hat{x}\,, \tag{7.18a}$$

$$\int_{E_{kn}} w(x)\,\mathrm{d}x \;=\; \frac{|E_{kn}|}{|\hat{E}_n|}\int_{\hat{E}_n} w\circ F_k(\hat{x})\,\mathrm{d}\hat{x} \;=\; \frac{|E_{kn}|}{|\hat{E}_n|}\int_{\hat{E}_n} \hat{w}(\hat{x})\,\mathrm{d}\hat{x}\,. \tag{7.18b}$$

In the same manner we introduce a mapping $\hat{\boldsymbol{\gamma}}_n : [0,1] \to \hat{E}_n$ from the reference interval $[0,1]$ to the nth edge of the reference element [80]. We use this to transform Eq. (7.18b) further

$$\frac{|E_{kn}|}{|\hat{E}_n|}\int_{\hat{E}_n} \hat{w}(\hat{x})\,\mathrm{d}\hat{x} = \frac{|E_{kn}|}{|\hat{E}_n|}\int_0^1 \hat{w}\circ\hat{\boldsymbol{\gamma}}_n(s)\,|\hat{\boldsymbol{\gamma}}_n'(s)|\,\mathrm{d}s = |E_{kn}|\int_0^1 \hat{w}\circ\hat{\boldsymbol{\gamma}}_n(s)\,\mathrm{d}s\,,$$

where we use the fact that $|\hat{\boldsymbol{\gamma}}_n'(s)| = |\hat{E}_n|$.

A difference compared to previous publications in the FESTUNG paper series are edge integrals with basis functions from an adjoining element and basis functions defined on the edge, e.g., $\int_{E_{kn}} \varphi_{ki}\,\mu_{\bar{k}j}\,\mathrm{d}x$ with $\bar{k} = \rho(k,n)$ as defined in (7.1). They are transformed according to transformation rules (7.18b) and $\hat{\boldsymbol{\gamma}}_n$ such that

$$\int_{E_{kn}} \varphi_{ki}\,\mu_{knj}\,\mathrm{d}x = |E_{kn}|\int_0^1 (\hat{\varphi}_i\circ\hat{\boldsymbol{\gamma}}_n(s))\,\left(\hat{\mu}_j\circ\hat{\beta}_{k\bar{k}}(s)\right)\,\mathrm{d}s\,,$$

where we introduced an additional mapping $\hat{\beta}_{k\bar{k}} : [0,1] \to [0,1]$ that adapts the edge orientation to match the definition of $\mu_{knj} = \mu_{\bar{k}j}$, which is given as

$$\hat{\beta}_{k\bar{k}}(s) = \begin{cases} s & \text{if } \kappa(\rho(k,n),1) = k\,, \\ 1-s & \text{if } \kappa(\rho(k,n),2) = k \end{cases} \tag{7.19}$$

with $\kappa(\rho(k,n),l) = \kappa(\bar{k},l)$ as given in (7.2).

As in the other FESTUNG publications [80, 195], triangle and edge integrals are approximated using numerical quadrature rules after transformation to reference element $\hat{T}$ or reference interval $[0,1]$, respectively. For the reference triangle it reads

$$\int_{\hat{T}} \hat{w}(\hat{x})\mathrm{d}\hat{x} \approx \sum_{r=1}^{R} \omega_r \hat{w}(\hat{x}_r)$$

with integration weight $\omega_r \in \mathbb{R}$ and integration points $\hat{x}_r \in \hat{T}$. Similar formulae are used for integration on the reference interval. Further information about the numerical integration can be found in the previous publications. In the experiments presented we use quadrature rules of order $2P+1$ on both elements and edges.

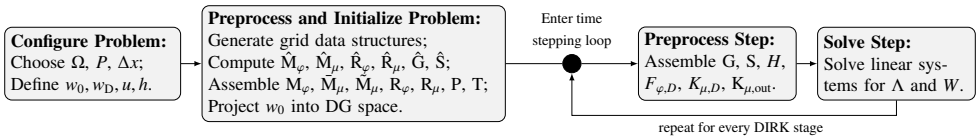


Figure 7.2.: Structure of the solution algorithm. The last two steps are executed repeatedly as part of the time stepping loop. The image is taken from our publication [124].

7.3.2. Program structure

The general control flow of the program is depicted in Fig. 7.2. At first, the implemented solver starts with a pre-processing phase that includes the definition of solver parameters, initial data, boundary conditions, and the right-hand side function followed by the assembly of time-independent matrix blocks and global matrices. In the time stepping loop, the time-dependent global matrices and right-hand side vectors are assembled before solving the resulting linear system. Note that this has to be done for every DIRK stage. Output files in VTK or Tecplot file formats are written after a user-defined number of time steps. Our implementation features a new program structure that differs from the one used in the previous FESTUNG publications. It has been introduced to allow for a more flexible implementation of DG schemes than before and will be discussed in detail in the fourth paper in the series [194] by other authors.

Solving the local problems

The local solves, see Sec. 3.3.2, require the inversion of a matrix

$$\begin{pmatrix} \Sigma \\ W \end{pmatrix} = \underbrace{\begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} \end{pmatrix}^{-1} \begin{pmatrix} R_\sigma \\ R_w \end{pmatrix}}_{=:L_1} - \underbrace{\begin{pmatrix} \mathrm{A}_{\sigma\sigma} & \mathrm{A}_{\sigma w} \\ \mathrm{A}_{w\sigma} & \mathrm{A}_{ww} \end{pmatrix}^{-1} \begin{pmatrix} \mathrm{A}_{\sigma\lambda} \\ \mathrm{A}_{w\lambda} \end{pmatrix}}_{=:\mathrm{L}_2} \Lambda, \qquad \text{(3.28 revisited)}$$

that is block-diagonal. In order to solve the resulting system of equation we can implement the algorithm in several different ways: In our MATLAB / GNU Octave implementation, it turned out to be most efficient to explicitly compute the inverse of the matrix in a block-wise fashion and then to apply the inverse matrix to the right hand side. Thus we select a number of elements and invert the corresponding blocks at once instead of inverting all element-blocks separately or inverting the entire matrix at once. This block-wise inversion is implemented in the routine `blkinv`. The optimal block size depends on the utilized hardware, especially on the cache sizes of the employed CPU. Heuristically, we determined $32 \cdot 2^{-P} \cdot N$ to be a good choice in our case (for hardware details see Tab. 7.2). Optionally, one could parallelize the local solves since they do not depend on each other.

7.3.3. Assembly

In this section, the vectorized assembly of the block matrices in (7.5) is outlined. For that purpose, the required terms are transformed to reference triangle $\hat{T}$ or reference interval $[0, 1]$ and then evaluated by numerical quadrature.

As in the previous papers in the series [80, 195], we make extensive use of the Kronecker product $\mathrm{A} \otimes \mathrm{B}$ of two matrices $\mathrm{A} \in \mathbb{R}^{m_a \times n_a}$, $\mathrm{B} \in \mathbb{R}^{m_b \times n_b}$ defined as

$$\mathrm{A} \otimes \mathrm{B} \coloneqq [[\mathrm{A}]_{i,j}\, \mathrm{B}] \in \mathbb{R}^{m_a m_b \times n_a n_b}\,. \tag{7.20}$$

We use the brackets $[\,\cdot\,]$ to refer to the entry of a matrix. Thus, $[\mathrm{A}]_{i,j}$ refers to the entry of matrix A in the ith row and jth column. The resulting matrix of the Kronecker product is a matrix where each entry is the product of one entry of A multiplied with the matrix B. Additionally, we employ the operation $\mathrm{A} \otimes_{\mathrm{V}} \mathrm{B}$ with $m_b = r\, m_a, r \in \mathbb{N}$ introduced in the second FESTUNG paper [195] as

$$\mathrm{A} \otimes_{\mathrm{V}} \mathrm{B} \coloneqq \left[[\mathrm{A}]_{i,j}\, [\mathrm{B}]_{(i-1)r:ir,:}\right] \in \mathbb{R}^{m_b \times n_a n_b}\,, \tag{7.21}$$

which can be interpreted as a Kronecker product that takes a different right-hand side for every row of the left-hand side. This operation is implemented in the routine `kronVec`. In many cases we must select edges matching a certain criterion, e.g., edges in the interior $E_{kn} \in \mathcal{E}_h^{\mathrm{in}}$. We denote this using the Kronecker delta symbol with a matching subscript that indicates the criterion to be met, for example

$$\delta_{E_{kn} \in \mathcal{E}_h^{\mathrm{in}}} \coloneqq \begin{cases} 1 & \text{if } E_{kn} \in \mathcal{E}_h^{\mathrm{in}}\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{7.22}$$

Some of the block-matrices in the system of equations (7.4) appeared in identical or only slightly different form in previous publications [80, 195]. In this theses, we describe the assembly of the matrices only for a small subset of the matrices needed to construct the system of equations (7.5). The description of the other matrices can be found in the earlier FESTUNG publications or in the third part [124] that focuses on the implementation of HDG methods in FESTUNG. Further differences to the standard DG implementation in [195] arise in all matrices that stem from edge integrals involving the new unknown λ_h and in matrices G^m that are assembled without projecting $u(t, x)$ into the broken polynomial space.

Assembly of G^m

For the assembly of matrices G^m from (7.7) we make use of the transformation rule for the gradient (7.17). Due to the time-dependent function $u_m(t, x)$ in the integrand we cannot

reduce the assembly to Kronecker products of reference matrices as it is possible for the mass matrix, for example. We apply transformation rules (7.17), (7.18a) and obtain

$$\int_{T_k} u_1(t,x)\,\varphi_{kj}\,\partial_x^1\varphi_{ki}\,\mathrm{d}x \approx \sum_{r=1}^{R}\left(\mathrm{B}_k^{22}\,[U_1]_{k,r}\,[\hat{\mathrm{G}}]_{1,r,i,j} - \mathrm{B}_k^{21}\,[U_1]_{k,r}\,[\hat{\mathrm{G}}]_{2,i,j,r}\right),$$
$$\int_{T_k} u_2(t,x)\,\varphi_{kj}\,\partial_x^2\varphi_{ki}\,\mathrm{d}x \approx \sum_{r=1}^{R}\left(-\mathrm{B}_k^{12}\,[U_2]_{k,r}\,[\hat{\mathrm{G}}]_{1,r,i,j} + \mathrm{B}_k^{11}\,[U_2]_{k,r}\,[\hat{\mathrm{G}}]_{2,i,j,r}\right)$$

with a multidimensional array $\hat{\mathrm{G}} \in \mathbb{R}^{2\times N\times N\times R}$ that represents a part of the contribution of the quadrature rule in every integration point $\hat{q}_r$ on the reference element $\hat{T}$. The arrays $U_m \in \mathbb{R}^{K\times R}$ hold the velocity components evaluated in each quadrature point of each element

$$[\hat{\mathrm{G}}]_{m,i,j,r} \coloneqq \omega_r\,\partial_{\hat{x}_m}\,\hat{\varphi}_{ki}\,\hat{\varphi}_{kj}\,, \qquad [U_m]_{k,r} \coloneqq u_m(t, F_k(\hat{q}_r))\,.$$

We integrate the integration weight ω_r into the multidimensional array. The element-local matrix $\mathrm{G}_{T_k}^1 \in \mathbb{R}^{N\times N}$ is then given as

$$\begin{aligned}\mathrm{G}_{T_k}^1 = \sum_{r=1}^{R}\Bigg(& B_k^{22}\begin{bmatrix}\omega_r\,u_1(t,F_k(\hat{q}_r))\,\partial_x^1\hat{\varphi}_1\,\hat{\varphi}_1 & \cdots & \omega_r\;u_1(t,F_k(\hat{q}_r))\,\partial_x^1\hat{\varphi}_1\,\hat{\varphi}_N\\ \vdots & \ddots & \vdots\\ \omega_r\,u_1(t,F_k(\hat{q}_r))\,\partial_x^1\hat{\varphi}_N\,\hat{\varphi}_1 & \cdots & \omega_r\,u_1(t,F_k(\hat{q}_r))\,\partial_x^1\hat{\varphi}_N\,\hat{\varphi}_N\end{bmatrix}\\ & -B_k^{21}\begin{bmatrix}\omega_r\,u_1(t,F_k(\hat{q}_r))\,\partial_x^2\hat{\varphi}_1\,\hat{\varphi}_1 & \cdots & \omega_r\,u_1(t,F_k(\hat{q}_r))\,\partial_x^2\hat{\varphi}_1\,\hat{\varphi}_N\\ \vdots & \ddots & \vdots\\ \omega_r\,u_1(t,F_k(\hat{q}_r))\,\partial_x^2\hat{\varphi}_N\,\hat{\varphi}_1 & \cdots & \omega_r\,u_1(t,F_k(\hat{q}_r))\,\partial_x^2\hat{\varphi}_N\,\hat{\varphi}_N\end{bmatrix}\Bigg)\\ =& \sum_{r=1}^{R}\left(B_k^{22}\,[U_1]_{k,r}\,[\hat{\mathrm{G}}]_{1,:,:,r} - B_k^{21}\,[U_1]_{k,r}\,[\hat{\mathrm{G}}]_{2,:,:,r}\right).\end{aligned}$$

On first sight, this procedure appears to closely follow the assembly of G^m in [195]; however, conceptually, there is a big difference: The presented procedure uses a quadrature rule which is applied to all elements at the same time using blocks of basis functions evaluated in each quadrature point, whereas the procedure to assemble G^m in [195] builds the full matrix from the contributions of each degree of freedom of the projected DG representation of the velocity using already integrated reference blocks. To speed up the implementation, we do not assemble a global sparse matrix in each iteration of the `for`-loop over quadrature points and adding to the sparse matrix from the previous iteration. Instead, we use the standard Kronecker product (7.20) to build a dense $KN \times N$ vector of blocks, from which the global sparse matrix is constructed using the vectorial Kronecker operator (7.21)

$$G^1 = \sum_{r=1}^{R} I_{K\times K} \otimes_{\mathrm{V}} \left(\begin{bmatrix}B_1^{22}\,[U_1]_{1,r}\\ \vdots\\ B_K^{22}[U_1]_{K,r}\end{bmatrix} \otimes [\hat{\mathrm{G}}]_{1,:,:,r} - \begin{bmatrix}B_1^{21}\,[U_1]_{1,r}\\ \vdots\\ B_K^{21}[U_1]_{K,r}\end{bmatrix} \otimes [\hat{\mathrm{G}}]_{2,:,:,r}\right),$$

where $I_{K\times K}$ is the $K \times K$ identity matrix. The matrix G^2 is assembled analogously.

The function `assembleMatElemDphiPhiFuncContVec` assembles the matrices G following the procedure described. Reference blocks $\hat{\mathrm{G}}$ are provided in parameter `refElemDphiPhiPerQuad` and have been pre-computed by `integrateRefElemDphiPhiPerQuad`. The two components of the continuous function in the integrand, in our case the velocity components, are given as function handles in `funcCont1` and `funcCont2`. This allows to evaluate the velocity components at each integration point. The integration points are obtained from `quadRule2D` and the evaluated velocities in x_1- and x_2-direction are stored in `valOnQuad1` and `valOnQuad2`. Note that no long `for`-loops are used, but most of the computations is done using vectorized operations. The assembled matrices G^1 and G^2 are stored in a `cell` named `ret` and returned by the function.

```
function ret = assembleMatElemDphiPhiFuncContVec(g, refElemDphiPhiPerQuad, funcCont1, funcCont2,
    ↪ qOrd)
K = g.numT; [N, ~, R] = size(refElemDphiPhiPerQuad{1});
if nargin < 5, p = (sqrt(8*N+1)-3)/2;  qOrd = max(2*p, 1); end
[Q1, Q2, ~] = quadRule2D(qOrd);
ret = { zeros(K*N, N), zeros(K*N,N) };
for r = 1 : R
  valOnQuad1 = funcCont1(g.mapRef2Phy(1, Q1(r), Q2(r)), g.mapRef2Phy(2, Q1(r), Q2(r)));
  ret{1} = ret{1} + kron(g.B(:,2,2) .* valOnQuad1, refElemDphiPhiPerQuad{1}(:, :, r)) ...
                  - kron(g.B(:,2,1) .* valOnQuad1, refElemDphiPhiPerQuad{2}(:, :, r));
  valOnQuad2 = funcCont2(g.mapRef2Phy(1, Q1(r), Q2(r)), g.mapRef2Phy(2, Q1(r), Q2(r)));
  ret{2} = ret{2} - kron(g.B(:,1,2) .* valOnQuad2, refElemDphiPhiPerQuad{1}(:, :, r)) ...
                  + kron(g.B(:,1,1) .* valOnQuad2, refElemDphiPhiPerQuad{2}(:, :, r));
end % for
ret{1} = kronVec(speye(K,K), ret{1});
ret{2} = kronVec(speye(K,K), ret{2});
end % function
```

Assembly of S and S_{out}

The matrices S and S_{out} are assembled together as they have the same structure, but stem from different edges of the mesh. Thus, the set of relevant edges is expanded to $\mathcal{E}_h^{\text{in}} \cup \mathcal{E}_h^{\text{BC,out}}$. On a relevant edge E_{kn} we transform terms of the form given in Eq. (7.9a) using the transformation rules and approximate the integral by a one-dimensional numerical quadrature rule:

$$\begin{aligned}\int_{E_{kn}} (u \cdot n_{kn})\, \varphi_{ki}\, \mu_{knj}\, \mathrm{d}s &= |E_{kn}| \int_0^1 \left((u(t) \circ F_k \circ \hat{\boldsymbol{\gamma}}_n(s)) \cdot n_{kn}\right)\, \hat{\varphi}_i \circ \hat{\boldsymbol{\gamma}}_n(s)\, \hat{\mu}_j \circ \hat{\beta}_{k\bar{k}}(s)\, \mathrm{d}s \\ &\approx \sum_{r=1}^{R} \underbrace{\left((u(t) \circ F_k \circ \hat{\boldsymbol{\gamma}}_n(\hat{q}_r)) \cdot n_{kn}\right)}_{=:[U_n]_{k,n,r}} \underbrace{\omega_r\, \hat{\varphi}_i \circ \hat{\boldsymbol{\gamma}}_n(\hat{q}_r)\, \hat{\mu}_j \circ \hat{\beta}_{k\bar{k}}(\hat{q}_r)}_{=:[\hat{\mathrm{S}}]_{i,j,n,r,l}},\end{aligned}$$

where $U_n \in \mathbb{R}^{K\times 3\times R}$ holds the normal velocity evaluated in each quadrature point, and the subscript l in $\hat{\mathrm{S}} \in \mathbb{R}^{N\times\overline{N}\times 3\times R\times 2}$ covers the two cases of $\hat{\beta}_{k\bar{k}}$ in (7.19). This allows us to

assemble the global matrix as

$$\mathrm{S} = \sum_{n=1}^{3} \Delta_n \otimes_{\mathrm{V}} \left(\sum_{r=1}^{R} \sum_{l=1}^{2} \begin{bmatrix} \delta_{E_{1n} \in \mathcal{E}_h^{\mathrm{in}} \cup \mathcal{E}_h^{\mathrm{BC,out}}} |E_{1n}| \, \delta_{\kappa(\rho(1,n),l)=1} \, [U_n]_{1,n,r} \\ \vdots \\ \delta_{E_{Kn} \in \mathcal{E}_h^{\mathrm{in}} \cup \mathcal{E}_h^{\mathrm{BC,out}}} |E_{Kn}| \, \delta_{\kappa(\rho(K,n),l)=K} \, [U_n]_{1,n,r} \end{bmatrix} \otimes [\hat{\mathrm{S}}]_{:,:,n,r,l} \right) \tag{7.23}$$

with

$$\Delta_n \coloneqq \begin{bmatrix} \delta_{E_{1n}=E_1} & \cdots & \delta_{E_{1n}=E_{\overline{K}}} \\ \vdots & \ddots & \vdots \\ \delta_{E_{Kn}=E_1} & \cdots & \delta_{E_{Kn}=E_{\overline{K}}} \end{bmatrix}. \tag{7.24}$$

We introduce the permutation matrix $\Delta_n \in \mathbb{R}^{K \times \overline{K}}$, $n \in \{1,2,3\}$ that has a single entry per row indicating the correspondence $E_{kn} = E_{\bar{k}}$ for all elements and edges. It takes care of the necessary permutations from the element-based view of the assembly to the edge-based view of the edge degrees of freedom.

The matrices are assembled in `assembleMatEdgePhiIntMuVal`. Both matrices are assembled at once by specifying all interior and outflow edges in `markE0T`. The reference block $\hat{\mathrm{S}}$ is evaluated at each quadrature point and pre-computed by `integrateRefEdgePhiIntMuPerQuad` and handed to the function in the parameter `refEdgePhiIntMuPerQuad`. The parameter `valOnQuad` stores the normal velocity U_n evaluated at the quadrature points of each edge. The matrices are then constructed following (7.23) with three nested `for` loops that reproduce the summation. The resulting matrix $\mathrm{S} + \mathrm{S}_{\mathrm{out}}$ is stored in `ret` and returned by the functions. As before all loops are short and are independent of the number of elements in the mesh.

```
function ret = assembleMatEdgePhiIntMuVal(g, markE0T, refEdgePhiIntMuPerQuad, valOnQuad)
K = g.numT; Kbar = g.numE;
[N, Nmu, ~, R] = size(refEdgePhiIntMuPerQuad{1});
ret = sparse(K*N, Kbar*Nmu);
for n = 1 : 3
  RknTimesVal = sparse(K*N, Nmu);
  markAreaE0T = markE0T(:, n) .* g.areaE0T(:, n);
  for l = 1 : 2
    markAreaSideE0T = markAreaE0T .*  g.markSideE0T(:, n, l);
    for r = 1 : R
      RknTimesVal = RknTimesVal + kron(markAreaSideE0T .* valOnQuad(:, n, r), ...
                    refEdgePhiIntMuPerQuad{l}(:, :, n, r));
    end % for r
  end % for l
  ret = ret + kronVec(sparse(1:K, g.E0T(:, n), ones(K, 1), K, Kbar), RknTimesVal);
end % for n
end % function
```

Assembly of $\bar{\mathrm{M}}_\mu$ and $\tilde{\mathrm{M}}_\mu$

The hybrid mass matrices $\bar{\mathrm{M}}_\mu$ and $\tilde{\mathrm{M}}_\mu$ (see Eq. (7.12) and (7.15)) stem from the hybrid equation. In contrast to the two types of matrices described before the hybrid mass matrices

rely only on hybrid test and basis functions μ_h. We apply the transformation rules to obtain

$$\int_{E_{kn}} \mu_{knj}\,\mu_{kni}\,\mathrm{d}s = |E_{kn}| \int_0^1 \hat{\mu}_j \circ \hat{\beta}_{k\bar{k}}(s)\,\hat{\mu}_i \circ \hat{\beta}_{k\bar{k}}(s)\,\mathrm{d}s = |E_{kn}| \underbrace{\int_0^1 \hat{\mu}_j(s)\,\hat{\mu}_i(s)\,\mathrm{d}s}_{=:[\hat{\mathrm{M}}_\mu]_{i,j}} .$$

We can use the permutation matrices Δ_n (see Eq. (7.24)) again. Then, we obtain the global matrices

$$\bar{\mathrm{M}}_\mu = \sum_{n=1}^{3} \left((\Delta_n)^{\mathrm{T}} \begin{bmatrix} |E_{1n}|\,\delta_{E_{1n}\in\mathcal{E}_h^{\mathrm{in}}} & & \\ & \ddots & \\ & & |E_{Kn}|\,\delta_{E_{Kn}\in\mathcal{E}_h^{\mathrm{in}}} \end{bmatrix} \Delta_n \right) \otimes \hat{\mathrm{M}}_\mu ,$$

$$\tilde{\mathrm{M}}_\mu = \sum_{n=1}^{3} \left((\Delta_n)^{\mathrm{T}} \begin{bmatrix} |E_{1n}|\,\delta_{E_{1n}\in\mathcal{E}_h^{\mathrm{BC}}} & & \\ & \ddots & \\ & & |E_{Kn}|\,\delta_{E_{Kn}\in\mathcal{E}_h^{\mathrm{BC}}} \end{bmatrix} \Delta_n \right) \otimes \hat{\mathrm{M}}_\mu .$$

The assembly only differs in the edges that the matrix has to be assembled. It is identified using the Kronecker delta as defined in (7.22).

In `assembleMatEdgeMuMu` the hybrid mass matrices $\bar{\mathrm{M}}_\mu$ and $\tilde{\mathrm{M}}_\mu$ are assembled. The array `markE0T` plays the role of the Kronecker delta in the matrix definition by selecting the relevant edges for either matrix. Depending on the edges marked either $\bar{\mathrm{M}}_\mu$ or $\tilde{\mathrm{M}}_\mu$ is assembled. The input parameter `refEdgeMuMu` is the mass matrix $\hat{\mathrm{M}}_\mu$ evaluated on the reference interval $[0, 1]$. It is pre-computed by `integrateRefEdgeMuMu`. Once again, the `for`-loop is short and does not depend on the mesh size. The assembled mass matrix is stored in `ret` and returned by the function.

```
function ret = assembleMatEdgeMuMu(g, markE0T, refEdgeMuMu)
Nmu = size(refEdgeMuMu, 1); Kedge = g.numE;
ret = sparse(Kedge * Nmu, Kedge * Nmu);
for n = 1 : 3
  Kkn = g.areaE0T(:, n) .*  markE0T(:, n) ;
  ret = ret + kron(sparse(g.E0T(:, n), g.E0T(:, n), Kkn, Kedge, Kedge), refEdgeMuMu);
end % for
end % function
```

7.4. Numerical results

In this section, we verify our implementation by means of convergence experiments followed by some performance analysis of the code. In addition, a runtime comparison of the presented HDG discretization against a time-implicit version of the DG discretization from the second FESTUNG publication [195] is given. Further results for different parameters or test cases that verify our implementation can be found in the related publication [124].

P	0		1		2		3	
j	$\|w_h - w\|$	EOC	$\|w_h - w\|$	EOC	$\|w_h - w\|$	EOC	$\|w_h - w\|$	EOC
1	2.69e-01	—	6.42e-02	—	7.35e-03	—	1.50e-03	—
2	1.86e-01	0.53	1.75e-02	1.88	7.41e-04	3.31	9.79e-05	3.94
3	1.18e-01	0.66	4.32e-03	2.02	8.55e-05	3.12	6.26e-06	3.97
4	6.86e-02	0.78	1.07e-03	2.01	1.04e-05	3.03	3.96e-07	3.98
5	3.78e-02	0.86	2.68e-04	2.00	1.30e-06	3.01	2.52e-08	3.97

Table 7.1.: L^2-discretization errors for the unsteady problem in Sec. 7.4.1 measured at $t_{\text{final}} = 2$ and experimental orders of convergence for different polynomial degrees using HDG. We have $\Delta x_j = \frac{1}{3 \cdot 2^j}$, $K = 18 \cdot 4^j$ triangles, and $\Delta t_j = \frac{1}{5 \cdot 2^j}$ in the jth refinement level. The CFL number of the test case is $\text{CFL} = \frac{3}{5} e^1 \approx 1.63$.

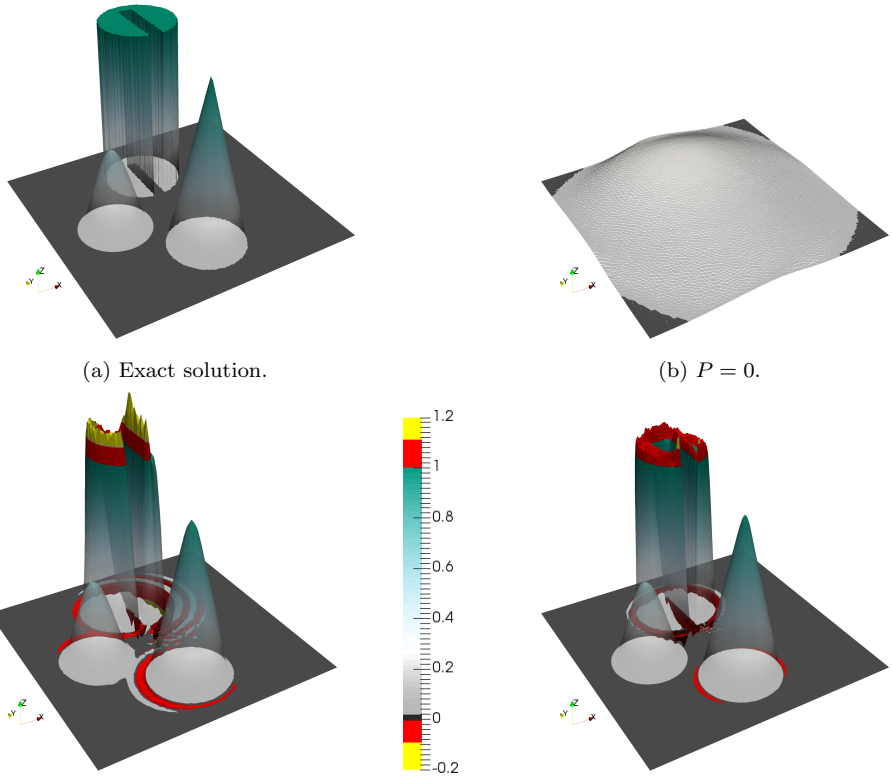
7.4.1. Analytical convergence tests

Our implementation is verified by comparing the experimental orders of convergence to the analytically predicted ones for smooth solutions. For that, we choose an exact solution $w(t, x)$ and a velocity field $u(t, x)$, with which we derive boundary data w_{D} and source term h analytically by substituting w and u into the convection equation. The discretization error $\|w_h - w\|_{L^2}$ is computed as the L^2-norm of the difference between the numerical and the analytical solutions at the end time (see [80]). From that, the experimental order of convergence EOC is given by

$$\text{EOC} \coloneqq \ln\left(\frac{\|w_{\Delta x_{j-1}} - w\|_{L^2}}{\|w_{\Delta x_j} - w\|_{L^2}}\right) \Bigg/ \ln\left(\frac{\Delta x_{j-1}}{\Delta x_j}\right).$$

We verify the time and space discretization using a time-dependent test case that is inspired by the steady state test case used in the previous FESTUNG publication [195]. The exact solution is given by $w(t, x) = \cos(7x_1)\cos(7x_2) + \exp(-t)$ and the velocity field is set to $u(x) \coloneqq [\exp((x_1 + x_2)/2), \exp((x_1 - x_2)/2)]^{\text{T}}$. The source term is determined accordingly. The problem is solved on $\Omega = [0, 1]^2$ up to $t_{\text{final}} = 2$. The mesh size and time step size are refined as $\Delta x_j = \frac{1}{3 \cdot 2^j}$ and $\Delta t_j = \frac{1}{5 \cdot 2^j}$ with j referring to the refinement level. The order of the DIRK scheme is chosen as $P + 1$.

In Tab. 7.1, we present the solution up to $P = 3$. In all cases the method shows the expected order of convergence of $\text{EOC} = P + 1$ in time and space. In our publication [124], we present additional test cases that test the correct implementation of the time and space discretization separately.



(a) Exact solution.

(b) $P = 0$.

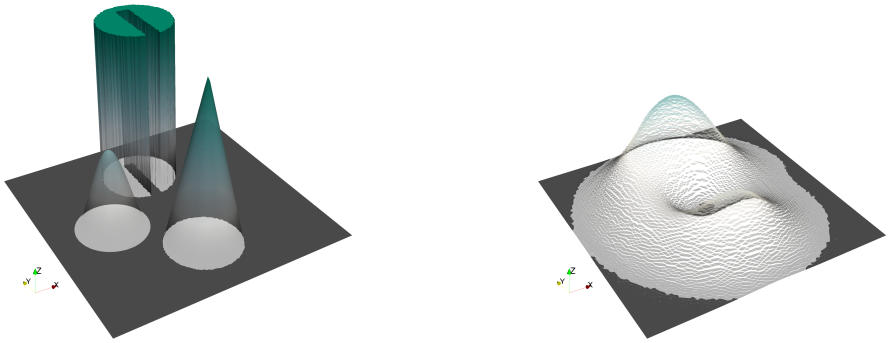
(c) $P = 1$.

(d) $P = 2$.

(e) $P = 3$.

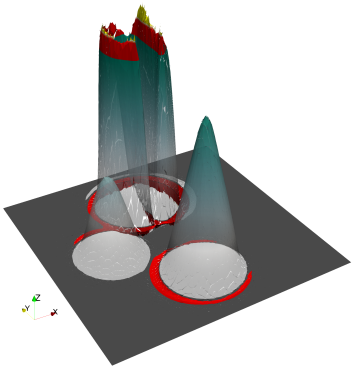
(f) $P = 4$.

Figure 7.3.: HDG solutions for the solid body rotation benchmark for different polynomial orders at end time $t_{\text{end}} = 2\pi$. The problem is solved on $\Omega = [0, 1]^2$ with velocity field $u(x) = [0.5 - x_2, x_1 - 0.5]^{\text{T}}$. The CFL-number is CFL ≈ 2.104.

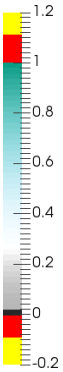


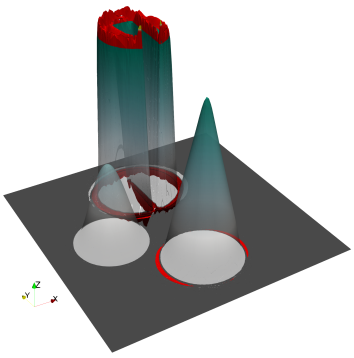
(a) Exact solution.

(b) $P = 0$.

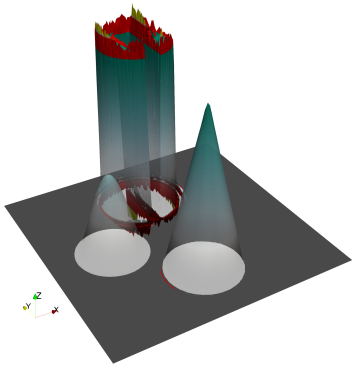


(c) $P = 1$.

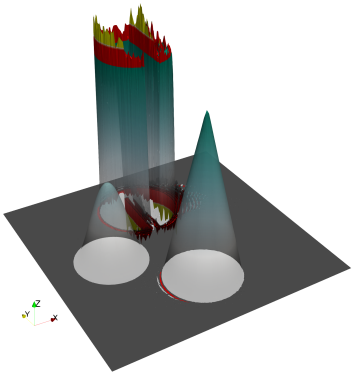




(d) $P = 2$.



(e) $P = 3$.



(f) $P = 4$.

Figure 7.4.: DG solutions for the solid body rotation benchmark for different polynomial orders at end time $t_{\text{end}} = 2\pi$. The problem is solved on $\Omega = [0, 1]^2$ with velocity field $u(x) = [0.5 - x_2, x_1 - 0.5]^{\text{T}}$. The CFL-number is CFL ≈ 2.104.

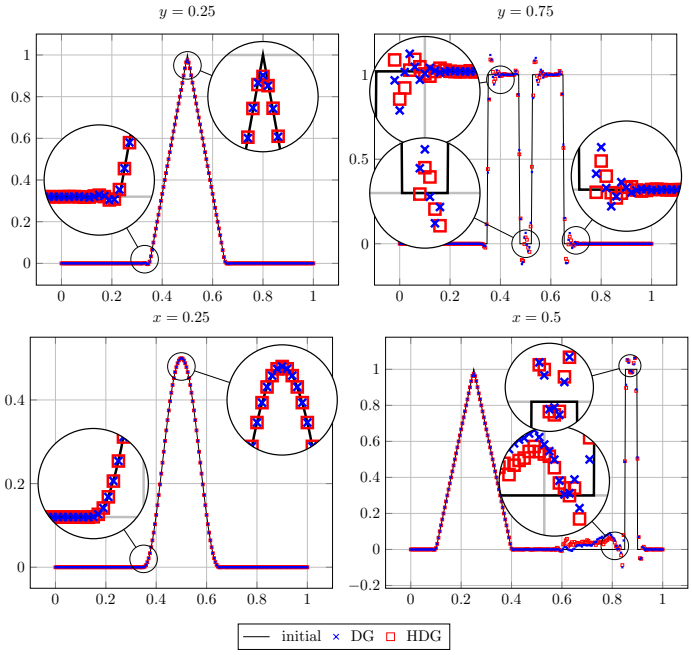


Figure 7.5.: Cross-sections of the two different DG solutions for the solid body rotation benchmark with $P = 3$ at end time $t_{\text{end}} = 2\pi$. The problem is solved on $\Omega = [0, 1]^2$ with velocity field $u(x) = [0.5 - x_2, x_1 - 0.5]^{\mathrm{T}}$. The CFL-number is CFL ≈ 2.104.

7.4.2. Comparison to the unhybridized DG implementation

The second part of the FESTUNG paper series was also concerned with the convection equation [195]. In contrast to the work presented here, an *unhybridized* (upwind) discontinuous Galerkin discretization with explicit strong stability preserving Runge-Kutta methods was used. In order to make both implementations comparable, we implemented a variant of the upwind DG solver that incorporates the DIRK schemes for time-stepping. Then, we compute solutions to the solid body rotation benchmark proposed by LeVeque [144] to compare the two space discretizations.

The initial solution consists of a slotted cylinder, a sharp cone, and a smooth hump (see Fig. 7.3a) placed in a square domain $\Omega = [0,1]^2$ with velocity field $u(x) = [0.5 - x_2, x_1 - 0.5]^{\mathrm{T}}$ producing a full counterclockwise rotation of the initial scene over time interval at $t_{\text{final}} = 2\pi$. The initial data satisfies

$$w_0(x) = \begin{cases} 1 & \text{if } \begin{array}{l} (x_1 - 0.5)^2 + (x_2 - 0.75)^2 \leq r \\ \wedge\, (x_1 \leq 0.475 \vee x_1 \geq 0.525 \vee x_2 \geq 0.85) \end{array} \\ 1 - G(x, [0.5, 0.25]^{\mathrm{T}}) & \text{if } (x_1 - 0.5)^2 + (x_2 - 0.25)^2 \leq r \\ \frac{1}{4}(1 + \cos(\pi G(x, [0.25, 0.5]^{\mathrm{T}}))) & \text{if } (x_1 - 0.25)^2 + (x_2 - 0.5)^2 \leq r \\ 0 & \text{otherwise} \end{cases}.$$

We set $r = 0.0225$ and $G(x, x_0) \coloneqq \frac{1}{0.15}\|x - x_0\|_2$, choose homogeneous Dirichlet boundary conditions $w_{\mathrm{D}} = 0$ and right-hand side function $h = 0$. The first line of the initial data refers to the slotted cylinder being initialized in the domain. The second line initializes the sharp cone in the domain and the third line initializes the smooth hump. The initial data at all other parts of the domain is zero. For our computations an unstructured mesh is generated by MATLAB's `initmesh`. We set the maximum element size to $\Delta x = 2^{-6}$ which results in a mesh with $K = 14006$ elements. The time step size is set to $\frac{2\pi}{320}$.

In Figs. 7.3 and 7.4, we present the computed solution at end time $t_{\text{end}} = 2\pi$ for different polynomial degrees. We chose a color map (inspired by [11]) that emphasizes violations of the discrete maximum principle. We have a color gradient from black via white to green in the range $[0, 1]$; values in the range $[-0.1, 0) \cup (1, 1.1]$ are colored red, and values in $(-\infty, -0.1) \cup (1.1, +\infty)$ are colored yellow. To make oscillations in the bottom range more visible, we gradually reduce the opacity to zero. Figure 7.5 presents intersection lines for $P = 3$.

Clearly, the lowest order approximation is unusable for this kind of problem in both implementations with numerical diffusion smearing out most (DG) or all (HDG) features of the solution. For $P > 0$, solutions from HDG and DG are in good agreement for all approximation orders with each other. This finding is substantiated by the intersection lines in Fig. 7.5 and the L^2-errors shown in Tab. 3.1 exhibiting only minor differences between both

	$\|w_h(t_{\text{end}}) - w_0\|_{L^2(\Omega)}$		runtime $[s]$			
P	DG	HDG	DG	HDG	Hardware/software details	
0	1.87e-01	2.35e-01	27.1	39.8	CPU	Intel Core-i7
1	7.25e-02	7.78e-02	157	218		4790
2	5.53e-02	5.44e-02	797	717		(Haswell)
3	4.02e-02	4.13e-02	3980	3166	RAM	32 GBytes
4	4.16e-02	4.18e-02	10996	7199	MATLAB	R2017a

Table 7.2.: Comparison of L^2-errors for the solid body rotation benchmark (see Sec. 7.4.2) at end time $t_{\text{end}} = 2\pi$ (using initial data as the exact solution) and runtimes for DG and HDG solvers (left). Details of the employed hardware and software (right). The runtimes have been presented previously in Tab. 3.1.

discretizations. However, also clearly visible are severe violations of the discrete maximum principles and oscillations in the wake of cylinder and cone, which do not become less pronounced with increasing approximation order. This type of behavior can be alleviated using slope limiters as shown in the second FESTUNG paper [195]. Unfortunately, designing slope limiters for implicit time stepping methods is a non-trivial task and lies beyond the scope of this work.

When comparing the runtimes for both discretizations in Table 3.1, it becomes clear that the static condensation outlined in Sec. 3.3.2 becomes advantageous especially for higher approximation orders making HDG a superior approach for time-implicit high-order discretizations.

7.4.3. Performance analysis

A major advantage of the hybridized DG method is the fact that the globally coupled linear equation system resulting from the discretization is relatively compact and easy to solve, see Sec. 3.3.2. In order to support this by numbers we present some performance results that show the runtime distribution among the different steps of the code. In order to highlight the benefits for the solving procedure we disregard pre-processing and initialization tasks. These are only performed once and are usually very similar among different kinds of space discretizations. We consider ten time steps of the solid body rotation benchmark presented in Sec. 7.4.2. The runtimes and their share are determined using MATLAB's profiler. When analyzing the runtimes of the instructions we see that the linear solvers, local and global problem, together with the assembly of the time-dependent block matrices, particularly G and S, are responsible for the majority of the computation time.

In Tab. 7.3, we present the runtimes in seconds and the runtime shares for the most

	$\Delta x_{\max} = 2^{-4}$ ($K = 872$)					$\Delta x_{\max} = 2^{-6}$ ($K = 14006$)				
	runtime	assembly step		solver step		runtime	assembly step		solver step	
P		G	S	local	global		G	S	local	global
0	0.641	0.181 (28.2%)		0.047 (7.3%)		1.156	0.367 (31.7%)		0.617 (53.4%)	
		18.2%	8.3%	25.5%	61.7%		13.4%	19.6%	4.5%	91.9%
1	0.919	0.351 (38.2%)		0.278 (30.3%)		6.698	1.820 (27.2%)		4.674 (69.8%)	
		29.3%	17.9%	33.1%	59.0%		42.6%	29.3%	23.8%	71.2%
2	1.818	0.655 (36.0%)		0.852 (46.9%)		21.945	6.271 (28.6%)		15.491 (70.6%)	
		41.5%	21.4%	34.5%	56.0%		55.8%	31.0%	25.3%	66.1%
3	5.627	3.079 (54.7%)		2.353 (41.8%)		100.089	53.218 (53.2%)		46.688 (46.6%)	
		72.2%	15.6%	33.1%	53.7%		79.2%	17.0%	26.5%	60.2%
4	10.159	6.273 (61.7%)		3.685 (36.3%)		230.148	151.809 (66.0%)		78.151 (34.0%)	
		80.7%	12.3%	27.3%	53.9%		88.2%	9.9%	26.8%	56.6%

Table 7.3.: Runtime distribution for 10 time steps of the solid body rotation benchmark (see Sec. 7.4.2) without initialization tasks (i.e., only time stepping loop) of the HDG method. The runtime is measured using MATLAB's profiler. We compare different mesh sizes and approximation orders with runtimes given in seconds. Percentages for assembly and solvers are relative to the runtime of the time stepping loop, percentages for the assembly of G and S (see Sec. 7.3.3) are relative to assembly runtime, and percentages for computation of local solves L_1 and L_2 (see Sec. 3.3.2) and solving the global system of equations for λ_h are relative to solver step runtimes. Details on the employed hardware and software are given in Table 7.2 (right).

expensive parts of the code; the assembly of matrices and solving of linear systems of equations. We run the experiments for different polynomial degrees $P \in \{0, 1, 2, 3, 4\}$ and on two different meshes with $K = 872$ and $K = 14006$ elements. We focus on the time spent for the assembly of the matrices and the solving of the local and global linear systems of equations.

The results show that the runtime increases also with increasing mesh size and polynomial degree. This is no surprise because in both cases the amount of computations that have to be done increases. It is more interesting to see that the runtime for the assembly grows stronger than for the solving of the linear systems for larger polynomial degrees. For $P > 2$ the assembly step takes up more time than the solver steps.

The percentage of time spend on solving the linear systems grows for increasing polynomial degrees up to $P = 2$ and it decreases afterwards when compared to the assembly routines. At the same time we observe that the percentage of time spent on the global solve, within the solver step, decreases for increasing polynomials degree. There is no big difference in the evolution of the runtime distribution on local and global solves when one compares the coarse with the fine mesh. We would like to point out that the computation of the local solve is element-local and thus could be easily parallelized with virtually perfect scaling.

The distribution of the runtime spent in the assembly step becomes dominated by the assembly of G for increasing polynomial degrees. This is probably caused by the way the matrix is constructed. It grows due to the increased number of degrees of freedom when the polynomial degree is increased. At the same time we also need more integration points when assembling the matrix. Nevertheless, the total runtime increase for higher polynomial approximation orders is not as pronounced as for the unhybridized DG solver as shown in Sec. 7.4.2. The cost of the assembly of G might be decreased by following the approach used in the second part of the FESTUNG paper series [195] in which the convection velocity was projected onto the polynomial space. In this work, we evaluated the convection velocity at every integration point and thus we need a loop over all integration points, see Sec. 7.3.3.

8. Flows with discontinuous solutions

It has been mentioned in the introduction of the governing equations, see Ch. 2, that some PDEs used to model fluid flow are nonlinear and that these nonlinearities can lead to non-smooth solutions. This is also the case for Euler and the Navier-Stokes equations. For flows with high velocities the Navier-Stokes equations are dominated by the nonlinear convective effects such that the viscous features are not enough to ensure a smooth solution when approximated with a numerical scheme. When the solution of such a non-smooth solution is approximated by a high-order method it may lead to serious nonphysical oscillations.

In order to give more insight into the problem of approximating disconinuities with high order polynomials we present the L^2-projection of the sign(x_1) function onto Legendre polynomials for different polynomial degrees, in Fig. 8.1. Oscillations appear due to the discontinuity in the sign(x_1) function. These oscillations are obviously undesirable, but common for methods being at least of second order accurate in space [143]. This behavior is also known as Gibbs phenomenon. As seen in the figure, the problem cannot be resolved by increasing the spatial resolution only, i.e. increasing P, but the numerical discretization has to recognize and respect the discontinuity in some way [62, 109, 138, 143, 162].

The problem of an oscillating solution at discontinuities for higher order methods has been identified early (at the latest in 1950 [162]) and different solution strategies have been proposed. Popular approaches are flux limiters and slope limiters [1, 2, 52, 58, 135, 140–142,

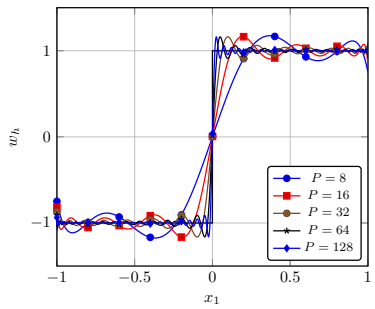


Figure 8.1.: Projection of the sign function onto Legendre polynomials of different degrees. Oscillations (Gibb's phenomenon) are visible.

149] and artificial viscosity (AV) models [17, 94, 133, 139, 154, 155, 162, 164, 171, 184]. A broader overview over different shock capturing approaches can also be found in the text books of LeVeque [143, 145] or Toro [229] and the application of these techniques to discontinuous Galerkin methods in the books of Hesthaven and Warburton [109] and of Pietro and Ern [62].

Flux limiter methods split the numerical flux function into a low order part (like an upwind flux) that behaves nicely at discontinuities. An additional high order part that allows for a high order space discretization is added to the low order part and augmented by a special limiter function. This limiter deactivates the high order contribution close to discontinuities to avoid oscillations.

Slope limiters are closely related to flux limiters and aim to eliminate oscillation in the solution by acting directly on the slope of the solution. Similar to the flux limiters a special slope limiter function is defined that identifies discontinuities and limits the slope in their vicinity while it should not affect regions of smooth solutions.

The name of artificial viscosity models already indicates how these methods work. They introduce an artificial viscosity that is introduced to the PDE discretized. The viscosity will smear out the discontinuity slightly such that it obtains a small width that can be captured by the numerical method and dampens out nonphysical oscillations. In order to keep consistency with the initial PDE the applied viscosity should be small and only applied in regions with discontinuities. Moreover, the artificial viscosity ε_{av} has to depend on the mesh size Δx such that it vanishes under refinement, i.e. $\Delta x \to 0 \Rightarrow \varepsilon_{\text{av}} \to 0$. For discontinuous Galerkin methods the artificial viscosity should be chosen as $\varepsilon_{\text{av}} = \mathcal{O}(\frac{\Delta x}{P})$ as less viscosity is necessary due to the sub-cell resolution of these schemes [184].

In this thesis we do not consider flux nor slope limiters. These approaches usually require non-differentiable operations that make it hard to apply these methods when implicit solution techniques are used as in our case [2, 135]. Moreover, the limiting procedures require information from neighboring elements which interferes with the special structure of the HDG methods. In the HDG methods the coupling of neighboring element is expressed through the hybrid variable on element edges without accessing the solution on neighboring elements. This is required to obtain the special structure of the linearized system of equations that is amenable to static condensation, see Sec. 3.3.2. Therefore, limiters cannot be applied straight forward, if at all, to HDG methods.

Instead we focus on the application of an artificial viscosity model. We use the model introduced by Persson and Peraire [182, 184] which has been constructed especially for discontinuous Galerkin methods and uses the special structure of the solution representation by polynomials. The method works locally on each element and thus keeps the locality of the HDG method. Moreover, the artificial viscosity model is differentiable.

The shock width in the approximated solution usually scales with the mesh size Δx of the element where the shock is present. In order to get a sharp approximation of the shock it

is often beneficial to refine the mesh locally at the shock location. This can be an involved procedure that requires additional attention for time-dependent problems. At the shock the mesh should be refined, but the shock often moves through the domain for unsteady problems. Therefore, the mesh should be coarsened after a discontinuity has passed through an element and the solution is smooth again. Otherwise the shock will create an overly refined mesh over time that would increase the computational costs tremendously. We present a simple approach to adapt the mesh based on a fine triangular triangulation. From this fine level a coarse level will be created by agglomeration of neighboring elements.

This chapter is structured as follows: First we describe the artificial viscosity model and its application to the PDE in more detail. Afterwards we discuss the mesh adaptation approach using agglomerations of elements. We conclude the chapter by numerical results for a 1D shock tube problem where we investigate the effectiveness of the shock capturing method and its coupling to time adaptive and space adaptive simulations. For this, we extend the 1D problem in x_2-direction and solve it on a very simple 2D mesh. Thus, the approach to adapt the mesh is expected to work for more involved two dimensional test cases as well. More involved 2D test cases on adaptive meshes have not been included in this thesis due to the limited computational efficiency of the current implementation and is left for future work.

To the knowledge of the author of this thesis the shock capturing method has not been used for HDG methods before. Moreover, previous publication about HDG methods and applied to problems with shocks have been focusing solely on steady problems. Some parts of the results discussed in this section have been presented at conferences and publications [122, 127].

8.1. Application of an artificial viscosity model to the HDG method

The PDEs that we consider in this thesis have the shape

$$\partial_t w + \nabla \cdot ((f_c(w) - f_v(w, \nabla w)) = h(w, \nabla w) \qquad \text{(2.3 revisited)}$$

with possibly nonlinear functions f_c and f_v. We mentioned previously, see Ch. 2, that these equations may have discontinuous solutions even if the initial data is smooth. This is often the case for purely hyperbolic PDEs like the Euler equations, see Sec. 2.3. The solution of the Navier-Stokes equations is smooth as long as the convective effects do not become too strong. However, for large (local) Mach numbers $\mathrm{Ma} > 1$ the regularizing effect of the viscous terms is not strong enough such that different kinds of discontinuities develop.

In the case of shocks, the high order approximation of the discontinuous Galerkin methods (for $P > 0$) will create nonphysical oscillations. These oscillation can be reduced by the introduction of an additional viscous term $f_{v,\mathrm{av}}$ that smooths the discontinuity and damps overshoots in the solution.

In the following, we focus on purely hyperbolic problems, see Sec. 2.3, to keep the notation more compact. This has no impact on the application of such a stabilization term on convection-diffusion equations as the Navier-Stokes equations. As mentioned before the viscous effects of the Navier-Stokes equation may not be strong enough to ensure a continuous solution. In these cases artificial viscosity models can be applied. Common cases that need stabilization are flows with shocks or turbulent flows.

We express the (nonlinear) hyperbolic problem as

$$\partial_t w + \nabla \cdot f_c(w) = 0 \tag{2.4 revisited}$$

as before. These type of equations do not have any diffusive term to regularize the solution. In order to stabilize the problem the artificial viscous flux $f_{v,\mathrm{av}}$ is added to the equation

$$\partial_t w + \nabla \cdot (f_c(w) - f_{v,\mathrm{av}}(w, \nabla w; \varepsilon_{\mathrm{av}})) = 0. \tag{8.1}$$

It simply acts as an additional viscous flux. The artificial viscosity $\varepsilon_{\mathrm{av}}$ has to be determined such that it scales with the mesh size, i.e. $\varepsilon_{\mathrm{av}} = \mathcal{O}(\Delta x)$, is nonzero only in the vicinity of the shock and is large enough to sufficiently smooth the discontinuity. Additionally the artificial viscosity should scale with the polynomial degree P such that $\varepsilon_{\mathrm{av}} = \mathcal{O}(\frac{\Delta x}{P})$ holds and may depend on the solution w_h, its gradient ∇w_h and some user-defined parameters.

The artificial viscosity model that we use in this theses has been introduced by Persson and Peraire [184]. In order to detect discontinuities it takes advantage of the way the discontinuous Galerkin method represents the approximated solution. In Sec. 3.2, we introduced the solution representation on an element T_k as

$$w_h^k(t, x) = \sum_{i=1}^{N} W_i^k(t) \varphi_{h,i}^k(x), \quad x \in T_k, \tag{3.6 revisited}$$

with orthogonal basis functions $\varphi_{h,i}^k(x)$ (Legendre polynomials) and coefficients $W_i^k(t)$. The idea of the shock detector by Persson and Peraire is that this representation is similiar to the Fourier representation of the solution. In case of a discontinuous solution representation where shocks are present oscillation occur, see Fig. 8.1, most information is stored in the highest order "modes". These modes can represent the oscillations best and thus the coefficients related to this mode are dominating. In the setting of the discontinuous Galerkin method the modes relate to the basis functions and their polynomial degree. The highest order polynomial basis functions represent the highest order modes.

In order to define a high and low order/mode representation recall that the number of basis functions is denoted by

$$N(d, P) = \Pi_{i=1}^{d} \frac{(P + i)}{i} \tag{3.30 revisited}$$

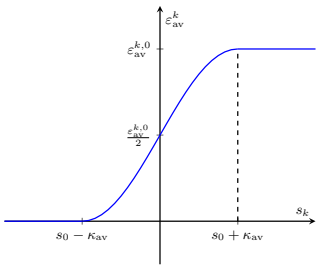


Figure 8.2.: Sketch of (8.3) to determine a the element-wise artificial viscosity $\varepsilon_{\text{av}}^k$ depending on the smoothness indicator s_k.

and depends on the space dimension d, in our case usually $d = 2$, and the polynomial degree P. A high mode representation of the solution is defined as

$$\hat{w}_h^k(t,x) = \sum_{i=N(2,P-1)+1}^{N(2,P)} W_i^k(t)\varphi_i^k(x),$$

where it is assumed that the basis functions are ordered according to their (total) polynomial degree. Thus, the basis functions $\varphi_i^k(x),\ i = N(2,P-1)+1, N(2,P-1)+2,\ldots,N(2,P)$ are the ones with (total) polynomial degree that is exactly P. The high mode solution $\hat{w}_h^k$ is the reconstructed solution that is based only on the highest order polynomial basis functions. Persson and Peraire define the smoothness indicator as

$$S^k := \frac{\left(\hat{w}_h^k, \hat{w}_h^k\right)_{L_2(T_k)}}{\left(w_h^k, w_h^k\right)_{L_2(T_k)}} \tag{8.2}$$

where $(\cdot,\cdot)_{L_2(T_k)}$ is the standard L_2-inner product on T_k. If the value of the smoothness indicator is large, most information is stored in high order modes and most likely has strong oscillations, which indicate a discontinuity. Otherwise the solution is assumed to be smooth. The smoothness indicator is easy to evaluate due to the hierarchical structure of the Legendre polynomials that are used as basis functions. This property is crucial to efficiently evaluate the smoothness indicator. The indicator has been found to be very reliable by the authors [184] and has been incorporated also into other shock capturing methods, see [16, 17, 133] for example.

Besides identifying regions with discontinuous solutions the value of the smoothness indicator (8.2) can be used to determine how much viscosity should be added. In regions of weak oscillations caused by a discontinuity only a small amount of viscosity is needed while more viscosity is needed at the shock location. Therefore, Persson and Peraire propose the

following (smooth) function

$$\varepsilon_{\text{av}}^{k} = \begin{cases} 0 & , \quad s_k < s_0 - \kappa_{\text{av}} \\ \frac{\varepsilon_{\text{av}}^{k,0}}{2}\left(1 + \sin\left(\frac{\pi(s_k - s_0)}{2\kappa_{\text{av}}}\right)\right) & , \quad s_0 - \kappa_{\text{av}} \leq s_k \leq s_0 + \kappa_{\text{av}} \\ \varepsilon_{\text{av}}^{k,0} & , \quad s_k > s_0 + \kappa_{\text{av}} \end{cases} . \tag{8.3}$$

with $s_k \coloneqq \log_{10} S^k$, see Fig. 8.2. User-defined parameters $\varepsilon_{\text{av}}^0$, $s_0 \sim \log P$ and κ_{av} have to be chosen such that the shock is accurately approximated [172, 184]. The bulk viscosity $\varepsilon_{\text{av}}^0$ determines the maximum amount of viscosity that can be added. The parameters s_0 and κ_{av} determine at what values of the smoothness indicator the viscosity should be added and how big the interval is where the applied viscosity is increased smoothly using a sinusoidal function. The effective bulk viscosity $\varepsilon_{\text{av}}^{k,0}$ in each element T_k is derived from the user-defined bulk viscosity $\varepsilon_{\text{av}}^0$ and is given by $\varepsilon_{\text{av}}^{k,0} = \varepsilon_{\text{av}}^0 \frac{\Delta x}{P}$ to ensure that the effective viscosity fulfills $\varepsilon_{\text{av}}^k = \mathcal{O}(\frac{\Delta x}{P})$. Note that the artificial viscosity has a constant value within each element.

We have not specified the quantity that should be used in the smoothness indicator (8.2). In the setting of a scalar convection-diffusion equation or Burgers' equation one would directly work with the unknown. The Euler and Navier-Stokes offer a variety of components in the vector of unknowns and additionally a physical interpretation of the flow. Common choices include quantities like the density ρ, entropy s, enthalpy H or derived quantities like the Mach number Ma that vary strongly at discontinuities. We choose the density ρ as indicator as this indicator is simple to employ and we found it to be a reliable indicator in the test cases we studied.

8.1.1. Choice of the stabilization term

The artificial viscosity model usually defines a way to detect discontinuities and how to compute the amount of artificial viscosity that should be added. It usually leaves different options open for the stabilization term that can be used. The stabilization term should introduce diffusive effects to smooth the solution and damp oscillations. A simple choice that works for scalar convection-diffusion, Burgers', but also Euler and Navier-Stokes equations is to choose the viscous flux to be

$$f_{v,\text{av}}(w, \nabla w; \varepsilon_{\text{av}}) \coloneqq \varepsilon_{\text{av}} \nabla w,$$

which is a Laplacian with artificial viscosity ε_{av}.

This term is added only in the vicinity of discontinuities to stabilize the solution. Thus, we do not expect this term to have a too big influence on the overall solution and we do not want this term to increase the amount of work too much. Therefore, we discretize this term

in an inconsistent way. We add

$$\int_{T_k} f_{v,\mathrm{av}} \cdot \nabla \varphi_h dx \approx \int_{T_k} \nabla \cdot f_{v,\mathrm{av}} \varphi_h dx$$

to (3.15b) and neglect the boundary integral terms. This term is used commonly in literature for stabilization [103, 115] and is the stabilization term we use in this thesis.

This is not the only way do add a (diffusive) stabilization term. It has been mentioned in Sec. 2.7 that the Euler equations represent the inviscid limit of the Navier-Stokes equations. Therefore, another obvious stabilization term would be the Navier-Stokes viscous flux (2.24). Due to the absence of viscous effects based on the flow physics in the Euler equations one would have to artificially define suitable parameters like the Prandtl number Pr and thus a heat conductivity.

The choice of the stabilization term and its influence on the stabilization and shock width has been discussed in the literature [95, 184] and the references in [73]. Persson and Peraire [184] found that the Navier-Stokes viscous flux would lead to a sharper shock representation than a Laplacian. Other authors argue against the physics based stabilization because it lacks the diffusive term for the mass conservation equation [133] and may lead to nonphysical solutions for the Euler equation [95]. Therefore, we use an (inconsistent) Laplacian for stabilization that also has a simpler structure than the viscous term of the Navier-Stokes equations.

8.1.2. Continuous reconstruction of the artificial viscosity

We have presented that problems with discontinuous solutions lead to oscillations in the approximated solution as shown in Fig. 8.1. Another problem that leads to spurious oscillations has been identified for artificial viscosity models if an element-wise constant viscosity is employed. The viscosity acts only locally near the discontinuity and thus may vary greatly between neighboring elements. This (large) discontinuity in the artificial viscosity might introduce spurious oscillations itself [16, 17]. As a remedy it has been proposed to use a (more) smoothly varying artificial viscosity [16, 17, 133, 182]. Barter and Darmofal introduced a PDE-based artificial viscosity that would be smooth, but introduce an additional PDE that has to be solved. Persson [182] and Klöckner et al. [133] introduced reconstructions to obtain a continuous artificial viscosity. Their investigations indicate that reconstructing the viscosity to have higher regularity such that it becomes differentiable does not offer additional advantages.

Whenever we use a continuous artificial viscosity we follow the C^0-reconstruction proposed by Persson [182]. First, the artificial viscosity is computed on every element. Afterwards, we compute at each vertex the maximum viscosity of its neighboring elements. The viscosity is then reconstructed linearly on each element using the viscosity values on the vertices.

The reconstruction procedure is non-local and would increase the stencil of the numerical method. Moreover, the determination of the maximum value on the vertices is a non-smooth operator that would pose a problem for the application of the Newton method. Thus, Persson also proposed to "weakly" couple the artificial viscosity model to the solver [182]. This means that the viscosity field is precomputed at the beginning of each time step and is held constant through the time step. By following this approach the artificial viscosity model is not part of the Jacobian needed for Newton's method.

8.2. Mesh adaptation through agglomeration

We have mentioned in the introduction that mesh adaptation is often employed to improve the approximation of discontinuities. Although the sub-cell resolution of DG methods allows to resolve discontinuities within an element and leading to shock widths smaller than the element size Δx the shock might still be rather wide if the element is large. Therefore, an adaptive mesh refinement is often used in this scenario [74, 155, 186].

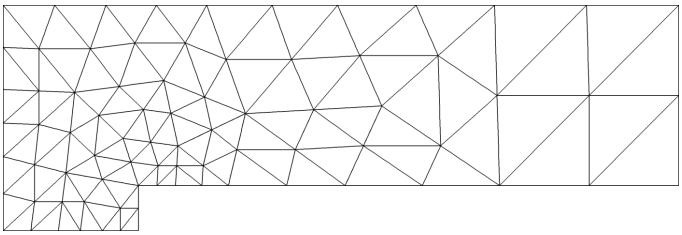
The adaptation procedure is especially involved in the case of time-dependent problems as it should not only allow refinement but also incorporate coarsening of the mesh. In Fig. 8.3 we show the situation of a simulation with refinement, but no coarsening after one eighth of simulation time. The super sonic flow enters a channel with a step from the left such that a shock develops. The shock starts from the step and subsequently grows and moves to the left [240]. Elements are refined if artificial viscosity applied to it and the element size is larger than $\Delta x > 0.04$.

In the Fig. 8.3 we present the initial mesh at $t = 0$ with $K = 120$ elements and the density, the applied artificial viscosity and the mesh at $t = 0.5$ with $K = 3654$ elements. One can clearly see how the mesh was refined, see Figs. 8.3a and 8.3d, due to the moving shock front. The mesh refinement was triggered by the artificial viscosity applied at the shock front and the forward facing step where a singularity is present, see Fig. 8.3c.

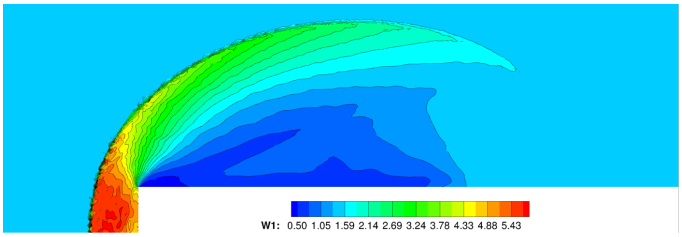
It is clear hat the number of elements increased greatly although the density solution, see Fig. 8.3b, does not incorporate that many features requiring such a high resolution. Nevertheless, many refined elements remain in the mesh that are not necessary to properly resolve the current structure of the solution and these elements could be removed.

The example shows that mesh refining and coarsening can be crucial to optimize the required computational work in time-dependent simulations. At the same time the refining and coarsening procedure should be cheap such that it does not nullify the run-time savings the additional computational work required by the mesh adaptation procedure.

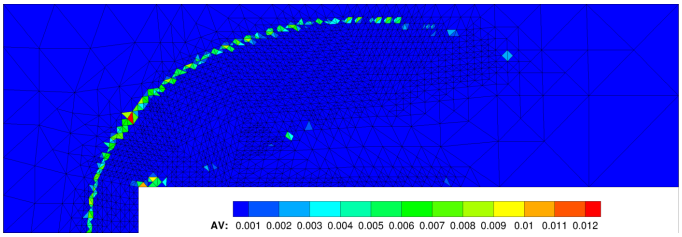
We investigate a mesh adaptation procedure that bases on the agglomeration of elements. The main idea is to generate a fine mesh consisting of triangular elements. These are connected to form arbitrarily shaped polygons to create the coarse level. The polygons then



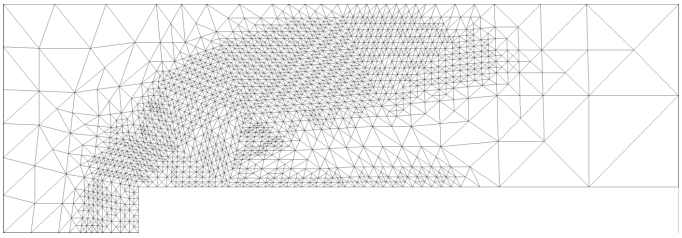
(a) Initial mesh at $t = 0$.



(b) Contour plot of the density ρ at $t = 0.5$.



(c) Plot denoting the amount of artificial viscosity applied to each element at $t = 0.5$.



(d) Refined mesh at $t = 0.5$.

Figure 8.3.: Plots showing the influence of mesh refinement without coarsening. The initial mesh and mesh, density ρ and artificial viscosity at $t = 0.5$ are shown. The HDG method has been used as space discretization.

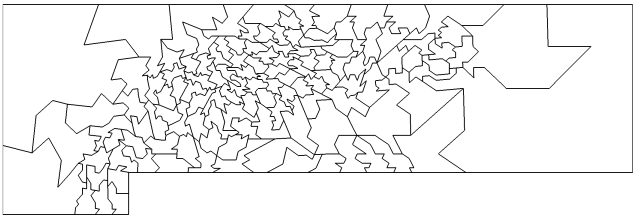


Figure 8.4.: A mesh that was constructed by agglomerating elements using METIS. Note the varying geometrical shapes.

can be split into smaller polygons until one eventually recovers the triangular elements on the finest level. This approach has the advantage that it is easy to introduce into an existing software framework and it also works nicely with meshing tools that allow to refine meshes, but do not allow for easy coarsening of the mesh. As we work on a fixed fine level mesh no active remeshing has to be done nor any hierarchy of meshes has to be stored. The repartitioning of polygons is comparably easy and cheap to do. However, this also means there is no anisotropic mesh refinement or similar to align with flow features. The polygonal elements require some extra work when a polygon is formed. In that case the basis functions are orthogonalized. The quadrature rules on the underlying triangular elements are used also for the polygon. This introduces way more quadrature points than needed, but ensures that the accuracy of the integration formula. This could obviously be optimized by computing tailored integration formulae for the newly formed elements, but this topic is out of the scope of this thesis. New integration formulae have been discussed in [157, 161], for example.

We implement the procedure in the following way. First a mesh is generated using Netgen that is deemed fine enough as the finest level for the approximation of the test case. Afterwards, a suitable set of partitions is computed using the graph partitioning tool METIS [131]. This results in a partitioning that minimizes the number of edges in the resulting polygonal mesh. The number of globally coupled unknowns of the HDG method depends on the number of edges in the triangulation. It is beneficial for the total number of globally coupled unknowns to have as few edges as possible. We define a maximum partition size $K^p \in \mathbb{N}$ that determines how many triangles are combined at most to define a polygon. Then, given the triangulation, partition size K^p and the connectivity graph of the mesh, METIS returns a map $i \mapsto m(i)$, $|m(i)| \leq K^p$. The map contains the element number of the elements T_k of the underlying triangular mesh. The coarse mesh $\mathcal{T}_h^p = \{T_k^p\}$ is then given by elements

$$T_k^p \coloneqq \bigcup_{i:m(i)=k} T_i.$$

If a polygon needs refinement and it consists of at least 4 triangles, METIS computes a

Figure 8.5.: A mesh with $K = 100$ elements as used for approximating Sod's shock tube problem.

new partitioning of the polygon. Otherwise it is split into the underlying triangles. If an element is flagged for coarsening we check for neighboring elements that have been flagged for coarsening, too, and merge then into a larger polygon. In Fig. 8.4, we show how the coarsest level of such a polygonal mesh could look like for the test case presented in Fig. 8.3. More information on discontinuous Galerkin methods on polygon meshes for different applications can be found e.g. in [18, 19, 37].

Remark 9 *Hybridized discontinuous methods are well-suited for the approach using arbitrary polygons as elements as part of the formulation is edge-based. In 2D, the edge shape does not depend on the shape of the element at hand. However, the effectiveness of the hybridization and static condensation process highly depends on the ratio of edges per element. Fewer edges per element induce a higher increase of globally coupled unknowns. Arbitrary polygons, however, can have an arbitrary number of edges and thus the hybridized discontinuous Galerkin method might in fact become less efficient than on standard elements like simplices or quadrilaterals. Similar behavior can be observed for unhybridized DG methods or other numerical schemes.*

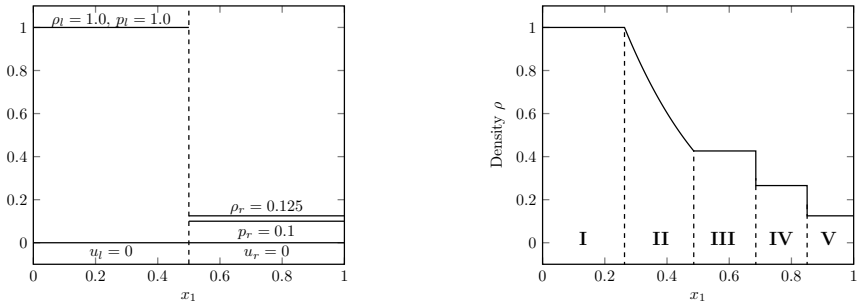
8.3. Numerical results

We present numerical results of a 1D test case to evaluate the shock capturing method and the time and space adaptation processes. Afterwards we discuss two different unsteady flows with shocks in 2D that create complex flow patterns.

8.3.1. Sod's shock tube

Sod's shock tube problem [218] is a one-dimensional Riemann problem on $\Omega = [0, 1]$. The initial flow condition is given by two constant states separated by a discontinuity. The HDG implementation is only capable of solving problems in two space dimensions. Therefore, we solve the problem on the two-dimensional domain $[0, 1] \times [0, \frac{2}{K}]$ with K being the number of elements of the mesh. An example of such a mesh is given in Fig. 8.5. This choice of domain leads to a mesh with minimal number of elements in x_2 direction. On the domain boundaries we apply slip wall boundary conditions.

In Fig. 8.6a we present the initial left w_l and right hand side states w_r. We set w_l for $x_1 < 0.5$ and w_r otherwise. In both regions the initial velocity is zero, but the pressure and density of the left state are much higher than of the right state. This is comparable to



(a) Initial data distribution.

(b) Final density distribution of the exact solution.

Figure 8.6.: Initial data and solution of Sod's shock tube problem.

the conditions in a shock tube where two constant flow states with zero flow velocity are separated by a membrane. The membrane is burst such that a supersonic flow condition develops.

The density distribution of the developing flow field at $t_{\text{final}} = 0.2$ is given in Fig. 8.6b. The fluid accelerates from the initial state (`I`) towards the right hand side through an expansion wave (`II`) until it reaches a region of constant density (`III`). This state transitions to another state of constant density (`IV`) through a contact discontinuity. The density and also the energy are discontinuous, but the velocity and pressure stay constant over this type of discontinuity. In order to reach the state in the fifth region, which is given by the initial right hand state, the flow has to undergo a shock where all quantities are discontinuous.

Justification of shock capturing method We consider the shock capturing method of Persson and Peraire [182, 184] with C^0-reconstruction (PePeC^0) and without C^0-reconstruction (PePe) for our simulations. This is not the only choice for the artificial viscosity model. There are other artificial viscosity methods by Nguyen and Peraire [164] (NgPe) and Moro et al. [155] (MoNgPe), for example. The authors have proposed and investigated these artificial viscosity models in the context of HDG methods. These artificial viscosity methods use the dilatation $\nabla \cdot u$ as indicating variable for shocks. In the vicinity of a shock the fluid velocity decreases rapidly over a small length which leads to negative dilatation with large absolute value.

We have found some minor issues with the dilatation based shock capturing methods with the model parameters supplied in their respective publications. The artificial viscosity should only be added in the vicinity of discontinuities, and other regions should not be affected by the artificial viscosity. We test this for the four artificial viscosity models on $\Omega = [0, 1] \times [0, 0.04]$ using the Euler equations. The mesh has $K = 50$ element and we use polynomials with

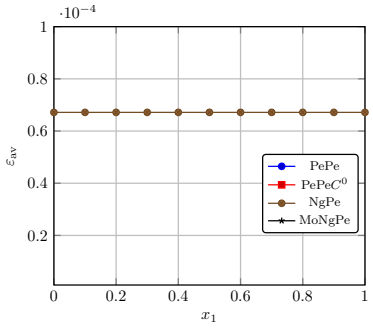
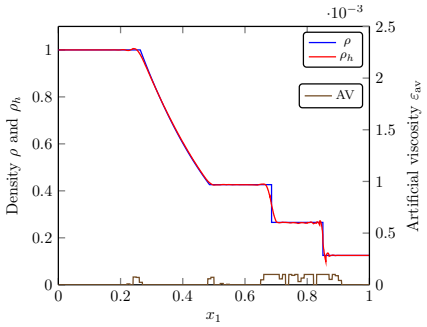


Figure 8.7.: Artificial viscosity of four different shock capuring methods when applied to the Euler equations with constant solution. Only the viscosity of the shock capturing model of Nguyen and Peraire is nonzero.

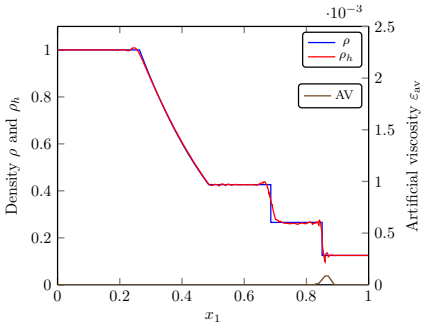
$P = 4$. The unknown w is initialized with $\rho = 1$ and $p = 1$ and all velocities being zero. Afterwards the shock capturing methods are applied and the resulting artificial viscosity computed. We extract the artificial viscosity from the (vertical) centerline, see Fig. 8.7. The method proposed by Nguyen and Peraire adds a constant amount of artificial viscosity to the problem. This undesired behavior has been addressed in the newer artificial viscosity model in [155].

In order to check the applicability of the shock capturing methods we also apply them to Sod's shock problem on a mesh with $K = 100$ elements, polynomials of degree $P = 2$ and time step size $\Delta t = 2 \cdot 10^{-4}$. The density ρ at final time $t_{\text{final}} = 0.2$ and the artificial viscosity that is applied at that time are presented in Fig. 8.8. Note the y-axis on the left hand side represents the scale of the density while the right hand side refers to the artificial viscosity. The scaling of the right hand side for the shock capturing by Nguyen and Peraire is two orders of magnitude smaller than for the other methods. There is no result reported for the shock capturing method of Moro et al. since the shock sensor creates an floating point exception that causes the simulation to crash for strong negative dilatation. This is unavoidable with the parameters stated in [155]. Nevertheless, it is expected to have large negative values for dilatation in real world applications as a strong deceleration is observed at shocks.

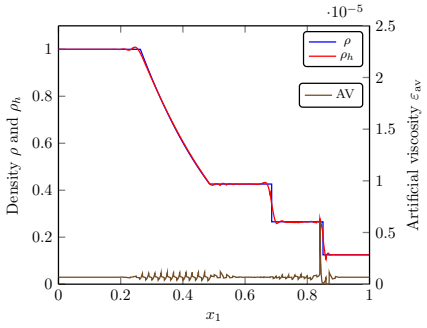
The results for the density are very similar to each other for all shock capturing methods, see Fig. 8.8. In all cases the numerical method is stabilized sufficiently and the discontinuities are smoothed. The contact discontinuity is smoothed a bit more than the shock for all schemes and there are small oscillations observable at the head and tail of the expansion wave. Larger overshoots are visible in the vicinity of the discontinuities. The overshoots near the contact discontinuity are least pronounced if we use the method of Persson and Peraire



Persson & Peraire.



Persson & Peraire with C^0-reconstruction.



Nguyen & Peraire.

Figure 8.8.: The density and artificial viscosity at final time $t_{\text{final}} = 0.2$ for Sod's shock tube. The problem is solved on a mesh with $K = 100$ elements, polynomials of degree $P = 2$ and time step size $\Delta t = 2 \cdot 10^{-4}$. The CFL-number is CFL ≈ 0.022. The results for different shock capturing methods are presented. The HDG method has been used as space discretization.

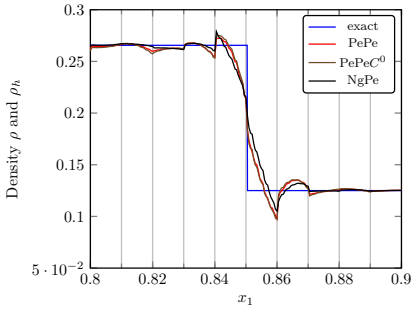


Figure 8.9.: Close-up of the shock of Sod's shock tube and the approxmiated solution. Different artificial viscosity models have been applied. The vertical lines indicate element boundaries. The problem is solved using the HDG method on a mesh with $K = 100$ elements, polynomials of degree $P = 2$ and time step size $\Delta t = 2 \cdot 10^{-4}$. The CFL-number is CFL ≈ 0.022.

without C^0-reconstruction. However, in this case the overshoots at the shock are a bit larger than for the method with C^0-reconstruction. The method of Nguyen and Peraire creates also some overshoots near the shock. In comparison to the other artificial viscosity model NgPe has the largest overshoot in front (left) of the shock. The other artificial viscosity models create the largest overshoot behind the shock, but the amplitudes of the overshoots of the methods still have the same magnitude. The method of Nguyen and Peraire and the method of Persson and Peraire with C^0-reconstruction also create very small oscillations between the expansion wave and the contact discontinuity that are much less pronounced in Persson's and Peraire's method if no C^0-reconstruction is used.

A close up of the solution around the shock is presented in Fig. 8.9. The vertical lines indicate element boundaries. One can see that the shock is captured within the elements next to the shock position and leads to some small oscillations in the neighborhood. All shock capturing methods lead to similar shock profiles. The method of Nguyen and Peraire has a slightly higher overshoot in front of the shock than the other methods which show a larger overshoot behind the shock.

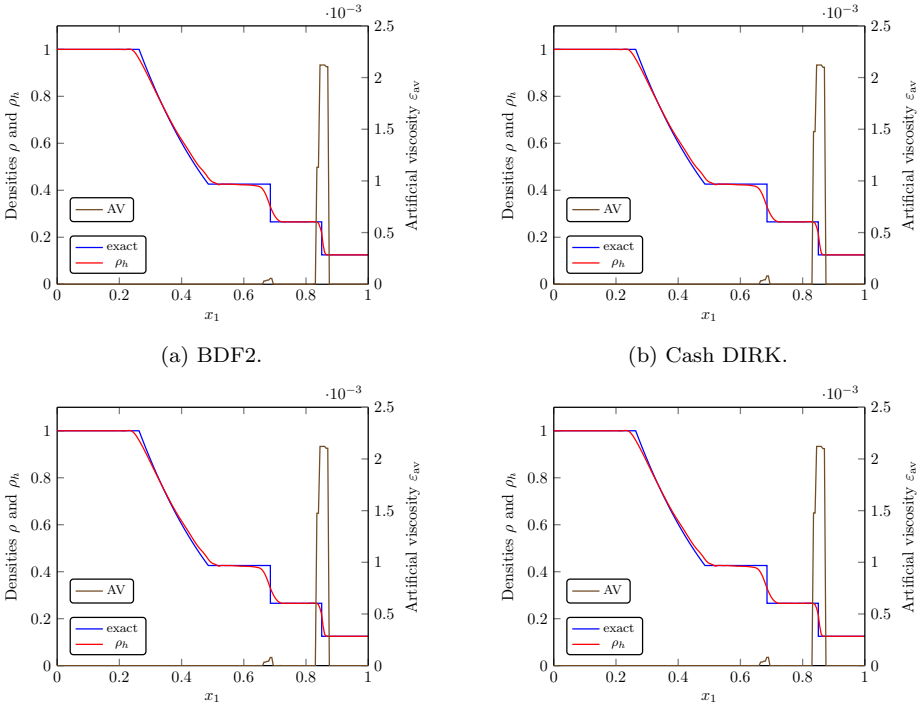
The artificial viscosity added at t_{final} and also the locations where the viscosity is added, differ considerably between the three methods, see Fig. 8.8. Nguyen's and Peraire's method adds significantly less artificial viscosity. The amount of viscosity is about two orders of magnitude less than for the other methods. However, in the figure the viscosity at the final time is presented so the figure does not contain information about the total amount of viscosity that has been applied to the problem over the whole duration of the simulation. The artificial viscosity model might have applied more viscosity in the beginning of the simulation while

the other methods might have done the opposite. For Nguyen's and Peraire's method we also observe an oscillating artificial viscosity, especially along the expansion wave and the region between the contact discontinuity and the shock indicating that the solution is oscillating in this region. Moreover, the method tends to add viscosity even at all smooth parts of the solution which is less pronounced in regions with positive dilatation. Persson's and Peraire's and artificial viscosity model shows similar behavior with and without C^0-reconstruction. When the viscosity is reconstructed the model adds viscosity only in the vicinity of the shock. Additionally viscosity at the head and tail of the expansion wave, in the region between the contact discontinuity, and the shock is added when the C^0-reconstruction is not used. The behavior to add viscosity at several locations is similar to the behavior the method of Nguyen and Peraire.

The results obtained in this section guided our choice of the shock capturing method. The smoothness indicator (8.2) is reliable when applied to the density. We do not need to tune any parameters in order to avoid floating point exceptions. The choice of the parameters s_0 and κ_{av} is still crucial to add sufficient viscosity to shock regions. At the same time the smoothness indicator decays quickly enough such that usually no viscosity is added to smooth flow regions.

Standard time integrators We investigate the influence of different time integrators on the shock approximation. In Fig. 8.10 we present results for a mesh containing $K = 100$ elements and polynomials of degree $P = 2$. The applied time integrators are the BDF2 and three different DIRK methods. All DIRK methods are suitable for a time-adaptive simulation through an integrated lower order DIRK method for error estimation, see Sec. 4.3. We use the third order method with $s = 3$ stages of Cash [41] that extends the DIRK method of Alexander [4] by a lower order error estimator. We refer to it as "Cash" in our plots. Additionally we use the fourth order $s = 4$ stage method of Al-Rabeh [191] and the fourth order $s = 5$ stage method of Hairer and Wanner [101]. For the fourth order methods we use the shorthand notations "AR" and "HW" in the plots. We use a constant time step size $\Delta t = 5 \cdot 10^{-4}$ for all methods. The parameters of the shock capturing method are set to $\varepsilon_{\text{av}}^0 = 0.45$, $s_0 = -14 \log(P)$ and $\kappa_{\text{av}} = 0.4$. The artificial viscosity ε_{av} is recomputed after each Newton step based on the updated solution. No reconstruction is applied, thus the artificial viscosity is constant element-wise. The data has been extracted from the center line at 200 equidistant points.

In Fig. 8.10 we present the density ρ and artificial viscosity ε_{av} at $t_{\text{final}} = 0.2$. The results indicate that for the given parameter settings the results barely depend on the accuracy of the time integrator. The applied artificial viscosity smooths the solution such that overshoots are barely visible. Such smoothing can be observed at the head and tail of the expansion wave, at the contact discontinuity and at the shock. Most viscosity is applied in the vicinity of the



(a) BDF2.

(b) Cash DIRK.

(c) Al-Rabeh DIRK.

(d) Hairer and Wanner DIRK.

Figure 8.10.: Sod's shock tube: Comparison of the final density with the exact solution for different time integration methods. The problem is solved using the HDG method on a mesh with $K = 100$ elements, polynomials of degree $P = 2$ and time step size $\Delta t = 5 \cdot 10^{-4}$. The CFL-number is CFL ≈ 0.055.

shock and a much smaller amount is applied at the contact discontinuity. The difference in amount of viscosity applied might stem from the fact that the contact discontinuity has been smoothed much more than the shock. Thus, the shock profile is quite sharp, especially when compared with the contact discontinuity.

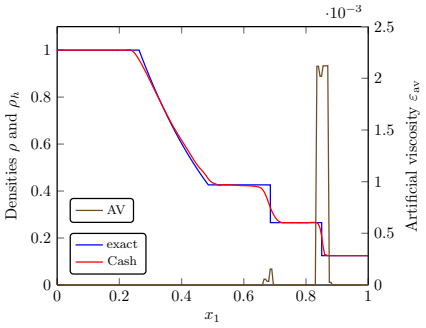
The results indicate that all time integrators are adequate to discretize the problem. The results are very similar so the choice of high order time integrator does not show an immediate advantage over low order time integrators. The employed shock capturing method stabilizes the discretization sufficiently and leads to a good approximation of the shock and only small overshoots.

Time step adaptation We have seen in the previous part that the accuracy of the time integrator barely affects the solution. We have mentioned before that the DIRK methods of Cash, Al-Rabeh, and Hairer and Wanner allow for an easy time step adaptation as described in Sec. 4.3. In [119] we have investigated this time step adaptation for smooth flow problems. In this section, we investigate the behavior of the time step adaptation and its influence on the solution for Sod's shock problem.

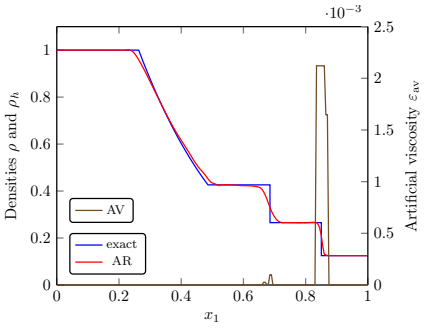
In our experiments we set the tolerance to $\text{tol} = 10^{-1}$ and set the lower and upper bounds, $\Delta t_{\min}$ and $\Delta t_{\max}$, for the time step sizes such that $10^{-4} \leq \Delta t^n \leq 5 \cdot 10^{-3}$ holds for all time step sizes. The simulation is started with $\Delta t^0 = \Delta t_{\min}$. All other parameters of the simulation are identical to the ones of the previous simulation with fixed time step size.

In Fig. 8.11h we present the density and artificial viscosity at final time $t_{\text{final}} = 0.2$ and additionally the time step evolution. The numerical results for the density are very similar to the one for the fixed time step size. Only small overshoots can be observed and the contact discontinuity is smoothed strongly. The shock, however, is approximated much sharper than the contact discontinuity. In contrast to the fixed time step simulation the artificial viscosity profiles are slightly different between the different time integrators, but roughly the same amount of viscosity is added near the shock for all time integrators. Much less viscosity is added at the contact discontinuity and the shape of the viscosity profiles at this position are very similar for Cash's and Al-Rabeh's DIRK method. The profiles mostly differ in the amount of viscosity added which is largest for Cash's DIRK method. For Hairer's and Wanner's method no viscosity is added at the contact discontinuity at final time $t_{\text{final}} = 0.2$.

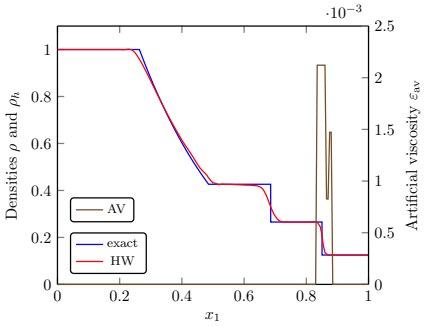
The evolution of the time step size, see Fig. 8.11d, shows that the time integrators show all qualitatively the same behavior. The time step sizes are increased for some period and then abruptly decreased again. The abrupt decrease of time steps is a result of time steps being rejected. After a longer time span of 0.03 the abrupt decrease of the time step size repeats after smaller time spans of roughly 0.01 until the end of the simulation and at the same times for each time integrator. It is noticeable that the time step bounds, $\Delta t_{\min}$ and $\Delta t_{\max}$, are never hit except for the first time step where we set $\Delta t^0 = \Delta t_{\min}$ and the last



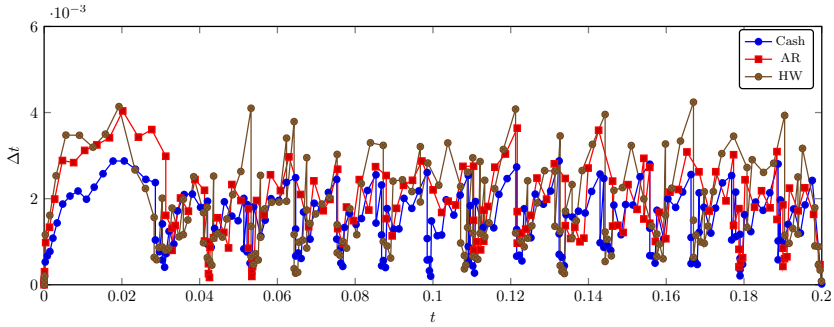
(a) Cash DIRK.



(b) Al-Rabeh DIRK.



(c) Hairer and Wanner DIRK.



(d) Evolution of the time step size Δt^n.

Figure 8.11.: Sod's shock tube: Comparison of the final density with the exact solution for different time integration methods and artificial viscosity at t_{final}. Adaptive time stepping has been used and the evolution of the time step size is plotted. The solution has been computed on a mesh with $K = 100$ elements and polynomials of degree $P = 2$. The largest possible CFL-number is CFL ≈ 0.548. The HDG method has been used as space discretization.

	Cash	AR	HW
Time steps accepted	149	116	133
Time steps rejected	37	25	43
Time steps computed	186	141	176
Total Newton steps	14261	15110	22659

Table 8.1.: Number of computed time steps, accepted time steps and total Newton steps for Sod's shock tube test case with adaptive time step adaptation. The HDG method has been used as space discretization.

time step where the time step is set to $\Delta t^n = t_{\text{final}} - t^n$. The method of Hairer and Wanner tends to increase the time step furthest while the method of Cash tends to use the smallest time step sizes of the three methods compared.

In Tab. 8.1 we give an overview over the number of computed time steps, the number of accepted time steps and the total number of Newton steps during the simulation. The table shows that the methods behave quite differently. The lowest order method, the method of Cash, computes the most time steps and roughly 80% of the time steps have been accepted. The high number of time steps computed fits into the image of the smallest time steps chosen with this time integrator. Al-Rabeh's DIRK methods computes much less time steps and has a slightly higher time step acceptance rate of roughly 82%. The method of Hairer and Wanner has slightly less computed time steps than Cash's method, but rejects the most time steps leading to an acceptance rate of around 75%.

We also compare the number of Newton steps as they are an indicator of the total computational costs. The Newton steps can be considered the most expensive part as they involve constructing the globally coupled system of equations and solving the local solves. Therefore, the number of Newton steps as a measure of costs is a reasonable indicator. However, the number of Newton steps alone does not account for the real computational time which is also affected by the speed of convergence of the employed linear solver and by the suitability of the integrator for efficient implementation. We observe that Cash's method is least expensive and Al-Rabeh's method is slightly more expensive although fewer time steps have been computed and rejected. Hairer's and Wanner's method needs by far the most Newton steps. The number of needed Newton steps is affected by the different number of stages of Cash's ($s = 3$), Al-Rabeh's ($s = 4$) and Hairer's and Wanner's ($s = 5$) in each time step and how fast the Newton solver converges in each of the time steps. Based on these observations the lowest order time integrator is most efficient as there is no obvious difference in the solutions of the different time integrators.

The large number of rejected time steps increases the costs of the time integrators tremen-

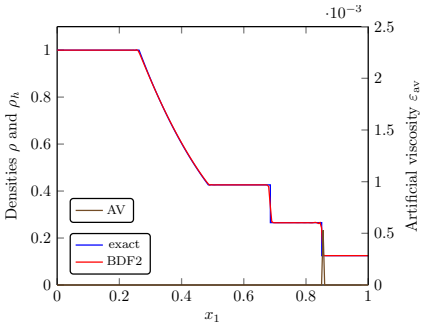
dously. A rather non-smooth variation of time step sizes has also been observed by other authors when applying adaptive time step control to test cases with shocks [133]. In order to decrease the number of time steps one could try different time step adaptation strategies. A proportional-integral-derivative (PID) controller based on the ideas of control theory could be a possible remedy [101]. The controller uses a history of the estimated error to lead to a much smoother variation of time step sizes Δt^n.

Again, the differences in the solution of the different DIRK methods are minimal (Figure 8.11a – 8.11c). In contrast to the previous case with fixed time step sizes a greater difference in the location and amount of viscosity added can be observed between the methods. For Cash's and Al-Rabeh's methods artificial viscosity is added near the contact discontinuity, but not for Hairer's and Wanner's method. Nonetheless, the greater variation of artificial viscosity added does not seem to have a visible effect on the obtained solutions by the different time integrators which still are in good agreement with each other.

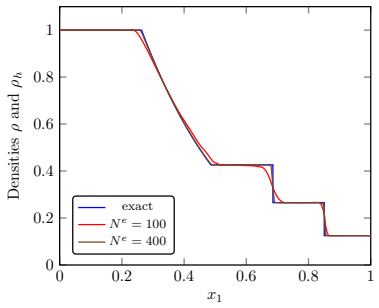
The evolution of the time step sizes in Figure 8.11d are fairly similar for all DIRK methods. Hairer's and Wanner's method tends to use the largest time steps and Cash's method tends to use the smallest time steps. This is most likely caused by the order of accuracy of the methods, as Hairer's and Wanner's method is of fourth order consistent in time while Cash's method is of third order consistent in time. Nevertheless, all methods struggle at the same points during the simulation where the time step has to be reduced. For all methods this leads to rejected time steps.

Mesh adaptation The solution of Sod's shock tube, see Sec. 8.3.1 and especially Fig. 8.6, can be represented by polynomials nicely. In most regions the solution is constant or varies smoothly as for the expansion wave. The only critical points that require a higher resolution are the discontinuities and the head and tail of the expansion wave as there is no smooth transition from the constant states to the expansion wave; i.e. the solution is not differentiable at these points.

In Fig. 8.12 we present the solution of Sod's shock tube computed with BDF2 time integration and the same simulation parameters as before, but on a finer mesh with $K = 400$ elements at final time $t_{\text{final}} = 0.2$. Moreover, we compare the solution against the solution on the coarser mesh as presented in Fig. 8.10a. The approximation of the solution is very good. The head and tail of the expansion wave are approximated nicely and the width of the shock and contact discontinuity approximation is very small. The amount of artificial viscosity is much lower than on the coarse mesh, see Fig. 8.10a, which is related to the fact that the effective viscosity is scaled with $\frac{\Delta x}{P}$. Moreover, the viscosity is only added at the shock location, but not at the contact discontinuity. As stated in the previous paragraph, the solution mostly improves at the discontinuities and the head and tail of the expansion wave on the fine mesh. The approximation is much closer to the exact solution. Thus, it would be



(a) Final density distribution using BDF2 on a mesh with $K = 400$ elements.



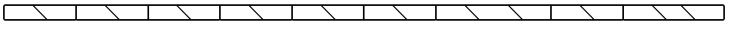
(b) Comparison of the final density distribution using BDF2 for time integration on meshes with $K = 100$ and $K = 400$ elements and the exact solution.

Figure 8.12.: Solution of Sod's shock tube computed with BDF2 time integration and $K = 400$ elements. The solution is compared against the solution with $K = 100$ elements from Fig. 8.10a. The problem is solved using the HDG method with polynomials of degree $P = 2$ and time step size $\Delta t = 2 \cdot 10^{-4}$. The CFL-number is CFL ≈ 0.219.

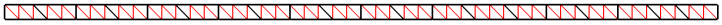
beneficial to have a mesh that is refined at the critical points of the solution where a higher polynomial degree alone would not improve the approximation. At the same time the mesh could be coarser at regions with constant or smooth solution which can be nicely represented by polynomials. The solution changes over time, hence the mesh should do allow so as well.

In order to respect the time-dependent solution and to adapt the mesh accordingly we employ the mesh adaptation strategy described in Sec. 8.2. The time step size is set to $\Delta t = 5 \cdot 10^{-4}$ and BDF2 is used for time integration. The shock capturing parameters are $\kappa_{\text{av}} = 0.4$, $\varepsilon_{\text{av}}^0 = 0.3$ and $s_0 = 3.75$. The polynomial degree is again $P = 2$. The underlying mesh has 100 elements and we investigate the two cases with $K^p = 5$, see Fig. 8.13, and $K^p = 10$, see Fig. 8.14. Thus, a polygon may consist of up to 5 and 10 triangles, respectively. The resulting mesh at the beginning of the simulation at $t = 0$ is shown in Figs. 8.13a and 8.14a. Additionally the coarse mesh with the underlying triangles indicated can be found in Figs. 8.13b and 8.14b.

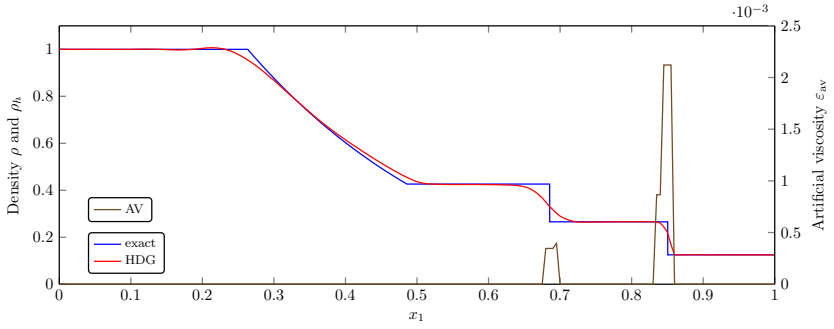
The solutions in both cases, see Figs. 8.13c and 8.14c, are in good agreement with each other. The shock is approximated relatively sharp while the contact discontinuity is smoothed much more. There are some small overshots at the end of the expansion wave. Both cases also show very similar distribution of artificial viscosity. The good agreement of the solutions is also recognizable in the adapted meshes at $t_{\text{final}} = 0.2$, see Figs. 8.13d and 8.14d. The



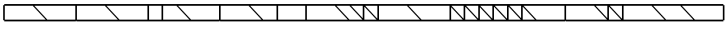
(a) The mesh at $t = 0.0$.



(b) The polygon mesh (black) at $t = 0.0$ with underlying triangular mesh (red).

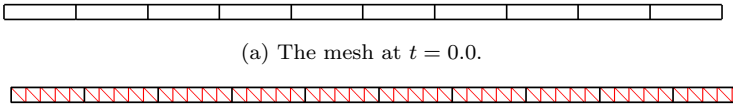


(c) Solution on polygon mesh at $t = 0.2$.



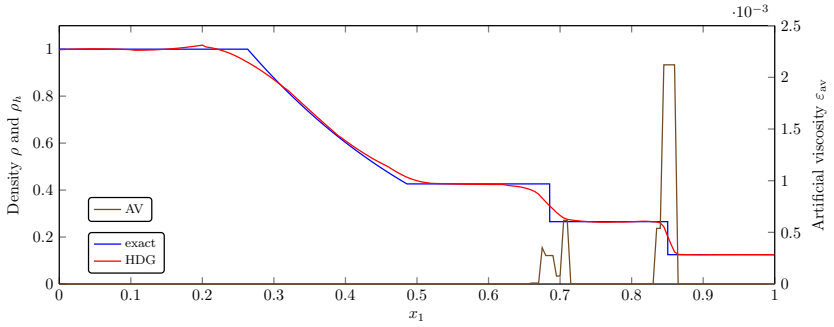
(d) The mesh at $t = 0.2$.

Figure 8.13.: Sod's shock tube: The problem is solved using the HDG method on an adaptive polygonal mesh with $K^p = 5$ and $K = 100$ elements, polynomials of degree $P = 2$ and time step size $\Delta t = 5 \cdot 10^{-4}$. The largest possible CFL-number is CFL ≈ 0.055.

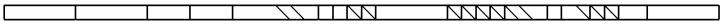


(a) The mesh at $t = 0.0$.

(b) The polygon mesh (black) at $t = 0.0$ with underlying triangular mesh (red).

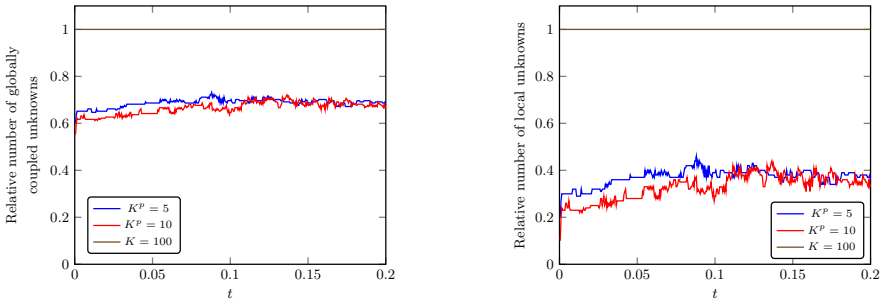


(c) Solution on polygon mesh at $t = 0.2$.



(d) The mesh at $t = 0.2$.

Figure 8.14.: The problem is solved using the HDG method on an adaptive polygonal mesh with $K^p = 10$ and $K = 100$ elements, polynomials of degree $P = 2$ and time step size $\Delta t = 5 \cdot 10^{-4}$. The largest possible CFL-number is CFL ≈ 0.055.



(a) Evolution of the relative total number of globally coupled degrees of freedom over time.

(b) Evolution of the relative total number of local degrees of freedom over time.

Figure 8.15.: Evolution of globally coupled and local unknowns over time for Sod's shock tube on an adaptive mesh. The HDG method has been used as space discretization.

meshes are slightly refined at the head and tail of the expansion wave and strongly refined at the discontinuities. The solution in both cases is also very similar to the results on fixed meshes from previous sections. The good agreement of the solutions on the adaptive meshes with the solution on a fixed mesh also underlines that the higher mesh resolution is in fact not necessary at all parts of the domain, but only where larger changes in the solution can be observed.

In Fig. 8.15 we present the evolution of the degrees of freedom, globally coupled and local, over time. The evolution shows that the number of unknowns is almost identical for both partition sizes, especially during the second half of the simulation. The number of unknowns even becomes larger for some time steps when the larger partition size $K^p = 10$ is used. This means that the larger allowable partition size $K^p = 10$ does not automatically lead to a coarser mesh and thus lower number of unknowns. Nevertheless, both partition sizes cause a reduced number of globally coupled unknowns of around 33% during the simulation and almost 66% reduced number of local unknowns when compared to the mesh with $K = 100$ triangular elements. This also shows two potential limitations of the used fine mesh and the polygonal elements when used with the HDG method.

The fine mesh has a large number of edges $\overline{K} = 201$ compared to the number of elements $K = 100$ since it has only two elements in x_2-direction. Thus there is a large number of hybrid unknowns that is rather untypical compared to meshes for real world applications, see Sec. 3.3.3. The other limitation is the increased number of edges for arbitrary polygons. In Sec. 3.3.3 and Rem. 9, it has been mentioned that the effectiveness of the HDG method in regards of decreasing the globally coupled unknowns depends on the number of edges per element. This effectiveness decreases for increasing number of edges. We have not put any

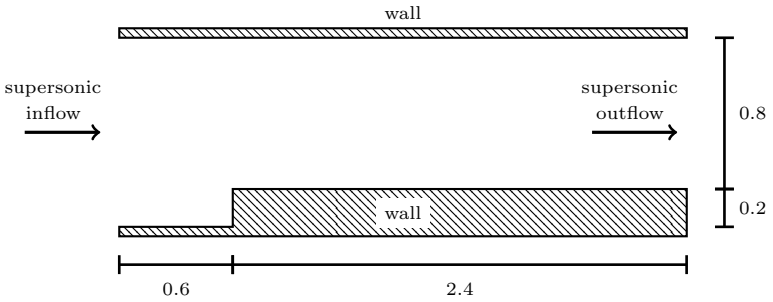
restriction on the shape of the polygons as this would also increase the complexity of the adaptation process. Nevertheless, the HDG method would still lead to a lower number of globally coupled unknowns than a standard DG method for sufficiently large polynomial degrees. The number of globally coupled unknowns per element increases with $\mathcal{O}(KP^2)$ for standard DG discretizations while it grows as $\mathcal{O}(\overline{K}P)$ if one considers the case with $d = 2$ space dimensions. The polynomial degree that has to be used in order to lead to less globally coupled unknowns for the HDG method depends strongly on the ratio of elements to edges in the mesh, see Sec. 3.3.3. The same holds for the resulting number of nonzero entries of the globally coupled system of equations.

8.3.2. Mach 3 flow over a forward facing step

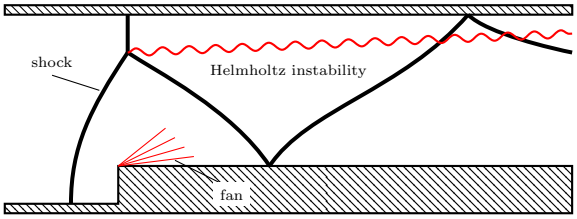
This 2D test case is based on the work of Emery [71] and describes a supersonic flow in a channel with a forward facing step. This test case has been used for the evaluation of several numerical schemes and has been recognized especially after the publication of Woodward and Colella [240] who used the test case for the comparison of different shock capturing methods.

In Fig. 8.16a we sketch the physical domain. The dimensions are normalized such that the height L of the channel inlet is $L = 1$. The channel is three times as long as it is high and has a step at $0.6L$ of $0.2L$ units. A supersonic, inviscid flow is entering the channel from the left and stays supersonic over the whole domain. On the top and bottom, walls are restricting the domain. As the flow is assumed to be inviscid it is described by the Euler equations, see Sec. 2.6. The simulation reaches a steady state solution after a certain time. However, for the evaluation of shock capturing methods in the unsteady case the solution at $t_{\text{final}} = 4$ is computed since it has a richer structure than the steady state solution. The flow in the channel is initialized with the flow conditions at the inlet. At the inlet the flow enters the domain with Mach number $\text{Ma} = 3$ and no velocity component in x_2-direction. An intermediate solution at $t = 0.5$ of this test case has been presented in Fig. 8.3.

In Fig. 8.16b we denote the expected solution structure of the density ρ at final time $t_{\text{final}} = 4$. The supersonic flow hits the step which leads to a bow shock in front of the step that is reflected by the top wall leading to a lambda shaped shock. The right leg of the lambda shock is reflected two more times at the walls until it leaves the domain. If the resolution of the numerical discretization is high enough a Helmholtz instability can be observed. The instability originates at the lambda shaped shock and travels with the flow out of the domain. Moreover, a rarefaction fan is created at the tip of the step. The rarefaction fan may create further instabilities depending on the resolution. The tip of the step might cause problems for the numerical method as it is a singular point. Woodward and Colella [240] suggested to correct the flow state by copying the flow state of neighboring cells of the discretization onto the cells at the tip. Other authors replace the sharp edge of the step by



(a) The physical domain of the Mach 3 step problem with boundary coniditions.



(b) Schematic of the solution of the density ρ for the Mach flow problem over a forward facing step at $t_{\text{final}} = 4$.

Figure 8.16.: A sketch of the physical domain of the Mach 3 step test case and a schematic of the expected structure of the solution at $t_{\text{final}} = 4$.

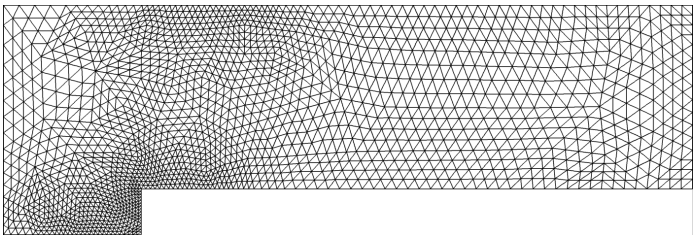
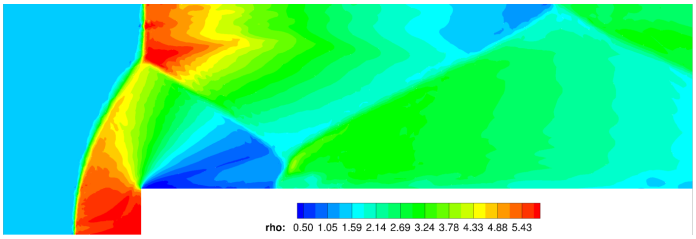


Figure 8.17.: The mesh used for the Mach 3 step test case. It has $K = 3552$ elements.

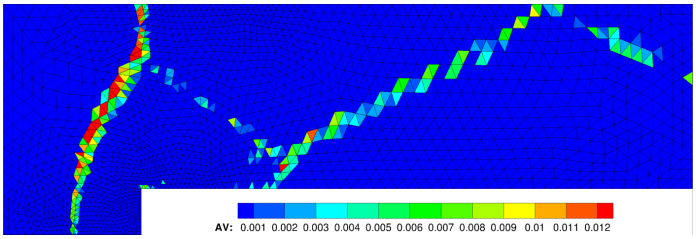
a rounded one with very small diameter and justify this by a completely sharp edge being unphysical [155]. We follow the approach of Persson [182] who did not apply any corrections, but rather let the shock capturing method handle the singular point.

In order to discretize the problem we use slipwall boundary conditions for the walls and supersonic boundary conditions at the inflow and outflow. We use a mesh with $K = 3552$ elements that is fixed in time, see Fig. 8.17. In Fig. 8.18, we present the density ρ_h and artificial viscosity $\varepsilon_{\text{av}}^k$ distribution. Polynomials of degree $P = 2$ have been used with BDF2 time stepping and fixed time step size $\Delta t = 2 \cdot 10^{-4}$. The parameters of the shock capturing method are set to $\varepsilon_{\text{av}}^0 = 0.75$, $\kappa_{\text{av}} = 0.25$ and $s_0 = 2.5$. Even with the rather low spatial resolution the pattern of the solution denoted in Fig. 8.16b is clearly visible. The bow shock has developed in front of the step and is reflected by the top wall to create a lambda shock. The shock is then reflected by the bottom and top wall. The shock itself is resolved within a few cells and is thus sharp regarding the used mesh size and polynomial degree. The bow shock is significantly wider than the remaining parts of the shock downstream because the shock is very strong and the mesh is coarse compared to the other regions of the domain. The rarefaction fan at the tip of the step is slightly visible. The instability developing from the lambda shock is not visible, but the behavior of the density behind the lambda shock indicates the presence of an instability that is not fully resolved. The artificial viscosity method adds viscosity only in the vicinity of the shock. In smooth regions no viscosity is added. Additionally, the simulation is stabilized by viscosity being added at and near the singular point. At these locations less viscosity is added than at the bow shock.

In Fig. 8.19 we present the final density solution on the same mesh with slightly varied parameters. We use polynomials of degree $P = 4$ and a DIRK22 time stepper with $\Delta t = 10^{-4}$. The parameters of the shock capturing are set to $\varepsilon_{\text{av}}^0 = 0.75$, $\kappa_{\text{av}} = 0.1$ and $s_0 = 2$. We additionally plot isolines of the density to improve the visibility of the shocks. Even though the mesh is the same as before the solution improves. The bow shock is approximated sharper than in the $P = 2$ case. Moreover, the influence of the instabilities developing from the lambda shock and the rarefaction fan are more obvious although still not completely resolved.



(a) Density ρ_h.



(b) Cell-wise artificial viscosity $\varepsilon_{\text{av}}^k$.

Figure 8.18.: The density ρ_h and element-wise artificial viscosity $\varepsilon_{\text{av}}^k$. The problem has been solved using the HDG method on a mesh with $K = 3552$ elements and polynomials of degree $P = 2$ up to $t_{\text{final}} = 4$. The CFL-number based on the inflow boundary conditions is CFL ≈ 0.064.

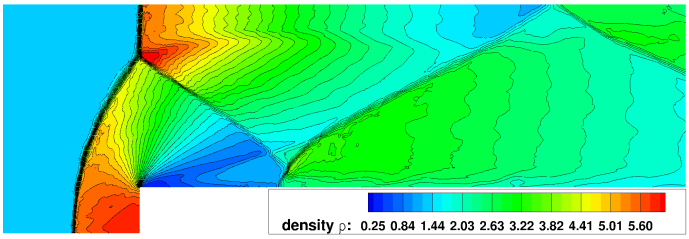
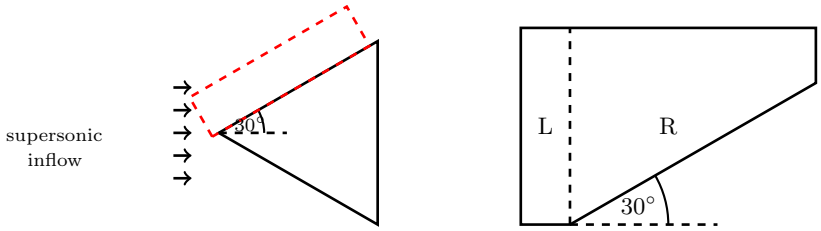
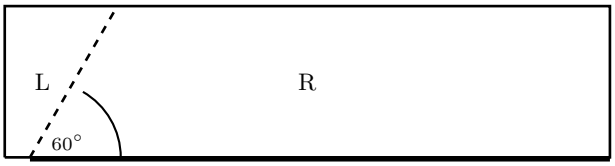


Figure 8.19.: Plot of the final density ρ_h distribution of the Mach 3 step test case. Polynomials of degree $P = 4$ and DIRK22 time stepping have been used. The problem has been solved using the HDG method on a mesh with $K = 3552$ elements up to $t_{\text{final}} = 4$. The CFL-number based on the inflow boundary conditions is CFL ≈ 0.032.



(a) The physical setting of the double Mach wedge test case. A supersonic flow coming from the left hits a wedge.

(b) "Physical domain" setting of the double Mach wedge case.



(c) "Rectangular domain" setting of the double Mach wedge case.

Figure 8.20.: The physical domain that motivates the double Mach wedge test case and the two domains used for numerical simulations. The capital letters indicate the parts of the domain where the left (L) and the right (R) hand side state is used. The dashed line indicates the border in between the initial states.

8.3.3. Double Mach wedge

The second 2D test case we study is the double Mach wedge problem that has been introduced by Woodward and Colella [240]. A supersonic flow hits a symmetric wedge that leads to a non-normal reflected shock which creates a complex flow pattern. In Fig. 8.20a we sketch the problem setting. The flow is coming from the left when it hits the wedge. The angle between the x_1-axis and the sides of the wedge is 30°. Due to the symmetry of the problem one usually focuses only on the upper part of the wedge.

We use two different domains for the approximation of the test case. One is close to the physical domain, see Fig. 8.20b, which we refer to as the "physical domain" in our discussion. The other one is a rectangular $[0, 4] \times [0, 1]$ excerpt of the mesh, see Fig. 8.20c, which we refer to as the "rectangular domain" and is indicated in Fig. 8.20a by the red dashed rectangle. Both domains contain the same interesting flow features. Focusing on the rectangular domain decreases the size of the computational domain and thus the size of the mesh needed at the cost of more complex boundary conditions at the upper boundary.

Both domains are initialized with two different flow states; the left (L) and right (R) hand side state. The simulation is run until $t_{\text{final}} = 0.2$, at which the solution has a structure as

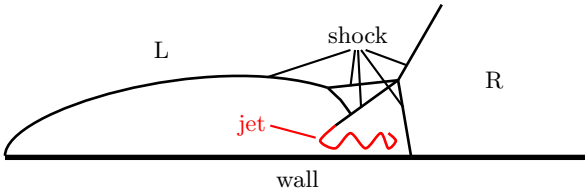


Figure 8.21.: Schematic of the structure of the solution of the double Mach wedge at $t_{\text{final}} = 0.2$.

indicated in Fig. 8.21. The shock given by the initial conditions travels through the domain and a bow shock develops from the tip of the wedge (left). When the bow shock approaches the traveling shock a complex shock interaction occurs, leading to several shocks and a jet. A Helmholtz instability can be observed in the jet if the resolution is high enough.

Physical domain The simulation on the physical domain is initialized with the free flow conditions

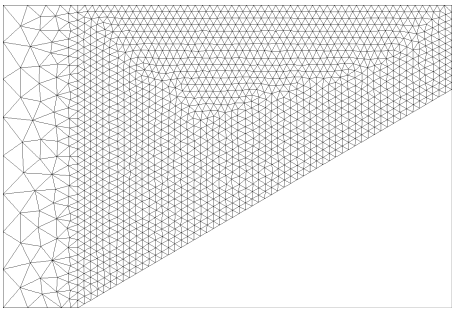
$$\rho_l = 8, u_{1,l} = 8.25, u_{1,l} = 0, p_l = 116.5$$

as left hand state (L) and the post shock conditions

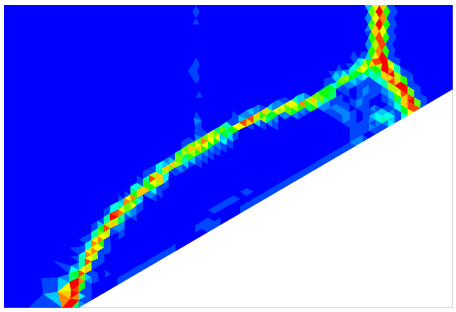
$$\rho_r = 1.4, u_{1,r} = 0, u_{1,r} = 0, p_r = 1$$

a right hand state (R). On the left hand boundary we use supersonic inflow boundary conditions and slip-wall boundary conditions everywhere else. On the top boundary symmetry boundary conditions are used. For the simulation we use a coarse mesh with $K = 3167$ elements, see Fig. 8.22a, with $\Delta t = 5 \cdot 10^{-4}$ and fine mesh $K = 8395$, see Fig. 8.23a, with $\Delta t = 2.5 \cdot 10^{-4}$ and polynomials of degree $P = 3$. DIRK22 time stepping has been applied.

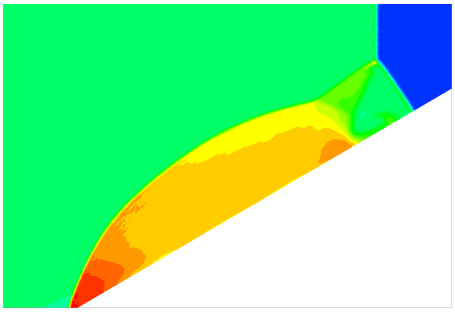
The solution on the coarse and the fine mesh are presented in Fig. 8.22 and Fig. 8.23. We present the used mesh, a pseudocolor and an isolines plot of the density and the element-wise artificial viscosity. As for the Mach 3 step test case the viscosity is only applied in the vicinity of the shock and not in areas with smooth solution. This indicates that the smoothness indicator detects the discontinuities correctly. The application of viscosity in a vertical line in the center of the domain above the shock results from an artifact due to the discontinuous initial conditions that travels through the domain. Most artificial viscosity is added at the shocks with less added in the shock interaction region. The shocks are approximated sharply and are thus clearly recognizable. On the fine mesh the shock is captured more compactly, which can also be seen in the thinner lines of artificial viscosity added. Moreover, the solution is captured more accurately in the region where the jet occurs. However, the Helmholtz instability cannot be observed on the coarse nor on the fine mesh. The shock capturing



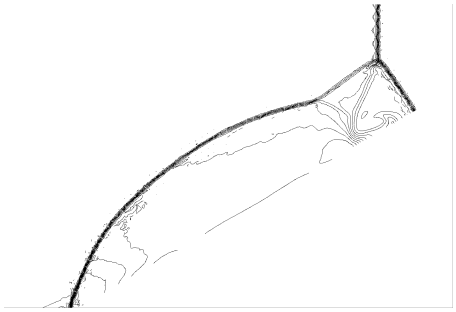
(a) Mesh with $K = 3167$ elements.



(b) Cell-wise artificial viscosity using 14 levels from $\varepsilon_{\text{av}}^{k,\min} = 0.0$ and $\varepsilon_{\text{av}}^{k,\max} = 0.014$.

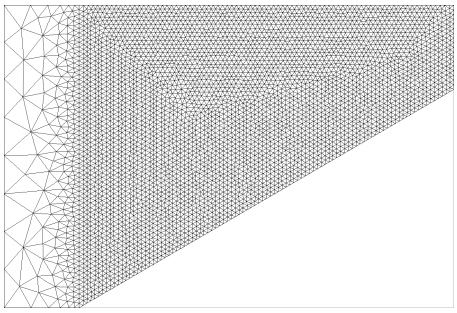


(c) Density distribution for 20 levels with $\rho_{\min} = 1$ and $\rho_{\max} = 19$.

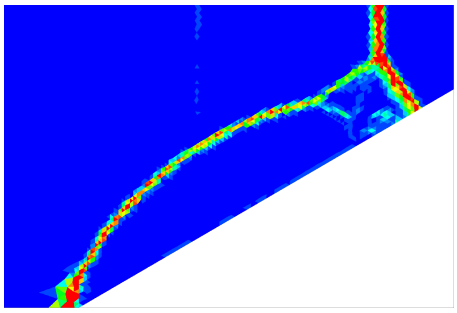


(d) Isolines of the density distribution for 20 levels with $\rho_{\min} = 1$ and $\rho_{\max} = 19$.

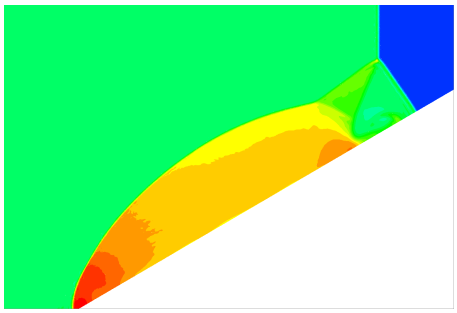
Figure 8.22.: Mesh and solution on the physical domain at $t = 0.2$ for the double Mach wedge (Euler equations) on a mesh with $K = 3167$ elements. The CFL-number is CFL ≈ 0.163. The HDG method with $P = 3$ has been used as space discretization.



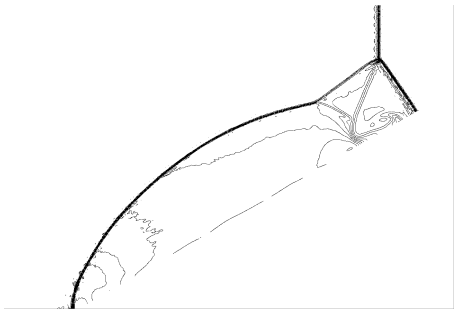
(a) Mesh with $K = 8395$ elements.



(b) Cell-wise artificial viscosity using 15 levels from $\varepsilon_{\text{av}}^{k,\min} = 0.0$ and $\varepsilon_{\text{av}}^{k,\max} = 0.007$.



(c) Density distribution for 20 levels with $\rho_{min} = 1$ and $\rho_{max} = 19$.



(d) Isolines of the density distribution for 20 levels with $\rho_{\min} = 1$ and $\rho_{\max} = 19$.

Figure 8.23.: Mesh and solution on the physical domain at $t = 0.2$ for the double Mach wedge (Euler equations) on a mesh with $K = 8395$ elements. The CFL-number is CFL ≈ 0.148. The HDG method with $P = 3$ has been used as space discretization.

parameters or the mesh resolution lead to too much dissipation such that no instability appears. These results have also been discussed in [127].

Rectangular domain The results on the "physical" domain show the structure of the solution as expected, but fails to resolve all features. Therefore, we redo the simulation on the rectangular domain and a much finer mesh with $K = 58240$ elements and polynomials of degree $P = 4$. We use DIRK22 time stepping with time step size $\Delta t = 10^{-5}$. We set the parameters of the shock capturing method to $\varepsilon_{\text{av}}^0 = 1.5$, $\kappa_{\text{av}} = 0.1$ and $s_0 = 2$. As mentioned before we have to slightly adjust the initial and boundary conditions for the rectangular domain. The left (L) hand state is given by

$$\rho_l = 8, u_{1,l} = 8.25\sin(60^\circ), u_{1,l} = -8.25\cos(60^\circ), p_l = 116.5$$

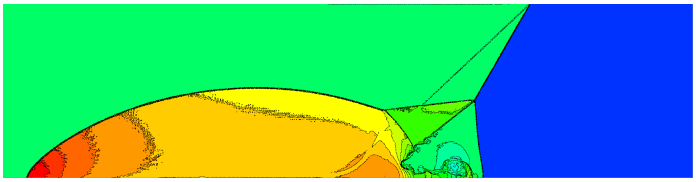
to account for the fact that the flow is not parallel to the x_1-axis when entering the domain. The right hand state (R) remains unchanged and is thus given by

$$\rho_r = 1.4, u_{1,r} = 0, u_{1,r} = 0, p_r = 1.$$

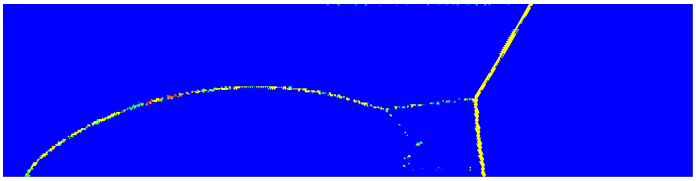
At the top boundary we prescribe the left and right hand state during the simulation. This boundary is time-dependent as we have to prescribe the shock position on the boundary. The x_1-coordinate of the shock is given by

$$x_s(t) = \frac{1}{6} + \frac{1}{\tan(60^\circ)} + \frac{10}{\sin(60^\circ)}t.$$

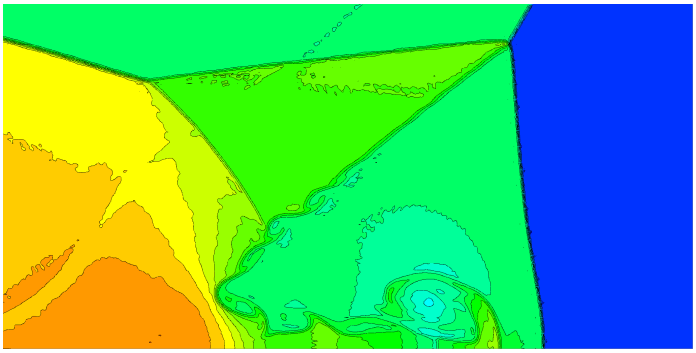
As for the simulation on the physical domain the general solution structure is clearly recognizable, see Fig. 8.24a. The shocks are approximated nicely and are very sharp. The sharp approximation of the shock is also reflected by the very thin strip of artificial viscosity added at the shock, see Fig. 8.24b. Moreover, the high spatial resolution makes the Helmholtz instability at the jet observable. In order to improve the visibility of the instability, we present also a zoom into the shock interaction region, see Fig. 8.24c. The instability is clearly visible and in good agreement with results in the literature, see [182], for example.



(a) Density distribution.



(b) Cell-wise artificial viscosity.



(c) Zoom into the shock inter action region.

Figure 8.24.: Density and artificial viscosity of the double Mach wedge test case (Euler equations). We present the solution on the rectangular domain and the mesh consists of $K = 58240$. The CFL-number is CFL ≈ 0.013. The HDG method with $P = 4$ has been used as space discretization.

9. Application of a hierarchical scale separation solver

We have discussed in Sec. 3.3.1 that assembling and solving the linear system of equations is the most expensive part of the numerical discretization. The amount of work and memory needed for the linear system has been reduced by using a hybridized discontinuous Galerkin method. The assembly process is more involved due to the computation of the local solves, but the solving process is usually faster due to the much smaller matrix and smaller number of nonzero entries, see Sec. 3.3.3.

Further opportunities to reduce the computational effort lie within the applied linear solver. The linear system

$$\mathrm{A}\lambda = b$$

arising from the discretization is large but sparse. The matrix could be inverted using direct solvers which would explicitly compute the inverse A^{-1}. However, the inverse is usually dense, i.e. has many nonzero entries, hence it would lead to unfeasible memory and run-time requirements. Thus, it is common to use iterative methods. One important ingredient for an efficient iterative method is a sufficiently good preconditioner. However, choice of a suitable iterative solver and preconditioner also rely heavily on the used space discretization and the discretized PDE.

In the field of CFD, the generalized minimal residual (GMRES) [201] method is very popular. Common choices for preconditioning are block Jacobi and Gauss-Seidel methods [151] or incomplete LU (ILU) factorization [183]. In practice, the GMRES method works very well, especially for time-dependent problems where the changes in the solution between two time steps usually are small compared to steady state problems. Therefore, it is not necessary to use an ILU preconditioner that is also very expensive in terms of run-time when used for time-dependent problems. Due to the impact on the overall efficiency of the CFD solver, especially efficient linear solvers [3, 75, 88, 134, 148, 150, 160, 227, 228] and preconditioners [63, 176, 180, 183, 217] for DG methods are a topic of great interest.

Approaches to efficiently solve linear systems arising from DG discretizations include p- and hp-multigrid methods [75, 148, 150] that use a hierarchy of meshes to improve or speed up the solution process. The p-part refers to generating such a hierarchy by varying the polynomial degree to get this hierarchy. Thus $p = 0$ is the coarsest level, $p = 1$ the next finest level and so on. The h-part refers to the generation of an hierarchy of meshes. Since varying p is usually more straight forward this is a popular choice for DG methods. Note that the

standard notation for the polynomial degree p and mesh size h from the literature have been used in this paragraph.

In [159, 160] the authors construct an iterative method particularly constructed for HDG methods for linear problems. A special two-grid method [24, 59] for discontinuous Galerkin methods first solves the problem on a very much coarse mesh. Afterwards the solution from the coarse mesh is used to linearize and solve the problem on the fine mesh such the overall computational costs are close to solving the linear system once on the fine mesh because Newton's method on the fine mesh converges after one step. A linear solver for discontinuous Galerkin methods using hierarchical basis functions has been introduced in [3]. The idea of this linear solver is to disassemble the solution representation in a fine and a coarse scale. The disassembly can be done cheaply if hierarchical basis functions are used [135] and the solver can be interpreted as a special version of a two-grid p-multigrid method. The approach has been studied for HDG and linear problems in [210]. Another variant of this scheme called the inexact hierarchical scale separation (IHSS) has been introduced in [227, 228]. The authors make extensive use of the features of the HSS for an efficient implementation that can be much faster that a classical GMRES method. A variant of their method for MATLAB is available at GitHub [78].

In the following sections, we discuss the hierarchical scale separation approach for HDG. It has been applied to nonlinear steady-state problems. We present and discuss the results in the end of this chapter. Some of the results for nonlinear test cases have been presented and discussed in [125].

9.1. Hierarchical scale separation solver

The solver employed in this section is a hierarchical scale separation solver [3, 135] which shares similarities with a P-multigrid method [75, 150]. The authors of the HSS method emphasize that it has two distinctive features separating it from other multigrid methods.

1. The hierarchical scale separation solver splits the degrees of freedom associated with the solution representation in only two different scales: a *coarse* and a *fine* scale.

2. All degrees of freedom of the *fine* scale are updated simultaneously. This is different from other schemes where these are updated consecutively.

In order to define the coarse and the fine scale one has to use a discontinuous Galerkin discretization with hierarchical basis functions. In the initial publication, Taylor polynomials have been used. We use Legendre polynomials as described in Sec. 3.2 that form a hierarchical basis, as well. In this case, the discrete unknown λ_h can be expressed as

$$\lambda = \bar{\lambda} + \lambda', \tag{9.1}$$

where the coarse scale $\bar{\lambda}$ refers to the solution based on the $P = 0$ basis function and the fine scale λ' to the remaining degrees of freedom. The polynomial representation of the hybrid unknown (3.14) can be used to identify coarse scale and fine scale solution as

$$\bar{\lambda}_h(t,x) = \Lambda_1^{\bar{k}} \mu_{h,1}^{\bar{k}}(x), \quad \lambda'_h(t,x) = \sum_{i=2}^{\overline{N}} \Lambda_i^{\bar{k}} \mu_{h,i}^{\bar{k}}(x), \quad x \in E_{\bar{k}}.$$

Obviously, this splitting is only interesting for $P > 0$. Otherwise, the system of equations simplifies to the piece-wise constant case where only a coarse scale exists.

The idea of the hierarchical scale separation approach is to solve the globally coupled system of equations for the coarse scale problem. The fine scale problem acts as a correction of the coarse scale solution that is decoupled from element dependencies. The decoupling allows to apply the fine scale correction locally in an iterative fashion and is carried out until the residual drops below a given threshold.

The splitting in a coarse and fine scale as given in (9.1) allows us to reorder the linear system of equations $\mathrm{A}\lambda = b$ such that it can be written as

$$\begin{pmatrix} \overline{\mathrm{A}} & \overline{\mathrm{B}} \\ \mathrm{B}' & \mathrm{A}' \end{pmatrix} \begin{pmatrix} \bar{\lambda} \\ \lambda' \end{pmatrix} = \begin{pmatrix} \bar{b} \\ b' \end{pmatrix}.$$

The reordering results in several matrices coupling the unknowns of different scales and a corresponding splitting of the vector of unknowns and the right hand side.

The matrix $\overline{\mathrm{A}}$ couples degrees of freedom of the coarse scale with each other and $\overline{\mathrm{B}}$ couples coarse scale degrees of freedom with fine scale degrees of freedom. Matrix A' couples fine scale degrees of freedom and B' couples fine scale degrees of freedom with coarse scale degrees of freedom. Accordingly, the vector $\bar{b}$ is the right hand side of the coarse scale and b' the one of the fine scale.

The iterative solution procedure uses this splitting in different scales, see Alg. 1. The coarse scale $\overline{\mathrm{A}}$ system is used to get a new approximation of the coarse solution. Afterwards, the fine scale system A' is used to improve the solution. In order to do so in an efficient manner the matrix is split into a block-diagonal part $\mathrm{A}'_{\mathrm{diag}}$ and its remainder $\mathrm{A}'_{\mathrm{offdiag}} = \mathrm{A}' - \mathrm{A}'_{\mathrm{diag}}$. The fine scale correction is constructed such that only the block diagonal matrix has to be inverted.

We have mentioned in Sec. 3.3.2 that the computation of local solves can be parallelized very efficiently because these computations are element-wise. The HSS approach introduces an additional layer of locality that can be exploited for an efficient implementation. The coarse scale system is globally coupled, but it only grows slowly as it depends on the number of elements rather than on the polynomial degree of the basis function. The assembly and solving of the coarse scale system of linear equations needs a certain amount of communication of all processes. Nevertheless, assembling and solving the coarse scale system of equations

Data: $\overline{\mathrm{A}}, \mathrm{A}', \overline{\mathrm{B}}, \mathrm{B}', \bar{b}, b', tol, i_{\max}$
Result: Solution λ_{n+1}.
$\bar{\lambda}_0 = 0$;
$\lambda'_0 = 0$;
$i = 0$;
$r_0 = \sqrt{\|\bar{b}\|_2^2 + \|b'\|_2^2}$;
while $(\frac{r_i}{r_0} > tol$ *and* $(r_i > 10^{-16})$ *and* $i < i_{\max}$ **do**
  Solve: $\overline{\mathrm{A}}\bar{\lambda}_{i+1} = -\widehat{\mathrm{B}}\lambda'_i + \bar{b}$;
  Solve: $\mathrm{A}'_{\mathrm{diag}}\lambda'_{i+1} = -\mathrm{B}'\bar{\lambda}_{i+1} - \mathrm{A}'_{\mathrm{offdiag}}\lambda'_i + b'$;
  $\lambda_{i+1} = \bar{\lambda}_{i+1} + \lambda'_{i+1}$;
  $\bar{r} = \widehat{\mathrm{B}}\lambda'_{i+1} + \overline{\mathrm{A}}\bar{\lambda}_{i+1} - \bar{b}$;
  $r' = \mathrm{B}'\bar{\lambda}_{i+1} + \mathrm{A}'\lambda'_{i+1} + \mathrm{A}'_{\mathrm{diag}}\lambda'_{i+1} - b'$;
  $r_{i+1} = \sqrt{\|\bar{r}\|_2^2 + \|r'\|_2^2}$;
  $i = i + 1$;
end
$\lambda_{n+1} = \mathrm{unmap}(\bar{\lambda}_i, \lambda'_i)$;

Algorithm 1: Implemented HSS algorithm as drop-in replacement for PETSc's GMRES.

is still much cheaper than for other methods due to the much lower number of degrees of freedom (one per element for HSS). The correction/smoothing process involving the degrees of freedom from $P > 0$ basis functions can be done completely locally because the system of equations is split such that only the block diagonal part $\mathrm{A}'_{\mathrm{diag}}$ must be inverted for the fine scale system of equations. No information from other edges nor elements is needed when $\mathrm{A}'_{\mathrm{diag}}$ is inverted. Similar to the local solves these can be done element-wise and could be done by a direct solver as the matrices are rather small. However, the implementation used to generate the results presented in this thesis, does not solve the fine scale problem in a local fashion. The authors in [227, 228] exploit the locality explicitly for an efficient solution process of the fine scales. They could observe superior behavior of the solver over GMRES in terms of runtime. In the initial publication of HSS [3], the authors have already shown that the HSS approach can be much more efficient than a more 'traditional' P-multigrid approach.

Implementation

We want to briefly discuss the implementation as it differs slightly from the one in other publications. We use a (preconditioned) restarted GMRES provided by PETSc for the coarse and fine scale solves. In order to keep the implementation simple, we construct the matrix $\mathrm{A}'_{\mathrm{diag}}$ containing all diagonal blocks instead of only assembling the matrix locally for each

edge. The resulting system for the update

$$\mathrm{A}'_{\mathrm{diag}}\lambda'_{i+1} = -\mathrm{B}'\bar{\lambda}_{i+1} - \mathrm{A}'_{\mathrm{offdiag}}\lambda'_i + b',$$

see Alg. 1, is solved by a preconditioned GMRES method. The authors of the IHSS method [227, 228] also employ a restarted GMRES method for the coarse scale and solve the fine scale solve problem using the QR decomposition of the local block matrices. The HSS solver applied to the HDG method in [210] solves the system of equations with a GMRES method without restarts nor a preconditioner.

The implementation of the HSS solver employed in this work aims to be a drop-in replacement for an already implemented GMRES solver based on PETSc. The term "drop-in" means that the newly implemented solver has the same programming interface as the previously used linear solver. The HSS solver obtains preassembled matrix blocks as for other linear solvers available through PETSc and disassembles these blocks to construct the matrices $\overline{\mathrm{A}}, \mathrm{A}'_{\mathrm{diag}}, \mathrm{A}'_{\mathrm{offdiag}}, \overline{\mathrm{B}}, \mathrm{B}'$ and vectors $\bar{b}, b'$ internally. In order to do so, it relies on the fact that the block format for matrices of PETSc is used. In this case, PETSc does expect the entries to be grouped in blocks. We choose blocks of size $m \times m$ with m being the number of unknowns in the vector of unknowns w. We briefly illustrate the construction of the linear system of equations based on the blocks at the example of the contribution of a single edge $E_{\bar{k}} \in \mathcal{E}_h$ to the linear system of the hybrid unknown (3.29). The contribution of a single edge is constructed such that

$$\mathrm{A}_{\bar{k}}\Lambda_{\bar{k}} = b_{\bar{k}}$$

holds. The matrix

$$\mathrm{A}_{\bar{k}} = \begin{pmatrix} \mathrm{A}_{1,1} & \dots & \mathrm{A}_{N,1} \\ \vdots & \ddots & \vdots \\ \mathrm{A}_{1,N} & \dots & \mathrm{A}_{N,N} \end{pmatrix} \tag{9.2}$$

consists of blocks $\mathrm{A}_{i,j}$ of size $m \times m$ that resemble the contribution

$$\mathrm{A}_{i,j} = a\left(\mu_{i,h}, \mu_{j,h}\right)_{m\times m}$$

of the ith basis function of each unknown tested against the jth test function. Accordingly, the vector of unknowns is structured in subvectors of length m as

$$\Lambda_{\bar{k}} = \left(\Lambda^{\bar{k}}_{1,1}, \dots, \Lambda^{\bar{k}}_{1,m}, \Lambda^{\bar{k}}_{2,1}, \dots, \Lambda^{\bar{k}}_{2,m}, \dots, \Lambda^{\bar{k}}_{N,1}, \dots, \Lambda^{\bar{k}}_{N,m}\right)^{\mathrm{T}}. \tag{9.3}$$

The right hand side vector b is ordered accordingly. The ordering of the vectors makes it easy to split them into the two scales

$$\overline{\Lambda}_{\bar{k}} = \left(\Lambda^{\bar{k}}_{1,1}, \dots, \Lambda_{1,m}\right)^{\mathrm{T}}$$
$$\Lambda'_{\bar{k}} = \left(\Lambda^{\bar{k}}_{2,1}, \dots, \Lambda^{\bar{k}}_{2,m}, \dots, \Lambda^{\bar{k}}_{N,1}, \dots, \Lambda^{\bar{k}}_{N,m}\right)^{\mathrm{T}}.$$

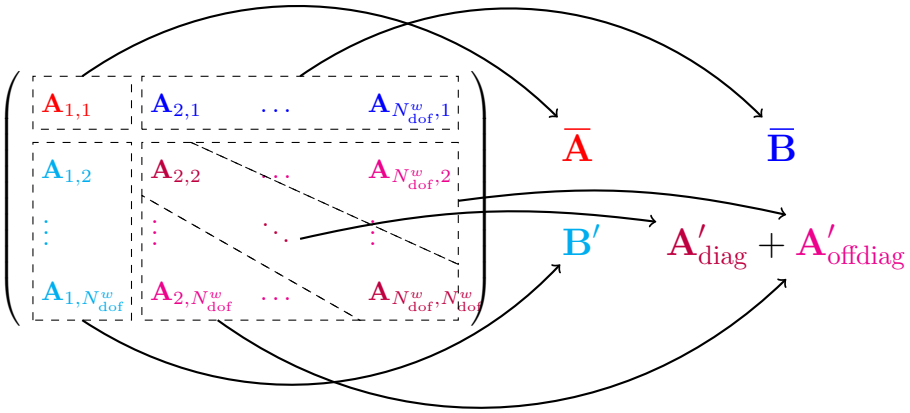


Figure 9.1.: Schematic of the splitting of the local matrix contribution of a single edge.

The matrix 9.2 is split accordingly to the vector into the coarse and fine scales, see Fig. 9.1. After solving the linear system using the HSS algorithm, see Alg. 1, the updated solution vector is unmapped into its initial order (9.3) and returned to the calling function.

9.2. Numerical results

We present the numerical results obtained from the hierarchical scale separation solver applied to the hybridized discontinuous Galerkin method. First, results for a linear test case are presented to verify the new implementation. Afterwards, the results of a variety of nonlinear problems, namely the viscous Burgers' equation, Euler equation and Navier-Stokes equation, are presented. In contrast to other chapters of this thesis we focus solely on steady-state problems to focus on the behavior of linear solver.

Linear convection-diffusion

We solve a boundary linear test case as described in [69]. The same test case has been used for HDG and HSS in [210]. The linear convection-diffusion equation, see Sec. 2.4, with given velocity and a given diffusion constant ε is solved on $\Omega = [0,1]^2$. Additionally, a source term

$$h(x) = \left(x_2 + \frac{e^{\frac{x_2}{\varepsilon}-1}}{1-e^{\frac{1}{\varepsilon}}}\right) + \left(x_1 + \frac{e^{\frac{x_1}{\varepsilon}-1}}{1-e^{\frac{1}{\varepsilon}}}\right)$$

Option	HSS	Coarse level	Fine level	PETSc
Relative tolerance	10^{-10}	10^{-12}	10^{-12}	10^{-10}
Norm of residual	$\|\cdot\|_2$	$\|\cdot\|_2$	$\|\cdot\|_2$	$\|\cdot\|_2$
Preconditioner	–	ILU(k)	ILU(k)	ILU(k)
ILU(k) depth	–	3	3	3
Iterative solver	–	restarted (F)GMRES	restarted (F)GMRES	restarted (F)GMRES
Restart iterations	–	50	50	50
Max. Iterations	1000	1000	1000	1000

Table 9.1.: Parameters of the HSS solver and pure PETSc/GMRES solver for the boundary layer test case.

is added. The solution for diffusion constant $\varepsilon = 0.01$, where a strong boundary layer develops, is shown in Fig. 9.2c and for inverted convection direction in Fig. 9.2d. The solution can be computed analytically as

$$w(x) = \left(x_1 + \frac{e^{\frac{x_1}{\varepsilon}} - 1}{1 - e^{\frac{1}{\varepsilon}}} \right) \left(x_2 + \frac{e^{\frac{x_2}{\varepsilon}} - 1}{1 - e^{\frac{1}{\varepsilon}}} \right)$$

and depends on the convection velocity and the diffusion constant. For an increasing diffusion constant ε the steepness of the boundary layers decreases. The initial mesh with $K = 8$ elements and the mesh after three refinements are displayed in Fig. 9.2a and Fig. 9.2b.

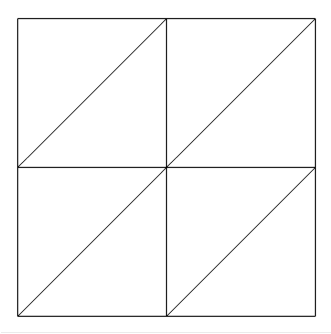
The solvers use similar settings. The linear solvers, let it be PETSc or HSS, solve the system of equations until the relative residual drops below 10^{-10} or the absolute residual gets too close to machine precision. Both solvers are limited to a maximum of 1000 iterations. The internal fine and coarse scale use the same options for the GMRES method. The settings are summarized in Tab. 9.1.

The Newton solver aims for an absolute residual below 10^{-10} within at most ten Newton steps. We use uniformly refined meshes with up to $K = 131072$ elements and diffusion constants $\varepsilon = \{0.01, 0.1, 1\}$.

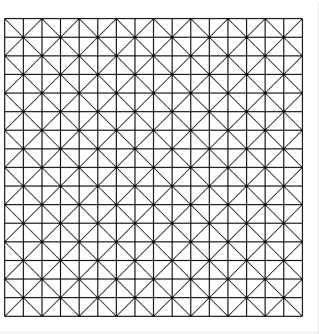
Scalar case

We expect that the method recovers an order of convergence of $P + 1$. Since we are focusing on steady problems, no time integrator has to be applied. Moreover, we expect that the results obtained from the two linear solvers are very similar because both solve the linear system while aiming for the same reduction of the relative residual.

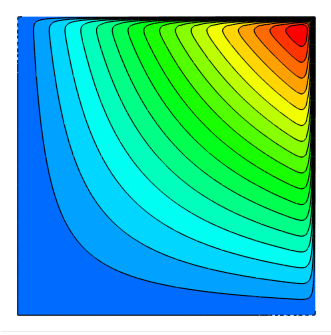
In Fig. 9.3 we present the evolution of the error under uniform refinement for the three different values of ε. Additionally, we present the errors obtained using the GMRES method from PETSc (left) that we refer to simply as PETSc and the HSS solver (right). There is virtually no difference in the solutions of the two linear solvers for neither combination



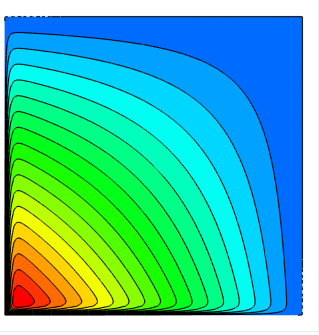
(a) Initial mesh with $K = 8$ elements.



(b) Mesh after three refinements with $K = 512$ elements.



(c) Solution for $\varepsilon = 0.01$ and $u = (1, 1)^{\mathrm{T}}$. We show the numerical solution of the HSS solver based on PETSc with $P = 3$ and $K = 2048$ elements.



(d) Solution for $\varepsilon = 0.01$ and $u = (-1, -1)^{\mathrm{T}}$. We show the numerical solution of the HSS solver based on PETSc with $P = 3$ and $K = 2048$ elements.

Figure 9.2.: Initial mesh and mesh after three refinements for the boundary layer test case (viscous Burgers' equation). Additionally we present the solution for two different convection velocities that lead to boundary layers in different corners of the domain. The HDG method has been used as space discretization.

of mesh size Δx, polynomial degree P and diffusion constant ε. This also indicates the correctness of our implementation of the HSS solver. For the sake of completeness we give also the errors for $P = 0$ although no fine scale exists in this case.

Additionally, we present the evolution of the relative residual over the number of iterations for the HSS solver in Fig. 9.4. The observed convergence behavior is in good agreement with results reported in [210] for the same test problem. The number of HSS iterations depends on the used diffusion constant ε and used polynomial degree P. This might be caused by the boundary layer that is very steep in this case and cannot be properly resolved on coarse meshes. The results for larger diffusion constants, where the boundary layer is resolved on coarser meshes, show less dependence of the number of iterations on the mesh size.

The dependence of HSS iterations on the polynomial degree P is also in good agreement with the results of [210]. Increasing P leads to a larger number of HSS iterations which is most prominent on larger meshes. This increase might be related to the resolution of the boundary layer, as well.

Another feature, also observed previously, is that the residual does not monotonically decrease in all cases. In many cases, the residual grows in the first HSS iteration and decreases monotonically after this.

A less desirable feature can be observed on the finest mesh for $\varepsilon = 1$ for all P and on the finest mesh for $\varepsilon = 0.1$ and $P = 3$. In these cases, the relative residual does not converge below 10^{-10}, but very close to it. On cannot expect the relative residual to drop arbitrarily because the absolute residual is already very small, i.e. $r_i = \mathcal{O}(10^{-14})$. Note that we only plot the residuals for the first 300 iterations, but the mentioned cases do not converge until the upper bound of 1000 iterations is reached.

Viscous Burgers' equation

The second test problem is the viscous Burgers' equation with fluxes

$$f_c(w) = \frac{1}{2}w^2, \quad f_v(w, \nabla w) = \varepsilon \nabla w$$

and constant diffusion constant ε. Additionally, a source term h is added to enforce the same solution as for the linear equation. Compared to previous publications dealing with the (I)HSS method, the problem is nonlinear due to the convective flux. Thus, the problem is linearized using Newton's method, see Sec. 3.3.1. The arising system of linear equations in each Newton step is solved using the HSS linear solver. The relative tolerances, for which the coarse and fine scale solve is stopped, are reduced to 10^{-13} to avoid the behavior on the finest meshes that was observed in the previous session. The Newton method is carried out until the relative residual drops by ten orders of magnitude or the absolute residual drops below 10^{-16}. All other parameters of the linear solvers and the test case remain unchanged.

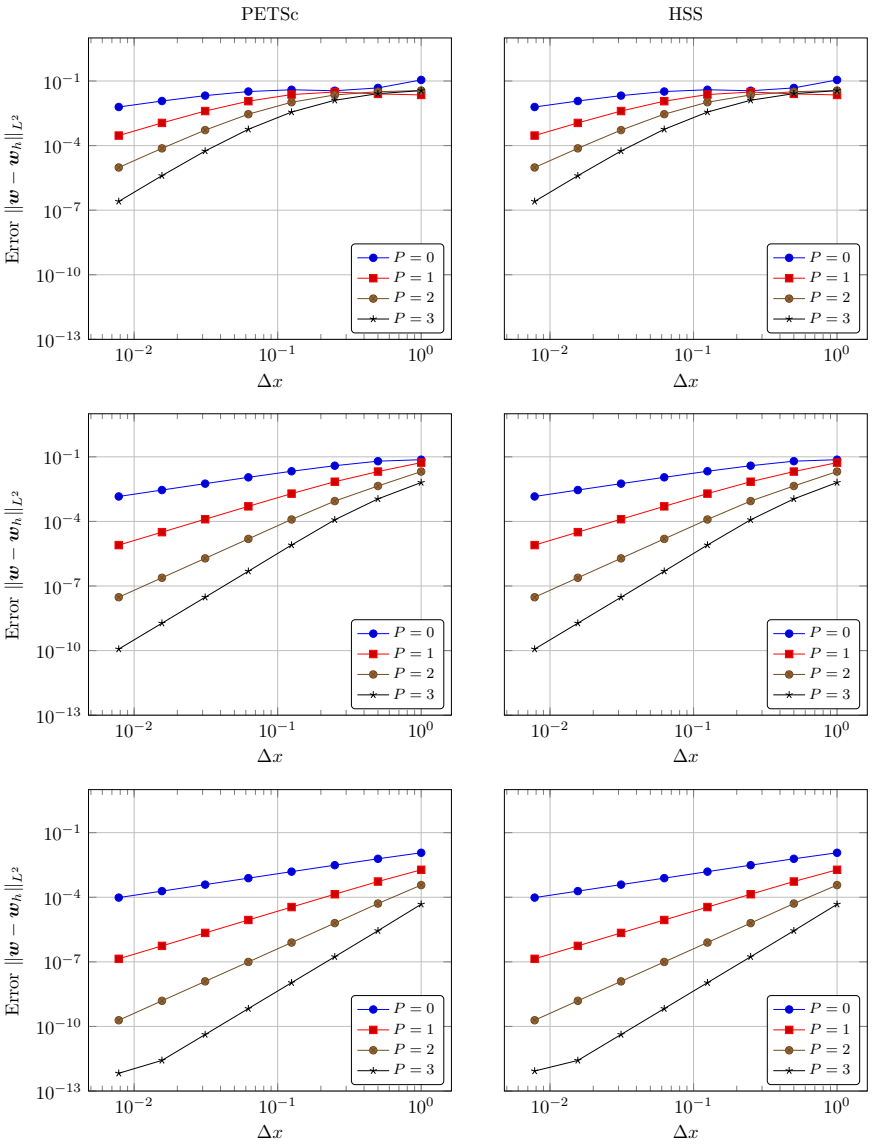


Figure 9.3.: The errors obtained by using the GMRES (left) and HSS (right) linear solver applied to the scalar problem (linear convection-diffusion equation). The error is plotted over the mesh size for different polynomial degrees $P = \{0, 1, 2, 3\}$. The diffusion constant grows from top to bottom $\varepsilon = \{0.01, 0.1, 1\}$. There is no visible difference in the errors of the two solutions. The HDG method has been used as space discretization.

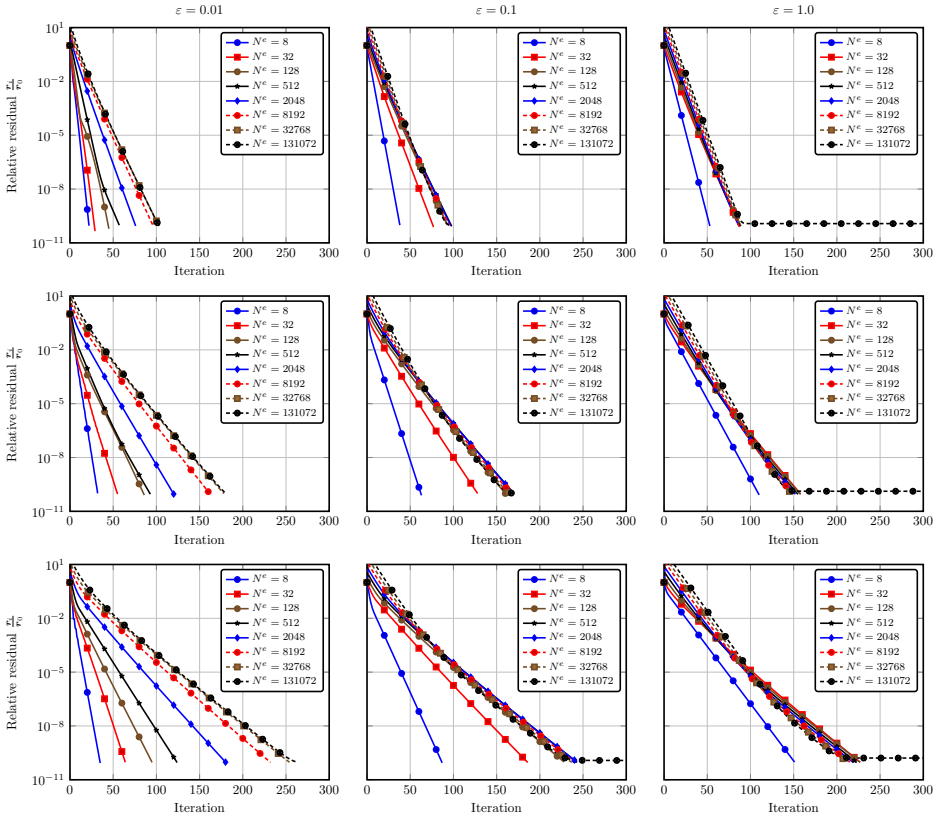


Figure 9.4.: Evolution of the relative residual $\frac{r_i}{r_0}$ of the HSS solver. The data presented has been obtained for the linear problem in the scalar case (convection-diffusion equation). The diffusion constant $\varepsilon = \{0.01, 0.1, 1\}$ grows from left to right and the polynomial degree $P = \{1, 2, 3\}$ grows from top to bottom. Each plot contains the convergence history for each mesh and 300 iterations are shown at most. Markers are only plotted for every 20th data point. The HDG method has been used as space discretization.

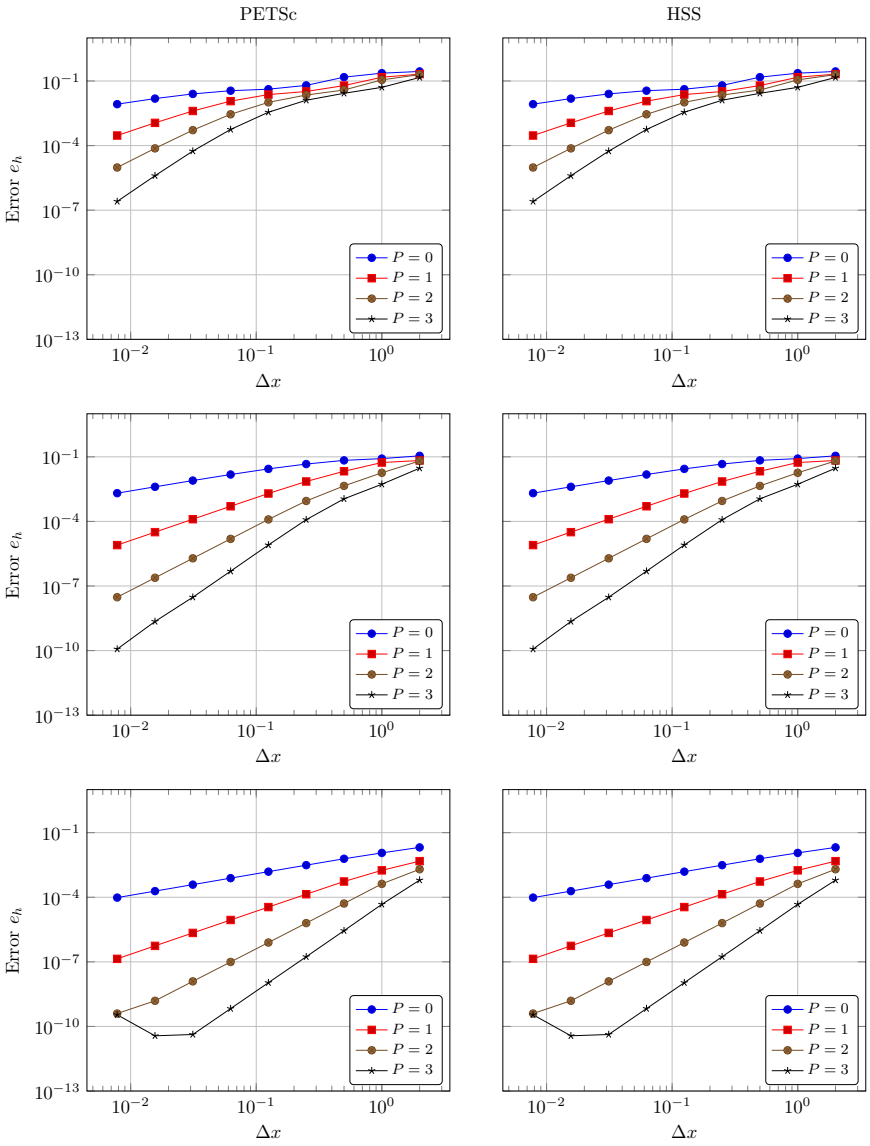


Figure 9.5.: Errors of obtained using the GMRES (left) and HSS (right) for the viscous Burgers' equation. The error is plotted over the mesh size for different polynomial degrees $P = \{0, 1, 2, 3\}$. The diffusion constant grows from top to bottom $\varepsilon = \{0.01, 0.1, 1\}$. There is no visible difference in the errors of the two solutions. The HDG method has been used as space discretization.

Like for the linear problem in the previous section, we present the errors obtained using PETSc's GMRES method and the HSS solver in Fig. 9.5. The results show again no obvious differences between the solutions of the two linear solvers. Moreover, the correct order of convergence is obtained after few refinements.

In Fig. 9.6 we present the convergence behavior of the HSS solver similar to the way we presented it for the linear problem in Fig. 9.4. However, the viscous Burgers' equation is nonlinear which means that the Newton solver usually needs more than one step. Therefore, we do not present the convergence behavior for all polynomial degrees and mesh sizes, but only for $P = 2$ on four different meshes $K = \{8, 128, 2048, 32768\}$ (top to bottom) and with diffusion constants $\varepsilon = \{0.01, 0.1, 1.0\}$ (left to right). The figure includes the convergence behavior for each Newton step.

It is obvious from Fig. 9.6 that the number of Newton steps needed depends on the spatial resolution. Especially when the boundary layer is pronounced, i.e. $\varepsilon = 0.01$, the most Newton steps are needed. For $\varepsilon = \{0.01, 0.1\}$ the number of steps reduces slightly when the mesh is refined and the boundary layer is better resolved. In the case of $\varepsilon = 1$ the mesh size does not seem to have an influence on the number of Newton iterations.

The HSS method also shows a slight dependence on the diffusion constant ε and mesh size. Especially for the coarse meshes $K = \{8, 32\}$ and steep boundary layer $\varepsilon = 0.01$ the largest number of HSS iterations are needed. Moreover, the number of HSS iterations per Newton step varies greatly with the first Newton step needing the most HSS iterations. In the most other cases, the first Newton step is the one that needs the most HSS iterations to solve the linearized problem, as well. On finer meshes the number of HSS iterations needed per Newton step varies much less and is between 100 and 200 iterations.

For the test cases with a less pronounced boundary layer, i.e. $\varepsilon = \{0.1, 1\}$, the number of HSS iterations only varies very little over the Newton steps, but also only very little over different mesh sizes or diffusion constants. The strongest dependency can again be observed on coarse meshes, but in most cases the number of HSS iterations needed is around 140. Similar as for the linear test case the case with $\varepsilon = 1$ is the least problematic test case. Moreover, one can also observe again that the residual tends to increase for the first HSS iteration for some problem settings before it monotonically decreases.

Euler equations

For the application of the HSS solver to nonlinear problems, we investigate inviscid flow described by the Euler equations around a NACA0012 airfoil. We use the same settings and mesh with $K = 2560$ as the authors in [213]. The Mach number at free stream conditions is $\mathrm{Ma} = 0.3$ and the angle of attack is $\alpha = 4\,\mathrm{deg}$. The flow parameters are chose such that no shocks occur. The domain is a circle centered around the airfoil with radius of 50 times the

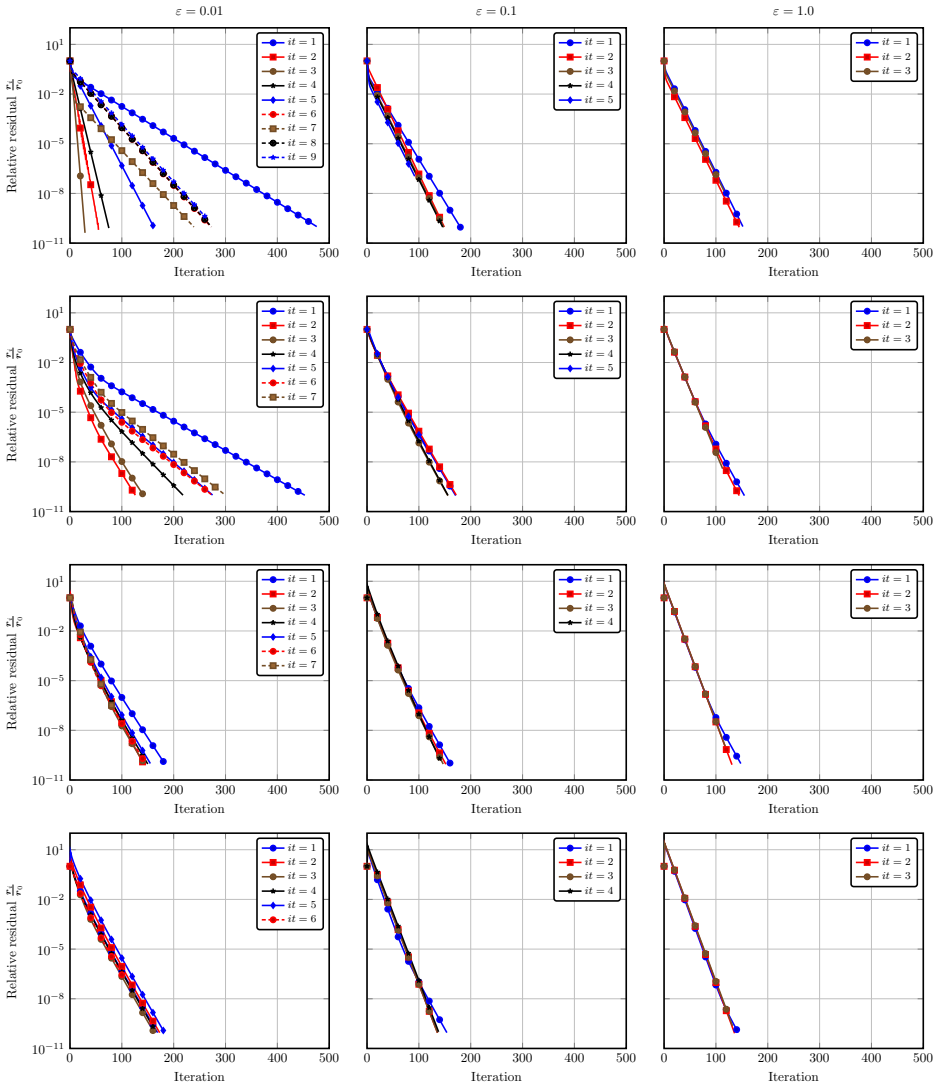


Figure 9.6.: The convergence behavior of the HSS method in each Newton iteration when solving the viscous Burgers' equation. The results are shown for increasing diffusion constant $\varepsilon = \{0.01, 0.1, 1.0\}$ (left to right) and refined meshes with $K = \{8, 128, 2048, 32768\}$ elements (top to bottom). In all cases polynomials of degree $P = 2$ are used. Markers are only shown for every 20th data point. The HDG method has been used as space discretization.

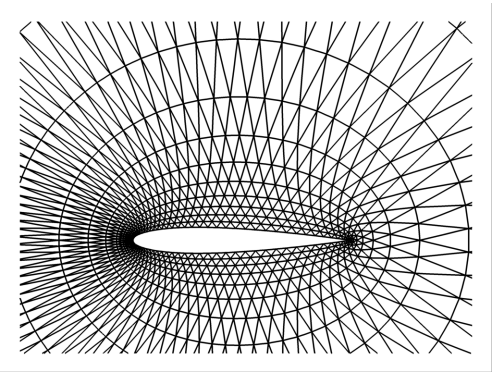


Figure 9.7.: Excerpt of the mesh used for inviscid flow around a NACA0012 airfoil.

	GMRES	HSS
$P = 0$	1.420757e+00	1.420757e+00
$P = 1$	1.944455e+00	1.944455e+00
$P = 2$	1.950037e+00	–
$P = 3$	1.951374e+00	–

(a) Lift coefficient.

	GMRES	HSS
$P = 0$	2.368942e-01	2.368942e-01
$P = 1$	3.034639e-03	3.034639e-03
$P = 2$	9.622760e-04	–
$P = 3$	8.007038e-04	–

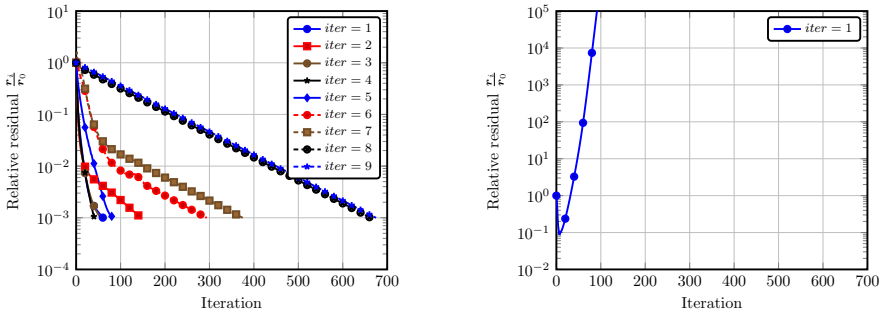
(b) Drag coefficient

Table 9.2.: Lift and drag cofficients the GMRES method and the HSS solver for different polynomial degrees P. Results for inviscid flow (Euler equations). The problem has been solved using the HDG method with different P on a mesh with $K = 2560$ elements.

chord length. An excerpt of the mesh is shown in Fig. 9.7.

This test case is quite complicated to solve without proper initial conditions. Therefore, a P-multigrid technique is used to solve the problem. The problem is first solved for $P = 0$. Then, the solution is projected onto $P = 1$ and so on until the desired P is reached. In our case we aim for a solution at $P = 3$. Both linear solvers aim to reduce the relative residual by three orders of magnitude. The relative tolerance of the GMRES methods for the coarse and fine scale are set to 10^{-8}. All other parameters are as in Tab. 9.1. The Newton solver stops when the absolute residual drops below 10^{-10} or more than 12 Newton steps are needed.

We compare the results obtained the by GMRES methods and the HSS solver by computing the lift and drag coefficient. The results are reported in Tab. 9.2. One can see that the results are indistinguishable up to $P = 1$. However, no results for the HSS solver and higher polynomial degrees are reported because the HSS solver does *not* converge.



(a) Relative error $P = 1$.

(b) Relative error $P = 2$.

Figure 9.8.: Evolution of the relative residual of the HSS solver for polynomials degrees $P = \{1, 2\}$. The data displayed has been obtained for the inviscid test case. The problem has been solved using the HDG method with different P on a mesh with $K = 2560$ elements.

In Fig. 9.8 we present the convergence behavior of the HSS solver for each Newton step. For $P = 1$ one observes a similar behavior as for the viscous Burgers' equation: The first Newton iteration is the one that needs the most HSS iterations. Consecutive Newton steps usually need less iterations. For $P = 2$, one sees that the HSS solver converges for the first few iterations (≈ 8), but then starts to diverge quickly. Problems with the HSS solver diverging for linear scalar problems have been reported in [210]. This is clearly undesired behavior and an open problem for further investigation.

Navier-Stokes equations

Although the results in the previous section are quite disappointing we want to apply the HSS solver also to the Navier-Stokes equations. We investigate again a NACA0012 airfoil. We use the same mesh as in [211] where the problem is solved on a quadratic domain. The angle of attack is set to $\alpha = 5\,\text{deg}$, the Mach number is $\text{Ma} = 0.2$ and the Reynolds number $\text{Re} = 100$. The settings of the linear solvers are the same as in the inviscid case. The tolerance of the Newton solver is set to 10^{-8} and the maximum number of Newton steps is set to 15. The solver uses the same P-multigrid startup procedure as in the inviscid case.

As for the inviscid test case we compute the lift and drag coefficient to compare the solutions obtained by the GMRES method and the HSS solver. We present the results in Tab. 9.3. Again, no results are reported for the HSS solver for polynomials of degree $P > 1$ because the HSS solver diverges. Nevertheless, the obtained results by the two different linear solvers are in good agreement. For the drag coefficient the results are in fact indistinguishable.

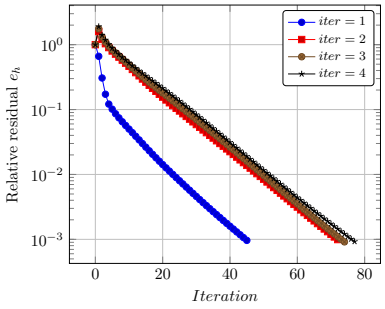
	GMRES	HSS
$P = 0$	7.262163e-02	7.262161e-02
$P = 1$	3.018314e-02	3.018304e-02
$P = 2$	2.902854e-02	–
$P = 3$	2.862259e-02	–

(a) Lift coefficient.

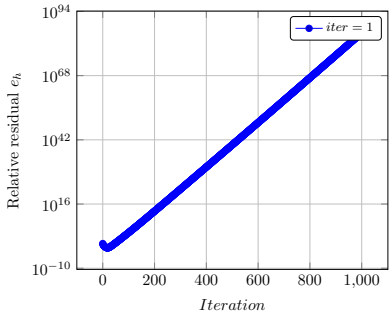
	GMRES	HSS
$P = 0$	2.511540e-01	2.511540e-01
$P = 1$	1.750759e-01	1.750759e-01
$P = 2$	1.766604e-01	–
$P = 3$	1.767823e-01	–

(b) Drag coefficient

Table 9.3.: Lift and drag cofficients of a NACA0012 airfoil obtained by a GMRES method and the HSS solver for different polynomial degrees P. The Navier-Stokes equations have been solved using the HDG method with different P on a mesh with $K = 2560$ elements. The angle of attack is set to $\alpha = 5\,\text{deg}$, the Mach number is $\text{Ma} = 0.2$ and the Reynolds number $\text{Re} = 100$.



(a) Relative error $P = 1$.



(b) Relative error $P = 2$.

Figure 9.9.: Convergence behavior of HSS solver based on PETSc solvers with different polynomials degrees. Navier-Stokes case. The Navier-Stokes equations have been solved using the HDG method with different P on a mesh with $K = 2560$ elements. The angle of attack is set to $\alpha = 5\,\text{deg}$, the Mach number is $\text{Ma} = 0.2$ and the Reynolds number $\text{Re} = 100$.

In Fig. 9.9 we plot the convergence behavior for each Newton step. Surprisingly, the first Newton step for $P = 1$ is the one needing the fewest HSS iterations. This behavior was not observed for the other test cases. For $P = 2$, the HSS converges for the first few (≈ 17) iterations and diverges quickly after that. This is very similar to what was observed in the inviscid case. Thus, the same remarks as for the inviscid problem hold. In general, the HSS solver looks promising for the solution of linear systems of equations stemming from DG discretizations. However, the divergent behavior for nonlinear systems of equations of higher polynomial degrees has to be tackled in future work. A possible remedy is to tune the stability parameter of the numerical flux functions. In [210] the authors could observe that the convergence behavior is dependent on the choice of the stability parameter. One can also try to use another numerical flux and other linear solver to solve the coarse and fine scale problems of the HSS solver.

10. Conclusion and future work

In this thesis we have discussed various steps to apply the hybridized discontinuous Galerkin methods to a variety of stationary and time-dependent flow problems in an efficient and stable manner. In order to construct such schemes we have discussed a variety of time integrators, shock capturing schemes, an efficient implementation in MATLAB and a new solver for linear systems of equations. We have presented an extensive set of numerical experiments to verify and evaluate the new approaches.

We have discussed implicit general linear methods that are the generalization of multi step and multistage time integrators. This combination allows for methods with higher stage order than classical Runge-Kutta methods. Their application requires the introduction of a startup procedure. The numerical results indicate that the integrators work well and produce almost identical results compared to classical Runge-Kutta method for standard problems. Although none of the studied test cases led to order reduction in the classical time integrators, it is beneficial to have these methods at hand as one can come across such problems also in CFD. Future work can consist of the evaluation of these methods for further stiff problems. Another field of work can be the study of time step adaptation [29, 113].

Another class of time integrators that we have investigated extensively, is implicit multi-derivative time integrators. These methods introduce additional time derivatives in order to realize schemes with higher order of convergence in time. We have discussed two different approaches that use the underlying discretization in space to approximate time derivatives. The first approach was based on a Cauchy-Kowalevski procedure where the PDE was differentiated with respect to time, which allows to replace time by space derivatives. This approach has earlier been studied successfully for methods using explicit multiderivative time integrators [96, 189, 214]. We have observed that the approach introduces severe time step restrictions. This was unexpected because we have used implicit time integration schemes that allow very large time steps when applied to ordinary differential equations. We have carried out a von Neumann stability analysis that confirmed the existence of such time step restrictions when a PDE is solved.

Based on these findings we have devised a second approach in which the semi-discrete version of the PDE would be differentiated in time. This approach leads to fewer additional unknowns that are introduced to the problem and does not introduce unexpected time step restrictions. We have carried out a von Neumann stability analysis that supports these

findings. We have also presented numerical results in one and two space dimensions that confirm these findings. In this work we have presented the new approach solely for linear problems and the numerical results in two space dimensions have been computed with a symmetric interior penalty discontinuous Galerkin method. Thus, obvious fields for future work are the extension to nonlinear problems, for which we have sketched an approach, and the implementation of these methods for hybridized discontinuous Galerkin methods in two space dimensions.

The development of numerical methods greatly benefits from tools that allow for a quick and easy implementation of new schemes and testing of new ideas. Therefore, we have developed an efficient implementation of the HDG method within FESTUNG, an open source framework for MATLAB / GNU Octave. The framework has been extended to incorporate the required hybrid unknowns on element edges and we have incorporated a new way of assembling terms that depend on space-dependent quantities. We have shown numerical results to validate the implementation. Moreover, a hybridized discontinuous Galerkin method has been compared to an upwind discontinuous Galerkin method. It is possible to observe the promised savings in run-time for simulations with large polynomial degree P. We have focused on a scalar, linear convection equation in our work. Future work can focus on nonlinear problems, convection-diffusion equations or systems of equations which have not been studied in the FESTUNG framework so far.

When describing real world flow problems, a common choice are Euler and Navier-Stokes equations. Both equations are nonlinear and thus might lead to non-smooth solutions. In the setting of high order methods, such as the hybridized discontinuous Galerkin methods, it is necessary to introduce a shock capturing scheme to prevent non-physical oscillations to appear. The shock approximation can be improved by an approach to adaptively refine the mesh close to discontinuities as the width of the discontinuities scales with the mesh size at its location. We have adopted the shock capturing scheme by Persson and Peraire [184] that behaves advantageous compared to other shock capturing methods that have been introduced for the HDG methods earlier. We have been able to solve several challenging time-dependent problems. Additionally, we have coupled the shock capturing method to a mesh adaptation procedure that is based on the agglomeration of mesh elements. This allowed for an accurate approximation of discontinuities while the number of degrees of freedom can be reduced. Future work could include the construction of shock capturing methods explicitly making use of the hybrid unknown and the application to further test problems. The mesh adaptation has been studied only on very simple and small meshes and with a simple indicator for refinement. Future work in this area can focus on more complicated meshes, improved integration formulae for elements of arbitrary shape, and more advanced mesh refinement indicators.

The last topic we have focused on was the evaluation of a new solver for the arising linear

system of equations. We have implemented a wrapper that allows for an easy incorporation of the linear solver that does not require any change of the programming interface. Moreover, we were able to apply the solver to nonlinear scalar problems. The results where in good agreement with results obtained by a GMRES method. We have observed problems when applying the solver to the Euler and Navier-Stokes equations. Thus, future work has to focus on these type of equations. We have already suggested that a possible solution would be the tuning of stabilization parameter of the numerical flux function or replacing the numerical flux function.

The extension and application of the HDG method to new types of problems has to be part of future work, too. We are investigating the application of the HDG method to flows with cavitation, i.e. a fluid that is present in liquid and vapor state. A typical application where cavitation can be observed is the injector of a gasoline engine. The gasoline in the injector nozzle is under high pressure. When the nozzle opens and the gasoline starts flowing, the local pressure can drop below the saturation pressure such that the liquid evaporates.

In order to model the cavitation we use the same model as Hickel et al. [110]. In their model they assume that the density ρ is a combined density $\bar{\rho}$ of the vapor phase pressure ρ_{vap} and liquid phase pressure ρ_{liq}

$$\bar{\rho} = \alpha\rho_{\text{vap}} + (1-\alpha)\rho_{\text{liq}}.$$

The parameter α indicates how much of the local volume is present in vapor phase. Thus, we have $\alpha = 0$ in the liquid phase and $\alpha > 0$ in the mixed liquid-vapor phase. For the liquid phase and the mixed liquid-vapor phase suitable models for the pressure

$$p(\bar{\rho}) = \begin{cases} (p_{\text{sat}} + B) \cdot \left(\frac{\bar{\rho}}{\rho_{\text{sat,liq}}}\right)^N - B, & \alpha = 0 \\ p_{\text{sat}} + C \cdot \left(\frac{1}{\rho_{\text{sat,liq}}} - \frac{1}{\bar{\rho}}\right), & 0 < \alpha < 1 \end{cases}$$

and the viscosity

$$\bar{\mu} = (1-\alpha)\left(1 + \frac{5}{2}\alpha\right)\mu_{\text{liq}} + \alpha\mu_{\text{vap}}$$

are formulated. The saturated pressure p_{sat}, saturated density of the liquid phase $\rho_{\text{sat,liq}}$, the viscosities in the liquid μ_{liq} and vapor μ_{vap} phase depend on the fluid studied. The parameters N, C and B have to be chosen accordingly. A big advantage of this approach is its simplicity. It can be used with the Euler or Navier-Stokes equations without any changes in the equations. Only the closures are adapted. Moreover, no interface tracking is necessary as a mean value state is modeled. The modeling of cavitation is stiff due to the big differences in pressure in the different phases. Thus, the approximation of flows with cavitation can greatly benefit from implicit time integration methods and we expect that the HDG method offers advantages over other schemes.

To summarize, we have developed and evaluated important parts that are necessary to make hybridized discontinuous Galerkin a versatile tool for problems from fluid dynamics. We have introduced a number of new time integrators. These offer new features, including additional stability properties, and possibly a more efficient time integration than established methods. Together with the work on an efficient solver for the linear system of equations these are key components to have a stable and efficient method. This is especially important for the application of the HDG methods in three space dimensions where the computational costs due to larger number of degrees of freedom grows extensively. The work on mesh adaptation combined with a shock capturing methods is necessary to approximate flows with shocks. Otherwise the method would become unstable or at least contain non-physical oscillations. A lack of mesh adaptation leads to either an unsatisfying resolution of shocks or unfeasible run-time requirements. The implementation of a hybridized discontinuous Galerkin method in an open source framework for MATLAB / GNU Octave helps to promote the scheme as it easily accessible and well documented. Furthermore, it allows for a simple extension of the HDG method or the application of the HDG method to new PDEs.

A. Coefficients of time integration methods

Several time integrators have been discussed in Sec. 4. For the sake of completeness we want to state the coefficients of the introduced methods if it has not been done in the method description already. The coefficients of BDF methods introduced in Sec. 4.2 can be found in Tab. A.1.

The coefficients of the diagonally implicit Runge-Kutta methods introduced by Alexander [4] are given in Tab. A.2. We do not explicitly state the tableau for the implicit Euler method where every entry is one. The coefficients of the third order method, see Tab. A.2b are obtained by setting $\alpha = 0.43586652150845$ and computing the missing values as

$$\tau = \frac{1}{2}(1+\alpha),\ b_1 = -\frac{1}{4}(6\alpha^2 - 16\alpha + 1),\ b_2 = \frac{1}{4}(6\alpha^2 - 20\alpha + 5).$$

In order to obtain Cash's method an lower order DIRK method is embedded [41]. The coefficients $\hat{b}_i$ for that are given as

$$\hat{b}_1 = \frac{\tau - 0.5}{\tau - \alpha}, \quad \hat{b}_2 = \frac{\alpha - 0.5}{\alpha - \tau}.$$

The coefficients of the fourth order method of Al-Rabeh [191] are stated in Tab. A.3 and the coefficients of Hairer's and Wanner's method are given in Tab. A.4.

The coefficients of the diagonally implicit multistage integration method, see Sec. 5.1, are given in Tab. A.5. This follows the notation of general linear methods in Nordsieck formulation (5.5).

	a_0	a_1	a_2	a_3	a_4
BDF1	1.0	-1.0	0.0	0.0	0.0
BDF2	$\frac{3}{2}$	-2.0	$\frac{1}{2}$	0.0	0.0
BDF3	$\frac{11}{6}$	-3.0	$\frac{3}{2}$	$-\frac{1}{3}$	0.0
BDF4	$\frac{25}{12}$	-4.0	3.0	$-\frac{4}{3}$	$\frac{1}{4}$

Source: Textbook of Dahmen and Reusken [61]

Table A.1.: Coefficients of BDFk-methods up to order 4.

$1-\frac{1}{\sqrt{2}}$	$1-\frac{1}{\sqrt{2}}$	0
1.0	$\frac{1}{\sqrt{2}}$	$1-\frac{1}{\sqrt{2}}$
	$\frac{1}{\sqrt{2}}$	$1-\frac{1}{\sqrt{2}}$

(a) Coefficients of Alexander's DIRK22 method.

α	α	0	0
τ	$\tau-\alpha$	α	0
1	b_1	b_2	α
b_i	b_1	b_2	α
$\hat{b}_i$	$\hat{b}_1$	$\hat{b}_2$	0

(b) Coefficients of Alexander's DIRK33 (without $\hat{b}_i$) and Cash's DIRK33 method.

Table A.2.: Coefficients of Alexander's and Cash's DIRK methods. The coefficients of the three stage method are given in the text.

0.4358665	0.4358665	0	0	0
0.0323722	-0.4034943	0.4358665	0	0
0.9676278	-0.3298751	0.8616364	0.4358665	0
0.5641335	0.5575315	-0.1930865	-0.2361781	0.4358665
b_i	0.3153914	0.1846086	0.1846086	0.3153914
$\hat{b}_i$	0.6307827	0.1413538	0.2278634	0

Table A.3.: Coefficients of Al-Rabeh's DIRK method.

$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	0	0	0
$\frac{11}{20}$	$\frac{17}{50}$	$-\frac{1}{25}$	$\frac{1}{4}$	0	0
$\frac{1}{2}$	$\frac{371}{1360}$	$-\frac{137}{2720}$	$\frac{15}{544}$	$\frac{1}{4}$	0
1	$\frac{25}{24}$	$-\frac{49}{48}$	$\frac{125}{16}$	$\frac{-85}{12}$	$\frac{1}{4}$
b_i	$\frac{25}{24}$	$-\frac{49}{48}$	$\frac{125}{16}$	$-\frac{85}{12}$	$\frac{1}{4}$
$\hat{b}_i$	$\frac{59}{48}$	$-\frac{17}{96}$	$\frac{225}{32}$	$-\frac{85}{12}$	0

Table A.4.: Coefficients of Hairer's and Wanner's DIRK method.

1	1	0
1	1	0
1	0	0

(a) Coefficients of DIMSIM1.

$1-\frac{1}{\sqrt{2}}$	0	1	$\frac{-2+\sqrt{2}}{2}$	0
$\frac{6+2\sqrt{2}}{7}$	$1-\frac{1}{\sqrt{2}}$	1	$\frac{3}{14}\left(\sqrt{2}-4\right)$	$\frac{\sqrt{2}}{2}-1$
$\frac{73-34\sqrt{2}}{28}$	$\frac{2\sqrt{2}-1}{4}$	1	$\frac{10\sqrt{2}-19}{14}$	$\frac{3-2\sqrt{2}}{4}$
0	1	0	0	0
-1	1	0	0	0

(b) Coefficients of DIMSIM2.

$\frac{1}{2}$	0	0	1	$-\frac{3}{2}$	1	$-\frac{5}{12}$
$\frac{5}{4}$	$\frac{1}{2}$	0	1	$-\frac{7}{4}$	$\frac{5}{4}$	$-\frac{5}{8}$
$\frac{7}{5}$	$\frac{4}{5}$	$\frac{1}{2}$	1	$-\frac{17}{10}$	$\frac{7}{5}$	$-\frac{47}{60}$
$\frac{17}{20}$	$\frac{41}{30}$	$\frac{1}{3}$	1	$-\frac{31}{20}$	$\frac{61}{60}$	$-\frac{17}{40}$
0	0	1	0	0	0	0
$\frac{1}{2}$	-2	$\frac{3}{2}$	0	0	0	0
1	-2	1	0	0	0	0

(c) Coefficients of DIMSIM3.

Table A.5.: Coefficients of DIMSIMs used by Jackiewicz [114].

B. Von Neumann stability analysis of Cauchy-Kowalevski multiderivative method

We have observed serious stability issues of the multiderivative methods using the Cauchy-Kowalevski/Lax-Wendroff approach in Sec. 6.2. In order to understand this better we carry out a von Neumann stability analysis for the linear convection equation using finite difference space discretizations.

We solve the linear convection equation

$$\partial_t w + u\partial_x w = 0, \quad u > 0 \tag{6.14 revisited}$$

such that we have $\partial_t w = g(w)$ with $g(w) := -u\partial_x w$. Then, we have

$$\partial_t^2 w = \partial_t \left(-u\partial_x w\right) = -u\partial_x \left(\partial_t w\right) = -u\partial_x \left(-u\partial_x w\right) = u^2\partial_x^2 w = \dot{g}(w)$$

for the time derivative of the right hand side. For simplicity we assume that the equation has been discretized on an infinite domain, e.g. by using periodic boundary conditions. We use a finite difference discretization with mesh size $\Delta x = x_j - x_{j-1}$.

B.1. Approximation using upwind and central differences

We approximate the differential operators by

$$\partial_x w(x_j) \approx \frac{w_j - w_{j-1}}{\Delta x}, \quad \partial_x^2 w(x_j) \approx \frac{w_{j-1} - 2w_j + w_{j+1}}{\Delta x^2}. \tag{6.16 revisited}$$

The first time derivative is approximated using an upwind disretization while the diffusive term is discretized by a central difference. This follows the common procedure when solving PDEs using finite differences. The upwind discretization is chosen to respect the direction of information propagation. The diffusive operator has no dominant propagation direction and thus central differences are a common choice.

B.1.1. Fourth order method

The fourth order method is given by the formula

$$w_h^{n+1} = w_h^n + \frac{\Delta t}{2}\left(g(w_h^n) + g(w_h^{n+1})\right) + \frac{\Delta t^2}{12}\left(\partial_t g(w_h^n) - \partial_t g(w_h^{n+1})\right).$$

B. Von Neumann stability analysis of Cauchy-Kowalevski multiderivative method

The method is applied to the convection equation (6.14)

$$w^{n+1} = w^n - \frac{u\Delta t}{2}\left(\partial_x w^n + \partial_x w^{n+1}\right) + \frac{u^2\Delta t^2}{12}\left(\partial_x^2 w^n - \partial_x^2 w^{n+1}\right) \qquad \text{(6.17 revisited)}$$

and the resulting terms are discretized using the finite differences (6.16)

$$\begin{aligned} w_j^{n+1} = w_j^n &- \frac{u\Delta t}{2\Delta x}\left(w_j^n - w_{j-1}^n + w_j^{n+1} - w_{j-1}^{n+1}\right) \\ &+ \frac{u^2\Delta t^2}{12\Delta x^2}\left(w_{j-1}^n - 2w_j^n + w_{j+1}^n - w_{j-1}^{n+1} + 2w_j^{n+1} - w_{j+1}^{n+1}\right). \end{aligned}$$

We substitute $\nu = \frac{u\Delta t}{\Delta x}$ and obtain

$$\begin{aligned} w_j^{n+1} = w_j^n &- \frac{\nu}{2}\left(w_j^n - w_{j-1}^n + w_j^{n+1} - w_{j-1}^{n+1}\right) \\ &+ \frac{\nu^2}{12}\left(w_{j-1}^n - 2w_j^n + w_{j+1}^n - w_{j-1}^{n+1} + 2w_j^{n+1} - w_{j+1}^{n+1}\right). \end{aligned}$$

For the von Neumann stability analysis we consider data that can be represented by a Fourier mode

$$w_j^n = e^{\mathrm{i}kx_j}, \quad w_{j+1}^n = e^{\mathrm{i}kx_j}e^{\mathrm{i}k\Delta x}, \quad w_{j-1}^n = e^{\mathrm{i}kx_j}e^{-\mathrm{i}k\Delta x}$$

with i being the imaginary unit and k being the wave number. The updated solution is given by

$$w_j^{n+1} = r(k, \Delta x, \Delta t) w_j^n$$

using the amplification factor $r(k, \Delta x, \Delta t)$. Before substituting we reorder

$$\begin{aligned} &w_j^{n+1} + \frac{\nu}{2}\left(w_j^{n+1} - w_{j-1}^{n+1}\right) + \frac{\nu^2}{12}\left(w_{j-1}^{n+1} - 2w_j^{n+1} + w_{j+1}^{n+1}\right) \\ &= w_j^n - \frac{\nu}{2}\left(w_j^n - w_{j-1}^n\right) + \frac{\nu^2}{12}\left(w_{j-1}^n - 2w_j^n + w_{j+1}^n\right) \end{aligned}.$$

such that after substituting the unknown by the Fourier mode and dividing by $e^{\mathrm{i}kx_j}$ one obtains

$$\begin{aligned} r(k, \Delta x, \Delta t)\,(1 \; &+ \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)\Big) \\ &= 1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right) \end{aligned}$$

such that the amplification factor can be expressed as

$$r(k, \Delta x, \Delta t) = \frac{1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)}{1 + \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)}.$$

If the discretization is stable, the Fourier mode may not be amplified, i.e. $|r(k, \Delta x, \Delta t)| \leq 1$. Therefore, we investigate the numerator

$$f(k, \Delta x, \Delta t) = 1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)$$

and the denominator

$$g(k, \Delta x, \Delta t) = 1 + \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)$$

with $r(k, \Delta x, \Delta t) = \frac{f(k,\Delta x,\Delta t)}{g(k,\Delta x,\Delta t)}$. Evaluating $|r(k, \Delta x, \Delta t)|$ needs the absolute value of the complex numbers given by the numerator and denominator. In order to avoid handling the square root resulting from the complex number we analyze $|r(k, \Delta x, \Delta t)|^2$. This leads to the same statement because $|r(k, \Delta x, \Delta t)| \leq 1 \Leftrightarrow |r(k, \Delta x, \Delta t)|^2 \leq 1$ due to the absolute values being positive. First, we replace the Fourier mode by its representation in trigonometric form $e^{\mathrm{i}kx_j} = \cos(kx_j) + \mathrm{i}\sin(kx_j)$. Then, the numerator and denominator are given by

$$\begin{aligned} f(k, \Delta x, \Delta t) =& 1 - \frac{\nu}{2}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12}\left(\cos(k\Delta x) - \mathrm{i}\sin(k\Delta x) - 2 + \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ =& 1 - \frac{\nu}{2}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12}\left(2\cos(k\Delta x) - 2\right) \end{aligned}$$

and

$$\begin{aligned} g(k, \Delta x, \Delta t) =& 1 + \frac{\nu}{2}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12}\left(\cos(k\Delta x) - \mathrm{i}\sin(k\Delta x) - 2 + \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ =& 1 + \frac{\nu}{2}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{12}\left(2\cos(k\Delta x) - 2\right). \end{aligned}$$

Now, we introduce the following substitution

$$y := \cos\left(\frac{k\Delta x}{2}\right) \Rightarrow \cos(k\Delta x) = 2y - 1,\ \sin(k\Delta x) = \pm 2\sqrt{y(1-y)} \qquad \text{(6.20 revisited)}$$

with $y \in [0, 1]$. It allows to write the numerator as

$$\begin{aligned} f(k, \Delta x, \Delta t) =& 1 - \frac{\nu}{2}\left(1 - 2y + 1 \pm \mathrm{i}2\sqrt{y(1-y)}\right) + \frac{\nu^2}{12}\left(4y - 2 - 2\right) \\ =& 1 - \nu\left(1 - y \pm \mathrm{i}\sqrt{y(1-y)}\right) + \frac{\nu^2}{3}\left(y - 1\right) \end{aligned}$$

and the denominator as

$$\begin{aligned} g(k, \Delta x, \Delta t) =& 1 + \frac{\nu}{2}\left(1 - 2y + 1 \pm \mathrm{i}2\sqrt{y(1-y)}\right) + \frac{\nu^2}{12}\left(4y - 2 - 2\right) \\ =& 1 + \nu\left(1 - y \pm \mathrm{i}\sqrt{y(1-y)}\right) + \frac{\nu^2}{3}\left(y - 1\right). \end{aligned}$$

The squared absolute value of the numerator and denominator are given as

$$|f(k,\Delta x,\Delta t)|^2 = \left(1-\nu(1-y)+\frac{\nu^2}{3}(y-1)\right)^2 + \left(\nu\sqrt{y(1-y)}\right)^2$$

and

$$|g(k,\Delta x,\Delta t)|^2 = \left(1+\nu(1-y)+\frac{\nu^2}{3}(y-1)\right)^2 + \left(\nu\sqrt{y(1-y)}\right)^2$$

Now, $|r(k,\Delta x,\Delta t)|^2 \leq 1$ is only fulfilled if $|f(k,\Delta x,\Delta t)|^2 \leq |g(k,\Delta x,\Delta t)|^2$. Thus, we check

$$\begin{aligned} &|f(k,\Delta x,\Delta t)|^2 \leq |g(k,\Delta x,\Delta t)|^2 \\ \Leftrightarrow &\left(1-\nu(1-y)+\frac{\nu^2}{3}(y-1)\right)^2 \leq \left(1+\nu(1-y)+\frac{\nu^2}{3}(y-1)\right)^2. \end{aligned}$$

After expanding the terms, we get to

$$\begin{aligned} &1-2\nu(1-y)+\nu^2(1-y)^2+\frac{1}{9}\nu^4(y-1)^2+\frac{2}{3}\nu^2(y-1)+\frac{2}{3}\nu^3(y-1)^2 \\ \leq &1+\nu^2(1-y)^2+\frac{1}{9}\nu^4(y-1)^2+2\nu(1-y)+\frac{2}{3}\nu^2(y-1)-\frac{2}{3}\nu^3(y-1)^2 \end{aligned}$$

were many terms cancel each other giving

$$-\nu(1-y)+\frac{1}{3}\nu^3(y-1)^2 \leq \nu(1-y)-\frac{1}{3}\nu^3(y-1)^2.$$

Further simplifications lead to

$$\frac{2}{3}\nu^3(1-y)^2 \leq 2\nu(1-y)$$

which obviously holds for $y=1$. Therefore, we assume in the following that $y \neq 1$ such that we can simplify this to

$$\begin{aligned} \frac{2}{3}\nu^3(1-y)^2 &\leq 2\nu(1-y) \\ \Leftrightarrow \nu^2 &\leq \frac{3}{1-y} \end{aligned}$$

and therefore $\nu \leq \sqrt{\frac{3}{1-y}}$, $y \in [0,1)$ as we consider only positive values for ν. From this follows, that the method is not stable for all $y \in [0,1]$ without restrictions regarding the CFL number ν. However, one would expect a stable method if a A-stable time integrator is used.

B.1.2. Third order method

The third order method is given by the formula

$$w_h^{n+1} = w_h^n + \frac{\Delta t}{3}\left(g(w_h^n)+2g(w_h^{n+1})\right) - \frac{\Delta t^2}{6}\partial_t g(w_h^{n+1}).$$

The method is applied to the convection equation (6.14)

$$w^{n+1} = w^n - \frac{u\Delta t}{3}\left(\partial_x w^n + 2\partial_x w^{n+1}\right) - \frac{u^2\Delta t^2}{6}\partial_x^2 w^{n+1}$$

and the resulting terms are discretized using the finite differences (6.16)

$$\begin{aligned} w_j^{n+1} = w_j^n &- \frac{u\Delta t}{3\Delta x}\left(w_j^n - w_{j-1}^n + 2w_j^{n+1} - 2w_{j-1}^{n+1}\right) \\ &- \frac{u^2\Delta t^2}{6\Delta x^2}\left(w_{j-1}^{n+1} - 2w_j^{n+1} + w_{j+1}^{n+1}\right). \end{aligned}$$

We substitute $\nu = \frac{u\Delta t}{\Delta x}$ and obtain

$$\begin{aligned} w_j^{n+1} = w_j^n &- \frac{\nu}{3}\left(w_j^n - w_{j-1}^n + 2w_j^{n+1} - 2w_{j-1}^{n+1}\right) \\ &- \frac{\nu^2}{6}\left(w_{j-1}^{n+1} - 2w_j^{n+1} + w_{j+1}^{n+1}\right). \end{aligned}$$

Again, we reorder such that unknowns at the new time level are on the right hand side

$$\begin{aligned} w_j^{n+1} + \frac{\nu}{3}\left(2w_j^{n+1} - 2w_{j-1}^{n+1}\right) + \frac{\nu^2}{6}\left(w_{j-1}^{n+1} - 2w_j^{n+1} + w_{j+1}^{n+1}\right) \\ = w_j^n - \frac{\nu}{3}\left(w_j^n - w_{j-1}^n\right). \end{aligned}$$

Then, the solution is expressed as Fourier modes

$$\begin{aligned} r(k,\Delta x,\Delta t)e^{\mathrm{i}kx_j}\left(1 + \frac{\nu}{3}\left(2 - 2e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{6}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)\right) \\ = e^{\mathrm{i}kx_j}\left(1 - \frac{\nu}{3}\left(1 - e^{-\mathrm{i}k\Delta x}\right)\right) \end{aligned}$$

and after eliminating $e^{\mathrm{i}kx_j}$ we obtain

$$\begin{aligned} r(k,\Delta x,\Delta t)\left(1 + \frac{\nu}{3}\left(2 - 2e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{6}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right)\right) \\ = 1 - \frac{\nu}{3}\left(1 - e^{-\mathrm{i}k\Delta x}\right). \end{aligned}$$

Again, we define the amplification factor as $r(k,\Delta x,\Delta t) = \frac{f(k,\Delta x,\Delta t)}{g(k,\Delta x,\Delta t)}$. We have

$$\begin{aligned} f(k,\Delta x,\Delta t) =& 1 - \frac{\nu}{3}\left(1 - e^{-\mathrm{i}k\Delta x}\right) \\ g(k,\Delta x,\Delta t) =& 1 + \frac{\nu}{3}\left(2 - 2e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{6}\left(e^{-\mathrm{i}k\Delta x} - 2 + e^{\mathrm{i}k\Delta x}\right) \end{aligned}$$

Using the trigonometric form of the complex numbers gives

$$f(k,\Delta x,\Delta t) = 1 - \frac{\nu}{3}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right)$$

and

$$\begin{aligned} g(k,\Delta x,\Delta t) =& 1+\frac{2\nu}{3}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right) \\ &+\frac{\nu^2}{6}\left(\cos(k\Delta x)-\mathrm{i}\sin(k\Delta x)-2+\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right) \\ =& 1+\frac{2\nu}{3}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right) \\ &+\frac{\nu^2}{6}\left(2\cos(k\Delta x)-2\right). \end{aligned}$$

Then, using the substitution (6.20) one obtains

$$f(k,\Delta x,\Delta t) = 1-\frac{2\nu}{3}\left(1-y\pm\mathrm{i}\sqrt{y(1-y)}\right)$$

and

$$g(k,\Delta x,\Delta t) = 1+\frac{4\nu}{3}\left(1-y\pm\mathrm{i}\sqrt{y(1-y)}\right)+\frac{2\nu^2}{3}\left(1y-1\right).$$

The squared absolute value are

$$|f(k,\Delta x,\Delta t)|^2 = \left(1-\frac{2\nu}{3}\left(1-y\right)\right)^2+\left(\frac{2\nu}{3}\sqrt{y(1-y)}\right)^2$$

and

$$|g(k,\Delta x,\Delta t)|^2 = \left(1+\frac{4\nu}{3}\left(1-y\right)+\frac{2\nu^2}{3}\left(1y-1\right)\right)^2+\left(\frac{4\nu}{3}\sqrt{y(1-y)}\right)^2.$$

Now, one has to check whether $|f(k,\Delta x,\Delta t)|^2 \leq |g(k,\Delta x,\Delta t)|^2$ holds. We get

$$\begin{aligned} &\left(1-\frac{2\nu}{3}\left(1-y\right)\right)^2+\left(\frac{2\nu}{3}\sqrt{y(1-y)}\right)^2 \\ \leq& \left(1+\frac{4\nu}{3}\left(1-y\right)+\frac{2\nu^2}{3}\left(1y-1\right)\right)^2+\left(\frac{4\nu}{3}\sqrt{y(1-y)}\right)^2 \end{aligned}$$

what gives

$$\begin{aligned} &1+\frac{4}{9}\nu^2(1-y)^2-\frac{4}{3}\nu(1-y)+\frac{4}{9}\nu^2y(1-y) \\ \leq& 1+\frac{16}{9}\nu^2(1-y)^2+\frac{4}{9}\nu^4(y-1)^2+\frac{8}{3}\nu(1-y)+\frac{4}{3}\nu^2(y-1) \\ &-\frac{16}{9}\nu^3(1-y)^2+\frac{16}{9}\nu^2y(1-y). \end{aligned}$$

Then, we bring all terms to the right hand side of the equation

$$\begin{aligned} 0 \leq& \frac{12}{9}\nu^2(1-y)^2+\frac{4}{9}\nu^4(y-1)^2+\frac{12}{3}\nu(1-y)+\frac{4}{3}\nu^2(y-1) \\ &-\frac{16}{9}\nu^3(1-y)^2+\frac{12}{9}\nu^2y(1-y). \end{aligned}$$

and multiply with $\frac{3}{4\nu}$ to obtain

$$\begin{aligned} 0 \leq & \nu(1-y)^2 + \frac{1}{3}\nu^3(y-1)^2 + 3(1-y) + \nu(y-1) - \frac{4}{3}\nu^2(1-y)^2 \\ & + \nu y(1-y). \end{aligned}$$

what can be slightly simplified to

$$0 \leq (1-y)\left(3 + \frac{1}{3}\nu^3(1-y) - \frac{4}{3}\nu^2(1-y)\right).$$

Now, the method is stable if the inequality holds for all $y \in [0,1]$. For an implicit method, that is A-stable, we expect this inequality to be true for all ν, as well. The special case $y = 1$ is simple as the right hand side vanishes. For $y \neq 0$, it is sufficient to check

$$0 \leq \underbrace{3 + \left(\frac{1}{3}\nu^3 - \frac{4}{3}\nu^2\right)(1-y)}_{=:h(\nu)}.$$

It turns out that this inequality is not fulfilled for all cases. For $y = 0$, we have plotted $h(x)$, see Fig. 6.3, and one clearly sees that the function becomes negative approximately for $\nu \in (2.30278, 3)$. The bounds have been determined numerically and rounded to five decimal places. Thus, the method is subject to a CFL-like condition for which it is stable.

B.2. Approximation using upwind only

In the previous section we have shown the extended von Neumann analysis for the case where an upwind difference for the convective term and a central difference for the diffusive term have been used. In both cases it does not lead to an unconditionally stable numerical method. In this section, we show the extended version of the analysis presented in Sec. 6.5.1. Therefore, we only use upwind differences

$$\partial_x w(x_j) \approx \frac{w_j - w_{j-1}}{\Delta x}, \quad \partial_x^2 w(x_j) \approx \frac{w_j - 2w_{j-1} + w_{j-2}}{\Delta x^2} \qquad \text{(6.23 revisited)}$$

for both — first and second — space derivatives. The expression for the second derivative can be obtained by applying the upwind finite difference twice

$$\begin{aligned} \partial_x^2 w(x_j) \approx \partial_x \left(\frac{w_j - w_{j-1}}{\Delta x}\right) &\approx \frac{1}{\Delta x}\left(\frac{w_j - w_{j-1}}{\Delta x} - \frac{w_{j-1} - w_{j-2}}{\Delta x}\right) \\ &= \frac{w_j - 2w_{j-1} + w_{j-2}}{\Delta x^2} \end{aligned}$$

and thus the approach presented in Sec. 6.5 follows this idea closely.

B.2.1. Fourth order method

The procedure for the von Neumann stability analysis is the same as already seen in Sec. 6.4, 6.5.1 and the previous section. Therefore, we do not introduce every quantity again.

The fourth order method

$$w_h^{n+1} = w_h^n + \frac{\Delta t}{2}\left(g(w_h^n) + g(w_h^{n+1})\right) + \frac{\Delta t^2}{12}\left(\partial_t g(w_h^n) - \partial_t g(w_h^{n+1})\right).$$

is applied to the convection equation (6.14) such that we can replace the time by space derivatives using the PDE. Then we get

$$w^{n+1} = w^n - \frac{u\Delta t}{2}\left(\partial_x w^n + \partial_x w^{n+1}\right) + \frac{u^2\Delta t^2}{12}\left(\partial_x^2 w^n - \partial_x^2 w^{n+1}\right) \qquad \text{(6.17 revisited)}$$

and the derivatives are approximated using the finite differences (6.23). This leads to

$$\begin{aligned} w^{n+1} = w^n &- \frac{u\Delta t}{2\Delta x}\left(w_j^n - w_{j-1}^n + w_j^{n+1} - w_{j-1}^{n+1}\right) \\ &+ \frac{u^2\Delta t^2}{12\Delta x^2}\left(w_j^n - 2w_{j-1}^n + w_{j-2}^n - w_j^{n+1} + 2w_{j-1}^{n+1} - w_{j-2}^{n+1}\right). \end{aligned}$$

where we substitute $\nu =: \frac{u\Delta t}{\Delta x} \geq 0$ and obtain

$$\begin{aligned} w^{n+1} = w^n &- \frac{\nu}{2}\left(w_j^n - w_{j-1}^n + w_j^{n+1} - w_{j-1}^{n+1}\right) \\ &+ \frac{\nu^2}{12}\left(w_j^n - 2w_{j-1}^n + w_{j-2}^n - w_j^{n+1} + 2w_{j-1}^{n+1} - w_{j-2}^{n+1}\right). \end{aligned}$$

We reorder the terms such that unknowns at the new time level $n+1$ are on one side and the terms of the known time level n are on the right hand side

$$\begin{aligned} &w^{n+1} + \frac{\nu}{2}\left(w_j^{n+1} - w_{j-1}^{n+1}\right) + \frac{\nu^2}{12}\left(w_j^{n+1} - 2w_{j-1}^{n+1} + w_{j-2}^{n+1}\right) \\ =&w^n - \frac{\nu}{2}\left(w_j^n - w_{j-1}^n\right) + \frac{\nu^2}{12}\left(w_j^n - 2w_{j-1}^n + w_{j-2}^n\right). \end{aligned}$$

For the von Neumann stability analysis we consider data that can be represented by a Fourier mode

$$w_j^n = e^{\mathrm{i}kx_j}, \quad w_{j-1}^n = e^{\mathrm{i}kx_j}e^{-\mathrm{i}k\Delta x}, \quad w_{j-2}^n = e^{\mathrm{i}kx_j}e^{-2\mathrm{i}k\Delta x} \tag{B.1}$$

with i being the imaginary unit and k being the wave number. The updated solution is given by

$$w_j^{n+1} = r(k, \Delta x, \Delta t)e^{\mathrm{i}kx_j}$$

using the amplification factor $r(k, \Delta x, \Delta t)$. Then, we obtain

$$\begin{aligned} &r(k, \Delta x, \Delta t)\left[1 + \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(1 - 2e^{-\mathrm{i}k\Delta x} + e^{-2\mathrm{i}k\Delta x}\right)\right] \\ =&1 - \frac{\nu}{2}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{12}\left(1 - 2e^{-\mathrm{i}k\Delta x} + e^{-2\mathrm{i}k\Delta x}\right) \end{aligned}$$

where we have already eliminated the term $e^{\mathrm{i}kx_j}$. Again, we define the amplification factor as $r(k,\Delta x,\Delta t) \coloneqq \frac{f(k,\Delta x,\Delta t)}{g(k,\Delta x,\Delta t)}$. We have

$$\begin{aligned} f(k,\Delta x,\Delta t) =& 1-\frac{\nu}{2}\left(1-e^{-\mathrm{i}k\Delta x}\right)+\frac{\nu^2}{12}\left(1-2e^{-\mathrm{i}k\Delta x}+e^{-2\mathrm{i}k\Delta x}\right)\\ g(k,\Delta x,\Delta t) =& 1+\frac{\nu}{2}\left(1-e^{-\mathrm{i}k\Delta x}\right)+\frac{\nu^2}{12}\left(1-2e^{-\mathrm{i}k\Delta x}+e^{-2\mathrm{i}k\Delta x}\right). \end{aligned}$$

Using the trigonometric form of the complex numbers gives

$$\begin{aligned} f(k,\Delta x,\Delta t) = 1 &- \frac{\nu}{2}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right)\\ &+\frac{\nu^2}{12}\left(1-2\cos(k\Delta x)+2\mathrm{i}\sin(k\Delta x)+\cos(2k\Delta x)-\mathrm{i}\sin(2k\Delta x)\right) \end{aligned}$$

and

$$\begin{aligned} g(k,\Delta x,\Delta t) = 1 &+ \frac{\nu}{2}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right)\\ &+\frac{\nu^2}{12}\left(1-2\cos(k\Delta x)+2\mathrm{i}\sin(k\Delta x)+\cos(2k\Delta x)-\mathrm{i}\sin(2k\Delta x)\right). \end{aligned}$$

We use the double angle formulae

$$\begin{aligned} \sin(2\theta) &= 2\sin(\theta)\cos(\theta)\\ \cos(2\theta) &= \cos^2(\theta)-\sin^2(\theta) = 1-2\sin^2(\theta) = 2\cos^2(\theta)-1 \end{aligned} \qquad \text{(6.24 revisited)}$$

in order to simplify the expressions. It allows to express the numerator as

$$\begin{aligned} f(k,\Delta x,\Delta t) = 1 &- \frac{\nu}{2}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right)\\ &+\frac{\nu^2}{12}\left(1-2\cos(k\Delta x)+2\mathrm{i}\sin(k\Delta x)+2\cos^2(k\Delta x)-1-2\mathrm{i}\sin(k\Delta x)\cos(k\Delta x)\right) \end{aligned}$$

and the denominator as

$$\begin{aligned} g(k,\Delta x,\Delta t) = 1 &+ \frac{\nu}{2}\left(1-\cos(k\Delta x)+\mathrm{i}\sin(k\Delta x)\right)\\ &+\frac{\nu^2}{12}\left(1-2\cos(k\Delta x)+2\mathrm{i}\sin(k\Delta x)+2\cos^2(k\Delta x)-1-2\mathrm{i}\sin(k\Delta x)\cos(k\Delta x)\right). \end{aligned}$$

The trigonometric function are substituted by

$$y \coloneqq \cos\left(\frac{k\Delta x}{2}\right) \Rightarrow \cos(k\Delta x) = 2y-1,\ \sin(k\Delta x) = \pm 2\sqrt{y(1-y)} \qquad \text{(6.20 revisited)}$$

with $y \in [0,1]$. This eventually leads to

$$\begin{aligned} f(k,\Delta x,\Delta t) &= 1-\frac{\nu}{2}\left(2-2y\pm\mathrm{i}2\sqrt{y(1-y)}\right)\\ &\quad+\frac{\nu^2}{12}\left(1-4y+2\pm 4\mathrm{i}\sqrt{y(1-y)}+8y^2-8y+2-1\mp 4\mathrm{i}(2y-1)\sqrt{y(1-y)}\right)\\ &= 1-\nu\left(1-y\right)\mp\nu\mathrm{i}\sqrt{y(1-y)}\\ &\quad+\frac{\nu^2}{3}\left(1-3y+2y^2\pm\mathrm{i}\sqrt{y(1-y)}\mp\mathrm{i}(2y-1)\sqrt{y(1-y)}\right) \end{aligned}$$

and

$$\begin{aligned} g(k,\Delta x,\Delta t) &= 1+\frac{\nu}{2}\left(2-2y\pm 2\mathrm{i}\sqrt{y(1-y)}\right) \\ &\quad+\frac{\nu^2}{12}\left(1-4y+2\pm 4\mathrm{i}\sqrt{y(1-y)}+8y^2-8y+2\mp 4\mathrm{i}(2y-1)\sqrt{y(1-y)}\right) \\ &= 1+\nu\,(1-y)\pm\nu\mathrm{i}\sqrt{y(1-y)} \\ &\quad+\frac{\nu^2}{3}\left(1-3y+2y^2\pm\mathrm{i}\sqrt{y(1-y)}\mp\mathrm{i}(2y-1)\sqrt{y(1-y)}\right). \end{aligned}$$

The squared absolute values introduce a lot of additional terms in contrast to the discretization using central finite differences for the diffusion term. We extract the term $y(1-y)$ from the squared absolute value of the complex part and slightly simplify it so we get

$$\begin{aligned} |f(k,\Delta x,\Delta t)|^2 &= \left(1-\nu\,(1-y)+\frac{\nu^2}{3}\left(1-3y+2y^2\right)\right)^2 \\ &\quad+\left(\mp\nu\sqrt{y(1-y)}+\frac{\nu^2}{3}\left(\pm\sqrt{y(1-y)}\mp(2y-1)\sqrt{y(1-y)}\right)\right)^2 \\ &= \left(1-\nu\,(1-y)+\frac{\nu^2}{3}\left(1-3y+2y^2\right)\right)^2+y(1-y)\left(\mp\nu+\frac{\nu^2}{3}\left(\pm 1\mp(2y-1)\right)\right)^2 \\ &= \left(1-\nu\,(1-y)+\frac{\nu^2}{3}\left(1-3y+2y^2\right)\right)^2+y(1-y)\left(\mp\nu+\frac{\nu^2}{3}\left(\pm 2\mp 2y\right)\right)^2. \end{aligned}$$

and by

$$\begin{aligned} |g(k,\Delta x,\Delta t)|^2 &= \left(1+\nu(1-y)+\frac{\nu^2}{3}(1-3y+2y^2)\right)^2 \\ &\quad+\left(\pm\nu\sqrt{y(1-y)}+\frac{\nu^2}{3}(\pm\sqrt{y(1-y)}\mp(2y-1)\sqrt{y(1-y)})\right)^2 \\ &= \left(1+\nu(1-y)+\frac{\nu^2}{3}(1-3y+2y^2)\right)^2+y(1-y)\left(\pm\nu+\frac{\nu^2}{3}(\pm 1\mp(2y-1))\right)^2 \\ &= \left(1+\nu(1-y)+\frac{\nu^2}{3}(1-3y+2y^2)\right)^2+y(1-y)\left(\pm\nu+\frac{\nu^2}{3}(\pm 2\mp 2y)\right)^2. \end{aligned}$$

The previous simplification make it easier to check whether $|f(k,\Delta x,\Delta t)|^2 \leq |g(k,\Delta x,\Delta t)|^2$ holds. We get

$$\begin{aligned} &\left(1-\nu\,(1-y)+\frac{\nu^2}{3}\left(1-3y+2y^2\right)\right)^2+y(1-y)\left(\mp\nu+\frac{\nu^2}{3}\left(\pm 2\mp 2y\right)\right)^2 \\ \leq &\left(1+\nu(1-y)+\frac{\nu^2}{3}(1-3y+2y^2)\right)^2+y(1-y)\left(\pm\nu+\frac{\nu^2}{3}(\pm 2\mp 2y)\right)^2 \end{aligned}$$

where both sides have to be expanded. This leads to the lengthy expression

$$\begin{aligned}
&1+\nu^2(1-y)^2+\frac{\nu^4}{9}\left(1-3y+2y^2\right)^2\\
&-2\nu(1-y)+\frac{2}{3}\nu^2\left(1-3y+2y^2\right)-\frac{2}{3}\nu^3(1-y)\left(1-3y+2y^2\right)\\
&+y(1-y)\left(\nu^2+\frac{\nu^4}{9}(\pm2\mp2y)^2+\frac{2\nu^2}{3}(\pm\nu)(\mp2\mp2y)\right)\\
\leq&1+\nu^2(1-y)^2+\frac{\nu^4}{9}\left(1-3y+2y^2\right)^2\\
&+2\nu(1-y)+\frac{2}{3}\nu^2\left(1-3y+2y^2\right)+\frac{2}{3}\nu^3(1-y)\left(1-3y+2y^2\right)\\
&+y(1-y)\left(\nu^2+\frac{\nu^4}{9}(\pm2\mp2y)^2+\frac{2\nu^2}{3}(\mp\nu)(\pm2\mp2y)\right).
\end{aligned}$$

where several terms — like all squared terms — cancel and in the last term of the left hand side we can extract -1 to give

$$\begin{aligned}
&-2\nu(1-y)-\frac{2}{3}\nu^3(1-y)\left(1-3y+2y^2\right)-y(1-y)\frac{2\nu^2}{3}(\pm\nu)(\pm2\mp2y)\\
\leq&+2\nu(1-y)+\frac{2}{3}\nu^3(1-y)\left(1-3y+2y^2\right)+y(1-y)\frac{2\nu^2}{3}(\pm\nu)(\pm2\mp2y).
\end{aligned}$$

and we can bring all terms of the left hand side to the right hand and simplify the expression further. This gives

$$\begin{aligned}
&0\leq4\nu(1-y)+\frac{4}{3}\nu^3(1-y)\left(1-3y+2y^2\right)+\frac{4\nu^2}{3}y(1-y)(\mp\nu)(\pm2\mp2y)\\
\Leftrightarrow\;&0\leq4\nu(1-y)+\frac{4}{3}\nu^3(1-y)\left(1-3y+2y^2\right)+\frac{4\nu^3}{3}y(1-y)(2-2y)\\
\Leftrightarrow\;&0\leq4\nu(1-y)+\frac{4\nu^3}{3}(1-y)\left(1-3y+2y^2+2y-2y^2\right)\\
\Leftrightarrow\;&0\leq4\nu(1-y)+\frac{2\nu^3}{3}(1-y)\,(1-y)\\
\Leftrightarrow\;&0\leq4\nu(1-y)+\frac{2\nu^3}{3}\,(1-y)^2\,.
\end{aligned}$$

This inequality is fulfilled for all y and ν as we require $\nu\geq0$ and $y\in[0,1]$. It follows that the scheme is stable for all time step sizes.

B.2.2. Third order method

We repeat the exact same steps as for the fourth order method also for the third order method

$$w_h^{n+1}=w_h^n+\frac{\Delta t}{3}\left(g(w_h^n)+2g(w_h^{n+1})\right)-\frac{\Delta t^2}{6}\partial_t g(w_h^{n+1}).$$

B. Von Neumann stability analysis of Cauchy-Kowalevski multiderivative method

The method is applied to the convection equation (6.14) such that we can replace the time by space derivatives (6.15) to get

$$w^{n+1} = w^n - \frac{u\Delta t}{3}\left(\partial_x w^n + 2\partial_x w^{n+1}\right) - \frac{u^2\Delta t^2}{6}\partial_x^2 w^{n+1}$$

and the resulting terms are discretized by upwind finite differences (6.23)

$$\begin{aligned} w_j^{n+1} = w_j^n &- \frac{u\Delta t}{3\Delta x}\left(w_j^n - w_{j-1}^n + 2w_j^{n+1} - 2w_{j-1}^{n+1}\right) \\ &- \frac{u^2\Delta t^2}{6\Delta x^2}\left(w_j^{n+1} - 2w_{j-1}^{n+1} + w_{j-2}^{n+1}\right). \end{aligned}$$

We introduce $\nu := \frac{u\Delta t}{\Delta x} \geq 0$ and obtain

$$w_j^{n+1} = w_j^n - \frac{\nu}{3}\left(w_j^n - w_{j-1}^n + 2w_j^{n+1} - 2w_{j-1}^{n+1}\right) - \frac{\nu^2}{6}\left(w_j^{n+1} - 2w_{j-1}^{n+1} + w_{j-2}^{n+1}\right).$$

Again, we reorder such that unknowns at the new time level are on the right hand side

$$\begin{aligned} &w_j^{n+1} + \frac{2}{3}\nu\left(w_j^{n+1} - w_{j-1}^{n+1}\right) + \frac{\nu^2}{6}\left(w_j^{n+1} - 2w_{j-1}^{n+1} + w_{j-2}^{n+1}\right) \\ =&w_j^n - \frac{\nu}{3}\left(w_j^n - w_{j-1}^n\right). \end{aligned}$$

Then, the solution is expressed using Fourier modes (B.1) so we obtain

$$\begin{aligned} r(k,\Delta x,\Delta t)\left(1 + \frac{2}{3}\nu\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{6}\left(1 - 2e^{-\mathrm{i}k\Delta x} + e^{-2\mathrm{i}k\Delta x}\right)\right) \\ = 1 - \frac{\nu}{3}\left(1 - e^{-\mathrm{i}k\Delta x}\right) \end{aligned}$$

after eliminating $e^{\mathrm{i}kx_j}$. Again, we define the amplification factor as $r(k,\Delta x,\Delta t) := \frac{f(k,\Delta x,\Delta t)}{g(k,\Delta x,\Delta t)}$. We have

$$\begin{aligned} f(k,\Delta x,\Delta t) =& 1 - \frac{\nu}{3}\left(1 - e^{-\mathrm{i}k\Delta x}\right) \\ g(k,\Delta x,\Delta t) =& 1 + \frac{2\nu}{3}\left(1 - e^{-\mathrm{i}k\Delta x}\right) + \frac{\nu^2}{6}\left(1 - 2e^{-\mathrm{i}k\Delta x} + e^{-2\mathrm{i}k\Delta x}\right) \end{aligned}$$

Using the trigonometric form of the complex numbers gives

$$f(k,\Delta x,\Delta t) = 1 - \frac{\nu}{3}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right)$$

and

$$\begin{aligned} g(k,\Delta x,\Delta t) =& 1 + \frac{2\nu}{3}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{6}\left(1 - 2\cos(k\Delta x) + 2\mathrm{i}\sin(k\Delta x) + \cos(2k\Delta x) - \mathrm{i}\sin(2k\Delta x)\right) \\ =& 1 + \frac{2\nu}{3}\left(1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)\right) \\ &+ \frac{\nu^2}{6}\left(-2\cos(k\Delta x) + 2\mathrm{i}\sin(k\Delta x) + \cos^2(k\Delta x) - 2\mathrm{i}\sin(k\Delta x)\cos(k\Delta x)\right) \end{aligned}$$

where we have used the double angle formulae (6.24). We use the substitution (6.20) to obtain

$$f(k,\Delta x,\Delta t) = 1 - \frac{2\nu}{3}\left(1 - y \pm \mathrm{i}\sqrt{y(1-y)}\right)$$

and

$$\begin{aligned} g(k,\Delta x,\Delta t) = 1 &+ \frac{4\nu}{3}\left(1 - y \pm \mathrm{i}\sqrt{y(1-y)}\right) \\ &+ \frac{2\nu^2}{3}\left(1 - 3y + 2y^2 \pm \mathrm{i}\sqrt{y(1-y)} \mp \mathrm{i}\sqrt{y(1-y)}(2y-1)\right). \end{aligned}$$

The squared absolute values are

$$|f(k,\Delta x,\Delta t)|^2 = \left(1 - \frac{2\nu}{3}(1-y)\right)^2 + \left(\frac{2\nu}{3}\sqrt{y(1-y)}\right)^2$$

and

$$\begin{aligned} |g(k,\Delta x,\Delta t)|^2 = &\left(1 + \frac{4\nu}{3}(1-y) + \frac{2\nu^2}{3}\left(1 - 3y + 2y^2\right)\right)^2 \\ &+ y(1-y)\left(\pm\frac{4\nu}{3} + \frac{2\nu^2}{3}(\mp 2y \pm 2)\right)^2. \end{aligned}$$

We have to check again whether $|f(k,\Delta x,\Delta t)|^2 \leq |g(k,\Delta x,\Delta t)|^2$ holds. This gives

$$\begin{aligned} \left(1 - \frac{2\nu}{3}(1-y)\right)^2 + \left(\frac{2\nu}{3}\sqrt{y(1-y)}\right)^2 \leq &\left(1 + \frac{4\nu}{3}(1-y) + \frac{2\nu^2}{3}\left(1 - 3y + 2y^2\right)\right)^2 \\ &+ y(1-y)\left(\pm\frac{4\nu}{3} + \frac{2\nu^2}{3}(\mp 2y \pm 2)\right)^2 \end{aligned}$$

and after expanding the terms we have

$$\begin{aligned} &1 + \frac{4}{9}\nu^2(1-y)^2 - \frac{4}{3}\nu(1-y) + \frac{4}{9}\nu^2 y(1-y) \\ \leq &1 + \frac{16}{9}\nu^2(1-y)^2 + \frac{4}{9}\nu^4(1-3y+2y^2)^2 \\ &+ \frac{8}{3}\nu(1-y) + \frac{4}{3}\nu^2(1-3y+2y^2) + \frac{16}{9}\nu^3(1-y)(1-3y+2y^2) \\ &+ y(1-y)\left(\frac{16}{9}\nu^2 + \frac{4}{9}\nu^4(\mp 2y \pm 2)^2 + \frac{16}{9}\nu^3(\pm 1)(\mp 2y \pm 2)\right). \end{aligned}$$

The following simplifications

$$(\mp 2y \pm 2)^2 = (2y-2)^2 = (-2y+2)^2$$

and

$$(\pm 1)(\mp 2y \pm 2) = -2y + 2$$

hold so we can simplify the inequality to get

$$\begin{aligned}0 \leq & \frac{4}{3}\nu^2(1-y)^2 + \frac{4}{9}\nu^4(1-3y+2y^2)^2 \\ & + 4\nu(1-y) + \frac{4}{3}\nu^2(1-3y+2y^2) + \frac{16}{9}\nu^3(1-y)(1-3y+2y^2) \\ & + y(1-y)\left(\frac{4}{3}\nu^2 + \frac{4}{9}\nu^4(2y-2)^2 + \frac{16}{9}\nu^3(-2y+2)\right).\end{aligned}$$

We can further simplify the terms

$$\frac{16}{9}\nu^3(1-y)(1-3y+2y^2) + y(1-y)\frac{16}{9}\nu^3(-2y+2) = \frac{16}{9}\nu^3(1-y)^2$$

and

$$\frac{4}{3}\nu^2(1-3y+2y^2) + \frac{4}{3}\nu^2 y(1-y) = \frac{4}{3}\nu^2(1-y)^2.$$

We can rewrite the inequality with these transformation to obtain

$$\begin{aligned}0 \leq & \frac{4}{3}\nu^2(1-y)^2 + \frac{4}{9}\nu^4(1-3y+2y^2)^2 \\ & + 4\nu(1-y) + \frac{4}{3}\nu^2(1-y)^2 + \frac{16}{9}\nu^3(1-y)^2 \\ & + \frac{4}{9}\nu^4 y(1-y)(2y-2)^2\end{aligned}$$

what we do need not simplify further. This inequality is fulfilled for all y and ν as we require $\nu \geq 0$ and $y \in [0,1]$. It follows that the scheme is stable for all time step sizes.

C. Compact formulation of two-point collocation methods using the new approach

The new approach of approximating the time derivatives introduced in [212] allows to represent the methods in two different ways. We call the way it has been done in the publication and explained in Sec. 6.5 the 'naive' way in this section. In order to sketch the idea we reuse the formulation in section for a linear advection-diffusion problem without source term

$$\partial_t w_{\mathrm{DG}} = \mathrm{A}_{\mathrm{DG}} w_{\mathrm{DG}}.$$

Then, the two-derivative two-point collocation method (6.34) has been written as

$$\frac{W_1^{n+1} - W_1^n}{\Delta t} = \mathrm{A}_{\mathrm{DG}} \left(\alpha_1 W_1^n - \beta_1 W_1^{n+1}\right) + \Delta t \mathrm{A}_{\mathrm{DG}} \left(\alpha_2 W_2^n - \beta_2 W_2^{n+1}\right).$$

This formulation uses the auxiliary unknowns for the time derivatives $W_i, i \geq 1$ and the fact that the next highest time derivative is given by applying the operator A_{DG} once more, i.e. $W_{i+1} = \mathrm{A}_{\mathrm{DG}} W_i$. In this case, the linear system of equations $\mathrm{A}x = b$ that has to be solved is given by matrix A (in block-matrix notation)

$$\begin{pmatrix} \mathrm{A}_{\sigma\sigma} & 0 & \mathrm{A}_{\sigma w} & 0 & \mathrm{A}_{\sigma\lambda} & 0 \\ 0 & \mathrm{A}_{\sigma\sigma} & 0 & \mathrm{A}_{\sigma w} & 0 & \mathrm{A}_{\sigma\lambda} \\ \beta_1 \Delta t \mathrm{A}_{w\sigma} & \beta_2 \Delta t^2 \mathrm{A}_{w\sigma} & \mathrm{M}_\varphi + \beta_1 \Delta t \mathrm{A}_{ww} & \beta_2 \Delta t^2 \mathrm{A}_{ww} & \beta_1 \Delta t \mathrm{A}_{w\lambda} & \beta_2 \Delta t^2 \mathrm{A}_{w\lambda} \\ \mathrm{A}_{w\sigma} & 0 & \mathrm{A}_{ww} & \mathrm{M}_\varphi & \mathrm{A}_{w\lambda} & 0 \\ \mathrm{A}_{\lambda\sigma} & 0 & \mathrm{A}_{\lambda w} & 0 & \mathrm{A}_{\lambda\lambda} & 0 \\ 0 & \mathrm{A}_{\lambda\sigma} & 0 & \mathrm{A}_{\lambda w} & 0 & \mathrm{A}_{\lambda\lambda} \end{pmatrix} \tag{C.1}$$

where we follow the naming scheme close to the one introduced in Section 3.3.2, but we state the mass matrix M_φ referring to the mass matrix obtained from (3.15b) explicitly. The matrices $\mathrm{A}_{(\cdot\cdot)}$ are still sparse matrices. Moreover, matrix A consists of several zero blocks and is thus sparse as well. Although the system of equations is larger, the static condensation process can still be carried out. However, the resulting system is coupled in M hybrid unknowns as we have to introduce one for each approximation of a time derivative. Moreover, many of the block matrices used to assemble A appear more than once. This might be exploited for an efficient implementation as described in the FESTUNG paper series [80, 124, 195].

The vector of unknowns and right hand side are given by

$$x = \left(\Sigma_1^{n+1}, \Sigma_2^{n+1}, W_1^{n+1}, W_2^{n+1}, \Lambda_1^{n+1}, \Lambda_2^{n+1}\right)^{\mathrm{T}}, \quad b = (0, 0, R_{\mathrm{Col}}, 0, 0, 0)^{\mathrm{T}}. \tag{C.2}$$

where the entry in the right hand side vector is given by

$$\begin{aligned} R_{\text{Col}} =& \mathrm{M}_{\varphi} W_1^n \\ &+ \alpha_1 \Delta t (\mathrm{A}_{w\sigma} \Sigma_1^n + \mathrm{A}_{ww} W_1^n + \mathrm{A}_{w\lambda} \Lambda_1^n) \\ &+ \alpha_2 \Delta t^2 (\mathrm{A}_{w\sigma} \Sigma_2^n + \mathrm{A}_{ww} W_2^n + \mathrm{A}_{w\lambda} \Lambda_2^n). \end{aligned} \tag{C.3}$$

As one can see the block-matrices can also be reused for the assembly of the right hand side.

The more *compact* way to represent is to use the newly introduced representations of the time derivatives. It allows us to write the time integrator as

$$\frac{W_1^{n+1} - W_1^n}{\Delta t} = \alpha_1 W_2^n - \beta_1 W_2^{n+1} + \Delta t \mathrm{A}_{\text{DG}} \left(\alpha_2 W_2^n - \beta_2 W_2^{n+1} \right)$$

where only the highest order time derivative has to be expressed by the operator A_{DG}. All other time derivatives are available by the introduced auxiliary unknowns. Following this approach, the matrix A can be expressed

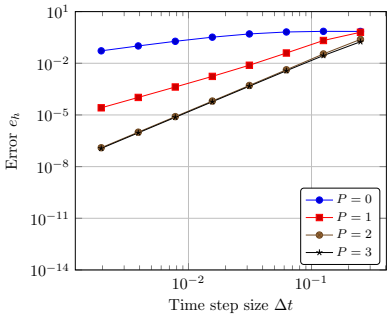
$$\begin{pmatrix} \mathrm{A}_{\sigma\sigma} & 0 & \mathrm{A}_{\sigma w} & 0 & \mathrm{A}_{\sigma\lambda} & 0 \\ 0 & \mathrm{A}_{\sigma\sigma} & 0 & \mathrm{A}_{\sigma w} & 0 & \mathrm{A}_{\sigma\lambda} \\ 0 & \beta_2 \Delta t^2 \mathrm{A}_{w\sigma} & 0 & \beta_1 \Delta t \mathrm{M}_{\varphi} + \beta_2 \Delta t^2 \mathrm{A}_{ww} & 0 & \beta_2 \Delta t^2 \mathrm{A}_{w\lambda} \\ \mathrm{A}_{w\sigma} & 0 & \mathrm{A}_{ww} & \mathrm{M}_{\varphi} & \mathrm{A}_{w\lambda} & 0 \\ \mathrm{A}_{\lambda\sigma} & 0 & \mathrm{A}_{\lambda w} & 0 & \mathrm{A}_{\lambda\lambda} & 0 \\ 0 & \mathrm{A}_{\lambda\sigma} & 0 & \mathrm{A}_{\lambda w} & 0 & \mathrm{A}_{\lambda\lambda} \end{pmatrix} \tag{C.4}$$

such that all matrices in the third row containing β_1 can be replaced by the mass matrix M_{φ}. The right hand side simplifies accordingly to

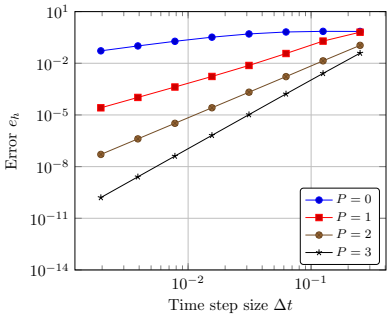
$$\begin{aligned} R_{\text{Col}} =& \mathrm{M}_{\varphi} W_1^n + \alpha_1 \Delta t \mathrm{M}_{\varphi} W_2^n \\ &+ \alpha_2 \Delta t^2 (\mathrm{A}_{w\sigma} \Sigma_2^n + \mathrm{A}_{ww} W_2^n + \mathrm{A}_{w\lambda} \Lambda_2^n). \end{aligned} \tag{C.5}$$

Lower time derivatives access the representation of the time derivative and only need the assembly of a mass matrix. It leads to a sparser system than the initial formulation and may lead to massive reductions in runtime because of the smaller assembly costs, memory requirements and possibly easier solving process of the linear system of equations. The approach becomes more beneficial for an increasing number of time derivatives M. Moreover, it promises to be an especially beneficial approach for nonlinear problems, where the Jacobian of the DG discretization is needed. Together with the multiderivative approach additional derivatives of the nonlinear functions are needed to the application of the new approach and the application of a nonlinear solver like Newton's method.

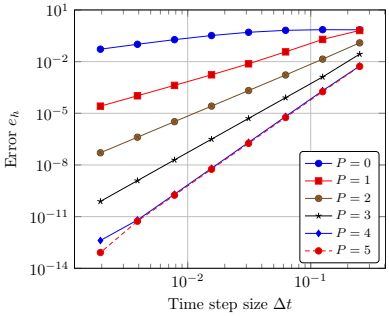
In Fig. C.1, we present the results of the linear advection test case used to verify the approach, see Sec. 6.5, for the compact notation of the two-point collocation methods. The results are nearly indistinguishable from the results obtained in the "non-compact" formulation, see Fig.6.5. The most apparent differences can be seen for large order in time and space where one is close to the machine precision/precision of the linear solver employed



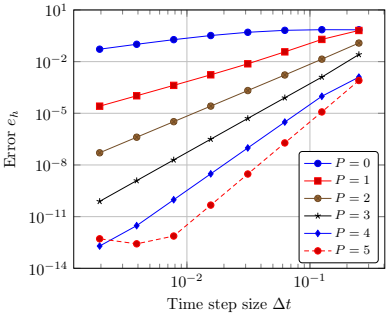
(a) Third order method ($k = 1, l = 2$).



(b) Fourth order method ($k = 2, l = 2$).



(c) Fifth order method ($k = 2, l = 3$).



(d) Sixth order method ($k = 3, l = 3$).

Figure C.1.: Numerical results for the advection equation with parameters $u = 1$, $t_{\text{final}} = 1.0$. Time step size for the coarsest triangular mesh, consisting of one element, was chosen to be $\Delta t = 0.5$. Two-point collocation schemes has been used in *compact* formulation for time integration.

by MATLAB. It seems beneficial to use the compact formulation if one aims for reduced computational costs.

Remark 10 *Most of the matrices can be reused efficiently only in the purely linear case and the unknowns must be reordered such that a local static condensation process can be carried out.*

Bibliography

[1] V. Aizinger. "A Geometry Independent Slope Limiter for the Discontinuous Galerkin Method". In: *Computational Science and High Performance Computing IV: The 4th Russian-German Advanced Research Workshop, Freiburg, Germany, October 12 to 16, 2009*. Ed. by E. Krause, Y. Shokin, M. Resch, D. Kröner, and N. Shokina. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 207–217.

[2] V. Aizinger, A. Kosík, D. Kuzmin, and B. Reuter. "Anisotropic slope limiting for discontinuous Galerkin methods". In: *International Journal for Numerical Methods in Fluids* 84.9 (2017), pp. 543–565.

[3] V. Aizinger, D. Kuzmin, and L. Korous. "Scale separation in fast hierarchical solvers for discontinuous Galerkin methods". In: *Applied Mathematics and Computation* 266 (2015), pp. 838–849.

[4] R. Alexander. "Diagonally Implicit Runge-Kutta Methods for Stiff O.D.E.'s". In: *SIAM Journal on Numerical Analysis* 14.14 (1977), pp. 1006–1021.

[5] M. S. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. "The FEniCS Project Version 1.5". In: *Archive of Numerical Software* 3.100 (2015).

[6] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Third edition. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[7] J. Anderson. *Fundamentals of Aerodynamics*. Sixth Edition. McGraw-Hill Education International Edition, 2017.

[8] P. F. Antonietti, P. Houston, X. Hu, M. Sarti, and M. Verani. "Multigrid algorithms for *hp*-version interior penalty discontinuous Galerkin methods on polygonal and polyhedral meshes". In: *Calcolo* 54.4 (2017), pp. 1169–1198.

[9] D. N. Arnold and F. Brezzi. "Mixed and nonconforming finite element methods: implementation, postprocessing and error estimates". In: *RAIRO Modélisation Mathématique et Analyse Numérique* 19.1 (1985), pp. 7–32.

[10] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. "Unified Analysis of Discontinuous Galerkin Methods for elliptic problems". In: *SIAM Journal of Numerical Analysis* 39.5 (2002), pp. 1749–1779.

[11] S. Badia, J. Bonilla, and A. Hierro. "Differentiable monotonicity-preserving schemes for discontinuous Galerkin methods on arbitrary meshes". In: *Computer Methods in Applied Mechanics and Engineering* 320 (2017), pp. 582–605.

[12] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. *PETSc Users Manual*. Tech. rep. ANL-95/11 - Revision 3.4. Argonne National Laboratory, 2013.

[13] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. *PETSc Web page*. 2013.

[14] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries". In: *Modern Software Tools in Scientific Computing*. Ed. by E. Arge, A. M. Bruaset, and H. P. Langtangen. Birkhäuser Press, 1997, pp. 163–202.

[15] W. Bangerth, R. Hartmann, and G. Kanschat. "deal.II – a General Purpose Object Oriented Finite Element Library". In: *ACM Transactions on Mathematical Software* 33.4 (2007).

[16] G. E. Barter and D. L. Darmofal. "Shock capturing with PDE-based artificial viscosity for DGFEM: Part I. Formulation". In: *Journal of Computational Physics* 229.5 (2010), pp. 1810–1827.

[17] G. Barter and D. Darmofal. "Shock Capturing with Higher-Order, PDE-Based Artificial Viscosity". In: *18th AIAA Computational Fluid Dynamics Conference*. AIAA 2007-3823. Miami, FL, 2007.

[18] F. Bassi, L. Botti, A. Colombo, D. D. Pietro, and P. Tesini. "On the flexibility of agglomeration based physical space discontinuous Galerkin discretizations". In: *Journal of Computational Physics* 231.1 (2012), pp. 45–65.

[19] F. Bassi, L. Botti, A. Colombo, and S. Rebay. "Agglomeration based discontinuous Galerkin discretization of the Euler and Navier–Stokes equations". In: *Computers and Fluids* 61 (2012), pp. 77–85.

[20] F. Bassi and S. Rebay. "A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations". In: *Journal of Computational Physics* 131.2 (1997), pp. 267–279.

[21] F. Bassi and S. Rebay. "High-Order Accurate Discontinuous Finite Element Solution of the 2D Euler Equations". In: *Journal of Computational Physics* 138.2 (1997), pp. 251–285.

[22] C. E. Baumann and J. T. Oden. "A discontinuous hp finite element method for convection-diffusion problems". In: *Computer Methods in Applied Mechanics and Engineering* 175.3 (1999), pp. 311–341.

[23] C. E. Baumann and J. T. Oden. "A discontinuous hp finite element method for the Euler and Navier-Stokes equations". In: *International Journal for Numerical Methods in Fluids* 31.1 (1999), pp. 79–95.

[24] C. Bi and V. Ginting. "Two-Grid Discontinuous Galerkin Method for Quasi-Linear Elliptic Problems". In: *Journal of Scientific Computing* 49.3 (2011), pp. 311–331.

[25] F. Brezzi, J. Douglas, and L. D. Marini. "Two Families of Mixed Finite Elements for Second Order Elliptic Problems". In: *Numerische Mathematik* 47.2 (1985), pp. 217–235.

[26] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*. Springer Series in Computational Mathematics New York, 1991.

[27] T. Bui-Thanh. "From Godunov to a unified hybridized discontinuous Galerkin framework for partial differential equations". In: *Journal of Computational Physics* 295 (2015), pp. 114–146.

[28] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. 2nd Edition. John Wiley & Sons, 2008.

[29] J. C. Butcher and Z. Jackiewicz. "A Reliable Error Estimation for Diagonally Implicit Multistage Integration Methods". In: *BIT Numerical Mathematics* 41.4 (2001), pp. 656–665.

[30] J. Butcher. "Diagonally-implicit multi-stage integration methods". In: *Applied Numerical Mathematics* 11.5 (1993), pp. 347–363.

[31] J. Butcher. "General linear methods". In: *Computers & Mathematics with Applications* 31.4 (1996), pp. 105–112.

[32] J. Butcher. "General linear methods". In: *Acta Numerica* 15 (2006), pp. 157–256.

[33] J. Butcher, P. Chartier, and Z. Jackiewicz. "Nordsieck representation of DIMSIMs". In: *Numerical Algorithms* 16.2 (1997), pp. 209–230.

[34] J. Butcher and H. Podhaisky. "On error estimation in general linear methods for stiff ODEs". In: *Applied Numerical Mathematics* 56.3 (2006), pp. 345–357.

[35] J. Butcher. "General linear methods for ordinary differential equations". In: *Mathematics and Computers in Simulation* 79.6 (2009), pp. 1834–1845.

[36] X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri. "Newton-Krylov-Schwarz Methods in CFD". In: *Numerical methods for the Navier-Stokes equations: Proceedings of the International Workshop Held at Heidelberg, October 25–28, 1993*. Ed. by F.-K. Hebeker, R. Rannacher, and G. Wittum. Wiesbaden: Vieweg+Teubner Verlag, 1994, pp. 17–30.

[37] A. Cangiani, E. H. Georgoulis, and P. Houston. "hp-Version discontinuous Galerkin methods on polygonal and polyhedral meshes". In: *Mathematical Models and Methods in Applied Sciences* 24.10 (2014), pp. 2009–2041.

[38] C. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. D. Grazia, S. Yakovlev, J.-E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. Kirby, and S. Sherwin. "Nektar++: An open-source spectral/hp element framework". In: *Computer Physics Communications* 192 (2015), pp. 205–219.

[39] A. Cardone, Z. Jackiewicz, J. H. Verner, and B. Welfert. "Order conditions for general linear methods". In: *Journal of Computational and Applied Mathematics* 290 (2015), pp. 44–64.

[40] J.-R. Carlson. *Inflow/outflow boundary conditions with application to FUN3D*. Tech. rep. NASA/TM-20110217181. NASA, 2011.

[41] J. R. Cash. "Diagonally Implicit Runge-Kutta Formulae with Error Estimates". In: *Journal of the Institute of Mathematics and its Applications* 24.3 (1979), pp. 293–301.

[42] R. P. K. Chan and A. Y. J. Tsai. "On explicit two-derivative Runge-Kutta methods". In: *Numerical Algorithms* 53.2 (2010), pp. 171–194.

[43] Y. Chen, B. Cockburn, and B. Dong. "Superconvergent HDG methods for linear, stationary, third-order equations in one-space dimension". In: *Mathematics of Computation* 85.302 (2015), pp. 2715–2742.

[44] A. Christophe, S. Descombes, and S. Lanteri. "An implicit hybridized discontinuous Galerkin method for the 3D time-domain Maxwell equations". In: *Applied Mathematics and Computation* 319 (2018), pp. 395–408.

[45] Cockburn, Bernardo and Shu, Chi-Wang. "The Runge-Kutta local projection P^1-discontinuous-Galerkin finite element method for scalar conservation laws". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 25.3 (1991), pp. 337–361.

[46] B. Cockburn, B. Dong, and J. Guzmán. "A Hybridizable and Superconvergent Discontinuous Galerkin Method for Biharmonic Problems". In: *Journal of Scientific Computing* 40.1 (2009), pp. 141–187.

[47] B. Cockburn, S.-Y. Lin, and C.-W. Shu. "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws III: One-dimensional Systems". In: *Journal of Computational Physics* 84.1 (1989), pp. 90–113.

[48] B. Cockburn. "Static Condensation, Hybridization, and the Devising of the HDG Methods". In: *Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations*. Ed. by G. R. Barrenechea, F. Brezzi, A. Cangiani, and E. H. Georgoulis. Cham: Springer International Publishing, 2016, pp. 129–177.

[49] B. Cockburn and J. Gopalakrishnan. "A Characterization of Hybridized Mixed Methods for Second Order Elliptic Problems". In: *SIAM Journal on Numerical Analysis* 42.1 (2004), pp. 283–301.

[50] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. "Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems". In: *SIAM Journal on Numerical Analysis* 47.2 (2009), pp. 1319–1365.

[51] B. Cockburn, J. Guzmán, S.-C. Soon, and H. K. Stolarski. "An Analysis of the Embedded Discontinuous Galerkin Method for Second-Order Elliptic Problems". In: *SIAM Journal on Numerical Analysis* 47.4 (2009), pp. 2686–2707.

[52] B. Cockburn, S. Hou, and C.-W. Shu. "The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV: The multidimensional case". In: *Mathematics of Computation* 54 (1990), pp. 545–581.

[53] B. Cockburn, G. Kanschat, I. Perugia, and D. Schötzau. "Superconvergence of the Local Discontinuous Galerkin Method for Elliptic Problems on Cartesian Grids". In: *SIAM Journal on Numerical Analysis* 39.1 (2001), pp. 264–285.

[54] B. Cockburn, G. E. Karniadakis, and C.-W. Shu. "The Development of Discontinuous Galerkin Methods". In: *Discontinuous Galerkin Methods: Theory, Computation and Applications*. Ed. by B. Cockburn, G. E. Karniadakis, and C.-W. Shu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 3–50.

[55] B. Cockburn and C.-W. Shu. "Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems". In: *Journal of Scientific Computing* 16.3 (2001), pp. 173–261.

[56] B. Cockburn and C.-W. Shu. "The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems". In: *SIAM Journal on Numerical Analysis* 35.6 (1998), pp. 2440–2463.

[57] B. Cockburn and C.-W. Shu. "The Runge-Kutta Discontinuous Galerkin Method for Conservation Laws V". In: *Journal of Computational Physics* 141.2 (1998), pp. 199–224.

[58] B. Cockburn and C.-W. Shu. "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework". In: *Mathematics of Computation* 52.186 (1989), pp. 411–435.

[59] S. Congreve. "Two-Grid *hp*-Version discontinuous Galerkin Finite Element Methods for Quasilinear PDEs". PhD thesis. The University of Nottingham, 2013.

[60] C. F. Curtiss and J. O. Hirschfelder. "Integration of stiff equations". In: *Proceedings of the National Academy of Sciences of the United States of America* 38 (1952), pp. 235–243.

[61] W. Dahmen and A. Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Verlag Berlin Heidelberg, 2006.

[62] D. A. Di Pietro and A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*. Springer-Verlag Berlin Heidelberg, 2012.

[63] V. A. Dobrev, R. D. Lazarov, P. S. Vassilevski, and L. T. Zikatanov. "Two-level preconditioning of discontinuous Galerkin approximations of second-order elliptic equations". In: *Numerical Linear Algebra with Applications* 13.9 (2006), pp. 753–770.

[64] B. Dong, J. Jiang, and Y. Chen. "Optimally convergent hybridizable discontinuous Galerkin method for fifth-order Korteweg-de Vries type equations". In: *ArXiv e-prints* (2017). arXiv: 1710.07734 [math.NA].

[65] B. Dong. "Optimally Convergent HDG Method for Third-Order Korteweg–de Vries Type Equations". In: *Journal of Scientific Computing* 73.2 (2017), pp. 712–735.

[66] J. Douglas and T. Dupont. "Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods". In: *Computing Methods in Applied Sciences: Second International Symposium December 15–19,1975*. Ed. by R. Glowinski and J. L. Lions. Berlin, Heidelberg: Springer Berlin Heidelberg, 1976, pp. 207–216.

[67] M. Dumbser and C.-D. Munz. "Arbitrary High Order discontinuous Galerkin Schemes". In: *Numerical Methods for Hyperbolic and Kinetic Problems*. Ed. by S. Cordier, T. Goudon, M. Gutnic, and E. Sonnendrucker. IRMA Series in Mathematics and Theoretical Problems. EMS Publishing House, 2005, pp. 295–333.

[68] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring. *GNU Octave version 4.2.0 manual: a high-level interactive language for numerical computations*. 2016. URL: `http://www.gnu.org/software/octave/doc/interpreter`.

[69] H. Egger and J. Schöberl. "A Hybrid Mixed Discontinuous Galerkin Finite Element Method for Convection-Diffusion Problems". In: *IMA Journal of Numerical Analysis* 30.4 (2010), pp. 1206–1234.

[70] H. Egger and C. Waluga. "A Hybrid Discontinuous Galerkin Method for Darcy-Stokes Problems". In: *Domain Decomposition Methods in Science and Engineering XX*. Ed. by R. Bank, M. Holst, O. Widlund, and J. Xu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 663–670.

[71] A. F. Emery. "An evaluation of several differencing methods for inviscid fluid flow problems". In: *Journal of Computational Physics* 2.3 (1968), pp. 306–331.

[72] P. Fernandez, N. Nguyen, and J. Peraire. "The hybridized Discontinuous Galerkin method for Implicit Large-Eddy Simulation of transitional turbulent flows". In: *Journal of Computational Physics* 336 (2017), pp. 308–329.

[73] P. Fernandez, N. C. Nguyen, and J. Peraire. In: American Institute of Aeronautics and Astronautics, 2018. Chap. A physics-based shock capturing method for unsteady laminar and turbulent flows.

[74] K. J. Fidkowski and D. L. Darmofal. "Review of Output-Based Error Estimation and Mesh Adaptation in Computational Fluid Dynamics". In: *AIAA Journal* 49.4 (2011), pp. 673–694.

[75] K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal. "p-Multigrid solution of high-order Discontinuous Galerkin discretizations of the compressible Navier-Stokes equations". In: *Journal of Computational Physics* 207.1 (2005), pp. 92–113.

[76] N. Fischer. "A parallel hybridized discontinuous Galerkin method on distributed memory systems for time-dependent flow problems". MA thesis. RWTH Aachen University, 2014.

[77] A. Fosso P., H. Deniau, N. Lamarque, and T. Poinsot. "Comparison of outflow boundary conditions for subsonic aeroacoustic simulations". In: *International Journal for Numerical Methods in Fluids* 68.10 (2012), pp. 1207–1233.

[78] F. Frank and C. Thiele. *Inexact hierarchical scale separation*. 2017. URL: `https://github.com/lephlaux/ihss`.

[79] F. Frank, B. Reuter, and V. Aizinger. *FESTUNG – The Finite Element Simulation Toolbox for UNstructured Grids*. 2017. URL: `http://www.math.fau.de/FESTUNG`.

[80] F. Frank, B. Reuter, V. Aizinger, and P. Knabner. "FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part I: Diffusion operator". In: *Computers & Mathematics with Applications* 70.1 (2015), pp. 11–46.

[81] R. Frank, J. Schneid, and C. W. Ueberhuber. "Order Results for Implicit Runge-Kutta Methods Applied to Stiff Systems". In: *SIAM Journal on Numerical Analysis* 22.3 (1985), pp. 515–534.

[82] Z. Fu, L. F. Gatica, and F.-J. Sayas. "Algorithm 949: MATLAB Tools for HDG in Three Dimensions". In: *ACM Transactions on Mathematical Software* 41.3 (2015).

[83] G. N. Gatica and F. A. Sequeira. "Analysis of the HDG method for the Stokes–Darcy coupling". In: *Numerical Methods for Partial Differential Equations* 33.3 (2017), pp. 885–917.

[84] L. F. Gatica and F. A. Sequeira. "A priori and a posteriori error analyses of an HDG method for the Brinkman problem". In: *Computers & Mathematics with Applications* 75.4 (2018), pp. 1191–1212.

[85] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1971.

[86] E. W. Gekeler. "On implicit Runge-Kutta methods with higher derivatives". In: *BIT Numerical Mathematics* 28.4 (1988), pp. 809–816.

[87] E. Gekeler and R. Widmann. "On the order conditions of Runge-Kutta methods with higher derivatives". In: *Numerische Mathematik* 50.2 (1986), pp. 183–203.

[88] E. H. Georgoulis. "Discontinuous Galerkin Methods for Linear Problems: An Introduction". In: *Approximation Algorithms for Complex Systems: Proceedings of the 6th International Conference on Algorithms for Approximation, Ambleside, UK, 31st August - 4th September 2009*. Ed. by E. H. Georgoulis, A. Iske, and J. Levesley. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 91–126.

[89] G. Giorgiani, S. Fernández-Méndez, and A. Huerta. "Hybridizable discontinuous Galerkin p-adaptivity for wave propagation problems". In: *International Journal for Numerical Methods in Fluids* 72.12 (2013), pp. 1244–1262.

[90] G. Giorgiani, D. Modesto, S. Fernández-Méndez, and A. Huerta. "High-order continuous and discontinuous Galerkin methods for wave problems". In: *International Journal for Numerical Methods in Fluids* 73.10 (2013), pp. 883–903.

[91] E. Godlewski and P.-A. Raviart. *Numerical Approximation of Hyperbolic Systems of Conservation Laws*. Springer-Verlag New York, 1996.

[92] A. Gopinath and A. Jameson. *Application of the Time Spectral Method to Periodic Unsteady Vortex Sheeding*. AIAA Paper 06-0449. 2006.

[93] R. Griesmaier and P. Monk. "Discretization of the Wave Equation Using Continuous Elements in Time and a Hybridizable Discontinuous Galerkin Method in Space". In: *Journal of Scientific Computing* 58.2 (2014), pp. 472–498.

[94] J.-L. Guermond, R. Pasquetti, and B. Popov. "Entropy viscosity method for nonlinear conservation laws". In: *Journal of Computational Physics* 230.11 (2011), pp. 4248–4267.

[95] J.-L. Guermond and B. Popov. "Viscous Regularization of the Euler Equations and Entropy Principles". In: *SIAM Journal on Applied Mathematics* 74.2 (2014), pp. 284–305.

[96] W. Guo, J.-M. Qiu, and J. Qiu. "A new Lax–Wendroff discontinuous Galerkin method with superconvergence". In: *Journal of Scientific Computing* 65.1 (2015), pp. 299–326.

[97] R. J. Guyan. "Reduction of stiffness and mass matrices". In: *AIAA Journal* 3.2 (1965), pp. 380–380.

[98] S. Güzey, B. Cockburn, and H. K. Stolarski. "The embedded discontinuous Galerkin method: application to linear shell problems". In: *International Journal for Numerical Methods in Engineering* 70.7 (2007), pp. 757–790.

[99] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I, Nonstiff Problems*. Vol. 8. Berlin Heidelberg: Springer-Verlag, 1993.

[100] E. Hairer and G. Wanner. "Multistep-multistage-multiderivative methods for ordinary differential equations". In: *Computing* 11.3 (1973), pp. 287–303.

[101] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*. Vol. 14. Berlin Heidelberg: Springer-Verlag, 1991.

[102] R. Hartmann and P. Houston. "Symmetric Interior Penalty DG Methods for the Compressible Navier–Stokes Equations II: Goal–Oriented A Posteriori Error Estimation". In: *International Journal of Numerical Analysis & Modeling* 3.2 (2006), pp. 141–162.

[103] R. Hartmann. "Adaptive Discontinuous Galerkin methods with shock-capturing for the compressible Navier-Stokes equations". In: *International Journal for Numerical Methods in Fluids* 51.9-10 (2006), pp. 1131–1156.

[104] R. Hartmann. *Numerical Analysis of Higher Order Discontinuous Galerkin Finite Element Methods*. Ed. by H. Deconinck. 2008. URL: `http://elib.dlr.de/57074/`.

[105] R. Hartmann and P. Houston. "Adaptive Discontinuous Galerkin Finite Element Methods for the Compressible Euler Equations". In: *Journal of Computational Physics* 183.2 (2002), pp. 508–532.

[106] R. Hartmann and P. Houston. "Symmetric Interior Penalty DG Methods for the Compressible Navier-Stokes Equations I: Method Formulation". In: *International Journal of Numerical Analysis and Modeling* 3.1 (2006), pp. 1–20.

[107] R. D. Henderson. "Details of the Drag Curve Near the Onset of Vortex Shedding". In: *Physics of Fluids* 7 (1995), pp. 2102–2104.

[108] M. Herlihy and N. Shavit. *The Art of Multiprocessor Programming*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[109] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. 1st. Springer Science & Business Media, 2007.

[110] S. Hickel, M. Mihatsch, and S. Schmidt. "Implicit Large Eddy Simulation of Cavitation in Micro Channel Flows". In: *Proceedings of WIMRC 3rd International Cavitation Forum 2011*. 2011.

[111] C. Hirsch. *Numerical Computation of Internal and External Flows, Volume 2: Computational Methods for Inviscid and Viscous Flows*. John Wiley & Sons, 1990.

[112] A. Huerta, A. Angeloski, X. Roca, and J. Peraire. "Efficiency of high-order elements for continuous and discontinuous Galerkin methods". In: *International Journal for Numerical Methods in Engineering* 96.9 (2013), pp. 529–560.

[113] Z. Jackiewicz. "Construction and Implementation of General Linear Methods for Ordinary Differential Equations: A Review". In: *Journal of Scientific Computing* 25.1 (2005), pp. 29–49.

[114] Z. Jackiewicz. "Implementation of DIMSIMs for Stiff Differential Systems". In: *Appl. Numer. Math.* 42.1-3 (2002), pp. 251–267.

[115] J. Jaffre, C. Johnson, and A. Szepessy. "Convergence of the discontinuous Galerkin finite element method for hyperbolic conservation laws". In: *Mathematical Models and Methods in Applied Sciences* 5.3 (1995), pp. 367–386.

[116] A. Jameson. "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings". In: *AIAA Paper 1991-1596* (1991).

[117] A. Jameson. In: American Institute of Aeronautics and Astronautics, 2015. Chap. Application of Dual Time Stepping to Fully Implicit Runge Kutta Schemes for Unsteady Flow Calculations.

[118] A. Jameson. "Evaluation of Fully Implicit Runge Kutta Schemes for Unsteady Flow Calculations". In: *Journal of Scientific Computing* 73.2 (2017), pp. 819–852.

[119] A. Jaust and J. Schütz. "A temporally adaptive hybridized discontinuous Galerkin method for time-dependent compressible flows". In: *Computers and Fluids* 98 (2014), pp. 177–185.

[120] A. Jaust and J. Schütz. "General linear methods for time-dependent PDEs". In: *Theory, Numerics and Applications of Hyperbolic Problems II*. Ed. by C. Klingenberg and M. Westdickenberg. in press.

[121] A. Jaust, J. Schütz, and D. C. Seal. "Multiderivative time-integrators for the hybridized discontinuous Galerkin method". In: *Conference Proceedings of the YIC GACM 2015*. Ed. by S. Elgeti and J.-W. Simon. Publication Server of RWTH Aachen University, 2015. URL: `https://publications.rwth-aachen.de/record/480970/files/ProceedingsYIC-GACM-ACCES.pdf`.

[122] A. Jaust, J. Schütz, and M. Woopen. "A Hybridized discontinuous Galerkin Method for Unsteady Flows with Shock-Capturing". In: *AIAA Paper 2014-2781* (2014).

[123] A. Jaust. "A Hybridized Discontinuous Galerkin Method for Time-Dependent Compressible Flows". MA thesis. RWTH Aachen University, 2013.

[124] A. Jaust, B. Reuter, V. Aizinger, J. Schütz, and P. Knabner. "FESTUNG: A MATLAB / GNU Octave toolbox for the discontinuous Galerkin method. Part III: Hybridized discontinuous Galerkin (HDG) formulation". In: *Computers and Mathematics with Applications* 2018.12 (2018), pp. 4505–4533.

[125] A. Jaust, J. Schütz, and V. Aizinger. "An efficient linear solver for the hybridized discontinuous Galerkin method". In: *PAMM* 16.1 (2016), pp. 845–846.

[126] A. Jaust, J. Schütz, and D. C. Seal. "Implicit multistage two-derivative discontinuous Galerkin schemes for viscous conservation laws". In: *Journal of Scientific Computing* 69.2 (2016), pp. 866–891.

[127] A. Jaust, J. Schütz, and M. Woopen. "An HDG Method for Unsteady Compressible Flows". English. In: *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014*. Ed. by R. M. Kirby, M. Berzins, and J. S. Hesthaven. Vol. 106. Springer International Publishing, 2015, pp. 267–274.

[128] G.-S. Jiang and C.-W. Shu. "Efficient Implementation of Weighted ENO Schemes". In: *Journal of Computational Physics* 126.1 (1996), pp. 202–228.

[129] H. Kabaria, A. J. Lew, and B. Cockburn. "A hybridizable discontinuous Galerkin formulation for non-linear elasticity". In: *Computer Methods in Applied Mechanics and Engineering* 283 (2015), pp. 303–329.

[130] S. Kang, F. X. Giraldo, and T. Bui-Thanh. "IMEX HDG-DG: a coupled implicit hybridized discontinuous Galerkin (HDG) and explicit discontinuous Galerkin (DG) approach for shallow water systems". In: *ArXiv e-prints* (2017). arXiv: `1711.02751 [cs.CE]`.

[131] G. Karypis and V. Kumar. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs". In: *SIAM Journal on Scientific Computing* 20.1 (1998), pp. 359–392.

[132] R. M. Kirby, S. J. Sherwin, and B. Cockburn. "To CG or to HDG: A Comparative Study". In: *Journal of Scientific Computing* 51.1 (2012), pp. 183–212.

[133] A. Klöckner, T. Warburton, and J. S. Hesthaven. "Viscous Shock Capturing in a Time-Explicit Discontinuous Galerkin Method". eng. In: *Mathematical Modelling of Natural Phenomena* 6.3 (2011), pp. 57–83.

[134] M. Kronbichler and W. A. Wall. "A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers". In: *ArXiv e-prints* (2016). arXiv: `1611.03029 [math.NA]`.

[135] D. Kuzmin. "Hierarchical slope limiting in explicit and implicit discontinuous Galerkin methods". In: *Journal of Computational Physics* 257.Part B (2014), pp. 1140–1162.

[136] P. D. Lax. "Hyperbolic systems of conservation laws II". In: *Communications on Pure and Applied Mathematics* 10.4 (1957), pp. 537–566.

[137] P. D. Lax. "The Formation and Decay of Shock Waves". In: *The American Mathematical Monthly* 79.3 (1972), pp. 227–241.

[138] P. D. Lax. "Weak solutions of nonlinear hyperbolic equations and their numerical computation". In: *Communications on Pure and Applied Mathematics* 7.1 (1954), pp. 159–193.

[139] P. Lax and B. Wendroff. "Systems of conservation laws". In: *Communications on Pure and Applied Mathematics* 13.2 (1960), pp. 217–237.

[140] B. van Leer. "Towards the ultimate conservative difference scheme III. Upstream-centered finite-difference schemes for ideal compressible flow". In: *Journal of Computational Physics* 23.3 (1977), pp. 263–275.

[141] B. van Leer. "Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme". In: *Journal of Computational Physics* 14.4 (1974), pp. 361–370.

[142] B. van Leer. "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method". In: *Journal of Computational Physics* 32.1 (1979), pp. 101–136.

[143] R. J. LeVeque. *Finite-Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2004.

[144] R. J. LeVeque. "High-Resolution Conservative Algorithms for Advection in Incompressible Flow". In: *SIAM Journal on Numerical Analysis* 33.2 (1996), pp. 627–665.

[145] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Ed. by O. E. Lanford. 2nd. Birkhäuser Verlag, 1992.

[146] L. Li, S. Lanteri, and R. Perrussel. "A Hybridizable Discontinuous Galerkin Method Combined to a Schwarz Algorithm for the Solution of 3D Time-harmonic Maxwell's Equation". In: *Journal of Computational Physics* 256 (2014), pp. 563–581.

[147] A. Logg, K.-A. Mardal, G. N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[148] H. Luo, J. D. Baum, and R. Löhner. "A discontinuous Galerkin method based on a Taylor basis for the compressible flows on arbitrary grids". In: *Journal of Computational Physics* 227.20 (2008), pp. 8875–8893.

[149] H. Luo, J. D. Baum, and R. Löhner. "A Hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids". In: *Journal of Computational Physics* 225.1 (2007), pp. 686–713.

[150] H. Luo, J. D. Baum, and R. Löhner. "A p-multigrid Discontinuous Galerkin Method for the Euler Equations on Unstructured Grids". In: *Journal of Computational Physics* 211.2 (2006), pp. 767–783.

[151] K. Mardal, T. K. Nilssen, and G. A. Staff. "Order-Optimal Preconditioners for Implicit Runge–Kutta Schemes Applied to Parabolic PDEs". In: *SIAM Journal on Scientific Computing* 29.1 (2007), pp. 361–375.

[152] G. May, F. Iacono, and A. Jameson. "A hybrid multilevel method for high-order discretization of the Euler equations on unstructured meshes". In: *Journal of Computational Physics* 229.10 (2010), pp. 3938–3956.

[153] P. D. McCormack and L. Crane. *Physical fluid dynamics*. Academic Press, 1973.

[154] D. Moro, N. C. Nguyen, and J. Peraire. "Navier-Stokes Solution Using Hybridizable Discontinuous Galerkin methods". In: *20th AIAA Computational Fluid Dynamics Conference*. AIAA 2011-3407. Honolulu, Hawaii, 2011.

[155] D. Moro, N. C. Nguyen, and J. Peraire. "Dilation-based shock capturing for high-order methods". In: *International Journal for Numerical Methods in Fluids* 82 (2016), pp. 398–416.

[156] H. M. Mott-Smith. "The Solution of the Boltzmann Equation for a Shock Wave". In: *Physical Review* 82.6 (1951), pp. 885–892.

[157] S. E. Mousavi, H. Xiao, and N. Sukumar. "Generalized Gaussian quadrature rules on arbitrary polygons". In: *International Journal for Numerical Methods in Engineering* 82.1 (2010), pp. 99–113.

[158] MPI Forum. *MPI: A Message-Passing Interface Standard. Version 2.2*. 2009.

[159] S. Muralikrishnan, M.-B. Tran, and T. Bui-Thanh. "An Improved Iterative HDG Approach for Partial Differential Equations". In: *ArXiv e-prints* (2017). arXiv: `1711.01175 [math.NA]`.

[160] S. Muralikrishnan, M.-B. Tran, and T. Bui-Thanh. "iHDG: An Iterative HDG Framework for Partial Differential Equations". In: *SIAM Journal on Scientific Computing* 39.5 (2017), S782–S808.

[161] S. Natarajan, S. Bordas, and D. R. Mahapatra. "Numerical integration over arbitrary polygonal domains based on Schwarz–Christoffel conformal mapping". In: *International Journal for Numerical Methods in Engineering* 80.1 (2009), pp. 103–134.

[162] J. von Neumann and R. D. Richtmyer. "A Method for the Numerical Calculation of Hydrodynamic Shocks". In: *Journal of Applied Physics* 21.3 (1950), pp. 232–237.

[163] N. C. Nguyen and J. Peraire. *An Adaptive Shock-Capturing HDG Method for Compressible Flows*. AIAA Paper 2011-3060. 20th AIAA Computational Fluid Dynamics Conference, 2011.

[164] N. C. Nguyen and J. Peraire. "An Adaptive Shock-Capturing HDG Method for Compressible Flows". In: *20th AIAA Computational Fluid Dynamics Conference*. AIAA 2011-3060. Honolulu, Hawaii, 2011.

[165] N. C. Nguyen and J. Peraire. "Hybridizable discontinuous Galerkin methods for partial differential equations in continuum mechanics". In: *Journal of Computational Physics* 231 (2012), pp. 5955–5988.

[166] N. C. Nguyen, J. Peraire, and B. Cockburn. "A class of embedded discontinuous Galerkin methods for computational fluid dynamics". In: *Journal of Computational Physics* 302 (2015), pp. 674–692.

[167] N. C. Nguyen, J. Peraire, and B. Cockburn. "An implicit high-order hybridizable Discontinuous Galerkin method for linear convection-diffusion equations". In: *Journal of Computational Physics* 228 (2009), pp. 3232–3254.

[168] N. C. Nguyen, J. Peraire, and B. Cockburn. "An implicit high-order hybridizable Discontinuous Galerkin method for nonlinear convection-diffusion equations". In: *Journal of Computational Physics* 228 (2009), pp. 8841–8855.

[169] N. C. Nguyen, J. Peraire, and B. Cockburn. "High-order implicit hybridizable discontinuous Galerkin methods for acoustics and elastodynamics". In: *Journal of Computational Physics* 230 (2011), pp. 3695–3718.

[170] N. C. Nguyen, J. Peraire, and B. Cockburn. "Hybridizable discontinuous Galerkin methods for the time-harmonic Maxwell's equations". In: *Journal of Computational Physics* 230.19 (2011), pp. 7151–7175.

[171] N. C. Nguyen, P.-O. Persson, and J. Peraire. "RANS solutions using high order discontinuous Galerkin methods". In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. Vol. 914. AIAA 2007-914. 2007.

[172] N. Nguyen, P.-O. Persson, and J. Peraire. In: American Institute of Aeronautics and Astronautics, 2007. Chap. RANS Solutions Using High Order Discontinuous Galerkin Methods.

[173] A. Nigro, C. D. Bartolo, A. Crivellini, and F. Bassi. "Second derivative time integration methods for discontinuous Galerkin solutions of unsteady compressible flows". In: *Journal of Computational Physics* 350 (2017), pp. 493–517.

[174] A. Nordsieck. "On numerical integration of ordinary differential equations". In: *Mathematics of Computation* 16 (1962), pp. 22–49.

[175] L. Pan, J. Li, and K. Xu. "A Few Benchmark Test Cases for Higher-Order Euler Solvers". In: *Numerical Mathematics: Theory, Methods and Applications* 10.4 (2017), pp. 711–736.

[176] W. Pazner and P.-O. Persson. "Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods". 2018. URL: `http://www.sciencedirect.com/science/article/pii/S0021999117307830`.

[177] W. Pazner and P.-O. Persson. "Stage-parallel fully implicit Runge-Kutta solvers for discontinuous Galerkin fluid simulations". In: *Journal of Computational Physics* 335 (2017), pp. 700–717.

[178] J. Peraire, N. C. Nguyen, and B. Cockburn. "An embedded discontinuous Galerkin method for the compressible Euler and Navier-Stokes equations". In: *20th AIAA Computational Fluid Dynamics Conference 2011*. 2011.

[179] J. Peraire and P.-O. Persson. "The Compact Discontinuous Galerkin (CDG) Method for Elliptic Problems". In: *SIAM Journal on Scientific Computing* 30.4 (2008), pp. 1806–1824.

[180] P.-O. Persson and J. Peraire. "Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier–Stokes Equations". In: *SIAM Journal on Scientific Computing* 30.6 (2008), pp. 2709–2733.

[181] P.-O. Persson. In: American Institute of Aeronautics and Astronautics, 2012. Chap. High-Order Navier-Stokes Simulations Using a Sparse Line-Based Discontinuous Galerkin Method.

[182] P.-O. Persson. "Shock capturing for high-order discontinuous Galerkin simulation of transient flow problems". In: *21st AIAA Computational Fluid Dynamics Conference*. AIAA 2013-3061. 2013. URL: http://dx.doi.org/10.2514/6.2013-3061.

[183] P.-O. Persson and J. Peraire. In: American Institute of Aeronautics and Astronautics, 2006. Chap. An Efficient Low Memory Implicit DG Algorithm for Time Dependent Problems.

[184] P.-O. Persson and J. Peraire. "Sub-cell shock capturing for discontinuous Galerkin methods". In: *44th AIAA Aerospace Sciences Meeting and Exhibit*. Vol. 112. AIAA 2006-112. 2006. URL: http://dx.doi.org/10.2514/6.2006-112.

[185] S. B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[186] S. Premasuthan, C. Liang, and A. Jameson. "Computation of flows with shocks using spectral difference scheme with artificial viscosity". In: *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. Vol. 1449. AIAA 2010-1449. 2010.

[187] A. E. Puckett and H. J. Stewart. "The thickness of a shock wave in air". In: *Quarterly of Applied Mathematics* 7.4 (1950), pp. 457–463.

[188] J. Qiu, M. Dumbser, and C.-W. Shu. "The discontinuous Galerkin method with Lax–Wendroff type time discretizations". In: *Computer Methods in Applied Mechanics and Engineering* 194.42-44 (2005), pp. 4528–4543.

[189] J. Qiu. "A Numerical Comparison of the Lax–Wendroff Discontinuous Galerkin Method Based on Different Numerical Fluxes". In: *Journal of Scientific Computing* 30.3 (2007), pp. 345–367.

[190] J. Qiu and C.-W. Shu. "Finite difference WENO schemes with Lax-Wendroff-type time discretizations". In: *SIAM Journal on Scientific Computing* 24.6 (2003), pp. 2185–2198.

[191] A. H. Al-Rabeh. "Embedded DIRK methods for the numerical integration of stiff systems of ODEs". In: *International Journal for Computer Mathematics* 21.1 (1987), pp. 65–84.

[192] L. Rayleigh. "Aerial plane waves of finite amplitude". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 84.570 (1910), pp. 247–284. eprint: http://rspa.royalsocietypublishing.org/content/84/570/247.full.pdf.

[193] W. H. Reed and T. R. Hill. *Triangular Mesh Methods for the Neutron Transport Equations*. Tech. rep. LA-UR-73-479. Los Alamos, NM: Los Alamos Scientific Laboratory, 1973.

[194] B. Reuter, A. Rupp, V. Aizinger, and P. Knabner. "FESTUNG: A MATLAB/GNUOctave toolbox for the discontinuous Galerkin method. Part IV: Generic problem framework and model coupling interface". In: *ArXiv e-prints* (2018). arXiv: 1806.03908 [math.NA].

[195] B. Reuter, V. Aizinger, M. Wieland, F. Frank, and P. Knabner. "FESTUNG: A MATLAB/GNU Octave toolbox for the discontinuous Galerkin method, Part II: Advection operator and slope limiting". In: *Computers & Mathematics with Applications* 72.7 (2016), pp. 1896–1925.

[196] B. Reuter and F. Frank. *FESTUNG: The Finite Element Simulation Toolbox for UNstructured Grids*. 2017. URL: https://github.com/FESTUNG.

[197] S. Rhebergen and B. Cockburn. "A space-time hybridizable discontinuous Galerkin method for incompressible flows on deforming domains". In: *Journal of Computational Physics* 231.11 (2012), pp. 4185–4204.

[198] S. Rhebergen and B. Cockburn. "Space-Time Hybridizable Discontinuous Galerkin Method for the Advection–Diffusion Equation on Moving and Deforming Meshes". In: *The Courant–Friedrichs–Lewy (CFL) Condition: 80 Years After Its Discovery*. Ed. by C. A. de Moura and C. S. Kubrusly. Boston: Birkhäuser Boston, 2013, pp. 45–63.

[199] X. Roca, N. C. Nguyen, and J. Peraire. In: American Institute of Aeronautics and Astronautics, 2013. Chap. Scalable parallelization of the hybridized discontinuous Galerkin method for compressible flow.

[200] V. Rusanov. "The calculation of the interaction of non-stationary shock waves and obstacles". In: *USSR Computational Mathematics and Mathematical Physics* 1.2 (1962), pp. 304–320.

[201] Y. Saad and M. H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pp. 856–869.

[202] A. Samii, C. Michoski, and C. Dawson. "A parallel and adaptive hybridized discontinuous Galerkin method for anisotropic nonhomogeneous diffusion". In: *Computer Methods in Applied Mechanics and Engineering* 304 (2016), pp. 118–139.

[203] A. Samii, N. Panda, C. Michoski, and C. Dawson. "A Hybridized Discontinuous Galerkin Method for the Nonlinear Korteweg–de Vries Equation". In: *Journal of Scientific Computing* 68.1 (2016), pp. 191–212.

[204] J. Schöberl. *C++11 Implementation of Finite Elements in NGSolve*. ASC Report 30/2014. Institute for Analysis and Scientific Computing, Vienna University of Technology, 2014.

[205] J. Schöberl. "NETGEN an advancing front 2D/3D-mesh generator based on abstract rules". In: *Computing and Visualization in Science* 1.1 (1997), pp. 41–52.

[206] S. Schoeder, M. Kronbichler, and W. A. Wall. "Arbitrary High-Order Explicit Hybridizable Discontinuous Galerkin Methods for the Acoustic Wave Equation". In: *Journal of Scientific Computing* (2018).

[207] W. Schröder. *Fluidmechanik*. Wissenschaftsverlag Mainz in Aachen, 2007.

[208] J. Schütz, M. Woopen, and G. May. "A Combined Hybridized discontinuous Galerkin / Hybrid Mixed Method for Viscous Conservation Laws". In: *Hyperbolic Problems: Theory, Numerics, Applications*. Ed. by F. Ancona, A. Bressan, P. Marcati, and A. Marson. American Institute of Mathematical Sciences, 2012, pp. 915–922.

[209] J. Schütz. "A hybrid mixed finite element scheme for the compressible Navier-Stokes equations and adjoint-based error control for target functionals". PhD thesis. RWTH Aachen University, 2011.

[210] J. Schütz and V. Aizinger. "A hierarchical scale separation approach for the hybridized discontinuous Galerkin method". In: *Journal of Computational and Applied Mathematics* 317 (2017), pp. 500–509.

[211] J. Schütz and G. May. "A Hybrid Mixed Method for the Compressible Navier-Stokes Equations". In: *Journal of Computational Physics* 240 (2013), pp. 58–75.

[212] J. Schütz, D. C. Seal, and A. Jaust. "Implicit Multiderivative Collocation Solvers for Linear Partial Differential Equations with Discontinuous Galerkin Spatial Discretizations". In: *Journal of Scientific Computing* 73.2 (2017), pp. 1145–1163.

[213] J. Schütz, M. Woopen, and G. May. In: American Institute of Aeronautics and Astronautics, 2012. Chap. A Hybridized DG/Mixed Scheme for Nonlinear Advection-Diffusion Systems, Including the Compressible Navier-Stokes Equations.

[214] D. C. Seal, Y. Güçlü, and A. J. Christlieb. "High-Order Multiderivative Time Integrators for Hyperbolic Conservation Laws". In: *Journal of Scientific Computing* 60.1 (2014), pp. 101–140.

[215] R. Sevilla and A. Huerta. "Tutorial on Hybridizable Discontinuous Galerkin (HDG) for Second-Order Elliptic Problems". In: *Advanced Finite Element Technologies*. Ed. by J. Schröder and P. Wriggers. Cham: Springer International Publishing, 2016, pp. 105–129.

[216] S. J. Sherwin, R. M. Kirby, J. Peiró, R. L. Taylor, and O. C. Zienkiewicz. "On 2D elliptic discontinuous Galerkin methods". In: *International Journal for Numerical Methods in Engineering* 65.5 (2006), pp. 752–784.

[217] P. van Slingerland and C. Vuik. "Fast linear solver for diffusion problems with applications to pressure computation in layered domains". In: *Computational Geosciences* 18.3 (2014), pp. 343–356.

[218] G. A. Sod. "A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws". In: *Journal of Computational Physics* 27 (1978), pp. 1–31.

[219] S.-C. Soon, B. Cockburn, and H. K. Stolarski. "A hybridizable discontinuous Galerkin method for linear elasticity". In: *International Journal for Numerical Methods in Engineering* 80.8 (2009), pp. 1058–1092.

[220] A. Souläimani, N. B. Salah, and Y. Saad. "Enhanced GMRES Acceleration Techniques for some CFD Problems". In: *International Journal of Computational Fluid Dynamics* 16.1 (2002), pp. 1–20.

[221] P. R. Spalart and V. Venkatakrishnan. "On the role and challenges of CFD in the aerospace industry". In: *The Aeronautical Journal* 120.1223 (2016), pp. 209–232.

[222] M. Stanglmeier, N. C. Nguyen, J. Peraire, and B. Cockburn. "An explicit hybridizable discontinuous Galerkin method for the acoustic wave equation". In: *Computer Methods in Applied Mechanics and Engineering* 300 (2016), pp. 748–769.

[223] K. Strehmel, R. Weiner, and H. Podhaisky. *Numerik gewöhnlicher Differentialgleichungen: nichtsteife, steife und differential-algebraische Gleichungen*. Vieweg+Teubner Verlag, 2012.

[224] M. C. Sukop and D. T. Thorne. *Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers*. 1st. Springer Publishing Company, Incorporated, 2010.

[225] W. Sutherland. "LII. The Viscosity of Gases and Molecular Force". In: *Philosophical Magazine Series 5* 36.223 (1893), pp. 507–531.

[226] The Mathworks, Inc., Natick, MA, USA. *MATLAB and Statistics Toolbox Release 2014a*.

[227] C. Thiele, M. Araya-Polo, F. O. Alpak, B. Rivière, and F. Frank. "Inexact hierarchical scale separation: a two-scale approach for linear systems from discontinuous Galerkin discretizations". In: *Computers and Mathematics with Applications* (2017).

[228] C. Thiele, M. Araya-Polo, F. O. Alpak, B. Rivière, and F. Frank. "Inexact hierarchical scale separation: an efficient linear solver for discontinuous Galerkin discretizations". In: Society of Petroleum Engineers, 2017.

[229] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. 2nd ed. Springer-Verlag Berlin Heidelberg, 1999.

[230] D. J. Tritton. *Physical fluid dynamics*. Springer Netherlands, 1977.

[231] A. Tsai, R. Chan, and S. Wang. "Two-derivative Runge-Kutta methods for PDEs using a novel discretization approach". In: *Numerical Algorithms* 65 (2014), pp. 687–703.

[232] F. Vidal-Codina, N. C. Nguyen, and J. Peraire. "Computing parametrized solutions for plasmonic nanogap structures". In: *ArXiv e-prints* (2017). arXiv: `1710.06539 [physics.optics]`.

[233] P. E. Vos, C. Eskilsson, A. Bolis, S. Chun, R. M. Kirby, and S. J. Sherwin. "A generic framework for time-stepping partial differential equations (PDEs): general linear methods, object-oriented implementation and application to fluid problems". In: *International Journal of Computational Fluid Dynamics* 25.3 (2011), pp. 107–125.

[234] Z. J. Wang, K. Fidkowski, R. Abrall, F. Bassi, D. Caraeni, A. Cary, H. Deconick, R. Hartmann, K. Hillewaert, H. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. "High-Order CFD Methods: Current Status and Perspective". In: *International Journal for Numerical Methods in Fluids* 72.8 (2012), pp. 811–845.

[235] J. S. Weifeng Qiu and K. Shi. "An HDG method for linear elasticity with strong symmetric stresses". In: *Mathematics of Computation* 87.309 (2018), pp. 69–93.

[236] F. M. White. *Fluid mechanics*. McGraw-Hill Inc., 1986.

[237] D. M. Williams. "An entropy stable, hybridizable discontinuous Galerkin method for the compressible Navier-Stokes equations". In: *Mathematics of Computation* 87.309 (2017), pp. 95–121.

[238] C. H. K. Williamson. "Vortex dynamics in the cylinder wake". In: *Annual Review of Fluid Mechanics* 28 (1996), pp. 477–539.

[239] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Springer-Verlag Berlin Heidelberg, 2000.

[240] P. Woodward and P. Colella. "The numerical simulation of two-dimensional fluid flow with strong shocks". In: *Journal of Computational Physics* 54.1 (1984), pp. 115–173.

[241] M. Woopen, A. Balan, G. May, and J. Schütz. "A Comparison of Hybridized and Standard DG Methods for Target-Based *hp*-Adaptive Simulation of Compressible Flow". In: *Computers and Fluids* 98 (2014), pp. 3–16.

[242] M. Woopen, G. May, and J. Schütz. "Adjoint-Based Error Estimation and Mesh Adaptation for Hybridized discontinuous Galerkin Methods". In: *International Journal for Numerical Methods in Fluids* 76 (2014), pp. 811–834.

[243] W. Wright. "The construction of order 4 DIMSIMs for ordinary differential equations". In: *Numerical Algorithms* 26.2 (2001), pp. 123–130.

[244] W. Wright. "General linear methods with inherent Runge-Kutta stability". PhD thesis. University of Auckland, 2002.

[245] S. Wulfinghoff, H. R. Bayat, A. Alipour, and S. Reese. "A low-order locking-free hybrid discontinuous Galerkin element formulation for large deformations". In: *Computer Methods in Applied Mechanics and Engineering* 323 (2017), pp. 353–372.

[246] S. Yakovlev, D. Moxey, R. M. Kirby, and S. J. Sherwin. "To CG or to HDG: A Comparative Study in 3D". In: *Journal of Scientific Computing* 67.1 (2016), pp. 192–220.

[247] J. Yan and C.-W. Shu. "Local Discontinuous Galerkin Methods for Partial Differential Equations with Higher Order Derivatives". In: *Journal of Scientific Computing* 17.1 (2002), pp. 27–47.

[248] Y. Zeldovich and Y. Raizer. *Physics of shock waves and high-temperature hydrodynamic phenomena*. Courier Corporation, 2002.

[249] F. Zhang. *The Schur Complement and Its Applications*. Springer US, 2005.

[250] L. Zhu, T.-Z. Huang, and L. Li. "A hybrid-mesh hybridizable discontinuous Galerkin method for solving the time-harmonic Maxwell's equations". In: *Applied Mathematics Letters* 68 (2017), pp. 109–116.

[251] O. C. Zienkiewicz. "Displacement and equilibrium models in the finite element method by B. Fraeijs de Veubeke, Chapter 9, Pages 145–197 of Stress Analysis, Edited by O. C. Zienkiewicz and G. S. Holister, Published by John Wiley & Sons, 1965". In: *International Journal for Numerical Methods in Engineering* 52.3 (2001), pp. 287–342.

[252] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier Science, 2013.