Weierstraß-Institut
für Angewandte Analysis und Stochastik

Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

On loss functionals for physics-informed neural networks for convection-dominated convection-diffusion problems

Derk Frerichs-Mihov[1], Linus Henning[2], Volker John[1,2]

submitted: December 11, 2023

[1] Weierstrass Institute
Mohrenstr. 39
10117 Berlin
Germany
E-Mail: derk.frerichs-mihov@wias-berlin.de
volker.john@wias-berlin.de

[2] Freie Universität Berlin
Department of Mathematics and Computer Science
Arnimallee 6
14195 Berlin
Germany
E-Mail: linus.henning@fu-berlin.de

No. 3063
Berlin 2023

2020 *Mathematics Subject Classification.* 65N99, 68T07.

Key words and phrases. Convection-diffusion problems, convection-dominated regime, physics-informed neural networks, loss functionals.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

On loss functionals for physics-informed neural networks for convection-dominated convection-diffusion problems

Derk Frerichs-Mihov, Linus Henning, Volker John

Abstract

In the convection-dominated regime, solutions of convection-diffusion problems usually possesses layers, which are regions where the solution has a steep gradient. It is well known that many classical numerical discretization techniques face difficulties when approximating the solution to these problems. In recent years, physics-informed neural networks (PINNs) for approximating the solution to (initial-)boundary value problems received a lot of interest. In this work, we study various loss functionals for PINNs that are novel in the context of PINNs and are especially designed for convection-dominated convection-diffusion problems. They are numerically compared to the vanilla and a hp-variational loss functional from the literature based on two benchmark problems whose solutions possess different types of layers. We observe that the best novel loss functionals reduce the $L^2(\Omega)$ error by 17.3% for the first and 5.5% for the second problem compared to the methods from the literature.

1 Introduction

The distribution of a scalar quantity like temperature or concentration inside a flowing medium can be modeled by convection-diffusion-reaction problems. In the steady-state case, which is considered in this work, they are formulated as follows: Find a sufficiently smooth solution $u : \overline{\Omega} \to \mathbb{R}$ such that

$$-\varepsilon \Delta u + \boldsymbol{b} \cdot \nabla u + cu = f \qquad \text{in } \Omega, \qquad\qquad u = g \qquad \text{along } \partial\Omega, \tag{1}$$

where $\Omega \subset \mathbb{R}^d$, $d \in \{ 2, 3 \}$, is a bounded domain with polyhedral Lipschitz boundary $\partial\Omega$, $0 < \varepsilon \in \mathbb{R}$ is a positive diffusion coefficient, $\boldsymbol{b} \in [W^{1,\infty}(\Omega)]^d$ models the convection field, $c \in L^\infty(\Omega)$ denotes the reaction coefficient, $f \in L^2(\Omega)$ describes external sources or sinks, and $g \in H^{1/2}(\partial\Omega)$ prescribes the value of u at the boundary. For the sake of simplifying the presentation, we only assume Dirichlet boundary conditions in this work. However, incorporating Neumann boundary conditions is straightforward both in the continuous problem and the numerical algorithms.

In practical applications, often the convection is orders of magnitudes stronger than the diffusion. In this so-called convection-dominated regime that is mathematically described by $\varepsilon \ll L\|\boldsymbol{b}\|_{[W^{1,\infty}(\Omega)]^d}$ with a characteristic length scale of the problem $0 < L \in \mathbb{R}$, the solution usually possesses layers, which are small subregions where the solution has a steep gradient. Usually the layers are so small that they cannot be resolved on affordable meshes. Thus, one encounters the typical situation of a multiscale problem, namely that (the most) important features of the solution cannot be resolved, they are subgrid scales. Because of the multiscale character it is challenging for many numerical methods to accurately approximate the solution in a vicinity of layers; see, e.g., [47, 28, 3, 26, 15]. Furthermore, the construction and numerical analysis of improved methods is an active field of research, see [4].

Within the last two decades, deep learning techniques started to reveal their full potential, and they have been already successfully applied to facilitate classical numerical methods; see, e.g., [46, 5, 50,

DOI 10.20347/WIAS.PREPRINT.3063 Berlin 2023

41, 18]. Moreover, deep neural networks can also be deployed instead of classical methods to directly approximate the solution of initial-boundary value problems (IBVPs). One of the earliest publications is [12] and dates back to 1994. However, it has been only in recent times that their true potential was fully grasped when they were rediscovered in a series of papers starting with [45]. Since then, numerous contributions and enhancements have been made, leading to their application in a diverse range of problems. For a comprehensive overview of PINNs and their variations, we refer to, e.g., [32, 7, 8] and the references therein.

As previously mentioned, PINNs are an alternative numerical approach to approximate solutions to IBVPs. The feasibility of this approach is based in the renowned universal approximation theorem, which states that a feed-forward multilayer perceptron model, possessing a linear activation function in the output layer and at least one hidden layer with any bounded non-polynomial activation function, can achieve an arbitrary level of accuracy in approximating any Borel measurable function that maps from a finite-dimensional space to another one if it has enough nodes in the hidden layer [22, p. 194]; see also [44] for an overview. Moreover, the theorem's scope is extended to encompass not only the function itself but also its derivatives up to order $m \in \mathbb{N}$, provided that the function is m times continuously differentiable [44, Section 4]. Generalizations of the theorem include a broader class of activation functions, incorporating the ReLU function, among others [22, p. 195]. However, it is noteworthy that the ability of multilayer perceptron models to approximate functions does not necessarily imply that typical algorithms will converge towards such a network.

In a nutshell, the main idea involves approximating the solution to (initial-)boundary value problems using multilayer perceptron models by minimizing a loss functional that typically encompasses the residual of the governing equations, the boundary, and initial conditions, and possibly other underlying physical laws or measured data [32]. The computation of the loss functional involves the use of so-called *collocation points*, which are points selected within the domain and on the (space-time-)boundary where the loss functional is evaluated. Two key advantages of PINNs are their mesh-free nature and their adaptability to various initial-boundary value problems. This versatility allows them to handle different geometries and problem types [8]. Furthermore, they offer a seamless framework for incorporating (noisy) data from measurements and can be employed to solve inverse problems to uncover unknown terms in the governing equations [32]. An additional strength of PINNs lies in their ability to handle the same initial-boundary value problem for different parameters, such as distinct diffusion coefficients in convection-diffusion-reaction problems. Once a PINN is trained to represent solutions for various parameters, it can efficiently provide solutions for previously unknown parameters with minimal additional effort. This sets them apart from many classical methods that often struggle to meet all these requirements [32].

However, PINNs also come with certain drawbacks. One significant disadvantage is the scarcity of theoretical understanding regarding their convergence towards the solution of the continuous problem; see, e.g., [32, 8] and the related references. Consequently, unlike classical methods, there are only few guarantees about the quality of the approximations. This limitation is partially due to the stochastic nature of the training process, making it difficult to ensure convergence to a global minimum. Another challenge is the selection of appropriate hyperparameters for the underlying multilayer perceptron models (MLPs). Currently, determining these hyperparameters typically relies on (automated) trial-and-error techniques, as finding optimal values remains an open problem. For insights on hyperparameter optimization approaches, we refer to [52] and references therein.

Despite these unresolved challenges, PINNs have demonstrated successful applications in diverse fields, especially when dealing with solutions that are in some sense smooth. However, they face difficulties when approximating the solution to perturbed problems [38]. The following literature survey

reveals that only a limited number of publications focus on PINNs for convection-dominated convection-diffusion-reaction problems in more than one space dimension. One notable reference, [35], introduces a variational form of the loss functional, which is tested on time-dependent one- and two-dimensional convection-diffusion-reaction problems with diffusion coefficients ranging between 0.1 and 0.001. The study examines various test cases, including one with an interior layer, and shows that the trained PINNs can reasonably capture the solution behavior. Reference [23] explores PINNs in the context of time-dependent advection-dispersion problems in one and two dimensions, particularly for moderate Péclet numbers of order $\mathcal{O}(1)$, $\mathcal{O}(10)$, and $\mathcal{O}(100)$. The results demonstrate that the choice of weights significantly influences the solution quality, with the PINN approximation being comparable in accuracy to the SUPG method [23]. These findings are extended in [54], which examines sharply perturbed initial conditions and proposes a normalized form of equations and PINNs, along with criteria for choosing the weights of the loss functionals. For two-dimensional convection-diffusion problems, [20, 21] report that PINNs can handle diffusion coefficients around $\mathcal{O}(10^{-1})$, but the accuracy of the approximation significantly deteriorates for smaller magnitudes. Nonetheless, PINNs are well-suited for approximating the parametrized solution with varying diffusion coefficients. In another approach [2], the authors introduce boundary-layer PINNs, inspired by the expansion theory of singularly perturbed boundary layer problems. This method generates two PINNs: one for the boundary layer and another for the remaining domain, which are then combined. The boundary-layer PINNs exhibit superior performance compared to standard PINNs when tested on one- and two-dimensional convection-dominated convection-diffusion problems. However, their approach has a higher computational cost since it needs to optimize two networks and requires a priori knowledge of the boundary layer's position. In the preprint [51], difficulties faced by traditional PINNs for convection-dominated convection-diffusion problems in one, two, and three space dimensions are discussed. The authors propose an adaptive approach for choosing weights for the interior loss and surprisingly find that selecting collocation points away from layers yields better results than increasing the number of points within the layer. Other related papers, including [11, 25, 48, 43, 39], typically focus on one-dimensional problems or mildly convection-dominated convection-diffusion problems.

The goal of this contribution consists in expanding upon the knowledge presented in the aforementioned references concerning PINNs for convection-dominated convection-diffusion-reaction problems. We propose and study various loss functionals that are new in the context of PINNs and deploy them to train hard-constrained PINNs for approximating the solution of two benchmark problems defined in [31]. These loss functionals are primarily based on cost functionals from [29, 27, 37], which are specifically designed to tackle convection-dominated convection-diffusion-reaction problems. In two numerical experiments, we compare them to PINNs trained with the classical loss functional and hp-variational PINNs from [34]. The proposed loss functionals lead to errors, in the $L^2(\Omega)$ norm, which are 17.3% and 5.5%, resp., smaller than the methods from the literature when trained to approximate the solution of the two benchmark problems. Furthermore, we observe that the problem with an interior layer can be approximated an order of magnitude better than the problem with boundary layers which is in agreement with the findings of the above-mentioned references. The code and the data used for this contribution can also be found online [17].

The structure of this contribution is as follows: In Section 2, we give a brief introduction into training deep neural networks, the loss functionals of vanilla PINNs and of hp-variational PINNs, and the idea of hard-constrained PINNs. This is followed in Section 3 by a description of the novel loss functionals for convection-dominated convection-diffusion equations proposed in this work. These ideas are numerically investigated in Section 4 using two benchmark problems. Finally, in Section 5 a summary is given and an outlook is provided.

2 Hard-constrained physics-informed neural networks for convection-diffusion problems

In a nutshell, physics-informed neural networks try to approximate the solution to initial-boundary value problems by minimizing the residual of the governing equations, and of boundary and initial conditions [32]. This section provides a short description of the basics of neural networks, two commonly known loss functionals from the literature to train PINNs, and a way how to prescribe the Dirichlet boundary conditions exactly.

2.1 Basic about neural networks

Let us briefly recall the structure of MLPs that are the underlying type of neural networks used in this work; see also [24] for a different presentation. MLPs consist of $n_L \in \mathbb{N}$ so-called *layers*, and each layer is build from $n_i \in \mathbb{N}$, $i = 1, 2, \dots, n_L$, nodes often referred to as *neurons* that each represent a real number. Let $\widehat{\boldsymbol{y}}_i \in \mathbb{R}^{n_i}$, $i = 1, 2, \dots, n_L$, be the vector of nodes of the i-th layer, i.e., each entry in the vector corresponds to the value of a particular node in the layer. Then, for a given $\boldsymbol{x} \in \overline{\Omega}$, the value $u_{\mathcal{N}}(\boldsymbol{x})$ is computed recursively by defining

$$\widehat{\boldsymbol{y}}_1 \coloneqq \boldsymbol{x}, \tag{2a}$$

$$\widehat{\boldsymbol{y}}_i \coloneqq \boldsymbol{\sigma}_i \left(W_i \widehat{\boldsymbol{y}}_{i-1} + \widehat{\boldsymbol{b}}_i \right), \qquad i = 2, 3, \dots, n_L, \tag{2b}$$

$$u_{\mathcal{N}}(\boldsymbol{x}) \coloneqq \widehat{\boldsymbol{y}}_{n_L}, \tag{2c}$$

where $\widehat{\boldsymbol{b}}_i \in \mathbb{R}^{n_i}$ is a vector called *bias*, the matrix $W_i \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_{i-1}}$ is the so-called *weight* matrix, and $\boldsymbol{\sigma}_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is a component-wise defined (non-linear) mapping, usually denoted as *activation function*. Note that by construction the regularity of the activation functions is transferred to the neural network. Usually, the first and the final layer are called *input* and *output layers*, resp., and their number of nodes is determined by the function they shall approximate, i.e., $n_1 \coloneqq \dim(\Omega)$ and $n_L \coloneqq \dim(\mathrm{Im}(u))$ in the case of PINNs. Since in this contribution we aim for PINNs that approximate the exact solution $u : \overline{\Omega} \to \mathbb{R}$ to the boundary value problem (BVP) in (1), it is $\dim(\Omega) = d$ and $\dim(\mathrm{Im}(u)) = 1$. Furthermore, all intermediate layers are referred to as *hidden layers*. In Figure 1 an example MLP that maps from $\mathbb{R}^3$ to $\mathbb{R}$ with two hidden layers is shown. Two examples of activation functions are the hyperbolic tangent $\tanh$ and mish first mentioned in [42] that are, for a given $x \in \mathbb{R}$, defined as

$$\tanh(x) \coloneqq \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \mathrm{mish}(x) \coloneqq x \cdot \tanh\left(\ln\left(1 + e^x\right)\right), \tag{3}$$

and which are depicted in Figure 2.

The collection of all entries of the weight matrices and the bias vectors are referred to as *parameters* $\boldsymbol{p} \in \mathbb{R}^{d_{\boldsymbol{p}}}$, where $d_{\boldsymbol{p}} \coloneqq \sum_{i=2}^{n_L} (n_i + n_i \cdot n_{i-1})$. All remaining user-chosen quantities needed to specify a network, e.g., the number of layers n_L, the number of nodes in each layer n_i, $i = 1, 2, \dots, n_L$, the choice of the activation functions and more regarding the training, introduced below, are called *hyperparameters*.

During the process that is denoted as *training* the parameters $\boldsymbol{p}$ are optimized to minimize a certain loss $\mathcal{L} : \mathbb{R}^{d_{\boldsymbol{p}}} \to \mathbb{R}$ that maps a neural network parametrized by its parameters to a non-negative real

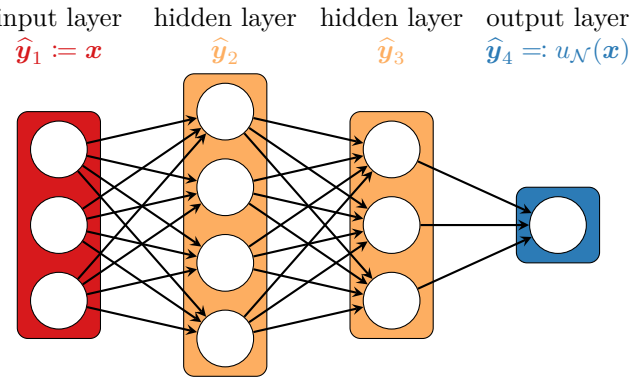


Figure 1: Visualization of a multilayer perceptron model $u_{\mathcal{N}}$ mapping from $\mathbb{R}^3$ to $\mathbb{R}$ with two hidden layers. Each circle represents a node and arrows indicate which previous nodes are used to compute the value of the node the arrow points to. For $\boldsymbol{x} \in \mathbb{R}^3$, each vector of nodes $\widehat{y}_i$, $i = 1, 2, 3, 4$, is computed using equations (2a) to (2c).

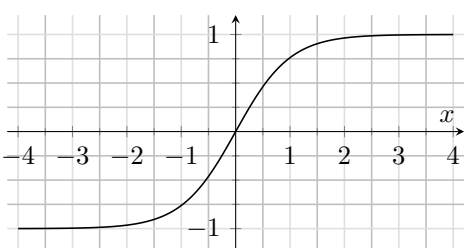
number. In other words, during the training we are searching for the optimal parameters $\boldsymbol{p}^* \in \mathbb{R}^{d_{\boldsymbol{p}}}$ such that

$$\boldsymbol{p}^* \in \operatorname*{arg\,min}_{\boldsymbol{p} \in \mathbb{R}^{d_{\boldsymbol{p}}}} \mathcal{L}(u_{\mathcal{N};\boldsymbol{p}}), \tag{4}$$

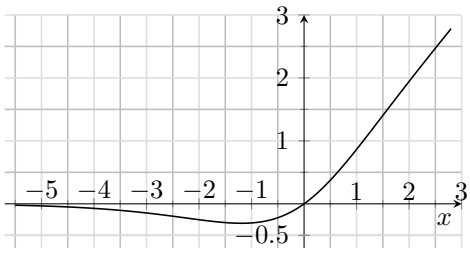
where we used $u_{\mathcal{N};\boldsymbol{p}}$ to encode that the neural network $u_{\mathcal{N}}$ is parametrized by the parameters. For the sake of brevity, in what follows we will write only $u_{\mathcal{N}}$ and $\mathcal{L}(u_{\mathcal{N}})$ if no confusion can occur. From a practical point of view, any optimization routine can be used to solve (4), but often a variant of the stochastic gradient descent method is chosen [24].

2.2 Examples of loss functionals

To ensure that neural networks approximate a certain mapping, a loss functional has to be defined that measures the difference to that mapping. In the context of PINNs, the neural network shall approximate the solution u to a given IBVP, here the problem given in (1). The idea of Dissanayake and Phan-Thien [12], which was recently rediscovered by Raissi, Perdikaris and Karniadakis [45], is to choose the loss functional as the sum of the strong form of the residual of the governing equations and the initial and boundary conditions of the problem; cf. [12, 45]. Let $\boldsymbol{x}_{i;\mathrm{I}} \in \Omega$, $i = 1, 2, \ldots N_{\mathrm{I}} \in \mathbb{N}$, be interior collocation points, and $\boldsymbol{x}_{i;\mathrm{D}} \in \Gamma_{\mathrm{D}} (= \partial\Omega$ here), $i = 1, 2, \ldots N_{\mathrm{D}} \in \mathbb{N}$, collocation points along the



(a) $\tanh(x)$.



(b) $\operatorname{mish}(x)$.

Figure 2: Activation functions $\tanh(x)$ and $\operatorname{mish}(x)$ defined in (3).

Dirichlet boundary. The standard loss functional $\mathcal{L}^{\mathrm{st}}$ as defined in [12, 45] is then given by

$$\mathcal{L}^{\mathrm{st}} \coloneqq \alpha_{\mathrm{I}}^{\mathrm{st}} \mathcal{L}_{\mathrm{I}}^{\mathrm{st}} + \alpha_{\mathrm{D}}^{\mathrm{st}} \mathcal{L}_{\mathrm{D}}^{\mathrm{st}}, \tag{5}$$

where $\alpha_{\mathrm{I}}^{\mathrm{st}}, \alpha_{\mathrm{D}}^{\mathrm{st}} \in \mathbb{R}$ are two non-negative user-chosen constants, and, in the context of convection-diffusion-reaction equations,

$$\mathcal{L}_{\mathrm{I}}^{\mathrm{st}}\left(u_{\mathcal{N}}\right) \coloneqq \frac{|\Omega|}{N_{\mathrm{I}}} \sum_{i=1}^{N_{\mathrm{I}}} \left(\left(-\varepsilon \Delta u_{\mathcal{N}} + \boldsymbol{b} \cdot \nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f \right)\left(\boldsymbol{x}_{i,\mathrm{I}}\right) \right)^2,$$

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

in which we omitted the dependency of $u_{\mathcal{N}}$ on its parameters $\boldsymbol{p}$ as stated above. For computing the derivatives of $u_{\mathcal{N}}$ in $\mathcal{L}_{\mathrm{I}}^{\mathrm{st}}$ it must be assumed that the activation functions are at least twice differentiable in the collocation points. Note that if a Neumann term would be present in (1), a third term of the form $\frac{|\Gamma_{\mathrm{N}}|}{N_{\mathrm{N}}} \sum_{i=1}^{N_{\mathrm{N}}} \left(\varepsilon \nabla u_{\mathcal{N}}\left(\boldsymbol{x}_{i,\mathrm{N}}\right) \cdot \boldsymbol{n}\left(\boldsymbol{x}_{i,\mathrm{N}}\right) - g_{\mathrm{N}}\left(\boldsymbol{x}_{i,\mathrm{N}}\right) \right)^2$ would be added that is based on N_{N} collocation points $\boldsymbol{x}_{i;\mathrm{N}} \in \Gamma_{\mathrm{N}}$, $i = 1, 2, \ldots N_{\mathrm{N}}$, along the Neumann boundary.

It is well known that strong solutions to (1) exist only under rather strong regularity assumptions on the domain and the data of the problem. Therefore, it is questionable whether the strong form of the residual is a good choice in (5). In [33, 35, 34] and [53, 10] loss functionals are introduced that are based on a variational formulation of the residual. They differ in the choice of the test function(s), where the former references use (piecewise) polynomial test functions and the latter use a single neural network-based test function together with a reformulation of the problem to a min-max problem.

Since in the numerical examples below we use the hp-variational loss functional of [34], it will be introduced next; cf. [34] for the original presentation. To this end, let $\mathcal{T}_h$ be a triangulation of $\overline{\Omega}$ into cells $K \in \mathcal{T}_h$. It is assumed that these cells are, depending on the dimension, either lines, quadrilaterals, or hexahedrons, and that they are the image of an affine or d-linear mapping $F_K : \widehat{K} \to K$ from the reference cell $\widehat{K} \coloneqq [-1, 1]^d$ to K. Furthermore, for a given polynomial degree $p \in \mathbb{N}$, let

$$\begin{aligned}
P_p([-1,1]^1) &\coloneqq \{\, \varphi_j(x) \,:\, \varphi_j \in L_p([-1,1]),\ j = 1, 2, \ldots, p-1 \,\}, \\
P_p([-1,1]^2) &\coloneqq \{\, \varphi_j(x)\varphi_k(y) \,:\, \varphi_j, \varphi_k \in L_p([-1,1]),\ j,k = 1, 2, \ldots, p-1 \,\}, \\
P_p([-1,1]^3) &\coloneqq \{\, \varphi_j(x)\varphi_k(y)\varphi_\ell(z) \,:\, \varphi_j, \varphi_k, \varphi_\ell \in L_p([-1,1]),\ j,k,\ell = 1, 2, \ldots, p-1 \,\},
\end{aligned} \tag{6}$$

be the set of test functions on the d-dimensional reference cells, where, with ϕ_k denoting the Legendre polynomial of order k, $L_p([-1,1]) \coloneqq \{\, \phi_{k+1}(x) - \phi_{k-1}(x) \,:\, k = 1, 2, \ldots, p-1 \,\}$. Consequently, the set of test functions $P_p(K)$ on a physical cell $K \in \mathcal{T}_h$ is given by $P_p(K) \coloneqq \{\, v : K \to \mathbb{R} \,:\, v = \widehat{v} \circ F_K^{-1}$ for a $\widehat{v} \in P_p([-1,1]^d) \,\}$, where F_K^{-1} is the inverse of the reference transform. Last but not least, the set of global polynomial test functions $P_p(\mathcal{T}_h)$ is then defined as

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

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

Finally, a loss functional for hp-variational PINNs is given as

$$\mathcal{L}^{\mathrm{hpvP}} \coloneqq \alpha_{\mathrm{I}}^{\mathrm{hpvP}} \mathcal{L}_{\mathrm{I}}^{\mathrm{hpvP}} + \alpha_{\mathrm{D}}^{\mathrm{hpvP}} \mathcal{L}_{\mathrm{D}}^{\mathrm{st}}, \tag{7}$$

where again $\alpha_{\mathrm{I}}^{\mathrm{hpvP}}, \alpha_{\mathrm{D}}^{\mathrm{hpvP}} \in \mathbb{R}$ are two non-negative user-chosen constants, and, in the context of convection-diffusion-reaction equations,

$$\mathcal{L}_{\mathrm{I}}^{\mathrm{hpvP}}\left(u_{\mathcal{N}}\right) \coloneqq \sum_{K \in \mathcal{T}_h} \frac{1}{|P_p(K)|} \sum_{v \in P_p(K)} \left(\int_K \left(-\varepsilon \Delta u_{\mathcal{N}} + \boldsymbol{b} \cdot \nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f \right) v \,\mathrm{d}\boldsymbol{x} \right)^2, \tag{8}$$

where $|P_p(K)| \coloneqq \dim(P_p(K))$. Note that in [34] the authors defined two more loss functionals by using integration by parts one and two times, respectively. However, in this work we restrict, for the sake of brevity, to form (8). The integrals need to be approximated by some numerical quadrature rule. In contrast to [34] in which the authors propose to exploit a Gauss–Lobatto integration rule, we apply a Gauss–Legendre formula.

2.3 Hard-constrained PINNs

As seen in equations (5) and (7), the Dirichlet boundary conditions have to be learned by the neural networks during the training. As a result, the boundary conditions are only satisfied approximately after the training and in the worst case huge differences to the exact boundary conditions can occur. Consequently, since the boundary conditions influence the shape of the solution also in the interior, because the conditions on the inlet boundary are transported into the domain, wrong boundary conditions may lead to a low quality of approximation of the solution by the neural networks. Furthermore, learning the boundary conditions costs training time, which is unnecessary since this information is known a-priori. Therefore, it seems to be reasonable to prescribe the Dirichlet boundary conditions exactly as it is done in many standard finite element methods.

One way to do so, is to utilize so-called *hard-constrained* PINNs as described in [40]. In this publication, the authors use

$$u_{\mathcal{N}} \coloneqq \widetilde{g}_{\mathrm{D}} + h_{\mathrm{ind}}\widetilde{u}_{\mathcal{N}}$$

as ansatz for the neural network, where $\widetilde{g}_{\mathrm{D}} : \overline{\Omega} \to \mathbb{R}$ is a continuous extension of the Dirichlet boundary condition g_{D} to $\overline{\Omega}$, $h_{\mathrm{ind}} : \overline{\Omega} \to \mathbb{R}$ is an indicator function that satisfies, for given $x \in \overline{\Omega}$,

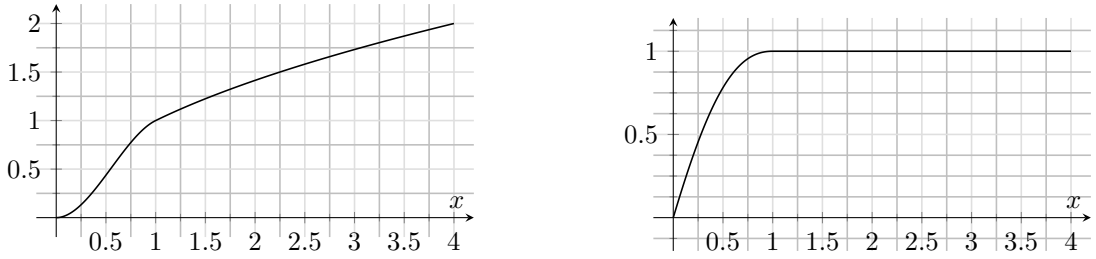
$$h_{\mathrm{ind}}(x) \begin{cases} = 0, & \text{if } x \in \Gamma_{\mathrm{D}}, \\ > 0, & \text{else}, \end{cases}$$

and $\widetilde{u}_{\mathcal{N}}$ is a neural network as described in Section 2.1. By construction, it holds that $u_{\mathcal{N}}|_{\Gamma_{\mathrm{D}}} = g_D$, which means that the Dirichlet boundary conditions are satisfied exactly. Consequently, the terms $\mathcal{L}_{\mathrm{D}}^{\mathrm{st}}$ in equations (5) and (7) are exactly zero, and, hence, can be neglected during the training, which in turn saves training time. Therefore, for a problem with a pure Dirichlet boundary, with this approach the concrete values of the weight terms in front of the individual contributions of the loss functional do not matter and no hyperparameter tuning with respect to these weights needs to be performed.

3 Novel proposals for loss functionals for physics-informed neural networks

In [29, 27, 37, 30], the authors optimize the SUPG parameters of a SUPG finite element method for convection-dominated convection-diffusion equations. In these works, it was observed that the strong form of the residual might not be the best choice as cost functional since inside layers already very small deviations of the numerical solution from the solution of the continuous problem lead to very large values of the strong residual. In this section, we are going to transfer these cost functionals to the PINN setting, which is essentially based on the work of [16].

Note that below we add again the term to approximate the Dirichlet boundary condition. However, if hard-constrained PINNs as described in Section 2.3 are used, then the boundary conditions are exactly satisfied, the term $\mathcal{L}_{\mathrm{D}}^{\mathrm{st}}$ is again exactly 0 and does not need to be considered during the optimization.



(a) $\Phi(x)$. (b) $\xi(x)$.

Figure 3: Functions $\Phi(x)$ and $\xi(x)$ used in the crosswind and the limited residual loss as defined in equations (9) and (12), respectively.

3.1 Crosswind loss functional

The first functional that is novel in the context of PINNs is the so-called *crosswind loss functional*. In [29, 27] it was observed that the strong form of the residual sometimes leads to a smearing of the layers. Consequently, the idea of the authors was to penalize this smearing, and the authors could show in their numerical studies that the crosswind loss functional leads to better results than considering only the strong form of the residual [29, 27].

To state the loss functional, let $\Phi : \mathbb{R} \to \mathbb{R}$ be given, for any $x \in \mathbb{R}$, by

$$\Phi(x) \coloneqq \begin{cases} \sqrt{x}, & \text{if } x \geq 1, \\ 0.5\left(5x^2 - 3x^3\right), & \text{else}, \end{cases} \tag{9}$$

which is depicted for positive values in Figure 3a. Furthermore, for any $\boldsymbol{x} \in \overline{\Omega}$, let us define a set of $(d-1)$ mutually orthonormal vectors which are all perpendicular to $\boldsymbol{b}(\boldsymbol{x})$. In two dimensions, there is only one vector in this set given by

$$\boldsymbol{b}^{\perp}(\boldsymbol{x}) \coloneqq \begin{cases} \frac{(b_2(\boldsymbol{x}), -b_1(\boldsymbol{x}))^T}{\|\boldsymbol{b}(\boldsymbol{x})\|_2}, & \text{if } \boldsymbol{b}(\boldsymbol{x}) \neq \boldsymbol{0}, \\ \boldsymbol{0}, & \text{else}, \end{cases}$$

where $\|\cdot\|_2$ denotes the Euclidean norm of a vector.

The crosswind loss functional for PINNs is then defined as

$$\mathcal{L}^{\mathrm{cw}} \coloneqq \alpha_{\mathrm{I}}^{\mathrm{cw}} \mathcal{L}_{\mathrm{I}}^{\mathrm{cw}} + \alpha_{\mathrm{D}}^{\mathrm{cw}} \mathcal{L}_{\mathrm{D}}^{\mathrm{st}}, \tag{10}$$

where as above $\alpha_{\mathrm{I}}^{\mathrm{cw}}, \alpha_{\mathrm{D}}^{\mathrm{cw}} \in \mathbb{R}$ are two non-negative user-chosen constants, and

$$\mathcal{L}_{\mathrm{I}}^{\mathrm{cw}}\left(u_{\mathcal{N}}\right) \coloneqq \mathcal{L}_{\mathrm{I}}^{\mathrm{st}} + \frac{|\Omega|}{N_{\mathrm{I}}} \sum_{i=1}^{N_{\mathrm{I}}} \left| \Phi\left(\left| \boldsymbol{b}^{\perp}(\boldsymbol{x}_{i,\mathrm{I}}) \cdot \nabla u_{\mathcal{N}}(\boldsymbol{x}_{i,\mathrm{I}}) \right| \right) \right|.$$

3.2 Limited residual loss functional

In order to mitigate the sensitivity of the strong residual on small differences of the analytic and numerical solution, the authors of [37] proposed to restrict large values of the residual. Since their work was done for finite element methods, their cost functional is based on cells collected in a triangulation.

(a) Random points. (b) Equally distanced points.

Figure 4: Voronoi tessellation of the unit square based on five random collocation points (left) and nine equally distanced points (right).

To transfer their idea to the PINN framework, let $\boldsymbol{x}_{i;\mathrm{I}} \in \Omega$, $i = 1, 2, \dots N_{\mathrm{I}} \in \mathbb{N}$, denote the interior collocation points. Then, a triangulation $\mathcal{T}_h$ of Ω can be constructed by computing a Voronoi tessellation of these points, which is a standard procedure in a popular class of finite volume methods; see also [14]. In other words, we define

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

and we set $\mathcal{T}_h \coloneqq \{\, K_i \,:\, i = 1, 2, \dots, N_{\mathrm{I}} \,\}$; see also Figure 4 for two examples. By construction, these cells contain only a single $\boldsymbol{x}_{i;\mathrm{I}}$, and, hence, they can be uniquely identified by the corresponding index i. Based on that triangulation, the ansatz

$$\sum_{i=1}^{N_{\mathrm{I}}} \xi \left(\frac{1}{t_0} \| - \varepsilon \Delta u_{\mathcal{N}} + \boldsymbol{b} \cdot \nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f \|^2_{L^2(K_i)} \right) \tag{11}$$

can be made, where $t_0 \in \mathbb{R}$ is a positive user-chosen constant, and $\xi : \mathbb{R} \to \mathbb{R}$, for any given $x \in \mathbb{R}$, is defined as

$$\xi(x) \coloneqq \begin{cases} \frac{1}{2}x^4 - x^3 - \frac{1}{2}x^2 + 2x, & \text{if } x \leq 1, \\ 1, & \text{else;} \end{cases} \tag{12}$$

cf. also I_h^{lim} in [37]. The function ξ is also depicted in Figure 3b. Approximating the integrals in equation (11) by the midpoint rule, and exploiting $|K_i| \approx |\Omega| / N_{\mathrm{I}}$, which is exactly true if the points are equally distanced, then leads to

$$\begin{aligned} (11) &\approx \sum_{i=1}^{N_{\mathrm{I}}} \xi \left(\frac{|K_i|}{t_0} \left(\left(-\varepsilon \Delta u_{\mathcal{N}} + \boldsymbol{b} \cdot \nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f \right) (\boldsymbol{x}_{i;\mathrm{I}}) \right)^2 \right) \\ &\approx \sum_{i=1}^{N_{\mathrm{I}}} \xi \left(\frac{|\Omega|}{N_{\mathrm{I}}\, t_0} \left(\left(-\varepsilon \Delta u_{\mathcal{N}} + \boldsymbol{b} \cdot \nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f \right) (\boldsymbol{x}_{i;\mathrm{I}}) \right)^2 \right). \end{aligned}$$

Finally, setting

$$\mathcal{L}^{\mathrm{lr}}_{\mathrm{I}} \left(u_{\mathcal{N}} \right) \coloneqq \sum_{i=1}^{N_{\mathrm{I}}} \xi \left(\frac{|\Omega|}{N_{\mathrm{I}}\, t_0} \left(\left(-\varepsilon \Delta u_{\mathcal{N}} + \boldsymbol{b} \cdot \nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f \right) (\boldsymbol{x}_{i;\mathrm{I}}) \right)^2 \right)$$

then gives the *limited residual loss functional* $\mathcal{L}^{\mathrm{lr}}$ defined as

$$\mathcal{L}^{\mathrm{lr}} \coloneqq \alpha^{\mathrm{lr}}_{\mathrm{I}} \mathcal{L}^{\mathrm{lr}}_{\mathrm{I}} + \alpha^{\mathrm{lr}}_{\mathrm{D}} \mathcal{L}^{\mathrm{st}}_{\mathrm{D}}, \tag{13}$$

where once more $\alpha^{\mathrm{lr}}_{\mathrm{I}}, \alpha^{\mathrm{lr}}_{\mathrm{D}} \in \mathbb{R}$ are two non-negative user-chosen constants.

3.3 Limited residual with crosswind loss functional

The ideas of Sections 3.1 and 3.2 can also be combined to form what we call the *limited residual with crosswind term loss functional*. This loss functional is given by

$$\mathcal{L}^{\mathrm{lrcw}} := \alpha_{\mathrm{I}}^{\mathrm{lrcw}} \mathcal{L}_{\mathrm{I}}^{\mathrm{lrcw}} + \alpha_{\mathrm{D}}^{\mathrm{lrcw}} \mathcal{L}_{\mathrm{D}}^{\mathrm{st}}, \tag{14}$$

where as before $\alpha_{\mathrm{I}}^{\mathrm{lrcw}}, \alpha_{\mathrm{D}}^{\mathrm{lrcw}} \in \mathbb{R}$ are two non-negative user-chosen constants, and

$$\begin{aligned}\mathcal{L}_{\mathrm{I}}^{\mathrm{lrcw}}\left(u_{\mathcal{N}}\right) := \sum_{i=1}^{N_{\mathrm{I}}} \xi\left(\frac{|\Omega|}{N_{\mathrm{I}}\, t_0}\left(\left(-\varepsilon\Delta u_{\mathcal{N}} + \boldsymbol{b}\cdot\nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f\right)(\boldsymbol{x}_{i;\mathrm{I}})\right)^2\right) \\ + \frac{|\Omega|}{N_{\mathrm{I}}} \sum_{i=1}^{N_{\mathrm{I}}} \left|\Phi\left(\left|\boldsymbol{b}^{\perp}(\boldsymbol{x}_{i,\mathrm{I}})\cdot\nabla u_{\mathcal{N}}(\boldsymbol{x}_{i,\mathrm{I}})\right|\right)\right|.\end{aligned}$$

4 Numerical studies

In this section, we assess the quality of the PINN approximations based on the loss functionals presented in Sections 2.2 and 3 numerically.

The implementation of the neural networks and the loss functionals is done in TensorFlow [1, 49]. The code for all experiments in this section is also publicly available at `https://doi.org/10.20347/wias.data.7` [17].

4.1 Set-up of experiments

To investigate the quality of the PINN approximations, we use two two-dimensional benchmark problems with a known solution first defined in [31]; see also Problems 1 and 2 below. All experiments are conducted in the convection-dominated regime, since for both problems we have $\varepsilon = 10^{-8}$ and $\|\boldsymbol{b}\|_{[L^\infty(\Omega)]^d} = \mathcal{O}(1)$. In this regime the solution to Problem 1 possesses an interior layer and to Problem 2 two boundary layers, resp.; cf. Figure 6.

The networks are trained to minimize the loss functionals

$$\mathcal{L} + \frac{\lambda_{\mathrm{wd}}}{2}\frac{n_{\mathrm{bs}}}{N_{\mathrm{I}}}\sum_j w_j^2, \tag{15}$$

where $\mathcal{L}$ is chosen to be $\mathcal{L}^{\mathrm{st}}, \mathcal{L}^{\mathrm{hpvP}}, \mathcal{L}^{\mathrm{cw}}, \mathcal{L}^{\mathrm{lr}},$ or $\mathcal{L}^{\mathrm{lrcw}}$ given in equations (5), (7), (10), (13) and (14), resp., $0 \le \lambda_{\mathrm{wd}} \in \mathbb{R}$ is the L^2-weight decay regularization hyperparameter, n_{bs} is the batch size and N_{I} denotes the interior points, and w_j are the components of the weight matrices but not the biases of the networks. The regularization term is often used in neural network optimization to counteract overfitting; see also [6], [22, pp. 116-119, 227-230] in general and [13] in the context of PINNs. With a slight misuse of the notation, we denote the loss functionals from equation (15) still by $\mathcal{L}^{\mathrm{st}}, \mathcal{L}^{\mathrm{hpvP}}, \mathcal{L}^{\mathrm{cw}}, \mathcal{L}^{\mathrm{lr}},$ and $\mathcal{L}^{\mathrm{lrcw}}$ directly, even though the weight-decay term is still present and must not be forgotten. In the loss functionals $\mathcal{L}^{\mathrm{st}}, \mathcal{L}^{\mathrm{hpvP}}, \mathcal{L}^{\mathrm{cw}}, \mathcal{L}^{\mathrm{lr}},$ and $\mathcal{L}^{\mathrm{lrcw}}$, the weight factors $\alpha_{\mathrm{I}}^{\mathrm{m}} := 1$ and $\alpha_{\mathrm{D}}^{\mathrm{m}} := 0$, $\mathrm{m} \in \{\,\mathrm{st}, \mathrm{hpvP}, \mathrm{cw}, \mathrm{lr}, \mathrm{lrcw}\,\}$, are used. The factor for the Dirichlet boundary condition can be set to 0, since hard-constrained PINNs are used; see also below.

Furthermore, to train the networks, the minibatch stochastic gradient descent [22, 24, Section 8.1.3] is used together with the Adam algorithm [36]. Except for the learning rate that is varied as described

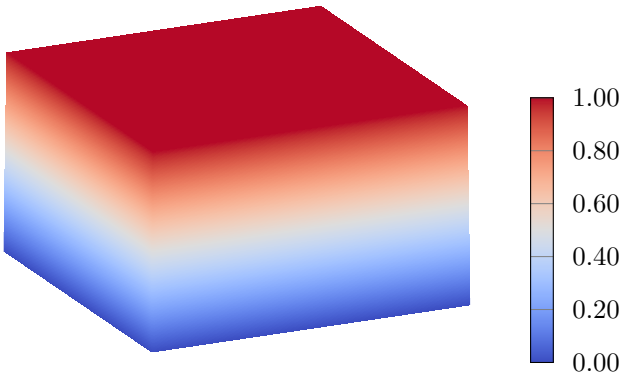


Figure 5: Indicator function h_{ind} for the unit square given in equation (16) with $\kappa = 10^9$.

below, TensorFlow's default values are applied for the optimization. After the training is finished, the network with the smallest loss value during the optimization is returned. Due to the stochastic gradient descent, this is not necessarily the neural network after the final optimization step.

We train the networks with in total $N_{\mathrm{I}} \coloneqq 6,400$ equally distanced interior points and the batch size n_{bs} is set to 32 for all loss functionals except the hp-variational loss. For $\mathcal{L}^{\mathrm{hpvP}}$, the unit square is divided into 64 squares of equal size and in each square polynomials up to degree $p = 6$ are used. This results in $5 \times 5 \times 64 = 1,600$ global basis functions, since we investigate two-dimensional test problems; cf. equation (6). To compute the integrals for the hp-variational loss, a Gauss–Legendre quadrature rule is deployed in each cell with 10×10 points and weights in each coordinate direction which leads to $6,400$ points for approximating the integrals; cf. equation (8). For this loss functional, the batch size and the number of interior points, which is not used at all for hp-vPINNs, is chosen such that $n_{\mathrm{bs}}/N_{\mathrm{I}}$ equals one, since in contrast to the other functionals no loop over batches of interior points needs to be performed.

The loss functionals $\mathcal{L}^{\mathrm{lr}}$ and $\mathcal{L}^{\mathrm{lrcw}}$ depend on the parameter t_0. In accordance with [37], various parameter values are investigated, namely $t_0 \in \{\, 10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4} \,\}$. The corresponding loss functionals are denoted by $\mathcal{L}^{\mathrm{lr}}_{t_0}$ and $\mathcal{L}^{\mathrm{lrcw}}_{t_0}$. However, for the sake of brevity, below only one result for these loss functionals is shown, which is the one with the t_0 that leads to the smallest error after the training.

What is left is to specify the architecture of the networks and how we measure the error. We deploy hard-constrained neural networks as described in Section 2.3. To this end, for both examples we use $\widetilde{g}_{\mathrm{D}} \coloneqq 0$ and

$$h_{\mathrm{ind}}(x, y) \coloneqq \left(1 - e^{-\kappa x}\right)\left(1 - e^{-\kappa y}\right)\left(1 - e^{-\kappa(1-x)}\right)\left(1 - e^{-\kappa(1-y)}\right), \tag{16}$$

where κ is a scaling factor, since both examples are defined on the unit square with homogeneous Dirichlet boundary conditions. The factor $\kappa \coloneqq 10/\varepsilon$ is chosen, because it is well known that the thickness of exponential layers of the exact solution is $\mathcal{O}(\varepsilon)$ [28]. Choosing the aforementioned κ ensures that boundary layers of the resulting PINN approximations are not smeared as a result of the choice of h_{ind}. The function h_{ind} is also visualized in Figure 5.

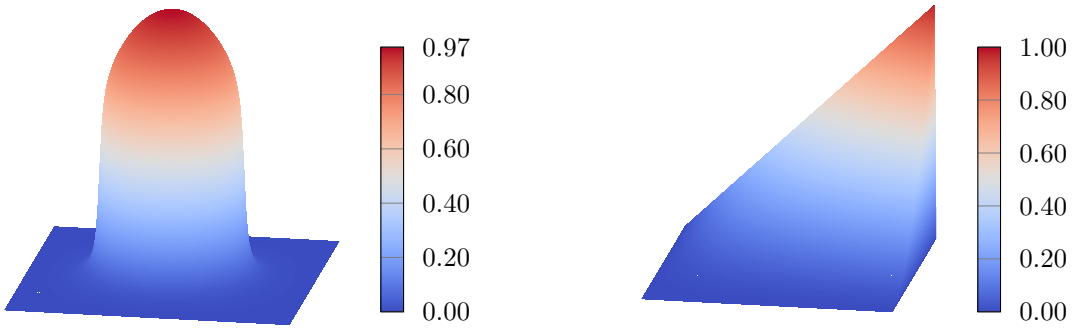
Since the optimal hyperparameters of the neural networks are not known a-priori, we train networks with various combinations of hyperparameters, namely all 630 possible combinations of parameters given in Table 1. Note that the choice of the hyperparameters is guided by the practical advises from [6]. The input layer and the output layer consist of $n_1 \coloneqq 2$ and $n_9 \coloneqq 1$ nodes, resp., since the networks approximate a mapping from a subset of $\mathbb{R}^2$ to $\mathbb{R}$. While in the output layer a linear activation function is used, in all intermediate layers one of the activation functions $\tanh$ or mish defined in (3) are deployed.

Table 1: Set of hyperparameters resulting in 630 different combinations. In the first column, the values $7 \times x$, $x \in \{ 20, 30, 40 \}$, means seven hidden layers with x nodes in each layer.

hidden layers $\times$ nodes	7×20, 7×30, 7×40
activation function	$\tanh$, mish
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}

In the beginning, the values of the weights are initialized based on Glorot initialization [19] with the seeds given in Table 1 and the initial biases are set to zero. These values are then optimized for $10,000$ epochs during the training. Afterwards, we measure the error $e := u - u_{\mathcal{N}}$ between the exact solution u and the PINN approximation $u_{\mathcal{N}}$ as stated below, average over the initialization seeds, and train the best ten networks resulting from each interior loss functional for all three seeds for another $90,000$ epochs. To get the final result, the error is measured again and the average over the seeds is computed.

To measure the error $e := u - u_{\mathcal{N}}$ between the exact solution u and the PINN approximation $u_{\mathcal{N}}$ a suitable norm has to be used, which seems to be a non-trivial question. We could observe that the networks with the smallest error measured in the H^1-semi norm might have a good shape but can be shifted by a non-negligible constant. When the results are evaluated in the L^∞ norm, the norm returned acceptable solutions for Problem 1 but not for Problem 2. The exact solution to Problem 2 has two boundary layers and the width of the boundary layers of the discrete solution $u_{\mathcal{N}}$ is essentially determined by the choice of h_{ind}; see equation (16). Since only a single point determines the quality of the solution in the L^∞ norm, a non-optimal choice of h_{ind} can significantly influence the value of the error. Finally, we decided to use the L^2 norm that, in the convection-dominated regime, is the dominating term in the energy norm $|||e||| := \left(\varepsilon \|\nabla e\|^2_{L^2(\Omega)} + \mu_0 \|e\|^2_{L^2(\Omega)} \right)^{1/2}$ naturally associated with the problem at hand. In the energy norm it is $\mu_0 \in \mathbb{R}$ such that $c - \mathrm{div}(\boldsymbol{b})/2 \geq \mu_0 > 0$, and for the examples below, we have $\varepsilon = 10^{-8}$, and $\mu_0 = 2$ and $\mu_0 = 1$, respectively. We can report that the L^2 norm returns acceptable solutions and hence below we present the results with respect to this norm. To approximate the error by numerical quadrature, the domain is divided into $10,000$ squares of equal size and a Gauss–Legendre quadrature rule with ten points in each coordinate direction is



(a) Exact solution to Problem 1.

(b) Exact solution to Problem 2.

Figure 6: Exact solutions to the problems used in Sections 4.2 and 4.3. The left one is a solution with an interior layer and the right solution possesses two boundary layers at the outflow boundaries.

Table 2: Minimal value of the error $\|u - u_{\mathcal{N}}\|_{L^2}$ after $100,000$ epochs of the best PINNs that approximate the solution to Problem 1. The smallest value is marked with bold font.

	$\mathcal{L}^{\mathrm{st}}$	$\mathcal{L}^{\mathrm{cw}}$	$\mathcal{L}^{\mathrm{lr}}_{0.1}$	$\mathcal{L}^{\mathrm{lrcw}}_{0.1}$	$\mathcal{L}^{\mathrm{hpvP}}$
min	$1.560 \cdot 10^{-3}$	$6.096 \cdot 10^{-3}$	$1.923 \cdot 10^{-3}$	$\mathbf{1.330 \cdot 10^{-3}}$	$7.573 \cdot 10^{-2}$

deployed. This leads in total to $1,000,000$ weights and points.

4.2 Circular interior layer

We begin with Example 2 of [31] which is a problem whose solution possesses an interior layer.

Problem 1 (Circular internal layer)**.** Let $\Omega := (0,1)^2$ be the unit square, and $\varepsilon := 10^{-8}$, $\boldsymbol{b} := (2,3)^T$, $c := 2$ be given. The right-hand side and the boundary conditions of the problem are chosen in correspondence with the analytic solution that is defined to be

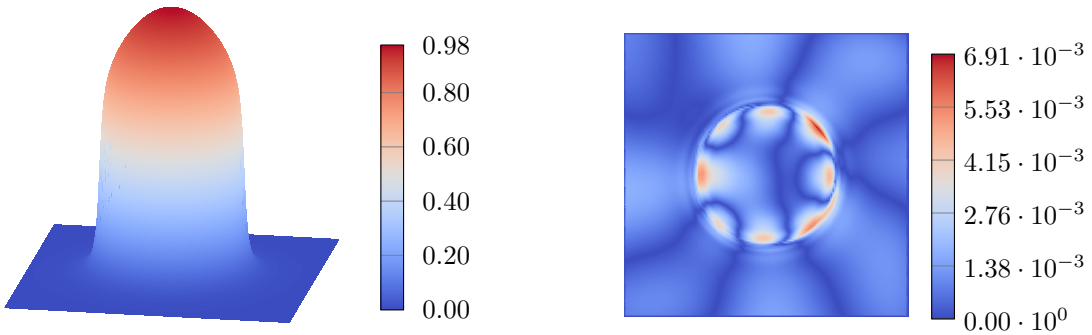
$$u(x,y) := 16x(1-x)y(1-y)\left(\frac{1}{2} + \frac{\arctan\left(200(r_0^2 - (x-x_0)^2 - (y-y_0)^2)\right)}{\pi}\right),$$

where $r_0 := 0.25$ and $x_0 := y_0 := 0.5$. A visualization of the exact solution can be seen in Figure 6a.

The results after training the networks with the different interior loss functionals for $100,000$ epochs are given in Table 2. It can be seen that the limited residual with crosswind loss with $t_0 = 0.1$ produced the smallest error followed by the standard loss and the limited residual loss with $t_0 = 0.1$ which are approximately 17.3% and 44.6%, resp., larger than the overall best error. The smallest errors achieved with the crosswind loss and the hp-variational loss are roughly four and a half times and 57 times, resp., as large as the error obtained with $\mathcal{L}^{\mathrm{lrcw}}_{0.1}$.

The PINN approximation that has overall the smallest error is shown in Figure 7 together with its pointwise error compared to the exact solution. We observe that the solution has the same shape as the exact solution and also it's largest and smallest value coincide up to two decimal digits. The pointwise error lies between 0 at the boundary as expected by exactly prescribing the boundary conditions and $\mathcal{O}(10^{-3})$ at the circle with radius 0.25 where the layer is located.

In total, we conclude that the limited residual with crosswind loss works significantly better than all other methods for the problem with an interior layer and leads to an acceptable solution.



(a) PINN solution $u_{\mathcal{N}}$.

(b) Pointwise error $|u - u_{\mathcal{N}}|$.

Figure 7: PINN approximation $u_{\mathcal{N}}$ of the solution u to Problem 1 (left) and point wise error $|u - u_{\mathcal{N}}|$ (right). The solution is trained with the limited residual with crosswind loss with $t_0 = 0.1$.

Table 3: Minimal value of the error $\|u - u_{\mathcal{N}}\|_{L^2}$ after $100,000$ epochs of the best PINNs that approximate the solution to Problem 2. The smallest value is marked with bold font.

	$\mathcal{L}^{\mathrm{st}}$	$\mathcal{L}^{\mathrm{cw}}$	$\mathcal{L}^{\mathrm{lr}}_{10.0}$	$\mathcal{L}^{\mathrm{lrcw}}_{10.0}$	$\mathcal{L}^{\mathrm{hpvP}}$
min	$6.457 \cdot 10^{-2}$	$4.180 \cdot 10^{-2}$	$\mathbf{3.419 \cdot 10^{-2}}$	$3.459 \cdot 10^{-2}$	$3.619 \cdot 10^{-2}$

4.3 Outflow layers

Next, we study a problem whose solution has outflow boundary layers. It was first defined in Example 3 in [31].

Problem 2 (Outflow layers)**.** Let again $\Omega := (0,1)^2$ be the unit square, and $\varepsilon := 10^{-8}$, $\boldsymbol{b} := (2,3)^T$, $c := 1$ be defined. The right-hand side and the boundary conditions of the problem are derived from the exact solution which is defined to be

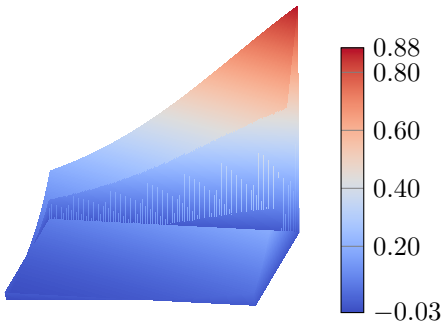
$$u(x,y) := xy^2 - y^2 \exp\left(\frac{2(x-1)}{\varepsilon}\right) - x \exp\left(\frac{3(y-1)}{\varepsilon}\right) + \exp\left(\frac{2(x-1)+3(y-1)}{\varepsilon}\right).$$

This solution is shown in Figure 6b.

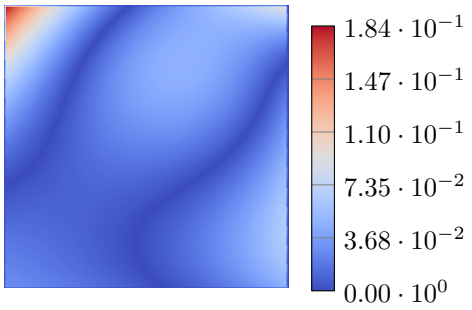
The smallest occurring error and the average error of the networks trained for $100,000$ epochs with the various interior loss functionals are presented in Table 3. It can be observed that the limited residual loss with $t_0 = 10.0$ leads to the smallest error followed by the limited residual with crosswind loss with the same t_0 and the hp-variational loss which are slightly worse than the limited residual loss. Moreover, the two novel loss functionals lead to errors that are roughly 47.0% and 46.4% better than the best result obtained with the standard loss which works the worst for this problem. The crosswind loss is somewhat in between the standard and the hp-variational loss.

A visualization of the PINN solution with the overall smallest error and its pointwise error compared to the exact solution is presented in Figure 8. The solution follows roughly the shape that we expect, but is not as close to the exact solution as for Problem 1. Both the largest and the smallest value of the discrete solution are 0.09 and 0.03 off the corresponding values of the exact solution. Moreover, the solution has even negative values which the exact solution does not have. Hence, the PINN solution does not satisfy a discrete maximum principle. The pointwise error is again 0 at the boundary and roughly 0.184 at the upper left corner of the domain. The reason for this might be that the boundary conditions at the inflow boundary are not propagated correctly to the interior due to the small value scaling factor κ in h_{ind}. This phenomenon was also identified as a typical reason why PINNs might converge to trivial solutions as reported in [9]. Compared to the previous experiment, the absolute values of $|u - u_{\mathcal{N}}|$ and the smallest $L^2(\Omega)$ error are two and one order of magnitude larger, resp., indicating that solutions with outflow boundary layers are more difficult to approximate than solutions with interior layers. The obtained results are in agreement with the expectation that the outflow boundary layer problem is more difficult to solve than Problem 1, since the layers are considerably steeper, as well as with results from the literature; cf. [35, 2].

For the problem with outflow layers we conclude that the limited residual loss is a significantly better choice than the standard loss functional. Taking into account the huge dominance of convection and the smallness of the layers, the obtained numerical solution provides at least an acceptable qualitative approximation of the exact solution. But from the quantitative point of view, the numerical solution is still somewhat unsatisfactory and needs to be improved in future works.



(a) PINN solution $u_{\mathcal{N}}$.



(b) Pointwise error $|u - u_{\mathcal{N}}|$.

Figure 8: PINN approximation $u_{\mathcal{N}}$ of the solution u to Problem 2 (left) and point wise error $|u - u_{\mathcal{N}}|$ (right).

5 Conclusions and outlook

In this work we proposed several novel loss functionals for physics-informed neural networks for convection-dominated convection-diffusion equations, which are based on corresponding objective functionals from the literature. We tested them numerically on two benchmark problems, where the dominance of convection is considerably larger than in the available literature for PINNs applied to convection-diffusion problems, and compared them to PINN approximations obtained with the vanilla loss functional and a hp-variational loss functional.

We observed that for both problems two of the three novel loss functionals significantly reduced the $L^2(\Omega)$ error compared to the standard and the hp-variational loss from the literature and, hence, they are promising alternatives in the case of convection-dominated convection-diffusion problems. The approaches led to reasonable solutions for both type of problems. Nevertheless, the tested PINNs could approximate the solution with an interior layer much better than the solution with boundary layers both when looking at the solution and comparing the $L^2(\Omega)$ errors. In the latter case, the $L^2(\Omega)$ error of the PINN solution was one order of magnitude larger than for the former problem.

This work can be seen as a first step towards a systematic investigation of PINNs for convection-dominated convection-diffusion equations. Since the benchmark problems used in our numerical studies are both driven by source terms, the next step might be to study problems where the solution is driven by boundary conditions. In that case, it might be more challenging for PINNs to approximate solutions with boundary layers since the interior loss functional and the boundary term might have counteracting roles.

Furthermore, as seen in the numerical experiments PINNs do not necessarily preserve maximums principle by default, which might be problematic in real-word scenarios. A systematic investigation with respect to discrete maximum principles and how to guarantee them will be studied in the future.

Moreover, how to choose a suitable set of collocation points is still an open problem and this holds true especially for problems with layers. Is it more useful to choose points in the vicinity of layers or apart from them? It might be intuitive to choose a large amount of points close to the layer regions, but the experiments of [51] indicate that this is not necessarily the case. This behavior will be studied in future works.

Last but not least, choosing an appropriate norm to measure the error for selecting good networks is an open question. In the current work, the $L^2(\Omega)$ norm was used because it turned out to be the best choice among the norms reported in Section 4.1. However, modern finite element error analysis does

neither use the energy norm nor the $L^2(\Omega)$ norm for proving error estimates, since these norms are too weak for obtaining so-called robust estimates, which are estimates where the constants in the error bound do not blow up for very small diffusion coefficients. Norms for which robust estimates can be proved contain terms from so-called stabilized finite element methods. The transfer of this concept for choosing appropriate norms to PINNs is open.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from `https://www.tensorflow.org/`.

[2] Amirhossein Arzani, Kevin W. Cassel, and Roshan M. D'Souza. Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation. *Journal of Computational Physics*, 473:111768, 2023.

[3] 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. *Computer Methods in Applied Mechanics and Engineering*, 200(47-48):3395–3409, November 2011.

[4] Gabriel R. Barrenechea, Volker John, and Petr Knobloch. Finite element methods respecting the discrete maximum principle for convection-diffusion equations. *SIAM Rev.*, 2024. to appear.

[5] Andrea Beck, David Flad, and Claus-Dieter Munz. Deep neural networks for data-driven LES closure models. *Journal of Computational Physics*, 398:108910, December 2019.

[6] Yoshua Bengio. Practical Recommendations for Gradient-Based Training of Deep Architectures. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 437–478. Springer, Berlin, Heidelberg, 2012.

[7] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, December 2021.

[8] 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. *Journal of Scientific Computing*, 92(3):88, July 2022.

[9] Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Mitigating propagation failures in physics-informed neural networks using retain-resample-release (r3) sampling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *ICML'23*, pages 7264–7302, Honolulu, Hawaii, USA, July 2023. JMLR.org.

[10] Tim De Ryck, Siddhartha Mishra, and Roberto Molinaro. wPINNs: Weak Physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws, July 2022. preprint.

[11] Taco de Wolff, Hugo Carrillo, Luis Martí, and Nayat Sanchez-Pi. Towards optimally weighted physics-informed neural networks in ocean modelling, 2021. preprint.

[12] Gamini Dissanayake and Nhan Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

[13] Nathan Doumèche, Gérard Biau, and Claire Boyer. Convergence and error analysis of PINNs, May 2023.

[14] Patricio Farrell, Nella Rotundo, Duy Hai Doan, Markus Kantner, Jürgen Fuhrmann, and Thomas Koprucki. Drift-Diffusion Models. In Joachim Piprek, editor, *Handbook of Optoelectronic Device Modeling and Simulation: Lasers, Modulators, Photodetectors, Solar Cells, and Numerical Methods, Vol. 2*, pages 733–771. CRC Press, Boca Raton, 1 edition, July 2017.

[15] Derk Frerichs and Volker John. On reducing spurious oscillations in discontinuous Galerkin (DG) methods for steady-state convection–diffusion equations. *Journal of Computational and Applied Mathematics*, 393:113487, September 2021.

[16] Derk Frerichs-Mihov. *On Slope Limiting and Deep Learning Techniques for the Numerical Solution to Convection-Dominated Convection-Diffusion Problems*. PhD thesis, Free University Berlin, Berlin, July 2023. Accepted / In publication.

[17] Derk Frerichs-Mihov, Linus Henning, and Volker John. Data and code from the paper "On loss functionals for physics-informed neural networks for convection-dominated convection-diffusion problems", 2023. https://doi.org/10.20347/wias.data.7.

[18] 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. *Journal of Scientific Computing*, 97(2):36, September 2023.

[19] 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*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 2010. PMLR.

[20] Antônio Tadeu Azevedo Gomes, Larissa Miguez da Silva, and Frédéric Valentin. Improving Boundary Layer Predictions Using Parametric Physics-Aware Neural Networks. In Philippe Navaux, Carlos J. Barrios H., Carla Osthoff, and Ginés Guerrero, editors, *High Performance Computing*, Communications in Computer and Information Science, pages 90–102, Porto Alegre, Brazil, 2022. Springer International Publishing.

[21] Antônio Tadeu Azevedo Gomes, Larissa Miguez da Silva, and Frédéric Valentin. Physics-Aware Neural Networks for Boundary Layer Linear Problems, 2022. preprint.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, Massachusetts, USA, 2016.

[23] QiZhi He and Alexandre M. Tartakovsky. Physics-Informed Neural Network Method for Forward and Backward Advection-Dispersion Equations. *Water Resources Research*, 57(7):e2020WR029479, 2021.

[24] Catherine F. Higham and Desmond J. Higham. Deep Learning: An Introduction for Applied Mathematicians. *SIAM Review*, 61(4):860–891, January 2019.

[25] 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. *Physics of Fluids*, 34(7):077108, 2022.

[26] Volker John and Petr Knobloch. On spurious oscillations at layers diminishing (SOLD) methods for convection–diffusion equations: Part I – A review. *Computer Methods in Applied Mechanics and Engineering*, 196(17-20):2197–2215, March 2007.

[27] Volker John and Petr Knobloch. Adaptive Computation of Parameters in Stabilized Methods for Convection-Diffusion Problems. In Andrea Cangiani, Ruslan L. Davidchack, Emmanuil Georgoulis, Alexander N. Gorban, Jeremy Levesley, and Michael V. Tretyakov, editors, *Numerical Mathematics and Advanced Applications 2011 – Proceedings of ENUMATH 2011*, volume 1, pages 275–283, Berlin, Heidelberg, 2013. Springer.

[28] Volker John, Petr Knobloch, and Julia Novo. Finite elements for scalar convection-dominated equations and incompressible flow problems: A never ending story? *Computing and Visualization in Science*, 19(5-6):47–63, December 2018.

[29] Volker John, Petr Knobloch, and Simona B. Savescu. A posteriori optimization of parameters in stabilized methods for convection–diffusion problems – Part I. *Computer Methods in Applied Mechanics and Engineering*, 200(41-44):2916–2929, 2011.

[30] Volker John, Petr Knobloch, and Ulrich Wilbrandt. A posteriori optimization of parameters in stabilized methods for convection-diffusion problems—Part II. *J. Comput. Appl. Math.*, 428:Paper No. 115167, 17, 2023.

[31] Volker John, Joseph M. Maubach, and Lutz Tobiska. Nonconforming Streamline-Diffusion-Finite-Element-Methods for Convection-Diffusion Problems. *Numerische Mathematik*, 78(2):165–188, December 1997.

[32] George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, June 2021.

[33] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. VPINNs: Variational Physics-Informed Neural Networks For Solving Partial Differential Equations, November 2019. preprint.

[34] Ehsan Kharazmi, Zhongqiang Zhang, and George E.M. Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, February 2021.

[35] 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*, pages 298–307, Virtual, Online, 2020. PMLR.

[36] Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method For Stochastic Optimization. In *ICLR 2015*, page 13, San Diego, California, USA, 2014. arXiv.

[37] Petr Knobloch, Petr Lukáš, and Pavel Solin. On error indicators for optimizing parameters in stabilized methods. *Advances in Computational Mathematics*, 45(4):1853–1862, 2019.

[38] 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*, volume 34, pages 26548–26560, Virtual, Online, 2021. Curran Associates, Inc.

[39] 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. *Nuclear Science and Engineering*, 197(9):1–31, 2023.

[40] Lu Lu, Raphaël Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. Physics-Informed Neural Networks with Hard Constraints for Inverse Design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, January 2021.

[41] Nils Margenberg, Christian Lessig, and Thomas Richter. Structure preservation for the Deep Neural Network Multigrid Solver. *ETNA - Electronic Transactions on Numerical Analysis*, 56:86–101, 2021.

[42] Diganta Misra. Mish: A Self Regularized Non-Monotonic Activation Function, August 2020. preprint.

[43] Rambod Mojgani, Maciej Balajewicz, and Pedram Hassanzadeh. Lagrangian PINNs: A causality-conforming solution to failure modes of physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 404:115810, 2023.

[44] Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[45] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.

[46] Deep Ray and Jan S. Hesthaven. Detecting troubled-cells on two-dimensional unstructured grids using a neural network. *Journal of Computational Physics*, 397:108845, November 2019.

[47] Hans-Görg Roos, Martin Stynes, and Lutz Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion-Reaction and Flow Problems*, volume 24 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Heidelberg, 2 edition, 2008.

[48] Mohammad Hossein Saadat, Blazhe Gjorgiev, Laya Das, and Giovanni Sansavini. Neural tangent kernel analysis or PINN for advection-diffusion equation, 2022. preprint.

[49] TensorFlow Developers. TensorFlow (v2.13.0), July 2023. `https://doi.org/10.5281/zenodo.8117732`.

[50] Henry von Wahl and Thomas Richter. Using a deep neural network to predict the motion of underresolved triangular rigid bodies in an incompressible flow. *International Journal for Numerical Methods in Fluids*, 93(12):3364–3383, 2021.

[51] Yufeng Wang, Cong Xu, Min Yang, and Jin Zhang. Less Emphasis on Difficult Layer Regions: Curriculum Learning for Singularly Perturbed Convection-Diffusion-Reaction Problems, 2023. preprint.

[52] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[53] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, June 2020.

[54] Yifei Zong, QiZhi He, and Alexandre M. Tartakovsky. Physics-Informed Neural Network Method for Parabolic Differential Equations with Sharply Perturbed Initial Conditions, 2022. preprint.