



On Loss Functionals for Physics-Informed Neural Networks for Steady-State Convection-Dominated Convection-Diffusion Problems

Derk Frerichs-Mihov¹ · Linus Henning² · Volker John^{1,2} 

Received: 8 December 2023 / Revised: 23 April 2024 / Accepted: 22 May 2024 /

Published online: 29 August 2024

© The Author(s) 2024

Abstract

Solutions of convection-dominated convection-diffusion problems usually possess 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 ((I)BVPs) received a lot of interest. This paper studies various loss functionals for PINNs that are especially designed for convection-dominated convection-diffusion problems and that are novel in the context of PINNs. They are numerically compared to the vanilla and an hp -variational loss functional from the literature based on two steady-state 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.

Keywords Steady-state convection-diffusion problems · Convection-dominated regime · Physics-informed neural networks (PINNs) · Loss functionals

Mathematics Subject Classification 65N99 · 68T07

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

✉ Volker John
john@wias-berlin.de

Derk Frerichs-Mihov
frerichs-mihov@wias-berlin.de

Linus Henning
linus.henning@fu-berlin.de

¹ Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Mohrenstr. 39, 10117 Berlin, Germany

² Department of Mathematics and Computer Science, Free University Berlin, Arnimallee 6, 14195 Berlin, Germany

is considered in this work, they are formulated as follows: find a sufficiently smooth solution $u: \overline{\Omega} \rightarrow \mathbb{R}$ such that

$$-\varepsilon \Delta u + \mathbf{b} \cdot \nabla u + cu = f \quad \text{in } \Omega, \quad u = g \quad \text{along } \partial\Omega, \quad (1)$$

where $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, is a bounded domain with the polyhedral Lipschitz boundary $\partial\Omega$, $0 < \varepsilon \in \mathbb{R}$ is a positive diffusion coefficient, $\mathbf{b} \in [W^{1,\infty}(\Omega)]^d$ 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 applications, convection is often stronger than diffusion by several orders of magnitude. In this so-called convection-dominated regime, which is mathematically described by a large Péclet number, i.e., $Pe := L\|\mathbf{b}\|_{[L^\infty(\Omega)]^d}/\varepsilon \gg 1$, where $0 < L \in \mathbb{R}$ is a characteristic length scale of the problem, 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 so-called subgrid scales. Because of the multiscale character, it is challenging for many numerical methods to accurately approximate the solution in the vicinity of layers; e.g., see [3, 14, 26, 28, 49]. 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; e.g., see, [5, 17, 41, 48, 52]. 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 [11] 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 [47]. Since then, numerous contributions and enhancements have been made, leading to their application in a diverse range of problems. For a comprehensive overview of physics-informed neural networks (PINNs) and their variations, we refer, e.g., to [7, 8, 32] 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 on the renowned universal approximation theorem, which states that a feed-forward MLP, 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 [21, p. 194]; see also [46] 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 [46, Sect. 4]. Generalizations of the theorem include a broader class of activation functions, incorporating the ReLU function, among others [21, p. 195]. However, it is noteworthy that the ability of MLPs to approximate functions does not necessarily imply that typical algorithms will converge toward such a network.

In a nutshell, the main idea for approximating the solution to an (initial-)boundary value problem using MLPs consists in 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 IBVPs. 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 IBVP 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 feature sets them apart from many classical methods that often struggle to meet all these requirements [32]. Even the first steps of an abstract error analysis for PINNs have been performed, e.g., in [42, 43].

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; e.g., see [8, 32] and the related references. Consequently, unlike classical methods, there are only a 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. For complicated problems, like problems of multiscale character, it is currently not even clear whether it is at all possible to compute accurate approximations with PINNs. 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. In the current paper, different combinations of hyperparameters are studied that lead altogether to 630 network architectures and the best results are reported. For insights on hyperparameter optimization approaches, we refer to [55] 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. Solutions of convection-diffusion-reaction problems that exhibit layers are not smooth in this sense. It is known that PINNs 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 [22] explores PINNs in the context of time-dependent advection-dispersion problems in one and two dimensions, particularly for moderate Péclet numbers of order $Pe = \mathcal{O}(1)$, $Pe = \mathcal{O}(10)$, and $Pe = \mathcal{O}(100)$. The results demonstrate that the choice of weights significantly influences the solution quality, with the PINN approximation being comparable in accuracy to the streamline-upwind Petrov-Galerkin (SUPG) method [22]. These findings are extended in [57], 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, [19, 20] 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 parameterized 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 [53], 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 layers. Other related papers, including [25, 39, 45, 50, 54], typically focus on one-dimensional problems or mildly convection-dominated convection-diffusion problems.

The goal of this paper consists in further exploring to which extent PINNs are able to predict accurate solutions for convection-dominated convection-diffusion problems. As already mentioned above, theoretical error bounds are not available, so performing systematic numerical studies is currently the only viable approach for addressing this question. The novelty of this paper consists of proposing and studying various loss functionals that are new in the context of PINNs. In addition, the PINNs are applied to problems with much larger Péclet numbers, and with this to much steeper layers, than in most papers from the literature mentioned above, namely to two benchmark problems with known solutions defined in [31]. The investigated loss functionals are primarily based on cost functionals from [27, 29, 37], which are specifically designed to tackle convection-dominated convection-diffusion-reaction problems. In two numerical studies, 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 for 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 better by one order of magnitude 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 be found online [16].

The structure of this contribution is as follows. In Sect. 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 Sect. 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 Sect. 4 using two benchmark problems. Finally, in Sect. 5 a summary is given and an outlook is provided.

2 Hard-Constrained Physics-Informed Neural Networks (PINNs) for Convection-Diffusion Problems

In a nutshell, PINNs try to approximate the solution to IBVPs 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.

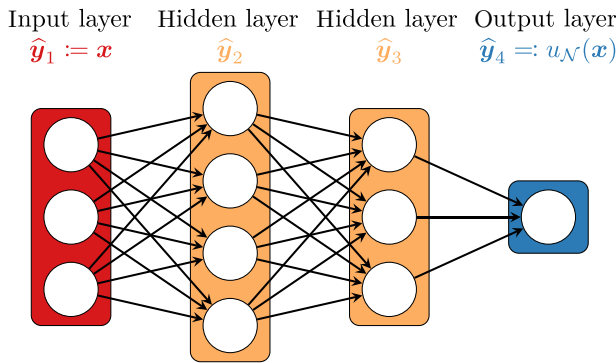


Fig. 1 Visualization of a multilayer perceptron model (MLP) $u_{\mathcal{N}}$ mapping from \mathbb{R}^3 to \mathbb{R} with two hidden layers. Each circle represents a node and arrows indicate which previous nodes are used to compute the value of the node the arrow points to. For $x \in \mathbb{R}^3$,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

2.1 Basics 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 built from $n_i \in \mathbb{N}, i = 1, 2, \dots, n_L$, nodes, often referred to as *neurons*, where each represents a real number. Let $\hat{y}_i \in \mathbb{R}^{n_i}, i = 1, 2, \dots, n_L$, 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 $x \in \overline{\Omega}$, the value $u_{\mathcal{N}}(x)$ is computed recursively by defining

$$\hat{y}_1 := x, \tag{2a}$$

$$\hat{y}_i := \sigma_i (W_i \hat{y}_{i-1} + \hat{b}_i), \quad i = 2, 3, \dots, n_L, \tag{2b}$$

$$u_{\mathcal{N}}(x) := \hat{y}_{n_L}, \tag{2c}$$

where $\hat{b}_i \in \mathbb{R}^{n_i}$ 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 $\sigma_i: \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_i}$ is a component-wise defined (non-linear) mapping, usually called the *activation function*. Note that by construction the regularity of the activation functions is transferred to the neural network. Usually, the first and the final layers are called *input* and *output layers*, respectively, and their number of nodes is determined by the function they shall approximate, i.e., $n_1 := \dim(\Omega)$ and $n_L := \dim(\text{Im}(u))$ in our application of PINNs. Since in this contribution we aim for PINNs that approximate the exact solution $u: \overline{\Omega} \rightarrow \mathbb{R}$ to the boundary value problem (BVP) in (1), it is $\dim(\Omega) = d$ and $\dim(\text{Im}(u)) = 1$. Furthermore, all intermediate layers are referred to as *hidden layers*. In Fig. 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 [44], that are, for a given $x \in \mathbb{R}$, defined as

$$\tanh(x) := \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \text{mish}(x) := x \cdot \tanh(\ln(1 + e^x)), \tag{3}$$

and which are depicted in Fig. 2.

The collection of all entries of the weight matrices and the bias vectors is referred to as *parameters* $p \in \mathbb{R}^{d_p}$, where $d_p := \sum_{i=2}^{n_L} (n_i + n_i \cdot n_{i-1})$. All remaining user-chosen

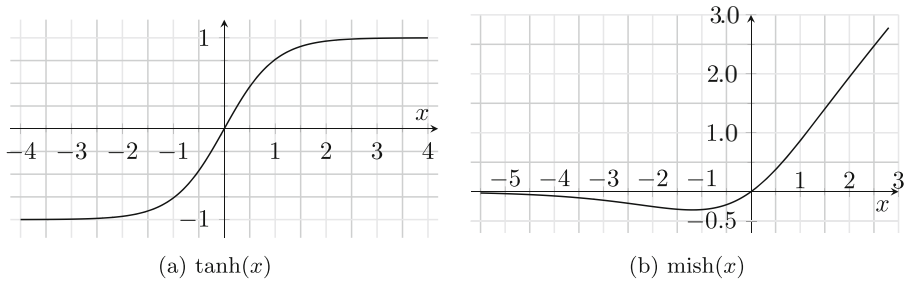


Fig. 2 Activation functions $\tanh(x)$ and $\text{mish}(x)$ defined in (3)

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 called *training*, the parameters \mathbf{p} are optimized to minimize a certain loss $\mathcal{L}: \mathbb{R}^{d_p} \rightarrow \mathbb{R}$ that maps a neural network parameterized by its parameters to a non-negative real number. In other words, during the training we are searching for the optimal parameters $\mathbf{p}^* \in \mathbb{R}^{d_p}$ such that

$$\mathbf{p}^* \in \arg \min_{\mathbf{p} \in \mathbb{R}^{d_p}} \mathcal{L}(u_{\mathcal{N}; \mathbf{p}}), \tag{4}$$

where we use $u_{\mathcal{N}; \mathbf{p}}$ to encode that the neural network $u_{\mathcal{N}}$ is parameterized 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 [11], which was recently rediscovered by Raissi et. al [47], 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. [11, 47]. Let $\mathbf{x}_{i;I} \in \Omega, i = 1, 2, \dots, N_I \in \mathbb{N}$, be interior collocation points, and $\mathbf{x}_{i;D} \in \Gamma_D (= \partial\Omega \text{ here}), i = 1, 2, \dots, N_D \in \mathbb{N}$, collocation points along the Dirichlet boundary. The standard loss functional \mathcal{L}^{st} as defined in [11, 47] is then given by

$$\mathcal{L}^{\text{st}} := \alpha_I^{\text{st}} \mathcal{L}_I^{\text{st}} + \alpha_D^{\text{st}} \mathcal{L}_D^{\text{st}}, \tag{5}$$

where $\alpha_I^{\text{st}}, \alpha_D^{\text{st}} \in \mathbb{R}$ are two non-negative user-chosen constants, and, in the context of convection-diffusion-reaction equations,

$$\begin{aligned} \mathcal{L}_I^{\text{st}}(u_{\mathcal{N}}) &:= \frac{|\Omega|}{N_I} \sum_{i=1}^{N_I} ((-\varepsilon \Delta u_{\mathcal{N}} + \mathbf{b} \cdot \nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f)(\mathbf{x}_{i;I}))^2, \\ \mathcal{L}_D^{\text{st}}(u_{\mathcal{N}}) &:= \frac{|\Gamma_D|}{N_D} \sum_{i=1}^{N_D} (u_{\mathcal{N}}(\mathbf{x}_{i;D}) - g_D(\mathbf{x}_{i;D}))^2, \end{aligned}$$

in which we omitted the dependency of $u_{\mathcal{N}}$ on its parameters \mathbf{p} as stated above, $|\Omega|$ is the d -dimensional measure of Ω and $|\Gamma_D|$ the $(d - 1)$ -dimensional measure of Γ_D . For computing the derivatives of $u_{\mathcal{N}}$ in $\mathcal{L}_1^{\text{st}}$ 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 $\frac{|\Gamma_N|}{N_N} \sum_{i=1}^{N_N} (\varepsilon \nabla u_{\mathcal{N}}(\mathbf{x}_{i,N}) \cdot \mathbf{n}(\mathbf{x}_{i,N}) - g_N(\mathbf{x}_{i,N}))^2$ would be added that is based on N_N collocation points $\mathbf{x}_{i,N} \in \Gamma_N, i = 1, 2, \dots, N_N$, 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 [10, 33–35, 56] 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 $\bar{\Omega}$ 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 $F_K: \hat{K} \rightarrow K$ from the reference cell $\hat{K} := [-1, 1]^d$ to K . Furthermore, for a given polynomial degree $p \in \mathbb{N}$, let

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

be the sets of test functions on the d -dimensional reference cells, where $L_p([-1, 1]) := \{\phi_{k+1}(x) - \phi_{k-1}(x) : k = 1, 2, \dots, p - 1\}$, and ϕ_k is the Legendre polynomial of order k . Consequently, the set of test functions $P_p(K)$ on a physical cell $K \in \mathcal{T}_h$ is given by $P_p(K) := \{v:K \rightarrow \mathbb{R} : v = \hat{v} \circ F_K^{-1} \text{ for a } \hat{v} \in P_p([-1, 1]^d)\}$, 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

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

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

$$\mathcal{L}^{\text{hpVP}} := \alpha_{\text{I}}^{\text{hpVP}} \mathcal{L}_{\text{I}}^{\text{hpVP}} + \alpha_{\text{D}}^{\text{hpVP}} \mathcal{L}_{\text{D}}^{\text{st}}, \tag{7}$$

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

$$\mathcal{L}_{\text{I}}^{\text{hpVP}}(u_{\mathcal{N}}) := \sum_{K \in \mathcal{T}_h} \frac{1}{|P_p(K)|} \sum_{v \in P_p(K)} \left(\int_K (-\varepsilon \Delta u_{\mathcal{N}} + \mathbf{b} \cdot \nabla u_{\mathcal{N}} + cu_{\mathcal{N}} - f) v \, d\mathbf{x} \right)^2, \tag{8}$$

where $|P_p(K)| := \dim(P_p(K))$. 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 Eqs. (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 in 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 approximation of the solution by the neural networks. Furthermore, learning the boundary conditions costs training time, which is unnecessary since this information is known as a-priori. Dirichlet boundary conditions are data of the problem, and in our opinion, all available data should be used in the training process. For all the mentioned reasons, it seems to be reasonable to prescribe the Dirichlet boundary conditions exactly as it is done in many standard finite element methods. In addition, this approach avoids the need to select user-chosen constants in (5), since $\alpha_D^{\text{st}} = 0$. The other constant just scales the functional, which does not change the problem. We use $\alpha_1^{\text{st}} = 1$.

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

$$u_{\mathcal{N}} := \tilde{g}_D + h_{\text{ind}} \tilde{u}_{\mathcal{N}}$$

as ansatz for the neural network, where $\tilde{g}_D: \overline{\Omega} \rightarrow \mathbb{R}$ is a continuous extension of the Dirichlet boundary condition g_D to $\overline{\Omega}$, $h_{\text{ind}}: \overline{\Omega} \rightarrow \mathbb{R}$ is an indicator function that satisfies, for given $x \in \overline{\Omega}$,

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

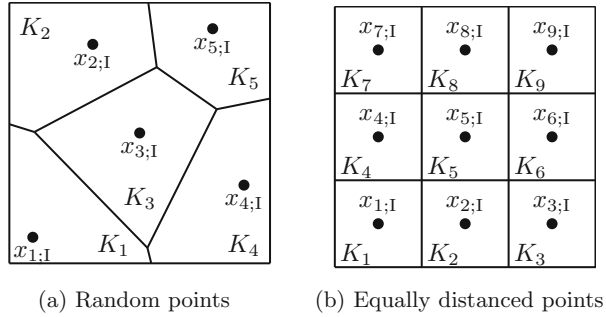
and $\tilde{u}_{\mathcal{N}}$ is a neural network as described in Sect. 2.1. By construction, it holds that $u_{\mathcal{N}}|_{\Gamma_D} = g_D$, which means that the Dirichlet boundary conditions are satisfied exactly. Consequently, the terms $\mathcal{L}_D^{\text{st}}$ in Eqs. (5) and (7) are exactly zero, and, hence, can be neglected during the training, as explained above.

3 Novel Proposals for Loss Functionals for Physics-Informed Neural Networks (PINNs)

In [27, 29, 30, 37], the authors optimize the SUPG parameters of an 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 the 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 [15].

Note that in the presentation below we add again the term to approximate the Dirichlet boundary condition. However, if hard-constrained PINNs as described in Sect. 2.3 are used,

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



Since the numerical studies use hard-constrained PINNs as explained in Sect. 2.3, it is $\alpha_D^{cw} = 0$ and we chose $\alpha_1^{cw} = 1$.

3.2 Limited Residual Loss Functional

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.

To transfer their idea to the PINN framework, let $x_{i;I} \in \Omega, i = 1, 2, \dots, N_I \in \mathbb{N}$, 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 [13]. In other words, we define

$$K_i := \{y \in \bar{\Omega} : |y - x_{i;I}| \leq |y - x_{j;I}| \text{ for all } j = 1, 2, \dots, N_I\}$$

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

$$\sum_{i=1}^{N_I} \xi \left(\frac{1}{t_0} \| -\varepsilon \Delta u_N + \mathbf{b} \cdot \nabla u_N + cu_N - f \|_{L^2(K_i)}^2 \right) \tag{11}$$

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

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

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

$$\begin{aligned} (11) &\approx \sum_{i=1}^{N_I} \xi \left(\frac{|K_i|}{t_0} ((-\varepsilon \Delta u_N + \mathbf{b} \cdot \nabla u_N + cu_N - f)(x_{i;I}))^2 \right) \\ &\approx \sum_{i=1}^{N_I} \xi \left(\frac{|\Omega|}{N_I t_0} ((-\varepsilon \Delta u_N + \mathbf{b} \cdot \nabla u_N + cu_N - f)(x_{i;I}))^2 \right). \end{aligned}$$

Finally, setting

$$\mathcal{L}_I^{\text{lr}}(u_{\mathcal{N}}) := \sum_{i=1}^{N_I} \xi \left(\frac{|\Omega|}{N_I t_0} \left((-\varepsilon \Delta u_{\mathcal{N}} + \mathbf{b} \cdot \nabla u_{\mathcal{N}} + cu_{\mathcal{N}} - f)(\mathbf{x}_{i;I}) \right)^2 \right)$$

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

$$\mathcal{L}^{\text{lr}} := \alpha_I^{\text{lr}} \mathcal{L}_I^{\text{lr}} + \alpha_D^{\text{lr}} \mathcal{L}_D^{\text{st}}, \tag{13}$$

where once more $\alpha_I^{\text{lr}}, \alpha_D^{\text{lr}} \in \mathbb{R}$ are two non-negative user-chosen constants. For hard-constrained PINNs, as used in the numerical studies, it is $\alpha_D^{\text{lr}} = 0$ and we used $\alpha_I^{\text{lr}} = 1$.

3.3 Limited Residual with Crosswind Loss Functional

The ideas of Sects. 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

$$\mathcal{L}^{\text{lrw}} := \alpha_I^{\text{lrw}} \mathcal{L}_I^{\text{lrw}} + \alpha_D^{\text{lrw}} \mathcal{L}_D^{\text{st}}, \tag{14}$$

where as before $\alpha_I^{\text{lrw}}, \alpha_D^{\text{lrw}} \in \mathbb{R}$ are two non-negative user-chosen constants, and

$$\begin{aligned} \mathcal{L}_I^{\text{lrw}}(u_{\mathcal{N}}) := & \sum_{i=1}^{N_I} \xi \left(\frac{|\Omega|}{N_I t_0} \left((-\varepsilon \Delta u_{\mathcal{N}} + \mathbf{b} \cdot \nabla u_{\mathcal{N}} + cu_{\mathcal{N}} - f)(\mathbf{x}_{i;I}) \right)^2 \right) \\ & + \frac{|\Omega|}{N_I} \sum_{i=1}^{N_I} \left| \Phi \left(|\mathbf{b}^\perp(\mathbf{x}_{i;I}) \cdot \nabla u_{\mathcal{N}}(\mathbf{x}_{i;I})| \right) \right|. \end{aligned}$$

4 Numerical Studies

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

The implementation of the neural networks and the loss functionals is done in TensorFlow [1, 51]. The code for all experiments in this section is also publicly available at <https://doi.org/10.20347/wias.data.7> [16].

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 Problems 1 and 2 below. All experiments are conducted in the convection-dominated regime, since for both problems we have $\varepsilon = 10^{-8}$ and $\|\mathbf{b}\|_{[L^\infty(\Omega)]^d} = \mathcal{O}(1)$. In this regime the solution to Problem 1 possesses an interior layer and to Problem 2 two boundary layers, respectively; cf. Fig. 6.

The networks are trained to minimize the loss functionals

$$\mathcal{L} + \frac{\lambda_{\text{wd}}}{2} \frac{n_{\text{bs}}}{N_I} \sum_j w_j^2, \tag{15}$$

where \mathcal{L} is chosen to be $\mathcal{L}^{\text{st}}, \mathcal{L}^{\text{hvp}}, \mathcal{L}^{\text{cw}}, \mathcal{L}^{\text{lr}}$, or \mathcal{L}^{lrw} given in Eqs. (5), (7), (10), (13), and (14), respectively, $0 \leq \lambda_{\text{wd}} \in \mathbb{R}$ is the L^2 -weight decay regularization hyperparameter,

n_{bs} is the batch size, 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], [21, pp. 116–119, 227–230] in general and [12] in the context of PINNs. With a slight misuse of the notation, we denote the loss functionals from Eq. (15) still by \mathcal{L}^{st} , \mathcal{L}^{hpvP} , \mathcal{L}^{cw} , \mathcal{L}^{lr} , and \mathcal{L}^{lrcw} directly, even though the weight-decay term is still present and must not be forgotten. In the loss functionals \mathcal{L}^{st} , \mathcal{L}^{hpvP} , \mathcal{L}^{cw} , \mathcal{L}^{lr} , and \mathcal{L}^{lrcw} , the weight factors $\alpha_1^m := 1$ and $\alpha_D^m := 0$, $m \in \{st, hpvP, cw, lr, lrcw\}$, are used. The factor for the Dirichlet boundary condition can be set to 0, since hard-constrained PINNs are used.

Furthermore, to train the networks, the minibatch stochastic gradient descent [21, 24, Sect. 8.1.3] is used together with the Adam algorithm [36]. Except for the learning rate that is varied as described 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 $N_I := 6\,400$ 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}^{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. Eq. (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 6400 points for approximating the integrals; cf. Eq. (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_{bs}/N_I equals one, since in contrast to the other functionals no loop over batches of interior points needs to be performed.

The functionals \mathcal{L}^{lr} and \mathcal{L}^{lrcw} depend on the parameter t_0 . In accordance with [37], various parameter values are investigated, namely $t_0 \in \{10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}\}$. The corresponding loss functionals are denoted by $\mathcal{L}_{t_0}^{lr}$ and $\mathcal{L}_{t_0}^{lrcw}$. The dependency of the results on t_0 will be discussed.

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 Sect. 2.3. To this end, for both examples we use $\tilde{g}_D := 0$ and

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

where κ is a scaling factor, since both examples are defined on the unit square with homogeneous Dirichlet boundary conditions. The factor $\kappa := 10/\varepsilon$ 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 visualized in Fig. 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 advice from [6]. The input layer and the output layer consist of $n_1 := 2$ and $n_9 := 1$ nodes, respectively, since the networks approximate a mapping from a subset of \mathbb{R}^2 to \mathbb{R} . Altogether, the networks are comparatively small, a forward sweep requires only a few thousands of multiplications plus the computation of the activation functions. Thus, once a network is trained, an approximation of the solution of the BVP (1) is computed very fast, in our simulations of the order of 10^{-2} seconds. While in the output layer a linear activation

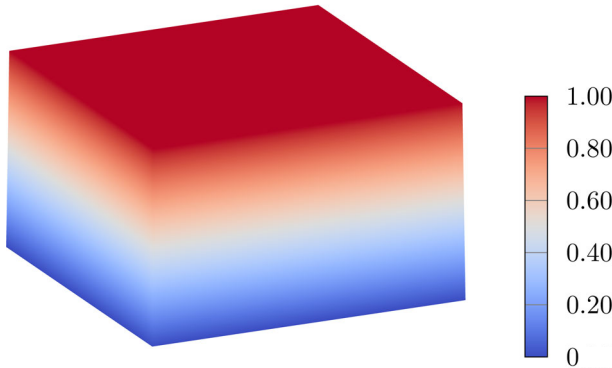


Fig. 5 Indicator function h_{ind} for the unit square given in Eq. (16) with $\kappa = 10^9$

Table 1 Set of hyperparameters resulting in 630 different combinations

Hidden layers \times nodes	$7 \times 20, 7 \times 30, 7 \times 40$
Activation function	tanh, mish
Learning rate	$0.01 \times 3^{-0}, 0.01 \times 3^{-1}, 0.01 \times 3^{-2}, 0.01 \times 3^{-3}, 0.01 \times 3^{-4},$ $0.01 \times 3^{-5}, 0.01 \times 3^{-6}$
Initialization seed	42, 43, 44
Weight decay λ_{wd}	$10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$

In the first column, the values $7 \times x, x \in \{20, 30, 40\}$, mean seven hidden layers with x nodes in each layer

function is used, in all intermediate layers one of the activation functions tanh or mish defined in (3) is deployed. In the beginning, the values of the weights are initialized based on Glorot initialization [18] 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 $L^\infty(\Omega)$ 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 Eq. (16). Since only a single point determines the quality of the solution in the $L^\infty(\Omega)$ norm, a non-optimal choice of h_{ind} can significantly influence the value of the error. Finally, we decided to use the $L^2(\Omega)$ norm that, in the convection-dominated regime, is the dominating term in the energy norm $\|e\| := \left(\varepsilon \|\nabla e\|_{L^2(\Omega)}^2 + \mu_0 \|e\|_{L^2(\Omega)}^2 \right)^{1/2}$ naturally associated with the problem at hand. In the energy norm it is $\mu_0 \in \mathbb{R}$ such that $c - \text{div}(\mathbf{b})/2 \geq \mu_0 > 0$, and for the examples below, we have $\varepsilon = 10^{-8}$, and $\mu_0 = 2$ and $\mu_0 = 1$, respectively. We can report that using the $L^2(\Omega)$ norm returns acceptable solutions and hence below we present

Table 2 Averaged computing time in seconds for one epoch

\mathcal{L}^{st}	\mathcal{L}^{cw}	$\mathcal{L}_{t_0}^{lr}$	$\mathcal{L}_{t_0}^{lrcw}$	\mathcal{L}^{hpvP}
0.66	0.62	0.68	0.61	0.14

The averages are taken over both examples

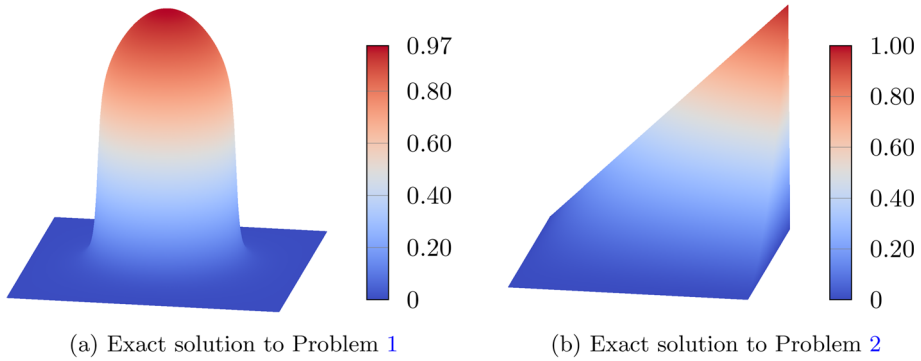


Fig. 6 Exact solutions to the problems used in Sects. 4.2 and 4.3. The left one is a solution with an interior layer and the right solution possesses two boundary layers at the outflow boundaries

the results with respect to this norm. To approximate the error by numerical quadrature, the domain is divided into 10000 squares of equal size and a Gauss-Legendre quadrature rule with ten points in each coordinate direction is deployed. This leads in total to 1 000 000 weights and points.

The training was performed on HP compute servers with Xeon Eighteen-Core 3 100 MHz processors. Information on averaged computing times is provided in Table 2. We observed very similar times for both examples and Table 2 shows that also the times for the PINNs with the various loss functionals are very similar. Only the training of the *hp*-vPINNs was faster by a factor of four to five. Altogether, the overall training times are quite long, but we like to recall that the aim of this paper consists in exploring if the studied methods can be principally used for predicting accurate solutions for steady-state convection-dominated convection-diffusion-reaction problems. After having obtained a positive evaluation of this aim, the next natural goal consists in increasing the efficiency of the methods.

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 $\varepsilon := 10^{-8}$, $\mathbf{b} := (2, 3)^T$, $c := 2$ 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

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

Table 3 Minimal value of the error $\|u - u_{\mathcal{N}}\|_{L^2(\Omega)}$ after 100 000 epochs of the best PINNs that approximate the solution to Problem 1

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

The smallest value is marked with bold font

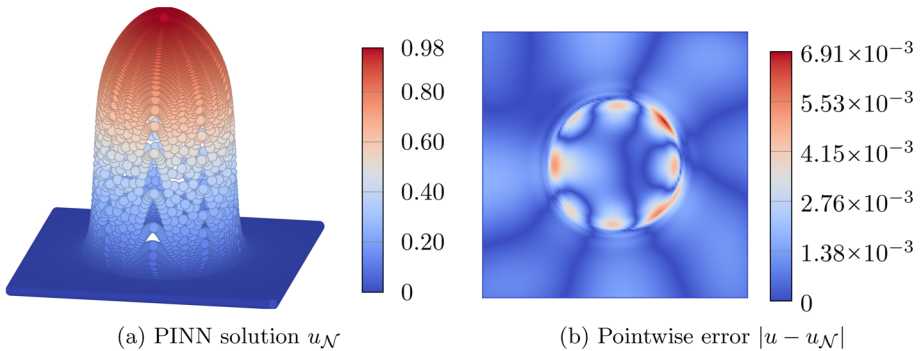


Fig. 7 PINN approximation $u_{\mathcal{N}}$ of the solution u to Problem 1 (left) and pointwise error $|u - u_{\mathcal{N}}|$ (right). The solution is trained with the limited residual with crosswind loss with $t_0 = 0.1$

where $r_0 := 0.25$ and $x_0 := y_0 := 0.5$. Choosing as the characteristic length scale of the problem $L = 1$, which is the length of the edges of the domain, then its Péclet number is given by $Pe = 3.6 \times 10^8$. A visualization of the exact solution can be seen in Fig. 6a.

The results after training the networks with the different interior loss functionals for 100 000 epochs are given in Table 3. It can be seen that the limited residual with the 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%, respectively, 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, respectively, as large as the error obtained with $\mathcal{L}_{0.1}^{lrcw}$.

The PINN approximation that has overall the smallest error is shown in Fig. 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 its largest and smallest values 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.

Table 4 presents results for the loss functionals $\mathcal{L}_{t_0}^{lr}$ and $\mathcal{L}_{t_0}^{lrcw}$ for different values of the parameter t_0 . It can be observed that solutions with similar errors are obtained when using $\mathcal{L}_{t_0}^{lr}$ with $t_0 \in \{0.1, 1, 10\}$ and if $\mathcal{L}_{t_0}^{lrcw}$ is used for $t_0 \in \{0.01, 0.1\}$.

In total, we conclude that the limited residual with the crosswind loss and with an appropriate parameter t_0 works significantly better than all other methods for the problem with an interior layer and leads to an acceptable solution.

Table 4 Minimal value of the error $\|u - u_{\mathcal{N}}\|_{L^2(\Omega)}$ after 100 000 epochs of the best PINNs trained with $\mathcal{L}_{t_0}^{\text{lr}}$ and $\mathcal{L}_{t_0}^{\text{lrw}}$ that approximate the solution to Problem 1

t_0	0.001	0.01	0.1	1	10
$\mathcal{L}_{t_0}^{\text{lr}}$	3.669×10^{-3}	3.337×10^{-3}	1.923×10^{-3}	1.974×10^{-3}	2.151×10^{-3}
$\mathcal{L}_{t_0}^{\text{lrw}}$	3.265×10^{-3}	1.431×10^{-3}	1.330×10^{-3}	2.773×10^{-3}	1.395×10^{-2}

Table 5 Minimal value of the error $\|u - u_{\mathcal{N}}\|_{L^2(\Omega)}$ after 100 000 epochs of the best PINNs that approximate the solution to Problem 2

	\mathcal{L}^{st}	\mathcal{L}^{cw}	$\mathcal{L}_{10.0}^{\text{lr}}$	$\mathcal{L}_{10.0}^{\text{lrw}}$	$\mathcal{L}^{\text{hpvP}}$
min	6.457×10^{-2}	4.180×10^{-2}	3.419×10^{-2}	3.459×10^{-2}	3.619×10^{-2}

The smallest value is marked with bold font

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 $\varepsilon := 10^{-8}$, $\mathbf{b} := (2, 3)^T$, $c := 1$ 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 Fig. 6b. The Péclet number is $Pe = 3.6 \times 10^8$, which is based on the characteristic length scale $L = 1$.

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 5. 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 the 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 Fig. 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 values of the discrete solution are 0.09 and 0.03 of 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

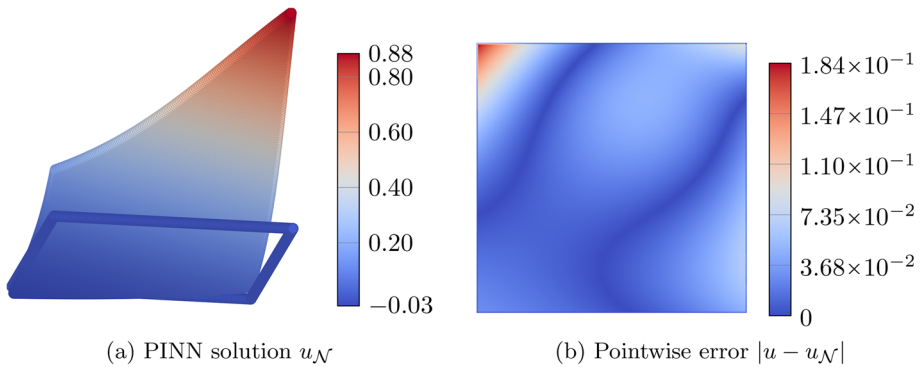


Fig. 8 PINN approximation $u_{\mathcal{N}}$ of the solution u to Problem 2 (left) and pointwise error $|u - u_{\mathcal{N}}|$ (right)

Table 6 Minimal value of the error $\|u - u_{\mathcal{N}}\|_{L^2(\Omega)}$ after 100 000 epochs of the best PINNs trained with $\mathcal{L}_{t_0}^{\text{lr}}$ and $\mathcal{L}_{t_0}^{\text{lrw}}$ that approximate the solution to Problem 2

t_0	0.001	0.01	0.1	1	10
$\mathcal{L}_{t_0}^{\text{lr}}$	1.120×10^{-1}	1.020×10^{-1}	1.624×10^{-1}	6.422×10^{-2}	3.419×10^{-2}
$\mathcal{L}_{t_0}^{\text{lrw}}$	2.227×10^{-1}	2.390×10^{-1}	2.032×10^{-1}	4.774×10^{-2}	3.459×10^{-2}

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, respectively, 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. [2, 35].

The best results for various values of the parameter t_0 in $\mathcal{L}_{t_0}^{\text{lr}}$ and $\mathcal{L}_{t_0}^{\text{lrw}}$ are presented in Table 6. Using comparatively large values is clearly the best approach. This observation is in contrast to Problem 1, compared to Table 4. We think that this behavior is connected to the fact that the layers of the solution of Problem 2 are much steeper than those of the solution of Problem 1. Since also the errors for $t_0 = 1$ are (somewhat) smaller than the error obtained with the standard loss \mathcal{L}^{st} , there is a wide interval for a good choice of t_0 .

For the problem with outflow layers we conclude that the limited residual loss with an appropriate parameter t_0 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.

5 Conclusions and Outlook

In this work we proposed several novel loss functionals for PINNs 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 an hp -variational loss functional.

We observed that for both problems two of the three novel loss functionals, if they are used with appropriate parameter t_0 , 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 types 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. The available results suggest that larger values of t_0 are necessary for solutions with steeper layers.

This work can be seen as a first step towards a systematic investigation of PINNs for steady-state convection-dominated convection-diffusion equations for very large Péclet numbers. Since the benchmark problems used in our numerical studies are both driven by source terms, the next step will be to study problems where the solution is driven by boundary conditions at the inlet. These problems are of a somewhat different character than the examples with prescribed solutions studied in the current paper. In corresponding benchmark problems, like the Hemker problem [23], the right-hand side f is zero in Ω and an analytic solution is not known. Consequently, one needs a different approach than the $L^2(\Omega)$ error for evaluating numerical solutions. In addition, the treatment of the boundary condition at the inlet is of utmost importance since this data determines the solution in the whole domain, if $f = 0$ in Ω . Finally, in contrast to the results presented here, we expect unphysical values (spurious oscillations) in PINN approximations of the solution. Thus, corresponding numerical studies should present and examine approaches for reducing or removing such oscillations. This is outside the scope of the present paper and it will be the subject of future research.

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 number of points close to the layer regions, but the experiments of [53] indicate that this is not necessarily the case. This behavior will be studied in future works.

Furthermore, 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 Sect. 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.

Last but not least, the efficiency of the training process should be increased. We think that the most promising way consists in utilizing GPU servers, instead of the currently used CPU servers, which requires implementation efforts.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability The code for all experiments is publicly available at <https://doi.org/10.20347/wias.data.7>.

Declarations

Conflict of Interest The authors have no relevant financial or non-financial interests to disclose, and they have no conflict of interest to declare that is relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: large-scale machine learning on heterogeneous systems. Software available from <https://www.tensorflow.org/> (2015)
2. Arzani, A., Cassel, K.W., D'Souza, R.M.: Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation. *J. Comput. Phys.* **473**, 111768 (2023). <https://doi.org/10.1016/j.jcp.2022.111768>
3. Augustin, M., Caiazzo, A., Fiebach, A., Fuhrmann, J., John, V., Linke, A., Umla, R.: An assessment of discretizations for convection-dominated convection-diffusion equations. *Comput. Methods Appl. Mech. Eng.* **200**(47/48), 3395–3409 (2011). <https://doi.org/10.1016/j.cma.2011.08.012>
4. Barrenechea, G.R., John, V., Knobloch, P.: Finite element methods respecting the discrete maximum principle for convection-diffusion equations. *SIAM Rev.* **66**, 3–88 (2024). <https://doi.org/10.1137/22M1488934>
5. Beck, A., Flad, D., Munz, C.-D.: Deep neural networks for data-driven LES closure models. *J. Comput. Phys.* **398**, 108910 (2019). <https://doi.org/10.1016/j.jcp.2019.108910>
6. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade: Second Edition. Lecture Notes in Computer Science*, vol. 7700, pp. 437–478. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-35289-8_26
7. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E.: Physics-informed neural networks (PINNs) for fluid mechanics: a review. *Acta. Mech. Sin.* **37**(12), 1727–1738 (2021). <https://doi.org/10.1007/s10409-021-01148-1>
8. Cuomo, S., Di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F.: Scientific machine learning through physics-informed neural networks: where we are and what's next. *J. Sci. Comput.* **92**(3), 88 (2022). <https://doi.org/10.1007/s10915-022-01939-z>
9. Daw, A., Bu, J., Wang, S., Perdikaris, P., Karpatne, A.: Mitigating propagation failures in physics-informed neural networks using retain-resample-release (r3) sampling. In: *Proceedings of the 40th International Conference on Machine Learning. ICML'23*, vol. 202, pp. 7264–7302. JMLR.org, Honolulu, Hawaii, USA (2023)

10. De Ryck, T., Mishra, S., Molinaro, R.: wPINNs: weak physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws. *SIAM J. Numer. Anal.* **62**(2), 811–841 (2024). <https://doi.org/10.1137/22M1522504>
11. Dissanayake, G., Phan-Thien, N.: Neural-network-based approximations for solving partial differential equations. *Commun. Numer. Methods Eng.* **10**(3), 195–201 (1994). <https://doi.org/10.1002/cnm.1640100303>
12. Doumèche, N., Biau, G., Boyer, C.: Convergence and error analysis of PINNs (2023). <https://doi.org/10.48550/arXiv.2305.01240>
13. Farrell, P., Rotundo, N., Doan, D.H., Kantner, M., Fuhrmann, J., Koprucki, T.: Drift-diffusion models. In: Piprek, J. (ed.) *Handbook of Optoelectronic Device Modeling and Simulation: Lasers, Modulators, Photodetectors, Solar Cells, and Numerical Methods*, vol. 2, 1st edn., pp. 733–771. CRC Press, Boca Raton (2017). <https://doi.org/10.4324/9781315152318>
14. Frerichs, D., John, V.: On reducing spurious oscillations in discontinuous Galerkin (DG) methods for steady-state convection-diffusion equations. *J. Comput. Appl. Math.* **393**, 113487 (2021). <https://doi.org/10.1016/j.cam.2021.113487>
15. Frerichs-Mihov, D.: On slope limiting and deep learning techniques for the numerical solution to convection-dominated convection-diffusion problems. Ph.D. Thesis. Free University Berlin, Berlin (2023)
16. Frerichs-Mihov, D., Henning, L., John, V.: 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>. https://archive.wias-berlin.de/receive/wias_mods_00007477
17. Frerichs-Mihov, D., Henning, L., John, V.: Using deep neural networks for detecting spurious oscillations in discontinuous Galerkin solutions of convection-dominated convection-diffusion equations. *J. Sci. Comput.* **97**(2), 36 (2023). <https://doi.org/10.1007/s10915-023-02335-x>
18. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research*, vol. 9, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy (2010)
19. Gomes, A.T.A., Silva, L.M., Valentin, F.: Physics-aware neural networks for boundary layer linear problems (2022). <https://doi.org/10.48550/arXiv.2208.12559>
20. Gomes, A.T.A., Silva, L.M., Valentin, F.: Improving boundary layer predictions using parametric physics-aware neural networks. In: Navaux, P., Barrios, H.C.J., Osthoff, C., Guerrero, G. (eds.) *High Performance Computing. Communications in Computer and Information Science*, pp. 90–102. Springer, Porto Alegre (2022). https://doi.org/10.1007/978-3-031-23821-5_7
21. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. MIT Press, Cambridge (2016). <https://www.deeplearningbook.org/>. Accessed 2023-07-19
22. He, Q., Tartakovsky, A.M.: Physics-informed neural network method for forward and backward advection-dispersion equations. *Water Resour. Res.* **57**(7), 2020–029479 (2021). <https://doi.org/10.1029/2020WR029479>
23. Hemker, P.W.: A singularly perturbed model problem for numerical computation. *J. Comput. Appl. Math.* **76**(1/2), 277–285 (1996). [https://doi.org/10.1016/S0377-0427\(96\)00113-6](https://doi.org/10.1016/S0377-0427(96)00113-6)
24. Higham, C.F., Higham, D.J.: Deep learning: an introduction for applied mathematicians. *SIAM Rev.* **61**(4), 860–891 (2019). <https://doi.org/10.1137/18M1165748>
25. Hou, Q., Sun, Z., He, L., Karemat, A.: Orthogonal grid physics-informed neural networks: a neural network-based simulation tool for advection-diffusion-reaction problems. *Phys. Fluids* **34**(7), 077108 (2022). <https://doi.org/10.1063/5.0095536>
26. John, V., Knobloch, P.: On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations: part I—a review. *Comput. Methods Appl. Mech. Eng.* **196**(17/18/19/20), 2197–2215 (2007). <https://doi.org/10.1016/j.cma.2006.11.013>
27. John, V., Knobloch, P.: Adaptive computation of parameters in stabilized methods for convection-diffusion problems. In: Cangiani, A., Davidchack, R.L., Georgoulis, E., Gorbun, A.N., Levesley, J., Tretyakov, M.V. (eds.) *Numerical Mathematics and Advanced Applications 2011—Proceedings of ENUMATH 2011*, vol. 1, pp. 275–283. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-33134-3_30
28. John, V., Knobloch, P., Novo, J.: Finite elements for scalar convection-dominated equations and incompressible flow problems: a never ending story? *Comput. Vis. Sci.* **19**(5/6), 47–63 (2018). <https://doi.org/10.1007/s00791-018-0290-5>
29. John, V., Knobloch, P., Savescu, S.B.: A posteriori optimization of parameters in stabilized methods for convection-diffusion problems—part I. *Comput. Methods Appl. Mech. Eng.* **200**(41/42/43/44), 2916–2929 (2011). <https://doi.org/10.1016/j.cma.2011.04.016>

30. John, V., Knobloch, P., Wilbrandt, U.: A posteriori optimization of parameters in stabilized methods for convection-diffusion problems—part II. *J. Comput. Appl. Math.* **428**, 115167–17 (2023). <https://doi.org/10.1016/j.cam.2023.115167>
31. John, V., Maubach, J.M., Tobiska, L.: Nonconforming streamline-diffusion-finite-element-methods for convection-diffusion problems. *Numer. Math.* **78**(2), 165–188 (1997). <https://doi.org/10.1007/s002110050309>
32. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. *Nat. Rev. Phys.* **3**(6), 422–440 (2021). <https://doi.org/10.1038/s42254-021-00314-5>
33. Kharazmi, E., Zhang, Z., Karniadakis, G.E.: VPINNs: variational physics-informed neural networks for solving partial differential equations (2019). <https://doi.org/10.48550/arXiv.1912.00873>
34. Kharazmi, E., Zhang, Z., Karniadakis, G.E.M.: *hp*-VPINNs: variational physics-informed neural networks with domain decomposition. *Comput. Methods Appl. Mech. Eng.* **374**, 113547 (2021). <https://doi.org/10.1016/j.cma.2020.113547>
35. Khodayi-Mehr, R., Zavlanos, M.: VarNet: variational neural networks for the solution of partial differential equations. In: Proceedings of the 2nd Conference on Learning for Dynamics and Control, pp. 298–307. PMLR, Virtual, Online (2020)
36. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. In: ICLR 2015, p. 13. San Diego, California, USA (2014). <https://doi.org/10.48550/ARXIV.1412.6980>
37. Knobloch, P., Lukáš, P., Solin, P.: On error indicators for optimizing parameters in stabilized methods. *Adv. Comput. Math.* **45**(4), 1853–1862 (2019). <https://doi.org/10.1007/s10444-019-09662-4>
38. Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., Mahoney, M.W.: Characterizing possible failure modes in physics-informed neural networks. In: Advances in Neural Information Processing Systems, vol. 34, pp. 26548–26560. Curran Associates, Inc., Virtual, Online (2021)
39. Laghi, L., Schiassi, E., De Florio, M., Furfaro, R., Mostacci, D.: Physics-informed neural networks for 1-D steady-state diffusion-advection-reaction equations. *Nucl. Sci. Eng.* **197**(9), 1–31 (2023). <https://doi.org/10.1080/00295639.2022.2160604>
40. Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., Johnson, S.G.: Physics-informed neural networks with hard constraints for inverse design. *SIAM J. Sci. Comput.* **43**(6), 1105–1132 (2021). <https://doi.org/10.1137/21M1397908>
41. Margenberg, N., Lessig, C., Richter, T.: Structure preservation for the deep neural network multigrid solver. *ETNA Electron. Trans. Numer. Anal.* **56**, 86–101 (2022). https://doi.org/10.1553/etna_vol56s86
42. Mishra, S., Molinaro, R.: Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. *IMA J. Numer. Anal.* **42**(2), 981–1022 (2022). <https://doi.org/10.1093/imanum/drab032>
43. Mishra, S., Molinaro, R.: Estimates on the generalization error of physics-informed neural networks for approximating PDEs. *IMA J. Numer. Anal.* **43**(1), 1–43 (2023). <https://doi.org/10.1093/imanum/drab093>
44. Misra, D.: Mish: a self regularized non-monotonic activation function. In: British Machine Vision Conference (2020). <https://api.semanticscholar.org/CorpusID:221113156>
45. Mojgani, R., Balajewicz, M., Hassanzadeh, P.: Lagrangian PINNs: a causality-conforming solution to failure modes of physics-informed neural networks. *Comput. Methods Appl. Mech. Eng.* **404**, 115810 (2023). <https://doi.org/10.1016/j.cma.2022.115810>. [arXiv:2205.02902](https://arxiv.org/abs/2205.02902)
46. Pinkus, A.: Approximation theory of the MLP model in neural networks. *Acta Numer.* **8**, 143–195 (1999). <https://doi.org/10.1017/S0962492900002919>
47. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019). <https://doi.org/10.1016/j.jcp.2018.10.045>
48. Ray, D., Hesthaven, J.S.: Detecting troubled-cells on two-dimensional unstructured grids using a neural network. *J. Comput. Phys.* **397**, 108845 (2019). <https://doi.org/10.1016/j.jcp.2019.07.043>
49. Roos, H.-G., Stynes, M., Tobiska, L.: Robust Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion-Reaction and Flow Problems. Springer Series in Computational Mathematics, vol. 24, 2nd edn. Springer, Berlin (2008). <https://doi.org/10.1007/978-3-540-34467-4>
50. Saadat, M.H., Gjorgiev, B., Das, L., Sansavini, G.: Neural tangent kernel analysis or PINN for advection-diffusion equation (2022). <https://doi.org/10.48550/arXiv.2211.11716>
51. TensorFlow Developers: TensorFlow (v2.13.0). Zenodo (2023). <https://doi.org/10.5281/zenodo.8117732>
52. von Wahl, H., Richter, T.: Using a deep neural network to predict the motion of underresolved triangular rigid bodies in an incompressible flow. *Int. J. Numer. Methods Fluids* **93**(12), 3364–3383 (2021). <https://doi.org/10.1002/flid.5037>
53. Wang, Y., Xu, C., Yang, M., Zhang, J.: Less emphasis on hard regions: curriculum learning of PINNs for singularly perturbed convection-diffusion-reaction problems. *East Asian J. Appl. Math.* **14**(1), 104–123 (2024). <https://doi.org/10.4208/eajam.2023-062.170523>

54. Wolff, T., Carrillo, H., Martí, L., Sanchez-Pi, N.: Towards optimally weighted physics-informed neural networks in ocean modelling (2021). <https://doi.org/10.48550/arXiv.2106.08747>
55. Yang, L., Shami, A.: On hyperparameter optimization of machine learning algorithms: theory and practice. *Neurocomputing* **415**, 295–316 (2020). <https://doi.org/10.1016/j.neucom.2020.07.061>
56. Zang, Y., Bao, G., Ye, X., Zhou, H.: Weak adversarial networks for high-dimensional partial differential equations. *J. Comput. Phys.* **411**, 109409 (2020). <https://doi.org/10.1016/j.jcp.2020.109409>
57. Zong, Y., He, Q., Tartakovsky, A.M.: Improved training of physics-informed neural networks for parabolic differential equations with sharply perturbed initial conditions. *Comput. Methods Appl. Mech. Eng.* **414**, 116125 (2023). <https://doi.org/10.1016/j.cma.2023.116125>