On slope limiting and deep learning techniques for the numerical solution to convection-dominated convection-diffusion problems

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von
Derk Frerichs-Mihov

Berlin 2023

Erstgutachter:	Prof. Dr. Volker John (Betreuer) Freie Universität Berlin und Weierstraß-Institut für angewandte Analysis und Stochastik, Berlin
Zweitgutachter:	doc. Mgr. Petr Knobloch, Dr., DSc. Karls-Universität, Prag

Tag der Disputation:

За любовта на живота ми

Abstract

This thesis is devoted to slope limiting and machine learning techniques to approximate the solution to steady-state convection-diffusion-reaction problems. In the convection-dominated regime, the solution to those problems usually possesses layers which are small regions where the gradient is steep. It is well known from the literature that it is challenging for many classical numerical methods to approximate the solution in those regions, and often the solution is polluted by unphysical values, so-called spurious oscillations.

In the first step, the model problem is derived and investigated under which conditions a unique weak solution exists. Afterwards, symmetric, incomplete, and non-symmetric interior penalty Galerkin methods are introduced to approximate the exact solution in the pure diffusion, convection-reaction, and the complete case numerically. A-priori error estimates are provided and verified numerically.

As the first main topic, several slope-limiting techniques from the literature are presented, and various novel methods are proposed. These post-processing techniques aim to automatically detect regions where the discrete solution has unphysical values and approximate the solution locally by a lower degree polynomial. This thesis's first major contribution is that two novel methods can reduce the spurious oscillations significantly and better than the previously known methods while preserving the mass locally, as seen in two benchmark problems with two different diffusion coefficients.

The second focus is showing how to incorporate techniques from machine learning into the framework of classical finite element methods. Hence, another significant contribution of this thesis is the construction of a machine learning-based slope limiter. It is trained with data from a lower-order DG method from a particular problem and applied to a higher-order DG method for the same and a different problem. It reduces the oscillations significantly compared to the standard DG method but is slightly worse than the classical limiters.

The third main contribution is related to physics-informed neural networks (PINNs) to approximate the solution to the model problem. Various ways to incorporate the Dirichlet boundary data, several loss functionals that are novel in the context of PINNs, and variational PINNs are presented for convection-diffusion-reaction problems. They are tested and compared numerically. The novel loss functionals improve the error compared to the vanilla PINN approach. It is observed that the approximations are free of oscillations and can cope with interior layers but have problems capturing boundary layers.

Contents

List of figures

List of tables

1. Introduction

> [Science] works. Planes fly. Cars drive. Computers compute. If you base medicine on science, you cure people; if you base the design of planes on science, they fly; if you base the design of rockets on science, they reach the moon. It works.
>
> *(Prof. em. Richard Dawkins)*

A vast variety of problems in physics, chemistry, and engineering sciences are described in terms of initial-boundary value problems whose solutions describe observed phenomena. For instance, Maxwell's equations are fundamental to describe classical electromagnetism, elasticity equations model the deformation of a given object under external forces, and the Navier–Stokes equations describe the motion of viscous fluids. Some of those problems are even so important that they are part of the famous Millennium Prize Problems for which one million U.S. dollars are rewarded for the solution to one of them [Ins23b; Ins23a].

This thesis focuses on another fundamental type of initial-boundary value problem: so-called *convection-diffusion-reaction* problems. To illustrate these problems, imagine a river flowing from one point to another. Unfortunately, a ship was damaged in the middle of the river, so it sank and now constantly leaks a chemical it carried. The chemical flows into the water, but how does it spread in the river? What is the concentration at a given point? The solution to convection-diffusion-reaction problems exactly gives answers to these questions.

Simply speaking, they describe how a scalar quantity inside a flowing medium is distributed in a given region. The quantity of interest can be, for instance, a species concentration like a chemical, a pollutant or biomass, or energy in the form of heat [JKN18; Eva10, p. 313]. In other words, both mass and heat transfer can be described. In light of the above-mentioned example of a ship from which a chemical flows into a river, the chemical concentration might be the modeled quantity. After the chemical enters the river, it does not stay at its initial position but is distributed in the water based on certain physical processes. As the name suggests, three are incorporated into convection-diffusion-reaction problems. First of all, the chemical is transported downstream by the flow of the water. This effect is called *convection* [RST08, p. 1]. Nevertheless, even if there was no river flow, it would spread in the water due to what is called *diffusion*. Whenever there are concentration differences of a quantity, it moves to regions with lower concentrations until the differences are leveled out [SJ02, section 1.2]. Lastly, this chemical may react with other particles and form other compounds. At the same time, it is also possible that other reactions may form

(a) Reaction of chemical in solvent.

(b) Mixing milk and coffee.

(c) Air pollution.

Figure 1.1.: Example processes that might be described by convection-diffusion-reaction problems. In (a), the reaction of a chemical in a moving solvent is illustrated. Mixing milk and coffee is shown in (b), where possibly no reaction occurs. In (c), air pollution that stems from industry is depicted. From left to right, the images are taken from [Pica; Picb; Picc].

the chemical again. Both effects affect the concentration of the quantity of interest, which is reflected by the so-called *reaction* term.

The previously mentioned ship example is a simple one. However, these problems are also widely applied in practice: Among others, they are used to model atmospheric pollution in the form of, e.g., denser-than-air gases [Lan78; Erm92] or radioactive tracers [Lee+16; Lee+18], and parts of marine ecosystems like the distribution of phytoplankton biomass [FN15, section 5.3]. While in the context of financial mathematics, they are called the Black-Scholes model, which models the price of an option and is rewarded with the Nobel Prize in Economics [SJ15], they are referred to as drift-diffusion models that describe the distribution of charge carrier densities when simulating semiconductor devices [Pip17, section 50.3.1]. Moreover, they are applied to model chemical tubular reactors [Alh07], groundwater pollution [Ata18, section 3.5], and the vertical ocean heat balance in ocean models [Hub+15; Hoc+21]. In particular, the last example indicates the importance of these problems since ocean heat uptake has been responsible for more than 90% of global earth warming since the 1970s [Fox+21; Lev+12]. Figure 1.1 illustrates possible further examples that might be modeled with convection-diffusion-reaction problems.

Most of the examples mentioned above model the time evolution of a particular quantity. However, from a mathematical point of view, the solutions' most important characteristic features can already be observed in the so-called *stationary state*. In this state, the concentration no longer changes with time; in some sense, the system has reached an equilibrium. Furthermore, the stationary problems are often easier to analyze than the time-dependent problem and often serve as a first step towards analyzing the time-dependent case. As a consequence, this work focuses on stationary convection-diffusion-reaction problems.

Mathematically speaking, stationary convection-diffusion-reaction equations can be

described as follows: Find a smooth enough unknown mass or energy concentration u such that

$$-\varepsilon\Delta u + \boldsymbol{b} \cdot \nabla u + cu = f$$

holds in a domain of interest Ω, where $\varepsilon > 0$ is the strength of the diffusion, $\boldsymbol{b}$ models the convection field, and c and f describe the reaction term and sources or sinks, respectively.

In practical applications, the convection is often orders of magnitude stronger than the diffusion [JKN18; Mor19, p. 1]. In this so-called *convection-dominated* regime, convection-diffusion-reaction problems can be seen as *singularly perturbed* problems [RST08, p. 2]. The solution to such problems typically possesses so-called *layers* [RST08, p. 2; Mor19, p. 1], which are parts of the solution where it rapidly changes. How fast the solution changes, i.e., the width of these layers, depends on the strength of the diffusion ε. Depending on whether they are exponential or characteristical layers, the width is of order $\mathcal{O}(\varepsilon)$ or $\mathcal{O}(\sqrt{\varepsilon})$, respectively, if the equations are scaled such that the convection field $\boldsymbol{b}$ is of order $\mathcal{O}(1)$ in a suitable norm [JKN18].

Finding the exact solution u to these problems is usually possible only in simple toy problems, but it is unlikely to happen in practice. Therefore, approximations to the solution are constructed based on numerical methods. Popular classical examples of such methods are so-called *finite difference methods*, *finite element methods*, and *finite volume methods*, but there also exist modern approaches like *virtual element methods* and *hybrid high-order methods*; see, e.g., [Bra07, sections I.3, II.4; RST08, sections I.2.1, I.2.2; EGH00, section 3.9; Bei+13; Bei+16; DT18; DEL16].

In most of those methods, the solution is approximated by a linear combination of a finite number of ansatz functions. They are defined on a given *mesh*, which is a decomposition of the domain Ω into small subregions. It can be proven for many problems that the discrete solution of the methods mentioned above converges towards the exact solution as the mesh becomes finer. However, the approximation quality might be low if too few ansatz functions are used. Unfortunately, using an arbitrary amount of cells is impossible in practice because more ansatz functions lead to higher computational costs, energy consumption, and computing time. Therefore, methods that produce acceptable quality solutions on computationally feasible meshes are sought.

1.1. Motivation

In the case of convection-dominated convection-diffusion-reaction problems, it is challenging for many widespread methods to compute an accurate approximation in the region of layers. Finite difference and many classical finite element methods, such as the popular streamline upwind Petrov–Galerkin (SUPG) or the discontinuous Galerkin method produce unphysical results; see, e.g., [JK07; Aug+11; JKN18]. In

regions of layers, the solution is polluted by over- and undershoots, so-called *spurious oscillations*, which result in, e.g., negative concentrations or higher amounts of energy than what is physically allowed. The reason is that the width of the layers is much smaller than the mesh width, which is why the layers cannot be adequately resolved [JKN18].

On the other hand, there also exist discrete methods that are provably free of spurious oscillations, e.g., the exponentially fitted Voronoi box finite volume method [Aug+11] and algebraic flux-corrected (AFC) methods [BJK16; Bar+18a]. However, the finite volume method tends to smear out the layers considerably, and only second-order convergence in the discrete L^2-norm has been observed [Aug+11; FLL11]. On the contrary, AFC methods are known to produce non-smeared layers without oscillations, but they require solving a system of non-linear equations and might be, therefore, computationally expensive [BJK16; Bar+18a]. Furthermore, there are, as of June 2023, only AFC methods with lowest-order continuous finite elements that have been applied successfully, which limits the order of convergence of such methods [BJK16; Jha20, p. 44]. In general, all available and reasonably accurate finite element methods that satisfy a global maximum principle are based on linear finite elements and optimal convergence rates which can be observed in some situations could not be proved so far, cf. [BJK23]. To summarize, there does not exist yet *the* optimal numerical method for convection-dominated convection-diffusion-reaction problems.

Nevertheless, what are the requirements of such a discrete method? In a nutshell, the optimal numerical method should

1. converge fast towards the exact solution as the mesh becomes finer,
2. be flexible with respect to the used mesh,
3. produce solutions with sharp layers, and
4. compute physically reasonable solutions free of spurious oscillations.

The first requirement is vital since a low-order approximation might be admissible in the presence of layers, but in smooth regions, it should approximate the solution as accurately as possible on the computationally feasible meshes. Therefore, the method should be of higher order.

Regarding the second requirement, polygonal meshes, compared to classical simplicial, quadrilateral, and hexahedral meshes allow for a more straightforward decomposition of the domain, have efficient refinement and coarsening strategies, and are more robust with respect to distortion and torsion of the mesh [Dro+21; BLV17]. Therefore, a method that can be used with more general or even without meshes might be advantageous in practice. Note that the first two requirements are vital not only for convection-diffusion-reaction problems but are desirable in general for numerical methods.

The third and the last property are related to the quality of the approximation of the solution to convection-dominated convection-diffusion-reaction problems. A solution with spurious oscillations or with smeared layers might not correspond to reality, but since these methods are used in critical practical applications, they should be able to compute physically consistent and reasonably accurate solutions.

Particular candidates for such a method are so-called *discontinuous Galerkin* (DG) methods, which are considered in the present work. Even though they were already invented in 1973 by Reed and Hill in [RH73], they gained much attention in the last decades; see, e.g., the monographs [CKS00; Kan07; Riv08; DE12; DF15]. Introduced for the neutron transport equation, they have been successfully applied to a huge variety of problems, e.g., boundary-layer equations for incompressible steady fluid flows, compressible Navier–Stokes equations [DE12, p. VI], linear elasticity, porous media flow [Riv08, part III], magneto-hydrodynamics and Hamilton–Jacobi equations [Coc03]. For a historical overview and more applications, the reader may be referred to [CKS00, part I; CS01; DE12, pp. V–VI].

In contrast to continuous finite element methods, the basis functions used in DG methods are, as the name suggests, discontinuous over element boundaries. This makes them, in particular, appealing for problems with discontinuous solutions, such as transport problems, but continuity can also be enforced by, e.g., interior penalty methods if needed. They can be seen as a generalization of finite volume methods to arbitrary higher polynomial degrees [CKS00, p. 4]. Therefore, in light of the above-mentioned requirements, DG methods can be constructed to converge with an arbitrarily high order of convergence. Of course, this happens in practice only if the exact solution is smooth enough.

Furthermore, they can be used on general polyhedral meshes and, in particular, also allow hanging nodes [DE12, section 1.4.1; CKS00, p. 5]. Moreover, they allow refinements of both the mesh and the polynomial degree, i.e., DG methods are particularly suited for *hp*-adaptivity [HSS02; Ant+16; CKS00, p. 5]. For convection-dominated convection-diffusion-reaction problems, they are also known to produce sharp layers [Aug+11]. However, at the same time, they are unfortunately also known to produce a significant amount of spurious oscillations [Aug+11]. To summarize, DG methods are good candidates for approximating the solution to convection-dominated convection-diffusion-reaction problems, but their spurious oscillations have to be reduced to be more useful in practice.

Adjusting the widely known symmetric, non-symmetric, and incomplete interior penalty Galerkin methods to produce significantly fewer spurious oscillations is one of the significant contributions of this work. To this end, so-called *slope limiters* are used to automatically detect regions where the numerical solution is unphysical, followed by a local solution adjustment in the corresponding cells. The advantages of these limiters are that they are easy to implement, they are computationally cheap, and in contrast to just clipping the extrema, they preserve the mass locally. Several slope limiters from the literature, generalizations, and novel approaches are presented and

extensively tested on two commonly used benchmark problems. As seen in this thesis, the best of these methods can reduce the oscillations significantly, and in many test cases, they can reduce the oscillations as much as possible while preserving the mass. However, none of them can remove the unphysical values completely.

Another major contribution of this work is to combine techniques from the field of deep learning with the ideas of the aforementioned slope limiters for convection-dominated convection-diffusion-reaction problems. Deep learning techniques are a particular method from the broader field of machine learning, whose algorithms are implemented in a way that they can "learn" rules from a given data set by minimizing a given cost functional, even if these rules are never explicitly implemented [GBC16, pp. 1–2]. The first ideas in the direction of what is nowadays called deep learning emerged in the second half of the last century. However, it took several years and rediscoveries to understand its full potential. For a historical overview, the readers may be referred to [Sch15; WR17]. In particular, in the last two decades, these methods have made a breakthrough, and nowadays, many applications cannot be imagined without them; see, e.g., [GBC16; Sar21; HH19] for examples and state-of-the-art techniques.

Roughly speaking, due to the famous *universal approximation theorem* with deep learning methods, it is possible to approximate a vast variety of mappings, e.g., the translation of sentences from one language into another, the mapping from an image to what is depicted on it, and also many mathematical functions. A particular type of deep learning architecture are the so-called *neural networks*. Based on the approximation property, the second main contribution of this thesis is the construction of a neural network-based slope limiter that, as its classical counterparts, predicts where spurious oscillations occur and corrects the solution locally.

Moreover, as a consequence of the universal approximation theorem, it is also possible to use neural networks to approximate the unknown solution to initial-boundary value problems (IBVPs) directly, even if only the problems are given. These so-called *physics-informed neural networks* (PINNs) can therefore be seen as another numerical method and, in this sense, as an alternative to classical ones. However, in contrast to classical methods, PINNs are, by construction, also able to easily cope with inverse problems and incorporate already observed data of the solution [Kar+21]. They were already introduced in 1994 in [DP94], but they had their breakthrough only after their recent rediscovery in a series of papers of the group by Karniadakis beginning with [RPK19]; see also [Cai+21; Cuo+22; Kar+21] for an overview about PINNs.

In a nutshell, PINNs are neural networks that are trained to represent the solution to given IBVPs by approximately satisfying the governing equations and their boundary conditions [Kar+21]. In their standard formulation, they are mesh-free and hence, in light of the above-mentioned requirements, are highly flexible with respect to the geometry of the problem. Theoretically, PINNs can approximate the unknown solution to an IBVP with arbitrary precision, but the accuracy depends on stochastic processes.

Therefore, it is usually proven that PINNs can approximate the solution but not how long it takes to reach a given precision. Since very few publications investigate PINNs for convection-dominated convection-diffusion-reaction problems, the third major contribution of this thesis is to develop, compare, and evaluate the quality of the solution of several known and novel variants of PINNs for these problems.

1.2. Outline

The structure of this thesis is as follows:

Chapter 2 is devoted to the primary type of boundary value problem tackled in this work, i.e., to convection-diffusion-reaction problems. In section 2.1, after the problems are derived, it is investigated under which conditions a unique weak solution exists. Section 2.2 then focuses on discontinuous Galerkin methods for such problems. The widely used symmetric, non-symmetric, and incomplete penalty Galerkin methods are introduced for the pure diffusion case, followed by an upwind DG discretization of convection-reaction problems. Afterwards, these methods are combined for the complete convection-diffusion-reaction problem. These methods are not only derived, but proofs are also given with respect to the convergence rates of these problems. Last but not least, this chapter closes with a two- and three-dimensional numerical example to verify the implementation and a summary; see sections 2.3 and 2.4.

In chapter 3, slope limiters for DG methods are treated. It is heavily inspired by and mainly based on the results already published in [FJ21; FJ22]. Several slope-limiting techniques from the literature, as well as generalized and novel versions, are presented in sections 3.1 to 3.3, and it is shown that they all preserve the mass locally. Moreover, in section 3.4, numerical studies with these slope limiters for two benchmark examples are presented. To this end, two measures are defined that rate the quality of the limited solution with respect to spurious oscillations, and two different diffusion coefficients and two types of meshes are used. The chapter ends with a summary in section 3.5.

Chapter 4 serves as the bridge between slope limiters and deep learning techniques. Its results are mainly also available as a preprint; see [FHJ22]. It starts in section 4.1 with an introduction to multilayer perceptron models, a particular type of neural network. Here the basic structure of multilayer perceptrons is shown, followed by an explanation of how these networks are trained. Afterwards, with section 4.2, follows a section about how data is generated with which the neural networks can be trained. It is explained how a vast amount of data can be created that needs to be reduced and divided into subsets. This is followed by section 4.3, in which the first numerical experiments are conducted. After specifying the setup of the networks and the measures to rank the performance, the networks are trained on the data set. The best networks are then applied to the DG solution to two benchmark problems in section 4.4. Finally, a summary of this chapter is given in section 4.5.

In chapter 5 PINNs for convection-diffusion-reaction problems are investigated. To begin with, in section 5.1, the basic ideas and the classical loss functional of PINNs are derived. Furthermore, a simple numerical example is shown to verify the implementation. It is followed by section 5.2, where several modifications of classical PINNs from the literature and new ideas are presented. These methods are then tested in section 5.3 for another two benchmark problems. Afterwards, the results are summarized in section 5.4.

The thesis finally closes in chapter 6 with a summary of what is treated in this work and gives ideas about possible paths for future research.

Throughout this thesis, standard notations are used. This includes the usual function spaces $L^2(\Omega)$, $W^{k,p}(\Omega)$, $H^k(\Omega)$, etc., and their respective norms; cf. [BS02]. A norm of a space X is denoted by $\|\cdot\|_X$, a seminorm by $|\cdot|_X$, the inner product in $L^2(\Omega)$ by $(\cdot,\cdot)$, and for a vector $\boldsymbol{x} \in \mathbb{R}^d$ its Euclidean norm is denoted by $|\boldsymbol{x}|$.

2. Convection-diffusion-reaction problems

This chapter mainly serves as an introduction to convection-diffusion-reaction problems as described in the previous chapter, and discontinuous Galerkin methods for these problems. To this end, in section 2.1, the problems are derived from scratch, and the existence of a unique solution to these problems is discussed. Section 2.2 follows with a presentation of discontinuous Galerkin methods for these problems. The commonly used symmetric, incomplete and non-symmetric penalty Galerkin methods are derived, and the error between the exact and the discrete solution is estimated in various norms. The findings are mainly based on the monographs [Riv08; DE12; Kan07; DF15]. This chapter proceeds in section 2.3 with numerical experiments to verify the implementation and ends with a summary in section 2.4.

2.1. Model problem

First, in this section, the statement of convection-diffusion-reaction problems is derived. Second, it is investigated under which conditions a unique solution to these problems exists.

2.1.1. Derivation of convection-diffusion-reaction problems

The presentation in this section is essentially based on [SJ02, section 2.1] and [Jha20, section 2.1].

The starting point for deriving the convection-diffusion-reaction problems is the following question: How does a quantity of interest spread in a flowing medium in a certain region of interest? To this end, let the region be a bounded domain $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, $\Gamma := \partial\Omega$ its boundary, and assume that $u : \Omega \times (0, T] \to \mathbb{R}$ is the concentration of the quantity of interest that shall be observed in the time interval

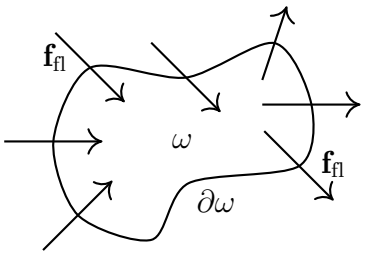


Figure 2.1.: Arbitrary control volume ω and flux $\mathbf{f}_{\mathrm{fl}}$ across its boundary $\partial\omega$.

$(0, T]$, $0 < T \in \mathbb{R}$. The evolution in time of the quantity is based on the physical observation that, in any fixed sub-domain $\omega \subset \Omega$, the rate of change of the quantity equals the flux of the quantity across the boundary $\partial\omega$ plus the total amount that is produced or destroyed in ω; see also figure 2.1 for an illustration. Translating this into mathematics leads to

$$\frac{\partial}{\partial t} \int_\omega u \, \mathrm{d}\boldsymbol{x} = - \int_{\partial\omega} \mathbf{f}_{\mathrm{fl}}(u) \cdot \boldsymbol{n} \, \mathrm{d}s + \int_\omega S(u) \, \mathrm{d}\boldsymbol{x},$$

where $\mathbf{f}_{\mathrm{fl}} : \overline{\Omega} \times (0, T] \to \mathbb{R}^d$ is the flux function depending on u, $S : \Omega \times (0, T]$ denotes sources or sinks which might also depend on u, $\boldsymbol{n}$ is the outer unit normal vector to the respective boundary, $\boldsymbol{x} \in \Omega$ denotes the space variable, $t \in (0, T]$ the time variable and s a $(d-1)$-dimensional space variable. For the sake of brevity, the dependence on space and time is not explicitly mentioned. Assuming the functions and the boundary to be smooth enough, Gauss' divergence theorem A.2 and a change of differentiation with respect to time and integration with respect to space lead to

$$\int_\omega \frac{\partial u}{\partial t} \, \mathrm{d}\boldsymbol{x} + \int_\omega \operatorname{div}\left(\mathbf{f}_{\mathrm{fl}}(u)\right) \, \mathrm{d}\boldsymbol{x} = \int_\omega S(u) \, \mathrm{d}\boldsymbol{x}.$$

Since ω was chosen arbitrarily, for any $(\boldsymbol{x}, t) \in (\Omega \times \mathbb{R}_+)$, it has to hold that

$$\frac{\partial u}{\partial t} + \operatorname{div}\left(\mathbf{f}_{\mathrm{fl}}(u)\right) = S(u). \tag{2.1}$$

The statement of this equation is basically what was already mentioned above: The rate of change of u is balanced by the flux and the source terms.

The next step is to specify the flux term. It can be assumed that diffusion and convection are linearly independent [SJ02, p. 24]. This makes sense since the diffusion models a movement due to concentration differences which is independent of the movement due to the underlying fluid. Therefore, these two movements can simply be added, which leads to

$$\mathbf{f}_{\mathrm{fl}}(u) = \mathbf{f}_{\mathrm{fl};\varepsilon}(u) + \mathbf{f}_{\mathrm{fl};\boldsymbol{b}}(u),$$

where $\mathbf{f}_{\mathrm{fl};\varepsilon}$ denotes the flux due to the diffusion and $\mathbf{f}_{\mathrm{fl};\boldsymbol{b}}$ the flux due to the convection. Modeling the diffusion by Fick's law[1] and the convection by the convective transport[2] leads to

$$\begin{aligned} \mathbf{f}_{\mathrm{fl};\varepsilon}(u) &\coloneqq -\varepsilon \nabla u, \\ \mathbf{f}_{\mathrm{fl};\boldsymbol{b}}(u) &\coloneqq \boldsymbol{b} u, \end{aligned} \tag{2.2}$$

[1] See, e.g., [SJ02, equation (1.15)].
[2] See, e.g., [SJ02, equation (2.3)].

where $0 \leq \varepsilon \in \mathbb{R}$ for the sake of simplicity is assumed to be a constant diffusion coefficient and $\boldsymbol{b} : \Omega \times (0, T] \to \mathbb{R}^d$ models the convection field of the fluid. Substituting equation (2.2) into equation (2.1) leads to

$$\frac{\partial u}{\partial t} - \varepsilon \Delta u + \boldsymbol{b} \cdot \nabla u + \operatorname{div}(\boldsymbol{b})\, u = S(u). \tag{2.3}$$

The last process to be incorporated is chemical reactions, i.e., production or depletion of the quantity of interest. To be precise, it is already incorporated, as it can be seen as a part of the source term. This is why the source term can be split into $S(u) = f - R(u)$, where $R : \Omega \times (0, T] \to \mathbb{R}$ models the reaction process and $f : \Omega \times (0, T] \to \mathbb{R}$ the remaining sources or sinks. For the sake of simplicity, the reaction term is set to

$$R(u) \coloneqq \widehat{c} u,$$

where $\widehat{c} : \Omega \times (0, T] \to \mathbb{R}$ is the reaction function. Plugging this into equation (2.3) and defining $c \coloneqq \operatorname{div}(\boldsymbol{b}) + \widehat{c}$ finally leads to

$$\frac{\partial u}{\partial t} - \varepsilon \Delta u + \boldsymbol{b} \cdot \nabla u + c u = f,$$

which is also referred to as *evolutionary convection-diffusion-reaction equation*.

This work focuses on the steady-state convection-diffusion-reaction problems, i.e., the case where $\partial u / \partial t = 0$. Equipping the steady-state convection-diffusion-reaction equations with appropriate boundary conditions leads to the following model problem tackled in this work.

Problem 2.1 (Strong form of the convection-diffusion-reaction problem)**.** Find a sufficiently smooth solution $u : \overline{\Omega} \to \mathbb{R}$ such that

$$-\varepsilon \Delta u + \boldsymbol{b} \cdot \nabla u + c u = f \qquad \text{in } \Omega, \tag{2.4a}$$

$$u = g_{\mathrm{D}} \qquad \text{along } \Gamma_{\mathrm{D}}, \tag{2.4b}$$

$$\varepsilon \nabla u \cdot \boldsymbol{n} = g_{\mathrm{N}} \qquad \text{along } \Gamma_{\mathrm{N}}, \tag{2.4c}$$

where $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, is a bounded domain with a polyhedral Lipschitz boundary $\Gamma \coloneqq \partial\Omega$, $\Gamma = \overline{\Gamma_{\mathrm{D}}} \cup \overline{\Gamma_{\mathrm{N}}}$ and $\Gamma_{\mathrm{D}} \cap \Gamma_{\mathrm{N}} = \emptyset$, and $\boldsymbol{n}$ is the outward unit normal vector to Γ. The parameter $\varepsilon \in \mathbb{R}$, $\varepsilon > 0$, is the constant diffusion coefficient, $\boldsymbol{b} \in [\mathrm{Lip}(\Omega)]^d$ models the convection field that is assumed to be Lipschitz continuous in each component, $c \in L^\infty(\Omega)$ is the reaction function, and $f \in L^2(\Omega)$ describes sources and sinks. While the boundary conditions along the Dirichlet boundary Γ_{D} are given by $g_{\mathrm{D}} \in H^{1/2}(\Gamma_{\mathrm{D}})$, the Neumann boundary conditions $g_{\mathrm{N}} \in H^{-1/2}(\Gamma_{\mathrm{N}})$ prescribe the derivative of the solution in the normal direction along the Neumann boundary Γ_{N}. In accordance with physics, Dirichlet boundary conditions have to be dictated at the inflow boundary, i.e., $\Gamma_- \coloneqq \{\, \boldsymbol{x} \in \Gamma \;:\; \boldsymbol{b}(\boldsymbol{x}) \cdot \boldsymbol{n} < 0 \,\} \subset \overline{\Gamma_{\mathrm{D}}}$. □

Multiplying equation (2.4a) with an appropriate test function, integration by parts, and using the Neumann boundary condition (2.4c) leads to the variational formulation for the convection-diffusion-reaction problem.

Problem 2.2 (Variational form of the convection-diffusion-reaction problem)**.** Under the assumptions of problem 2.1, find $u \in H^1_{\mathrm{D},g_\mathrm{D}}(\Omega)$ such that

$$(\varepsilon \nabla u, \nabla v) + (\boldsymbol{b} \cdot \nabla u + cu, v) = (f, v) + (g_\mathrm{N}, v)_{\Gamma_\mathrm{N}}, \qquad \text{for any } v \in H^1_{\mathrm{D},0}, \tag{2.5}$$

where $(\cdot, \cdot)$ is the standard L^2-scalar product in Ω, $(v, w)_{\Gamma_\mathrm{N}} \coloneqq \int_{\Gamma_\mathrm{N}} vw \,\mathrm{d}s$, and

$$H^1_{\mathrm{D},g}(\Omega) \coloneqq \left\{ v \in H^1(\Omega) \;:\; v|_{\Gamma_\mathrm{D}} = g \text{ in the sense of traces} \right\}.$$

□

A solution to equation (2.5) is called *weak solution* to problem 2.1; see, e.g., [Eva10, section 6.1.2] for an introduction to the notion of weak solutions.

After defining the problem, it needs to be investigated under which conditions a unique weak solution exists. This is the purpose of the following section.

2.1.2. Uniqueness of the solution

To comply with the notation of the Lax–Milgram lemma A.5 that is used below, the notation

$$\begin{aligned} (V, \|\cdot\|_V) &\coloneqq \left(H^1_{\mathrm{D},0}(\Omega), \|\cdot\|_{H^1}\right), \\ a(v, w) &\coloneqq (\varepsilon \nabla v, \nabla w) + (\boldsymbol{b} \cdot \nabla v + cv, w) && (v, w \in H^1(\Omega)), \\ F(v) &\coloneqq (f, v) + (g_\mathrm{N}, v)_{\Gamma_\mathrm{N}} && (v \in H^1(\Omega)), \end{aligned}$$

is introduced.

The following theorem states conditions under which a unique weak solution to the convection-diffusion-reaction problem 2.1 exists.

Theorem 2.3. *If the assumptions on the domain and the data of problem 2.1 and*

$$c - \frac{1}{2} \nabla \cdot \boldsymbol{b} \geq 0, \qquad |\Gamma_\mathrm{D}| > 0$$

hold, where $|\Gamma_\mathrm{D}|$ is the $(d-1)$-dimensional measure of Γ_D, then there exists a unique $u \in H^1_{\mathrm{D},g_\mathrm{D}}(\Omega)$ such that equation (2.5) holds for any $v \in H^1_{\mathrm{D},0}(\Omega)$. In other words, there exists a unique weak solution to the convection-diffusion-reaction problem 2.1.

Proof. Due to the assumptions on the boundary, there exists a linear and continuous extension operator $E_{\Gamma_\mathrm{D}} : H^{1/2}(\Gamma_\mathrm{D}) \to H^1(\Omega)$ [Wil19, theorem 4.2.4]. Therefore, let

$e_{g_\mathrm{D}} \coloneqq E_{\Gamma_\mathrm{D}}(g_\mathrm{D})$. Since E_{Γ_D} is continuous, it is also bounded, and it follows that $\|e_{g_\mathrm{D}}\|_V \leq C_{E_{\Gamma_\mathrm{D}}} \|g_\mathrm{D}\|_{H^{1/2}(\Gamma_\mathrm{D})}$ with some real number $0 < C_{E_{\Gamma_\mathrm{D}}} < \infty$.

Using the Lax–Milgram lemma A.5, it can be proven that there exists a unique $\widetilde{u} \in V$ such that

$$a(\widetilde{u}, v) = F(v) - a(e_{g_\mathrm{D}}, v), \qquad \text{for any } v \in V, \tag{2.6}$$

which is done in the following.

It is well known that $(V, \|\cdot\|_V)$ is a Hilbert space. Since integrals are linear operators, $a(\cdot,\cdot)$ is a bilinear mapping, and the right-hand side $F(\cdot) - a(e_{g_\mathrm{D}}, \cdot)$ is a linear functional on V.

Moreover, it holds $\|\boldsymbol{b}\|_{[L^\infty(\Omega)]^d} < \infty$, since $\boldsymbol{b} \in [\mathrm{Lip}(\Omega)]^d$ [BS02, p. 30]. Due to this, the other regularity assumptions on the data, Hölder's inequality A.3 and the definition of $\|\cdot\|_V$ it holds, for any $v, w \in V$,

$$\begin{aligned} |a(v,w)| &\leq \varepsilon \|\nabla v\|_{L^2(\Omega)} \|\nabla w\|_{L^2(\Omega)} + \|\boldsymbol{b}\|_{[L^\infty(\Omega)]^d} \|\nabla v\|_{L^2(\Omega)} \|w\|_{L^2(\Omega)} \\ &\quad + \|c\|_{L^\infty(\Omega)} \|v\|_{L^2(\Omega)} \|w\|_{L^2(\Omega)} \\ &\leq \alpha \|v\|_V \|w\|_V, \end{aligned}$$

where $\alpha \coloneqq \big(\varepsilon + \|\boldsymbol{b}\|_{[L^\infty(\Omega)]^d} + \|c\|_{L^\infty(\Omega)}\big)$, which is why the bilinear form is continuous.

Due to integration by parts A.1 it holds, for any $v \in V$,

$$\int_\Omega \boldsymbol{b} \cdot \nabla v \, v \,\mathrm{d}\boldsymbol{x} = -\frac{1}{2} \int_\Omega \mathrm{div}\,(\boldsymbol{b})\, v^2 \,\mathrm{d}\boldsymbol{x} + \frac{1}{2} \int_{\partial\Omega} \boldsymbol{b} \cdot \boldsymbol{n}\, v^2 \,\mathrm{d}\boldsymbol{x} \geq -\frac{1}{2} \int_\Omega \mathrm{div}\,(\boldsymbol{b})\, v^2 \,\mathrm{d}\boldsymbol{x},$$

where in the last step it is used that $v = 0$ along Γ_D and, since $\Gamma_- \subset \overline{\Gamma_\mathrm{D}}$, it is $\boldsymbol{b} \cdot \boldsymbol{n} \geq 0$ along Γ_N. From this estimate and the assumptions it follows

$$\int_\Omega (\boldsymbol{b} \cdot \nabla v + cv)\, v \,\mathrm{d}\boldsymbol{x} \geq \int_\Omega \left(c - \frac{1}{2} \mathrm{div}\,(\boldsymbol{b}) \right) v^2 \,\mathrm{d}\boldsymbol{x} \geq 0.$$

Together with Friedrichs' inequality A.4 it can be concluded

$$a(v,v) \geq \varepsilon \|\nabla v\|^2_{L^2(\Omega)} \geq \beta \|v\|^2_V,$$

where $\beta \coloneqq \varepsilon / (1 + C^2_{\mathrm{Fr}})$, which shows that $a(\cdot,\cdot)$ is V-elliptic.

Last but not least, due to Cauchy's and Hölder's inequality A.3, the boundedness of $a(\cdot,\cdot)$, the boundedness of the extension operator, the continuity of the trace operator, and the regularity of the data it holds, for any $v \in V$,

$$|F(v) - a(e_{g_\mathrm{D}}, v)| \leq |(f, v)| + |(g_\mathrm{N}, v)_{\Gamma_\mathrm{N}}| + |a(e_{g_\mathrm{D}}, v)| \leq C \|v\|_V$$

with a real number $C \coloneqq \Big(\|f\|_{L^2(\Omega)} + C_{\mathrm{tr}_{\Gamma_\mathrm{N}}} \|g_\mathrm{N}\|_{H^{-1/2}(\Gamma_\mathrm{N})} + \alpha C_{E_{\Gamma_\mathrm{D}}} \|g_\mathrm{D}\|_{H^{1/2}(\Gamma_\mathrm{D})} \Big) < \infty$, where $C_{\mathrm{tr}_{\Gamma_\mathrm{N}}}$ is the continuity constant of the trace operator onto Γ_N. This means the right-hand side of equation (2.6) is bounded.

Hence, the assumptions of the Lax–Milgram lemma A.5 are satisfied, and it follows that the problem with a homogeneous Dirichlet boundary condition (2.6) admits a unique solution.

To conclude the proof, $u \coloneqq \widetilde{u} + e_{g_\mathrm{D}}$ is defined. By construction, it holds that $u \in H^1_{\mathrm{D},g}(\Omega)$ and that u satisfies equation (2.5) for any $v \in V$. □

Unfortunately, it is usually very difficult or even impossible to compute the weak solution in practice. Therefore, discrete algorithms are designed to compute an approximation. The construction of such an algorithm and the quality of the resulting approximation is addressed in the following sections.

2.2. Discontinuous Galerkin methods for convection-diffusion-reaction problems

Several discretization techniques exist for approximating the solution to convection-diffusion-reaction problems, e.g., finite difference, finite volume, and finite element methods. For an overview of these methods in the context of these problems, see the monograph by Roos, Stynes, and Tobiska [RST08, parts I–III]. Besides the popular classical continuous finite element methods (CG) and their stabilized variants, discontinuous Galerkin (DG) methods drew attention in the last decades, see, e.g., the monographs [CKS00; Kan07; Riv08; DE12; DF15] and the references therein, even though the original technique was introduced by Reed and Hill [RH73] in 1973. For the sake of brevity, the reader may be referred to [CKS00, part I] and [DE12, preface] for an overview of the history and evolution of DG methods, not necessarily but also in the context of convection-diffusion-reaction problems.

In contrast to classical methods of a Lagrangian type that yield a continuous approximation, the fundamental idea of DG methods is, as the name suggests, to use discontinuous approximations. In the context of convection-diffusion-reaction problems, the DG methods are, therefore, non-conforming methods. Their idea and definition offer some advantages but, as so often, also some drawbacks.

On the one hand, DG methods facilitate so-called *hp-refinements*, i.e., local refinement of the mesh or the polynomial degree, which is more complicated for CG methods. On the other hand, in DG methods, many degrees of freedom couple leading to a comparatively dense stiffness matrix, and hence efficient solvers become an issue. Moreover, in DG methods, it is unnecessary to use a conforming grid closure step when performing adaptive refinements as long as the number of hanging nodes is bounded. This makes mesh refinements easier and saves computational resources, which can be helpful, especially in three dimensions.

These advantages come at the price of a more involved implementation and the challenge of choosing suitable parameters. The main reason for that is that DG methods for convection-diffusion-reaction problems try to approximate often a continuous

solution with piecewise continuous functions, and hence care has to be taken at points of discontinuities.

The first step when describing discrete methods, whether CG or DG, is to decompose the domain Ω into a finite number of subsets. Therefore, assumptions concerning the so-called *mesh* are made in what follows.

Mesh notation and assumptions

Let $\mathcal{T}$ be a decomposition of $\overline{\Omega}$ into simplices or quadrilaterals, resp. hexahedrons, such that $\overline{\Omega} = \bigcup_{K\in\mathcal{T}} K$. The decomposition is called *mesh* or *triangulation* and shall be admissible in the usual sense of Ciarlet; see [Cia02, pp. 38, 51]. This means that

- each $K \in \mathcal{T}$ is closed and its interior $\overset{\circ}{K}$ is non-empty,
- for each pair $K_1, K_2 \in \mathcal{T}$ it holds $\overset{\circ}{K}_1 \cap \overset{\circ}{K}_2 = \emptyset$ and
- any $(d-1)$-dimensional facet of any $K \in \mathcal{T}$ is either a subset of the boundary Γ or a facet of another mesh cell.

The measure of a mesh cell $K \in \mathcal{T}$ is denoted by $|K|$, its diameter by h_K, and the radius of the largest ball contained in K by ϱ_K. That being said, $\mathcal{T}_h$ denotes a triangulation with a given mesh size $h \coloneqq \max_{K\in\mathcal{T}} h_K$.

Families of decompositions $\{\,\mathcal{T}_h\,\}_{h\in\mathcal{H}}$, where $\mathcal{H}$ is a countable set of positive real numbers having 0 as their only accumulation point, are supposed to be regular in the sense of Ciarlet, i.e.,

$$\sup_{h\in\mathcal{H}} \max_{K\in\mathcal{T}_h} \frac{h_K}{\varrho_K} < \infty,$$

and constructed such that each open $(d-1)$-dimensional facet of a mesh cell lying on Γ is either contained in Γ_D or Γ_N.

For each mesh cell $K \in \mathcal{T}_h$, the set of its facets $E \subset \partial K$ is denoted by $\mathcal{E}_h(K)$. The set of all facets of the triangulation is then called $\mathcal{E}_h \coloneqq \bigcup_{K\in\mathcal{T}_h} \mathcal{E}_h(K)$. This set is further divided into boundary facets $\partial\mathcal{E}_h \coloneqq \{\, E \in \mathcal{E}_h \;:\; E \subset \partial\Omega \,\}$ and the remaining interior facets $\mathcal{E}_h^\mathrm{I}$, i.e., it holds $\mathcal{E}_h = \mathcal{E}_h^\mathrm{I} \cup \partial\mathcal{E}_h$. Concerning the different boundary conditions, the boundary facets are divided into Dirichlet boundary facets $\mathcal{E}_h^\mathrm{D} \coloneqq \{\, E \in \partial\mathcal{E}_h \;:\; E \subset \Gamma_\mathrm{D} \,\}$ and inflow boundary facets $\mathcal{E}_h^-\{\, E \in \partial\mathcal{E}_h \;:\; E \subset \Gamma^- \,\}$, and the notation $\mathcal{E}_h^\mathrm{ID} \coloneqq \mathcal{E}_h^\mathrm{I} \cup \mathcal{E}_h^\mathrm{D}$ is used. Since the family of decompositions is regular, there exists a constant $C > 0$ such that, for all $\mathcal{T}_h$ and all $K \in \mathcal{T}_h$, it holds $h_E \le h_K \le C h_E$, where h_E is the diameter of the facet E of K [DE12, lemma 1.42].

Two adjacent mesh cells $K_i, K_j \in \mathcal{T}_h$ are called neighbors along a facet $E \in \mathcal{E}_h$ if $E = K_i \cap K_j$. The outer unit normal vector on ∂K of a cell K is denoted by $\boldsymbol{n}_K$.

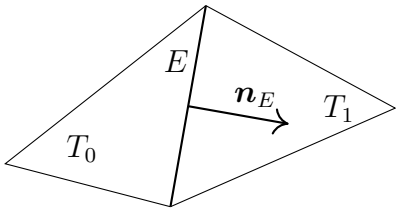


Figure 2.2.: Unit normal vector of an edge $E = T_0 \cap T_1$. Idea taken from [DE12, figure 1.4].

Using an arbitrary but fixed numbering of the cells $K_0, K_1, \ldots$, the unit normal vector $\boldsymbol{n}_E$ on a facet $E \in \mathcal{E}_h$ is defined as

$$\boldsymbol{n}_E \coloneqq \begin{cases} \boldsymbol{n}_K, & \text{if } E \in \partial\mathcal{E}_h \cap \mathcal{E}_h(K) \text{ for a } K \in \mathcal{T}_h, \\ \boldsymbol{n}_{K_i}, & \text{if } K_i \text{ and } K_j \text{ are neighbors along facet } E \text{ and i} < j. \end{cases}$$

In figure 2.2, the unit normal vector is depicted for two triangular cells.

Continuous and discrete spaces

Let $\mathcal{T}_h$ be a triangulation of the domain Ω. Discontinuous Galerkin methods seek their approximate solution in a discrete subspace of the broken Sobolev space

$$H^s(\mathcal{T}_h) \coloneqq \{\, v \in L^2(\Omega) \;:\; v|_K \in H^s(K), \text{ for any } K \in \mathcal{T}_h \,\}, \qquad s \geq 0,$$

which equipped with its usual norm and semi norm

$$\|v\|^2_{H^s(\mathcal{T}_h)} \coloneqq \sum_{K\in\mathcal{T}_h} \|v\|^2_{H^s(K)}, \qquad |v|^2_{H^s(\mathcal{T}_h)} \coloneqq \sum_{K\in\mathcal{T}_h} |v|^2_{H^s(K)},$$

is also a Hilbert space.

Analogously, the broken gradient $\nabla_h : H^1(\mathcal{T}_h) \to [L^2(\Omega)]^d$ can be introduced, which, for any $v \in H^1(\mathcal{T}_h)$ and any $K \in \mathcal{T}_h$, is defined as $(\nabla_h v)\,|_K \coloneqq \nabla\,(v|_K)$. Note that $H^s(\mathcal{T}_h)$ and ∇_h can be seen as generalizations of $H^s(\Omega)$ and ∇, resp., in the sense that $H^s(\Omega) \subset H^s(\mathcal{T}_h)$ and, for any $v \in H^1(\Omega)$, it is $\nabla_h v = \nabla v$ in $[L^2(\Omega)]^d$ [DE12, lemma 1.22]. In the following, the index h at the broken gradient might be dropped if a certain cell is fixed.

The definition of $H^s(\mathcal{T}_h)$ implies that a function $v \in H^s(\mathcal{T}_h)$ might be discontinuous over interior cell interfaces. Indeed, $v|_E$ along a facet $E \in \mathcal{E}_h$ is not even well defined. Therefore, the sign-dependent jump $[\![v]\!]_E$ of a function $v \in H^s(\mathcal{T}_h)$ along facet E is introduced, which is defined as

$$[\![v]\!]_E \coloneqq \begin{cases} v|_{\partial K_i} - v|_{\partial K_j}, & \text{if } K_i \text{ and } K_j \text{ are neighbors along facet } E \text{ and } i < j, \\ v|_{\partial K}, & \text{if } E \in \partial\mathcal{E}_h \cap \mathcal{E}_h(K) \text{ for a } K \in \mathcal{T}_h, \end{cases}$$

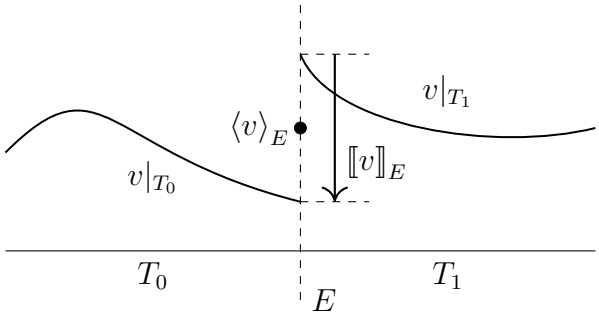


Figure 2.3.: Illustration of the jump and the average in one dimension. Idea taken from [DE12, figure 1.3].

and the average $\langle v \rangle_E$ along E by

$$\langle v \rangle_E \coloneqq \begin{cases} \frac{1}{2}\left(v|_{\partial K_i} + v|_{\partial K_j}\right), & \text{if } K_i \text{ and } K_j \text{ are neighbors along facet } E \text{ and } i < j, \\ v|_{\partial K}, & \text{if } E \in \partial\mathcal{E}_h \cap \mathcal{E}_h(K) \text{ for a } K \in \mathcal{T}_h. \end{cases}$$

In figure 2.3, both the jump and the average are illustrated in the one-dimensional case. For both, the index E is often omitted if no confusion about different facets can arise.

Furthermore, it holds, for any $v \in H^1(\mathcal{T}_h)$, that $v \in H^1(\Omega)$ if and only if $[\![v]\!]_E = 0$ for all $E \in \mathcal{E}_h^{\mathrm{I}}$ [DE12, lemma 1.23].

Last but not least, a discrete function space is needed in which an approximation is sought. Let $P_p(K)$ be the space of polynomials of at most degree $p \in \mathbb{N}$ on simplicial mesh cells K and $Q_p(K)$ the space of tensor products of polynomials of at most degree p on quadrilateral, resp., hexahedral, mesh cells. The discrete space V_h^p in which an approximation is sought is defined as

$$V_h^p \coloneqq \{\, v \in L^2(\Omega) \;:\; v|_K \in \mathcal{R}_p(K), \text{ for any } K \in \mathcal{T}_h \,\} \subset H^s(\mathcal{T}_h),$$

where $\mathcal{R}$ is either P on simplices or Q on quadrilaterals, resp., hexahedrals. It is $V_h^p \subset H^s(\mathcal{T}_h)$, but $V_h^p \not\subset H^s(\Omega)$, and its functions might also be discontinuous over cell interfaces.

2.2.1. Discontinuous Galerkin formulation for diffusion problems

To obtain a DG formulation for the full convection-diffusion-reaction problem 2.1, the problem is split into a pure diffusion problem and a convection-reaction problem.

For the derivation below, it is assumed that $u \in H^1_{\mathrm{D},g_{\mathrm{D}}}(\Omega) \cap H^2(\Omega)$ for the exact solution u of equation (2.5), from which it follows that

$$[\![u]\!]_E = 0, \qquad \text{for any } E \in \mathcal{E}_h, \tag{2.7a}$$

$$[\![\nabla u]\!]_E \cdot \boldsymbol{n} = 0, \qquad \text{for any } E \in \mathcal{E}_h^{\mathrm{I}}, \tag{2.7b}$$

as proven in [DE12, lemma 4.3]. Note that, as stated by Rivière [Riv08, p. 29] and Di Pietro and Ern [DE12, p. 121], it is also sufficient to assume that $u \in H^1_{\mathrm{D},g_\mathrm{D}}(\Omega) \cap H^s(\Omega)$ for some $s > 3/2$.

In the pure diffusive case, i.e., $\boldsymbol{b} \equiv \boldsymbol{0},\ c \equiv 0$, the convection-diffusion-reaction equation (2.4a) is multiplied with a test function $v \in H^1(\mathcal{T}_h)$ and integrated over Ω. Using integration by parts A.1 piecewise on each cell $K \in \mathcal{T}_h$ leads to

$$\sum_{K\in\mathcal{T}_h} \varepsilon \left(\int_K \nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x} - \int_{\partial K} \nabla u \cdot \boldsymbol{n}_K v \,\mathrm{d}s \right) = \int_\Omega f v \,\mathrm{d}\boldsymbol{x}. \tag{2.8}$$

The integrals along the boundaries can be transformed to

$$\sum_{K\in\mathcal{T}_h} \varepsilon \int_{\partial K} \nabla u \cdot \boldsymbol{n}_K v \,\mathrm{d}s = \sum_{E\in\partial\mathcal{E}_h} \varepsilon \int_E \nabla u \cdot \boldsymbol{n}_E v \,\mathrm{d}s + \sum_{E\in\mathcal{E}_h^\mathrm{I}} \varepsilon \int_E [\![\nabla u v]\!] \cdot \boldsymbol{n}_E \,\mathrm{d}s,$$

where $\boldsymbol{n}_{K_j} = -\boldsymbol{n}_{K_i}$ for neighboring cells K_i and K_j, $i \leq j$, and the definition of jumps along facets is utilized. With the Neumann boundary condition, equation (2.4c), and

$$[\![\nabla u v]\!] = [\![\nabla u]\!] \langle v \rangle + \langle \nabla u \rangle [\![v]\!]$$

the integrals can be further manipulated to arrive at

$$\begin{aligned}\sum_{K\in\mathcal{T}_h} \varepsilon \int_{\partial K} \nabla u \cdot \boldsymbol{n}_K v \,\mathrm{d}s = &\sum_{E\in\mathcal{E}_h^\mathrm{D}} \varepsilon \int_E \nabla u \cdot \boldsymbol{n}_E v \,\mathrm{d}s + \int_{\Gamma_\mathrm{N}} g_\mathrm{N} v \,\mathrm{d}s \\ &+ \sum_{E\in\mathcal{E}_h^\mathrm{I}} \varepsilon \int_E [\![\nabla u]\!] \cdot \boldsymbol{n}_E \langle v \rangle + \langle \nabla u \rangle \cdot \boldsymbol{n}_E [\![v]\!] \,\mathrm{d}s.\end{aligned}$$

Due to the assumptions on the regularity of the exact solution u, equation (2.7b) and the definition of the jumps and averages can be used to deduce

$$\sum_{K\in\mathcal{T}_h} \varepsilon \int_{\partial K} \nabla u \cdot \boldsymbol{n}_K v \,\mathrm{d}s = \sum_{E\in\mathcal{E}_h^\mathrm{ID}} \varepsilon \int_E \langle \nabla u \rangle \cdot \boldsymbol{n}_E [\![v]\!] \,\mathrm{d}s + \int_{\Gamma_\mathrm{N}} g_\mathrm{N} v \,\mathrm{d}s.$$

Plugging this in equation (2.8) and rearranging a bit yields

$$\sum_{K\in\mathcal{T}_h} \varepsilon \int_K \nabla u \cdot \nabla v \,\mathrm{d}\boldsymbol{x} - \sum_{E\in\mathcal{E}_h^\mathrm{ID}} \varepsilon \int_E \langle \nabla u \rangle \cdot \boldsymbol{n}_E [\![v]\!] \,\mathrm{d}s = \int_\Omega f v \,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_\mathrm{N}} g_\mathrm{N} v \,\mathrm{d}s. \tag{2.9}$$

Since the bilinear form a defined in section 2.1.2 is symmetric for $\boldsymbol{b} = \boldsymbol{0}$, $c = 0$, one can hope for some sort of symmetry in the discrete formulation as well. To this end, for $\kappa \in \{-1, 0, 1\}$, the terms

$$-\sum_{E\in\mathcal{E}_h^\mathrm{ID}} \varepsilon\kappa \int_E \langle \nabla v \rangle \cdot \boldsymbol{n}_E [\![u]\!] \,\mathrm{d}s + \sum_{E\in\mathcal{E}_h^\mathrm{D}} \varepsilon\kappa \int_E \nabla v \cdot \boldsymbol{n}_E g_\mathrm{D} \,\mathrm{d}s = 0 \tag{2.10}$$

are added to equation (2.9), which are 0 due to the Dirichlet boundary condition (2.4b) and equation (2.7a). In the case $\kappa = 1$, the discrete bilinear form will indeed be symmetric. The choice $\kappa = 0$ leaves it the same, and it will be non-symmetric for $\kappa = -1$.

Moreover, as shown below, the jumps along interfaces and the boundary must be penalized to achieve discrete coercivity. Following the ideas of Kanschat [Kan07], the equation

$$\sum_{E\in\mathcal{E}_h^{\mathrm{I}}} \frac{\sigma_E}{h_E} \int_E [\![u]\!]\,[\![v]\!] \,\mathrm{d}s + \sum_{E\in\mathcal{E}_h^{\mathrm{D}}} \frac{2\sigma_E}{h_E} \int_E [\![u]\!]\,[\![v]\!] \,\mathrm{d}s = \sum_{E\in\mathcal{E}_h^{\mathrm{D}}} \frac{2\sigma_E}{h_E} \int_E g_{\mathrm{D}} v \,\mathrm{d}s \tag{2.11}$$

is considered, where, for any facet $E \in \mathcal{T}_h$, $\sigma_E \in \mathbb{R}$ is a constant, user-dependent value, and once more, equation (2.4b) as well as equation (2.7a) are utilized.

Finally, adding equations (2.10) and (2.11) to equation (2.9) leads to the bilinear form $a_\varepsilon : H^1(\mathcal{T}_h) \times H^1(\mathcal{T}_h) \to \mathbb{R}$ and the functional $F_\varepsilon : H^1(\mathcal{T}_h) \to \mathbb{R}$, which, for $v, w \in H^1(\mathcal{T}_h)$, are defined as

$$\begin{aligned} a_\varepsilon(v,w) \coloneqq & \sum_{K\in\mathcal{T}_h} \varepsilon \int_K \nabla v \cdot \nabla w \,\mathrm{d}\boldsymbol{x} - \sum_{E\in\mathcal{E}_h^{\mathrm{ID}}} \varepsilon \int_E \langle \nabla v \rangle \cdot \boldsymbol{n}_E [\![w]\!] + \kappa \langle \nabla w \rangle \cdot \boldsymbol{n}_E [\![v]\!] \,\mathrm{d}s \\ & + \sum_{E\in\mathcal{E}_h^{\mathrm{I}}} \frac{\sigma_E}{h_E} \int_E [\![v]\!]\,[\![w]\!] \,\mathrm{d}s + \sum_{E\in\mathcal{E}_h^{\mathrm{D}}} \frac{2\sigma_E}{h_E} \int_E [\![v]\!]\,[\![w]\!] \,\mathrm{d}s, \end{aligned} \tag{2.12}$$

$$\begin{aligned} F_\varepsilon(v) \coloneqq & \int_\Omega f v \,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_{\mathrm{N}}} g_{\mathrm{N}} v \,\mathrm{d}s - \sum_{E\in\mathcal{E}_h^{\mathrm{D}}} \varepsilon\kappa \int_E \nabla v \cdot \boldsymbol{n}_E g_{\mathrm{D}} \,\mathrm{d}s \\ & + \sum_{E\in\mathcal{E}_h^{\mathrm{D}}} \frac{2\sigma_E}{h_E} \int_E g_{\mathrm{D}} v \,\mathrm{d}s. \end{aligned} \tag{2.13}$$

Note that even though during the construction $u \in H^1_{\mathrm{D},g_{\mathrm{D}}}(\Omega) \cap H^2(\Omega)$ is assumed, the bilinear form is still well defined for any $u \in H^1(\mathcal{T}_h)$.

Remark 2.4 (SIPG, IIPG, NIPG). If the parameter κ is chosen to equal one, a_ε is symmetric and called *symmetric interior penalty Galerkin* (SIPG) bilinear form. In the case $\kappa = 0$, it is denoted as *incomplete interior penalty Galerkin* (IIPG) bilinear form. Lastly, if $\kappa = -1$, it is often referred to as *non-symmetric interior penalty Galerkin* (NIPG) bilinear form. □

Finally, it is mentionable that the choice of the user-dependent penalty parameter σ_E depends on the choice of κ and the polynomial degree and is discussed below.

To conclude, in the case of pure diffusion, the discontinuous Galerkin formulation reads as follows.

Problem 2.5 (DG formulation for pure diffusion problems)**.** For given $\kappa \in \{-1, 0, 1\}$ and $\sigma_E \in \mathbb{R}$ for any $E \in \mathcal{T}_h$, find $u_h \in V_h^p$ such that

$$a_\varepsilon(u_h, v_h) = F_\varepsilon(v_h), \qquad \text{for any } v_h \in V_h^p, \tag{2.14}$$

where a_ε and F_ε are defined in equations (2.12) and (2.13). □

Remark 2.6 (Consistency and Galerkin orthogonality). By construction, it is clear that a_ε is *consistent* for any fixed $\kappa \in \{-1, 0, 1\}$ and given $\sigma_E \in \mathbb{R}$ for all $E \in \mathcal{T}_h$, in the sense that, for the exact solution $u \in H^1_{\mathrm{D},g_\mathrm{D}}(\Omega) \cap H^2(\Omega)$ and all $v_h \in V_h^p$, it is

$$a_\varepsilon(u, v_h) = F_\varepsilon(v_h).$$

Or in other words, assuming that there exists a $u_h \in V_h^p$ such that equation (2.14) holds true for all $v_h \in V_h^p$, the so-called *Galerkin orthogonality* is satisfied, i.e.,

$$a_\varepsilon(u - u_h, v_h) = 0, \qquad \text{for any } v_h \in V_h^p.$$

□

Unique existence of a discrete solution

In this section, it is investigated under which conditions problem 2.5 is well defined in the sense that it has a unique solution. To this extent, a discrete version of the Lax–Milgram lemma, corollary A.6, is employed for which a discrete norm and discrete coercivity of the bilinear form are needed.

Definition 2.7 (Energy norm in the diffusive case)**.** Assume $\sigma_E > 0$ for all $E \in \mathcal{E}_h$. Let $|||\cdot|||_\varepsilon : H^1(\mathcal{T}_h) \to \mathbb{R}$ be defined, for any $v \in H^1(\mathcal{T}_h)$, as

$$|||v|||_\varepsilon^2 \coloneqq \sum_{K \in \mathcal{T}_h} \varepsilon \|\nabla v\|^2_{L^2(K)} + \sum_{E \in \mathcal{E}_h^\mathrm{I}} \frac{\sigma_E}{h_E} \| [\![v]\!] \|^2_{L^2(E)} + \sum_{E \in \mathcal{E}_h^\mathrm{D}} \frac{2\sigma_E}{h_E} \| [\![v]\!] \|^2_{L^2(E)}.$$

This norm is called *energy norm.* □

Indeed, $|||\cdot|||_\varepsilon$ is a norm on V_h^p, if $\sigma_E > 0$ for all $E \in \mathcal{E}_h$, which follows directly from $\|\cdot\|_{L^2(K)}$ and $\|\cdot\|_{L^2(E)}$ being norms on K and E, respectively.

The following lemma states conditions under which $a_\varepsilon(\cdot, \cdot)$ is coercive with respect to $|||\cdot|||_\varepsilon$, i.e., it exists a positive constant $\beta_h \in \mathbb{R}$ such that $a_\varepsilon(v_h, v_h) \geq \beta_h |||v_h|||^2_\varepsilon$, for any $v_h \in V_h^p$.

Lemma 2.8. *Let n_0 be the maximal number of facets an element $K \in \mathcal{T}_h$ can have, i.e., three for triangles, four for quadrilaterals and tetrahedrons and six for hexahedrons, and C_tr be the constant from the trace inequality[3]. If one of the conditions*

- *$\kappa = 1$ and $\sigma_E \geq 2\varepsilon C_\mathrm{tr}^2 n_0$ for all $E \in \mathcal{T}_h$,*
- *$\kappa = 0$ and $\sigma_E \geq \varepsilon C_\mathrm{tr}^2 n_0$ for all $E \in \mathcal{T}_h$,*

[3] See, e.g., [Riv08, equation 2.1]

- $\kappa = -1$ *and* $\sigma_E > 0$ *for all* $E \in \mathcal{T}_h$

hold, then the discrete bilinear form a_ε *given in equation* (2.12) *is coercive with respect to* $|||\cdot|||_\varepsilon$.

Proof. The discrete coercivity in the non-symmetric case, i.e., $\kappa = -1$ follows directly from the definition of a_ε. The other two assertions can be proven using Cauchy–Schwarz', the trace, and Young's inequality; see [Riv08, section 2.7.1]. □

With the discrete coercivity, it follows quickly that problem 2.5 possesses a unique solution.

Theorem 2.9. *Let* σ_E *be such that* a_ε *is coercive with respect to* $|||\cdot|||_\varepsilon$, *cf. lemma 2.8. Then, the discrete problem 2.5 has a unique solution.*

Proof. In light of corollary A.6 it is $V_h = V_h^p$, $\|\cdot\|_h = |||\cdot|||_\varepsilon$, $a_h = a_\varepsilon$ and $F_h = F_\varepsilon$. By definition, V_h^p is a Hilbert space, and due to the linearity of integrals, the functional F_ε is linear. With the given coercivity of a_ε corollary A.6 can be applied, and the assertion of this theorem follows. □

Convergence analysis

Since both problem 2.1 and problem 2.5 have a unique (weak) solution, it can be asked how large the error between these functions is. This question is tackled in this section.

First, error estimates with respect to the energy norm $|||\cdot|||_\varepsilon$ can be proven.

Theorem 2.10 (Error estimates in the energy norm)**.** *Let* u *be the weak solution to problem 2.1 and* u_h *be the solution to problem 2.5. If* $u \in H^1_{\mathrm{D},g_\mathrm{D}}(\Omega) \cap H^{p+1}(\Omega)$, *the penalty parameters are chosen as in lemma 2.8, and the Dirichlet data* g_D *is a continuous piecewise polynomial of at most degree* p, *then it holds*

$$|||u - u_h|||_\varepsilon \leq Ch^p |u|_{H^{p+1}(\mathcal{T}_h)},$$

where $0 < C \in \mathbb{R}$ *is a constant independent of the mesh width* h.

Proof. See [Riv08, theorem 2.13]. □

Remark 2.11. With similar but slightly different conditions for the penalty terms σ_E the same result can be proven without the assumption that g_D is a continuous piecewise polynomial of at most degree p; see, e.g., [DF15, theorem 2.44] for the SIPG, IIPG and NIPG, and [Kan07, theorem 2.2.10] as well as [DE12, corollary 4.18] for the SIPG method. □

Next, the error with respect to the L^2-norm can be estimated.

Theorem 2.12 (Error estimates in the L^2-norm)**.** *If the assumptions of theorem 2.10 are satisfied and elliptic regularity*[4] *holds true for the model problem, then it holds*

$$\begin{aligned} \|u - u_h\|_{L^2(\Omega)} &\leq Ch^{p+1}\|u\|_{H^{p+1}(\mathcal{T}_h)} && \text{(SIPG)}, \\ \|u - u_h\|_{L^2(\Omega)} &\leq Ch^{p}\|u\|_{H^{p+1}(\mathcal{T}_h)} && \text{(IIPG, NIPG)}, \end{aligned}$$

where again $0 < C \in \mathbb{R}$ *is a constant independent of the mesh width* h*.*

Proof. See, e.g., [Riv08, theorem 2.14] and [DE12, corollary 4.26]. □

Hence, the DG method converges with optimal order in the energy norm and the L^2-norm in the SIPG case and suboptimally in the L^2-norm in the IIPG and NIPG cases. On the one hand, optimal error estimates in the L^2-norm for all variants can be proven if the so-called *superpenalization* is used [Riv08, theorem 2.14]. On the other hand, this may lead to more computational costs [DF15, remark 2.51].

Remark 2.13 (Optimal order for IIPG and NIPG for odd degrees). Numerically, it can be observed that on uniform meshes the IIPG and NIPG method show optimal converge rates for odd degrees and suboptimal rates for even degrees, i.e.,

$$\|u - u_h\|_{L^2(\Omega)} \leq C\|u\|_{H^{p+1}(\mathcal{T}_h)} \begin{cases} h^{p+1}, & \text{if } \kappa = 1 \text{ (SIPG)}, \\ h^{p+1}, & \text{if } p \text{ odd, } \kappa = 0 \text{ (IIPG) or } \kappa = -1 \text{ (NIPG)}, \\ h^{p}, & \text{if } p \text{ even, } \kappa = 0 \text{ (IIPG) or } \kappa = -1 \text{ (NIPG)} \end{cases}$$

can be observed [Riv08, p. 49]. The theoretical reason for this is still not found. On general meshes, examples can be constructed where these methods produce suboptimal rates even for odd polynomial degrees, showing that the estimates in theorem 2.35 are sharp [Riv08, p. 49]. □

2.2.2. Discontinuous Galerkin formulation for convection-reaction problems

In this section a DG formulation for the convective-reactive part of problem (2.4) is derived, i.e., $\varepsilon \equiv 0$ and $\boldsymbol{b} \neq \mathbf{0}$ a.e. in Ω are assumed. The problem then reads as follows.

Problem 2.14 (Strong form of the convection-reaction problem)**.** Let the assumptions on the domain and the coefficient functions from problem 2.1 hold. Furthermore, assume that there exists a $\mu_0 \in \mathbb{R}$, $\mu_0 > 0$ such that

$$c - \frac{1}{2}\operatorname{div} \boldsymbol{b} \geq \mu_0$$

[4] See, e.g., [DE12, definition 4.24] for a definition of elliptic regularity.

a.e. in Ω, and it is $g_\mathrm{D} \in \left\{ v \text{ is measurable on } \partial\Omega \ : \ \int_{\partial\Omega} |\boldsymbol{b}\cdot\boldsymbol{n}| v^2 \,\mathrm{d}s < \infty \right\}$ after g_D is extended by 0 to $\partial\Omega$.

Find a smooth enough u such that

$$\boldsymbol{b}\cdot\nabla u + cu = f \qquad \text{in } \Omega, \tag{2.15a}$$

$$u = g_\mathrm{D} \qquad \text{along } \Gamma_-, \tag{2.15b}$$

□

The problem can be recast in the following weak formulation, cf. [DE12, sections 2.1.4, 2.1.6] for the derivation.

Problem 2.15 (Variational form of the convection-reaction problem)**.** Under the assumptions of problem 2.14, find a $u \in V \coloneqq \{ v \in L^2(\Omega) \ : \ \boldsymbol{b}\cdot\nabla v \in L^2(\Omega) \}$ such that, for any, $v \in V$, it holds

$$\sum_{K\in\mathcal{T}_h} \int_K (\boldsymbol{b}\cdot\nabla u + cu)\, v \,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_-} \boldsymbol{b}\cdot\boldsymbol{n} uv \,\mathrm{d}s = \int_\Omega fv \,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_-} \boldsymbol{b}\cdot\boldsymbol{n} g_\mathrm{D} v \,\mathrm{d}s.$$

□

Unique existence of a solution to problem 2.15 does not follow from its weak counterpart, problem 2.2, directly. Nevertheless, it can be proven.

Theorem 2.16 (Weak solution to problem 2.14)**.** *It exists a unique solution to problem 2.15, i.e., problem 2.14 has a unique weak solution.*

Proof. See [DE12, theorem 2.12]. □

The next step is to obtain a DG discretization of problem 2.14. To this extent, it is assumed that the exact solution is in $V \cap H^1(\mathcal{T}_h)$.

To start the derivation, equation (2.15a) is multiplied with a test function $v \in H^1(\mathcal{T}_h)$ and integrated over the domain leading to

$$\sum_{K\in\mathcal{T}_h} \int_K (\boldsymbol{b}\cdot\nabla u + cu)\, v \,\mathrm{d}\boldsymbol{x} = \int_\Omega fv \,\mathrm{d}\boldsymbol{x}.$$

To incorporate information about the inflow boundary condition (2.15b) the equality

$$\int_{\Gamma_-} \boldsymbol{b}\cdot\boldsymbol{n} uv \,\mathrm{d}s = \int_{\Gamma_-} \boldsymbol{b}\cdot\boldsymbol{n} g_\mathrm{D} v \,\mathrm{d}s$$

is subtracted yielding

$$\sum_{K\in\mathcal{T}_h} \int_K (\boldsymbol{b}\cdot\nabla u + cu)\, v \,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_-} \boldsymbol{b}\cdot\boldsymbol{n} uv \,\mathrm{d}s = \int_\Omega fv \,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_-} \boldsymbol{b}\cdot\boldsymbol{n} g_\mathrm{D} v \,\mathrm{d}s.$$

To achieve coercivity of the bilinear form the term

$$\sum_{E\in\mathcal{E}_h^{\mathrm{I}}}\int_E \boldsymbol{b}\cdot\boldsymbol{n}_E\,[\![u]\!]\,\langle v\rangle\;\mathrm{d}s = 0$$

is subtracted that vanishes, since for $u \in V \cap H^1(\mathcal{T}_h)$ it holds $[\![\boldsymbol{b}\cdot\boldsymbol{n}u]\!] = 0$ [DE12, lemma 2.14].

At this point, the method would converge, however, only suboptimal; see, e.g., [DE12, corollary 2.24]. Therefore, the method is stabilized by adding

$$\sum_{E\in\mathcal{E}_h^{\mathrm{I}}}\frac{\eta}{2}\int_E |\boldsymbol{b}\cdot\boldsymbol{n}_E|\,[\![u]\!]\,[\![v]\!]\;\mathrm{d}s = 0,$$

with some $\eta \geq 0$, which again vanishes due to $[\![\boldsymbol{b}\cdot\boldsymbol{n}u]\!] = 0$. Especially interesting are the choices $\eta = 1$ and $\eta = 0$, resp., which in context of finite volume schemes lead to so-called *upwind fluxes* and *centered fluxes*; see, e.g., [BMS04; DE12, sections 2.2.3, 2.3.4]. However, letting η be a user-dependent parameter gives the user more freedom since, in principle, other values than zero and one might be useful as well [BMS04].

Adding the aforementioned equations leads to the discontinuous Galerkin formulation for the convection-reaction problem.

Problem 2.17 (DG formulation for convection-reaction problems)**.** For a given $\eta \geq 0$, find $u_h \in V_h^p$ such that

$$a_{\boldsymbol{bc}}(u_h, v_h) = F_{\boldsymbol{bc}}(v_h), \qquad \text{for any } v_h \in V_h^p, \tag{2.16}$$

where $a_{\boldsymbol{bc}} : H^1(\mathcal{T}_h)\times H^1(\mathcal{T}_h) \to \mathbb{R}$ and $F_{\boldsymbol{bc}} : H^1(\mathcal{T}_h) \to \mathbb{R}$ are, for $v, w \in H^1(\mathcal{T}_h)$, defined as

$$\begin{aligned} a_{\boldsymbol{bc}}(v,w) \coloneqq{}& \sum_{K\in\mathcal{T}_h}\int_K (\boldsymbol{b}\cdot\nabla v + cv)\,w\,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_-}\boldsymbol{b}\cdot\boldsymbol{n}vw\,\mathrm{d}s \\ &- \sum_{E\in\mathcal{E}_h^{\mathrm{I}}}\int_E \boldsymbol{b}\cdot\boldsymbol{n}_E\,[\![v]\!]\,\langle w\rangle\;\mathrm{d}s + \sum_{E\in\mathcal{E}_h^{\mathrm{I}}}\frac{\eta}{2}\int_E |\boldsymbol{b}\cdot\boldsymbol{n}_E|\,[\![v]\!]\,[\![w]\!]\;\mathrm{d}s \\ F_{\boldsymbol{bc}}(v) \coloneqq{}& \int_\Omega fv\,\mathrm{d}\boldsymbol{x} - \int_{\Gamma_-}\boldsymbol{b}\cdot\boldsymbol{n}g_{\mathrm{D}}v\,\mathrm{d}s. \end{aligned}$$

□

Remark 2.18 (Consistency and Galerkin orthogonality). As for the pure diffusive case, the problem is *consistent* under the assumption that the exact solution u is in $V \cap H^1(\mathcal{T}_h)$. That is, assuming there is a $u_h \in V_h^p$ that satisfies equation (2.16) for every $v_h \in V_h^p$, the *Galerkin orthogonality*

$$a_{\boldsymbol{bc}}(u - u_h, v_h) = 0 \qquad \text{for every } v_h \in V_h^p$$

holds. □

Unique existence of a discrete solution

As in section 2.2.1, after the discrete problem 2.17 is defined, it has to be verified that it is well posed, i.e., that it exists a (unique) solution to the problem.

To this end, the following norm is defined.

Definition 2.19 (Energy norm in the convection-reaction case)**.** For any $v \in H^1(\mathcal{T}_h)$, let $|||\cdot|||_{\boldsymbol{bc}} : H^1(\mathcal{T}_h) \to \mathbb{R}$ be defined as

$$|||v|||_{\boldsymbol{bc}}^2 \coloneqq \mu_0 \|v\|_{L^2(\Omega)}^2 + \int_{\partial\Omega} \frac{1}{2}|\boldsymbol{b} \cdot n| v^2 \, \mathrm{d}s + \sum_{E \in \mathcal{E}_h^{\mathrm{I}}} \frac{\eta}{2} \int_E |\boldsymbol{b} \cdot \boldsymbol{n}_E| \, [\![v]\!]^2 \, \mathrm{d}s,$$

where μ_0 is given in problem 2.14 and $\eta \geq 0$. This norm is also called *energy norm.* □

It holds that $a_{\boldsymbol{bc}}$ is coercive with respect to this norm, as the following lemma states.

Lemma 2.20. *Let $\eta \geq 0$. It holds, for any $v_h \in V_h^p$,*

$$a_{\boldsymbol{bc}}(v_h, v_h) \geq C|||v_h|||_{\boldsymbol{bc}}^2,$$

where $C \in \mathbb{R}$ is a positive constant.

Proof. See [DE12, lemma 2.27, remark 2.19]. □

Based on the coercivity, it can be proven that problem 2.17 has a unique solution.

Theorem 2.21. *Let $\eta \geq 0$. Then, it holds that the discrete problem 2.17 has a unique solution.*

Proof. This follows from lemma 2.20 and again the discrete version of the Lax–Milgram lemma, corollary A.6, with $(V_h, \|\cdot\|_h) = (V_h^p, |||\cdot|||_{\boldsymbol{bc}})$, $a_h = a_{\boldsymbol{bc}}$ and $F_h = F_{\boldsymbol{bc}}$. □

Convergence analysis

As before, after it is proven that the discrete problem 2.17 has a unique solution, the difference between the exact weak solution and the discrete solution can be examined.

Theorem 2.22 (Error estimates in the energy norm)**.** *Let u be the weak solution to problem 2.14 and u_h be the solution to problem 2.17. If $u \in V \cap H^{p+1}(\Omega)$ and $\eta = 1$, then it holds*

$$|||u - u_h|||_{\boldsymbol{bc}} \leq C h^{p+1/2} \|u\|_{H^{p+1}(\mathcal{T}_h)},$$

where $0 < C \in \mathbb{R}$ is a constant independent of the mesh width h.

Proof. See [DE12, corollary 2.32]. □

Remark 2.23. In the previous theorem the upwind bilinear form, i.e., $\eta = 1$, is used. If the non-stabilized version, i.e., $\eta = 0$, is used, then the convergence rate deteriorates to h^p [DE12, corollary 2.24]. □

Remark 2.24. As proven, e.g., in [DE12, section 2.3.3], if the mesh is fine enough, $\eta = 1$ and the problem is not reaction-dominated, then it is also possible to obtain the same convergence rates in a stronger energy norm $|||\cdot|||_{\boldsymbol{b}c,*}$ that is defined, for $v \in H^1(\mathcal{T}_h)$, as

$$|||v|||_{\boldsymbol{b}c,*}^2 \coloneqq |||v|||_{\boldsymbol{b}c}^2 + \sum_{K \in \mathcal{T}_h} \frac{h_K}{\|\boldsymbol{b}\|_{[L^\infty(\Omega)]^d}} \|\boldsymbol{b} \cdot \nabla v\|_{L^2(K)}^2.$$

In other words, the upwind DG method controls the streamline derivative even without another special stabilization term. □

Theorem 2.25 (Error estimates in the L^2-norm)**.** *Under the assumptions of theorem 2.22 it holds that*

$$\|u - u_h\|_{L^2(\Omega)} \leq C h^{p+1/2} \|u\|_{H^{p+1}(\mathcal{T}_h)},$$

where $0 < C \leq \infty$ *is independent of* h.

Proof. See, e.g., [Kan07, corollary 1.4.14]. □

Remark 2.26. The previous theorem states that the convergence rate in the L^2-norm is suboptimal by a factor of 1/2. Unfortunately, this estimate is sharp, as shown by Peterson [Pet91]. However, if $\boldsymbol{b}$ is constant, the mesh consists of rectangles only, and the solution is in $W^{p+2,2}(\Omega) \cap W^{p+1,\infty}(\Omega)$, then optimal error estimates can be proven [Kan07, theorem 1.4.16]. □

2.2.3. Discontinuous Galerkin formulation for the full problem

At this point, the individual parts of the problem are examined. This section brings it all together to formulate and investigate a DG formulation for the full convection-diffusion-reaction problem 2.2.

Combining the discretizations of sections 2.2.1 and 2.2.2 results in the discontinuous Galerkin formulation for the convection-diffusion-reaction problem 2.1:

Problem 2.27 (DG formulation for convection-diffusion-reaction problems)**.** Let $\kappa \in \{1, 0, -1\}$ and $\sigma_E \in \mathbb{R}$ for all $E \in \mathcal{T}_h$, and $\eta \geq 0$ be given. Find $u_h \in V_h^p$ such that

$$a_h(u_h, v_h) = F_h(v_h), \qquad \text{for any } v_h \in V_h^p, \tag{2.17}$$

where the bilinear and linear form $a_h : H^1(\mathcal{T}_h) \times H^1(\mathcal{T}_h) \to \mathbb{R}$ and $F_h : H^1(\mathcal{T}_h) \to \mathbb{R}$, for every $v, w \in H^1(\mathcal{T}_h)$, are given by

$$\begin{aligned}
a_h(v,w) \coloneqq & \sum_{K\in\mathcal{T}_h} \varepsilon \int_K \nabla v \cdot \nabla w + (\boldsymbol{b}\cdot\nabla v + cv)\, w \,\mathrm{d}\boldsymbol{x} \\
& - \sum_{E\in\mathcal{E}_h^{\mathrm{ID}}} \varepsilon \int_E \langle \nabla v\rangle \cdot \boldsymbol{n}_E \,[\![w]\!] + \kappa \,\langle \nabla w\rangle \cdot \boldsymbol{n}_E \,[\![v]\!] \,\mathrm{d}s \\
& + \sum_{E\in\mathcal{E}_h^{\mathrm{I}}} \frac{\sigma_E}{h_E} \int_E [\![v]\!]\,[\![w]\!] \,\mathrm{d}s + \sum_{E\in\mathcal{E}_h^{\mathrm{D}}} \frac{2\sigma_E}{h_E} \int_E [\![v]\!]\,[\![w]\!] \,\mathrm{d}s - \int_{\Gamma_-} \boldsymbol{b}\cdot\boldsymbol{n} v w \,\mathrm{d}s \\
& - \sum_{E\in\mathcal{E}_h^{\mathrm{I}}} \int_E \boldsymbol{b}\cdot\boldsymbol{n}_E \,[\![v]\!]\,\langle w\rangle \,\mathrm{d}s + \sum_{E\in\mathcal{E}_h^{\mathrm{I}}} \frac{\eta}{2} \int_E |\boldsymbol{b}\cdot\boldsymbol{n}_E| \,[\![v]\!]\,[\![w]\!] \,\mathrm{d}s, \\
F_h(v) \coloneqq & \int_\Omega f v \,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_{\mathrm{N}}} g_{\mathrm{N}} v \,\mathrm{d}s - \sum_{E\in\mathcal{E}_h^{\mathrm{D}}} \varepsilon\kappa \int_E \nabla v\cdot\boldsymbol{n}_E g_{\mathrm{D}} \,\mathrm{d}s + \sum_{E\in\mathcal{E}_h^{\mathrm{D}}} \frac{2\sigma_E}{h_E} \int_E g_{\mathrm{D}} v \,\mathrm{d}s \\
& - \int_{\Gamma_-} \boldsymbol{b}\cdot\boldsymbol{n} g_{\mathrm{D}} v \,\mathrm{d}s.
\end{aligned}$$

□

A direct consequence of the previous sections is that the method is consistent.

Remark 2.28 (Consistency and Galerkin orthogonality). Assume that the exact solution u is in $H^1_{\mathrm{D},g_{\mathrm{D}}}(\Omega) \cap H^2(\mathcal{T}_h)$. As before, the problem is *consistent*, and assuming there is a $u_h \in V_h^p$ that satisfies equation (2.17) for every $v_h \in V_h^p$, it holds the *Galerkin orthogonality*

$$a_h(u - u_h, v_h) = 0, \qquad \text{for any } v_h \in V_h^p.$$

□

Unique existence of a discrete solution

The unique existence of a solution to problem 2.27 follows from the unique existence of solutions to problem 2.5 and problem 2.17. To this end, an energy norm is defined that is the sum of the contributions of the energy norms of the individual problems.

Definition 2.29 (Energy norm in the convection-diffusion-reaction case)**.** For any $v \in H^1(\mathcal{T}_h)$, let $|||\cdot|||_h : H^1(\mathcal{T}_h) \to \mathbb{R}$ be defined as

$$|||v|||_h^2 \coloneqq |||v|||_\varepsilon^2 + |||v|||_{\boldsymbol{b}c}^2,$$

where $|||v|||_\varepsilon$ and $|||v|||_{\boldsymbol{b}c}$ are defined in definitions 2.7 and 2.19, respectively. This norm is once more called *energy norm*. □

The map $|||\cdot|||_h$ is a norm on V_h^p, which follows from $|||v|||_\varepsilon$ and $|||v|||_{\boldsymbol{bc}}$ being norms, if $\sigma_E > 0$ for all $E \in \mathcal{E}_h$, $c - \operatorname{div}(\boldsymbol{b})/2 \geq \mu_0 > 0$ and $\eta \geq 0$.

From this definition, it follows that a_h is coercive with respect to the energy norm, i.e., there exists a $\beta_h > 0$ such that $a_h(v_h, v_h) \geq \beta_h |||v_h|||_h^2$, for any $v_h \in V_h^p$.

Lemma 2.30. *Let n_0 be the maximal number of facets an element $K \in \mathcal{T}_h$ can have, and C_{tr} be the constant from the trace inequality[5]. If $\eta \geq 0$, and one of the conditions*

- *$\kappa = 1$ and $\sigma_E \geq 2\varepsilon C_{\mathrm{tr}}^2 n_0$ for all $E \in \mathcal{T}_h$,*
- *$\kappa = 0$ and $\sigma_E \geq \varepsilon C_{\mathrm{tr}}^2 n_0$ for all $E \in \mathcal{T}_h$,*
- *$\kappa = -1$ and $\sigma_E > 0$ for all $E \in \mathcal{T}_h$*

hold, then the discrete bilinear form a_h given in problem 2.27 is coercive with respect to $|||\cdot|||_h$.

Proof. This is a direct consequence of lemmas 2.8 and 2.20 and the definitions of a_h and $|||\cdot|||_h$. □

Therefore, it follows that problem 2.27 has a unique discrete solution.

Theorem 2.31. *Let $\eta \geq 0$ and σ_E be such that a_h is coercive with respect to $|||\cdot|||_h$, cf. lemma 2.30. Then, it holds that the discrete problem 2.27 has a unique solution.*

Proof. This follows from lemma 2.30 and corollary A.6 with $(V_h, \|\cdot\|_h) = (V_h^p, |||\cdot|||_h)$. □

Convergence analysis

Since it is now proven that the continuous problem 2.2 has a unique weak solution u and the discrete problem 2.27 has a unique discrete solution u_h, in this section it is investigated how large the error $u - u_h$ is.

Theorem 2.32 (Error estimates in the energy norm)**.** *Let u be the weak solution to problem 2.1 and u_h be the solution to problem 2.27. If $u \in H^1_{\mathrm{D},g_{\mathrm{D}}}(\Omega) \cap H^{p+1}(\Omega)$, the penalty parameters are chosen as in lemma 2.30, the Dirichlet data g_{D} is a continuous piecewise polynomial of at most degree p and $\eta = 1$, then it holds*

$$|||u - u_h|||_h \leq \left(C_\varepsilon + C_{\boldsymbol{bc}} h^{1/2}\right) h^p \|u\|_{H^{p+1}(\mathcal{T}_h)},$$

where $C_\varepsilon \in \mathbb{R}$ and $C_{\boldsymbol{bc}} \in \mathbb{R}$ are positive constants independent of the mesh width h, but dependent on the data ε, $\boldsymbol{b}$ and c, respectively.

Proof. This follows from theorems 2.10 and 2.22. □

[5] See, e.g., [Riv08, equation 2.1].

Example 2.33 (Examples for C_ε and $C_{\boldsymbol{bc}}$)**.** In the literature some explicit dependencies of C_ε and $C_{\boldsymbol{bc}}$ with respect to the parameters of the convection-diffusion-reaction problem 2.1 are given:

1. Kanschat [Kan07, theorem 5.1.7] has proven in a different norm that, if $\kappa = 1$ and $c \geq 0$ is constant, then it is
$$C_\varepsilon \propto \sqrt{\varepsilon}, \qquad C_{\boldsymbol{bc}} \propto \left(\sqrt{\|\boldsymbol{b}\|_{[L^\infty(\Omega)]^d}} + \sqrt{c}h^{1/2}\right).$$
2. Again in the SIPG case ($\kappa = 1$), Di Pietro and Ern [DE12, equation (4.82)] have proven for a slightly different method and in a slightly different norm that
$$C_\varepsilon \propto \sqrt{\varepsilon}, \qquad C_{\boldsymbol{bc}} \propto \left(\sqrt{\|\boldsymbol{b}\|_{[L^\infty(\Omega)]^d}} + \sqrt{\max\left(\|c\|_{L^\infty(\Omega)}, L_{\boldsymbol{b}}\right)}h^{1/2}\right),$$
where $L_{\boldsymbol{b}}$ is the Lipschitz constant for $\boldsymbol{b}$.
3. In the NIPG case ($\kappa = -1$), Houston, Schwab, and Süli [HSS02] have proven in a slightly different norm that
$$C_\varepsilon \propto \sqrt{\varepsilon}, \qquad C_{\boldsymbol{bc}} \propto \left(\sqrt{\|\boldsymbol{b}\|_{[L^\infty(\Omega)]^d}} + C(\operatorname{div}(\boldsymbol{b}), c, \mu)\|\mu\|_{L^\infty(\Omega)}h^{1/2}\right),$$
where $\mu^2 \coloneqq c - \operatorname{div}(\boldsymbol{b})/2 \geq 0$, and $C(\operatorname{div}(\boldsymbol{b}), c, \mu) \in \mathbb{R}$ is a positive constant that depends on $\operatorname{div}(\boldsymbol{b})$, c and μ.

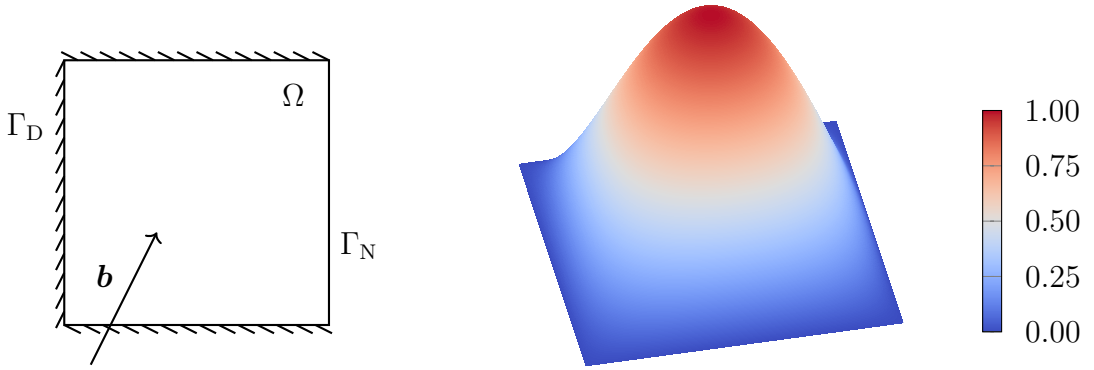
Note that none of the above-mentioned constants deteriorate if $\varepsilon \to 0$, i.e., also the constant in the previous error estimate does not deteriorate as ε tends to zero. □

Remark 2.34. The previous theorem shows that if the diffusion is dominant, the energy error converges with order p. On the other hand, in the convection-dominated case, an error reduction order of $p + 1/2$ can be expected. □

Last but not least, the error in the L^2-norm can be estimated.

Theorem 2.35 (Error estimates in the L^2-norm)**.** *Under the assumptions of theorem 2.32, it holds*
$$\|u - u_h\|_{L^2(\Omega)} \leq \frac{1}{\sqrt{\mu_0}}\left(C_\varepsilon + C_{\boldsymbol{bc}}h^{1/2}\right)h^p\|u\|_{H^{p+1}(\mathcal{T}_h)},$$
where μ_0 is given in problem 2.14, and $C_\varepsilon \in \mathbb{R}$ and $C_{\boldsymbol{bc}} \in \mathbb{R}$ are given in theorem 2.32.

Proof. Using the definitions of $|||\cdot|||_{\boldsymbol{bc}}$ and $|||\cdot|||_h$ such as theorem 2.32 it follows
$$\begin{aligned}\|u - u_h\|^2_{L^2(\Omega)} &= \frac{1}{\mu_0}\left(\mu_0\|u - u_h\|^2_{L^2(\Omega)}\right) \leq \frac{1}{\mu_0}\left(|||u - u_h|||^2_{\boldsymbol{bc}}\right) \leq \frac{1}{\mu_0}\left(|||u - u_h|||^2_h\right)\\ &\leq \frac{1}{\mu_0}\left(\left(C_\varepsilon + C_{\boldsymbol{bc}}h^{1/2}\right)h^p\|u\|_{H^{p+1}(\mathcal{T}_h)}\right)^2.\end{aligned}$$
Applying the square root on both sides concludes the proof. □



(a) Domain with Dirichlet and Neumann boundaries and direction of convection field.

(b) Exact solution $u(x, y) = \sin(\pi x)\sin(\pi y)$.

Figure 2.4.: Domain, direction of convection field and exact solution to example 2.37. Lines with ticks indicate Dirichlet boundary Γ_D and lines without ticks indicate Neumann boundary Γ_N.

Remark 2.36. The previous theorem shows that the L^2-error converges suboptimally by a factor of 1 in the diffusion-dominated case and of $1/2$ in the convection-dominated case. However, it is important to mention that this is only a lower bound, and the convergence rates might be better in practice. Especially in the diffusion-dominated case, the convergence rate described in theorem 2.12 and remark 2.13 can often be seen in practice. □

2.3. Numerical studies

To verify the implementation and the theoretical results of the previous sections, the DG method (2.17) is tested in this section on examples with known solutions.

For the experiments, the in-house research software ParMooN is used [Gan+16; Wil+17]. The sparse direct solver UMFPACK is utilized to solve the occurring linear systems of equations [Dav04].

2.3.1. A two-dimensional problem

To begin with, the following two-dimensional example is investigated.

Example 2.37 (Sine Laplace problem in 2D)**.** This problem is stated in the unit square $\Omega := (0, 1)^2$ with $\boldsymbol{b} := (1, 2)^T$ and $c := 3$. The right edge connecting $(1, 0)$ and $(1, 1)$ is set as the Neumann boundary, while on the remaining boundary, Dirichlet conditions are prescribed; see also figure 2.4a. The right-hand side f and the boundary

conditions are chosen such that

$$u(x, y) \coloneqq \sin(\pi x) \sin(\pi y)$$

is the exact solution which is depicted in figure 2.4b. □

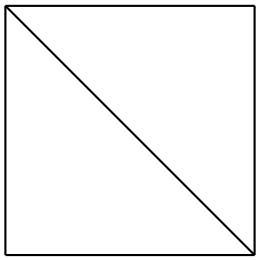
To gain insight into convergence orders of the SIPG, IIPG, and the NIPG discretization in the diffusion-dominated and the convection-dominated regime, the diffusion constant is chosen to be $\varepsilon = 1$ and $\varepsilon = 10^{-8}$. While the upwind discretization is utilized for all $\kappa \in \{ 1, 0, -1 \}$, i.e., $\eta = 1$ in (2.17), the choice of the penalty parameter σ_E differs between the flavors of the DG discretization. Following lemma 2.30, [Riv08, equation (2.8)] and [HSS02], the penalty parameters are chosen to be

$$\sigma_E = \begin{cases} \varepsilon n_0 (p+1)(p+2), & \text{if } \kappa = +1 \text{ (SIPG)}, \\ \varepsilon n_0 (p+1)(p+2)/2, & \text{if } \kappa = 0 \text{ (IIPG)}, \\ \varepsilon p^2, & \text{if } \kappa = -1 \text{ (NIPG)} \end{cases}$$

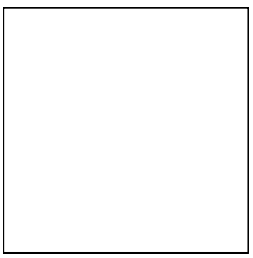
for all $E \in \mathcal{E}_h$, where n_0 is the number of edges of a cell, i.e., three for a triangular and four for a quadrilateral mesh. The polynomial degrees $p = 1, 2, 3, 4$ are investigated. The computations are performed on standard triangulations of the unit square; see figure 2.5.

The results for $\varepsilon = 1$ on the triangular grid for all polynomial degrees and all $\kappa \in \{ 1, 0, -1 \}$ are shown in figure 2.6. For the SIPG discretization, the L^2-error converges optimally with a rate of $p + 1$ with respect to h for all polynomial degrees. This is in accordance with theorem 2.12 and supports remark 2.36. Only for $p = 3$, the rates decrease slightly between the second to last and the last refinement step, possibly due to round-off errors. For the IIPG and NIPG discretization, it can be observed that the methods with even polynomial degrees converge with a rate of $p+1$ and with odd degrees with a rate of p with respect to h, which is better than proven but is in accordance with remark 2.13, hence supporting remark 2.36.

The convergence rates of the energy norm for all polynomial degrees and the SIPG, the IIPG, and the NIPG method are optimal and in accordance with the theory, cf. theorem 2.32.



(a) Triangular grid.



(b) Quadrilateral grid.

Figure 2.5.: Initial grids for example 2.37.

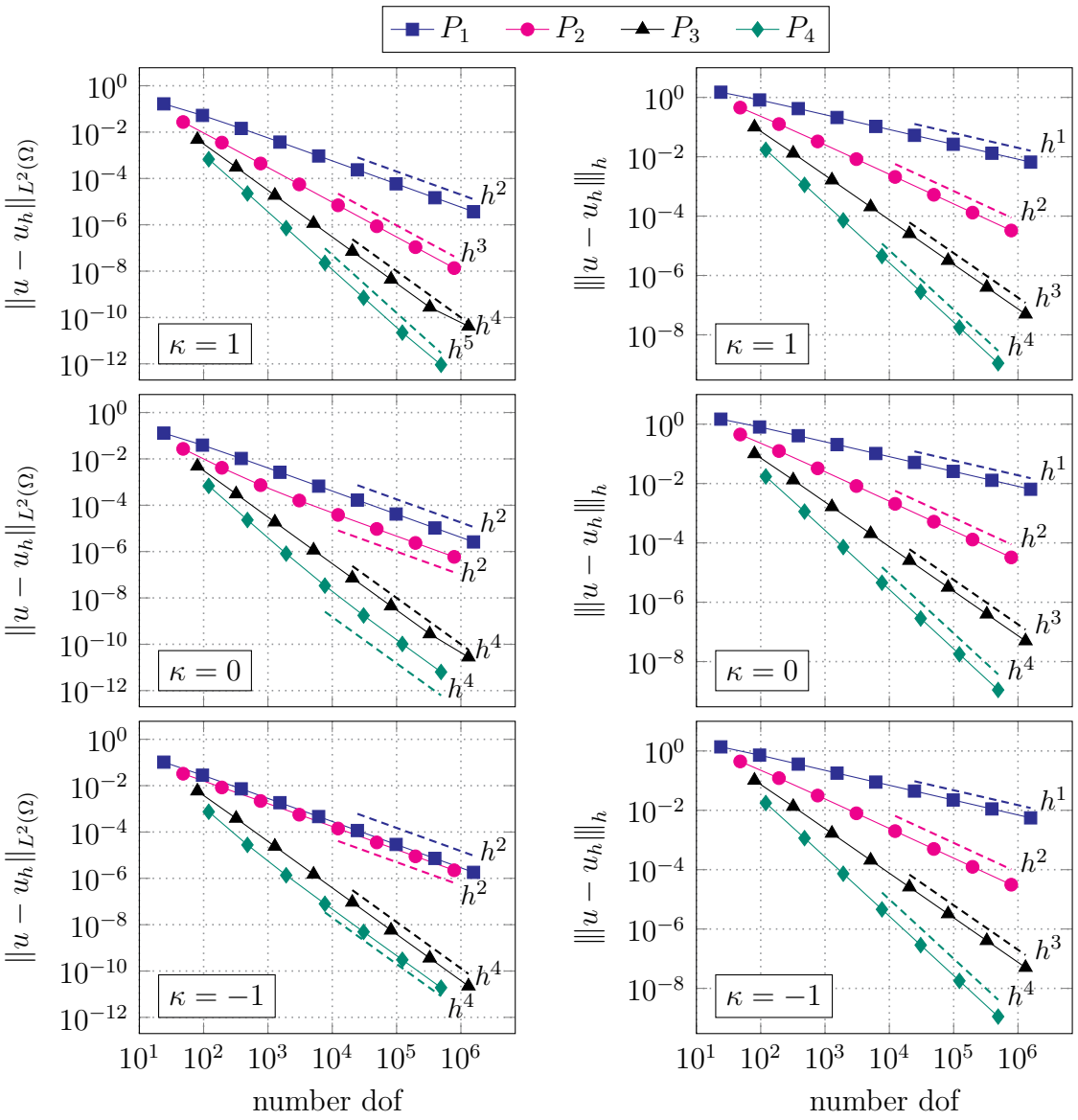


Figure 2.6.: Convergence rates for different discretizations for example 2.37 with $\varepsilon = 1$ on the triangular grid. Top row: $\kappa = +1$ (SIPG), middle row: $\kappa = 0$ (IIPG), bottom row: $\kappa = -1$ (NIPG). Left: $L^2(\Omega)$-norm, right: energy norm.

In figure 2.7, the convergence rates of all methods and polynomial degrees for $\varepsilon = 10^{-8}$ on the quadrilateral grid can be found. While for all methods and all polynomial degrees, the L^2-error converges with order $p+1$, the energy error shows a rate of $p+0.5$. The first rate is faster than proven in theorem 2.35, but in accordance with remark 2.26. The convergence rates in the energy norm are expected since the experiment is conducted in the convection-dominated regime, cf. remark 2.34.

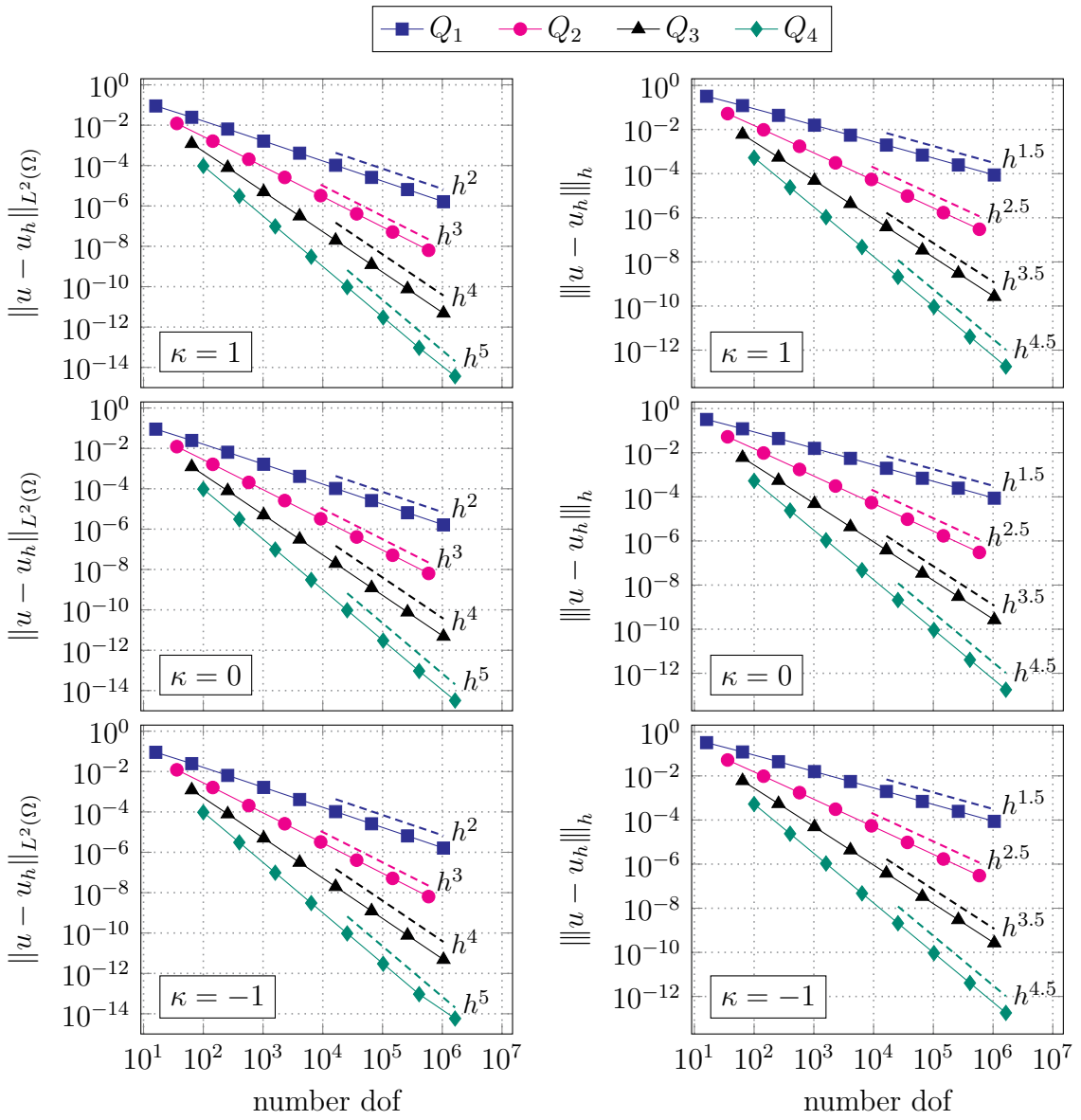


Figure 2.7.: Convergence rates for different discretizations for example 2.37 with $\varepsilon = 10^{-8}$ on the quadrilateral grid. Top row: $\kappa = +1$ (SIPG), middle row: $\kappa = 0$ (IIPG), bottom row: $\kappa = -1$ (NIPG). Left: $L^2(\Omega)$-norm, right: energy norm.

2.3.2. A three-dimensional problem

In this section, a three-dimensional problem is solved, which can be seen as an extension of the previous problem.

Example 2.38 (Sine Laplace problem in 3D)**.** Let $\Omega := (0,1)^3$ be the unit cube,

(a) Tetrahedral grid. (b) Hexahedral grid.

Figure 2.8.: Initial grids for example 2.38.

$\boldsymbol{b} \coloneqq (1,2,3)^T$, and $c \coloneqq 3$. Dirichlet boundary conditions are prescribed everywhere, i.e., $\Gamma_{\mathrm{D}} \coloneqq \partial\Omega$. The right-hand side f and the boundary conditions are chosen in accordance with the exact solution that is set to

$$u(x,y,z) \coloneqq \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

□

The computations take place on the grids depicted in figure 2.8. As before, the diffusion coefficient is set to be either $\varepsilon = 1$ or $\varepsilon = 10^{-8}$, and the SIPG, IIPG, and NIPG methods are tested. Furthermore, the upwind discretization is used, i.e., $\eta = 1$, and the penalty parameter is in accordance with lemma 2.30, [Riv08, equation (2.9)] and [HSS02] set to

$$\sigma_E = \begin{cases} 2\varepsilon n_0(p+1)(p+3)/3, & \text{if } \kappa = +1 \text{ (SIPG)}, \\ \varepsilon n_0(p+1)(p+3)/3, & \text{if } \kappa = 0 \text{ (IIPG)}, \\ \varepsilon p^2, & \text{if } \kappa = -1 \text{ (NIPG)} \end{cases}$$

for all $E \in \mathcal{E}_h$, where n_0 is four for the tetrahedral grid and six for the hexahedral grid. The polynomial degrees $p = 1,2,3$ are used on the tetrahedral grids, whereas $p = 1,2$ is investigated on the hexahedral grids.

Figure 2.9 shows the convergence history for $\varepsilon = 1$ on the tetrahedral grid for all flavors of the DG method and various polynomial degrees. It can be seen that the L^2-error converges with an order of $p+1$ for all polynomial degrees in the case $\kappa = 1$. For the IIPG and SIPG, only the even polynomial degrees converge with a rate of $p+1$ and the odd degrees with a rate of p with respect to h. This is better than expected from theorem 2.35 but is in accordance with remark 2.36. Furthermore, it can be noted that the tested iterative solvers in ParMooN cannot solve the resulting linear system of equations, which might be due to the condition number of the system. Solving this problem on finer meshes needs to be addressed in the future.

In the energy norm, the convergence rate is exactly p, which is in accordance with theorem 2.32.

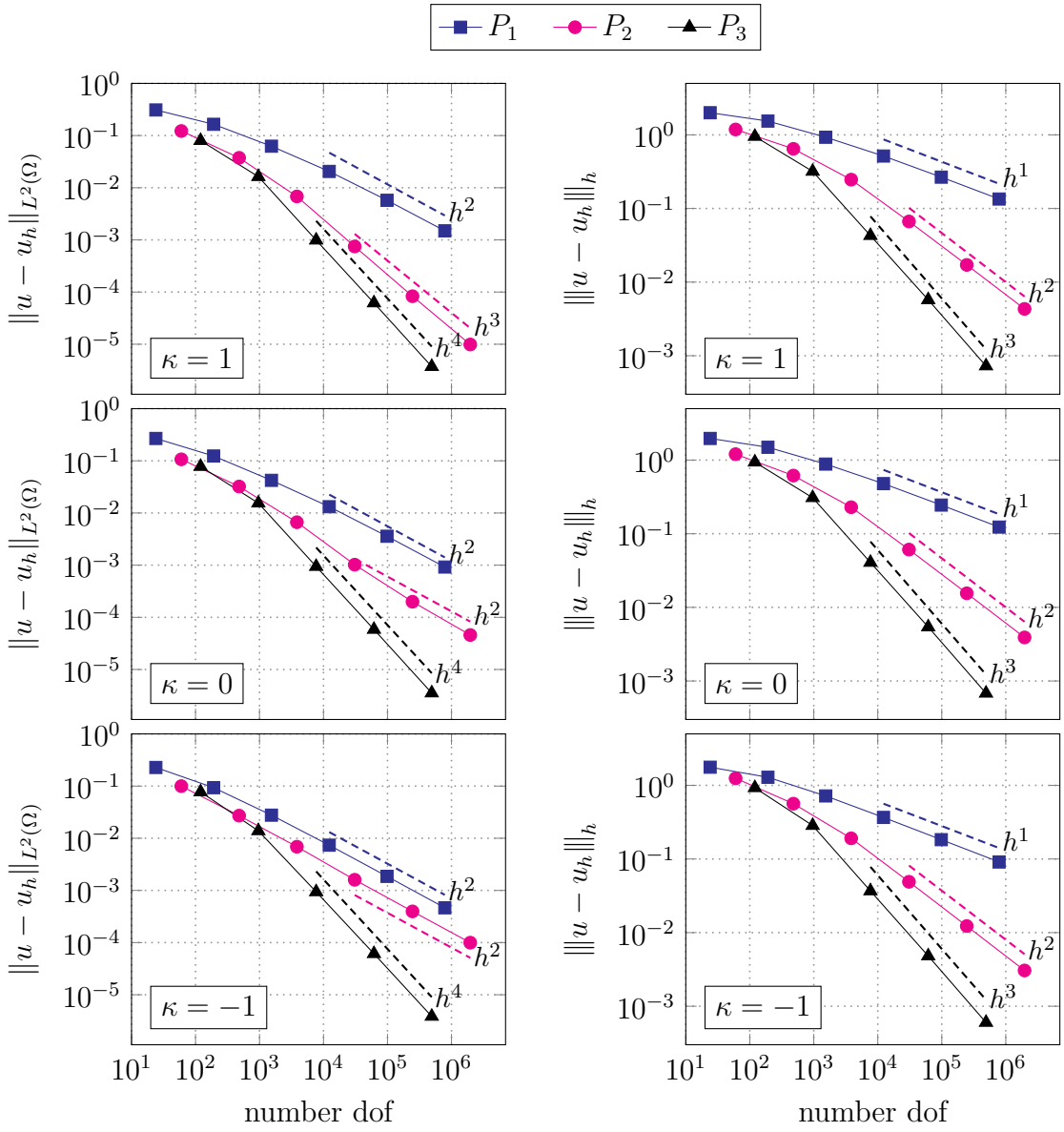


Figure 2.9.: Convergence rates for different discretizations for example 2.38 with $\varepsilon = 1$ on the tetrahedral grid. Top row: $\kappa = +1$ (SIPG), middle row: $\kappa = 0$ (IIPG), bottom row: $\kappa = -1$ (NIPG). Left: $L^2(\Omega)$-norm, right: energy norm.

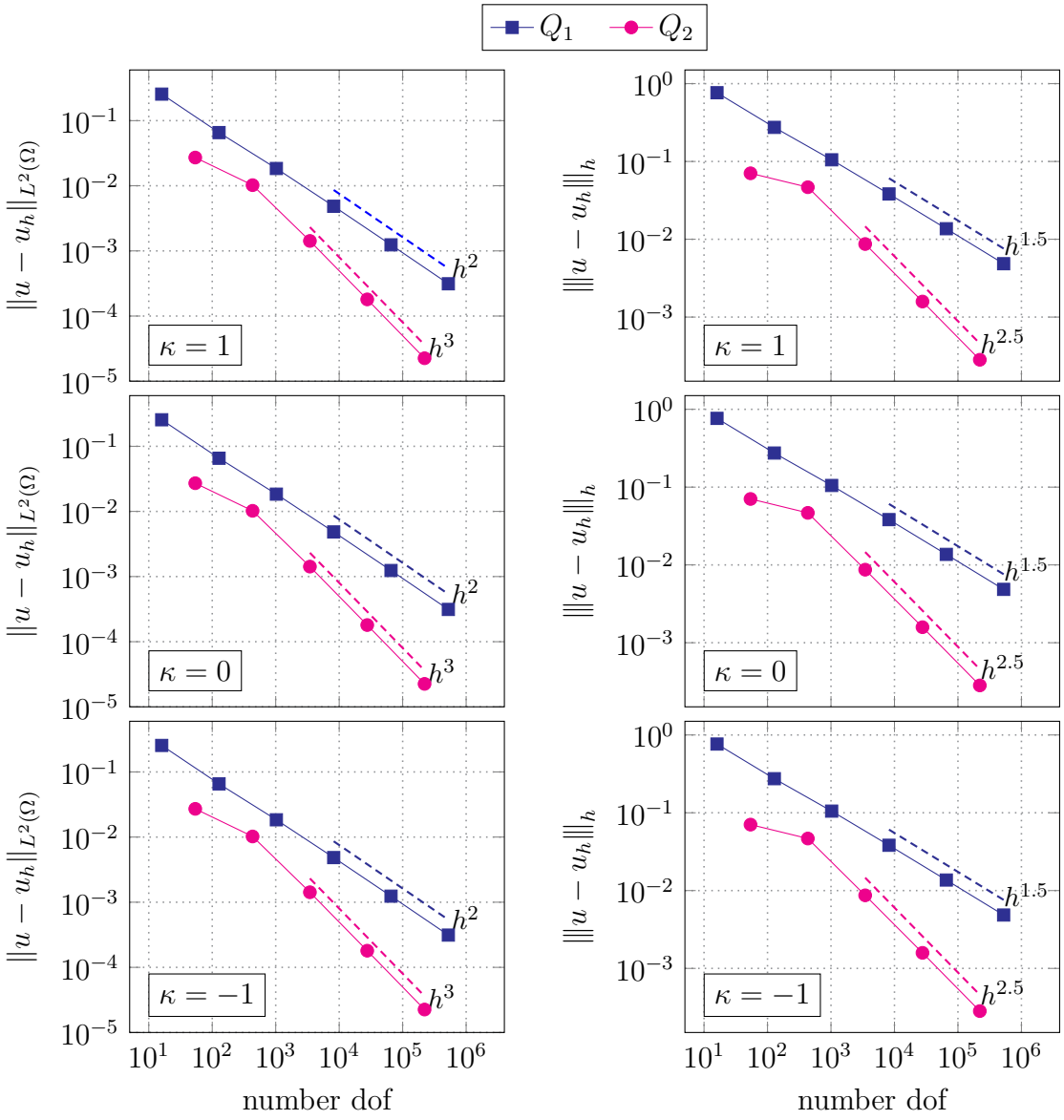


Figure 2.10.: Convergence rates for different discretizations for example 2.38 with $\varepsilon = 10^{-8}$ on the hexahedral grid. Top row: $\kappa = +1$ (SIPG), middle row: $\kappa = 0$ (IIPG), bottom row: $\kappa = -1$ (NIPG). Left: $L^2(\Omega)$-norm, right: energy norm.

The results for $\varepsilon = 10^{-8}$ computed on the hexahedral grid are depicted in figure 2.10. As before, in the L^2-norm the error for all κ and all p converge with optimal order of $p+1$, which is better than proven in theorem 2.35. However, this is in accordance with the two-dimensional results.

As before, the error in the error norm decreases with a rate of $p + 1/2$ for all κ as expected from theorem 2.32 since the experiment is conducted in the convection-dominated regime.

2.4. Summary

In this chapter, convection-diffusion-reaction problems and their discontinuous Galerkin discretization have been introduced.

In section 2.1, the model problems have been derived. The derivation is mainly based on the observation that the rate of change in a sub-domain equals the flux of the quantity across the boundary plus the total amount that is destroyed or created in the domain. Afterwards, the notion of weak solutions to these problems was introduced, and it was investigated under which conditions a unique weak solution exists. The proof is based on the famous Lax–Milgram lemma.

Section 2.2 has derived the discontinuous Galerkin discretization of convection-diffusion-reaction problems. To this end, the problem was split into pure diffusion and convection-reaction cases. It was also investigated under which conditions a unique discrete solution exists, and it was estimated how large the error between the exact and the discrete solution is in the L^2-norm and suitable energy norms in terms of the mesh width h.

Last but not least, numerical results have been presented in section 2.3 to verify the implementation and the theory. In both the two-dimensional and the three-dimensional test cases, the convergence rates have been observed to be at least what is expected from the theory. Therefore, this indicates that the implementation is correct and the choices of the penalty parameters are useful.

3. On reducing spurious oscillations using slope limiters

In the previous chapter, it is proven that the discrete solution of the discontinuous Galerkin method converges asymptotically towards the weak solution as the mesh width becomes smaller. However, only meshes up to a certain mesh width can be used in practice due to physical limitations such as time, money, and memory. On these computationally feasible meshes, both continuous Galerkin and discontinuous Galerkin methods lead to suitable approximations of the solution to convection-diffusion-reaction problems in the case of "moderate" diffusion constants, i.e., if the diffusion is about as strong as the convection. However, from a practical point of view, often convection-dominated problems with rather "small" diffusion constants relative to the convection are of interest for which it is computationally challenging to compute appropriate approximations [JKN18; JJ20; FJ21].

In the convection-dominated regime that is mathematically characterized by $\varepsilon \ll L \left\|\boldsymbol{b}\right\|_{[L^\infty(\Omega)]^d}$ with a characteristic length scale L, so-called *layers* in the solution may occur that have to be captured correctly by numerical methods [JKN18]. These layers are small subregions where the solution obtains a steep gradient. Unfortunately, in many cases, these layers cannot be resolved correctly on computationally feasible meshes since the size of the layers is much smaller than the mesh width. For many numerical methods, this leads to so-called *spurious oscillations*, i.e., physically unreasonable values such as negative concentrations, that pollute the solution globally [JKN18; FJ21]. Figure 3.1 shows exemplary such unphysical values for a convection-dominated problem that is given on the unit square with $\Gamma_\mathrm{D} = \partial\Omega$, $\varepsilon = 10^{-6}$, $\boldsymbol{b} = (2, 3)^T$, $c = 2$, and where the right-hand side of the problem and the boundary conditions are chosen such that

$$16x(1-x)y(1-y)\left(\frac{1}{2} + \frac{\arctan\left(2\varepsilon^{-1/2}\left(0.25^2 - (x-0.5)^2 - (y-0.5)^2\right)\right)}{\pi}\right)$$

is the exact solution; see also [JKS11, example 5.1] for another description of the problem.

To counteract these spurious oscillations, often stabilizing methods are used; see, e.g., [RST08, section III.3] for an overview and [Bar+18b] for a modern approach. In the context of continuous Galerkin methods, the streamline upwind Petrov–Galerkin (SUPG) method or spurious oscillations at layers diminishing (SOLD) methods are widely used to solve such problems; see, e.g., [RST08, section III.3.2.1] and [JK07;

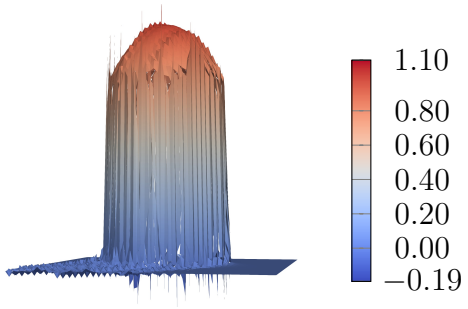


Figure 3.1.: Discontinuous Galerkin solution with finite elements of degree $p = 4$ to a convection dominated convection-diffusion-reaction problem. In equation (2.17), the parameter $\kappa = 1$, $\eta = 1$ and $\sigma_E = 9 \cdot 10^{-5}$ for all $E \in \mathcal{E}_h$ are used, and the computation is performed on the mesh that originates from refining the mesh given in figure 2.5a six times. Spurious oscillations pollute the solution that should lie between 1.00 and 0.00.

JK08]. These methods add artificial diffusion terms, and in doing so, they can reduce the spurious oscillations significantly compared to plain continuous Galerkin methods. On the other hand, a significant drawback of these methods is that they depend on parameters for which, at most asymptotic bounds are known and which influence the quality of the discrete solution [JK07; Aug+11]. Some attempts have been made to find optimal parameters a-posteriori [JKS11; JK13; KLS19; JKW23], but choosing optimal parameters a-priori is still an open problem.

It is also possible to use SUPG-like stabilization schemes based on discontinuous Galerkin methods as mentioned in [AM09, remark 3.1]. However, the DG method controls the streamline derivative without introducing another term; see also remark 2.24.

This work instead focuses on a post-processing technique to reduce spurious oscillations in the discrete solution. One advantage of these techniques is that they do not require solving a global linear or non-linear problem and therefore are computationally cheap. This is in contrast to many SOLD methods, which often require many iterations to solve a non-linear problem; see, e.g., [JK07]. Moreover, the proposed post-processing techniques allow for an arbitrary polynomial degree in the approximation. This comes in handy since, away from layers, more accurate approximations can be obtained with higher polynomial degrees, and hence these polynomials are favorable compared to a lowest-order discretization. However, a lower polynomial degree is usually a good choice in the vicinity of layers. The latter is because Sobolev norms of the analytic solution at layers scale with an inverse power of the diffusion coefficient. Lower polynomial degrees require Sobolev norms of lower degrees, and therefore, the influence of the inverse power on the error is smaller than for higher-order approximations [FJ21].

The basic idea behind the considered post-processing techniques is that in the vicinity of layers, the discrete solution gets replaced by a lower-order approximation

using thereby so-called *slope limiters*. Changing the polynomial degree of the discrete solution locally can be easily done with DG methods since the degrees of freedom in a single cell are independent of the degrees of freedom of the neighboring cells. This is in contrast to many other discretization techniques, e.g., continuous Galerkin methods, where neighboring cells share degrees of freedom, and hence such an easy replacement of the polynomial degree in a single cell is not possible.

To change the solution locally, a solution for a given polynomial degree must be computed first. Second, regions have to be detected where nonphysical behavior occurs. And third, the polynomial degree for the solution in these cells can be reduced. Therefore, the strategy for the post-processing techniques can be summarized in the following steps:

1. Mark the cells where spurious oscillations in the solution occur.

2. Approximate the solution on the marked cells by a lower degree polynomial using slope limiters.

In sections 3.1 to 3.3, some already known approaches are presented that follow the above-mentioned steps as well as further generalizations and modifications of these methods. They are mainly presented for the two-dimensional case since a three-dimensional extension is generally straightforward. However, the extension to three dimensions is shortly addressed at the end of the respective sections. Last but not least, it is notable that all methods share the property that they preserve the mass cell wise. This is in contrast to the naive idea of just clipping the solution locally where it exceeds the theoretical bounds. This is also explicitly mentioned for each method presented below.

This chapter is based on [FJ21; FJ22]. The methods presented in this work are the same, but different values for the penalty parameters σ_E are used for the numerical studies.

The structure is as follows: In sections 3.1 to 3.3, seven different slope limiting techniques are presented. They differ mainly in the way the cells are marked, and they replace the solution with a piecewise affine or constant function. In section 3.4, the methods are tested numerically based on two classical benchmark problems. This chapter closes with a summary in section 3.5.

3.1. Utilizing linear reconstructions across facets of mesh cells

3.1.1. Original method

This approach was introduced by Cockburn and Shu [CS98] and described again by Rivière [Riv08, section 4.3.2]. Originally, the method was proposed for triangular

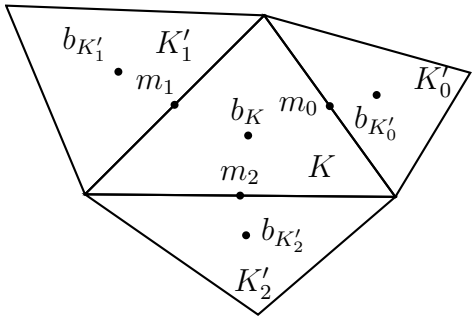


Figure 3.2.: A triangle K with its three edge midpoints m_i, $i = 0, 1, 2$, its three neighbors K_i', $i = 0, 1, 2$, and the barycenters b_{K^*} of the triangles, $K^* \in \{ K, K_0', K_1', K_2' \}$. Figure taken and adapted from [FJ21].

grids, but as described below, it also works with quadrilateral meshes. It is based on the *assumption* that spurious oscillations in the discrete solutions only arise if they are present in its linear part, i.e., the L^2-projection of u_h into the space of piecewise linear functions [CS98; Riv08]. The verification of this assumption is still an open question.

To describe the procedure, some notation has to be fixed. Let $K \in \mathcal{T}_h$ be an interior triangular mesh cell with neighbors K_0', K_1', K_2' where the numbering follows the local edge or face numbering. The midpoints of the facets of K are denoted by m_i, $i = 0, 1, 2$, and the barycenter of the triangles is labeled with $b_{K'}$ where $K' \in \{ K, K_0', K_1', K_2' \}$; see also figure 3.2 for a visualization.

After fixing the notation, the two previously mentioned steps of this post-processing approach can be described. To check whether the cell K has to be marked, the cell wise integral means $\overline{u}_{h,K^*} \coloneqq \int_{K^*} u_h \, \mathrm{d}\boldsymbol{x} / |K^*|$, $K^* \in \{ K, K_0', K_1', K_2' \}$, of u_h have to be computed. The algorithm then *marks* the cell K if, for at least one edge $E \in \mathcal{E}_h(K)$, it holds that $u_h|_K$ evaluated at the edge midpoint is not between the cell averages of the two adjacent cells, i.e., if for at least one $i = 0, 1, 2$,

$$u_h|_K(m_i) \notin \left[\min\left(\overline{u}_{h,K}, \overline{u}_{h,K_i'}\right) - \mathrm{tol}, \max\left(\overline{u}_{h,K}, \overline{u}_{h,K_i'}\right) + \mathrm{tol} \right],$$

where tol is set to 10^{-11}. The tolerance tol is introduced from a practical point of view to avoid marking cells due to numerical round-off errors. Note that in the original method, this tolerance is not proposed.

For each marked cell, three affine functions $L_j : K \to \mathbb{R}$, $j = 0, 1, 2$, are computed. The functions can be written as $L_j(x, y) \coloneqq a_0^j + a_1^j x + a_2^j y$, $j = 1, 2, 3$, and are determined by

$$L_j(b_K) = \overline{u}_{h,K}, \qquad L_j(b_{K_{j+1}'}) = \overline{u}_{h,K_{j+1}'}, \qquad L_j(b_{K_{j+2}'}) = \overline{u}_{h,K_{j+2}'},$$

where $K_3' \coloneqq K_0'$ and $K_4' \coloneqq K_1'$. These affine functions can be ordered afterwards by

decreasing Euclidean norm

$$\sqrt{\left(a_1^j\right)^2 + \left(a_2^j\right)^2}$$

of their affine coefficients, i.e., loosely speaking by the steepness of their gradients.

Concerning this ordering, it can be successively checked whether the affine functions fulfill the marking criterion, i.e., if $L_j(m_i)$ is between $\overline{u}_{h,K}$ and $\overline{u}_{h,K_i'}$ plus the tolerance for all $i = 0, 1, 2$. If this is the case for the current affine function, then u_h is *approximated* locally by L_j, i.e., locally u_h gets replaced by L_j, and the other affine functions are discarded. If none of the affine functions satisfy this criterion, u_h gets replaced by its integral mean $\overline{u}_{h,K}$.

Remark 3.1 (Local preservation of the mass). If the solution is replaced by its integral mean, the mass is preserved by definition. In the case that the solution is replaced by some L_j, $j = 1, 2, 3$, it holds by the midpoint quadrature rule

$$\frac{1}{|K|} \int_K L_j(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = L_j(b_K)$$

since L_j is an affine function. Moreover, since by construction $L_j(b_K) = \overline{u}_{h,K}$, also in this case, the integral mean is preserved. □

Remark 3.2. It can be noted that even though this method was defined on triangles, it can also be applied to quadrilateral meshes. Both during the marking and the reconstruction, just a fourth edge has to be considered, e.g., during the reconstruction, four candidates are computed for each pair of consecutive edges, and then the best one is chosen. □

Remark 3.3 (Extension to three dimensions). In three dimensions, the marking criterion works as in two dimensions. Four candidates are constructed during the reconstruction, i.e., one for each face. They are given by $L_j(x, y, z) \coloneqq a_0^j + a_1^j x + a_2^j y + a_3^j z$, $j = 1, 2, 3, 4$, and their coefficients are computed using

$$L_j(b_K) = \overline{u}_{h,K}, \quad L_j(b_{K_{j+1}'}) = \overline{u}_{h,K_{j+1}'}, \quad L_j(b_{K_{j+2}'}) = \overline{u}_{h,K_{j+2}'}, \quad L_j(b_{K_{j+3}'}) = \overline{u}_{h,K_{j+3}'},$$

where $K_4' \coloneqq K_0'$ and $K_5' \coloneqq K_1'$ and $K_6' \coloneqq K_2'$. As in the two-dimensional case, the functions are sorted in descending order according to the Euclidean norm

$$\sqrt{\left(a_1^j\right)^2 + \left(a_2^j\right)^2 + \left(a_3^j\right)^2}$$

of their affine coefficients and it is proceeded as before. □

3.1.2. Modified method

The algorithm presented above has three drawbacks:

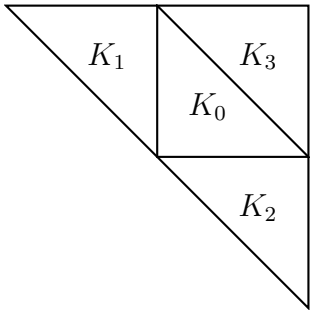


Figure 3.3.: Triangulation of the domain $\operatorname{conv}\{(-1,1),(1,-1),(1,1)\}$. A red-refinement of the domain generates the triangles.

1. It may reduce overshoots while simultaneously introducing undershoots or vice versa.

   Example 3.4. Let $\Omega \coloneqq \operatorname{conv}\{(-1,1),(1,-1),(1,1)\}$ be red-refined into four triangles of equal size as depicted in figure 3.3. If on this domain the function u_h is defined as

   $$u_h|_{K_0} \coloneqq 1 - x - y, \qquad u_h|_{K_1} \coloneqq u_h|_{K_2} \coloneqq 0, \qquad u_h|_{K_3} \coloneqq 1,$$

   then the previously described algorithm replaces $u_h|_{K_0}$ by $x + y - 1/3$. It can be seen that the original function has values in $[0,1]$, whereas the minimum of the modified function is $-1/3$. Hence, the reconstruction decreased the global minimum in this example and introduced an undershoot.

   This can happen not only for simple examples but also in practice, as it can be seen in the numerical experiments performed in section 3.4; see, e.g., figure 3.7. □

2. Evaluating the function at the edge midpoint does not help if the oscillations happen near a vertex.

   Example 3.5. Let the domain be given as in example 3.4 and let u_h be defined as

   $$\begin{aligned} u_h|_{K_0} &\coloneqq -7 + 20x + 20y - 12x^2 - 12xy - 12y^2, \\ u_h|_{K_1} &\coloneqq u_h|_{K_2} \coloneqq 0, \qquad u_h|_{K_3} \coloneqq 1. \end{aligned}$$

   On the one hand, calculations show $\overline{u}_{h,K_0} = 1/3$, $u_h(0,1/2) = u_h(1/2,0) = 0$ and $u_h(1/2,1/2) = 1$. On the other hand, on K_0, the function takes values in $[-7,4/3]$ and should be limited. □

3. The algorithm relies on neighbors and hence is a-priori only defined for interior cells.

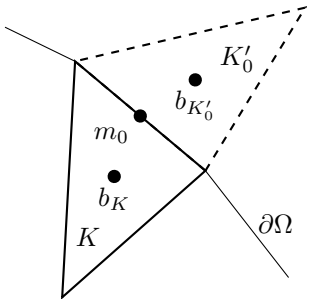


Figure 3.4.: Virtual triangle reflected at the boundary edge which in this example is locally the first edge. Figure taken and adapted from [FJ21].

To counteract the problem described in item 1, it might be better always to replace the function by its integral mean. With this strategy, the extrema are strictly clipped, but at the price that the order of approximation is more reduced. However, since the idea is to reduce the order in the vicinity of layers anyway, it is a fair trade-off.

Regarding item 2, evaluating the function at the edge midpoint can be seen as an integration rule for polynomials of degree one. In this view and to detect oscillations at other points along the boundary, it could be better to check whether the integral mean of u_h along and edge lies between the cell averages of the adjacent cells plus the tolerance. In light of example 3.5, using the integral mean along the edges, the function u_h on K_0 would have been limited, since it holds $\int_{E_1} u_h \,\mathrm{d}s = \int_{E_2} u_h \,\mathrm{d}s = -1$.

To apply the algorithm also on boundary cells, and thus counteract item 3, the following approach can be used: For every boundary edge E_i a virtual triangle K_i' is constructed by reflecting the original triangle along edge E_i, cf. figure 3.4. Then on this reflected triangle, the virtual solution $u_h|_{K_i'}$ is defined as the continuation of $u_h|_K$ from K to K_i' which is well defined since $u_h|_K$ is a polynomial. Having this virtual cell(s) at hand, the algorithm can be applied as before, i.e., for marking the cell.

Remark 3.6 (Local preservation of the mass). Since, in the modified case, the solution is replaced by its integral mean, the mass is preserved. □

Remark 3.7. This modification can also be applied on quadrilateral grids in the same manner as described in remark 3.2. □

Remark 3.8 (Extension to three dimensions). The three-dimensional case works as the two-dimensional case. □

3.2. Utilizing weighted mean derivatives

3.2.1. Original method

Initially, this approach was introduced by Cockburn and Shu [CS98] for axis-parallel rectangular grids and again revisited by Rivière [Riv08, section 4.3.2]. This method

can be generalized to quadrilaterals being the image of a reference cell under affine transformation, which is also presented here.

The idea of the generalization stems from mapped finite elements, i.e., the basis functions and nodal functionals of a finite element are defined on a reference cell $\widehat{K}$, and the functions and functionals on the physical cell K are given by a reference transformation $F_K : \widehat{K} \to K$; see figure 3.5. Hence, it seems natural to base the slope-limiting approach on the reference cell and the transformation. Here, $\widehat{K} \coloneqq [-1, 1]^2$ is used as reference cell. In what follows, the reference transformation is furthermore assumed to be affine.

On K, the functions

$$\psi(x, y) \coloneqq \frac{F_{K,1}^{-1}(x, y)}{2}, \qquad \xi(x, y) \coloneqq \frac{F_{K,2}^{-1}(x, y)}{2} \tag{3.1}$$

are defined, where $F_{K,1}^{-1}$ and $F_{K,2}^{-1}$ are the first and second component of the inverse of the reference transformation F_K, respectively. Note that for axis-parallel rectangles, both functions differ by a factor of $1/2$ from their respective definitions in [CS98] and equal their definitions in [Riv08, section 4.3.2].

Moreover, the functionals

$$\begin{aligned} N_{K,0}(v_h) &\coloneqq \frac{1}{|K|} \int_K v_h \, \mathrm{d}\boldsymbol{x}, \\ N_{K,1}(v_h) &\coloneqq C_1 \int_{\widehat{K}} v_h(F_K(\widehat{x}, \widehat{y}))\widehat{x} \, \mathrm{d}\widehat{\boldsymbol{x}}, \\ N_{K,2}(v_h) &\coloneqq C_2 \int_{\widehat{K}} v_h(F_K(\widehat{x}, \widehat{y}))\widehat{y} \, \mathrm{d}\widehat{\boldsymbol{x}} \end{aligned}$$

are defined, where the normalizing constants C_1 and C_2 are determined by the condition $N_{\widehat{K},1}(\psi) = N_{\widehat{K},2}(\xi) = 1$.

Locally on K, the function u_h can be expanded by

$$u_h(x, y)|_K = a_{0,K} + a_{1,K}\psi(x, y) + a_{2,K}\xi(x, y) + \text{higher-order terms},$$

where the coefficients are given by

$$a_{0,K} = N_{K,0}(u_h), \qquad a_{1,K} = N_{K,1}(u_h), \qquad a_{2,K} = N_{K,2}(u_h).$$

By the definition of $N_{K,0}$, it is clear that $a_{0,K}$ is the integral mean over cell K. On axis-parallel rectangular grids, the coefficients $a_{1,K}$ and $a_{2,K}$ can be seen as a collection of the linear parts of u_h in x- and y-direction, respectively. This corresponds to the linear parts in $F_K(1, 0)$- and $F_K(0, 1)$-direction for affine rectangular grids.

A different point of view can be achieved by clipping the higher-order terms. Denoting the resulting affine function by $\tilde{u}_h$ and the matrix of the affine transformation by B_K, it holds

$$\nabla \tilde{u}_h = \frac{B_K^{-1}}{2} \begin{pmatrix} a_{1,K} \\ a_{2,K} \end{pmatrix},$$

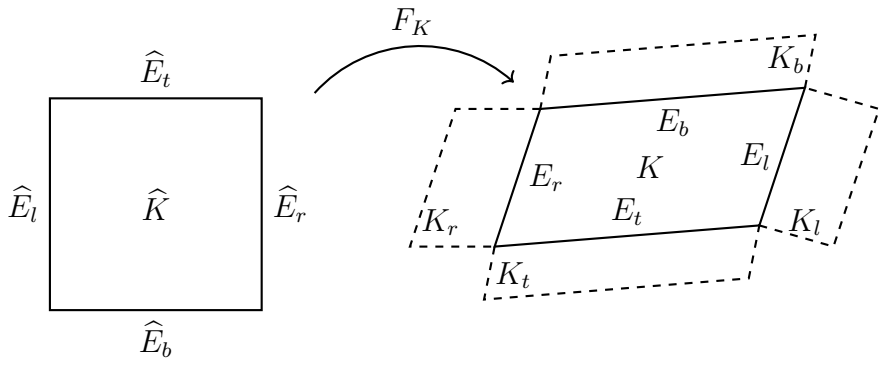


Figure 3.5.: Reference domain $\widehat{K} = [-1, 1]^2$ for section 3.2 and reference transformation F_K to physical cell K. The edges in the physical cell are given as images of the corresponding reference edges under F_K.

and hence the coefficients can be thought of providing information on a weighted mean derivative of u_h in K [FJ21].

It is furthermore noteworthy that neither basis functions of the higher-order terms have to be defined nor coefficients have to be computed.

With these coefficients at hand, it can be decided whether slope limiting must be applied on the cell K or not. To this end, it is assumed that $\widehat{E}_l \coloneqq \overline{(-1,1)(-1,-1)}$, $\widehat{E}_b \coloneqq \overline{(-1,-1)(1,-1)}$, $\widehat{E}_r \coloneqq \overline{(1,-1)(1,1)}$, and $\widehat{E}_t \coloneqq \overline{(1,1)(-1,1)}$ are the left, bottom, right, and top edge of the reference cell, respectively. Let K_l be the neighbor of K across $E_l \coloneqq F_K(\widehat{E}_l)$, and use analog notations for K_b, K_r and K_t; see also figure 3.5.

Using two user-dependent constants $M_{\text{lim}} \geq 0$ and $\gamma \geq 0$, the adjusted coefficients

$$\overline{a}_{1,K} := \begin{cases} a_{1,K}, & \text{if } |a_{1,K}| \leq M_{\text{lim}}, \\ \text{minmod}\left(a_{1,K}, \gamma(a_{0,K_r} - a_{0,K}), \gamma(a_{0,K} - a_{0,K_l})\right), & \text{else,} \end{cases}$$

$$\overline{a}_{2,K} := \begin{cases} a_{2,K}, & \text{if } |a_{2,K}| \leq M_{\text{lim}}, \\ \text{minmod}\left(a_{2,K}, \gamma(a_{0,K_t} - a_{0,K}), \gamma(a_{0,K} - a_{0,K_b})\right), & \text{else,} \end{cases}$$

have to be computed. The minmod function is thereby given by

$$\text{minmod}(a_0, a_1, a_2) := \begin{cases} s \, \min_{i=0,1,2} |a_i|, & \text{if } s := \text{sign}(a_0) = \text{sign}(a_1) = \text{sign}(a_2), \\ 0, & \text{else.} \end{cases}$$

In the case of boundary cells, the minmod function is applied without missing entries.

The cell K is then *marked* if $a_{1,K} \neq \overline{a}_{1,K}$ or $a_{2,K} \neq \overline{a}_{2,K}$.

For all marked cells, the higher-order terms are clipped, and the adjusted coefficients are used, i.e., u_h is *approximated* by

$$u_h(x, y)|_K = a_{0,K} + \overline{a}_{1,K}\psi(x, y) + \overline{a}_{2,K}\xi(x, y).$$

Remark 3.9 (Local preservation of the mass). Following the above-mentioned algorithm the solution gets replaced by some $a_{0,K} + \overline{a}_{1,K}\psi(x,y) + \overline{a}_{2,K}\xi(x,y)$, with some coefficients $a_{0,K}$, $\overline{a}_{1,K}$ and $\overline{a}_{2,K}$, and ψ and ξ defined in equation (3.1). Then by construction it holds

$$\frac{1}{|K|}\int_K a_{0,K} + \overline{a}_{1,K}\psi(x,y) + \overline{a}_{2,K}\xi(x,y)\,\mathrm{d}\boldsymbol{x} = a_{0,K},$$

since $\int_K \psi(x,y)\,\mathrm{d}\boldsymbol{x}/|K| = N_{K,0}(\psi) = 0$ and $\int_K \xi(x,y)\,\mathrm{d}\boldsymbol{x}/|K| = N_{K,0}(\xi) = 0$. And by definition, it is $a_{0,K} = \overline{u}_{h,K}$, which is why the mass is preserved. □

Remark 3.10 (Extension to three dimensions). In three dimensions, the function

$$\eta(x,y,z) \coloneqq \frac{F_{K,3}^{-1}(x,y,z)}{2}$$

has to be introduced that is based on the third component of the inverse of the reference transform. Furthermore, the corresponding functional

$$N_{K,3}(v_h) \coloneqq C_3 \int_{\widehat{K}} v_h(F_K(\widehat{x},\widehat{y},\widehat{z}))\widehat{z}\,\mathrm{d}\widehat{\boldsymbol{x}}$$

can be defined where C_3 is determined by $N_{K,3}(\eta) = 1$. The expansion of the solution gets another linear term $a_{3,K}\eta(x,y,z)$ added with $a_{3,K} \coloneqq N_{K,3}(u_h)$. In addition, also a third adjusted coefficient

$$\overline{a}_{3,K} \coloneqq \begin{cases} a_{3,K}, & \text{if } |a_{3,K}| \leq M_{\mathrm{lim}}, \\ \mathrm{minmod}\left(a_{3,K}, \gamma(a_{0,K_f} - a_{0,K}), \gamma(a_{0,K} - a_{0,K_{ba}})\right), & \text{else,} \end{cases}$$

is computed for the neighbors in the third direction K_f, K_{ba}. Afterwards, the algorithm can be followed as in the two-dimensional case. □

3.2.2. Modified method

As before, replacing the function on marked cells by the integral mean, i.e., by $a_{0,K}$ could be better. If the function is replaced by its integral mean, the slope is reduced as much as possible, and therefore chances are higher that the spurious oscillations are leveled out.

Remark 3.11 (Local preservation of the mass). The mass is preserved since the solution is replaced by its integral mean. □

Remark 3.12 (Extension to three dimensions). As in the two-dimensional case, the solution gets replaced by its cell integral mean. □

3.3. Utilizing evaluations of jumps across facets

3.3.1. Original method

This method can be traced back to Dolejší, Feistauer, and Schwab [DFS02], who introduced the method and further analyzed it in [DFS03]. It uses information about the jumps of u_h along facets to detect spurious oscillations and replaces the function locally with a constant approximation if needed.

The indicator to mark cells is based on observations from numerical studies for lowest-order DG methods, i.e., with polynomial degree $p = 1$. In regions where the solution is smooth, it holds

$$\sum_{E \in \mathcal{E}_h(K) \cap \mathcal{E}_h^{\mathrm{I}}} \int_E \frac{[\![u_h]\!]^2}{h_K^5} \,\mathrm{d}s \approx \sum_{E \in \mathcal{E}_h(K) \cap \mathcal{E}_h^{\mathrm{I}}} \int_E \frac{\mathcal{O}(h_K^2)^2}{h_K^5} \,\mathrm{d}s \approx \mathcal{O}(1),$$

whereas in the vicinity of layers

$$\sum_{E \in \mathcal{E}_h(K) \cap \mathcal{E}_h^{\mathrm{I}}} \int_E \frac{[\![u_h]\!]^2}{h_K} \,\mathrm{d}s \approx \sum_{E \in \mathcal{E}_h(K) \cap \mathcal{E}_h^{\mathrm{I}}} \int_E \frac{\mathcal{O}(1)^2}{h_K} \,\mathrm{d}s \approx \mathcal{O}(1)$$

holds true.

Dolejší, Feistauer, and Schwab [DFS03] hence deduce that, for any $\alpha \in (1, 5)$, the quantity

$$\sum_{E \in \mathcal{E}_h(K) \cap \mathcal{E}_h^{\mathrm{I}}} \int_E \frac{[\![u_h]\!]^2}{h_K^\alpha} \,\mathrm{d}s \tag{3.2}$$

may serve as a smoothness indicator, and in particular, they proposed to use $\alpha = 5/2$. Precisely, they *mark* all the cells where it holds

$$\sum_{E \in \mathcal{E}_h(K) \cap \mathcal{E}_h^{\mathrm{I}}} \int_E \frac{[\![u_h]\!]^2}{h_K |K|^{3/4}} \,\mathrm{d}s \geq 1, \tag{3.3}$$

which corresponds to $\alpha = 5/2$ since on two-dimensional shape-regular grids it is $|K| \approx h_K^2$.

After the cells are marked, the solution is *approximated* on that cells by replacing it by its integral mean $\overline{u}_{h,K}$.

It can be noted that this method uses the asymptotic behavior of the solution, and therefore, care has to be taken when the denominator is large, i.e., if $h_K |K|^{3/4} > 1$ or even $h_K |K|^{3/4} \gg 1$. Hence, this method should only be used on small cells since for large cells, equation (3.3) might not be satisfied even if the jump is large.

Remark 3.13. Note that this method works independently of the geometry, i.e., for triangular as well as for quadrilateral grids. □

Remark 3.14 (Local preservation of the mass). The mass is preserved since the solution is replaced by its integral mean. □

Remark 3.15 (Extension to three dimensions). In three dimensions α might be chosen to lie between 1 and 6. Besides that, equation (3.2) can be used without changes. □

3.3.2. Modified method

The authors' choice of $\alpha = 5/2$ is based on their experience and experiments, but in general other choices are valid as well. For instance, increasing α would lead to more cells being marked, and decreasing α would result in marking fewer cells. Instead of choosing a different value, a slightly different approach is conducted here, which is inspired by equation (3.2).

If the discrete solution has a non-smooth behavior across just one facet aligned with a layer, the original indicator equation (3.3) might not detect the cell correctly. Examining individual facets might be better in this case, and if the facets of a cell are of much different size. In the latter, small facets might have too little impact on the original indicator.

Having this in mind and inspired by equation (3.2), the ansatz

$$\int_E [\![u_h]\!]^2 \, \mathrm{d}s = C_0 L u_0^2 \left(\frac{h_E}{L}\right)^{\alpha_E} \quad \Longrightarrow \quad \alpha_E = \frac{\ln\left(\frac{1}{C_0 L u_0^2} \int_E [\![u_h]\!]^2 \, \mathrm{d}s\right)}{\ln\left(\frac{h_E}{L}\right)}$$

as an indicator is chosen, where $C_0 \in \mathbb{R}$, $C_0 > 0$, is a constant, $L \in \mathbb{R}$, $L > h_E$, is the characteristic length scale of the problem and $u_0 \in \mathbb{R}$, $u_0 \neq 0$, is a characteristic scale of the solution. The quantity α_E is computed for each facet, and for a cell K, $\alpha_K \coloneqq \min\{\alpha_E : E \in \mathcal{E}_h(K)\}$ is deduced.

The cell K is then *marked* if $\alpha_K \leq \alpha_{\mathrm{ref}}$ with some reference $\alpha_{\mathrm{ref}} \in \mathbb{R}$. As before, the solution on the marked cells is *approximated* by its integral mean.

Possible choices for L are 3/2 times the longest facet in the initial triangulation or an adaptively chosen value of 3/2 times the longest facet of the current mesh. The parameter u_0 has to be adapted to the problem and might be chosen as the largest expected value of the solution if this is known analytically. Lastly, the constant C_0 can be used to fine-tune the limiter and has to be adapted to the problem.

In contrast to the original approach, the modified method is well defined in the case that $h_E \geq 1$ if L is chosen appropriately. Furthermore, the method is scaling invariant, and α_E is dimensionless as expected from an exponent.

Remark 3.16 (Local preservation of the mass). The mass is preserved since the solution is replaced by its integral mean. □

Remark 3.17 (Extension to three dimensions). In three dimensions the ansatz

$$\int_E [\![u_h]\!]^2 \, \mathrm{d}s = C_0 L^2 u_0^2 \left(\frac{h_E}{L}\right)^{\alpha_E} \quad \Longrightarrow \quad \alpha_E = \frac{\ln\left(\frac{1}{C_0 L^2 u_0^2} \int_E [\![u_h]\!]^2 \, \mathrm{d}s\right)}{\ln\left(\frac{h_E}{L}\right)}$$

might be chosen. □

3.3.3. Derived method

Marking cells in both previously mentioned methods is essentially based on the magnitude of the square of the integral of the jump of the discrete solution along an edge E, which is nothing else than $\| [\![u_h]\!] \|^2_{L^2(E)}$. Hence, with $r \in [1, \infty]$, another idea is to assess the $L^r(E)$-norm of the jump directly. For each interior facet

$$\beta_E \coloneqq \frac{\| [\![u_h]\!] \|_{L^r(E)}}{|E|^{\frac{1}{r}}}$$

can be computed, and a cell is *marked* if $\max_{E \in \mathcal{E}_h(K)} \beta_E \geq \beta_{\mathrm{ref}}$ for some user-chosen $\beta_{\mathrm{ref}} \in \mathbb{R}$. The solution is then *approximated* again by its integral mean.

This approach also has the advantage of being scaling-invariant, being easily computed, and working on simplicial, quadrilateral, and tetrahedral grids, respectively. A drawback might be that it is a-priori not clear how to choose β_{ref}, which can be adapted to the problem. The arithmetic mean value of all β_E may be a possible choice. In [FJ22], different values for β_{ref} are investigated, namely increasing and decreasing the value by twice the standard deviation of all β_E. However, the difference in the results is small, which indicates that the arithmetic mean itself is already a proper choice.

Remark 3.18 (Local preservation of the mass). The mass is preserved since the solution is replaced by its integral mean. □

Remark 3.19 (Extension to three dimensions). In three dimensions, the same can be done as in two dimensions. □

3.4. Numerical studies

In this section, the previously described limiters are tested for different standard benchmark problems to investigate how much they can reduce spurious oscillations compared to the standard Galerkin method (2.17). The experiments are conducted with the software package ParMooN [Wil+17]. In the following experiments, the SIPG discretization of the diffusive term and the upwind discretization of the convective term in (2.17) are deployed, i.e., $\kappa = 1$ and $\eta = 1$ are used. Inspired by [Riv08, equation (2.8), p. 39], the last parameter is chosen to be $\sigma = \varepsilon n_0 (p+1)(p+2)$, where

n_0 is the number of facets of the cell. Several polynomial degrees are used namely $p = 1, 2, 3, 4$. Note that in contrast to [FJ21; FJ22] a different penalty parameter is chosen, which is why the results are slightly different.

On the one hand, the post-processing techniques can be compared to the unmodified DG method to evaluate how much the oscillations decrease. On the other hand, it is also helpful to compare the algorithms with an optimal one, i.e., a method that reduces the over- and undershoots and the measures given below as much as possible while it preserves at the same time the mass cell wise. This method is obtained by locally replacing the solution with its integral mean in every cell.

In the numerical studied the following methods are tested:

- Galerkin: DG method (2.17) without post-processing,
- Optimal: DG method (2.17) where the solution is cell wise replaced by its integral mean in every cell,
- LinTriaReco: Post-processing on triangular grids using locally linear approximations based on reconstructions along facets; see section 3.1.1,
- ConstTriaReco: Post-processing on triangular grids using locally constant approximations based on reconstructions along facets; see section 3.1.2,
- LinQuadReco: Linear reconstruction on quadrilateral grids following LinTriaReco; see remark 3.2,
- ConstQuadReco: Constant reconstruction on quadrilateral grids following ConstTriaReco; see remark 3.7,
- LinQuadDeriv: Post-processing on quadrilateral grids using locally linear approximations based on a mean derivative, $M_{\text{lim}} = 0$, $\gamma = 1$; see section 3.2.1,
- ConstQuadDeriv: Post-processing on quadrilateral grids using locally constant approximations based on a mean derivative, $M_{\text{lim}} = 0$, $\gamma = 1$; see section 3.2.2,
- ConstJump: Post-processing on any type of grid using locally constant approximations based on evaluating the jump across facets, $\alpha = 2.5$; see section 3.3.1,
- ConstJumpMod: Modification of ConstJump, $\alpha_{\text{ref}} = 4$, $C_0 = 1$; see section 3.3.2,
- ConstJumpNorm: Post-processing on any type of grid using locally constant approximations based on a norm of the jump along facets, $r = \infty$, β_{ref} chosen to be the arithmetic mean of all β_E; see section 3.3.3. To prevent limiting smooth solutions, if $\beta_{\text{ref}} \leq 10^{-13}$, then ConstJumpNorm is set to leave the solution unchanged.

The choice of the parameters in LinQuadDeriv, ConstQuadDeriv, and ConstJump are guided by the literature; see [Riv08; DFS02]. Below, the results $r = \infty$ in ConstJumpNorm are depicted. The $r = 1$ and $r = 2$ results show only minor differences compared to $r = \infty$. Therefore, for the sake of brevity, they are not shown. To approximate the $L^\infty(E)$-norm, the function is evaluated at the vertices of the edge E and at the quadrature points of the Gauss–Legendre quadrature rule of degree $2r$ on this edge.

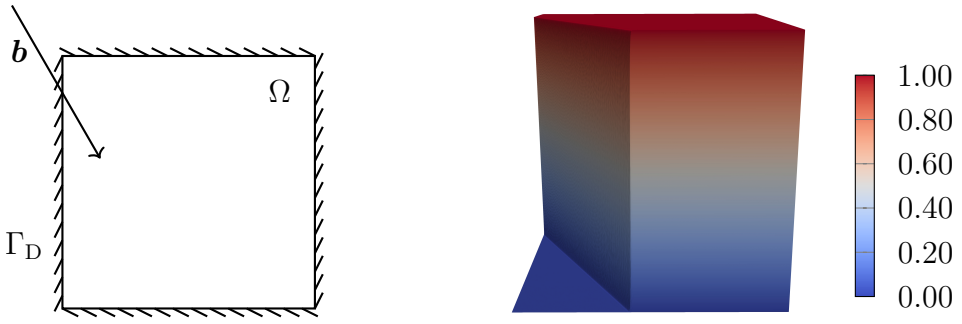
Two measures are defined to assess the limiters' capability to reduce spurious oscillations. The first quantity

$$\operatorname{osc}_{\max}(u_h) \coloneqq \max_{(x,y)\in\overline{\Omega}} u_h(x,y) - u_{\max} + u_{\min} - \min_{(x,y)\in\Omega} u_h(x,y) \tag{3.4}$$

compares the global maximum and minimum of the discrete solution u_h with the maximum $u_{\max}$ and minimum $u_{\min}$ of the continuous solution to investigate maximal oscillations. This measure can be negative since the boundary conditions are only imposed weekly. In order to take more than two values into account,

$$\begin{aligned}\operatorname{osc}_{\text{mean}}(u_h) \coloneqq \frac{1}{|\mathcal{T}_h|} \sum_{K\in\mathcal{T}_h} \Bigg[&\max\left\{0, \max_{(x,y)\in K} u_h(x,y) - u_{\max}\right\} \\ &+ \max\left\{0, u_{\min} - \min_{(x,y)\in K} u_h(x,y)\right\}\Bigg]\end{aligned} \tag{3.5}$$

is defined to assess the mean oscillations. It has the advantage of distinguishing between solutions with a few but large oscillations and one with many oscillations that are close to the largest oscillation. To compute $\operatorname{osc}_{\max}$ and $\operatorname{osc}_{\text{mean}}$, the discrete solution u_h is evaluated at certain points, which are the points of the local nodal functionals defining the continuous P_p/Q_p elements of the same order.



(a) Domain. (b) Sketch of the solution.

Figure 3.6.: Domain for and sketch of solution to example 3.21 for $\varepsilon = 10^{-8}$. Lines with ticks indicate the Dirichlet boundary Γ_{D}. The solution is computed with a non-linear algebraic flux-corrected (AFC) scheme using a Kuzmin limiter; see [Bar+18b].

Remark 3.20. It needs to be mentioned that even the optimal method Optimal might have values of $\mathrm{osc}_{\max}$ and $\mathrm{osc}_{\mathrm{mean}}$ that are not identically zero. This is the case if the original DG method computes a solution where the oscillations are so strong that even the integral mean is above or below $u_{\max}$ and $u_{\min}$, respectively. □

3.4.1. Application to the discrete solution to the HMM example

Example 3.21 (Convection skew to the mesh)**.** This example is based on a standard benchmark problem first proposed by Hughes, Mallet, and Mizukami [HMM86]. The problem is designed for the unit square, i.e., $\Omega \coloneqq (0,1)^2$. The convection field is given by $\boldsymbol{b} \coloneqq (\cos(-\pi/3), \sin(-\pi/3))^T$ and $c \coloneqq f \coloneqq 0$. The problem is augmented with Dirichlet boundary conditions on $\Gamma_{\mathrm{D}} \coloneqq \partial\Omega$ that are prescribed by

$$g_{\mathrm{D}} \coloneqq \begin{cases} 1, & \text{if } (y = 1 \ \wedge \ x > 0) \text{ or } (x = 0 \ \wedge \ y > 0.75), \\ 0, & \text{else.} \end{cases}$$

The domain and the convection field are also depicted in figure 3.6a. □

The solution has values in $[0,1]$ and possesses several layers: An interior layer in the direction of the convection starting at the jump of the boundary condition and two boundary layers at the outflow boundary; see figure 3.6b for a visualization of the solution. In contrast to the original problem, the discontinuity point of the boundary condition is shifted from $(0, 0.7)$ to $(0, 0.75)$ to ensure that the discontinuity point is at a vertex of the meshes used in this work. This is especially important for DG methods since the smoothness of the solution might be very low in mesh cells with a discontinuity along an edge but not at the vertices [FJ21].

The limiters are tested for $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-8}$ on triangular and quadrilateral meshes starting from the initial grids shown in figure 2.5.

The results of $\mathrm{osc}_{\max}$ and $\mathrm{osc}_{\mathrm{mean}}$ for $\varepsilon = 10^{-4}$ on the triangular grid are shown in figures 3.7 and 3.8, respectively. It can be seen that even though no method reduces the maximal spurious oscillations completely, ConstJump, ConstJumpNorm, and ConstJumpMod decrease the oscillations significantly compared to Galerkin, of which ConstJump behaves worse than ConstJumpNorm and ConstJumpMod, especially on coarser grids. The finer the mesh becomes, both ConstJumpMod and ConstJumpNorm approach the optimal method, which means that they detect the largest occurring over- and undershoot. The smaller the polynomial degree, the earlier they detect these cells. LinTriaReco does not significantly reduce $\mathrm{osc}_{\max}$ compared to Galerkin and may worsen the situation, as seen for P_2. ConstTriaReco also significantly reduces the oscillations compared to Galerkin, especially on coarser meshes where it is optimal for $p = 2$. However, the finer the grid becomes, the less it reduces the oscillations.

In figure 3.8, it can be observed that the mean oscillations $\mathrm{osc}_{\mathrm{mean}}$ decrease for all polynomial degrees the finer the mesh becomes. This behavior can be expected since the total number of mesh cells scales quadratically, whereas the number of cells in

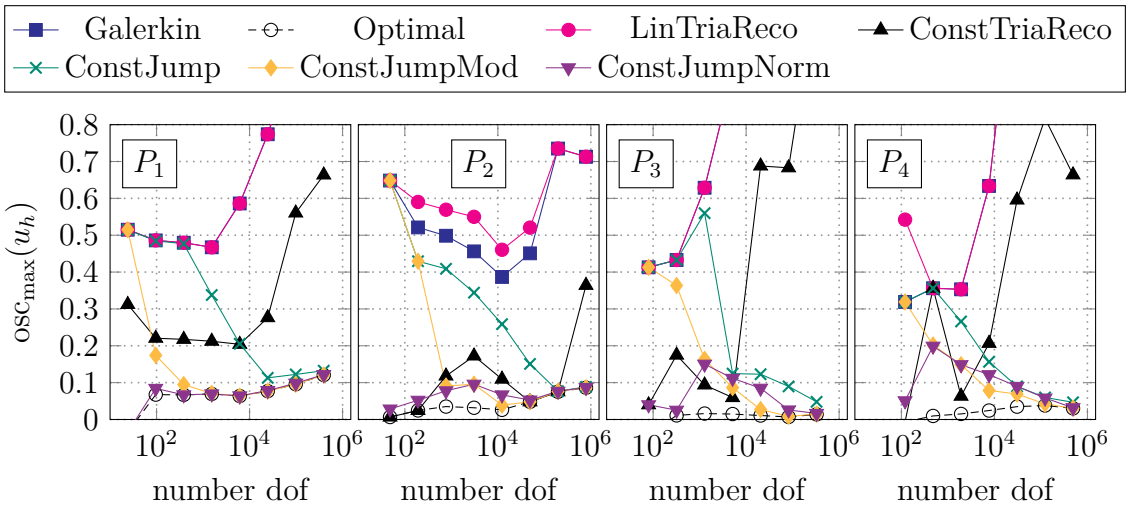


Figure 3.7.: Results of $\mathrm{osc}_{\max}$ for example 3.21 with $\varepsilon = 10^{-4}$ on the triangular mesh. Methods that are not visible have values outside the range of the plot. The results of LinTriaReco might hide the results of Galerkin.

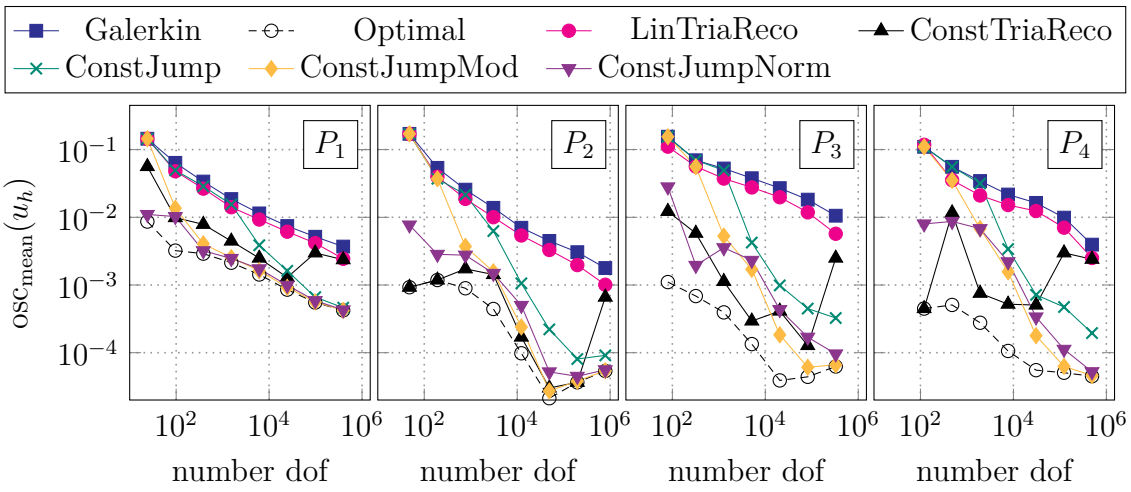


Figure 3.8.: Results of $\mathrm{osc}_{\mathrm{mean}}$ for example 3.21 with $\varepsilon = 10^{-4}$ on the triangular mesh.

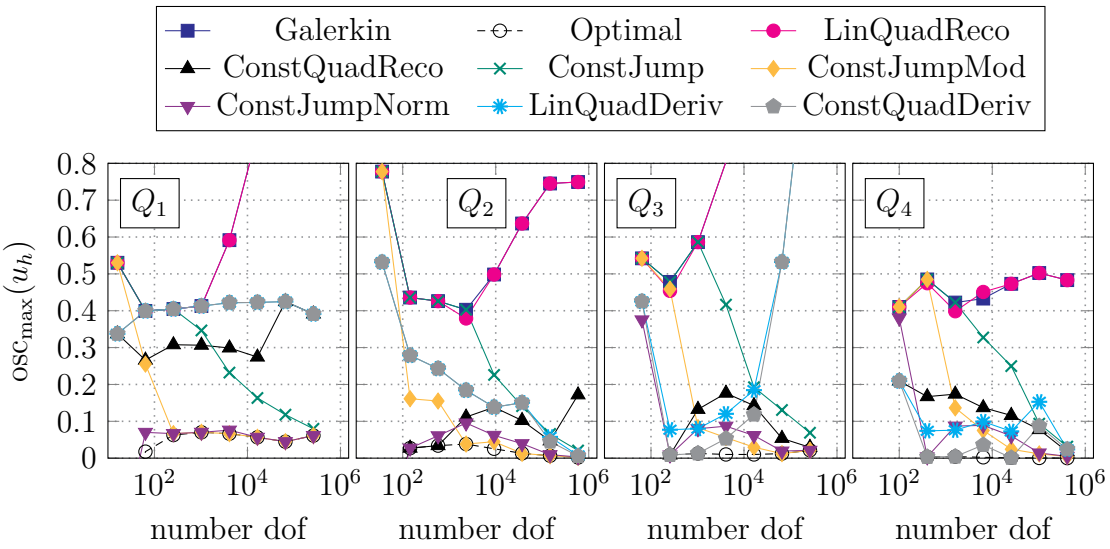


Figure 3.9.: Results of $\mathrm{osc}_{\max}$ for example 3.21 with $\varepsilon = 10^{-4}$ on the quadrilateral mesh. Methods that are not visible have values outside the range of the plot. The results of Galerkin, LinTriaReco, LinQuadDeriv, and ConstQuadDeriv might lie on each other.

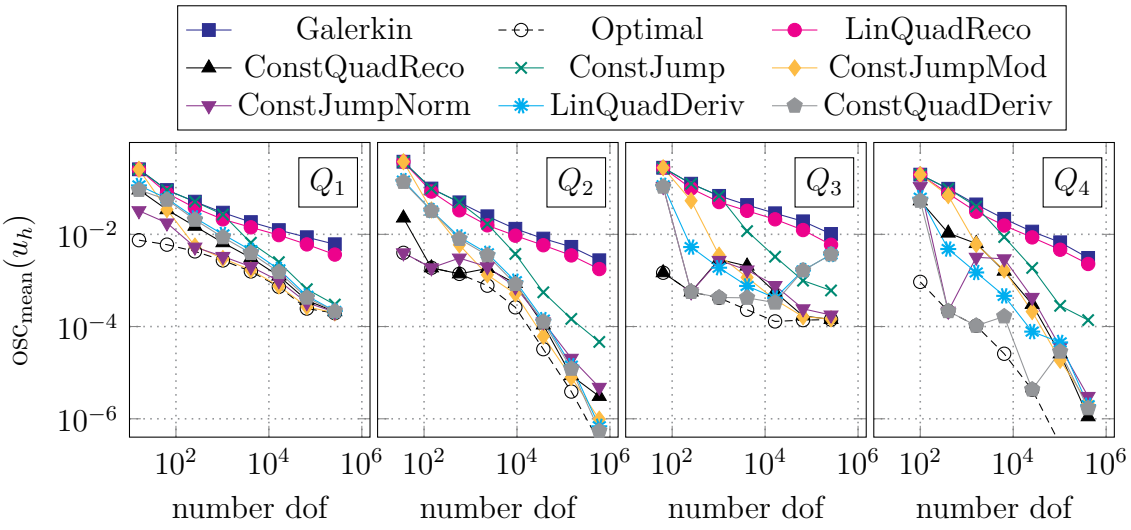


Figure 3.10.: Results of $\mathrm{osc}_{\mathrm{mean}}$ for example 3.21 with $\varepsilon = 10^{-4}$ on the quadrilateral mesh. The results of Galerkin, LinQuadDeriv and ConstQuadDeriv might lie on top of each other.

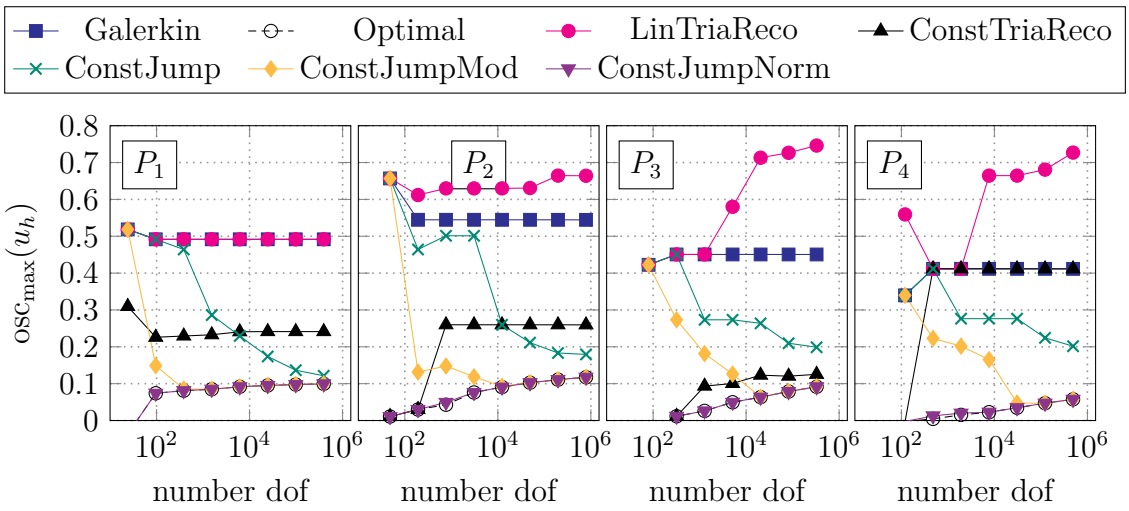


Figure 3.11.: Results of $\mathrm{osc}_{\max}$ for example 3.21 with $\varepsilon = 10^{-8}$ on the triangular mesh. The results of LinTriaReco or ConstTriaReco hide the ones of Galerkin and the results of ConstJumpMod might be hidden by the ones of ConstJumpNorm.

the vicinity of the layer, which should be marked, scales linearly. All methods reduce the mean oscillations $\mathrm{osc}_{\mathrm{mean}}$ compared to Galerkin, and ConstJumpMod and ConstJumpNorm are again among the best ones. Both almost equal the optimal method on the finest meshes, which indicates that they can detect almost all oscillations. Also, ConstJump shows a significant reduction but is not as good as ConstJumpMod and ConstJumpNorm. Furthermore, ConstTriaReco reduces the mean oscillations on coarser levels the most for $p = 2, 3, 4$ and is also optimal for P_2 and P_4 on the coarsest level.

Last but not least, it can be noted that the trends are in agreement with [FJ21].

Figures 3.9 and 3.10 show the results for $\varepsilon = 10^{-4}$ using the quadrilateral grid. As before, ConstJumpMod and ConstJumpNorm significantly reduce $\mathrm{osc}_{\max}$ compared to Galerkin, and again both are optimal on the finest meshes. ConstQuadReco and ConstJump also improve the situation, and ConstJump even works almost as well as ConstJumpNorm on the finest meshes. LinQuadDeriv and ConstQuadDeriv can also reduce the maximal oscillations, and ConstQuadDeriv is even optimal on coarser meshes for $p = 3, 4$ and the finest mesh for $p = 2, 4$. Finally, LinQuadReco does not improve the situation, and the latter makes it even worse for Q_4.

As for the triangular case, the mean measure $\mathrm{osc}_{\mathrm{mean}}$ becomes smaller for all methods as the mesh becomes finer; see figure 3.10. The methods reducing the oscillations the most are ConstQuadReco, ConstJumpMod, ConstJumpNorm, and ConstQuadDeriv and LinQuadDeriv except for the finest level for Q_3. They are almost optimal on the finest meshes for $p = 1, 3$ but far away for $p = 4$. It is concluded that they, on the

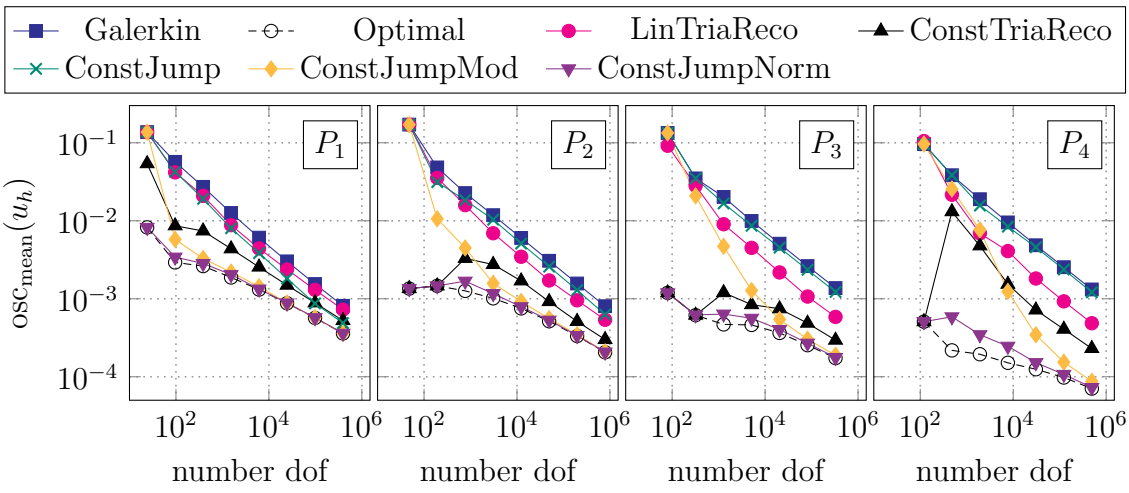


Figure 3.12.: Results of $\mathrm{osc}_{\mathrm{mean}}$ for example 3.21 with $\varepsilon = 10^{-8}$ on the triangular mesh.

one hand, detect the largest oscillations but, on the other hand, miss to find many oscillations. LinQuadReco makes almost no improvements compared to Galerkin, and ConstJump is somewhat between LinTriaReco and ConstJumpNorm.

This also corresponds to the findings in [FJ21] except that in that paper, LinQuadDeriv and ConstQuadDeriv worked worse.

In figures 3.11 and 3.12 the results of $\mathrm{osc}_{\mathrm{max}}$ and $\mathrm{osc}_{\mathrm{mean}}$, resp., for $\varepsilon = 10^{-8}$ on triangular grids are shown. As for the larger diffusion constant, ConstJumpMod and ConstJumpNorm reduce $\mathrm{osc}_{\mathrm{max}}$ the most compared to Galerkin, of which the latter one works better on coarser grids. It is even optimal for almost all degrees and refinement levels. The limiter ConstJump improves the values of maximal oscillations as well as ConstTriaReco except for P_4. LinTriaReco is no improvement compared to Galerkin (P_1) or even increases $\mathrm{osc}_{\mathrm{max}}$ (P_2, P_3, P_4). Concerning $\mathrm{osc}_{\mathrm{mean}}$ depicted in figure 3.12, ConstJumpMod and ConstJumpNorm work the best, and the reduction in the mean compared to Galerkin is even optimal on the finest meshes. On coarser grids, ConstJumpNorm is better than ConstJumpMod. ConstTriaReco also improves $\mathrm{osc}_{\mathrm{mean}}$ and an coarser grids better than ConstJumpMod. Both LinTriaReco and ConstJump improve the mean oscillations only slightly. The trends of these findings agree with the results in [FJ21].

Figures 3.13 and 3.14 show the results of $\mathrm{osc}_{\mathrm{max}}$ and $\mathrm{osc}_{\mathrm{mean}}$ for $\varepsilon = 10^{-8}$ on the quadrilateral grid. Unfortunately, LinQuadDeriv and ConstQuadDeriv do not reduce the maximal oscillations compared to Galerkin for Q_1. However, the higher the polynomial degree, the better they work, and ConstQuadDeriv is even optimal for $Q4$ and on coarser meshes for Q_3. The method LinQuadReco sometimes improves the values, but it increases the values on finer grids for Q_3 and Q_4. ConstJump

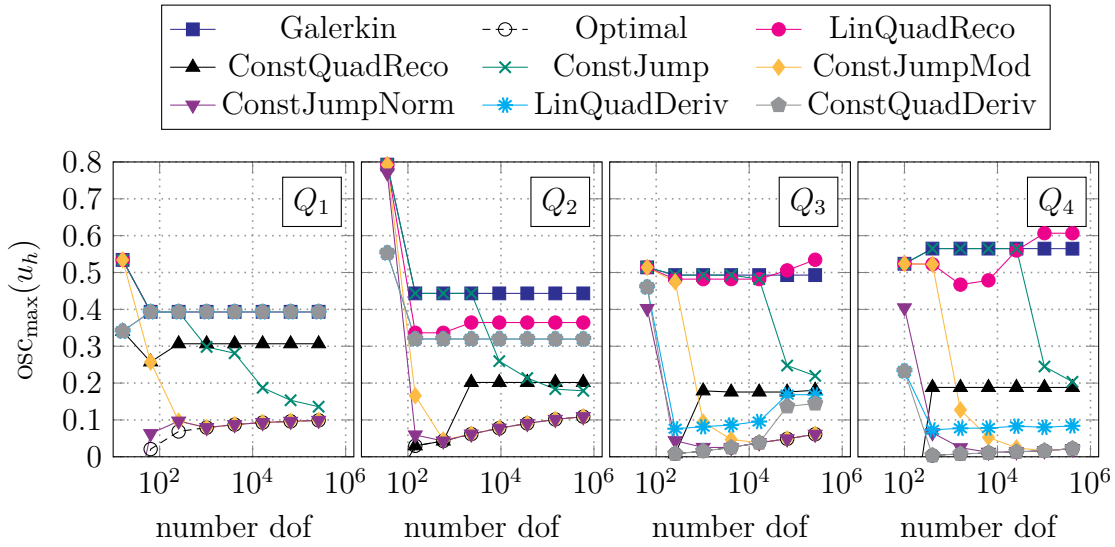


Figure 3.13.: Results of $\mathrm{osc}_{\max}$ for example 3.21 with $\varepsilon = 10^{-8}$ on the quadrilateral mesh. The results of ConstQuadDeriv lie above the ones of LinQuadReco, Galerkin, ConstJump and LinQuadDeriv.

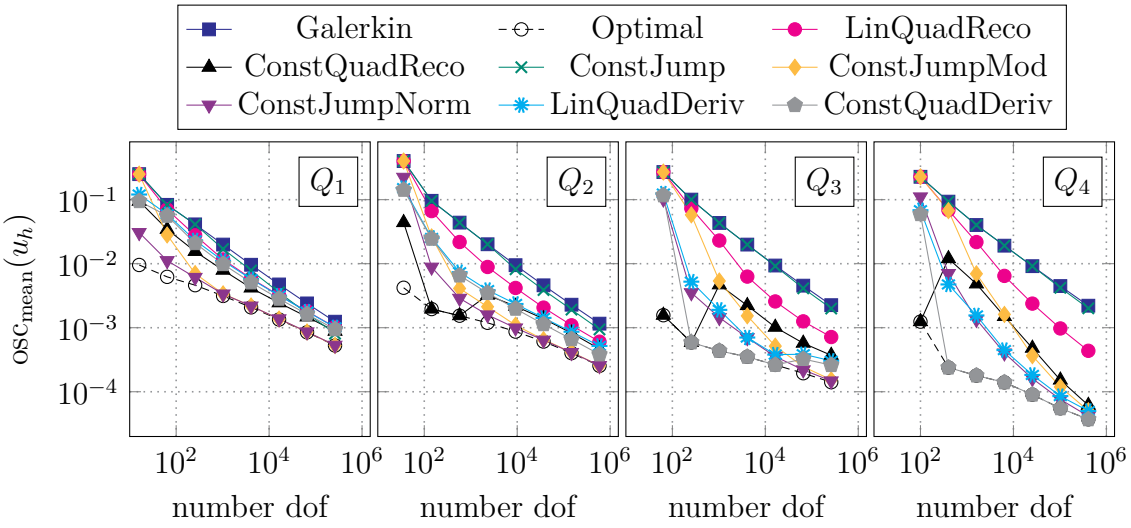
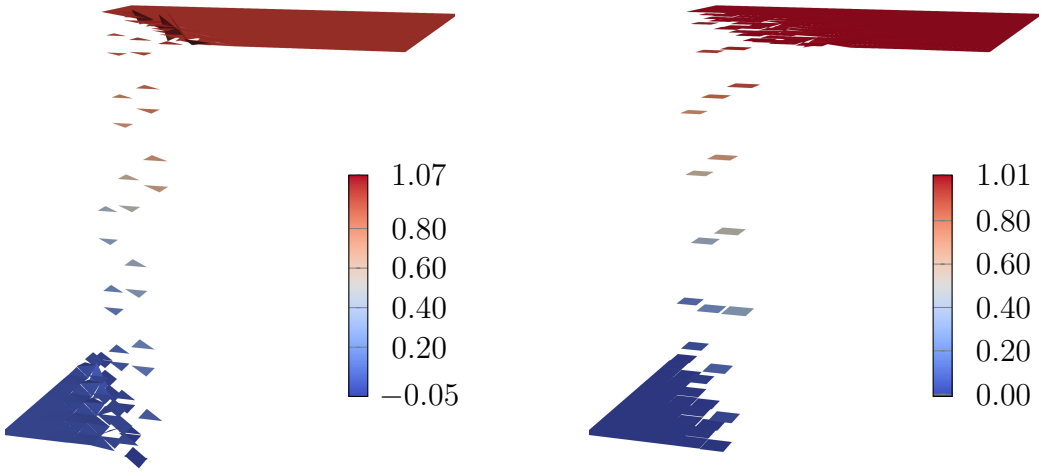


Figure 3.14.: Results of $\mathrm{osc}_{\mathrm{mean}}$ for example 3.21 with $\varepsilon = 10^{-8}$ on the quadrilateral mesh. The results of ConstQuadDeriv lie above the ones of Galerkin, ConstJump and LinQuadDeriv.

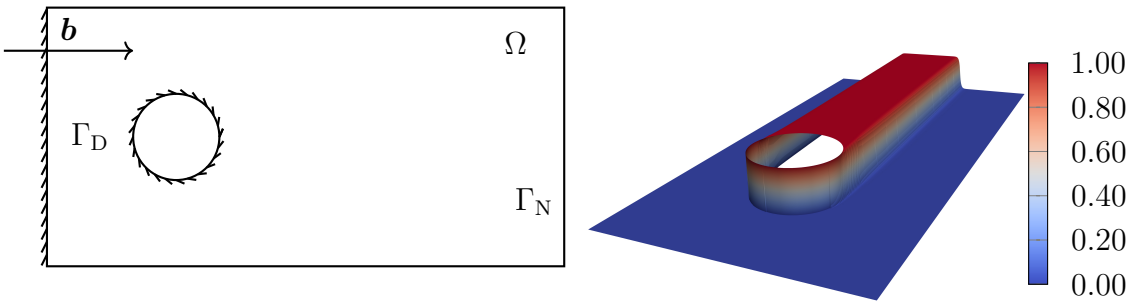


(a) P_2 solution with ConstJumpMod. (b) Q_4 solution with ConstJumpNorm.

Figure 3.15.: Limited solutions to example 3.21 on the fourth level for $\varepsilon = 10^{-8}$. The solutions are projected to piecewise linear or bilinear functions by the visualization software.

reduces the oscillations if the grid is fine enough but does not help much if the grid is coarse. ConstQuadReco also reduces the maximal oscillations, but it shows a constant behavior after a few refinements. On the other hand, ConstJumpMod and ConstJumpNorm show the best results with an optimal reduction if the mesh is fine enough. On coarser grids, ConstJumpNorm is again better than ConstJumpMod and even optimal. Those methods are also the best ones with respect to $\mathrm{osc}_{\mathrm{mean}}$, both approaching Optimal the finer the mesh becomes, as seen in figure 3.14. Only ConstQuadReco on coarser grids is slightly better for $p = 2, 3, 4$. LinQuadDeriv and ConstQuadDeriv also show a reduction compared to Galerkin, especially the higher the polynomial degree is. For Q_3 and Q_4, ConstQuadDeriv is optimal on almost all levels. ConstJump shows almost no reduction compared to Galerkin. Lust but not least, LinQuadReco reduces the mean oscillations moderately compared to Galerkin. The results are similar to the one obtained in [FJ21; FJ22].

In figure 3.15, two selected numerical solutions of approaches that lead to good results are shown. Since the boundary conditions are imposed weakly, the boundary layers at the outflow boundary are absent. In the left solution that is limited with ConstJumpMod, some under- and overshoots remain because the corresponding cells are not marked. All cells in the layer are correctly marked in the right solution gained by applying ConstJumpNorm. As a result, almost no over- and undershoots are present.



(a) Domain. (b) Sketch of the solution.

Figure 3.16.: Domain for and sketch of the solution for $\varepsilon = 10^{-8}$ to example 3.22. Lines with ticks indicate the Dirichlet boundary Γ_D, and lines without ticks indicate the Neumann boundary Γ_N. The solution is computed with a non-linear algebraic flux-corrected (AFC) scheme using a Kuzmin limiter; see [Bar+18b]

3.4.2. Application to the discrete solution to the Hemker example

This example takes values in $[0, 1]$ which can also be seen in a sketch of the solution; see figure 3.16b. Again $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-8}$ are chosen, and the computations take place on a mesh where the initial grid can be seen in figure 3.17.

Example 3.22 (Hemker problem)**.** The second benchmark problem defined first by Hemker [Hem96] is stated in $\Omega := \{(-3, 9) \times (-3, 3)\} \setminus \{(x, y) \ : \ x^2 + y^2 \leq 1\}$, with the coefficients $\boldsymbol{b} := (1, 0)^T$ and $c := f := 0$. The problem is augmented with Dirichlet boundary conditions with $g_D := 0$ at $x = -3$, with $g_D := 1$ at the circular boundary given by $x \in [-1, 1]$ and $y^2 = 1 - x^2$. Homogeneous Neumann boundary conditions are prescribed on all other boundaries; see figure 3.16a.

This example models the transport of a quantity, e.g., temperature, through a channel. The quantity of interest flows in the direction of the convection field starting at the circle and spreads out due to diffusion. □

In figures 3.18 and 3.19 the results for $\varepsilon = 10^{-4}$ are shown. The best results are achieved with ConstJumpMod and ConstJumpNorm, which lead to almost equal

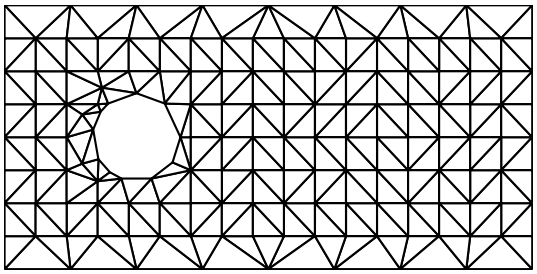


Figure 3.17.: Initial grid for example 3.22.

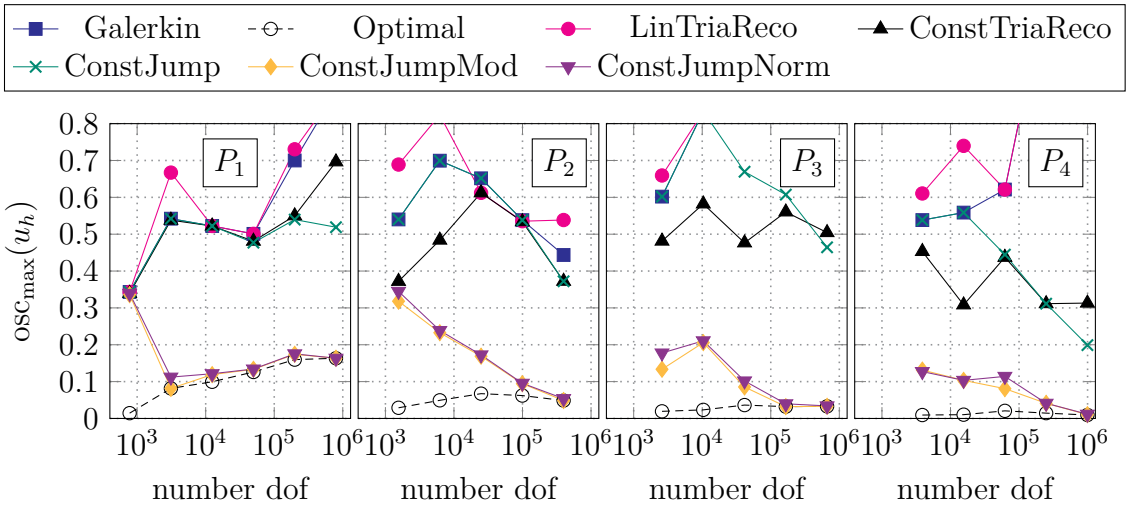


Figure 3.18.: Results of $\mathrm{osc}_{\max}$ for example 3.22 with $\varepsilon = 10^{-4}$. Methods that are not visible have values outside the range of the plot. The results of ConstJumpNorm lie above the ones of ConstJumpMod.

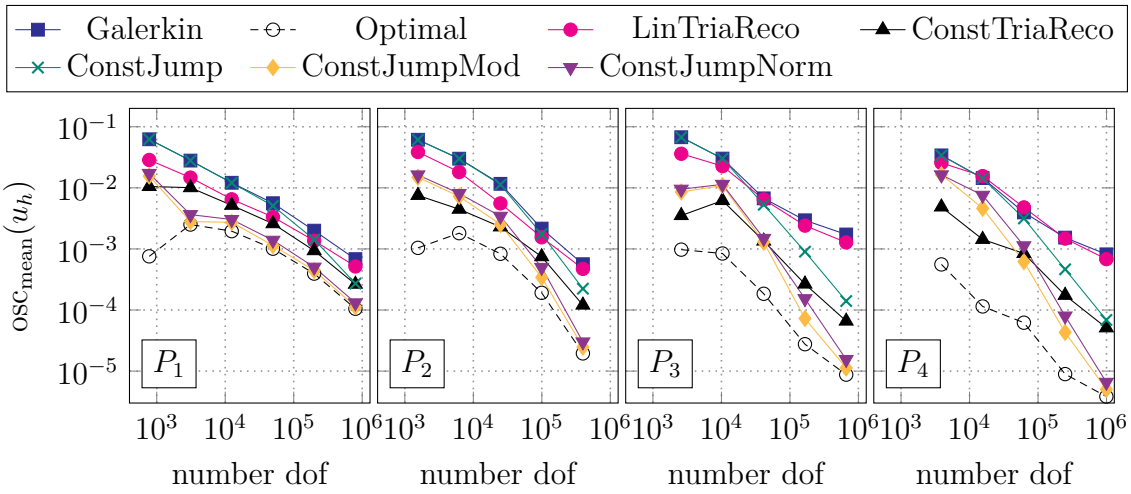


Figure 3.19.: Results of $\mathrm{osc}_{\mathrm{mean}}$ for example 3.22 with $\varepsilon = 10^{-4}$.

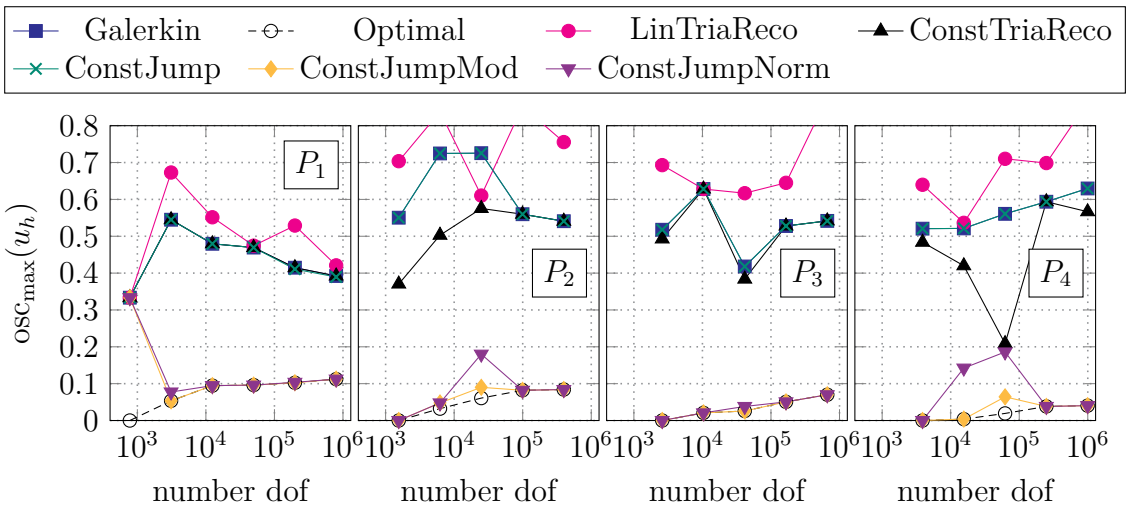


Figure 3.20.: Results of $\mathrm{osc}_{\max}$ for example 3.22 with $\varepsilon = 10^{-8}$. Methods that are not visible have values outside the range of the plot. The results of ConstJump might lie above the ones of Galerkin and ConstTriaReco.

values. They are also optimal on the finest meshes, and for P_4, almost all oscillations are removed. LinTriaReco for all degrees shows almost no improvement at all. The methods ConstTriaReco and ConstJump are better than Galerkin but far from optimal. For the second quantity of interest $\mathrm{osc}_{\mathrm{mean}}$, again ConstJumpMod and ConstJumpNorm reduce the oscillations the most, followed by ConstTriaReco, which in turn is better on coarser grids. Only on the finest grid, ConstJumpMod and ConstJumpNorm come close to the optimal reduction of oscillations. The other methods also reduce the mean values as expected but much less than the best ones.

For $\varepsilon = 10^{-8}$ the results are shown in figures 3.20 and 3.21. Concerning $\mathrm{osc}_{\max}$, ConstJump does not improve the values compared to Galerkin, and LinTriaReco again increases the maximal oscillations. ConstTriaReco also slightly decreases $\mathrm{osc}_{\max}$ for odd degrees and notably reduces the oscillations for even degrees on coarser grids. The best methods are ConstJumpMod and ConstJumpNorm, where ConstJumpMod is slightly better than ConstJumpNorm on medium fine grids. They both are optimal on the finest grids and ConstJumpMod is also already optimal on coarser grids for odd degrees. Also for $\mathrm{osc}_{\mathrm{mean}}$ the method ConstJump does reduce the mean oscillations only slightly (P_1) or almost not at all (P_2, P_3,P_4). The methods LinTriaReco and ConstTriaReco improve the values but not as much as ConstJumpMod and ConstJumpNorm, which in fact, are again the best methods. They are on the finest meshes, also optimal for $p = 1, 2$ and almost optimal for $p = 3, 4$. Similar results are also obtained in [FJ21; FJ22].

In figure 3.22, two limited solutions are depicted. It can be observed that in the

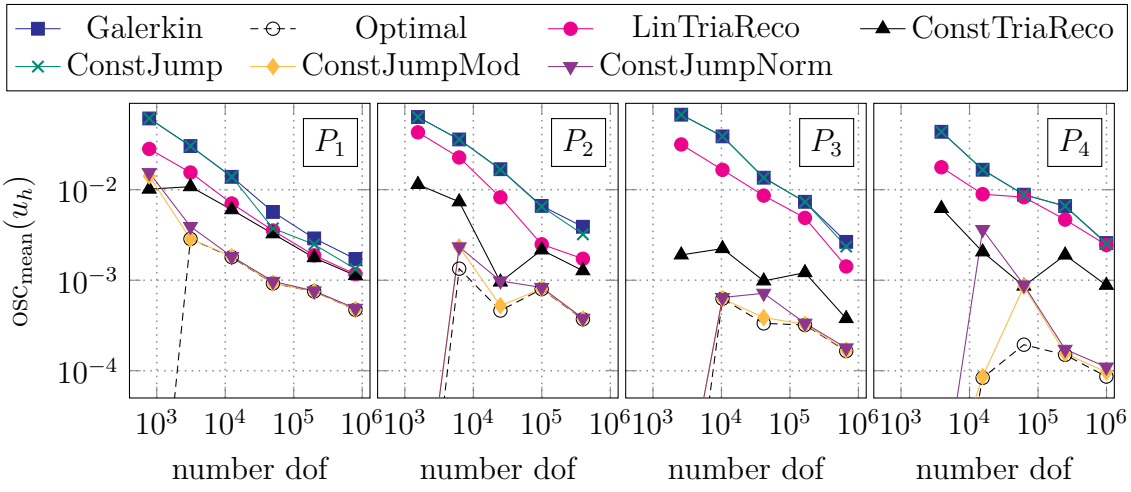
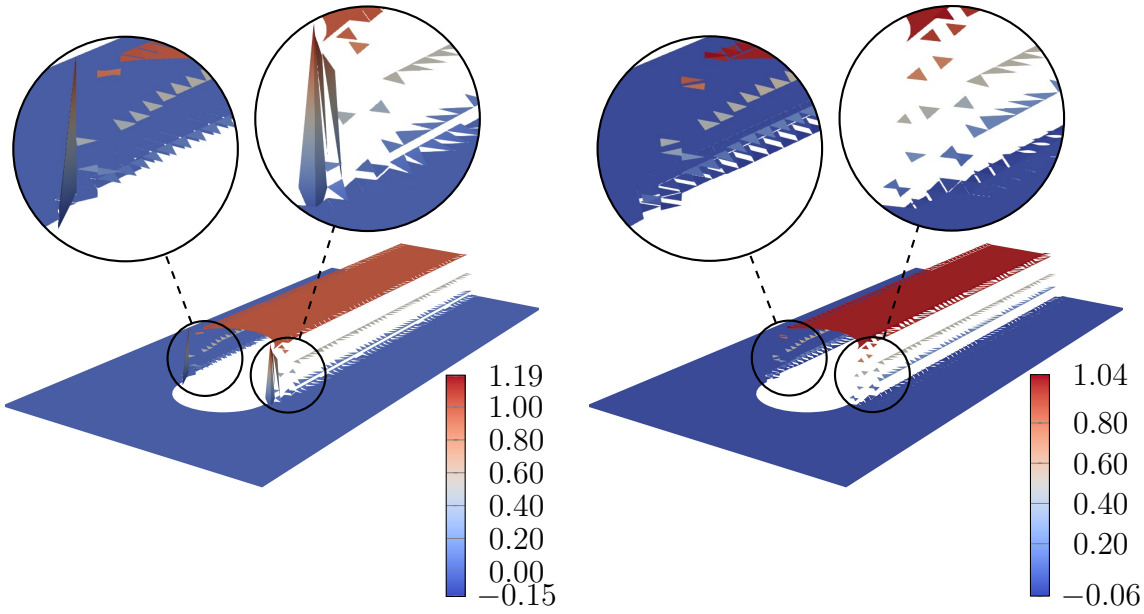


Figure 3.21.: Results of $\mathrm{osc}_{\mathrm{mean}}$ for example 3.22 with $\varepsilon = 10^{-8}$.



(a) P_3 solution with ConstTriaReco.

(b) P_1 solution with ConstJumpNorm.

Figure 3.22.: Limited solutions to example 3.22 on the third level for $\varepsilon = 10^{-8}$. The solutions are projected to piecewise linear functions by the visualization software.

solution that is limited with ConstTriaReco, some cells in the layer close to the circle are not marked, which is why the solution still possesses a notable amount of over- and undershoots. This is not the case for the solution limited with ConstJumpNorm as seen on the right of figure 3.22.

3.5. Summary

In this chapter, several post-processing techniques from the literature and novel methods were presented and numerically investigated. Coming from the plain DG solution, these methods first mark cells that detect spurious oscillations and change the solution on the marked cells to an affine or constant approximation. These methods are computationally cheap, and they all share the property of preserving the mass locally, as shown above, which clearly distinguishes them from just clipping the extrema.

The methods were tested on two benchmark problems for different diffusion coefficients, meshes, and polynomial degrees. Two measures, $\mathrm{osc}_{\mathrm{max}}$ and $\mathrm{osc}_{\mathrm{mean}}$, were introduced to evaluate the methods' performance. It can be noted that the above-described methods that use an affine reconstruction worked worse in most cases than methods with piecewise constant approximations. The former methods were able to reduce the oscillations in the mean, but at the price that they sometimes even increased the maximal oscillations. In many experiments, even the best methods could not remove the oscillations altogether, but they reduced the oscillations significantly. This is because the oscillations in the plain DG method were so strong that even the integral mean is too large or too small, resp., leading to positive values of $\mathrm{osc}_{\mathrm{max}}$ and $\mathrm{osc}_{\mathrm{mean}}$. Usually among the best and often the best methods were ConstJumpMod and ConstJumpNorm, two novel methods. Other methods might have produced better results for a particular setting but were worse in many other configurations. However, all methods might be improved if the parameters they rely on are optimized for the problem. Especially on the finest meshes, ConstJumpMod and ConstJumpNorm were able to detect all relevant cells, i.e., they reduced the oscillations as much as possible while preserving the integral mean. In this way, they were optimal, and improvements for such post-processing methods are only possible on coarser meshes.

As mentioned above, some methods might be improved if their parameters are optimized with respect to the problem. However, since even the optimal reference method still shows over- and undershoots, reducing these oscillations in the plain DG method might be more beneficial before further improving the slope limiters. One option in that direction would be to optimize the plain solution with respect to the stabilization parameter σ_E in the same light as it was done, e.g., in [JKS11; JK13; KLS19; JKW23].

4. Deep neural networks as spurious oscillations detector

Within the last two decades, deep learning techniques in general and neural networks especially started to reveal their full potential. They are frequently used in many scientific fields as well as in businesses, and a world without them is nowadays hard to imagine; see, e.g., [GBC16; Sar21; HH19] for applications and state-of-the-art techniques. Due to their ability to be universal function approximators [HSW89; Cyb89; GBC16, chapter 6.4.1], they can approximate highly non-linear mappings or even mappings without a known representation.

Their ability to approximate unknown mappings can also be exploited in the context of numerical methods for initial-boundary value problems, and there exist already many papers that do so; see, for instance, [Jos+21; MLR21; SC21; BFM19; vWR21; Kar+21]. In this chapter, a way is drawn to combine neural networks with ideas of the slope limiters introduced in the previous chapter. Similar ideas have already been presented in [RH18; RH19; Liu+19; VA18; AH20; Mor+20; Bec+20]. However, these publications deal with discretization schemes for hyperbolic (systems of) equations, mainly Euler's Gas equations. Therefore, the goal of this chapter is to extend these results to convection-dominated convection-diffusion-reaction problems, i.e., the construction of a neural network-based slope limiter for these problems.

Let us shortly recall the essential ingredients of classical slope limiters: In the first step, they mark cells where the solution might show spurious oscillations. Afterwards, the solution is changed locally on these cells. However, marking cells is not unique to slope limiters. For instance, in adaptive refinement strategies, cells have to be selected where the error of the discrete solution to the exact solution is large, and in parameter-dependent methods like SUPG or DG methods, parameters for every cell need to be chosen. Therefore, many applications would benefit from a black box that, given some features of the solution, predicts a particular output, e.g., whether a cell is marked or not or how to choose a particular parameter. Hence, what is presented and the understanding gained in this chapter has to be viewed in a broader field of use cases.

This chapter is structured as follows: In section 4.1 the basic ideas of a particular type of neural network are explained including their construction and how they are trained to approximate a particular mapping. Afterwards, section 4.2 deals with how the data set is constructed with which in section 4.3 multilayer perceptron models are trained. Their quality is measured, and the best is applied to the solution to the HMM and the Hemker example in section 4.4. This chapter finally ends with a

summary given in section 4.5.

This chapter's primary results and content are also published in the preprint [FHJ22]. Only in the numerical examples, a slightly different set-up is used compared to that reference.

4.1. Basics of multilayer perceptron models

Deep neural networks are a machine learning technique to approximate a possibly unknown mapping $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^m$ and $n, m \in \mathbb{N}$ [GBC16, pp. 1–8]. There exist different types of neural networks, e.g., convolutional neural networks, recurrent neural networks, or generative adversarial networks, to name just a few. However, in the following so-called *multilayer perceptrons* (MLPs) are used and hence introduced, cf., e.g., [HH19; GBC16, chapter 6] for different presentations.

4.1.1. Structure of multilayer perceptron models

MLPs consist of $\ell \in \mathbb{N}$ so-called *layers*, which itself are build from $n_i \in \mathbb{N}$, $i = 1, 2, \dots, \ell$, nodes. They are also often referred to as *neurons*, and each represents a scalar value. The ith layer takes some input and returns some output $\widehat{\boldsymbol{y}}^{[i]} \in \mathbb{R}^{n_i}$. The first layer, the so-called *input layer*, is special since it only represents the input $\boldsymbol{x} \in \mathbb{R}^n$, i.e., $n_1 \coloneqq n$, and $\widehat{\boldsymbol{y}}^{[1]} \coloneqq \boldsymbol{x}$. All subsequent layers take as input the output $\widehat{\boldsymbol{y}}^{[i-1]} \in \mathbb{R}^{n_{i-1}}$, $i = 2, 3, \dots, \ell$, of the previous layer and perform the following computations to return the current output $\widehat{\boldsymbol{y}}^{[i]}$. The jth node, $j = 1, 2, \dots, n_i$, in the ith layer returns the jth component $\widehat{y}_j^{[i]}$ of the current layer of $\widehat{\boldsymbol{y}}^{[i]}$ by

$$\widehat{y}_j^{[i]}\left(\widehat{\boldsymbol{y}}^{[i-1]}\right) \coloneqq \sigma_j^{[i]}\left(\sum_{k=1}^{n_{i-1}} w_{j,k}^{[i]}\, \widehat{y}_k^{[i-1]} + b_j^{[i]}\right) \in \mathbb{R}, \tag{4.1}$$

where $w_{j,k}^{[i]} \in \mathbb{R}$ are called *weights*, $b_j^{[i]} \in \mathbb{R}$ is the jth component of the so-called *bias* and $\sigma_j^{[i]} : \mathbb{R} \to \mathbb{R}$ is a non-linear mapping called *activation function*. While all intermediate layers, i.e., layers where $i = 2, 3, \dots, \ell - 1$, are called *hidden layers*, the last layer is referred to as the *output layer* and has, therefore, $n_\ell = m$ nodes. Furthermore, the output of the last layer $\widehat{\boldsymbol{y}}^{[\ell]}$ may be denoted by $\hat{\boldsymbol{y}}$. Equation (4.1) can be more compactly written in matrix form as

$$\widehat{\boldsymbol{y}}^{[i]}\left(\widehat{\boldsymbol{y}}^{[i-1]}\right) = \boldsymbol{\sigma}^{[i]}\left(\boldsymbol{W}^{[i]}\widehat{\boldsymbol{y}}^{[i-1]} + \boldsymbol{b}^{[i]}\right) \in \mathbb{R}^{n_i} \qquad (i = 2, 3, \dots, \ell),$$

where $\boldsymbol{W}^{[i]} \in \mathbb{R}^{n_i \times n_{i-1}}$ collects all weights, $\boldsymbol{b}^{[i]} \in \mathbb{R}^{n_i}$ is the bias vector, and $\boldsymbol{\sigma}^{[i]} : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is the component wise defined activation function.

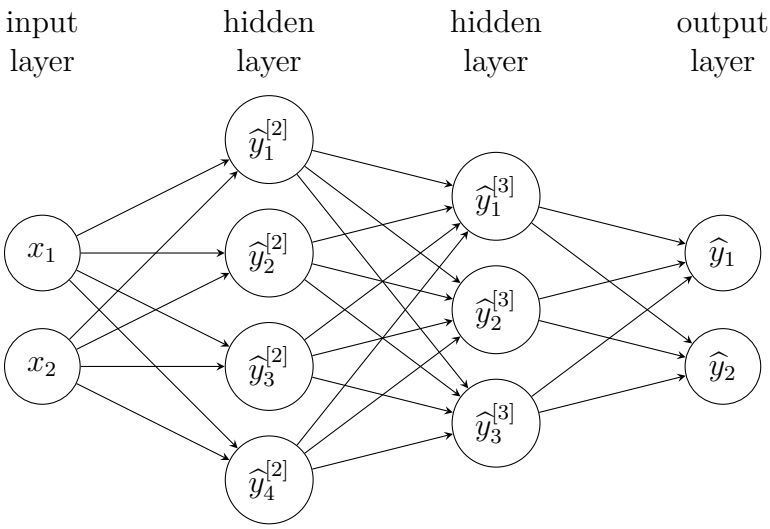


Figure 4.1.: Representation of a multilayer perceptron that maps from $\mathbb{R}^2$ to $\mathbb{R}^2$ with two hidden layers of size 4 and 3, respectively. The value in the nodes is computed by equation (4.1), and the arrows indicate which quantities of the previous layer are used to compute the value in the respective node.

Altogether, given an input $\boldsymbol{x} \in \mathbb{R}^n$ an MLP returns an output $\widehat{\boldsymbol{y}}$ given by

$$\begin{aligned}\widehat{\boldsymbol{y}}(\boldsymbol{x}) &= \widehat{\boldsymbol{y}}^{[\ell]}\left(\widehat{\boldsymbol{y}}^{[\ell-1]}\left(\dots\widehat{\boldsymbol{y}}^{[2]}\left(\boldsymbol{x}\right)\dots\right)\right)\\ &= \boldsymbol{\sigma}^{[\ell]}\left(\boldsymbol{W}^{[\ell]}\boldsymbol{\sigma}^{[\ell-1]}\left(\boldsymbol{W}^{[\ell-1]}\dots\boldsymbol{\sigma}^{[2]}\left(\boldsymbol{W}^{[2]}\boldsymbol{x}+\boldsymbol{b}^{[2]}\right)\dots+\boldsymbol{b}^{[\ell-1]}\right)+\boldsymbol{b}^{[\ell]}\right).\end{aligned}$$

MLPs can be parametrized by their so-called *architecture*, i.e., their respective number of hidden layers, nodes per layer, activation functions, weights, and biases. While the latter two are often referred to as *parameters*, the former are called *hyperparameters*.

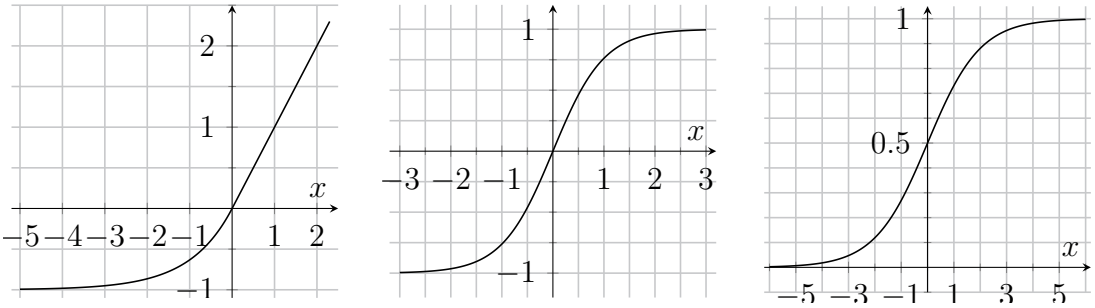
Figure 4.1 shows a typical representation of a multilayer perceptron with $\ell = 4$ layers (input, output, and two hidden layers) and $n_1 = 2$, $n_2 = 4$, $n_3 = 3$, $n_4 = 2$ nodes per layer. Examples of activation functions are the commonly used exponential linear unit (ELU) function, the hyperbolic tangent tanh, and the sigmoid function. They are given by

$$\mathrm{ELU}(x) \coloneqq \begin{cases} e^x - 1, & \text{if } x < 0,\\ x, & \text{else,}\end{cases} \tag{4.2a}$$

$$\tanh(x) \coloneqq \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{4.2b}$$

$$\mathrm{sigmoid}(x) \coloneqq \frac{1}{1 + e^{-x}}, \tag{4.2c}$$

and are depicted in figure 4.2.



(a) ELU. (b) tanh. (c) sigmoid.

Figure 4.2.: Different non-linear activation functions given in equations (4.2a) to (4.2c).

4.1.2. Training process

To reach the goal that an MLP with given hyperparameters approximates an unknown function, the parameters have to be adapted correspondingly. This is done in a process called *training*. There exist different training strategies, e.g., supervised, unsupervised and reinforcement learning; see also [GBC16, pp. 103–104]. In the following, the supervised training technique is introduced. To approximate the unknown function $\boldsymbol{f} : \mathcal{X} \to \mathcal{Y}$ a finite data set $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$ is needed. This set is made up from pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{D}$, $i \in \mathbb{N}$, of *features* $\boldsymbol{x}_i$ and *labels* $\boldsymbol{y}_i \coloneqq \boldsymbol{f}(\boldsymbol{x}_i)$. Let $\boldsymbol{p}$ denote the parameters of an MLP, i.e., the collection of all weights and biases. In the training process, the parameters are adapted or rather optimized to minimize a given loss functional $\mathcal{L}$ that maps a concrete choice of parameters to a real number. For instance, if $\mathcal{Y} = \{\, 0, 1 \,\}$ a typical loss functional is the so-called *binary cross-entropy* loss given by

$$\mathcal{L}(\boldsymbol{p}; \mathcal{D}) \coloneqq -\frac{1}{N} \sum_{i=1}^{N} y_i \log\left(\widehat{y}_i\right) + (1 - y_i) \log\left(1 - \widehat{y}_i\right), \tag{4.3}$$

where N denotes the number of examples in the data set $\mathcal{D}$, y_i is the ith label in the data set and $\widehat{y}_i$ the output of the MLP for the corresponding feature $\boldsymbol{x}_i$. Note that the loss functional also depends on the chosen data set. The parameters are then updated iteratively by some optimization routine. In the case of gradient descent, they are updated in each step by

$$\boldsymbol{p} \to \boldsymbol{p} - \eta \nabla_{\boldsymbol{p}} \mathcal{L}(\boldsymbol{p}; \mathcal{D}),$$

where $0 < \eta \in \mathbb{R}$ is a positive step width that is often referred to as *learning rate*, and $\nabla_{\boldsymbol{p}}$ denotes the partial derivatives with respect to the parameters.

4.2. On the data set

The data set is crucial for training an MLP because the parameters are chosen such that the MLP fits the data as well as possible. In this chapter, MLPs are created, which try to mimic the behavior of the post-processing techniques from chapter 3. It is advantageous to rewrite the techniques to obtain a better description of the data used to train the MLPs. Refining the steps on page 41, the post-processing techniques consist of

1. marking suspicious cells by
   a) computing (cell wise) features of the solution,
   b) based on the features deciding whether to mark a cell or not, and
2. replacing the solution locally on all marked cells.

To this extent, the function $\mathcal{F}_l : V_h^p \times \mathcal{T}_h \to \mathbb{R}^{n_l}$ that maps locally a discrete function to $n_l \in \mathbb{N}$ features, and a decision-maker function $\mathcal{M}_l : \mathbb{R}^{n_l} \to \{0, 1\}$ can be defined, which encode items 1a and 1b, respectively. The post-processing techniques then can be rewritten as mappings $l : V_h^p \to V_h^p$ that are cell wise defined on a cell $K \in \mathcal{T}_h$ as

$$l(u_h)|_K \coloneqq \begin{cases} u_h|_K, & \text{if } \mathcal{M}_l\left(\mathcal{F}_l\left(u_h, K\right)\right) = 0, \\ \Pi_{l,K}\left(u_h\right), & \text{else,} \end{cases}$$

where $u_h \in V_h^p$, and $\Pi_{l,K} : V_h^p \to \mathcal{R}_p$ is a local reconstruction function.

The different post-processing techniques from chapter 3 can now be recast by defining the corresponding mappings. Since in this chapter only LinTriaReco, ConstTriaReco, ConstJumpMod, and ConstJumpNorm are used, they are recalled in the following. However, for the other methods, the definition of corresponding mappings would be possible as well.

LinTriaReco To reformulate LinTriaReco, let the notation be as in section 3.1.1, and $\mathrm{tol} \in \mathbb{R}$ be a positive tolerance. The feature mapping $\mathcal{F}_{\mathrm{LTR}}(u_h, K)$ of LinTriaReco is given by

$$\mathcal{F}_{\mathrm{LTR}}(u_h, K) \coloneqq \{\, \overline{u}_{h,K_0'},\ u_h|_K(m_0),\ \overline{u}_{h,K_1'},\ u_h|_K(m_1),\ \overline{u}_{h,K_2'},\ u_h|_K(m_2),\ \overline{u}_{h,K},\ \mathrm{tol} \,\} \tag{4.4}$$

and it is $n_{\mathrm{LTR}} = 8$.

Let $[a, b; \mathrm{tol}] \coloneqq [\min(a, b) - \mathrm{tol}, \max(a, b) + \mathrm{tol}]$, $a, b \in \mathbb{R}$, be a closed interval.

The decision-maker function $\mathcal{M}_{\mathrm{LTR}}$ can then be defined as

$$\mathcal{M}_{\mathrm{LTR}}(\mathcal{F}_{\mathrm{LTR}}(u_h, K)) \coloneqq \begin{cases} 1, & \text{if } \mathcal{E}_h(K) \cap \partial\mathcal{E}_h = \emptyset \wedge \\ & \Big(u_h|_K(m_0) \notin \left[\overline{u}_{h,K_0'}, \overline{u}_{h,K}; \mathrm{tol}\right] \ \vee \\ & \ u_h|_K(m_1) \notin \left[\overline{u}_{h,K_1'}, \overline{u}_{h,K}; \mathrm{tol}\right] \ \vee \\ & \ u_h|_K(m_2) \notin \left[\overline{u}_{h,K_2'}, \overline{u}_{h,K}; \mathrm{tol}\right] \Big), \\ 0, & \text{else.} \end{cases} \tag{4.5}$$

For the sake of brevity, the reconstruction procedure is not recapitulated; see section 3.1.1 for details.

ConstTriaReco Recasting ConstTriaReco from section 3.1.2 works as follows: The feature mapping is defined as

$$\mathcal{F}_{\mathrm{CTR}}(u_h, K) \coloneqq \{\, \overline{u}_{h,K_0'},\ \overline{u}_{h,K}^{E_0},\ \overline{u}_{h,K_1'},\ \overline{u}_{h,K}^{E_1},\ \overline{u}_{h,K_2'},\ \overline{u}_{h,K}^{E_2},\ \overline{u}_{h,K},\ \mathrm{tol} \,\}, \tag{4.6}$$

where $\overline{u}_{h,K}^{E_i} \coloneqq \int_{E_i} u_h|_K \,\mathrm{d}s/|E_i|$ is the integral mean along edge $E_i \in \mathcal{E}_h(K)$, $i = 0, 1, 2$. It follows that $n_{\mathrm{CTR}} = 8$.

The corresponding decision-maker function is given by

$$\mathcal{M}_{\mathrm{CTR}}(\mathcal{F}_{\mathrm{CTR}}(u_h, K)) \coloneqq \begin{cases} 1, & \text{if } \overline{u}_{h,K}^{E_0} \notin \left[\overline{u}_{h,K_0'}, \overline{u}_{h,K}; \mathrm{tol}\right] \ \vee \\ & \ \overline{u}_{h,K}^{E_1} \notin \left[\overline{u}_{h,K_1'}, \overline{u}_{h,K}; \mathrm{tol}\right] \ \vee \\ & \ \overline{u}_{h,K}^{E_2} \notin \left[\overline{u}_{h,K_2'}, \overline{u}_{h,K}; \mathrm{tol}\right], \\ 0, & \text{else.} \end{cases} \tag{4.7}$$

To reconstruct the solution, $\Pi_{\mathrm{CTR},K}(u_h) \coloneqq \overline{u}_{h,K}$ is used; see also section 3.1.2.

ConstJumpMod This post-processing technique from section 3.3.2 has as feature mapping

$$\mathcal{F}_{\mathrm{CJM}}(u_h, K) \coloneqq \{\, \alpha_{E_0}, \alpha_{E_1}, \alpha_{E_2}, \alpha_{\mathrm{ref}} \,\}, \tag{4.8}$$

where $E_i \in \mathcal{E}_h(K)$, $i = 0, 1, 2$. Hence, it follows that $n_{\mathrm{CJM}} = 4$.

The decision-maker function for ConstJumpMod is given by

$$\mathcal{M}_{\mathrm{CJM}}(\mathcal{F}_{\mathrm{CJM}}(u_h, K)) \coloneqq \begin{cases} 1, & \text{if } \min_{i=0,1,2} \alpha_{E_i} < \alpha_{\mathrm{ref}}, \\ 0, & \text{else.} \end{cases} \tag{4.9}$$

By the definition of α_{E_i}, they may be of infinite value, which is problematic from an implementational point of view. To circumvent this, these values can be changed to α_{ref} without modifying the outcome of $\mathcal{M}_{\mathrm{CJM}}$.

The solution on the marked cells is then approximated by $\Pi_{\mathrm{CJM},K}(u_h) \coloneqq \overline{u}_{h,K}$, i.e., by its integral mean.

(a) Regular grid. (b) Irregular grid.

Figure 4.3.: Initial meshes for creating the data set described in section 4.2.1.

ConstJumpNorm To rewrite the method from section 3.3.3, the decision-maker function

$$\mathcal{F}_{\mathrm{CJN}}(u_h, K) \coloneqq \{\, \beta_{E_0}, \beta_{E_1}, \beta_{E_2}, \beta_{\mathrm{ref}} \,\}, \tag{4.10}$$

where $E_i \in \mathcal{E}_h(K)$, $i = 0, 1, 2$, can be defined which implies $n_{\mathrm{CJN}} = 4$.

The decision whether to mark a cell or not is based on

$$\mathcal{M}_{\mathrm{CJN}}(\mathcal{F}_{\mathrm{CJN}}(u_h, K)) \coloneqq \begin{cases} 1, & \text{if } \max_{i=0,1,2} \beta_{E_i} \geq \beta_{\mathrm{ref}}, \\ 0, & \text{else} \end{cases} \tag{4.11}$$

and the solution is replaced by $\Pi_{\mathrm{CJN},K}(u_h) \coloneqq \overline{u}_{h,K}$.

4.2.1. Generating the data set

The concrete idea is to approximate the decision-maker functions $\mathcal{M}_{\mathrm{LTR}}$, $\mathcal{M}_{\mathrm{CTR}}$, $\mathcal{M}_{\mathrm{CJM}}$ and $\mathcal{M}_{\mathrm{CJN}}$ by MLPs. Therefore, data is needed consisting of pairs of features of the respective post-processing technique and the corresponding label 0 or 1, i.e., the output of the respective decision-maker functions.

To generate the data, the discrete solution to the HMM example given in example 3.21 is computed on a series of uniformly refined meshes starting with the initial meshes depicted in figure 4.3. Note that the solution to this problem has the property that it is piecewise constant in most parts of the domain. This is also true for the second benchmark problem defined in example 3.22, but in general, this is not the case for the solutions to convection-diffusion-reaction problems. On each refinement level, after the discrete solution has been calculated, the features of LinTriaReco, ConstTriaReco, ConstJumpMod, and ConstJumpNorm and the corresponding labels are stored for every cell. This way, many data can be created fast since the number of cells scales quadratically in two dimensions when the grid is refined. This path is possible because the decision-maker functions act only locally.

To create the data, the diffusion coefficient $\varepsilon = 10^{-8}$ is applied, and the solution is approximated by discontinuous linear finite elements P_1. The same parameters as in

the beginning of section 3.4 are chosen, i.e., $\kappa = \eta = 1$, $\sigma = \varepsilon n_0(p+1)(p+2) = 18\varepsilon$, tol $= 10^{-11}$, $C_0 = 1$, $L = 1.5$,, $u_0 = 1$, $\alpha_{\text{ref}} = 4$, $r = \infty$ and β_{ref} is the arithmetic mean of all β_E.

Rotation invariance of the data

Some of the features that are used in equations (4.4), (4.6), (4.8) and (4.10) are either related to neighbors of the cell or edges in the cell, which is why the features depend on the numbering of the neighbors and edges, respectively. To counteract this dependency, every data point is stored three times, namely for each particular counterclockwise numbering of the edges.

Magnitude invariance of the data

The features of the data set are tuples consisting of several components. Often data sets are scaled or normalized component wise but not feature wise to introduce some sort of magnitude invariance of the data; see, e.g., [AH20] for an example. In other words, the first component of each feature is scaled independently of the other components of the same feature but dependently on the first component of all other features. The same holds for all other components of the features as well. Using this scaling method, the ratio of the components within each particular feature changes. Since the decision-maker functions described above compare the magnitude of the components of each feature with other components of the same feature, inconsistent data can be introduced if the features are scaled component wise. The data might be inconsistent in the sense that the stored label does not fit to the feature anymore. Therefore, in contrast to the previously mentioned reference, the features are not scaled in this work to prevent this inconsistent state.

4.2.2. Restricting the data set

With the above-mentioned recipe, much data can be generated since every cell yields three data points. To be precise, the generated data set consists of 4,456,437 data points that result from refining the regular grid nine times and the irregular one eight times. Unfortunately, there are many duplicates in the data set. They stem from the property of the solution being piecewise constant in huge parts of the domain resulting in equal data points. This might be the case for the data of an individual limiter but also for all limiters simultaneously. These duplicates are removed to prevent the MLP from overfitting the duplicates, i.e., learning a pattern characteristic to these duplicates that do not generalize to unseen data. Therefore, duplicates are removed depending on the concrete numerical examples described below.

Moreover, it can be observed that even after removing the duplicates, there are significantly fewer marked cells than not-marked cells. For instance, after removing the duplicates for the data of LinTriaReco, 77% of the data points have label 0, and

23% possess label 1. If the whole data set is considered, it can be observed that data points where all limiters have agreed not to mark a cell are way more common (93.6%) than points where at least one limiter has marked the cell (6.4%). It is well known that it can be challenging to train MLPs on heavily unbalanced data sets, resulting in worse approximations of the underlying function compared to balanced data sets [Kub21, section 11.2]. To have a better-balanced data set, the set is further restricted. If a single limiter is learned, the number of data points labeled 0 is reduced to equal the number of marked cells in the set. If several limiters are treated simultaneously, the amount of the combination where all limiters do not mark the cell is reduced to the second most occurring combination of labels. Either way, the removed data points are chosen randomly with a fixed random seed to guarantee reproducibility.

4.2.3. Splitting the data set

When applying the previous two steps, the data set becomes smaller but still contains hundreds of thousands of data points. This is useful on the one hand since the more data points exist, the more likely it is to approximate the function lying behind the data. On the other hand, the larger the data set, the longer the optimization takes during the network training. Therefore, the optimal set would consist of as few data points as possible representing all essential features of the function. The previous steps are already an attempt in this direction, i.e., to shrink the data set by dropping data points that bring no new information. Nevertheless, a smaller set might be helpful to speed up the computation further. Therefore, a subset of 7,500 data points is randomly chosen to be the so-called *training data* for the networks.

When MLPs are trained, the networks may fit the training data very well, but they do not generalize well to unseen data; a process which is called overfitting; see, e.g., [GBC16, section 5.3]. One method to prevent overfitting is introducing a so-called *validation set* for which another 1,875 data points are chosen. After every training step during the optimization of the MLPs, the value of the loss functional for the validation set can be computed. During the beginning of the training, the value of the loss functional decreases, as well as the value for the training set. However, at some point, the former value stays constant or even increases while the latter continues to decrease. At this point, the training can be stopped to prevent overfitting.

Finally, the so-called *test set* can be introduced that in this work consists of the validation set and all remaining data points. After the optimization has stopped, the networks are applied to the test set to measure the overall performance of the networks.

4.3. Numerical studies on training multilayer perceptron models

The MLP networks are implemented within the TensorFlow framework [Aba+15; Dev22], an open-source library for deep learning. To open and deploy trained networks in ParMooN, CppFlow [Izq22] is used.

For many problems, the optimal architecture of an MLP to approximate the problem is unknown a-priori. A common way to figure out the architecture is to try various combinations of hyperparameters. The hyperparameters tested in this chapter are given in table 4.1, leading to 360 combinations. A particular choice of hidden layers refers to the number of nodes in the hidden layer, e.g., $[96, 48, 28]$ encodes that the MLP has three hidden layers with 96, 48, and 28 nodes, respectively. The number of nodes in the input and output layer is given by the different tasks and is specified in the examples below. Two different activation functions are tested for the hidden layers, namely ELU and tanh, given in equations (4.2a) and (4.2b). The activation function of the output layer is specified in the concrete experiment. While the biases are initialized with 0, the weights are initialized based on Glorot initialization [GB10]. Fixed random seeds are used to have a deterministic initialization behavior such that the initial weights are reproducible.

During the optimization of the parameters, the minibatch stochastic gradient descent [HH19; GBC16, section 8.1.3] and the Adam [KB14] algorithm are used with TensorFlow's default values except for the learning rate. In the context of deep learning, an *epoch* usually denotes a complete cycle through the whole training data set. The MLPs are trained for at most 10,000 epochs, but the training is stopped earlier if the loss of the validation set has not decreased within the last 100 epochs. Afterwards, the model with the best accuracy is selected.

The numerical experiments strongly follow the experiments in [FHJ22] except for the chosen hyperparameters. In the present work, different hidden layers and initialization seeds are used as well as one learning rate less, and the ELU activation is employed instead of ReLU, cf. [FHJ22, table 1].

4.3.1. Measuring the performance

After the MLPs are trained, their quality has to be assessed. To this extent, the measures,

$$\begin{aligned} \text{accuracy} \qquad & \text{acc} := \frac{t_p + t_n}{t_p + f_p + t_n + f_n}, \\ \text{precision} \qquad & \text{prc} := \frac{t_p}{t_p + f_p}, \\ \text{recall} \qquad & \text{rec} := \frac{t_p}{t_p + f_n} \end{aligned}$$

Table 4.1.: Set of hyperparameters used in section 4.3.2. This gives in total 360 combinations of hyperparameters.

hidden layers	$[96, 48, 28]$, $[55, 55, 55]$, $[96, 48, 28, 14]$, $[45, 45, 45, 45]$, $[96, 48, 28, 14, 7]$, $[40, 40, 40, 40, 40]$
learning rate	0.001, 0.0005, 0.0001, $5 \cdot 10^{-5}$, $1 \cdot 10^{-5}$
batch size	32, 64, 128
activation	ELU, tanh
initialization seed	20, 21

are introduced, where t_p stands for the true positive, t_n denotes the true negative, f_p are the false positive and f_n are the false negative classifications. Accuracy measures how well a network performs over all classes by computing the ratio of correctly classified data to all data. The second measure assesses the ratio of correctly positive classification and all positive classifications. Last but not least, recall investigates the proportion of positives identified as such.

To reduce spurious oscillations, it is worse not to detect a cell that should be marked than marking a cell that should not be. Therefore, recall is considered more important than precision. To combine the above-mentioned measures to a single rating, r_{tot} is introduced, which is defined as

$$r_{\text{tot}} \coloneqq \frac{2}{5}\text{acc} + \frac{1}{5}\text{prc} + \frac{2}{5}\text{rec}. \tag{4.12}$$

Here, acc, prc and rec are computed based on the above mentioned test set.

4.3.2. Approximating the decision-maker functions

Approximating the functions individually

The first experiment investigates whether it is possible with MLPs to approximate the decision-maker functions of LinTriaReco, ConstTriaReco, ConstJumpMod, and ConstJumpNorm from equations (4.5), (4.7), (4.9) and (4.11). To this extent, the data is preprocessed for each limiter as described in sections 4.2.2 and 4.2.3. The input layer has the size of the number of features of the respective limiters, i.e., eight for LinTriaReco and ConstTriaReco and four for ConstJumpMod and ConstJumpNorm. Since all decision-maker functions map to $\{0, 1\}$, only a single node is present in the output layer, and the sigmoid activation function (4.2c) is used in that layer. The binary cross-entropy loss given in equation (4.3) is applied to optimize the parameters. After the MLPs are trained, their performance is measured by equation (4.12) based on the test set.

The results for all configurations are shown in figure 4.4, where each particular configuration of hyperparameters is assigned a unique number from 0 to 359 to

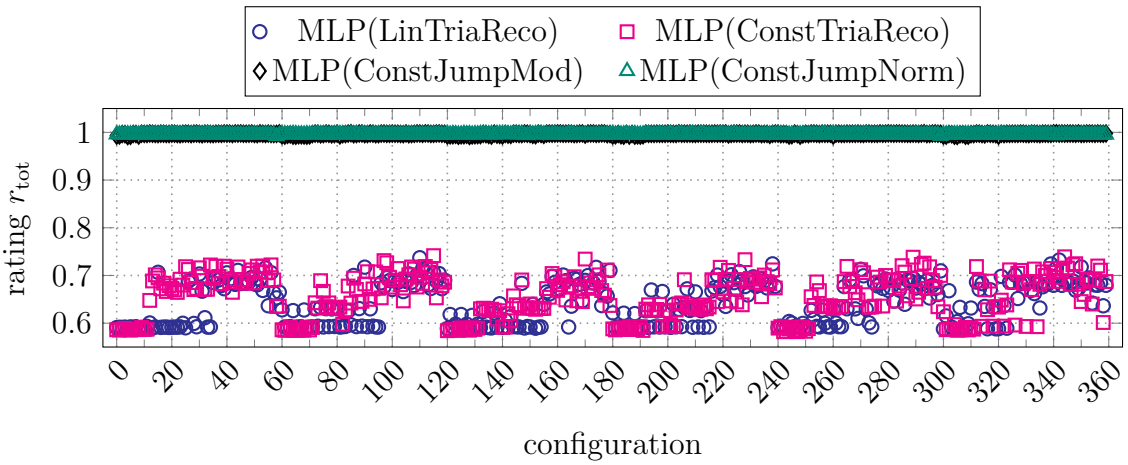


Figure 4.4.: Ratings of all configurations of MLPs after being trained to approximate the decision-maker function of individual limiters. The values of MLP(ConstJumpMod) lie behind the ones of MLP(ConstJumpNorm).

distinguish them. The results for the MLPs that approximate the decision-maker function of LinTriaReco and ConstTriaReco achieve similar total ratings, as well as the MLPs that approximate ConstJumpMod and ConstJumpNorm. Furthermore, the former MLPs have a significantly worse total rating than the latter. The MLPs that approximate LinTriaReco and ConstTriaReco with the best rating have a value of 0.737 and 0.742, resp., compared to the best ratings of the ones that approximate ConstJumpMod (0.998) and ConstJumpNorm (0.997); see also table 4.2. In addition, the mean rating for the latter is significantly higher, and the standard deviation is about a factor of 45 smaller compared to the standard deviations of the former. Hence, it can be concluded that the decision-maker functions of ConstJumpMod and ConstJumpNorm can be approximated better with these configurations than the ones of LinTriaReco and ConstTriaReco, which is in agreement with the findings in [FHJ22, section 4.2]. There is a pattern of hyperparameters for which the MLPs can approximate LinTriaReco and ConstTriaReco better than others. Table 4.3 shows

Table 4.2.: Statistics of total ratings of MLPs that approximate the decision-maker function of individual limiters. LTR, CTR, CJM and CJN are abbreviations for LinTriaReco, ConstTriaReco, ConstJumpMod and ConstJumpNorm, resp., standard deviation is abbreviated by std.

	r_{tot} MLP(LTR)	r_{tot} MLP(CTR)	r_{tot} MLP(CJM)	r_{tot} MLP(CJN)
max	0.737	0.742	0.998	0.997
mean	0.642	0.654	0.996	0.996
std	0.044	0.046	0.001	0.001

Table 4.3.: Pearson correlation coefficients between the hyperparameters and the total ratings of MLPs that approximate individual limiters. LTR and CTR are abbreviations for LinTriaReco and ConstTriaReco. Colors indicate the magnitude of the values.

	r_{tot} MLP(LTR)	r_{tot} MLP(CTR)
hidden layers	0.112	−0.067
learning rate	0.693	0.598
batch size	−0.103	−0.117
activation	0.098	−0.041
initialization seed	−0.052	0.035

the Pearson correlation coefficients between the hyperparameters and the ratings of the MLPs that approximate LinTriaReco and ConstTriaReco. The most significant impact has the learning rate, where the values indicate that the smaller the learning is chosen, the worse the results are, which might explain the pattern in figure 4.4. This effect might be reduced if the MLPs are trained significantly longer than $10,000$ epochs with small learning rates.

Approximating LinTriaReco

This experiment describes overcoming the difficulties approximating the decision-maker function of LinTriaReco. The idea is to train MLPs to take as input all features of all limiters and return the label of LinTriaReco. Therefore, the input layer consists of $n_{\text{LTR}} + n_{\text{CTR}} + n_{\text{CJM}} + n_{\text{CJN}} = 8 + 8 + 4 + 4 = 24$ nodes and the output layer of a single node. Again the activation function in the output layer is chosen to be the sigmoid function, and the binary cross-entropy loss given in equation (4.3) is used during the training. The data that consists of all features of all limiters and the label

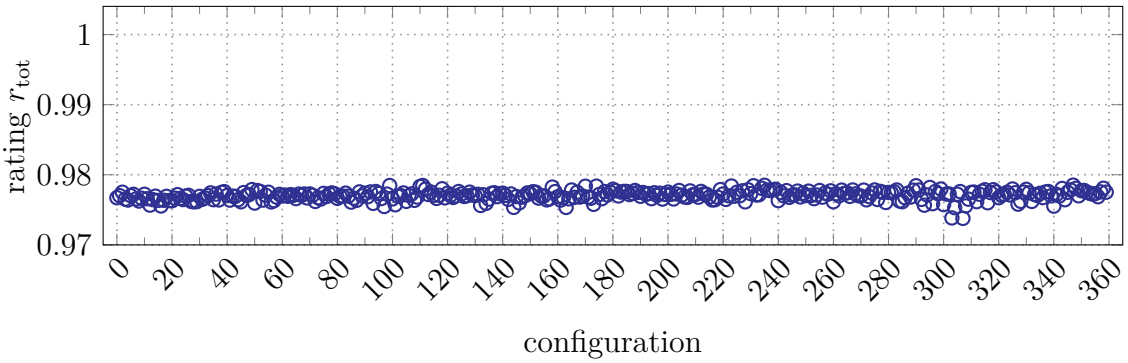


Figure 4.5.: Ratings of all configurations of MLPs after being trained to approximate the mapping from all features of all limiters to the label of LinTriaReco.

Table 4.4.: Statistics of total ratings of MLPs that approximate the mapping from all features of all limiters to the label of LinTriaReco. Standard deviation is abbreviated by std.

	r_{tot} MLP(LinTriaReco)
max	0.979
mean	0.977
std	0.001

of LinTriaReco is preprocessed as described in sections 4.2.2 and 4.2.3 and the total rating defined in equation (4.12) is used to measure the performance of the trained MLPs.

Figure 4.5 shows the results for all configurations of hyperparameters for this experiment. It can be seen that all configurations can predict the label of LinTriaReco significantly better compared to the previous experiment. In this experiment, the best MLP possesses a total rating of around 0.979; see also table 4.4. This is 0.242 better than the best result of an MLP that predicts the label of LinTriaReco from the previous experiment and is comparable to the overall best result. All tested configurations work almost equally well, as indicated by the large value of the mean of total ratings (0.977) and the small standard deviation of around 0.001. This is also in agreement with the results in [FHJ22, section 4.3].

Approximating all decision-maker functions at once

The findings of the previous experiment raise the question of whether it is possible to approximate the decision-maker functions of all investigated limiters at once. To answer this question, MLPs are trained to approximate a mapping from all features of all limiters to all labels simultaneously. This problem hence can also be seen as a

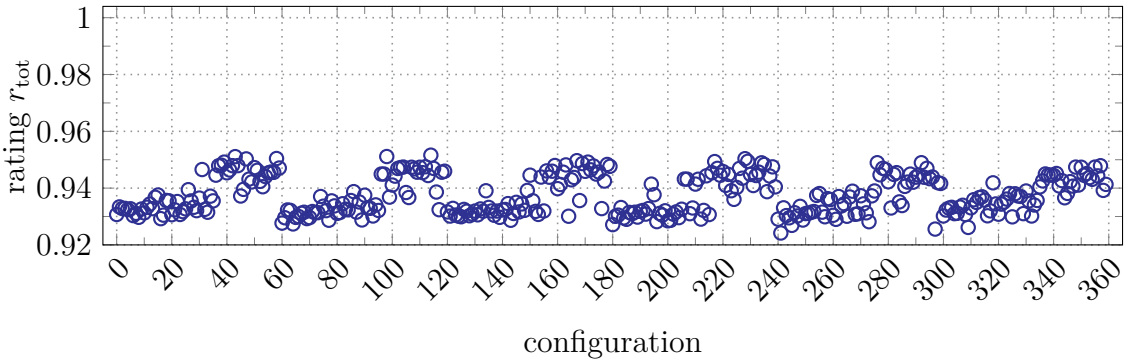


Figure 4.6.: Ratings of all configurations of MLPs after being trained to approximate the mapping from all features to all labels of the limiters at once.

Table 4.5.: Statistics of total ratings of MLPs that approximate the mapping from all features of all limiters to all labels. Standard deviation is abbreviated by std.

	r_{tot}
max	0.952
mean	0.937
std	0.007

multi-label classification task. The size of the input layer is again 24, but in contrast to this experiment, the number of nodes in the output layer is four, i.e., one for each limiters' label. The activation function in the output layer is once more chosen to be the sigmoid function such that the results of the MLPs lie in $[0,1]^4$. An average of the binary cross-entropy loss accounts for four labels being approximated simultaneously. To be precise,

$$\mathcal{L}(\boldsymbol{p};\mathcal{D}) \coloneqq \frac{1}{4}\sum_{j=1}^{4}\left(-\frac{1}{N}\sum_{i=1}^{N} y_{i,j}\log\left(\widehat{y}_{i,j}\right) + (1-y_{i,j})\log\left(1-\widehat{y}_{i,j}\right)\right)$$

is used as loss functional, where N denotes the number of points in the data set $\mathcal{D}$, $y_{i,j}$ is the jth component of the ith label and $\widehat{y}_{i,j}$ is the jth component of the ith prediction of an MLP. This loss functional is nothing but the average of the loss functional from equation (4.3) of all predicted labels. The data is prepared as described in sections 4.2.2 and 4.2.3. The measure in equation (4.12) is used to evaluate the quality of the MLPs, where accuracy, precision, and recall are evaluated element wise and not vector wise for the output of the MLPs.

The results of all tested configurations of hyperparameters are depicted in figure 4.6, which indicates that all configurations achieve a good result. The best MLP possesses a rating of 0.952, and the mean of all ratings is around 0.937, which is a little less than the results of the previous experiment; see also table 4.5. There are differences in the total rating between the configurations as indicated by the standard deviation of 0.007. This can also be seen in figure 4.6 where there is a small pattern of better and worse working configurations again. This is once more also in agreement with the results of [FHJ22, section 4.4]. A possible explanation for the pattern is, as before, that too small learning rates lead to worse results, as also shown by the large Pearson correlation coefficient between the learning rate and r_{tot} in table 4.6. This time the used activation function influences the results moderately as well.

Remark 4.1 (Multi-class approach). It is also possible to reformulate the multi-label problem as a multi-class problem by assigning every unique label combination a unique number. This number can be put in a probability vector, i.e., a vector with positive values summing up to 1, that gives the probability that the outcome belongs to a certain label combination. The results of MLPs that are trained to approximate

this multi-class problem achieve comparable total ratings, cf. [FHJ22, section 4.4] for details. □

4.4. Numerical studies on applying a multilayer perceptron slope limiter

The previous experiments only investigate how well different MLPs can approximate the data, but they have yet to be applied to a problem in practice. This is done in the following for the two benchmark problems given in examples 3.21 and 3.22.

To this end, the best MLP of the last experiment is used to mark cells based on the features of the classical limiters. It has four hidden layers with 45 nodes each and uses the hyperbolic tangent as activation function. This MLP is trained with a learning rate of 0.001 based on a batch size of 64. The initial values of the parameters are based on the initialization seed 20.

The MLP is trained with data that stems from linear discontinuous finite elements. In this section, it is investigated how the MLP performs if it is applied to the discrete solution to the HMM problem obtained with discontinuous finite elements of order $p = 1, 2, 3, 4$.

4.4.1. Applying a multilayer perceptron limiter to the higher-order solution to the HMM problem

To mark a cell, the MLP takes the features of the classical limiters as input and predicts four labels. A cell is marked if at least $n_{\min} \in \{1, 2, 3, 4\}$ of the four predicted labels are true, i.e., larger than 0.5. If a cell is marked, then the solution is locally replaced by the integral mean $\Pi_{\mathrm{MLP},K}(u_h) \coloneqq \overline{u}_{h,K}$, since this reconstruction led to the best results in section 3.4. These limiters are denoted in the experiments below with $\mathrm{MLP}_{n_{\min}}$, where $n_{\min} \in \{1, 2, 3, 4\}$.

Table 4.6.: Pearson correlation coefficients between the hyperparameters and the total ratings of MLPs that approximate all decision-maker functions at once. Colors indicate the magnitude of the values.

	r_{tot}
hidden layers	−0.022
learning rate	0.760
batch size	−0.137
activation	0.236
initialization seed	−0.044

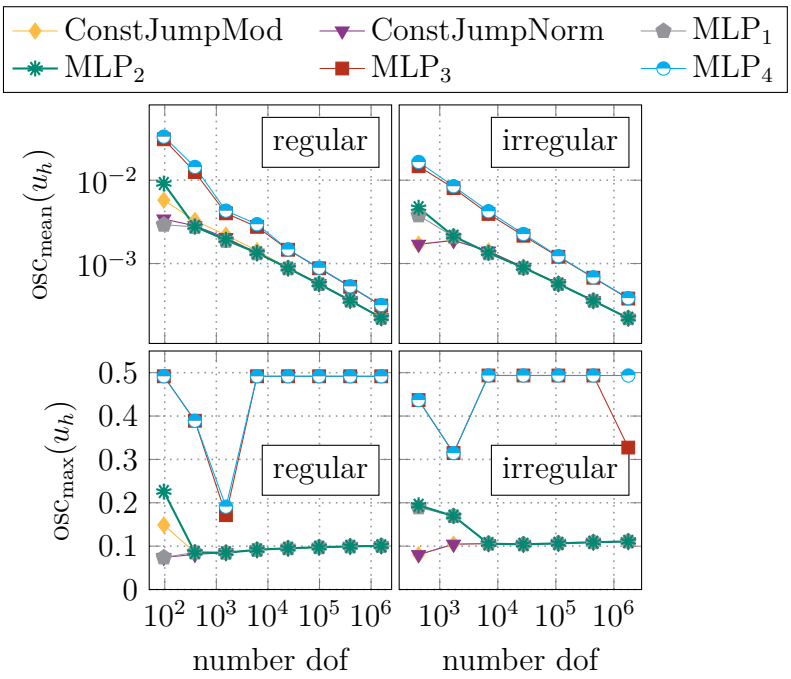


Figure 4.7.: Results for example 3.21 on the regular and irregular grid depicted in figure 4.3 for P_1 finite elements without post-processing and post-processed with ConstJumpNorm and various versions of the MLP limiter.

Determining the minimum number of predicted marks

The first step determines the minimum number of predicted marks $n_{\min}$. Generally, it holds that the smaller $n_{\min}$, the more cells are marked. More marked cells, on the one hand, result in less spurious oscillations, but on the other hand, marking too many cells reduces the order of accuracy of the discrete solution and leads to a computational overhead. Therefore, the number of predicted marks should be small enough to mark as many cells as needed but not too small to lose accuracy. To determine a good value for $n_{\min}$, the MLP limiter with different choices of $n_{\min}$ is applied to the discrete P_1 solution to the HMM problem, i.e., the same problem from which the training data stems; see also section 4.2.1. Hence, it can be expected that the MLP predicts the labels correctly in most cases. The result of the limited solution is compared to the solution post-processing with ConstJumpMod and ConstJumpNorm, which are among the best classical limiters for this problem, as seen in section 3.4. The measures $\mathrm{osc}_{\max}$ and $\mathrm{osc}_{\mathrm{mean}}$ defined in equations (3.4) and (3.5) are used to assess the quality of the discrete solution.

The results of the MLP limiter with various choices for $n_{\min}$ are compared with the solution obtained with the two classical limiters in figure 4.7. It can be seen that MLP_1 and MLP_2 produce similar results to the classical limiters, and MLP_3 and

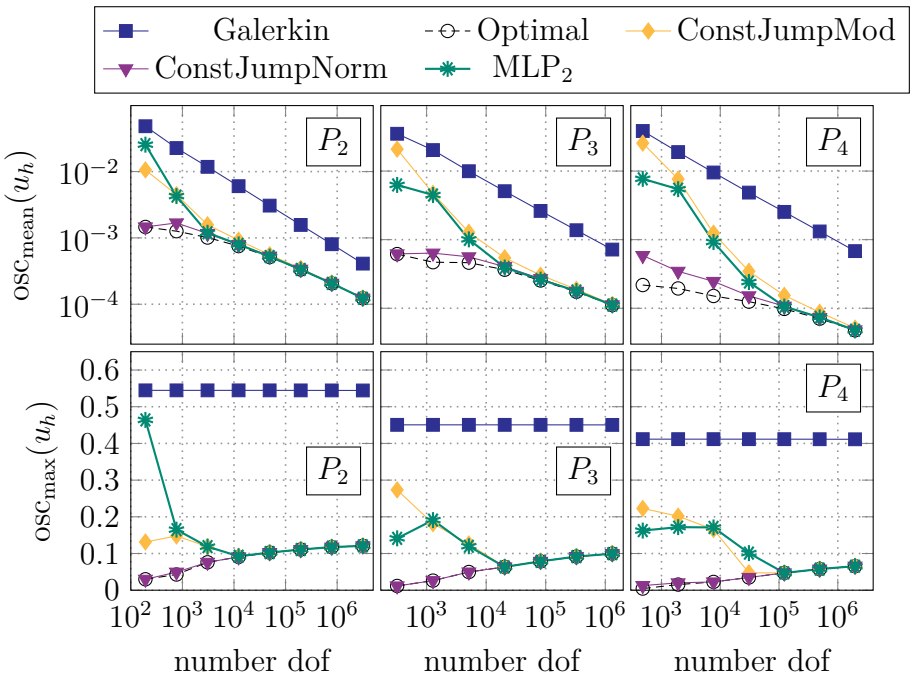


Figure 4.8.: Results of measures of the discrete solution to example 3.21 with various polynomial degrees on the regular grid for various limiters.

MLP_4 perform worse. On coarser grids, MLP_2 works slightly worse than MLP_1, but after a few refinements, they behave the same. This is the case for both measures and both types of meshes. Since slightly worse results on coarser grids can be accepted, MLP_2 can be seen as the best choice between fewer oscillations but not too many marked cells.

Application to the higher-order discrete solution to the HMM problem

After having fixed the minimal number of required marks, the MLP limiter can be applied to the discrete solutions of higher order. Again, example 3.21 is used with the same parameters as in section 4.2.1 except for $\sigma = 3\varepsilon(p+1)(p+2)$ that is chosen in correspondence with the polynomial degree $p = 2, 3, 4$.

The results for the measures $\mathrm{osc}_{\mathrm{mean}}$ and $\mathrm{osc}_{\mathrm{max}}$ for both types of grids are depicted in figures 4.8 and 4.9, respectively. Compared to Galerkin, all limiters significantly reduce the mean and the maximal occurring oscillations on both meshes. Concerning $\mathrm{osc}_{\mathrm{mean}}$ on the regular grid, the MLP limiter works slightly better than ConstJumpMod, is worse than ConstJumpNorm on coarser grids, and just as good as ConstJumpNorm on finer grids. The MLP limiter reduces the maximal occurring oscillations on finer regular grids equally well compared to ConstJumpMod and ConstJumpNorm. On coarser grids $\mathrm{osc}_{\mathrm{max}}$ for the MLP limiter is worse than for ConstJumpMod for P_2 and

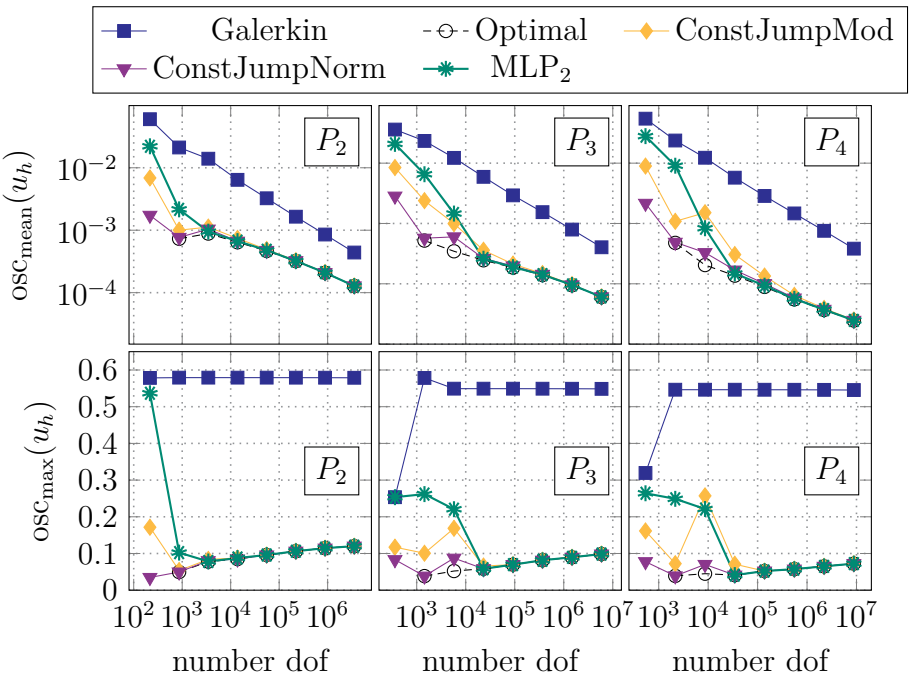


Figure 4.9.: Results of measures of the discrete solution to example 3.21 with various polynomial degrees on the irregular grid for various limiters.

better than for ConstJumpMod for P_3 and P_4. Moreover, it is worse than $\mathrm{osc}_{\mathrm{max}}$ for ConstJumpNorm on all levels. Since the MLP limiter is approaching ConstJumpNorm as the mesh becomes finer, it is clear that it also approaches Optimal. In this sense, the MLP limiter can be seen as optimal on finer meshes.

For the irregular grid, as seen in figure 4.9, again $\mathrm{osc}_{\mathrm{mean}}$ and $\mathrm{osc}_{\mathrm{max}}$ for the MLP limiter are the same as for ConstJumpMod and ConstJumpNorm on finer meshes for all polynomial degrees. Therefore, it can also be considered optimal. The mean oscillations for the P_2 and P_3 solution on coarser meshes of the MLP limiter are slightly worse than those of ConstJumpMod and worse compared to ConstJumpNorm. For P_4, the MLP limiter is better than ConstJumpMod already on medium fine meshes. The maximal occurring oscillations of the MLP limiter are worse than those of the classical limiters except for P_4 on the medium fine meshes, where it is slightly better than ConstJumpMod but still significantly better than the pure Galerkin solution.

Altogether, it can be concluded that the MLP limiter behaves on finer grids equally well compared to its classical companions and is also optimal. Furthermore, it is worse than the best classical one, ConstJumpNorm, on coarser grids. A possible reason for this can be the fact that way more training data on finer levels exists than on coarser ones since the number of cells and hence the number of training data scales quadratically with the refinements. The results are very similar to the findings in [FHJ22, section 4.6.2].

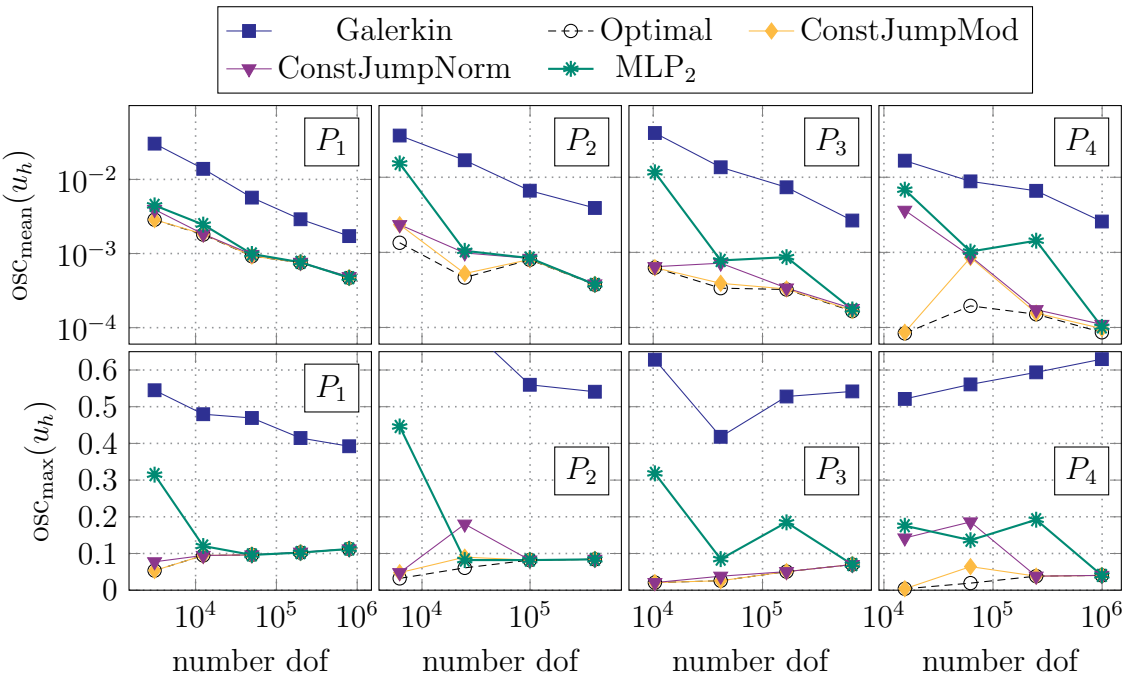


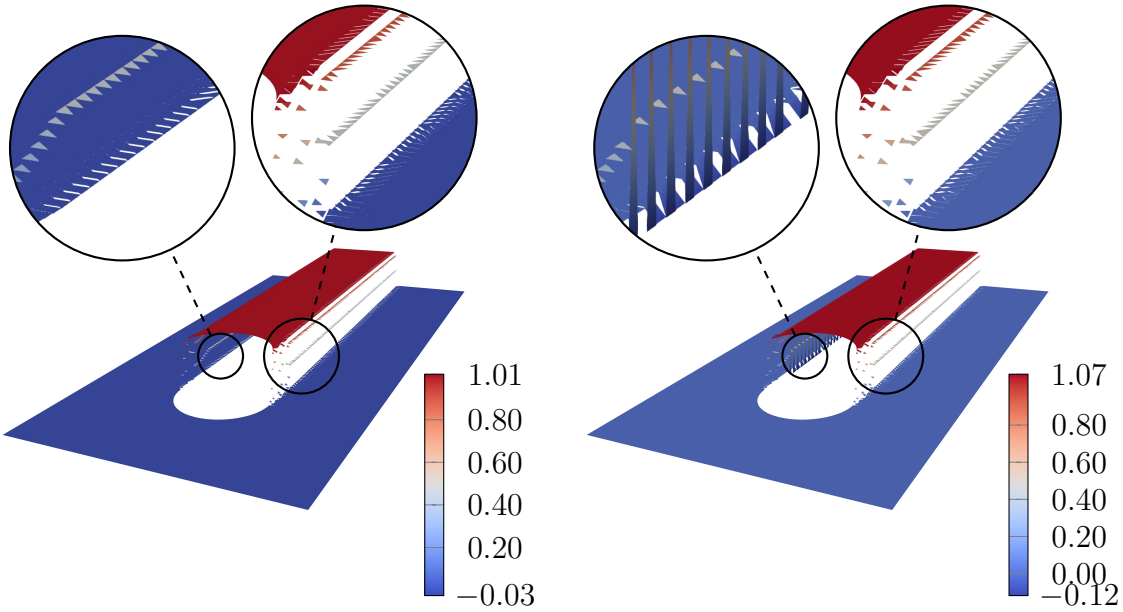
Figure 4.10.: Results of measures of the discrete solution to example 3.22 with various polynomial degrees and for various limiters.

4.4.2. Applying a multilayer perceptron limiter to the discrete solution to the Hemker problem

In the previous section, the MLP limiter is applied to higher-order discrete solutions to the HMM example, i.e., the same example with which the training data was generated. In this section, the limiter is applied to example 3.22, the Hemker problem, to see how it behaves in a setting for which it was not trained. However, the Hemker problem has a similar solution since it is piecewise constant in most parts of the domain and possesses boundary and interior layers.

The parameters are chosen as in the previous experiment, i.e., $\kappa = \eta = 1$, $\sigma = 3\varepsilon(p+1)(p+2)$, where p denotes the polynomial degree, $\mathrm{tol} = 10^{-11}$, $C_0 = 1$, $u_0 = 1$, $\alpha_{\mathrm{ref}} = 4$, $r = \infty$ and β_{ref} is the arithmetic mean of all β_E. The only difference to the previous experiment is that the characteristic length scale of the solution L is set to 13.5. The problem is solved for $p = 1, 2, 3, 4$ on a series of meshes starting from the one in figure 3.17.

The values for $\mathrm{osc}_{\mathrm{mean}}$ and $\mathrm{osc}_{\mathrm{max}}$ for the tested limiters are shown in figure 4.10. Compared to Galerkin, i.e., the discrete solution without any post-processing, ConstJumpMod, ConstJumpNorm, and the MLP limiter reduce both the mean and the maximal oscillations significantly. Concerning the mean oscillations, the MLP limiter



(a) ConstJumpNorm. (b) MLP.

Figure 4.11.: Discrete P_4 solution to example 3.22 limited by the ConstJumpNorm limiter and the MLP limiter on the second finest grid.

usually works worse than the traditional limiters on coarser grids. For P_1, the difference is minimal, and for P_2, the MLP limiter works as well as ConstJumpMod from the second depicted refinement level. For P_3 and P_4, the MLP limiter is only as good as the classical ones on the finest level. However, when it approaches the classical ones, it is also optimal or almost optimal (P_4).

Concerning osc_{max}, it can be noted that the MLP limiter works as well as ConstJumpMod and ConstJumpNorm after at most two depicted refinements (P_1 and P_2) and only on the finest mesh for P_3 and P_4, respectively. On coarser meshes for even polynomial degrees, ConstJumpNorm behaves worse than ConstJumpMod and achieves worse results on some refinement levels than the MLP limiter. Again, the MLP limiter is optimal as early as it reaches ConstJumpNorm.

The P_4 solutions obtained with ConstJumpNorm and the MLP limiter on the second finest grid are depicted in figure 4.11. It can be seen that the MLP limiter does not mark some cells in the interior layer that ConstJumpNorm marks. This results in larger values of osc_{mean} and osc_{max}.

To summarize, the MLP limiter is an alternative to classical approaches for the Hemker example. However, it usually achieves (slightly) worse results, probably because the limiter was trained with data from the HMM example. This is also in agreement with the results in [FHJ22, section 4.7].

4.5. Summary

In this chapter, techniques from deep learning have been used to predict where in the DG solution spurious oscillations occur. To this end, first, it was explained how multilayer perceptron models are constructed and trained to approximate a given mapping. Second, it was shown how data could be created with which MLPs could be trained to mimic classical limiters. The trained limiter was finally applied to the discrete solution to two benchmark problems and compared to the best classical limiters.

Overall, it can be concluded that the MLP limiter is an alternative to classical limiters. The finer the mesh became, the better the limiter became, likely due to the higher amount of training data available on finer meshes than on coarser meshes. To this end, it might be helpful in the future to remove more data points from the finer meshes to have a more balanced data set. On the finest meshes, it was also optimal, in the sense that the spurious oscillations are reduced as much as possible but preserving the mass locally at the same time.

Two advantages of the MLP limiter are that its evaluation is fast and gives the results of four classical limiters simultaneously. However, it still relies on the features and user-dependent parameters of the classical limiters. Therefore, to improve this limiter and exhibit its full potential, other features that represent the solution must be found. It can be investigated whether it is possible to use the local degrees of freedom of the solution as features since, together with the basis functions, they are a complete description of the solution. However, rotation invariance against a different numbering of the degrees of freedom is a severe challenge in that direction. It also needs to be ensured that the basis functions of the underlying DG method are not changed. This makes it less portable to other users compared to more general features such as, for instance, the integral mean.

Closely related to the choice of features is also the question of the type of neural network. In this chapter multilayer perceptron models are used, but it is also possible to use convolutional neural networks as it was done, e.g., in [Bec+20]. These types of networks are primarily used to classify images. A possible way is to use point values of the solution on a cell and possibly on its neighbors as grayscale values representing an image. These can be plugged into a convolutional neural network which might detect structures like steep gradients or spurious oscillations in the solution.

Another research question, which might be the most important, is how to get the labels in the training data set. As long as the labels come from the classical limiters, the best that an MLP limiter can achieve is to be as good as the classical ones. To be better, the labels have to be chosen differently. One option is to define a set of functions on a particular domain with steep gradients and spurious oscillations. These functions have to be interpolated into a finite element space defined on the same domain. Then the labels can be deduced from the definition of the function since the positions of its steep gradients and oscillations are known a-priori.

Furthermore, much improvement is not expected as the MLP limiter was already optimal on the finest levels. Therefore, it can be suggested to find different ways to improve classical finite element solutions with the help of deep learning techniques. One way is to create a neural network that predicts optimal parameters in parameter-dependent finite element methods like SUPG or SOLD. This could be a valuable way to support classical finite element methods.

5. Physics-informed neural networks for convection-diffusion-reaction problems

The previous chapter indicates that techniques from the field of deep learning can support classical finite element methods and enhance the solution to convection-dominated convection-diffusion-reaction problems. However, it is not only possible to apply these techniques to facilitate classical methods but also to use other deep learning techniques instead of classical schemes. One popular approach in recent years is so-called *physics-informed neural networks* (PINNs).

It is not easy to agree on the very first publication that introduced PINNs, but one of the first certainly was [DP94] from 1994. However, their full potential got understood just recently after they got rediscovered in [RPK19] by Raissi, Perdikaris, and Karniadakis. Since then, many contributions and improvements have been made, and they got applied to a wide variety of problems; see, e.g., [Kar+21; Cai+21; Cuo+22] and the references therein for an overview about PINNs and many of their variations.

As said above, PINNs can be seen as another numerical method to approximate the solution to initial-boundary and boundary value problems. This should be possible in principle due to the famous universal approximation theorem. It states that if a feed-forward multilayer perceptron model, as described in section 4.1.1, with linear activation function in the output layer and at least one hidden layer with any bounded non-polynomial activation function possesses enough nodes in the hidden layer, it can approximate with any desired positive accuracy any Borel measurable function that maps from one finite-dimensional space to another [GBC16, p. 194]; see also [Pin99] for an overview. In addition to that, not only the function but also its derivatives up to order $m \in \mathbb{N}$ can be approximated if the function is m times continuously differentiable [Pin99, section 4]. Nowadays, this theorem is extended in the sense that it also holds if a broader class of activation functions are used that, e.g., includes also the ReLU function [GBC16, p. 195]. It is noted that deeper neural networks are usually deployed since they tend to have better generalization properties [GBC16, pp. 195–197]. However, the existence of such a network that approximates a function does not necessarily imply that the usual algorithms converge toward such a network.

In a nutshell, the idea is to approximate the solution to (initial-)boundary value problems with multilayer perceptron models by minimizing a loss functional that usually contains in some form the residual of the governing equations, the boundary, and the initial conditions, and potentially also other underlying physical laws or measured data [Kar+21]. To compute the loss functional so-called *collocation points*

are chosen, i.e., points in the domain and on the (space-time-)boundary, at which the loss functional is calculated. Two of PINNs' unique selling points are that they are usually mesh-free and they can be easily adapted to any initial-boundary value problem, which is why they can cope with various geometries and problems [Cuo+22]. Moreover, they provide a framework to incorporate also (noisy) data from measurements flawlessly, and they can be used to solve inverse problems to discover unknown terms in the governing equations [Kar+21]. Last but not least, they can become really powerful if the same initial-boundary value problem has to be solved for different parameters, e.g., different diffusion coefficients in the case of convection-diffusion-reaction problems. Once a PINN is trained to represent the solution for various parameters, it can also give the solution for a previously unknown parameter almost for free. This is in contrast to many classical methods, for which it is usually challenging to satisfy all these requirements [Kar+21].

However, there are also drawbacks of PINNs. A significant disadvantage is that there is only very few theory about PINNs and their convergence toward the exact solution available; see also [Kar+21; Cuo+22] and the references therein. Therefore, in contrast to classical methods, few guarantees about the approximation quality can be given. This is not only but also because the training contains stochastic elements, which is why it cannot be guaranteed to end at a global minimum. Furthermore, choosing appropriate hyperparameters of the underlying MLPs is still an open problem. Nowadays, they are usually still determined by (automatic) trial-and-error approaches; see, e.g., [YS20].

Despite these open problems, PINNs have been successfully applied to many different problems in various fields, and they usually work well if the solution is in some sense smooth. However, they struggle to approximate perturbed problems [Kri+21]. Moreover, when screening the literature, only some publications deal with PINNs for convection-dominated convection-diffusion-reaction problems with more than one space dimension. In the reference [KZ20], a variational form of a loss functional is presented and tested for time-dependent one- and two-dimensional convection-diffusion-reaction problems with diffusion coefficients between 0.1 and 0.001. The authors test their novel loss functional with several test cases, including one where the solution has an interior layer. They observe that the trained PINNs can capture the behavior of the solution reasonably well [KZ20]. He and Tartakovsky investigate PINNs in the context of time-dependent advection-dispersion problems in one and two dimensions for moderate Peclet numbers of order $\mathcal{O}(1)$, $\mathcal{O}(10)$ and $\mathcal{O}(100)$. They observe that the choice of the weights has a significant impact on the quality of the solution and that the PINN approximation is comparable in accuracy with the SUPG method [HT21]. The results were extended by the group to the case of sharply perturbed initial conditions for which they propose to use a normalized form of the equations and the PINNs, and criteria for how to choose the weights of the loss functionals [ZHT22]. Gomes, Silva, and Valentin report that PINNs for two-dimensional convection-diffusion problems can cope with diffusion coefficients

of around $\mathcal{O}(10^{-1})$ but the quality of the approximation deteriorates drastically for smaller magnitudes. However, their experiments indicate that PINNs are well suited to approximate the parametrized solution in terms of varying diffusion coefficients [GSV22a; GSV22b]. In [ACD23], the authors are inspired by the expansion theory of singularly perturbed boundary layer problems and construct what they call boundary-layer PINNs. To this end, two PINNs are generated, one for the boundary layer and one for the rest of the domain, that are glued together. They test their approximation on one- and two-dimensional convection-dominated convection-diffusion problems and observe that it is able to approximate the solution much better than standard PINNs [ACD23]. However, their approach has twice the computational cost of standard PINNs, and the boundary layer's position must be known a-priori. In a preprint of Wang et al., they observe difficulties for traditional PINNs for convection-dominated convection-diffusion problems in one, two, and three space dimensions. They present a way to choose the weight for the interior loss adaptively, and their numerical results indicate surprisingly that choosing collocations points away from layers works better than increasing the number of points inside the layer [Wan+23]. Other papers, including [dWol+21; Hou+22; Saa+22; MBH23; Lag+23], that deal with convection-diffusion problems usually cover either one-dimensional problems or at most slightly convection-dominated problems.

This chapter extends the knowledge about PINNs of the aforementioned references for convection-dominated convection-diffusion-reaction problems. First, different ways to incorporate the boundary data are treated: a pretraining to approximate the inflow boundary condition in advance and hard-constraining the Dirichlet boundary conditions in general. Furthermore, several new loss functionals for PINNs are proposed and numerically tested. They are mainly based on cost functionals from [JKS11; JK13; KLS19] that are particularly designed for convection-dominated convection-diffusion-reaction problems. In addition, a variational form of PINNs from the literature is presented in the context of convection-diffusion-reaction problems. All ideas are numerically tested using two benchmark problems defined in [JMT97].

The structure of this chapter is as follows: In section 5.1 the basic ideas of physics-informed neural networks are presented, and the loss functional used in the vanilla PINN setting is derived. Section 5.2 presents the various modifications of vanilla PINNs proposed in this work. These modifications are numerically tested with two benchmark problems in section 5.3, and finally, section 5.4 summarizes the content of this chapter

5.1. Basic idea of physics-informed neural networks

To familiarize with physics-informed neural networks, the loss functional used in vanilla PINNs is derived in the following. Afterwards, PINNs are tested for a problem with a smooth solution to verify the implementation.

5.1.1. Derivation of vanilla loss functional

As said in the introduction of this chapter, the idea of PINNs is to use a neural network to approximate the solution u of an initial-boundary value problem directly by a multilayer perceptron model. But how can an MLP learn the unknown solution u? Or in other words, how to choose the parameters $\boldsymbol{p} \subset \mathbb{R}^r$, $r \in \mathbb{N}$, of a neural network $u_{\mathcal{N};\boldsymbol{p}}$ such that

$$u_{\mathcal{N};\boldsymbol{p}} \approx u?$$

In the following the dependency on $\boldsymbol{p}$ is omitted if no confusion can occur, i.e., only $u_{\mathcal{N}}$ is used to refer to the neural network approximation. The parameters $\boldsymbol{p}$ are adapted during the training process, i.e., by minimizing a certain loss functional $\mathcal{L} : \mathbb{R}^r \to \mathbb{R}$ that maps a concrete choice of parameters to a real number; see also section 4.1.2. Alternatively, in mathematical words, given such an $\mathcal{L}$, the goal of the training is to come as close as possible to the optimal parameters

$$\boldsymbol{p}^* \in \operatorname*{arg\,min}_{\boldsymbol{p}\in\mathbb{R}^r} \mathcal{L}(\boldsymbol{p}).$$

If the exact solution u were known, then the loss functional $\mathcal{L}$ could be chosen, e.g., as $1/N \sum_{i=1}^{N} \left(u(\boldsymbol{x}_i) - u_{\mathcal{N}}(\boldsymbol{x}_i)\right)^2$, where N is the number of training points $\boldsymbol{x}_i \in \overline{\Omega}$. However, u is not known in general, and if it were, it would be unnecessary to approximate it. The essence of physics-informed neural networks is choosing a clever $\mathcal{L}$ such that the neural network approximates the unknown u. What is known about u is that it is the solution to a given (initial-)boundary value problem, in this work, the convection-diffusion-reaction problem 2.1. Let

$$\operatorname{Res}(v) \coloneqq -\varepsilon\Delta v + \boldsymbol{b} \cdot \nabla v + cv - f$$

be defined for a sufficiently smooth function $v : \overline{\Omega} \to \mathbb{R}$. Under the assumption that u is the strong solution to problem 2.1, then it holds

$$\|\operatorname{Res}(u)\|^2_{L^2(\Omega)} + \|u - g_{\mathrm{D}}\|^2_{L^2(\Gamma_{\mathrm{D}})} + \|\varepsilon\nabla u \cdot \boldsymbol{n} - g_{\mathrm{N}}\|^2_{L^2(\Gamma_{\mathrm{N}})} = 0. \tag{5.1}$$

In this sense u can be understood as the solution to a corresponding least-squares problem, i.e.,

$$u \in \operatorname*{arg\,min}_{v\in C^2(\Omega)\cap C(\overline{\Omega})} \left(\|\operatorname{Res}(v)\|^2_{L^2(\Omega)} + \|v - g_{\mathrm{D}}\|^2_{L^2(\Gamma_{\mathrm{D}})} + \|\varepsilon\nabla v \cdot \boldsymbol{n} - g_{\mathrm{N}}\|^2_{L^2(\Gamma_{\mathrm{N}})}\right).$$

This serves as an ansatz for the loss functional with which the desired MLP approximation $u_{\mathcal{N}}$ can be optimized. In other words, $u_{\mathcal{N}}$ is the MLP such that its parameters $\boldsymbol{p}^*$ are given by

$$\boldsymbol{p}^* \in \operatorname*{arg\,min}_{\boldsymbol{p}\in\mathbb{R}^r} \left(\|\operatorname{Res}(u_{\mathcal{N};\boldsymbol{p}})\|^2_{L^2(\Omega)} + \|u_{\mathcal{N};\boldsymbol{p}} - g_{\mathrm{D}}\|^2_{L^2(\Gamma_{\mathrm{D}})} + \|\varepsilon\nabla u_{\mathcal{N};\boldsymbol{p}} \cdot \boldsymbol{n} - g_{\mathrm{N}}\|^2_{L^2(\Gamma_{\mathrm{N}})}\right). \tag{5.2}$$

The right-hand side of the previous equation is well defined almost everywhere if the activation functions $\boldsymbol{\sigma}^{[i]}$, $i = 1, 2, \cdots, \ell$, used in all ℓ layers are twice differentiable almost everywhere.

The integrals encoded by the norms in the right-hand side of equation (5.2) might be extremely difficult to compute exactly, so they are approximated by some numerical quadrature formula. In vanilla PINNs the Monte Carlo integration method[1] is used to approximate these integrals. In principle, other numerical integration techniques, such as Gauss–Legendre integration, would also be possible. However, Monte Carlo integration has the advantage that no underlying partition of the domain is needed, and it does not suffer from an exponential number of evaluations of the integrands. To this end, $N_\mathrm{I} \in \mathbb{N}$ points $\boldsymbol{x}_{i,\mathrm{I}} \in \Omega$, $i = 1, 2, \dots N_\mathrm{I}$, inside the domain Ω, $N_\mathrm{D} \in \mathbb{N}$ points $\boldsymbol{x}_{i,\mathrm{D}} \in \Gamma_\mathrm{D}$, $i = 1, 2, \dots N_\mathrm{D}$, along the boundary Γ_D, and $N_\mathrm{N} \in \mathbb{N}$ points $\boldsymbol{x}_{i,\mathrm{N}} \in \Gamma_\mathrm{N}$, $i = 1, 2, \dots N_\mathrm{N}$, along the boundary Γ_N are (randomly) chosen. Then, the terms that appear on the right-hand side of equation (5.2) can be approximated by

$$\| \operatorname{Res}(u_\mathcal{N}) \|^2_{L^2(\Omega)} \approx \frac{|\Omega|}{N_\mathrm{I}} \sum_{i=1}^{N_\mathrm{I}} \left(\operatorname{Res}(u_\mathcal{N})(\boldsymbol{x}_{i,\mathrm{I}}) \right)^2 =: \mathcal{L}^{\mathrm{st}}_\mathrm{I}, \tag{5.3a}$$

$$\| u_\mathcal{N} - g_\mathrm{D} \|^2_{L^2(\Gamma_\mathrm{D})} \approx \frac{|\Gamma_\mathrm{D}|}{N_\mathrm{D}} \sum_{i=1}^{N_\mathrm{D}} \left(u_\mathcal{N}\left(\boldsymbol{x}_{i,\mathrm{D}}\right) - g_\mathrm{D}\left(\boldsymbol{x}_{i,\mathrm{D}}\right) \right)^2 =: \mathcal{L}^{\mathrm{st}}_\mathrm{D}, \tag{5.3b}$$

$$\| \varepsilon \nabla u_\mathcal{N} \cdot \boldsymbol{n} - g_\mathrm{N} \|^2_{L^2(\Gamma_\mathrm{N})} \approx \frac{|\Gamma_\mathrm{N}|}{N_\mathrm{N}} \sum_{i=1}^{N_\mathrm{N}} \left(\varepsilon \nabla u_\mathcal{N}\left(\boldsymbol{x}_{i,\mathrm{N}}\right) \cdot \boldsymbol{n}\left(\boldsymbol{x}_{i,\mathrm{N}}\right) - g_\mathrm{N}\left(\boldsymbol{x}_{i,\mathrm{N}}\right) \right)^2 =: \mathcal{L}^{\mathrm{st}}_\mathrm{N}. \tag{5.3c}$$

Finally, let $\alpha^{\mathrm{st}}_\mathrm{I}, \alpha^{\mathrm{st}}_\mathrm{D}, \alpha^{\mathrm{st}}_\mathrm{N} \in \mathbb{R}$ be three non-negative weights. Then the loss functional $\mathcal{L}^{\mathrm{st}}$ for vanilla PINNs is given by

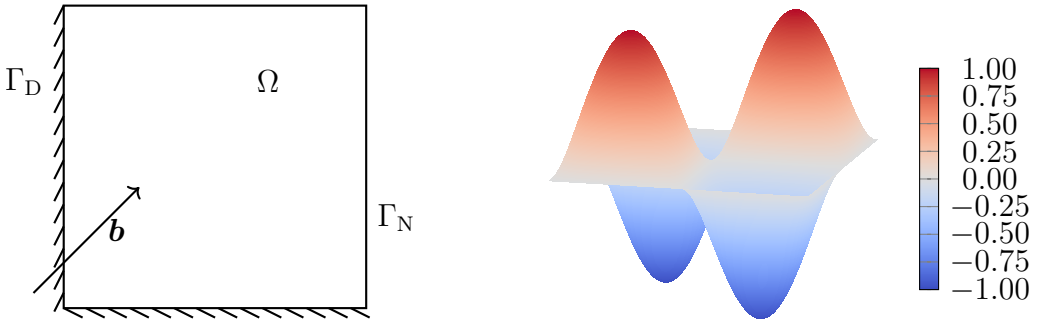
$$\mathcal{L}^{\mathrm{st}} \coloneqq \alpha^{\mathrm{st}}_\mathrm{I} \mathcal{L}^{\mathrm{st}}_\mathrm{I} + \alpha^{\mathrm{st}}_\mathrm{D} \mathcal{L}^{\mathrm{st}}_\mathrm{D} + \alpha^{\mathrm{st}}_\mathrm{N} \mathcal{L}^{\mathrm{st}}_\mathrm{N} \tag{5.4}$$

and during the training the parameters of the MLP are optimized such that they minimize $\mathcal{L}^{\mathrm{st}}$; see also section 4.1.2. In practice, this can be done, e.g., by using a minibatch stochastic gradient descent method [HH19; GBC16, section 8.1.3], but also other optimization methods are possible. Last but not least, it can be emphasized that the measures of the domain and of the boundaries, such as the number of points that are used in the single contributions defined in equations (5.3a) to (5.3c) are not hidden in the weights $\alpha^{\mathrm{st}}_\mathrm{I}, \alpha^{\mathrm{st}}_\mathrm{D}, \alpha^{\mathrm{st}}_\mathrm{N}$. However they still contribute to the loss functional and must not be forgotten in what follows.

5.1.2. Numerical experiment with a smooth known solution

To see how PINNs work in practice and to verify the implementation used in this section, a PINN approximation for a convection-diffusion-reaction problem is computed

[1] See, e.g., [RC04, chapter 3.2].



(a) Domain and convection field. (b) Exact solution.

Figure 5.1.: Domain, direction of convection field and exact solution to example 5.1. Lines with ticks indicate Dirichlet boundary Γ_D and lines without ticks indicate Neumann boundary Γ_N.

and compared to the exact known solution.

To this end, the following example is defined that is an adapted version of example 2.37.

Example 5.1 (Adapted Sine Laplace problem in 2D)**.** Let $\Omega := (0,1)^2$ be the unit square, $\varepsilon := 10^{-8}$, $\boldsymbol{b} := (1,1)^T$ and $c := 1$. The right and the top edge of the unit square are chosen to be the Neumann boundary Γ_N, while on the left and the bottom edge Dirichlet boundary conditions are prescribed; see figure 5.1a. The right-hand side and the boundary conditions are chosen such that

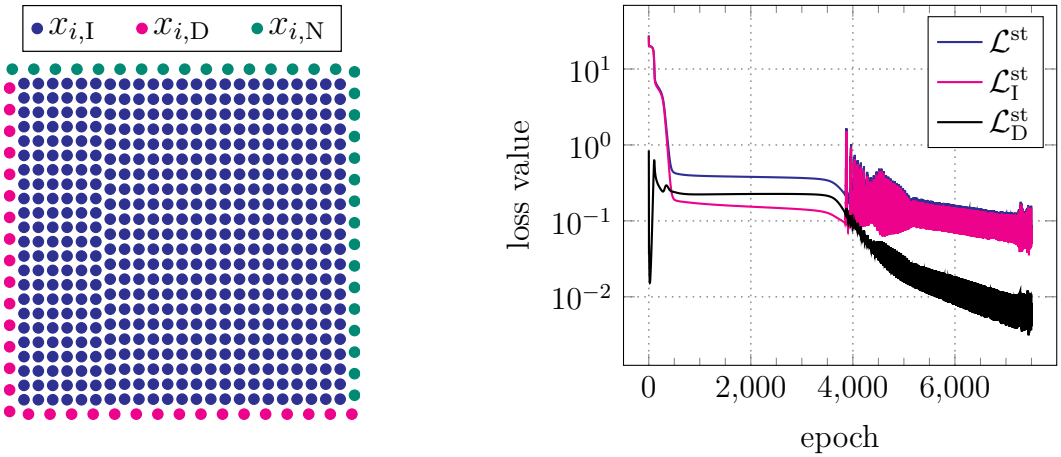
$$u(x,y) := \sin(2\pi x)\sin(2\pi y)$$

is the exact solution which is visualized in figure 5.1b. □

Note that this experiment is conducted in the convection-dominated regime since ε is set to be 10^{-8}.

Implementation aspects

The implementation that is used in this section as well as in section 5.3 is based on the open-source library TensorFlow [Aba+15; Dev22]. For the optimization step, the minibatch stochastic gradient descent [HH19; GBC16, section 8.1.3] together with the Adam algorithm [KB14] is deployed where TensorFlow's default values are chosen except for the learning rate that is defined individually for each numerical example. After the training, the MLP with the smallest loss value during the optimization is returned as the final approximation. Note that this is not necessarily the MLP after the last optimization step due to the stochastic gradient descent method.



(a) Training points.

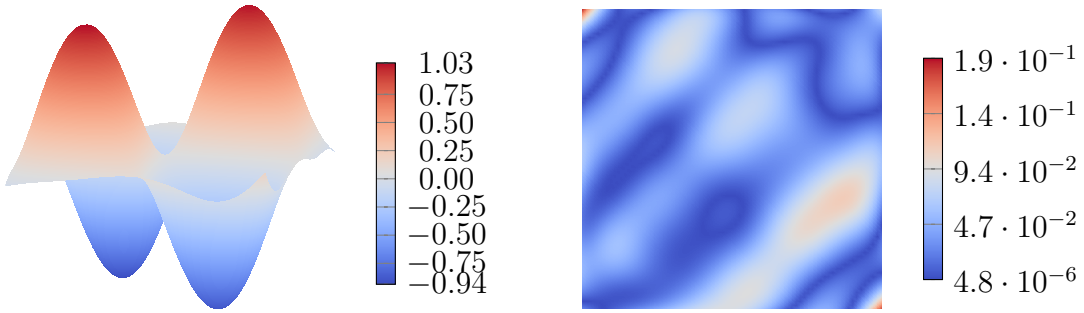
(b) Training history.

Figure 5.2.: Training points and training history for a PINN approximation of the solution to example 5.1. The loss $\mathcal{L}^{\mathrm{st}}$ is the sum of the interior, Dirichlet and Neumann contributions. The contribution of $\mathcal{L}^{\mathrm{st}}_{\mathrm{N}}$ is not shown since it has values below 10^{-14}.

Results

The solution is approximated by an MLP that consists of an input layer with two nodes, eight hidden layers with 20 nodes each, and an output layer with a single node. Except for the last layer where a linear activation function is used, all other activation functions are chosen as the tanh function given in equation (4.2b). The weights of the MLP are initialized based on the Glorot initialization [GB10] with random seed 41, and the initial biases are set to 0. During the training, the MLP seeks to minimize the loss functional given in equation (5.4), in which the weights $\alpha^{\mathrm{st}}_{\mathrm{I}} = \alpha^{\mathrm{st}}_{\mathrm{D}} = \alpha^{\mathrm{st}}_{\mathrm{N}} = 1$ are used. The MLP is trained for $7,500$ epochs with a learning rate of 10^{-4}, and the integrals are approximated using 320 equally distanced boundary points and 512 equally distanced interior points. Half of the boundary points lie on the Dirichlet boundary and the other half on the Neumann boundary; see also figure 5.2a.

The values of the loss functional and its components during the training process are depicted in figure 5.2b. The loss term concerning the Neumann boundary is not shown since its values are always below 10^{-14} during the training. It can be seen that over the whole training, the loss decreases. The loss starts to oscillate for higher numbers of epochs and stops decreasing monotonically due to the minibatch stochastic gradient descent method used. The final PINN approximation $u_{\mathcal{N}}$ and its point wise error $|u - u_{\mathcal{N}}|$ are shown in figure 5.3. It can be seen that the MLP is a reasonable approximation to the solution, even though the maximum and the minimum are not perfectly met. Furthermore, it is notable that the Dirichlet boundary conditions are not satisfied exactly, which stems from the fact that the boundary conditions



(a) PINN approximation. (b) $|u - u_{\mathcal{N}}|$.

Figure 5.3.: A PINN approximation $u_{\mathcal{N}}$ of solution u to example 5.1 and point wise error $|u - u_{\mathcal{N}}|$.

are part of the loss functional and must be learned during the training. This is in contrast to continuous Galerkin finite element methods, where the Dirichlet boundary conditions are usually exactly satisfied. Compared to the exact solution, the PINN approximation has an L^2-error of about $\|u - u_{\mathcal{N}}\|_{L^2(\Omega)} \approx 0.047$ and a H^1-semi norm error of about $|u - u_{\mathcal{N}}|_{H^1(\Omega)} \approx 0.498$. However, the approximation probably becomes more accurate if more training points are chosen, or the MLP is trained for more epochs since after $7,500$ epochs, the loss is still decreasing.

Altogether, it can be concluded that the implementation seems correct, and PINNs are, in general, able to approximate the solution to convection-dominated convection-diffusion-reaction problems. However, it is already known in the literature that they might terribly fail when it comes to solutions that possess layers [Kri+21; GSV22a].

5.2. Modifications of vanilla physics-informed neural networks

In this section, several modifications of vanilla PINNs concerning treating Dirichlet boundary conditions and the interior loss functional are introduced.

5.2.1. Pretraining and hard-constrained physics-informed neural networks

Boundary conditions are an essential part of the convection-diffusion-reaction problem. Not only do they affect the well-posedness of the problem, but they also influence the solution in the interior, especially the inflow part of the boundary conditions. Therefore, it seems reasonable to emphasize the inflow boundary conditions or to impose the Dirichlet boundary conditions exactly. In the following, two ideas are presented on how to achieve this.

Pretraining of inflow boundary conditions

Since the inflow boundary conditions influence the solution in the interior significantly, it might be helpful to start the optimization of the MLPs with an initialization that satisfies the boundary conditions along Γ^-. This hopefully serves as a reasonable guess such that during the actual optimization of the MLP, the chances are higher to end in a global instead of a local minimum.

Unfortunately, it is almost impossible to prescribe the weights and the biases of an MLP such that it satisfies the inflow boundary conditions exactly. However, they can be learned in advance. Therefore, the idea of this section is to add another optimization loop to the training process before the actual optimization occurs. This new optimization loop is called pretraining in what follows.

Assume that the Dirichlet training points $\boldsymbol{x}_{i,\mathrm{D}}$ are given. For the pretraining optimization, the subset of points are used that lie on the inflow part of the boundary, i.e., all points $\boldsymbol{x}_{i,\mathrm{D}}$ such that $\boldsymbol{b}(\boldsymbol{x}_{i,\mathrm{D}}) \cdot \boldsymbol{n}(\boldsymbol{x}_{i,\mathrm{D}}) < 0$. During the pretraining, the MLPs are optimized with respect to minimizing the loss $\mathcal{L}^{\mathrm{pr}} \coloneqq \mathcal{L}_{\mathrm{D}}$ given the inflow boundary points, i.e., only the Dirichlet term of the complete loss $\mathcal{L}^{\mathrm{st}}$.

After this pretraining optimization has ended, the actual optimization can start with an initial guess that satisfies approximately the inflow boundary conditions.

Hard-constrained PINNs

The above-mentioned algorithm still has the property that the Dirichlet boundary conditions are only approximately satisfied and must be learned during training. Nevertheless, since this part of the solution is known, why does not the MLP approximation satisfy these values exactly, and why is computing time spent to learn an a-priori known pattern? In the context of continuous Galerkin finite element methods, the Dirichlet boundary conditions are usually imposed exactly. Inspired by this, the idea of hard-constrained PINNs as introduced by Lu et al. [Lu+21] is to satisfy the Dirichlet boundary conditions exactly by defining

$$\widetilde{u}_{\mathcal{N}} \coloneqq \widetilde{g}_{\mathrm{D}} + l u_{\mathcal{N}} \tag{5.5}$$

to be the modified PINN approximation. Here $\widetilde{g}_{\mathrm{D}} : \overline{\Omega} \to \mathbb{R}$ is a continuous extension of the Dirichlet boundary condition g_{D} from Γ_{D} to Ω and $l : \overline{\Omega} \to \mathbb{R}$ is a function that satisfies

$$\begin{cases} l(\boldsymbol{x}) = 0, & \text{if } \boldsymbol{x} \in \Gamma_{\mathrm{D}}, \\ l(\boldsymbol{x}) > 0, & \text{if } \boldsymbol{x} \in \Omega. \end{cases}$$

By construction it holds $\widetilde{u}_{\mathcal{N}}|_{\Gamma_{\mathrm{D}}} = g_{\mathrm{D}}$, i.e., the Dirichlet boundary conditions are exactly satisfied.

In simple cases, the function $\widetilde{g}_{\mathrm{D}}$ might be constructed explicitly. Alternatively, it is also possible to train another MLP $\widetilde{g}_{\mathrm{D},\mathcal{N}}$ by minimizing, e.g.,

$$\frac{1}{N_{\mathrm{I}}}\sum_{i=1}^{N_{\mathrm{I}}}\left(\widetilde{g}_{\mathrm{D},\mathcal{N}}\left(\boldsymbol{x}_{i,\mathrm{I}}\right)\right)^2+\frac{1}{N_{\mathrm{D}}}\sum_{i=1}^{N_{\mathrm{D}}}\left(\widetilde{g}_{\mathrm{D},\mathcal{N}}\left(\boldsymbol{x}_{i,\mathrm{D}}\right)-g_{\mathrm{D}}\left(\boldsymbol{x}_{i,\mathrm{I}}\right)\right)^2+\frac{1}{N_{\mathrm{N}}}\sum_{i=1}^{N_{\mathrm{N}}}\left(\widetilde{g}_{\mathrm{D},\mathcal{N}}\left(\boldsymbol{x}_{i,\mathrm{N}}\right)\right)^2.$$

While with the help of the first and the last term, the MLP is trained to satisfy $\widetilde{g}_{\mathrm{D},\mathcal{N}}|_{\Omega\cup\Gamma_{\mathrm{N}}}\approx 0$, the remaining term is present to enforce that $\widetilde{g}_{\mathrm{D},\mathcal{N}}|_{\Gamma_{\mathrm{D}}}\approx g_{\mathrm{D}}$. Again, if the boundary Γ_{D} has a simple form, also l might be constructed explicitly. Else, it might be approximated using splines [Lu+21].

Alternatively, inspired by the proof of theorem 2.3, given such a $\widetilde{g}_{\mathrm{D}}$, it is also possible to optimize $lu_{\mathcal{N}}$ to be the solution to the modified convection-diffusion-reaction problem with homogeneous Dirichlet boundary conditions: Find a sufficiently smooth u such that

$$\begin{aligned}
-\varepsilon\Delta u+\boldsymbol{b}\cdot\nabla u+cu &= f-(-\varepsilon\Delta\widetilde{g}_{\mathrm{D}}+\boldsymbol{b}\cdot\nabla\widetilde{g}_{\mathrm{D}}+c\widetilde{g}_{\mathrm{D}}) && \text{in } \Omega,\\
u &= 0 && \text{along } \Gamma_{\mathrm{D}},\\
\varepsilon\nabla u\cdot\boldsymbol{n} &= g_{\mathrm{N}}-(\varepsilon\nabla\widetilde{g}_{\mathrm{D}}\cdot\boldsymbol{n}) && \text{along } \Gamma_{\mathrm{N}}.
\end{aligned}$$

Since $l|_{\Gamma_{\mathrm{D}}}=0$, the MLP approximation $lu_{\mathcal{N}}$ satisfies the homogeneous Dirichlet boundary conditions exactly, and after the optimization $\widetilde{g}_{\mathrm{D}}+lu_{\mathcal{N}}$ can be used as an approximation to u.

5.2.2. Non-standard loss functionals

In [JKS11; JK13; KLS19], several cost functionals are investigated to optimize the SUPG solution with respect to the SUPG parameter. It is observed that the cost functional consisting only of the strong form of the residual works worse than other tested functionals [JKS11; JK13]. Therefore, another idea is to adapt them to work within the PINN framework. Moreover, it can be emphasized that all functionals presented in this section have never been considered in the context of PINNs before and are numerically investigated in this framework in section 5.3.2 for the first time.

Crosswind loss

The first variant of the loss functional is based on the loss functional proposed by John, Knobloch, and Savescu [JKS11] in equation (22). Guided by their ideas,

$$\|\operatorname{Res}(u_{\mathcal{N}})\|_{L^2(\Omega)}^2+\|\phi(|\boldsymbol{b}^{\perp}\cdot\nabla u_{\mathcal{N}}|)\|_{L^1(\Omega)}$$

might serve as an ansatz for the interior part of the loss functional, where the functions

$$\boldsymbol{b}^{\perp}(\boldsymbol{x}):=\begin{cases}\frac{(b_2(\boldsymbol{x}),-b_1(\boldsymbol{x}))^T}{|\boldsymbol{b}(\boldsymbol{x})|}, & \text{if } \boldsymbol{b}(\boldsymbol{x})\neq 0,\\ 0, & \text{else,}\end{cases}$$

and

$$\phi(x) \coloneqq \begin{cases} \sqrt{x}, & \text{if } x \geq 1, \\ 0.5(5x^2 - 3x^3), & \text{else,} \end{cases}$$

are employed. The integrals can again be approximated by the Monte Carlo integration method, i.e.,

$$\begin{aligned} \| \operatorname{Res}(u_{\mathcal{N}}) \|_{L^2(\Omega)}^2 + \| \phi(|\boldsymbol{b}^\perp \cdot \nabla u_{\mathcal{N}}|) \|_{L^1(\Omega)} \approx \\ \mathcal{L}_{\mathrm{I}}^{\mathrm{st}} + \frac{|\Omega|}{N_{\mathrm{I}}} \sum_{i=1}^{N_{\mathrm{I}}} |\phi(|\boldsymbol{b}^\perp(\boldsymbol{x}_{i,\mathrm{I}}) \cdot \nabla u_{\mathcal{N}}(\boldsymbol{x}_{i,\mathrm{I}})|)| =: \mathcal{L}_{\mathrm{I}}^{\mathrm{cw}}. \end{aligned}$$

Finally, the crosswind loss can be defined as

$$\mathcal{L}^{\mathrm{cw}} \coloneqq \alpha_{\mathrm{I}}^{\mathrm{cw}} \mathcal{L}_{\mathrm{I}}^{\mathrm{cw}} + \alpha_{\mathrm{D}}^{\mathrm{cw}} \mathcal{L}_{\mathrm{D}}^{\mathrm{st}} + \alpha_{\mathrm{N}}^{\mathrm{cw}} \mathcal{L}_{\mathrm{N}}^{\mathrm{st}}, \tag{5.6}$$

where $\alpha_{\mathrm{I}}^{\mathrm{cw}}, \alpha_{\mathrm{D}}^{\mathrm{cw}}, \alpha_{\mathrm{N}}^{\mathrm{cw}} \in \mathbb{R}$ are three non-negative weights.

Remark 5.2. John, Knobloch, and Savescu [JKS11] argue that it is important to exclude cells from the computation of their loss functionals that share an edge with the Dirichlet boundary, since in those cells the strong residual cannot be reduced significantly. This also has to be kept in mind in the context of PINNs. Therefore, it might be advantageous to use only interior points that have a distance to the Dirichlet boundary of at least $C\sqrt{\varepsilon}$, where $C \in \mathbb{R}$ is a positive factor, e.g., $C = 20$. □

Limited residual loss

The next variant of the loss functional is related to the loss functional I_h^{lim} of Knobloch, Lukáš, and Solin [KLS19]. Their functional is based on a triangulation of the domain since it is proposed for finite element computations. To derive a loss functional for PINNs, it is assumed that $N_{\mathrm{I}} \in \mathbb{N}$ interior points $\boldsymbol{x}_{i,\mathrm{I}} \in \Omega$, $i = 1, 2, \ldots N_{\mathrm{I}}$, are given. From these points a Voronoi tessellation[2] can be constructed by defining the mesh cells

$$K_i \coloneqq \{\, \boldsymbol{y} \in \overline{\Omega} \;:\; |\boldsymbol{y} - \boldsymbol{x}_{i,\mathrm{I}}| < |\boldsymbol{y} - \boldsymbol{x}_{j,\mathrm{I}}| \text{ for all } j = 1, 2, \ldots, N_{\mathrm{I}} \,\};$$

see also figure 5.4. These cells clearly contain $\boldsymbol{x}_{i,\mathrm{I}}$ but no other $\boldsymbol{x}_{j,\mathrm{I}}$, $j = 1, 2, \ldots, N_{\mathrm{I}}$, $j \neq i$. In this sense, every cell K_i can be identified by a corresponding $\boldsymbol{x}_{i,\mathrm{I}}$.

Using this triangulation $\mathcal{T}_h \coloneqq \{\, K_i \;:\; i = 1, 2, \ldots, N_{\mathrm{I}} \,\}$, the quantity

$$\sum_{K_i \in \mathcal{T}_h} \psi\left(\| \operatorname{Res}(u_{\mathcal{N}}) \|_{L^2(K_i)}^2 \right) \tag{5.7}$$

[2] See, e.g., [Pip17, chapter 50.6.1].

Figure 5.4.: Voronoi tessellations of the unit square with respect to five and nine interior points resulting in five and nine cells, respectively.

may serve as an ansatz for the interior part of the loss functional, where for a fixed non-negative $t_0 \in \mathbb{R}$ and a given $x \in \mathbb{R}$

$$\psi(x) \coloneqq \xi\left(\frac{x}{t_0}\right)$$

with

$$\xi(x) \coloneqq \begin{cases} \frac{1}{2}x^4 - x^3 - \frac{1}{2}x^2 + 2x, & \text{if } x \leq 1, \\ 1, & \text{else,} \end{cases}$$

is defined, cf. also I_h^{lim} in [KLS19]. Cell wise the integrals can be approximated by

$$\|\operatorname{Res}(u_{\mathcal{N}})\|^2_{L^2(K_i)} \approx \psi\left(|K_i|\left(\operatorname{Res}(u_{\mathcal{N}})(\boldsymbol{x}_{i,\mathrm{I}})\right)^2\right) \approx \psi\left(\frac{|\Omega|}{N_{\mathrm{I}}}\left(\operatorname{Res}(u_{\mathcal{N}})(\boldsymbol{x}_{i,\mathrm{I}})\right)^2\right), \tag{5.8}$$

where in the last step $|K_i| \approx |\Omega|/N_{\mathrm{I}}$ is used. The last approximation is exact if the interior points are equally distanced and becomes worse the more the smallest and the largest value of the distance between the points and their closest neighbors and boundaries differ.

Combining equations (5.7) and (5.8), identifying each cell with its corresponding interior point and using the definition of ψ leads to

$$\sum_{i=1}^{N_{\mathrm{I}}} \xi\left(\frac{|\Omega|}{N_{\mathrm{I}}\, t_0}\left(\operatorname{Res}(u_{\mathcal{N}})(x_{i,\mathrm{I}})\right)^2\right) =: \mathcal{L}_{\mathrm{I}}^{\mathrm{lr}}.$$

Finally, let $\alpha_{\mathrm{I}}^{\mathrm{lr}}, \alpha_{\mathrm{D}}^{\mathrm{lr}}, \alpha_{\mathrm{N}}^{\mathrm{lr}} \in \mathbb{R}$ again be three non-negative weights. Then, the limited residual loss is defined as

$$\mathcal{L}^{\mathrm{lr}} \coloneqq \alpha_{\mathrm{I}}^{\mathrm{lr}}\mathcal{L}_{\mathrm{I}}^{\mathrm{lr}} + \alpha_{\mathrm{D}}^{\mathrm{lr}}\mathcal{L}_{\mathrm{D}}^{\mathrm{st}} + \alpha_{\mathrm{N}}^{\mathrm{lr}}\mathcal{L}_{\mathrm{N}}^{\mathrm{st}}. \tag{5.9}$$

Remark 5.3. Also Knobloch, Lukáš, and Solin [KLS19] exclude boundary cells from the computation of the residual. However, with the same argument as in remark 5.2, it is proposed to use only interior points that are further away from the boundary than $C\sqrt{\varepsilon}$, and, e.g., $C = 20$. □

Limited residual and crosswind loss

Of course, combining the ideas of the limited residual and the crosswind loss is also possible. Replacing $\mathcal{L}_{\mathrm{I}}^{\mathrm{st}}$ by $\mathcal{L}_{\mathrm{I}}^{\mathrm{lr}}$ in the definition of $\mathcal{L}_{\mathrm{I}}^{\mathrm{cw}}$ leads to

$$\sum_{i=1}^{N_{\mathrm{I}}} \xi \left(\frac{1}{N_{\mathrm{I}}^2 t_0} \left(\mathrm{Res}(u_{\mathcal{N}})(\boldsymbol{x}_{i,\mathrm{I}}) \right)^2 \right) + \frac{|\Omega|}{N_{\mathrm{I}}} \sum_{i=1}^{N_{\mathrm{I}}} |\phi(|\boldsymbol{b}^{\perp}(\boldsymbol{x}_{i,\mathrm{I}}) \cdot \nabla u_{\mathcal{N}}(\boldsymbol{x}_{i,\mathrm{I}})|)| =: \mathcal{L}_{\mathrm{I}}^{\mathrm{lrcw}}.$$

Once more, let $\alpha_{\mathrm{I}}^{\mathrm{lrcw}}, \alpha_{\mathrm{D}}^{\mathrm{lrcw}}, \alpha_{\mathrm{N}}^{\mathrm{lrcw}} \in \mathbb{R}$ again be three non-negative weights. Then, the limited residual loss with crosswind term can be defined as

$$\mathcal{L}^{\mathrm{lrcw}} := \alpha_{\mathrm{I}}^{\mathrm{lrcw}} \mathcal{L}_{\mathrm{I}}^{\mathrm{lrcw}} + \alpha_{\mathrm{D}}^{\mathrm{lrcw}} \mathcal{L}_{\mathrm{D}}^{\mathrm{st}} + \alpha_{\mathrm{N}}^{\mathrm{lrcw}} \mathcal{L}_{\mathrm{N}}^{\mathrm{st}}. \tag{5.10}$$

5.2.3. Variational physics-informed neural networks

The standard PINN formulation of the loss functional and all variants described in the previous section are based on the strong form of the residual. However, as seen in section 2.1.2, it can be proven that a unique weak solution exists, whereas strong solutions only exist under somewhat restrictive regularity assumptions on the data. Therefore, the notion of weak solutions is generally better suited for convection-diffusion-reaction problems, and it might be advantageous if PINNs reflect this.

Several attempts in this direction have already been made in the literature, e.g., (*hp*)-variational physics-informed neural networks [KZK19; KZK21], weak (adversarial) physics-informed neural networks [DMM22; Zan+20], and the references therein. They differ significantly in the concrete choice of the test function(s). While in the former two references, polynomial test functions are used, in the latter two, only a single MLP-based test function is used, and the problem is reformulated to be a min-max problem. Using an MLP-based test function in contrast to polynomial ones has the advantage that no underlying mesh is needed for classical PINNs and that the results do not depend on a maximal chosen polynomial degree. On the contrary, it has the disadvantage that the computational costs are proportional to $2nm$, where n is the number of min-max optimizations and m is the number of epochs each network is trained, i.e., in the worst case, $n = m$, the costs scale quadratically. Furthermore, from a numerical point of view, converging towards the approximate solution is more challenging than in a standard minimization problem because a certain saddle point of the loss functional must be found [DMM22, section 4.5.2].

In this section, the path of Kharazmi, Zhang, and Karniadakis [KZK21] is followed, and the *hp*-variational physics-informed neural network (*hp*-vPINN) approach is explained for convection-diffusion-reaction problems.

To this end, let $\mathcal{T}_h$ be a triangulation of $\overline{\Omega} \subset \mathbb{R}^d$, $d = 1, 2, 3$, into cells $K \in \mathcal{T}_h$ that are, depending on the dimension, either lines, quadrilaterals or hexahedrons. The cells are assumed to be the image of an affine or d-linear mapping $F_K : \widehat{K} \to K$ from

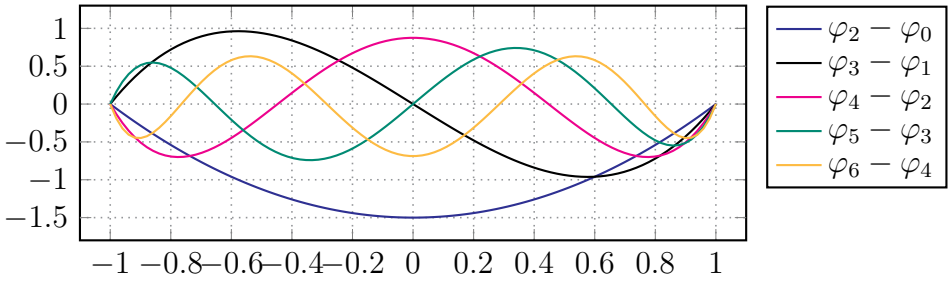


Figure 5.5.: Set of one-dimensional test functions on the reference cell.

the reference cell $\widehat{K} \coloneqq [-1,1]^d$ to K. In the spirit of mapped finite elements, the following quantities are defined on the reference cell and then mapped to the physical cell.

To start with, let

$$P_p(\widehat{K}) \coloneqq \{\, \widehat{v}_i \;\; : \;\; i \in \mathbb{N} \,\}$$

be a set of test functions on the reference cell, where, for all $i \in \mathbb{N}$, v_i is a polynomial of degree at most p, $\widehat{v}_i$ is not constantly zero, and $\widehat{v}_i|_{\partial\widehat{K}} = 0$. For all $K \in \mathcal{T}_h$, the set of local test functions $P_p(K)$ can then be defined as

$$P_p(K) \coloneqq \{\, v : K \to \mathbb{R} \;\; : \;\; v = \widehat{v} \circ F_K^{-1} \text{ for a } \widehat{v} \in P_p(\widehat{K}) \,\},$$

where F_K^{-1} is the inverse of the reference transform. These polynomials can be continuously extended to $\overline{\Omega}$ by defining $v|_{\overline{\Omega}\setminus K} \coloneqq 0$ for every $v \in P_p(K)$ and all $K \in \mathcal{T}_h$. The set of global polynomial test functions $P_p(\mathcal{T}_h)$ is then defined as

$$P_p(\mathcal{T}_h) \coloneqq \{\, v : \overline{\Omega} \to \mathbb{R} \;\; : \;\; v|_K \in P_p(K) \text{ for a } K \in \mathcal{T}_h,\; v|_{\overline{\Omega}\setminus K} = 0 \,\}.$$

Note that by construction all test functions $v \in P_p(\mathcal{T}_h)$ have a compact support and satisfy $v|_{\Gamma_D} = 0$.

Kharazmi, Zhang, and Karniadakis propose to use in one dimension on the reference cell

$$P_p([-1,1]) \coloneqq \{\, \varphi_{k+1}(x) - \varphi_{k-1}(x) \;\; : \;\; k = 1, 2, \ldots, p-1 \,\},$$

where φ_k is the Legendre polynomial of order k [KZK19; KZK21]. These polynomials are also depicted in figure 5.5 up to order $p = 6$. In two dimensions the product space of $P_p([-1,1])$ with itself is considered [KZK19; KZK21], i.e.,

$$P_p([-1,1]^2) \coloneqq \{\, \varphi_k(x)\varphi_\ell(y) \;\; : \;\; \varphi_k, \varphi_\ell \in P_p([-1,1]),\; k, \ell = 1, 2, \ldots, p-1 \,\}, \quad (5.11)$$

which will also be used in section 5.3.3. In principle, instead of equal polynomial degrees in x and y direction, also different ones can be used.

Given the set of global test functions $P_p(\mathcal{T}_h)$, the quantity

$$\sum_{K\in\mathcal{T}_h}\frac{1}{|P_p(K)|}\sum_{v\in P_p(K)}\int_\Omega \mathrm{Res}(u_\mathcal{N})\,v\,\mathrm{d}\boldsymbol{x}=\sum_{K\in\mathcal{T}_h}\frac{1}{|P_p(K)|}\sum_{v\in P_p(K)}\int_K \mathrm{Res}(u_\mathcal{N})\,v\,\mathrm{d}\boldsymbol{x} \tag{5.12}$$

serves as ansatz for the interior loss functional. The integrals can then be transformed to the reference element, which leads to

$$\sum_{v\in P_p(K)}\int_K \mathrm{Res}(u_\mathcal{N})\,v\,\mathrm{d}\boldsymbol{x}=\sum_{\widehat{v}\in P_p(\widehat{K})}\int_{\widehat{K}} \mathrm{Res}(u_\mathcal{N})\circ F_K\,\widehat{v}\,|\det(DF_K)|\,\mathrm{d}\widehat{\boldsymbol{x}}.$$

As in the classical PINN setting, after the integrals are transformed, they must be approximated by a quadrature rule. In contrast to the vanilla PINN approach Kharazmi, Zhang, and Karniadakis propose to use a Gauss–Lobatto integration rule with n^d, $n\in\mathbb{N}$, points for lower-dimensional problems and a Monte Carlo integration method for higher-dimensional problems [KZK19; KZK21]. Following this idea, a Gauss–Legendre integration rule with n^d weights and points is applied in this work. Let ω_i and $\widehat{x}_{i,\mathrm{I}}$, $i=1,2,\dots,n$, be the weights and points of the Gauss–Legendre integration rule of order n on $[-1,1]$. Then, a Gauss–Legendre integration rule in two dimensions is given by $\omega_{i,j}\coloneqq\omega_i\omega_j$ and $\widehat{\boldsymbol{x}}_{i,j,\mathrm{I}}\coloneqq(\widehat{x}_{i,\mathrm{I}},\widehat{x}_{j,\mathrm{I}})$, $i,j=1,2,\dots,n$, which can be used to arrive at

$$\begin{aligned}\sum_{v\in P_p(K)}\int_\Omega \mathrm{Res}(u_\mathcal{N})\,v\,\mathrm{d}\boldsymbol{x}\approx\\ \sum_{\widehat{v}\in P_p(\widehat{K})}\sum_{i,j=1}^{n}\omega_{i,j}\,\mathrm{Res}(u_\mathcal{N})(F_K(\widehat{\boldsymbol{x}}_{i,j,\mathrm{I}}))\,\widehat{v}(\widehat{\boldsymbol{x}}_{i,j,\mathrm{I}})\,|\det(DF_K(\widehat{\boldsymbol{x}}_{i,j,\mathrm{I}}))|.\end{aligned} \tag{5.13}$$

Combining equations (5.12) and (5.13) finally leads to

$$\sum_{K\in\mathcal{T}_h}\frac{1}{|P_p(K)|}\sum_{\widehat{v}\in P_p(\widehat{K})}\sum_{i,j=1}^{n}\omega_{i,j}\,\mathrm{Res}(u_\mathcal{N})(F_K(\widehat{\boldsymbol{x}}_{i,j,\mathrm{I}}))\,\widehat{v}(\widehat{\boldsymbol{x}}_{i,j,\mathrm{I}})\,|\det(DF_K(\widehat{\boldsymbol{x}}_{i,j,\mathrm{I}}))|=:\mathcal{L}_\mathrm{I}^\mathrm{hpvP}$$

that is used as interior loss functional in *hp*-vPINNs. The derivation works analogously in three dimensions.

Finally, a weighted sum of the interior loss and the boundary losses is applied to get the loss functional for *hp*-vPINNs

$$\mathcal{L}^\mathrm{hpvP}\coloneqq\alpha_\mathrm{I}^\mathrm{hpvP}\mathcal{L}_\mathrm{I}^\mathrm{hpvP}+\alpha_\mathrm{D}^\mathrm{hpvP}\mathcal{L}_\mathrm{D}^\mathrm{st}+\alpha_\mathrm{N}^\mathrm{hpvP}\mathcal{L}_\mathrm{N}^\mathrm{st}, \tag{5.14}$$

where $\alpha_\mathrm{I}^\mathrm{hpvP},\alpha_\mathrm{D}^\mathrm{hpvP},\alpha_\mathrm{N}^\mathrm{hpvP}\in\mathbb{R}$ as above are three non-negative weights.

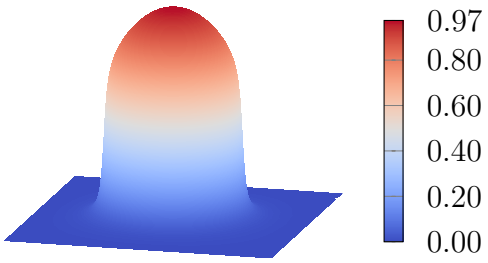


Figure 5.6.: Exact solution to example 5.6.

Remark 5.4. In [KZK19; KZK21] the authors propose two more variants of $\mathcal{L}_{\mathrm{I}}^{\mathrm{hpvP}}$ where they integrate by parts $\mathcal{L}_{\mathrm{I}}^{\mathrm{hpvP}}$ one and two, resp., times. For their two-dimensional test problems, the authors report no significant differences between the standard formulation of $\mathcal{L}_{\mathrm{I}}^{\mathrm{hpvP}}$ and the one that is integrated by parts once. The last variant is not tested. In the numerical experiments in section 5.3.3, the loss functional given in equation (5.14) is used. □

Remark 5.5. The difference between *hp*-vPINNs and vPINNs is that in the latter, the domain is not decomposed into several cells but rather treated as a single cell, cf. [KZK19].

The vanilla PINN approach can also be seen as a *hp*-vPINN version, namely for a single cell $K = \overline{\Omega}$ and using Dirac delta functions as test functions, i.e., $v_i(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{x}_{i,\mathrm{I}})$, where $\delta(\mathbf{0}) \coloneqq 1$ and $\delta(\boldsymbol{x} - \boldsymbol{x}_{i,\mathrm{I}}) \coloneqq 0$ for all $\boldsymbol{x} \in \Omega \setminus \{\boldsymbol{x}_{i,\mathrm{I}}\}$. □

5.3. Numerical studies

In this part, the ideas presented in the previous section are tested numerically on two benchmark problems. All implementation aspects mentioned on page 96 also hold in this part of the thesis.

In contrast to the previous chapters, two examples are used that have a known exact solution. The reason is that with these examples, the error between the PINN approximations and the exact solution can be measured to assess the quality of the various modifications of PINNs.

The first example is the same as example 2 of [JMT97, p. 181] and is defined as follows.

Example 5.6 (Circular internal layer)**.** Let $\Omega \coloneqq (0,1)^2$ be the unit square, $\varepsilon \coloneqq 10^{-8}$, $\boldsymbol{b} \coloneqq (2,3)^T$, $c \coloneqq 2$, and $\Gamma_{\mathrm{D}} \coloneqq \partial\Omega$. The right-hand side and the boundary conditions of the problem are chosen such that

$$u(x,y) \coloneqq 16x(1-x)y(1-y)\left(\frac{1}{2} + \frac{\arctan\left(200(r_0^2 - (x-x_0)^2 - (y-y_0)^2)\right)}{\pi}\right)$$

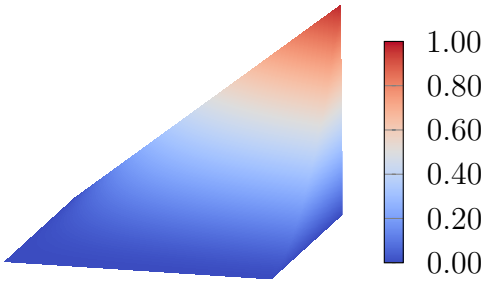


Figure 5.7.: Exact solution to example 5.7.

is the exact solution, where $r_0 \coloneqq 0.25$ and $x_0 \coloneqq y_0 \coloneqq 0.5$. The exact solution is also depicted in figure 5.6. □

As seen in figure 5.6, the solution to the previous example possesses an interior layer at the circle with a radius 0.25. To test how well PINNs cope with boundary layer problems, the following example is defined, which is taken from [JMT97, example 3].

Example 5.7 (Outflow layer)**.** Let $\Omega \coloneqq (0,1)^2$ be the unit square, $\varepsilon \coloneqq 10^{-8}$, $\boldsymbol{b} \coloneqq (2,3)^T$, $c \coloneqq 1$, and $\Gamma_\mathrm{D} \coloneqq \partial\Omega$. The right-hand side and the boundary conditions of the problem are chosen such that

$$u(x,y) \coloneqq xy^2 - y^2 \exp\left(\frac{2(x-1)}{\varepsilon}\right) - x \exp\left(\frac{3(y-1)}{\varepsilon}\right) + \exp\left(\frac{2(x-1)+3(y-1)}{\varepsilon}\right)$$

is the exact solution, which is also shown in figure 5.7. □

Note that all experiments are conducted in the convection-dominated regime since in both examples $\varepsilon = 10^{-8}$ and $\boldsymbol{b} = (2,3)^T$.

The error $e \coloneqq u - u_\mathcal{N}$ between the exact solution u and a PINN approximation $u_\mathcal{N}$ is measured by $|||\cdot|||$ which is defined as

$$|||e|||^2 \coloneqq \varepsilon \|\nabla e\|^2_{L^2(\Omega)} + \mu_0 \|e\|^2_{L^2(\Omega)}, \tag{5.15}$$

where $\varepsilon = 10^{-8}$, $\mu_0 \coloneqq 2$ for example 5.6 and $\mu_0 \coloneqq 1$ for example 5.7. To compute an approximation of the error by numerical quadrature, the unit square is divided into $10,000$ squares of equal size in which each a Gauss–Legendre quadrature rule with ten points in each coordinate direction is used. This results in $1,000,000$ quadrature points and weights in total.

5.3.1. Comparing pretrained and hard-constrained PINNs with vanilla PINNs

To start off the experiments, the ideas presented in section 5.2.1 are tested on examples 5.6 and 5.7 and compared to the vanilla PINN approach of section 5.1.1.

Table 5.1.: Set of hyperparameters used in section 5.3.1. This gives in total 945 combinations of hyperparameters.

nodes in hidden layers	20, 30, 40
activation	ELU, tanh, ReLU
learning rate	$0.01 \cdot 3^{-0}$, $0.01 \cdot 3^{-1}$, $0.01 \cdot 3^{-2}$, $0.01 \cdot 3^{-3}$, $0.01 \cdot 3^{-4}$, $0.01 \cdot 3^{-5}$, $0.01 \cdot 3^{-6}$
initialization seed	42, 43, 44
weight decay λ_{wd}	10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}

To this end, in each experiment 945 different configurations of MLPs are optimized whose hyperparameters are given in table 5.1. The hyperparameters are chosen in correspondence with the practical recommendations from [Ben12]. Since the MLPs shall approximate a function mapping from a subset of $\mathbb{R}^2$ to $\mathbb{R}$, the input layer consists of two nodes and the output layer of a single node. All networks have seven hidden layers with the same number of nodes in each layer, i.e., either 20, 30, or 40, cf. table 5.1. While the values of the weights are initialized in the beginning based on Glorot initialization [GB10] with the seeds given in table 5.1, the initial biases are set to zero. In all hidden layers, the same activation function is used, namely either ELU or tanh depicted in figure 4.2, or the ReLU function defined for a given $x \in \mathbb{R}$ as $\text{ReLU}(x) \coloneqq \max\{\, 0, x \,\}$. Finally, in the output layer, a linear activation function is applied.

The MLPs are trained for $10,000$ epochs using $N_{\text{I}} = 4,096$ equally distanced interior and $N_{\text{D}} = 512$ equally distanced Dirichlet boundary points that are split into batches of size $n_{\text{bs}} \coloneqq 32$. During the training, the MLPs are optimized to minimize

$$\mathcal{L}^{\text{st}} + \frac{\lambda_{\text{wd}}}{2} \frac{n_{\text{bs}}}{(N_{\text{I}} + N_{\text{D}})} \sum_j w_j^2, \tag{5.16}$$

where $\mathcal{L}^{\text{st}}$ is defined in equation (5.4), λ_{wd} is given in table 5.1, and w_j are the components of the weight matrices of the MLPs. The second term is the so-called *L^2-weight decay regularization*, also commonly known as *ridge regression* or *Tikhonov*

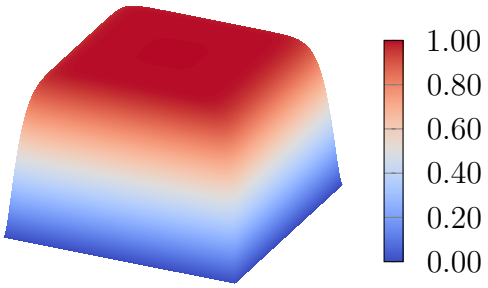


Figure 5.8.: Interior indicator l for the unit square given in equation (5.17).

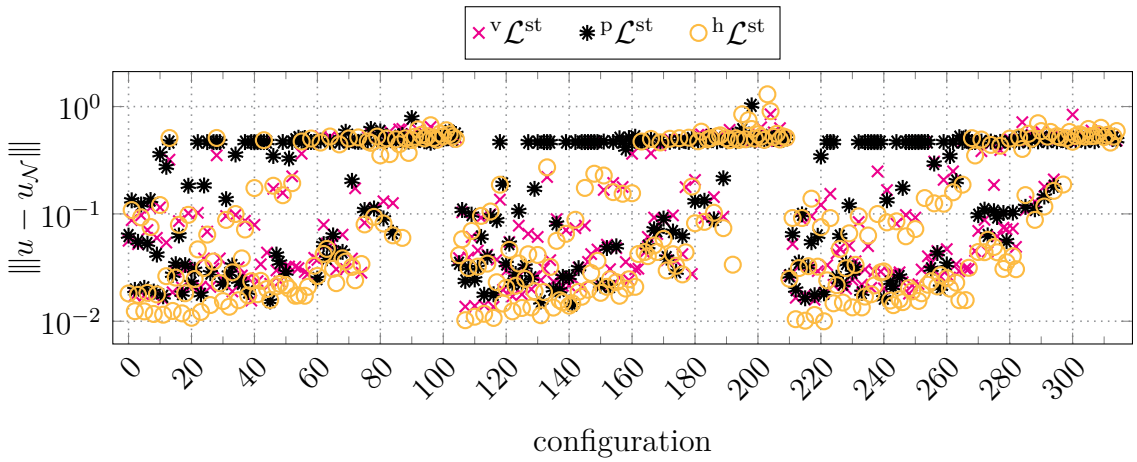


Figure 5.9.: Errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs for all tested configurations of vanilla PINNs (${}^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$), PINNs with pretraining (${}^{\mathrm{p}}\mathcal{L}^{\mathrm{st}}$) and hard-constrained PINNs (${}^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$) that approximate the solution to example 5.6 from section 5.3.1. The PINNs are trained to minimize the loss given in equation (5.16). The error is averaged over the seeds of the configurations.

regularization, and is often used in MLP optimizations to counteract overfitting; see also [Ben12; GBC16, pp. 116–119, 227–230] in general and [DBB23] in the context of PINNs. Note that this term does not include the biases but only the weights of the MLP. The factors in front of the individual terms of $\mathcal{L}^{\mathrm{st}}$ are set to $\alpha_{\mathrm{I}}^{\mathrm{st}} = \alpha_{\mathrm{D}}^{\mathrm{st}} = 1$.

Three methods are tested, namely the vanilla PINNs from section 5.1.1, vanilla PINNs that are pretrained for 1,000 epochs on the inflow training points before the actual optimization takes place as described in section 5.2.1, and hard-constrained PINNs as also described in section 5.2.1, equation (5.5). Below the superscripts ${}^{\mathrm{v}}$, ${}^{\mathrm{p}}$, and ${}^{\mathrm{h}}$ indicate the usage of vanilla PINNs, pretrained PINNs, and hard-constrained PINNs, respectively. For the latter, the continuous extension of the boundary condition $\widetilde{g}_{\mathrm{D}}$ is defined, for $(x, y) \in \overline{\Omega}$, as $\widetilde{g}_{\mathrm{D}}(x, y) \coloneqq 0$, and the indicator function l is set to be

$$l(x, y) \coloneqq \left(1 - e^{-\kappa x}\right)\left(1 - e^{-\kappa y}\right)\left(1 - e^{-\kappa(1-x)}\right)\left(1 - e^{-\kappa(1-y)}\right), \tag{5.17}$$

where $\kappa \coloneqq 30$ is a scaling factor. The function l is also visualized in figure 5.8.

After the MLPs are trained for 10,000 epochs, the error between the MLPs and the exact solution is measured in terms of the energy norm defined in equation (5.15), and a mean value of the MLPs over the seeds is computed. The 315 arising configurations are numbered to be identified. The six best MLPs of each method are then trained for all three seeds for another 90,000 epochs, after which again the error is computed and averaged over the seeds.

Table 5.2.: Mean and minimal value of errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of all vanilla PINNs (${}^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$), PINNs with pretraining (${}^{\mathrm{p}}\mathcal{L}^{\mathrm{st}}$) and hard-constrained PINNs (${}^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$) that approximate the solution to example 5.6 from section 5.3.1. The PINNs are trained to minimize the loss given in equation (5.16).

	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{p}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$
mean	0.2136	0.3065	0.2019
min	0.0136	0.0141	0.0100

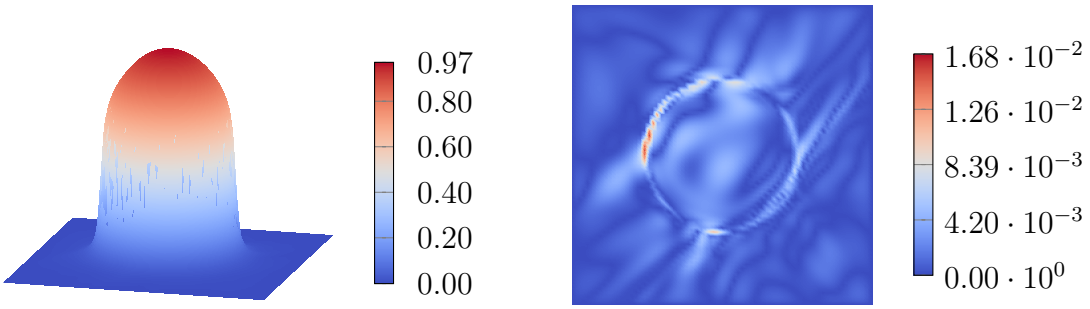
Circular interior layer problem

First, the results for MLPs trained to approximate the solution to example 5.6 are discussed. The errors of the MLPs after 10,000 epochs for all 315 configurations are depicted in figure 5.9. The pretrained PINNs seem to show larger errors compared to the vanilla and the hard-constrained PINNs, of which the latter work slightly better. This is also supported by the mean and, more importantly, the minimal values of these methods shown in table 5.2. Note that in this thesis, the minimal value is considered to be of higher importance than the mean because the best method is the one that would be used in practice. The trend of the methods also proceeds after the six best configurations are trained for another 90,000 epochs. A hard-constrained PINN shows the smallest overall error (0.0088), the best vanilla PINN has an error of approximately 0.0094, and the best pretrained PINN owes an error of around 0.0100.

Furthermore, it can be observed that in the first optimization steps in the actual training loop, the pretrained MLPs "forget" the inflow boundary values they learned during the pretraining. This is because the interior loss decreases more than the boundary loss increases. Moreover, after both 10,000 and 100,000 epochs, the pretrained MLPs show a worse mean and minimal error than vanilla PINNs. The pretrained PINNs may be overfitted to the inflow boundary data, and it takes more

Table 5.3.: Pearson correlation coefficients between the hyperparameters and the errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of vanilla PINNs (${}^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$), PINNs with pretraining (${}^{\mathrm{p}}\mathcal{L}^{\mathrm{st}}$) and hard-constrained PINNs (${}^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$) that approximate the solution to example 5.6 from section 5.3.1. The PINNs are trained to minimize the loss given in equation (5.16). Colors indicate the magnitude of the values.

	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{p}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$
model size	0.009	0.016	0.007
learning rate	0.747	0.484	0.758
weight decay λ_{wd}	0.125	0.378	0.083
activation	0.076	0.055	0.113



(a) $^{\mathrm{h}}$PINN approximation $u_{\mathcal{N}}$. (b) $|u - u_{\mathcal{N}}|$.

Figure 5.10.: Hard-constrained PINN approximation $u_{\mathcal{N}}$ of the solution to example 5.6 based on minimizing the loss given in equation (5.16) for 100,000 epochs and its point wise error $|u - u_{\mathcal{N}}|$ compared with the exact solution u.

minimization steps for the MLPs to counteract this overfitting and to fit the actual solution compared to vanilla PINNs.

The Pearson correlation coefficients between the hyperparameters and error of all 315 MLPs after 10,000 epochs for all three methods are given in table 5.3. By far, the learning rate has the most considerable influence on the result, where a positive value indicates that the smaller the learning rate, the smaller the error. For the pretrained PINNs, the weight decay factor also has a significant influence. A smaller weight decay factor might result in less overfitting during the pretraining, and therefore the MLPs start the actual training loop with a less overfitted initial guess.

The solution and the point wise error $|u - u_{\mathcal{N}}|$ of the hard-constrained PINN with the configuration that has the smallest error after 100,000 epochs are depicted in figure 5.10. The absolute value point wise error is between precisely 0 at the boundary and at most around $1.68 \cdot 10^{-2}$ close to the circle with radius 0.25 where the solution possesses the interior layer. In total, it can be seen that the PINN approximation captures the solution appropriately well, as it could also be expected by the norm of the error. The maximal and minimal values of the approximation also coincide with those of the exact solution up to two decimal places.

To conclude, for this problem with an interior layer, the hard-constrained PINNs work the best. Not only do they have the MLP with the smallest error, but they also need less training effort than the other methods since the boundary data does not need to be learned, and the Dirichlet boundary data are met exactly. Moreover, pretraining is not worth the computational effort for this example since the results are the worst and it needs the most computational resources.

Outflow layer problem

After the problem with the interior layer is treated, in this section, the results for the problem with the boundary layer at the outflow boundary from example 5.7 are

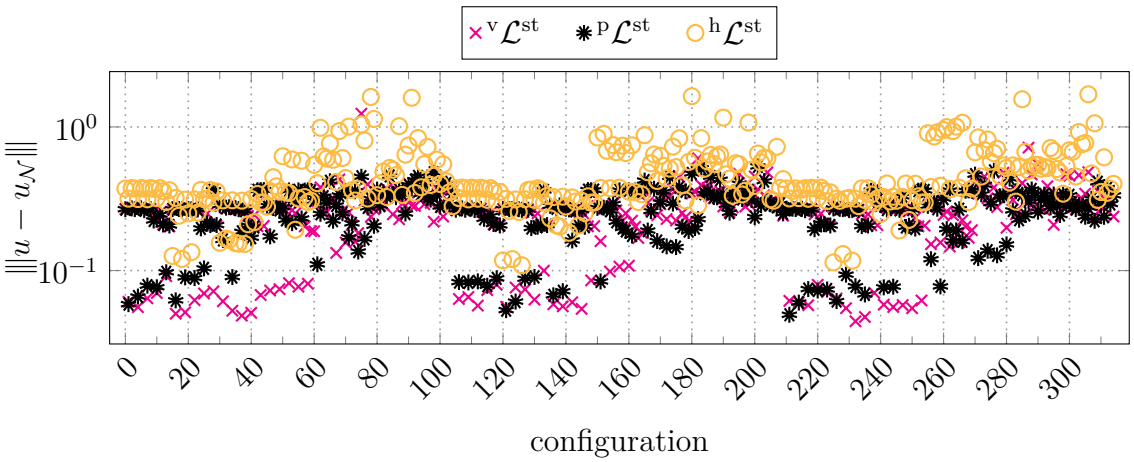


Figure 5.11.: Errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs for all tested configurations of vanilla PINNs ($^{v}\mathcal{L}^{st}$), PINNs with pretraining ($^{p}\mathcal{L}^{st}$) and hard-constrained PINNs ($^{h}\mathcal{L}^{st}$) that approximate the solution to example 5.7 from section 5.3.1. The PINNs are trained to minimize the loss given in equation (5.16). The error is averaged over the seeds of the configurations.

presented.

The errors $|||u - u_{\mathcal{N}}|||$ of all configurations averaged over the seeds for the three tested methods after 10,000 epochs are depicted in figure 5.11. For this example, the hard-constrained PINNs work worse than the vanilla and the pretrained PINNs. This is also supported by the mean over the errors and the smallest error given in table 5.4. The average error for hard-constrained PINNs is almost twice as large as the average value for vanilla PINNs. The vanilla PINN approximation with the smallest error also has the smallest error compared to the best pretrained and hard-constrained approximation, and its error is also around half the error of the best hard-constrained configuration.

This trend stays the same even after the six best configurations are trained for another 90,000 epochs. The smallest error shows a vanilla PINN configuration with a value of approximately 0.0316, followed by a pretrained PINN approximation of

Table 5.4.: Mean and minimal value of errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of all vanilla PINNs ($^{v}\mathcal{L}^{st}$), PINNs with pretraining ($^{p}\mathcal{L}^{st}$) and hard-constrained PINNs ($^{h}\mathcal{L}^{st}$) that approximate the solution to example 5.7 from section 5.3.1. The PINNs are trained to minimize the loss given in equation (5.16).

	$\|\|\|u - u_{\mathcal{N}}\|\|\|$ $^{v}\mathcal{L}^{st}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|$ $^{p}\mathcal{L}^{st}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|$ $^{h}\mathcal{L}^{st}$
mean	0.2514	0.2658	0.4652
min	0.0444	0.0493	0.1089

Table 5.5.: Pearson correlation coefficients between the hyperparameters and the errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of vanilla PINNs (${}^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$), PINNs with pretraining (${}^{\mathrm{p}}\mathcal{L}^{\mathrm{st}}$) and hard-constrained PINNs (${}^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$) that approximate the solution to example 5.7 from section 5.3.1. The PINNs are trained to minimize the loss given in equation (5.16). Colors indicate the magnitude of the values.

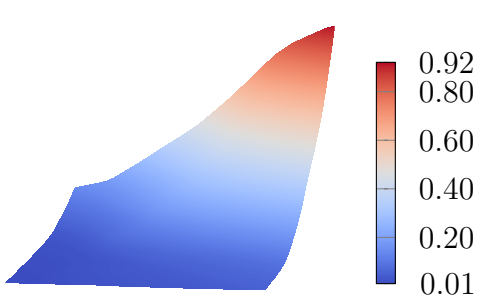
	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{p}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$
model size	0.060	−0.052	0.092
learning rate	0.404	0.374	0.317
weight decay λ_{wd}	−0.163	0.137	−0.173
activation	−0.048	−0.090	−0.074

approximately 0.0318. The best hard-constrained PINN shows an error of 0.0391. However, after 100,000 epochs, the difference between the best of the three methods decreased compared to the results after 10,000 epochs. Compared to the errors of the previous experiment, the smallest minimal error after 100,000 is around four times larger. This indicates that PINNs might cope better with interior layers than boundary layers or problems with outflow boundary layers are more difficult to approximate.

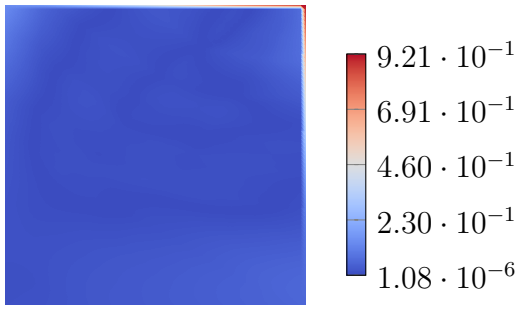
As in the previous experiment, it can be observed that the PINNs that are pretrained on the inflow boundary data change abruptly in the first optimization step of the actual optimization loop. The reason is probably the same as before: the interior loss can be decreased more than the boundary loss increases if the inflow boundary data are met worse.

To investigate which model parameters have the most significant influence on the training of the three methods, the Pearson correlation coefficients between the hyperparameters and the errors are given in table 5.5. As in the previous example, the learning rate has the largest influence, even if it is for the vanilla and the hard-constrained PINNs only around half as much as in the previous experiment.

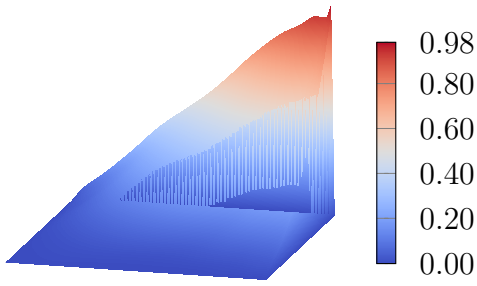
The solution of the best vanilla PINN approximation and the point wise error $|u - u_{\mathcal{N}}|$ compared to the exact solution u to example 5.7 are shown in figures 5.12a and 5.12b. It can be observed that in the approximation, no outflow layer occurs and that it does not meet the boundary data at the outflow boundary. This might also be the reason why the error measured in the $|||\cdot|||$-norm is larger than in the previous experiment. The maximal value of the approximation (≈ 0.92) also is smaller than the largest value of the exact solution (≈ 1.00). This is also in contrast to the previous experiment, where the minimum and the maximum are better satisfied. A possible explanation is that the interior loss functional of vanilla PINNs can be decreased more than the boundary term increases. However, the boundary loss still plays a role and makes it more difficult for the PINNs to meet the maximum of the solution. The point wise error is in the majority of the domain usually small, with a smallest value of order $\mathcal{O}(10^{-6})$. However, the error is obviously larger along the outflow boundary



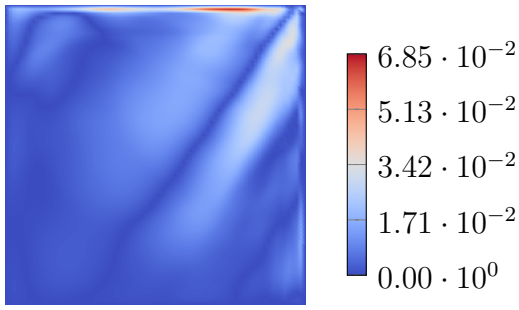
(a) $^{\text{v}}$PINN approximation $u_{\mathcal{N}}$.



(b) $|u - u_{\mathcal{N}}|$ of $^{\text{v}}$PINN approximation.



(c) $^{\text{h}}$PINN approximation $u_{\mathcal{N}}$.



(d) $|u - u_{\mathcal{N}}|$ of $^{\text{h}}$PINN approximation.

Figure 5.12.: Vanilla (top row) and hard-constrained (bottom row) PINN approximations $u_{\mathcal{N}}$ of the solution to example 5.7 based on minimizing the loss given in equation (5.16) for $100,000$ epochs and their point wise errors $|u - u_{\mathcal{N}}|$ compared to the exact solution u.

at $y = 1$ or $x = 1$ and especially at the corner at $(1, 1)$ since the outflow boundary data is not met at all.

The solution of the best hard-constrained PINN and its point wise error $|u - u_{\mathcal{N}}|$ for the same problem are shown in figures 5.12c and 5.12d for comparison. It is visible that an outflow boundary layer has formed and that the boundary data are met exactly due to prescribing them in a hard-constrained manner. Therefore, the point wise error is smaller and in most parts of the domain of order $\mathcal{O}(10^{-2})$. It is the largest at the upper part of the outflow layer. This is not surprising when looking at the solution. The solution is deformed and wiggles the closer it gets to the upper right corner.

An analog mechanism might also be why hard-constrained PINNs do not work as well as in the previous experiment. The hard-constrained PINNs seem to have difficulties introducing the steep gradient in the outflow layer because the outflow boundary data is fixed and must be satisfied.

To conclude, vanilla PINNs seem to approximate the solution better in the interior than hard-constrained PINNs. With a longer training time, hard-constrained PINNs may also catch up with the vanilla ones. As a drawback, a particular feature of the

solution, namely the outflow boundary layer, is not reconstructed by vanilla PINNs at all. However, for this example with a boundary layer, at the outflow boundary, vanilla PINNs work best in terms of the energy error $|||\cdot|||$ at least after $10,000$ and $100,000$ epochs, which is why in the following vanilla PINNs are used for this particular example.

5.3.2. Non-standard loss functionals

The experiments in this section test which of the loss functionals defined in section 5.2.2 works best and compare them to the vanilla loss functional from section 5.1.1. In other words, the effect of the four loss functionals $\mathcal{L}^{\mathrm{st}}$, $\mathcal{L}^{\mathrm{cw}}$, $\mathcal{L}^{\mathrm{lr}}$, $\mathcal{L}^{\mathrm{lrcw}}$ on the quality of the approximation is investigated.

To this end, for each method, 675 MLPs are trained with different configurations of hyperparameters. The same hyperparameters are chosen as given in table 5.1, except for the learning rate, of which the largest two learning rates are neglected. Depending on the experiment, the boundary data is incorporated in either the hard-constrained or vanilla fashion. The particular choice is mentioned in each experiment and indicated in the results by either the superscript $^{\mathrm{h}}$ or $^{\mathrm{v}}$. As before, to each tested loss functional an L^2-weight decay regularization term is added, i.e.,

$$\mathcal{L} + \frac{\lambda_{\mathrm{wd}}}{2} \frac{n_{\mathrm{bs}}}{N_{\mathrm{I}} + N_{\mathrm{D}}} \sum_j w_j^2, \tag{5.18}$$

are used as loss functionals, where $\mathcal{L} \in \{ \mathcal{L}^{\mathrm{st}}, \mathcal{L}^{\mathrm{cw}}, \mathcal{L}^{\mathrm{lr}}, \mathcal{L}^{\mathrm{lrcw}} \}$. Below, the losses are nevertheless denoted by $\mathcal{L}^{\mathrm{st}}$, $\mathcal{L}^{\mathrm{cw}}$, $\mathcal{L}^{\mathrm{lr}}$, and $\mathcal{L}^{\mathrm{lrcw}}$, but it must be remembered that the weight decay term is active in all experiments.

The rest of the set-up of the MLPs and the training described in section 5.3.1 stays the same, e.g., the number of layers, training points, and training epochs. The evaluation also stays the same, i.e., the error is calculated in the $|||\cdot|||$-norm, and an average is computed over the seeds resulting in errors of 225 different configurations for each method.

The loss functionals $\mathcal{L}^{\mathrm{lr}}$ and $\mathcal{L}^{\mathrm{lrcw}}$ depend on the parameter t_0. The corresponding loss functionals are denoted by $\mathcal{L}^{\mathrm{lr}}_{t_0}$ and $\mathcal{L}^{\mathrm{lrcw}}_{t_0}$, where t_0 is chosen to be one of the values of $\{ 10^1,\ 10^0,\ 10^{-1},\ 10^{-2},\ 10^{-3},\ 10^{-4} \}$. The values of this set follow the suggestions in [KLS19]. To summarize, together with $\mathcal{L}^{\mathrm{st}}$ and $\mathcal{L}^{\mathrm{cw}}$ in total 14 different loss functionals are tested. However, only one value for t_0 is shown below for the sake of brevity, namely for the t_0 for which $\mathcal{L}^{\mathrm{lr}}$ and $\mathcal{L}^{\mathrm{lrcw}}$ have the smallest minimal energy error after $100,000$ epochs.

Circular interior layer problem

Again, the experiments start with PINNs approximating the solution to example 5.6 that possesses an interior layer. Since section 5.3.1 shows that for this problem hard-

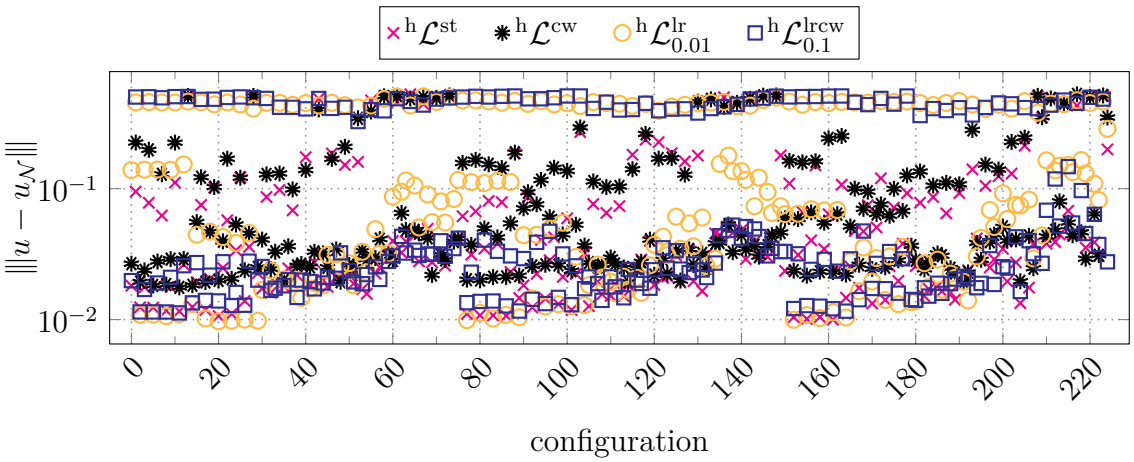


Figure 5.13.: Errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs for all tested configurations of hard-constrained PINNs with various loss functionals given in equation (5.18) that approximate the solution to example 5.6 from section 5.3.1. The error is averaged over the seeds of the configurations.

constrained PINNs work the best, also in these experiments, the Dirichlet boundary data is imposed exactly as indicated by the superscript $^{\mathrm{h}}$.

The results of the vanilla, the crosswind functional, $\mathcal{L}^{\mathrm{lr}}_{0.01}$ and $\mathcal{L}^{\mathrm{lrcw}}_{0.1}$ after 10,000 epochs for all configurations are depicted in figure 5.13. It looks like that in the mean, both $\mathcal{L}^{\mathrm{lr}}_{0.01}$ and $\mathcal{L}^{\mathrm{lrcw}}_{0.1}$ work worse than the vanilla and the crosswind losses. This is also supported by the values presented in table 5.6, where it can be seen that the mean value of the limited residual and the limited residual with crosswind loss are around twice as large as the mean value of the vanilla PINNs. However, the smallest error is obtained by a PINN trained with limited residual loss. The best vanilla PINN has only a slightly larger error, and the best one trained with the crosswind loss has an error of almost twice as much as the overall smallest one. The best PINN trained with the limited residual with crosswind loss has an error between the vanilla and the crosswind loss.

The results have slightly changed after the six best networks of each method are trained for another 90,000 epochs. Still, the loss that produces the smallest minimal

Table 5.6.: Mean and minimal value of errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of all hard-constrained PINNs with various loss functionals given in equation (5.18) that approximate the solution to example 5.6 from section 5.3.1.

	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ ^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ ^{\mathrm{h}}\mathcal{L}^{\mathrm{cw}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ ^{\mathrm{h}}\mathcal{L}^{\mathrm{lr}}_{0.01}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ ^{\mathrm{h}}\mathcal{L}^{\mathrm{lrcw}}_{0.1}$
mean	0.0940	0.1127	0.1856	0.1706
min	0.0100	0.0176	0.0097	0.0112

Table 5.7.: Pearson correlation coefficients between the hyperparameters and the errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of hard-constrained PINNs with various loss functionals given in equation (5.18) that approximate the solution to example 5.6 from section 5.3.1. Colors indicate the magnitude of the values.

	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{h}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{h}}\mathcal{L}^{\mathrm{cw}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{h}}\mathcal{L}^{\mathrm{lr}}_{0.01}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{h}}\mathcal{L}^{\mathrm{lrcw}}_{0.1}$
model size	−0.0392	−0.0121	0.0134	0.0080
learning rate	0.3876	0.3217	0.1118	0.0299
weight decay λ_{wd}	0.2286	0.2005	−0.0054	0.0045
activation	0.0135	0.0007	−0.0370	−0.0049

error is the limited residual loss (0.00867), but this is only marginally smaller than the results obtained with the limited residual with crosswind loss (0.00870), which is the second to best minimal error. The vanilla loss is after 100,000 in the third place (0.0092), followed by the crosswind loss (0.0177). Note that the results obtained with the crosswind loss even got slightly worse compared to the result after 10,000 epochs. Furthermore, the energy error of the best MLPs trained with the limited residual and limited residual with crosswind loss are around 5.7% and 5.4%, resp., smaller than the best vanilla PINN obtained in the previous section.

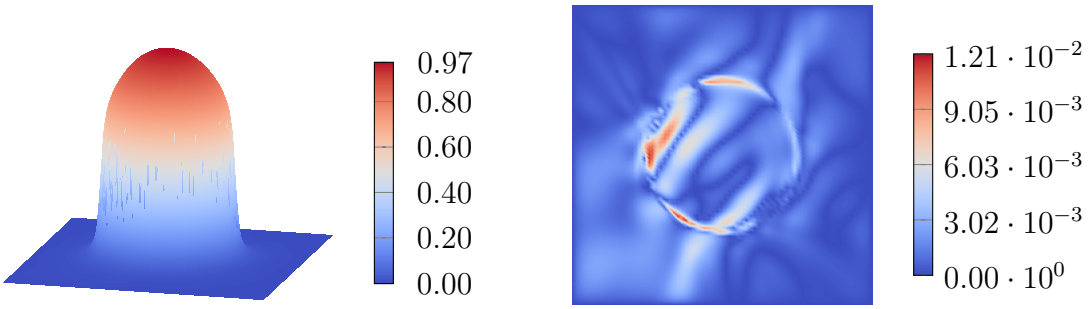
The Pearson correlation coefficient between the hyperparameters and the errors of the methods are given in table 5.7. It can be followed that the standard and the crosswind loss are most sensitive to the choice of the learning rate and the weight decay parameter. Both $\mathcal{L}^{\mathrm{lr}}_{0.01}$ and $\mathcal{L}^{\mathrm{lrcw}}_{0.1}$ are not notably sensitive to any particular hyperparameter which is in agreement with the large mean error. Only the learning rate has a minor influence on the result of the PINNs trained with $\mathcal{L}^{\mathrm{lr}}_{0.01}$.

The PINN approximation that has the smallest error overall obtained with $\mathcal{L}^{\mathrm{lr}}_{0.01}$ is depicted in figure 5.14 together with its point wise error compared to the exact solution u to example 5.6. Again, the approximation's minimal and maximal value coincides with the ones of the exact solution up to two decimal places. The point wise error is between 0 at the boundary and around $1.21 \cdot 10^{-2}$ obtained at the circle, which is around 28% smaller than the largest error obtained in section 5.3.1. Moreover, it is less localized compared to the previous section 5.3.1.

To conclude, both the energy and the largest point wise error for this example obtained with the limited residual and limited residual with crosswind loss are smaller than those with the vanilla loss. Moreover, the crosswind loss leads to PINNs with larger errors and is, therefore, less suited to problems with interior layers.

Outflow layer problem

To evaluate the influence of the loss functionals on the errors of PINNs that approximate the solution to a problem with a boundary layer, in this section the results for example 5.7 are presented. As seen in the previous section, vanilla PINNs worked the



(a) $^{\mathrm{h}}$PINN approximation $u_{\mathcal{N}}$. (b) $|u - u_{\mathcal{N}}|$.

Figure 5.14.: Hard-constrained PINN approximation $u_{\mathcal{N}}$ with $\mathcal{L}^{\mathrm{lr}}_{0.01}$ loss of the solution to example 5.6 after 100,000 epochs and its point wise error $|u - u_{\mathcal{N}}|$ compared to the exact solution u.

best compared to pretrained and hard-constrained PINNs. Therefore, the Dirichlet data is not prescribed exactly but learned during the training as indicated by the superscript $^{\mathrm{v}}$.

The errors of the PINNs for all configurations obtained with $\mathcal{L}^{\mathrm{st}}$, $\mathcal{L}^{\mathrm{cw}}$, $\mathcal{L}^{\mathrm{lr}}_{1.0}$ and $\mathcal{L}^{\mathrm{lrcw}}_{1.0}$ are shown in figure 5.15. It looks like the errors obtained with $\mathcal{L}^{\mathrm{lrcw}}_{1.0}$ are better in the mean than those with $\mathcal{L}^{\mathrm{lr}}_{1.0}$, and indeed, this is true as seen in table 5.8. The PINNs that used the former loss have even the smallest mean error, followed by the crosswind loss, the vanilla loss, and $\mathcal{L}^{\mathrm{lrcw}}_{1.0}$. However, the overall smallest error is obtained by a PINN trained with the limited residual loss, followed by one trained with the limited residual with crosswind loss.

After the networks are trained for in total 100.000 epochs, the best PINN is one trained with $\mathcal{L}^{\mathrm{lrcw}}_{1.0}$ (≈ 0.0287). The best PINN trained with $\mathcal{L}^{\mathrm{lr}}_{1.0}$ has an error of around 0.0308, the best one trained with the vanilla loss an error of around 0.0328, and finally, the best one trained with the crosswind loss an error of around 0.0358.

Again, the hyperparameter with the largest influence on the error is the learning rate, as shown in table 5.9, followed by the weight decay parameter. The influence on the loss functionals, especially on the limited residual and limited residual with crosswind loss, is more considerable than for the circular interior layer problem. The other hyperparameters have no significant influence on the errors.

The solution and the error of the best PINN approximation to example 5.7 that is trained with $\mathcal{L}^{\mathrm{lrcw}}_{1.0}$ is depicted in figure 5.16. As in the previous section, the PINN does not reach the maximal value of the exact function and has a slightly smaller minimal value. Furthermore, the boundary layer is not present, so the values at the outflow boundary are not correct. Consequently, the approximation also has the largest absolute errors in these regions. The smallest absolute error is of order $\mathcal{O}(10^{-7})$, which is one order of magnitude than for the best network obtained with the standard loss functional.

To summarize, the crosswind loss works again the worst, and the PINNs trained

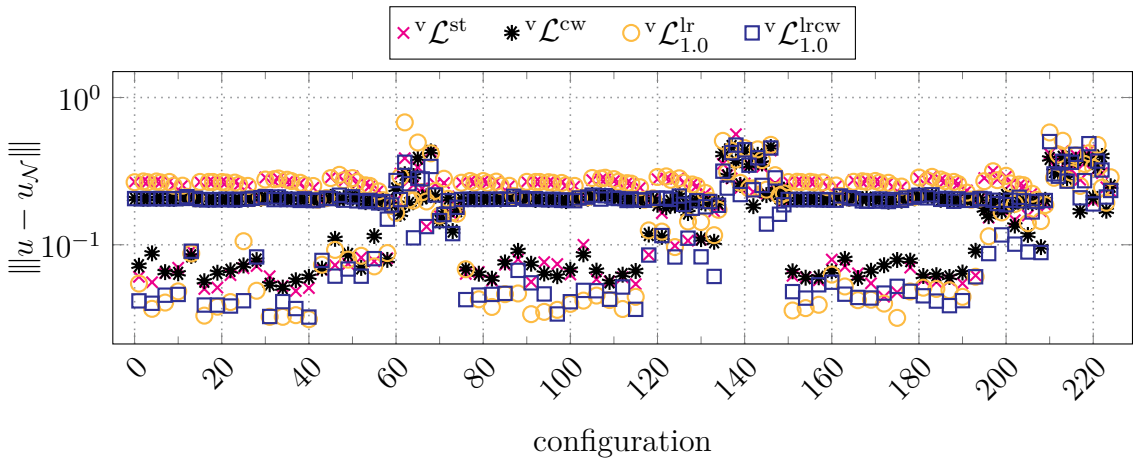


Figure 5.15.: Errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs for all tested configurations of vanilla PINNs with various loss functionals given in equation (5.18) that approximate the solution to example 5.7 from section 5.3.1. The error is averaged over the seeds of the configurations.

with the limited residual and limited residual with crosswind loss are 12.5% and around 6% better than those trained with the vanilla loss in the case of the problem with the boundary layer. If the boundary layer were captured correctly, the results for those two losses might improve since the influence of ξ is particularly significant in the boundary layer because the residuum is expected to be large in that region.

5.3.3. Variational physics-informed neural networks

The final two experiments of this chapter investigate how well *hp*-variational PINNs presented in section 5.2.3 work for convection-dominated convection-diffusion-reaction problems.

For this purpose, 675 MLPs with different configurations of hyperparameters are trained to approximate the solution to examples 5.6 and 5.7, respectively. As in the previous experiment, the hyperparameters given in table 5.1 are used, except for the learning rate where the first two learning rates are not treated. The variational PINNs

Table 5.8.: Mean and minimal value of errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of all vanilla PINNs with various loss functionals given in equation (5.18) that approximate the solution to example 5.7 from section 5.3.1.

	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ ^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ ^{\mathrm{v}}\mathcal{L}^{\mathrm{cw}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ ^{\mathrm{v}}\mathcal{L}^{\mathrm{lr}}_{1.0}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ ^{\mathrm{v}}\mathcal{L}^{\mathrm{lrcw}}_{1.0}$
mean	0.2154	0.1881	0.2171	0.1801
min	0.0444	0.0512	0.0312	0.0323

Table 5.9.: Pearson correlation coefficients between the hyperparameters and the errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of vanilla PINNs with various loss functionals given in equation (5.18) that approximate the solution to example 5.7 from section 5.3.1. Colors indicate the magnitude of the values.

	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{v}}\mathcal{L}^{\mathrm{st}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{v}}\mathcal{L}^{\mathrm{cw}}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{v}}\mathcal{L}^{\mathrm{lr}}_{1.0}$	$\|\|\|u - u_{\mathcal{N}}\|\|\|\ {}^{\mathrm{v}}\mathcal{L}^{\mathrm{lrcw}}_{1.0}$
model size	0.0907	0.0920	0.0740	0.0888
learning rate	0.4299	0.5835	0.4654	0.5295
weight decay λ_{wd}	−0.1973	−0.0726	−0.1952	−0.0523
activation	−0.0241	0.0217	0.0376	−0.0244

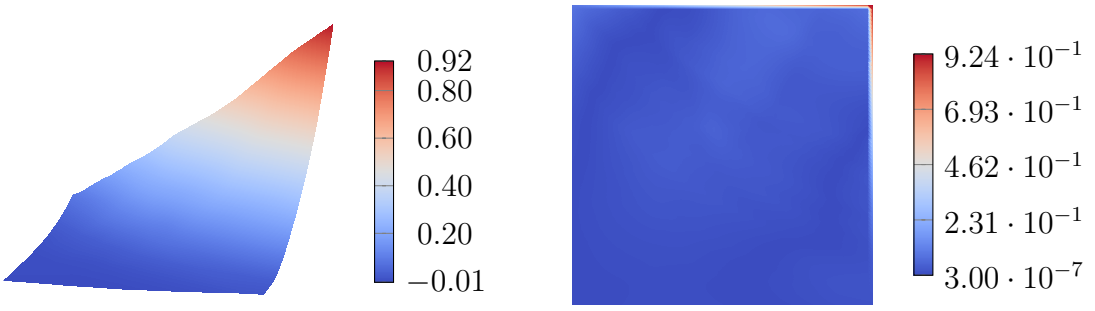
that approximate the solution to example 5.6 are constructed to satisfy the Dirichlet boundary conditions in a hard-constrained manner since the previous experiments indicate that for this problem, hard-constrained PINNs work the best. The *hp*-vPINNs that approximate the solution to example 5.7 learn the Dirichlet boundary conditions during the training, as it is also done for vanilla PINNs. In the following, once more, the superscripts $^{\mathrm{h}}$ and $^{\mathrm{v}}$ encode that either hard-constrained or vanilla *hp*-vPINNs are used.

To be precise, the variational PINNs are optimized to minimize

$$\mathcal{L}^{\mathrm{hpvP}} + \frac{\lambda_{\mathrm{wd}}}{2} \frac{n_{\mathrm{bs}}}{n_{\mathrm{bs}} + N_{\mathrm{D}}} \sum_j w_j^2, \tag{5.19}$$

where $\mathcal{L}^{\mathrm{hpvP}}$ is given in equation (5.14) and the weight decay factor λ_{wd} is given in table 5.1. The factors in front of the individual terms of $\mathcal{L}^{\mathrm{hpvP}}$ are set to $\alpha_{\mathrm{I}}^{\mathrm{hpvP}} = 1$, $\alpha_{\mathrm{D}}^{\mathrm{hpvP}} = 1$ for example 5.6 and $\alpha_{\mathrm{D}}^{\mathrm{hpvP}} = 10,000$ for example 5.7. After the first experiments, it could be observed that for example 5.7 with the choice $\alpha_{\mathrm{D}}^{\mathrm{hpvP}} = 1$ all MLPs end up with an approximation of the constant 0-function, probably due to the boundary conditions. A simple trial-and-error search for a single network guided the concrete choice of the interior weight. In the future, the impact of the choice of the interior and boundary weights in the loss functionals has to be investigated more thoroughly.

For the approximation of the integrals in the interior term of the loss functional, Legendre polynomials of degree at most six in each direction on the reference cell are used, i.e., $P_6([-1,1]^2)$ as defined in equation (5.11). This corresponds exactly to the one-dimensional basis functions depicted in figure 5.5. The unit square is divided into 64 squares of equal size, which means that in total, $25 \cdot 64 = 1,600$ global test functions are deployed. In contrast to [KZK19; KZK21] where a Gauss–Lobatto integration rule is used, in this work, the interior integrals in the loss functional are approximated by a Gauss–Legendre quadrature rule with 10×10 points and weights, i.e., ten per coordinate direction. This corresponds to, in total, 6,400 interior collocation points, compared to 4,096 points used in the previous experiments. However, the boundary



(a) $^{\text{v}}$PINN approximation $u_{\mathcal{N}}$. (b) $|u - u_{\mathcal{N}}|$.

Figure 5.16.: Vanilla PINN approximation $u_{\mathcal{N}}$ with $\mathcal{L}^{\text{lrcw}}_{1.0}$ loss of the solution to example 5.7 after 100,000 epochs and its point wise error $|u - u_{\mathcal{N}}|$ compared with the exact solution u.

integrals are still approximated by 512 equally distanced boundary points as described at the beginning of section 5.3.1. Furthermore, the rest of the set-up described in section 5.3.1 stays the same. This includes that for each trained MLP, the error compared to the exact solution is measured in the $|||\cdot|||$-norm and is averaged over the seeds. The six best configurations after 10,000 epochs are trained for another 90,000 epochs, and again the error is computed and averaged over the seeds to get the final result.

Circular interior layer problem

To begin with, the results for example 5.6 are shown. Again in this experiment, the Dirichlet boundary data is prescribed exactly for all *hp*-vPINNs as indicated by the superscript $^{\text{h}}$.

The errors compared to the exact solution for the hard-constrained variational PINNs for all configurations are shown in figure 5.17. The best variational PINN has an error of around 0.1270 which is one order of magnitude larger than the best PINNs based on the strong formulation of the residual tested in section 5.3.2. After another 90,000 epochs, the error still is with a value of around 0.1166 not significantly smaller than after 10,000 epochs, and stays one order of magnitude larger than the best error in the previous section.

Interestingly, compared to the previous experiments, the learning rate has only a minor influence on the errors of the *hp*-vPINNs as shown in table 5.10. It seems that for variational PINNs the weight decay parameter is most important.

The best approximation and its point wise error compared to the exact solution can be seen in figure 5.18. Notably, the variational PINN has a larger maximal value than the exact solution. The approximation is also slightly deformed and looks more tapered than the exact solution. Consequently, the point wise error is less localized than in the previous experiments. It ranges between 0 at the boundary and around

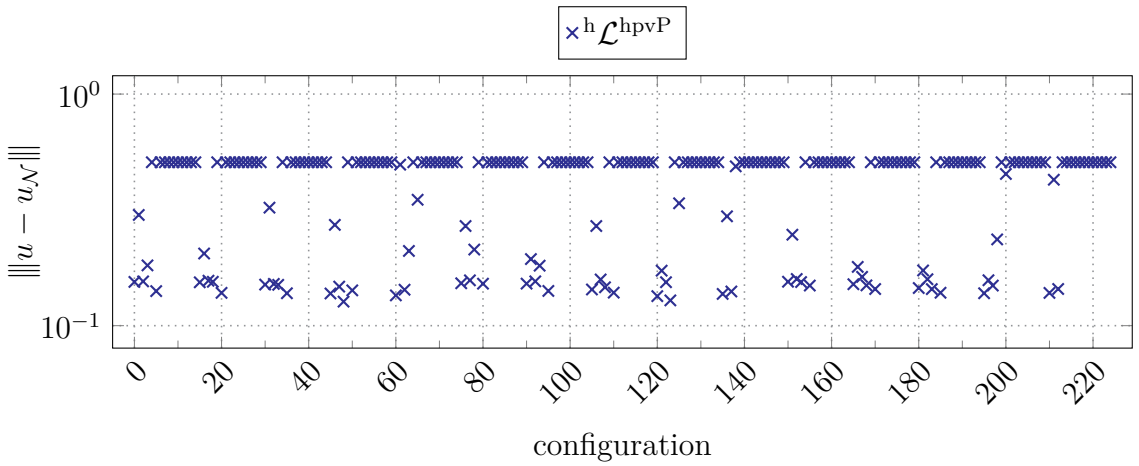


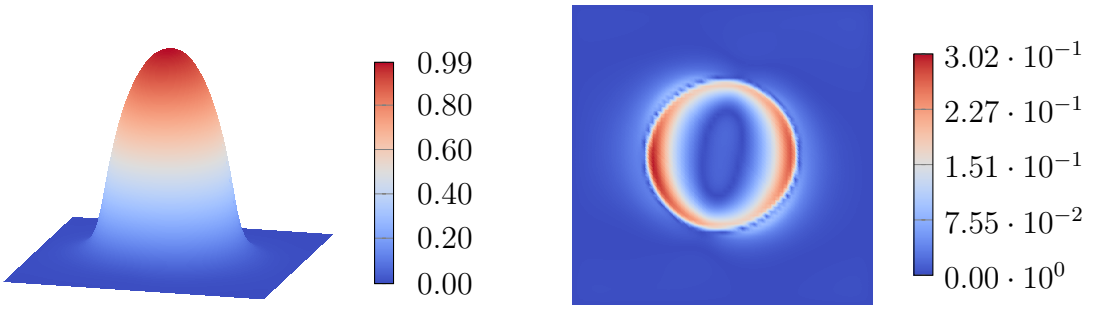
Figure 5.17.: Errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs for all tested configurations of hard-constrained *hp*-vPINNs that approximate the solution to example 5.6 from section 5.3.1. The PINNs are trained to minimize $\mathcal{L}^{\mathrm{hpvP}}$. The error is averaged over the seeds of the configurations.

$3.02 \cdot 10^{-1}$ at the circle. This is also one order of magnitude larger than the best results presented in the previous section.

The results of variational PINNs might improve if more test functions, a higher quadrature rule, or a mesh tailored to the problem are used. However, with the present set-up of the experiment, the variational PINNs are worse than the PINNs investigated in the previous section. This might be counterintuitive since the variational formulation is the canonical one and, in general, the better option compared to the strong form for convection-diffusion-reaction problems.

Table 5.10.: Pearson correlation coefficients between the hyperparameters and the errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of hard-constrained *hp*-vPINNs that approximate the solution to example 5.6 from section 5.3.1. The PINNs are trained to minimize the loss given in equation (5.19). Colors indicate the magnitude of the values.

	$\|\|\|u - u_{\mathcal{N}}\|\|\|$	${}^{\mathrm{h}}\mathcal{L}^{\mathrm{hpvP}}$
model size	0.016	
learning rate	0.112	
weight decay λ_{wd}	0.371	
activation	0.018	



(a) $^{\mathrm{h}}hp$-vPINN approximation $u_{\mathcal{N}}$. (b) $|u - u_{\mathcal{N}}|$.

Figure 5.18.: Hard-constrained hp-vPINN approximation $u_{\mathcal{N}}$ of the solution to example 5.6 after 100,000 epochs and its point wise error $|u - u_{\mathcal{N}}|$ compared with the exact solution u.

Outflow layer problem

Finally, the results of the variational PINNs that approximate the solution to example 5.7 that possesses a boundary layer are presented. For this experiment, the boundary data is again learned with the help of the loss functional and not prescribed exactly. Again, the superscript $^{\mathrm{v}}$ is used to remind of this.

The errors of all configurations after 10,000 epochs are shown in figure 5.19. The error of the configuration with the smallest error is approximately 0.0381, which is approximately 18% worse than the best PINNs tested in section 5.3.2 for this example. After 100,000 epochs, the best configuration shows an error of approximately 0.0354 and is, therefore, 7% smaller than after 10,000 epochs. As before, this is around 19% worse than the best one observed in the previous section.

A single hyperparameter that significantly influences the errors cannot be identified as indicated by the small Pearson correlation coefficients shown in table 5.11. However, as seen in figure 5.19 the results for the concrete configurations vary, which means that particular combinations of hyperparameters work better than others.

Last but not least, the solution whose error is the smallest is depicted together with its point wise error in figure 5.20. The results differ slightly from the best approximation gained in section 5.3.2. As before, neither a boundary layer has formed nor does the approximation has the same values at the outflow boundary as the exact solution. Therefore, again the error is large at the outflow boundary and small in the domain's interior.

To conclude, the results of variational PINNs for this example are worse than PINNs trained with loss functionals based on the strong form of the residual. It could be that a mesh that is adaptively refined towards the boundary layer, more test functions, or a higher quadrature rule would lead to better results. However, this is outside the scope of this chapter and is postponed to future research.

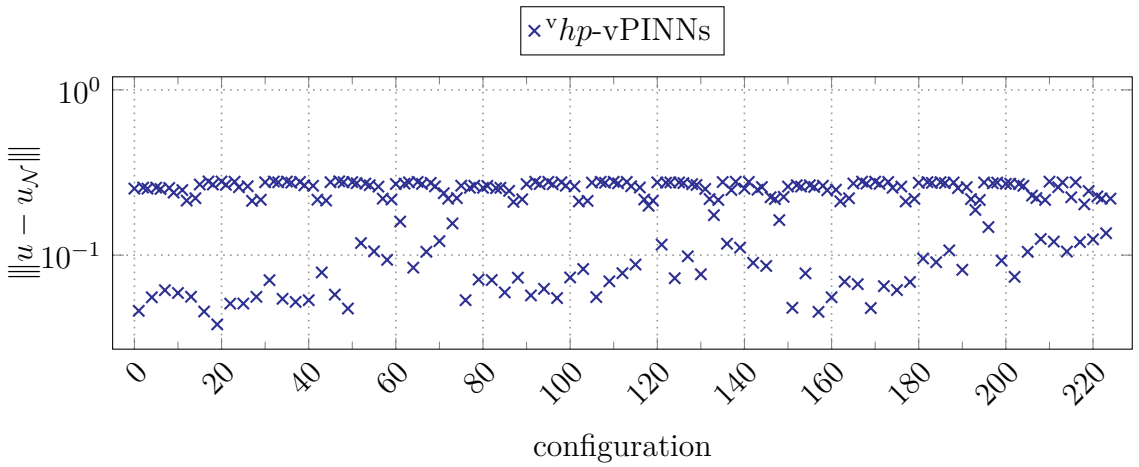


Figure 5.19.: Errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs for all tested configurations of vanilla *hp*-vPINNs that approximate the solution to example 5.7 from section 5.3.1. The error is averaged over the seeds of the configurations.

5.4. Summary

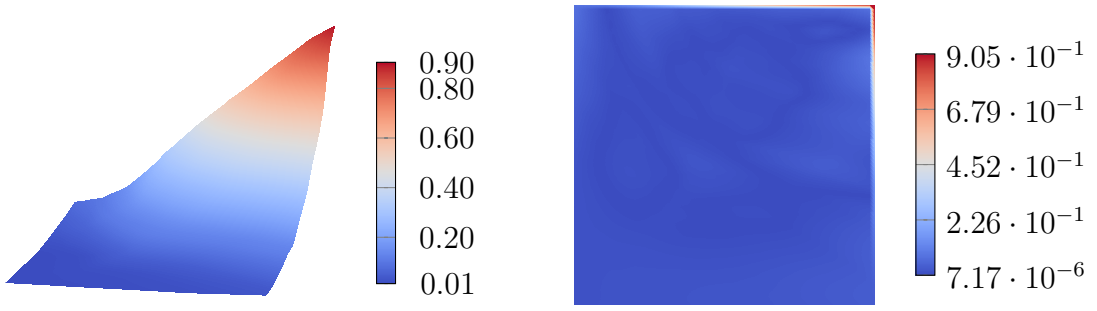
This chapter was dedicated to physics-informed neural network approximations of the solution of convection-dominated convection-diffusion-reaction problems.

To this end, in section 5.1, vanilla PINNs have been introduced, and their loss functional has been derived. Afterwards, the implementation was verified based on a known smooth solution. In section 5.2, several modifications of vanilla PINNs have been presented concerning the treatment of (inflow) Dirichlet boundary data, non-standard loss functionals, and *hp*-variational PINNs. Lastly, these ideas have been tested on two benchmark problems in section 5.3.

From the experiments, it can be concluded that a pretraining of the inflow boundary data does not improve the quality of the solution in terms of the energy error $|||\cdot|||$. Not

Table 5.11.: Pearson correlation coefficients between the hyperparameters and the errors $|||u - u_{\mathcal{N}}|||$ after 10,000 epochs of vanilla *hp*-vPINNs that approximate the solution to example 5.7 from section 5.3.1. The PINNs are trained to minimize the loss given in equation (5.19). Colors indicate the magnitude of the values.

	$|||u - u_{\mathcal{N}}|||$	$^{\mathrm{v}}\mathcal{L}^{\mathrm{hpvP}}$
model size	0.018	
lerning rate	0.068	
weight decay λ_{wd}	−0.099	
activation	−0.014	



(a) $^{\mathrm{v}}hp$-vPINN approximation $u_{\mathcal{N}}$. (b) $|u - u_{\mathcal{N}}|$.

Figure 5.20.: Vanilla hp-vPINN approximation $u_{\mathcal{N}}$ of the solution to example 5.7 after $100,000$ epochs and its point wise error $|u - u_{\mathcal{N}}|$ compared with the exact solution u.

only gets the trained information lost after very few optimization steps of the actual optimization loop, it even reduces the quality of the final approximation. A reason might be that the PINN approximations are overfitted to the inflow boundary data after the pretraining. Moreover, for the problem with interior layers, hard-constrained PINNs worked the best, while for the other benchmark example with a boundary layer, vanilla PINNs showed the best results. Hard-constrained PINNs might have problems with boundary layer problems since they are forced to introduce the needed strong gradient due to the boundary data, whereas vanilla PINNs can neglect the boundary layer and focus on the optimization in the interior.

Furthermore, from the experiments based on the novel loss functionals, it might be concluded that optimization with respect to the crosswind loss leads to worse results than with respect to the standard loss functional. The crosswind loss was introduced in the literature to penalize a smearing of the layers. However, for PINNs, such smearing could not be observed, and adding another term to the loss functional makes the loss landscape more complicated. Consequently, it might be more challenging to find the global maximum. The limited residual and limited residual with crosswind loss worked almost equally well after $100,000$ epochs, and, depending on the problem, between 6% and 12.5% better than vanilla PINNs in terms of the energy error.

Finally, it can be concluded that, surprisingly, variational PINNs work worse than PINNs trained with the novel loss functional for the presented test problems. They had difficulties with approximating the interior layer and, as also their counterparts based on a strong formulation of the residual, did not form the correct boundary layer. The results might be better with a more localized mesh, i.e., a mesh tailored to the problem. However, the dependency on meshes, in general, renders the originally mesh-free method to be mesh dependent. Therefore, it might be more promising to investigate weak adversarial PINNs as described in [Zan+20; DMM22].

These results are the first steps toward systematically investigating PINNs for convection-dominated convection-diffusion-reaction problems. In the future, also

other benchmark problems should be taken into consideration, e.g., the HMM and the Hemker example used in the previous chapters. In contrast to the examples defined in this chapter they are problems without a source term. In this case, PINNs might have more problems approximating problems with boundary layers since the interior and boundary loss might have counteracting roles. Furthermore, the influence of the weights in front of the different terms of the loss functionals needs to be investigated since it could be observed that for some problems, the quality of the results of PINNs depends significantly on the ratio of these weights as also reported in [HT21]. Furthermore, it needs to be tested whether other variational loss functionals or adversarial PINNs are better suited to approximate the solution to convection-diffusion-reaction problems.

A topic neither tackled in this thesis nor the literature so far is discrete maximum principles for PINNs. Even though in most of the optimizations in this work, the PINNs did not break a global maximum principle, in some cases, hard-constrained PINNs produced solutions containing overshoots. Furthermore, there is no guarantee that PINNs respect these principles at all. This behavior needs to be investigated systematically, and ways can be discussed to enforce a discrete maximum principle. One option is to add another term to the loss functional that penalizes unphysical values. Another possible path is to impose a discrete maximum principle through the activation function of the output layer, e.g., using a sigmoid-related activation function.

Last but not least, the best choice of collocation points is still an open question. Adaptively chosen collocation points can be imagined based on modified a-posteriori error estimators of classical finite element methods. However, whereas it might seem natural to use more collocation points in the vicinity of layers, the experiments of [Wan+23] indicate that this is not necessarily the case. This behavior might need to be understood in more detail to facilitate PINNs' quality further.

6. Conclusion and outlook

Since the main ideas of this thesis are described in the previous chapters, it is time to summarize and conclude what can be learned from the present work. This and an outlook on possible future research questions are presented below.

6.1. Conclusion

This work has dealt with steady-state convection-diffusion-reaction problems and numerical methods to approximate their solution. As presented in the first chapter, they are a widely used model to describe the distribution of a scalar quantity inside a flowing medium, and computing an accurate and physically reasonable solution to them in the convection-dominated regime is challenging for most classical numerical methods.

In the following chapters, alternatives to the classical methods have been presented and tested on benchmark problems. To conclude whether these methods are serious alternatives to classical approaches, recall the requirements for a numerical scheme to be suitable for convection-dominated convection-diffusion-reaction problems. To quote from chapter 1, the optimal scheme method should

1. convergence fast towards the exact solution as the mesh becomes finer,
2. be flexible with respect to the used mesh,
3. produce solutions with sharp layers, and
4. compute physically reasonable solutions free of spurious oscillations.

In chapter 2, the model problems of this thesis are derived, and it is proven that under certain conditions, a unique weak solution to these problems exists. After discontinuous Galerkin methods for pure diffusion, convection-reaction, and complete problems have been introduced and examined with respect to their convergence properties, the schemes were successfully tested for a two- and three-dimensional problem with a known solution. The observed convergence rates have been in line with the theory, and it can be concluded that the software package ParMooN is capable of performing DG simulations of steady-state convection-diffusion-reaction problems. As predicted from the theory, DG methods converge fast towards the exact solution, i.e., optimally in the energy norm but suboptimally by a factor of 1 in the L^2-norm. However, optimal L^2-convergence rates can also be observed in practice.

They are known to be flexible with respect to the used mesh and produce solutions with sharp layers. On the other hand, they show significant spurious oscillations in the convection-dominated regime.

The third chapter has presented several known but also novel post-processing methods that can be applied to the discrete solution of DG schemes. As a first step, these methods mark cells where the discrete solution shows unphysical values and, second, replace the solution locally by a polynomial of at most degree one. All these methods have in common that they are computationally cheap, reduce unphysical values and still preserve the mass locally, which is in contrast to just clipping the extrema. It was observed that usually, methods replacing the solution by its integral mean reduced spurious oscillations significantly more than methods that use an affine reconstruction. The latter even increased the oscillations in some of the test cases. In most of the investigated test problems, two novel methods based on examining the jump of the discrete solution were among the best ones. Furthermore, they behaved optimally with respect to reducing the oscillations as much as possible while preserving the mass locally. Even though they reduce the oscillations significantly compared to the plain DG method, they could not remove them entirely in most cases. Altogether, it can be concluded that they improved DG methods concerning the last requirement. Hence, in applications where small oscillations are acceptable, they can be seen as a serious alternative to, e.g., SUPG or SOLD methods. However, if a method that provably preserves maximum principles is desired, other schemes like non-linear AFC methods must be applied.

Deep learning methods are a powerful tool that can be used to support and enhance classical methods. In chapter 4, a way was presented to incorporate multilayer perceptron models into the framework of classical DG methods. To this end, an MLP-based slope limiter was constructed and successfully applied to the DG solution. Even though it has been trained with data from a particular example and polynomial degree, it was also possible to apply it to higher-order solutions of the same and a different problem. On the one hand, the MLP-based slope limiter reduced the spurious oscillations, but on the other hand, it was not better than its classical counterparts. However, this can still be seen as a first step towards other deep learning-supported numerical methods.

Finally, physics-informed neural networks are another deep learning technique that approximates the solution to initial-boundary value problems. In the literature, they have already been applied to various problems, and it was observed that PINNs usually have difficulties approximating the solution to singularly perturbed problems. To gain further insight into this behavior, chapter 5 addressed the question of how well PINNs can approximate the solution to convection-dominated convection-diffusion-reaction problems. For this purpose, after the classical loss functionals of vanilla PINNs have been derived, alternatives were presented that are inspired by cost functionals of optimization problems that dealt with convection-diffusion-reaction problems. These functionals, different treatments of boundary conditions, and another type of

loss functionals based on a variational formulation were tested numerically on two benchmark problems. It was observed that hard-constrained PINNs are better suited in terms of the energy norm for problems with interior layers than those with boundary layers. For the latter, classical treatment of the Dirichlet boundary data is preferable. Moreover, two novel loss functionals worked better than the classical loss functional, and the third worked worse. The variational PINNs showed larger errors than the vanilla PINNs for both test problems. Regarding the requirements mentioned above, PINNs are the most flexible method concerning the geometry treated in this work, and spurious oscillations were not observed in the experiments. However, variational PINNs are mesh-dependent in contrast to classical PINN approaches. Furthermore, the investigated configurations of vanilla PINNs have produced sharp layers only in the case of interior layers. The hard-constrained PINNs showed both sharp interior and boundary layers but had difficulties approximating the solution in the interior close to the boundary layer. At the current stage of development, appropriate finite element methods are still better suited to solve convection-dominated convection-diffusion-reaction problems.

In a nutshell, in cases where small spurious oscillations are acceptable, DG methods treated with one of the slope-limiting techniques presented in this work can be a serious alternative to classical methods. PINNs might also be considered if a parametrized solution has to be calculated, noisy data needs to be incorporated into the solution, or some parts of the underlying governing equations are not fully known.

6.2. Outlook

As concluded above, the journey of finding an optimal scheme for approximating the solution to convection-dominated convection-diffusion-reaction problems is ongoing. There are still open questions, and almost all methods treated in this thesis allow for improvements.

As seen in chapter 3, some slope-limiting methods already mark all cells where spurious oscillations pollute the solution. Hence, significant improvements concerning the marking criterion might not be expected. Replacing the solution locally with its integral mean already reduces the oscillations as much as possible. Unfortunately, the linear reconstructions work worse, but since their reconstruction uses a higher polynomial degree, the accuracy measured in certain norms is less reduced. Therefore, paths that use a higher-order reconstruction might be found while reducing the oscillations. To this end, ideas might be borrowed from the community dealing with hyperbolic transport problems, including, e.g., (W)ENO reconstructions and total variation diminishing schemes. Furthermore, in this work, the influence of these techniques on the error to the exact solution is not investigated, but this topic should be treated in the future. For problems without a known solution, an AFC method computed on a very fine grid might serve as a reference solution. Last but not least,

what happens for time-dependent problems still needs to be investigated. In that case, it might be crucial not to mark too many cells since this has a non-negligible influence on computational time. However, the algorithms still must mark as many cells as needed to reduce the oscillations since the approximation at a particular time step influences the discrete solution of the following steps.

Since incorporating deep learning techniques into the framework of classical numerical schemes has just begun, there is much potential for new ideas. Concerning the MLP-based slope limiters of this thesis, it is mandatory in the future to obtain the labels in the training data without using classical limiters. They cannot become better than their classical companions if they are still based on classical methods. A possible way in that direction is to generate data where the labels are chosen "by hand", e.g., by using analytically known functions of which points are known a-priori where the solution has a steep gradient or spurious oscillations. Furthermore, finding other feature sets serving as input to neural networks might be helpful. First of all, all user-dependent features of classical slope limiters need to be removed to have parameter-free limiters. Moreover, reducing the input space might improve the efficiency of the networks. It is still an open question what features of the solution most definitely indicate the presence of spurious oscillations. It could be investigated whether the degrees of freedom of the solution are suitable to this end since they encode the complete discrete solution. However, the same basis functions and order must always be used for this approach. Another way is to use CNN-based networks together with local values of the discrete solution. This approach removes the dependency on the basis functions but still incorporates almost all details of the discrete solution. Having a representative feature set at hand is also preferable in another conceivable idea. Since optimal parameters of SUPG and SOLD methods are not known a-priori, a desirable application might be to have a neural network that predicts optimal parameters based on these optimal local features. To generate a labeled data set, optimal parameters must be generated beforehand. A way that does not rely on a-priori known optimal parameters might be to use reinforcement learning strategies. In this context, a so-called *reward* function is needed, which rates an action of the network. To this end, the norm of the residuum of the discrete solution might be used as it is done in loss functionals of PINNs.

PINNs' most relevant drawback is still the lack of their theoretical understanding. Improving the theoretical knowledge about neural networks, especially PINNs, might be mandatory to design better algorithms. In the context of convection-dominated convection-diffusion-reaction problems, it would be interesting to investigate how well PINNs work for problems without a driving source. Furthermore, the influence of the weights in the loss functional can be analyzed, as well as the choice of the collocation points with which the integrals of the loss functional are approximated. In the latter regard, constructing a-posteriori error estimators inspired by classical residual-based estimators from finite element methods might be essential to sharpen the understanding. With such, it should be possible to choose the collocation points

adaptively and, in particular, adapt them to the problem.

As said above, many open questions still exist that pave the way to a fascinating future of research concerning convection-dominated convection-diffusion-reaction problems.

A. Mathematical background

For the sake of brevity, weak derivatives, Lebesgue, and Sobolev spaces are not presented. For a detailed introduction to those topics, please refer to [Eva10, chapter 5; BS02, chapter 1]. Nevertheless, some essential theorems used in this work are presented below.

Theorem A.1 (Integration by parts)**.** *Let $\Omega \subset \mathbb{R}^d$, $1 \leq d \in \mathbb{N}$, be a bounded Lipschitz domain. Then, for any $\mathbf{f} \in [H^1(\Omega)]^d$ and $g \in H^1(\Omega)$, it holds*

$$\int_\Omega \mathbf{f} \cdot \nabla g \, \mathrm{d}\boldsymbol{x} + \int_\Omega \operatorname{div}(\mathbf{f})\, g \, \mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \mathbf{f} \cdot \boldsymbol{n} g \, \mathrm{d}s,$$

where $\boldsymbol{n}$ is the outer unit normal vector defined almost everywhere on $\partial\Omega$ [Neč12, chapter 2, lemma 4.2], and $\mathbf{f}$ along $\partial\Omega$ has to be understood in the sense of traces.

Proof. See, e.g., [Neč12, chapter 3, theorem 1.1] or [Wil19, corollary 4.4.1]. □

Theorem A.2 (Gauss Divergence)**.** *Under the assumptions of theorem A.1 it holds, for any $\mathbf{f} \in [H^1(\Omega)]^d$,*

$$\int_\Omega \operatorname{div}(\mathbf{f}) \, \mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \mathbf{f} \cdot \boldsymbol{n} \, \mathrm{d}s,$$

where $\boldsymbol{n}$ is the outer unit normal vector to $\partial\Omega$.

Proof. Follows directly from theorem A.1 with $g = 1$. □

Lemma A.3 (Hölder's inequality)**.** *For all $1 \leq p, q \leq \infty$ such that $1/p + 1/q = 1$, and all $v \in L^p(\Omega)$ and $w \in L^q(\Omega)$ it holds $vw \in L^1(\Omega)$ and*

$$\int_\Omega vw \, \mathrm{d}\boldsymbol{x} \leq \|v\|_{L^p(\Omega)} \|w\|_{L^q(\Omega)}.$$

Proof. See, e.g., [Eva10, B.2.e]. □

Theorem A.4 (Friedrichs' inequality)**.** *Let $\Omega \subset \mathbb{R}^d$, $1 \leq d \in \mathbb{N}$ be a bounded Lipschitz domain with some subset $\Gamma_\mathrm{D} \subset \partial\Omega$ with positive surface measure, i.e., $\Gamma_\mathrm{D} \neq \emptyset$. Then it holds, for any $u \in H^1_{\mathrm{D},0}(\Omega) \coloneqq \{\, v \in H^1(\Omega) \;:\; v|_{\Gamma_\mathrm{D}} = 0 \text{ in the sense of traces} \,\}$,*

$$\|u\|_{L^p(\Omega)} \leq C_\mathrm{Fr} \|\nabla u\|_{L^p(\Omega)},$$

where $0 < C_\mathrm{Fr}(\Omega, \Gamma_\mathrm{D}, p) < \infty$ is a real number.

Proof. See, e.g., [Wil19, equation (4.12)] and [Bra07, 1.5 Poincaré–Friedrichs Inequality and 1.6 Remark]. □

Lemma A.5 (Lax–Milgram)**.** *Let $(V, \|\cdot\|_V)$ be a Hilbert space and $a : V \times V \to \mathbb{R}$ a bilinear form for which there exist two constants $\alpha, \beta \in \mathbb{R}$ such that*

$$\begin{aligned} |a(v,w)| &\leq \alpha \|v\|_V \|w\|_V && \text{for any } v, w \in V, && (\textit{boundedness}), \\ a(v,v) &\geq \beta \|v\|_V^2, && \text{for any } v \in V && (\textit{coercivity}). \end{aligned}$$

Then, for any bounded linear functional $F : V \to \mathbb{R}$ on V, it exists a unique $u \in V$ with

$$a(u,v) = F(v) \qquad \text{for all } v \in V.$$

Proof. The proof can be found in standard textbooks, e.g., by Brenner and Scott [BS02, p. 62]. □

Corollary A.6 (Discrete Lax–Milgram)**.** *Let $(V_h, \|\cdot\|_h)$ be a finite-dimensional Hilbert space and $a_h : V_h \times V_h \to \mathbb{R}$ a bilinear form for which there exists a constant $\beta_h \in \mathbb{R}$ such that*

$$a(v_h, v_h) \geq \beta_h \|v_h\|_h^2, \qquad \text{for any } v_h \in V_h \qquad (\textit{discrete coercivity}).$$

Then, for any linear functional $F_h : V_h \to \mathbb{R}$ on V_h, it exists a unique $u_h \in V_h$ with

$$a(u_h, v_h) = F_h(v_h) \qquad \text{for all } v_h \in V_h.$$

Proof. This follows directly from the Lax–Milgram lemma A.5; see, e.g., [DF15, corollary 1.7]. □

Bibliography

[Aba+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. Software available from `www.tensorflow.org/`. Google Research, Sept. 2015.

[AH20] Rémi Abgrall and Maria Han Veiga. "Neural Network-Based Limiter with Transfer Learning". In: *Communications on Applied Mathematics and Computation* (Sept. 2020). ISSN: 2096-6385. DOI: `10.1007/s42967-020-00087-1`.

[Alh07] Khalid Alhumaizi. "Flux-Limiting Solution Techniques for Simulation of Reaction–Diffusion–Convection System". In: *Communications in Nonlinear Science and Numerical Simulation* 12.6 (Sept. 2007), pp. 953–965. ISSN: 1007-5704. DOI: `10.1016/j.cnsns.2005.11.005`.

[Ant+16] Paola F. Antonietti, Andrea Cangiani, Joe Collis, Zhaonan Dong, Emmanuil H. Georgoulis, Stefano Giani, and Paul Houston. "Review of Discontinuous Galerkin Finite Element Methods for Partial Differential Equations on Complicated Domains". In: *Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations*. Ed. by Gabriel R. Barrenechea, Franco Brezzi, Andrea Cangiani, and Emmanuil H. Georgoulis. Vol. 114. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, Apr. 2016, pp. 281–310. ISBN: 978-3-319-41640-3. DOI: `10.1007/978-3-319-41640-3_9`.

[ACD23] Amirhossein Arzani, Kevin W. Cassel, and Roshan M. D'Souza. "Theory-Guided Physics-Informed Neural Networks for Boundary Layer Problems with Singular Perturbation". In: *Journal of Computational Physics* 473 (Jan. 2023), p. 111768. ISSN: 0021-9991. DOI: `10.1016/j.jcp.2022.111768`.

[Ata18] Abdon Atangana. *Fractional Operators with Constant and Variable Order with Application to Geo-Hydrology*. Academic Press, 2018. 396 pp. ISBN: 978-0-12-809670-3. DOI: `10.1016/C2015-0-05711-2`.

[Aug+11] Matthias Augustin, Alfonso Caiazzo, André Fiebach, Jürgen Fuhrmann, Volker John, Alexander Linke, and Rudolf Umla. "An Assessment of Discretizations for Convection-Dominated Convection–Diffusion Equations". In: *Computer Methods in Applied Mechanics and Engineering* 200.47-48

(Nov. 2011), pp. 3395–3409. ISSN: 00457825. DOI: 10.1016/j.cma.2011.08.012.

[AM09] Blanca Ayuso and L. Donatella Marini. "Discontinuous Galerkin Methods for Advection-Diffusion-Reaction Problems". In: *SIAM Journal on Numerical Analysis* 47.2 (Jan. 2009), pp. 1391–1420. ISSN: 0036-1429. DOI: 10.1137/080719583.

[BJK16] Gabriel R. Barrenechea, Volker John, and Petr Knobloch. "Analysis of Algebraic Flux Correction Schemes". In: *SIAM Journal on Numerical Analysis* 54.4 (Jan. 2016), pp. 2427–2451. ISSN: 0036-1429. DOI: 10.1137/15M1018216.

[BJK23] Gabriel R. Barrenechea, Volker John, and Petr Knobloch. *Finite Element Methods Respecting the Discrete Maximum Principle for Convection-Diffusion Equations*. May 2023. DOI: 10.48550/arXiv.2204.07480. arXiv: 2204.07480. preprint, accepted in SIAM Review.

[Bar+18a] Gabriel R. Barrenechea, Volker John, Petr Knobloch, and Richard Rankin. "A Unified Analysis of Algebraic Flux Correction Schemes for Convection–Diffusion Equations". In: *SeMA Journal* 75.4 (Dec. 2018), pp. 655–685. ISSN: 2254-3902. DOI: 10.1007/s40324-018-0160-6.

[Bar+18b] Gabriel R. Barrenechea, Volker John, Petr Knobloch, and Richard Rankin. "A Unified Analysis of Algebraic Flux Correction Schemes for Convection–Diffusion Equations". In: *SeMA Journal* 75.4 (Dec. 2018), pp. 655–685. ISSN: 2254-3902. DOI: 10.1007/s40324-018-0160-6.

[BFM19] Andrea Beck, David Flad, and Claus-Dieter Munz. "Deep Neural Networks for Data-Driven LES Closure Models". In: *Journal of Computational Physics* 398 (Dec. 2019), p. 108910. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.108910.

[Bec+20] Andrea D. Beck, Jonas Zeifang, Anna Schwarz, and David G. Flad. "A Neural Network Based Shock Detection and Localization Approach for Discontinuous Galerkin Methods". In: *Journal of Computational Physics* 423 (Dec. 2020), p. 109824. ISSN: 00219991. DOI: 10.1016/j.jcp.2020.109824.

[Bei+13] Lourenco Beirão da Veiga, Franco Brezzi, Andrea Cangiani, Gianmarco Manzini, L. Donatella Marini, and Alessandro Russo. "Basic Principles of Virtual Element Methods". In: *Mathematical Models and Methods in Applied Sciences* 23.01 (Jan. 2013), pp. 199–214. ISSN: 0218-2025. DOI: 10.1142/S0218202512500492.

[Bei+16] Lourenco Beirão da Veiga, Franco Brezzi, Luisa Donatella Marini, and Alessandro Russo. "Virtual Element Method for General Second-Order Elliptic Problems on Polygonal Meshes". In: *Mathematical Models and Methods in Applied Sciences* 26.04 (Apr. 2016), pp. 729–750. ISSN: 0218-2025. DOI: 10.1142/S0218202516500160.

[BLV17] Lourenco Beirão da Veiga, Carlo Lovadina, and Giuseppe Vacca. "Divergence Free Virtual Elements for the Stokes Problem on Polygonal Meshes". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 51.2 (Mar. 2017), pp. 509–535. ISSN: 0764-583X. DOI: 10.1051/m2an/2016032.

[Ben12] Yoshua Bengio. "Practical Recommendations for Gradient-Based Training of Deep Architectures". In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Vol. 7700. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 437–478. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_26.

[Bra07] Dietrich Braess. *Finite Elements Theory, Fast Solvers, and Applications in Elasticity Theory*. Cambridge: Cambridge University Press, 2007. XVII + 365. ISBN: 978-0-521-70518-9.

[BS02] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Red. by J. E. Marsden, L. Sirovich, M. Golubitsky, and S. S. Antman. 2nd ed. Vol. 15. Texts in Applied Mathematics. New York, NY, USA: Springer, New York, NY, 2002. XV+363. ISBN: 978-1-4757-3660-1. DOI: 10.1007/978-1-4757-3658-8.

[BMS04] Franco Brezzi, L. Donatella Marini, and Endre Süli. "Discontinuous Galerkin Methods for First-Order Hyperbolic Problems". In: *Mathematical Models and Methods in Applied Sciences* 14.12 (Dec. 2004), pp. 1893–1903. ISSN: 0218-2025. DOI: 10.1142/S0218202504003866.

[Cai+21] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. "Physics-Informed Neural Networks (PINNs) for Fluid Mechanics: A Review". In: *Acta Mechanica Sinica* 37.12 (Dec. 2021), pp. 1727–1738. ISSN: 1614-3116. DOI: 10.1007/s10409-021-01148-1.

[Cia02] Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems*. Classics in Applied Mathematics 40. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2002. XXVI+530. ISBN: 978-0-89871-514-9. DOI: 10.1137/1.9780898719208.

[Coc03] B. Cockburn. "Discontinuous Galerkin Methods". In: *ZAMM* 83.11 (Nov. 2003), pp. 731–754. ISSN: 0044-2267. DOI: 10.1002/zamm.200310088.

[CKS00] Bernardo Cockburn, George E. Karniadakis, and Chi-Wang Shu, eds. *Discontinuous Galerkin Methods: Theory, Computation and Applications*. Red. by M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick. Vol. 11. Lecture Notes in Computational Science and Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. XI + 473. ISBN: 978-3-642-64098-8. DOI: 10.1007/978-3-642-59721-3.

[CS98] Bernardo Cockburn and Chi-Wang Shu. "The Runge–Kutta Discontinuous Galerkin Method for Conservation Laws V: Multidimensional Systems". In: *Journal of Computational Physics* 141.2 (Apr. 1998), pp. 199–224. ISSN: 00219991. DOI: 10.1006/jcph.1998.5892.

[CS01] Bernardo Cockburn and Chi-Wang Shu. "Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems". In: *Journal of Scientific Computing* 16.3 (Sept. 2001), pp. 173–261. ISSN: 1573-7691. DOI: 10.1023/A:1012873910884.

[Cuo+22] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. "Scientific Machine Learning Through Physics–Informed Neural Networks: Where We Are and What's Next". In: *Journal of Scientific Computing* 92.3 (July 2022), p. 88. ISSN: 1573-7691. DOI: 10.1007/s10915-022-01939-z.

[Cyb89] George Cybenko. "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pp. 303–314. ISSN: 1435-568X. DOI: 10.1007/BF02551274.

[Dav04] Timothy A. Davis. "Algorithm 832: UMFPACK V4.3—an Unsymmetric-Pattern Multifrontal Method". In: *ACM Transactions on Mathematical Software* 30.2 (June 2004), pp. 196–199. ISSN: 0098-3500. DOI: 10.1145/992200.992206.

[DMM22] Tim De Ryck, Siddhartha Mishra, and Roberto Molinaro. *wPINNs: Weak Physics Informed Neural Networks for Approximating Entropy Solutions of Hyperbolic Conservation Laws*. July 2022. arXiv: 2207.08483 [cs, math]. preprint.

[Dev22] TensorFlow Developers. *TensorFlow*. Version v2.11.0. Nov. 2022. DOI: 10.5281/zenodo.7604226.

[dWol+21] Taco de Wolff, Hugo Carrillo, Luis Martí, and Nayat Sanchez-Pi. *Towards Optimally Weighted Physics-Informed Neural Networks in Ocean Modelling*. June 2021. DOI: 10.48550/arXiv.2106.08747. arXiv: 2106.08747 [physics]. preprint.

[DEL16] Daniele A. Di Pietro, Alexandre Ern, and Simon Lemaire. "A Review of Hybrid High-Order Methods: Formulations, Computational Aspects, Comparison with Other Methods". In: *Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations*. Ed. by Gabriel R. Barrenechea, Franco Brezzi, Andrea Cangiani, and Emmanuil H. Georgoulis. Vol. 114. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2016, pp. 205–236. ISBN: 978-3-319-41640-3. DOI: 10.1007/978-3-319-41640-3_7.

[DE12] Daniele Antonio Di Pietro and Alexandre Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*. 1st ed. Vol. 69. Mathématiques et Applications. Berlin, Heidelberg: Springer Verlag Berlin Heidelberg, 2012. XVIII+384. ISBN: 978-3-642-22979-4. DOI: 10.1007/978-3-642-22980-0.

[DT18] Daniele Antonio Di Pietro and Roberta Tittarelli. "An Introduction to Hybrid High-Order Methods". In: *Numerical Methods for PDEs*. Ed. by Daniele Antonio Di Pietro, Alexandre Ern, and Luca Formaggia. Vol. 15. Cham: Springer International Publishing, 2018, pp. 75–128. ISBN: 978-3-319-94675-7. DOI: 10.1007/978-3-319-94676-4_4.

[DP94] Gamini Dissanayake and Nhan Phan-Thien. "Neural-Network-Based Approximations for Solving Partial Differential Equations". In: *Communications in Numerical Methods in Engineering* 10.3 (1994), pp. 195–201. ISSN: 1099-0887. DOI: 10.1002/cnm.1640100303.

[DFS02] Vit Dolejší, Miroslav Feistauer, and Christoph Schwab. "On Discontinuous Galerkin Methods for Nonlinear Convection-Diffusion Problems and Compressible Flow". In: *Proceedings of EQUADIFF 10*. Mathematica Bohemica. Vol. 127. Prague, 2002, pp. 163–179. DOI: 10.21136/MB.2002.134171.

[DF15] Vít Dolejší and Miloslav Feistauer. *Discontinuous Galerkin Method: Analysis and Applications to Compressible Flow*. 1st ed. Vol. 48. Springer Series in Computational Mathematics. Cham: Springer International Publishing, 2015. XIV+572. ISBN: 978-3-319-19266-6. DOI: 10.1007/978-3-319-19267-3.

[DFS03] Vít Dolejší, Miloslav Feistauer, and Christoph Schwab. "On Some Aspects of the Discontinuous Galerkin Finite Element Method for Conservation Laws". In: *Mathematics and Computers in Simulation* 61.3-6 (Jan. 2003), pp. 333–346. ISSN: 03784754. DOI: 10.1016/S0378-4754(02)00087-3.

[DBB23] Nathan Doumèche, Gérard Biau, and Claire Boyer. *Convergence and Error Analysis of PINNs*. May 2023. DOI: 10.48550/arXiv.2305.01240. arXiv: 2305.01240 [math, stat]. preprint.

[Dro+21] Jérôme Droniou, Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. "Non-Conforming Finite Elements on Polytopal Meshes". In: *Polyhedral Methods in Geosciences*. Ed. by Daniele Antonio Di Pietro, Luca Formaggia, and Roland Masson. SEMA SIMAI Springer Series. Cham: Springer International Publishing, 2021, pp. 1–35. ISBN: 978-3-030-69363-3. DOI: 10.1007/978-3-030-69363-3_1.

[Erm92] Donald L. Ermak. "Dense-gas dispersion advection-diffusion model". In: *1992 JANNAF Safety and Environmental Subcommittee meeting* (July 1992).

[Eva10] Lawrence C. Evans. *Partial Differential Equations*. 2nd ed. Vol. 19. Graduate Studies in Mathematics. Providence, RI, USA: American Mathematical Society, 2010. XXI+749. ISBN: 978-0-8218-4974-3.

[EGH00] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. "Finite Volume Methods". In: *Handbook of Numerical Analysis*. Vol. 7. Solution of Equation in $\mathbb{R}^n$ (Part 3), Techniques of Scientific Computing (Part 3). Elsevier, Jan. 2000, pp. 713–1018. DOI: 10.1016/S1570-8659(00)07005-8.

[FN15] Wolfgang Fennel and Thomas Neumann. *Introduction to the Modelling of Marine Ecosystems*. 2nd ed. Boston: Elsevier, 2015. 331 pp. ISBN: 978-0-444-63363-7. DOI: 10.1016/C2013-0-13520-9.

[Fox+21] B. Fox-Kemper et al. "Ocean, Cryosphere and Sea Level Change". In: *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Ed. by V. Masson-Delmotte et al. Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press, 2021, pp. 1211–1362. DOI: 10.1017/9781009157896.011.

[FJ21] Derk Frerichs and Volker John. "On Reducing Spurious Oscillations in Discontinuous Galerkin (DG) Methods for Steady-State Convection–Diffusion Equations". In: *Journal of Computational and Applied Mathematics* 393 (Sept. 2021), p. 113487. ISSN: 0377-0427. DOI: 10.1016/j.cam.2021.113487.

[FHJ22] Derk Frerichs-Mihov, Linus Henning, and Volker John. *Using deep neural networks for detecting spurious oscillations in discontinuous Galerkin solutions of convection-dominated convection-diffusion equations*. Berlin, Dec. 2022. DOI: 10.20347/WIAS.PREPRINT.2986. preprint, submitted to Journal of Scientific Computing.

[FJ22] Derk Frerichs-Mihov and Volker John. "On a Technique for Reducing Spurious Oscillations in DG Solutions of Convection–Diffusion Equations". In: *Applied Mathematics Letters* 129 (July 2022), p. 107969. ISSN: 08939659. DOI: 10.1016/j.aml.2022.107969.

[FLL11] Jürgen Fuhrmann, Alexander Linke, and Hartmut Langmach. "A Numerical Method for Mass Conservative Coupling between Fluid Flow and Solute Transport". In: *Applied Numerical Mathematics* 61.4 (Apr. 2011), pp. 530–553. ISSN: 0168-9274. DOI: 10.1016/j.apnum.2010.11.015.

[Gan+16] Sashikumaar Ganesan, Volker John, Gunar Matthies, Raviteja Meesala, Abdus Shamim, and Ulrich Wilbrandt. "An Object Oriented Parallel Finite Element Scheme for Computations of PDEs: Design and Implementation". In: *2016 IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW)*. 2016 IEEE 23rd International Conference on High Performance Computing Workshops (HiPCW). Hyderabad, India: IEEE, Dec. 2016, pp. 106–115. ISBN: 978-1-5090-5774-0. DOI: 10.1109/HiPCW.2016.023.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 249–256.

[GSV22a] Antônio Tadeu Azevedo Gomes, Larissa Miguez da Silva, and Frédéric Valentin. "Improving Boundary Layer Predictions Using Parametric Physics-Aware Neural Networks". In: *High Performance Computing*. Ed. by Philippe Navaux, Carlos J. Barrios H., Carla Osthoff, and Ginés Guerrero. Communications in Computer and Information Science. Cham: Springer International Publishing, 2022, pp. 90–102. ISBN: 978-3-031-23821-5. DOI: 10.1007/978-3-031-23821-5_7.

[GSV22b] Antônio Tadeu Azevedo Gomes, Larissa Miguez da Silva, and Frédéric Valentin. *Physics-Aware Neural Networks for Boundary Layer Linear Problems*. July 2022. DOI: 10.48550/arXiv.2208.12559. arXiv: 2208.12559 [cs, math]. preprint.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. XVI+781.

[HT21] QiZhi He and Alexandre M. Tartakovsky. "Physics-Informed Neural Network Method for Forward and Backward Advection-Dispersion Equations". In: *Water Resources Research* 57.7 (2021), e2020WR029479. ISSN: 1944-7973. DOI: 10.1029/2020WR029479.

[Hem96] Pieter W. Hemker. "A singularly perturbed model problem for numerical computation". In: *Journal of Computational and Applied Mathematics* 76.1–2 (Dec. 1996). Citation Key: hemkerSingularlyPerturbedModel1996, pp. 277–285. ISSN: 03770427. DOI: 10.1016/S0377-0427(96)00113-6.

[HH19] Catherine F. Higham and Desmond J. Higham. "Deep Learning: An Introduction for Applied Mathematicians". In: *SIAM Review* 61.4 (Jan. 2019), pp. 860–891. ISSN: 0036-1445. DOI: 10.1137/18M1165748.

[Hoc+21] Antoine Hochet, Rémi Tailleux, Till Kuhlbrodt, and David Ferreira. "Global Heat Balance and Heat Uptake in Potential Temperature Coordinates". In: *Climate Dynamics* 57.7 (Oct. 2021), pp. 2021–2035. ISSN: 1432-0894. DOI: 10.1007/s00382-021-05832-7.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

[Hou+22] Qingzhi Hou, Zewei Sun, Li He, and Alireza Karemat. "Orthogonal Grid Physics-Informed Neural Networks: A Neural Network-Based Simulation Tool for Advection–Diffusion–Reaction Problems". In: *Physics of Fluids* 34.7 (July 2022), p. 077108. ISSN: 1070-6631. DOI: 10.1063/5.0095536.

[HSS02] Paul Houston, Christoph Schwab, and Endre Süli. "Discontinuous *hp*-Finite Element Methods for Advection-Diffusion-Reaction Problems". In: *SIAM Journal on Numerical Analysis* 39.6 (Jan. 2002), pp. 2133–2163. ISSN: 0036-1429. DOI: 10.1137/S0036142900374111.

[Hub+15] Markus Huber, Remi Tailleux, David Ferreira, Till Kuhlbrodt, and Jonathan Gregory. "A Traceable Physical Calibration of the Vertical Advection-Diffusion Equation for Modeling Ocean Heat Uptake". In: *Geophysical Research Letters* 42.7 (Apr. 2015), pp. 2333–2341. ISSN: 00948276. DOI: 10.1002/2015GL063383.

[HMM86] Thomas J.R. Hughes, Michel Mallet, and Akira Mizukami. "A New Finite Element Formulation for Computational Fluid Dynamics: II. Beyond SUPG". In: *Computer Methods in Applied Mechanics and Engineering* 54.3 (1986), pp. 341–355. ISSN: 0045-7825. DOI: 10.1016/0045-7825(86)90110-6.

[Ins23a] Clay Mathematics Institute. *Navier–Stokes Equation*. May 2023. URL: www.claymath.org/millennium-problems/navier%E2%80%93stokes-equation (visited on 05/28/2023).

[Ins23b] Clay Mathematics Institute. *The Millennium Prize Problems*. May 2023. URL: www.claymath.org/millennium-problems/millennium-prize-problems (visited on 05/28/2023).

[Izq22] Sergio Izquierdo. *CppFlow*. Version v2.0.0. https://github.com/serizba/cppflow/tree/v2.0.0. Sept. 2022.

[Jha20] Abhinav Jha. "Numerical Algorithms for Algebraic Stabilizations of Scalar Convection-Dominated Problems". PhD thesis. Berlin: Free University of Berlin, Nov. 2020. XIII+172.

[JJ20] Abhinav Jha and Volker John. "On Basic Iteration Schemes for Nonlinear AFC Discretizations". In: *Boundary and Interior Layers, Computational and Asymptotic Methods BAIL 2018*. Ed. by Gabriel R. Barrenechea and John Mackenzie. Vol. 135. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2020, pp. 113–128. ISBN: 978-3-030-41799-4. DOI: 10.1007/978-3-030-41800-7_7.

[JK07] Volker John and Petr Knobloch. "On Spurious Oscillations at Layers Diminishing (SOLD) Methods for Convection–Diffusion Equations: Part I – A Review". In: *Computer Methods in Applied Mechanics and Engineering* 196.17-20 (Mar. 2007), pp. 2197–2215. ISSN: 00457825. DOI: 10.1016/j.cma.2006.11.013.

[JK08] Volker John and Petr Knobloch. "On Spurious Oscillations at Layers Diminishing (SOLD) Methods for Convection–Diffusion Equations: Part II – Analysis for P_1 and Q_1 Finite Elements". In: *Computer Methods in Applied Mechanics and Engineering* 197.21-24 (Apr. 2008), pp. 1997–2014. ISSN: 00457825. DOI: 10.1016/j.cma.2007.12.019.

[JK13] Volker John and Petr Knobloch. "Adaptive Computation of Parameters in Stabilized Methods for Convection-Diffusion Problems". In: *Numerical Mathematics and Advanced Applications 2011 - Proceedings of ENUMATH 2011*. 9th European Conference on Numerical Mathematics and Advanced Applications. Ed. by Andrea Cangiani, Ruslan L. Davidchack, Emmanuil Georgoulis, Alexander N. Gorban, Jeremy Levesley, and Michael V. Tretyakov. Vol. 1. Leicester: Springer Berlin, Heidelberg, 2013, pp. 275–283. ISBN: 978-3-642-33133-6. DOI: 10.1007/978-3-642-33134-3_30.

[JKN18] Volker John, Petr Knobloch, and Julia Novo. "Finite Elements for Scalar Convection-Dominated Equations and Incompressible Flow Problems: A Never Ending Story?" In: *Computing and Visualization in Science* 19.5-6 (Dec. 2018), pp. 47–63. ISSN: 1432-9360. DOI: 10.1007/s00791-018-0290-5.

[JKS11] Volker John, Petr Knobloch, and Simona B. Savescu. "A Posteriori Optimization of Parameters in Stabilized Methods for Convection–Diffusion Problems – Part I". In: *Computer Methods in Applied Mechanics and Engineering* 200.41-44 (Apr. 2011), pp. 2916–2929. ISSN: 00457825. DOI: 10.1016/j.cma.2011.04.016.

[JKW23] Volker John, Petr Knobloch, and Ulrich Wilbrandt. "A Posteriori Optimization of Parameters in Stabilized Methods for Convection–Diffusion Problems — Part II". In: *Journal of Computational and Applied Mathematics* 428 (Aug. 2023), p. 115167. ISSN: 0377-0427. DOI: 10.1016/j.cam.2023.115167.

[JMT97] Volker John, Joseph M. Maubach, and Lutz Tobiska. "Nonconforming Streamline-Diffusion-Finite-Element-Methods for Convection-Diffusion Problems". In: *Numerische Mathematik* 78.2 (Dec. 1997), pp. 165–188. ISSN: 0029-599X. DOI: 10.1007/s002110050309.

[Jos+21] Subodh M. Joshi, Thivin Anandh, Bhanu Teja, and Sashikumaar Ganesan. "On the Choice of Hyper-Parameters of Artificial Neural Networks for Stabilized Finite Element Schemes". In: *International Journal of Advances in Engineering Sciences and Applied Mathematics* 13.2 (Sept. 2021), pp. 278–297. ISSN: 0975-5616. DOI: 10.1007/s12572-021-00306-9.

[Kan07] Guido Kanschat. *Discontinuous Galerkin Methods for Viscous Incompressible Flow*. 1st ed. Advances in Numerical Mathematics. Wiesbaden: Teubner Research, Dt. Univ.-Verl, 2007. 183 pp. ISBN: 978-3-8350-4001-4.

[Kar+21] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. "Physics-Informed Machine Learning". In: *Nature Reviews Physics* 3.6 (June 2021), pp. 422–440. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00314-5.

[KZK21] Ehsan Kharazmi, Zhongqiang Zhang, and George E.M. Karniadakis. "*hp*-VPINNs: Variational physics-informed neural networks with domain decomposition". In: *Computer Methods in Applied Mechanics and Engineering* 374 (Feb. 2021), p. 113547. ISSN: 00457825. DOI: 10.1016/j.cma.2020.113547.

[KZK19] Ehsan Kharazmi, Zhongqiang Zhang, and George Em Karniadakis. *VPINNs: Variational Physics-Informed Neural Networks For Solving Partial Differential Equations*. Nov. 2019. DOI: 10.48550/arXiv.1912.00873. arXiv: 1912.00873 [physics, stat]. preprint.

[KZ20] Reza Khodayi-Mehr and Michael Zavlanos. "VarNet: Variational Neural Networks for the Solution of Partial Differential Equations". In: *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. Learning for Dynamics and Control. PMLR, July 2020, pp. 298–307.

[KB14] Diederik P. Kingma and Jimmy Lei Ba. "Adam: A Method For Stochastic Optimization". In: *ICLR 2015*. International Conference on Learning Representations (ICLR). arXiv, 2014, p. 13. DOI: 10.48550/ARXIV.1412.6980.

[KLS19] Petr Knobloch, Petr Lukáš, and Pavel Solin. "On Error Indicators for Optimizing Parameters in Stabilized Methods". In: *Advances in Computational Mathematics* 45.4 (Feb. 2019), pp. 1853–1862. ISSN: 1019-7168. DOI: 10.1007/s10444-019-09662-4.

[Kri+21] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. "Characterizing Possible Failure Modes in Physics-Informed Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 26548–26560.

[Kub21] Miroslav Kubat. *An Introduction to Machine Learning*. 3rd ed. Cham: Springer Nature Switzerland AG, 2021. XVIII+458. ISBN: 978-3-030-81934-7. DOI: 10.1007/978-3-030-81935-4.

[Lag+23] Laura Laghi, Enrico Schiassi, Mario De Florio, Roberto Furfaro, and Domiziano Mostacci. "Physics-Informed Neural Networks for 1-D Steady-State Diffusion-Advection-Reaction Equations". In: *Nuclear Science and Engineering* 0.0 (Feb. 2023), pp. 1–31. ISSN: 0029-5639. DOI: 10.1080/00295639.2022.2160604.

[Lan78] Rolf Lange. "ADPIC—A Three-Dimensional Particle-in-Cell Model for the Dispersal of Atmospheric Pollutants and Its Comparison to Regional Tracer Studies". In: *Journal of Applied Meteorology and Climatology* 17.3 (Mar. 1978), pp. 320–329. DOI: 10.1175/1520-0450(1978)017<0320:ATDPIC>2.0.CO;2.

[Lee+18] Ádám Leelőssy, István Lagzi, Attila Kovács, and Róbert Mészáros. "A Review of Numerical Models to Predict the Atmospheric Dispersion of Radionuclides". In: *Journal of Environmental Radioactivity* 182 (Feb. 2018), pp. 20–33. ISSN: 0265-931X. DOI: 10.1016/j.jenvrad.2017.11.009.

[Lee+16] Ádám Leelőssy, Tamás Mona, Róbert Mészáros, István Lagzi, and Ágnes Havasi. "Eulerian and Lagrangian Approaches for Modelling of Air Quality". In: *Mathematical Problems in Meteorological Modelling*. Ed. by András Bátkai, Petra Csomós, István Faragó, András Horányi, and Gabriella Szépszó. Mathematics in Industry. Cham: Springer International Publishing, 2016, pp. 73–85. ISBN: 978-3-319-40157-7. DOI: 10.1007/978-3-319-40157-7_5.

[Lev+12] Sydney Levitus et al. "World Ocean Heat Content and Thermosteric Sea Level Change (0–2000 m), 1955–2010". In: *Geophysical Research Letters* 39.10 (2012). ISSN: 1944-8007. DOI: 10.1029/2012GL051106.

[Liu+19] Yang Liu, Yutong Lu, Yueqing Wang, Dong Sun, Liang Deng, Fang Wang, and Yan Lei. "A CNN-based Shock Detection Method in Flow Visualization". In: *Computers & Fluids* 184 (Apr. 2019), pp. 1–9. ISSN: 00457930. DOI: 10.1016/j.compfluid.2019.03.022.

[Lu+21] Lu Lu, Raphaël Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. "Physics-Informed Neural Networks with Hard Constraints for Inverse Design". In: *SIAM Journal on Scientific Computing* 43.6 (Jan. 2021), B1105–B1132. ISSN: 1064-8275. DOI: 10.1137/21M1397908.

[MLR21] Nils Margenberg, Christian Lessig, and Thomas Richter. "Structure Preservation for the Deep Neural Network Multigrid Solver". In: *ETNA - Electronic Transactions on Numerical Analysis* 56 (2021), pp. 86–101. ISSN: 1068-9613. DOI: 10.1553/etna_vol56s86.

[MBH23] Rambod Mojgani, Maciej Balajewicz, and Pedram Hassanzadeh. "Lagrangian PINNs: A Causality-Conforming Solution to Failure Modes of Physics-Informed Neural Networks". In: *Computer Methods in Applied Mechanics and Engineering* 404 (Feb. 2023), p. 115810. ISSN: 00457825. DOI: 10.1016/j.cma.2022.115810. arXiv: 2205.02902 [physics, stat].

[Mor+20] Nathaniel R. Morgan, Svetlana Tokareva, Xiaodong Liu, and Andrew Morgan. "A Machine Learning Approach for Detecting Shocks with High-Order Hydrodynamic Methods". In: *AIAA Scitech 2020 Forum*. AIAA Scitech 2020 Forum. Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2020. ISBN: 978-1-62410-595-1. DOI: 10.2514/6.2020-2024.

[Mor19] Keith William Morton. *Revival: Numerical Solution Of Convection-Diffusion Problems (1996)*. 1st ed. Boca Raton: CRC Press, 2019. 286 pp. ISBN: 978-1-138-10578-2. DOI: 10.1201/9780203711194.

[Neč12] Jindřich Nečas. *Direct Methods in the Theory of Elliptic Equations*. Springer Monographs in Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. XVI + 372. ISBN: 978-3-642-10454-1. DOI: 10.1007/978-3-642-10455-8.

[Pet91] Todd E. Peterson. "A Note on the Convergence of the Discontinuous Galerkin Method for a Scalar Hyperbolic Equation". In: *SIAM Journal on Numerical Analysis* 28.1 (Feb. 1991), pp. 133–140. ISSN: 0036-1429. DOI: 10.1137/0728006.

[Pica] PickPik. *clear drinking glass with blue liquid*. URL: https://www.pickpik.com/ink-water-water-glass-liquid-drip-flow-123481 (visited on 05/29/2023).

[Picb] PickPik. *person holding white ceramic mug pouring latte art*. URL: https://www.pickpik.com/beverage-caffeine-coffee-cup-drink-milk-119779 (visited on 05/29/2023).

[Picc] PickPik. *stock photo of factory plant*. URL: https://www.pickpik.com/industry-power-energy-industrial-plant-factory-110038 (visited on 05/29/2023).

[Pin99] Allan Pinkus. "Approximation Theory of the MLP Model in Neural Networks". In: *Acta Numerica* 8 (Jan. 1999), pp. 143–195. ISSN: 1474-0508. DOI: 10.1017/S0962492900002919.

[Pip17] Joachim Piprek, ed. *Handbook of Optoelectronic Device Modeling and Simulation: Lasers, Modulators, Photodetectors, Solar Cells, and Numerical Methods, Vol. 2*. 1st ed. Boca Raton: CRC Press, Sept. 2017. 886 pp. ISBN: 978-1-315-15231-8. DOI: 10.4324/9781315152318.

[RPK19] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations". In: *Journal of Computational Physics* 378 (Feb. 2019), pp. 686–707. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.10.045.

[RH18] Deep Ray and Jan S. Hesthaven. "An Artificial Neural Network as a Troubled-Cell Indicator". In: *Journal of Computational Physics* 367 (Aug. 2018), pp. 166–191. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.04.029.

[RH19] Deep Ray and Jan S. Hesthaven. "Detecting Troubled-Cells on Two-Dimensional Unstructured Grids Using a Neural Network". In: *Journal of Computational Physics* 397 (Nov. 2019), p. 108845. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.07.043.

[RH73] W. H. Reed and T. R. Hill. "Triangular Mesh Methods for the Neutron Transport Equation". In: *Proceedings of the American Nuclear Society*. National Topical Meeting on Mathematical Models and Computational Techniques for Analysis of Nuclear Systems, Ann Arbor, Michigan, USA. Vol. LA-UR-73-479; CONF-730414-2. Los Alamos Scientific Laboratory, Los Alamos, NM, USA: Los Alamos Scientific Laboratory, Oct. 1973.

[Riv08] Béatrice Rivière. *Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation*. Vol. 35. Frontiers in Applied Mathematics. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, Jan. 2008. XXII+190. ISBN: 978-0-89871-656-6. DOI: 10.1137/1.9780898717440.

[RC04] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. 2nd ed. Springer Texts in Statistics. New York, NY: Springer, 2004. XXX+649. ISBN: 978-1-4419-1939-7. DOI: 10.1007/978-1-4757-4145-2.

[RST08] Hans-Görg Roos, Martin Stynes, and Lutz Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion-Reaction and Flow Problems*. 2nd ed. Vol. 24. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer-Verlag, 2008. XIV+604. ISBN: 978-3-540-34466-7. DOI: 10.1007/978-3-540-34467-4.

[SJ15] Shakila Saad and Maheran Mohd Jaffar. "Review on the Structural Approach of the Black-Scholes Model". In: *AIP Conference Proceedings*. International Conference on Mathematics, Engineering & Industrial Applications 2014. Vol. 1660. AIP Publishing, May 2015. DOI: 10.1063/1.4915717.

[Saa+22] Mohammad Hossein Saadat, Blazhe Gjorgiev, Laya Das, and Giovanni Sansavini. *Neural Tangent Kernel Analysis of PINN for Advection-Diffusion Equation*. Nov. 2022. DOI: 10.48550/arXiv.2211.11716. arXiv: 2211.11716 [physics, stat]. preprint.

[Sar21] Iqbal H. Sarker. "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions". In: *SN Computer Science* 2.6 (Aug. 2021), p. 420. ISSN: 2661-8907. DOI: 10.1007/s42979-021-00815-1.

[Sch15] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003.

[SC21] Rômulo Montalvão Silva and Alvaro L. G. A. Coutinho. "PINNs for Parametric Incompressible Newtonian Flows". In: *Proceedings of the XLII Ibero-Latin-American Congress on Computational Methods in Engineering and III Pan-American Congress on Computational Mechanic*. CILAMCE-PANACM. Rio de Janeiro, Brazil, Nov. 2021.

[SJ02] Scott A. Socolofsky and Gerhard H. Jirka. *Environmental Fluid Mechanics Part I: Mass Transfer and Diffusion*. 2nd ed. Karlsruhe, Germany: Karlsruhe Institute of Technology, 2002. DOI: 10.5445/IR/1542004.

[VA18] Maria Han Veiga and Rémi Abgrall. "Towards a General Stabilisation Method for Conservation Laws Using a Multilayer Perceptron Neural Network: 1D Scalar and System of Equations". In: European Conference on Computational Mechanics and VII European Conference on Computational Fluid Dynamics. Glasgow, Scotland: ECCM, June 2018, pp. 2525–2550.

[vWR21] Henry von Wahl and Thomas Richter. "Using a Deep Neural Network to Predict the Motion of Underresolved Triangular Rigid Bodies in an Incompressible Flow". In: *International Journal for Numerical Methods in Fluids* 93.12 (2021), pp. 3364–3383. ISSN: 1097-0363. DOI: 10.1002/fld.5037.

[WR17] Haohan Wang and Bhiksha Raj. *On the Origin of Deep Learning*. Mar. 2017. DOI: 10.48550/arXiv.1702.07800. arXiv: 1702.07800 [cs, stat]. preprint.

[Wan+23] Yufeng Wang, Cong Xu, Min Yang, and Jin Zhang. *Less Emphasis on Difficult Layer Regions: Curriculum Learning for Singularly Perturbed Convection-Diffusion-Reaction Problems*. Mar. 2023. DOI: 10.48550/arXiv.2210.12685. arXiv: 2210.12685 [cs, math]. preprint.

[Wil19] Ulrich Wilbrandt. *Stokes–Darcy Equations: Analytic and Numerical Analysis*. Advances in Mathematical Fluid Mechanics. Cham: Springer International Publishing, 2019. vii+212. ISBN: 978-3-030-02903-6. DOI: 10.1007/978-3-030-02904-3.

[Wil+17] Ulrich Wilbrandt et al. "ParMooN—A Modernized Program Package Based on Mapped Finite Elements". In: *Computers & Mathematics with Applications* 74.1 (July 2017), pp. 74–88. ISSN: 08981221. DOI: 10.1016/j.camwa.2016.12.020.

[YS20] Li Yang and Abdallah Shami. "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice". In: *Neurocomputing* 415 (Nov. 2020), pp. 295–316. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.07.061.

[Zan+20] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. "Weak Adversarial Networks for High-Dimensional Partial Differential Equations". In: *Journal of Computational Physics* 411 (June 2020), p. 109409. ISSN: 00219991. DOI: 10.1016/j.jcp.2020.109409.

[ZHT22] Yifei Zong, QiZhi He, and Alexandre M. Tartakovsky. *Physics-Informed Neural Network Method for Parabolic Differential Equations with Sharply Perturbed Initial Conditions*. Aug. 2022. DOI: 10.48550/arXiv.2208.08635. arXiv: 2208.08635 [cs, math]. preprint.

Zusammenfassung

Diese Arbeit widmet sich sogenannten Slopelimitern (wörtlich „Anstiegsbegrenzern“) und Techniken des maschinellen Lernens zur Approximation der Lösung von stationären Konvektions-Diffusions-Reaktionsproblemen. Im konvektionsdominierten Fall weist die Lösung in der Regel Grenzschichten auf, d.h. kleine Bereiche, in denen ein steiler Gradient vorherrscht. Aus der Literatur ist bekannt, dass es für viele klassische numerische Methoden schwierig ist, die Lösung in diesen Regionen zu approximieren und dass die Lösung oft von unphysikalischen Werten, so genannten Störoszillationen, überlagert ist.

Zuerst werden die Modellgleichungen hergeleitet und es wird untersucht, unter welchen Bedingungen eine eindeutige schwache Lösung für sie existiert. Anschließend werden symmetrische, unvollständige und nicht-symmetrische innere Straf-Galerkin-Methoden (DG) eingeführt, um die exakte Lösung im reinen Diffusions-, Konvektions-Reaktions- und im vollständigen Fall numerisch zu approximieren. A-priori Fehlerabschätzungen werden bereitgestellt und numerisch verifiziert.

As erstes Hauptthema werden mehrere Slopelimiter aus der Literatur und verschiedene neuartige Methoden vorgestellt. Diese Nachbearbeitungsverfahren zielen darauf ab, automatisch Regionen zu erkennen, in denen die diskrete Lösung unphysikalische Werte aufweist, und die Lösung lokal durch ein Polynom niedrigeren Grades zu approximieren. Der erste wichtige Beitrag dieser Arbeit besteht darin, dass zwei der neuen Methoden die Störoszillationen deutlich und besser als die bisher bekannten Methoden reduzieren können, während die Masse lokal erhalten bleibt, wie in zwei Benchmarkproblemen mit zwei verschiedenen Diffusionskoeffizienten gezeigt ist.

Der zweite Schwerpunkt besteht darin, zu zeigen, wie Techniken des maschinellen Lernens in den Rahmen der klassischen Finite-Elemente-Methoden integriert werden können. Der zweite wichtige Beitrag dieser Arbeit ist die Konstruktion eines auf maschinellem Lernen basierenden Slopelimiters. Er wird mit Daten einer DG-Methode niedrigerer Ordnung für ein bestimmtes Problem trainiert und auf eine DG-Methode höherer Ordnung für dasselbe und ein anderes Problem angewendet. Er reduziert die Oszillationen im Vergleich zur standart DG-Methode erheblich, ist aber etwas schlechter als die klassischen Slopelimiter.

Der dritte wichtige Beitrag bezieht sich auf Physik-informierte neuronale Netzwerke (PINNs) zur Annäherung an die Lösung der Modellgleichungen. In Bezug auf Konvektions-Diffusions-Reaktionsgleichungen werden verschiedene Möglichkeiten zur Einbeziehung der Dirichlet-Randdaten, mehrere Kostenfunktionale, die im Zusammenhang mit PINNs neu sind, und Variations-PINNs vorgestellt. Sie werden numerisch getestet und verglichen. Die neuartigen Kostenfunktionale verbessern den Fehler im Vergleich zum klassischen PINN-Ansatz. Es wird festgestellt, dass die Näherungen frei von Oszillationen sind und mit inneren Grenzschichten zurechtkommen, aber Probleme beim Approximieren von Randgrenzschichten haben.

Danksagung

Die finalen Zeilen sind geschrieben, die letzten Korrekturen gemacht und die Druckerschwärze steht bereit. Zeit also, sich bei allen zu bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Zuallererst gilt mein Dank meinem Betreuer, Volker John. Er gab mir die Möglichkeit erste Schritte in der Forschung und der akademischen Welt zu tätigen. Von seinen immerwährend konstruktiven Vorschlägen, inspirierenden Diskussionen und seiner Fähigkeit, Dinge zu einem Abschluss zu bringen, konnte ich während all der Zeit enorm profitieren. Herzlichen Dank dafür!

Ein großer Dank gilt auch all meinen Kolleg:innen aus der FG3. Allen voran möchte ich mich bei Christian Merdon und Ulrich Wilbrandt bedanken, nicht nur, aber auch für das detaillierte Korrekturlesen. Ohne Christian wäre ich vermutlich nie ans Institut gekommen und er ermöglichte es mir, meine ersten Vorträge auf verschiedenen Konferenzen zu halten. Ulrich half mir besonders dabei, einen Einstieg in C++ und ParMooN zu finden. Sein unermüdliches Finden von Bugs sowie seine unzähligen Erläuterungen bezüglich ParMooNs, erleichterten mein Leben ungemein. Jede Diskussion und jedes Gespräch half mir dabei, Struktur in meine Gedanken zu bringen. Ich kann getrost sagen, ohne Ulrich wäre diese Arbeit nur schwer möglich gewesen. Nicht vergessen bleiben soll auch die oft ungesehene und geräuschlose, aber gleichzeitig sehr geschätzte Arbeit von Marion Lawrenz und Imke Weitkamp. Sie gaben mir die Freiheit, mich hauptsächlich der Forschung und weniger der Bürokratie zu widmen. Ich danke euch allen sehr!

Ganz besonderer Dank gilt auch allen anderen Menschen am Institut, die mich gerne ins Büro haben kommen lassen. Insbesondere zu nennen sind Camilla, Leonie, Alexandra, Mina, Stefanie, und die 3DZ-Gruppe um Zeina, Daniel und Dilara. Ihr standet immer mit Rat und Tat zur Seite, fülltet das Institut mit Leben und von jedwedem Austausch mit euch, habe ich stets profitiert. Es war mir immer eine Freude, gemeinsam mit euch am Institut zu sein. Ihr seid einfach klasse!

Natürlich sollen auch meine tollen Freundinnen und Freunde außerhalb der Arbeit Erwähnung finden. Joshua, Stephen, Janice und Julian, ich danke euch für alle gemeinsamen Erlebnisse und Erinnerungen, die wir teilen. Für alle schönen Abende, eure Unterstützung und eure Bestärkungen, alle Zweifel, die ich teilen durfte und einfach dafür, dass ihr meine Freunde seid. Vielen Dank für alles!

Ohne meine Eltern, Imke und Johannes, wäre ich heute nicht, wo ich bin. Sie gaben mir Liebe und Geborgenheit, unterstützten mich bei allem, gaben mir metaphorisch einen auf den Deckel, wenn ich über die Stränge schlug und ermöglichten es mir, zu studieren und mein Leben so zu leben, wie ich es jetzt tue. Was habe ich doch für ein

Glück, euch als Eltern zu haben!

Abschließend bleibt nur noch mich bei meiner von Herzen geliebten Frau, Klara, zu bedanken. Aus tiefstem Herzen bin ich dir dankbar für deine nicht in Gold aufwiegbare Unterstützung, das Rückenfreihalten in stressigen Phasen und deine bedingungslose Liebe. Du gibst mir Kraft, wenn ich sie brauche, bist die Stütze meines Lebens und sorgst dafür, dass ich nie den Mut verliere. Worte können nicht beschreiben, wie dankbar ich dir bin. Ohne dich wäre all das niemals möglich gewesen!

Selbstständigkeitserklärung

Ich, Derk Frerichs-Mihov, erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Berlin, den 6. Juli 2023 .....
(Unterschrift)