Weierstraß-Institut
für Angewandte Analysis und Stochastik

Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

Bound-preserving PINNs for steady-state convection-diffusion-reaction problems

Volker John[1,2], Marina Matthaiou[3], Marwa Zainelabdeen[1,2]

submitted: October 16, 2024

[1] Weierstraß-Institut
Mohrenstr. 39
10117 Berlin
Germany

E-Mail: volker.john@wias-berlin.de
zainelabdeen.marwa@wias-berlin.de

[2] Freie Universität Berlin
Department of Mathematics and Computer Science
Arnimallee 6
14195 Berlin
Germany

[3] Technische Universität Berlin
Institute of Mathematics
Straße des 17. Juni 136
10623 Berlin
Germany

E-Mail: matthaiou@campus.tu-berlin.de

No. 3134
Berlin 2024

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

Key words and phrases. steady-state convection-diffusion-reaction problems, convection-dominated regime, PINNs, hp-VPINNs, bound-preservation.

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/

Bound-preserving PINNs for steady-state convection-diffusion-reaction problems

Volker John, Marina Matthaiou, Marwa Zainelabdeen

Abstract

Numerical approximations of solutions of convection-diffusion-reaction problems should take only physically admissible values. Provided that bounds for the admissible values are known, this paper presents several approaches within PINNs and hp-VPINNs for preserving these bounds. Numerical simulations are performed for convection-dominated problems. One of the proposed approaches turned out to be superior to the other ones with respect to the accuracy of the computed solutions.

1 Introduction

Boundary and initial-boundary value problems with partial differential equations model many processes in nature and industry. Usually it is not possible to compute solutions of those problems analytically. It is necessary to utilize some numerical method for approximating the solution. The choice of the method might be guided by different preferences, e.g., accuracy (in some norm) or efficiency. A crucial aspect in many applications is the physical consistency of the method, i.e., that important physical properties of the solution of the continuous problem are preserved by the numerical solution. Such properties comprise balance laws, physically admissible values, or divergence-free flow fields in incompressible flow problems. The physical consistency of a numerical method is not only a desirable property for the considered equation, but it might be crucial in coupled problems, where the numerical solution of one equation might be the input data for other equations. Using unphysical input data could lead to difficulties in solving the other equations and even to a blow-up of simulations, e.g., as reported in [13].

This paper considers steady-state convection-diffusion-reaction boundary value problems. Let $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, be a Lipschitz domain with boundary $\partial\Omega$, then such a problem, in dimensionless form, is given by

$$\begin{aligned} -\varepsilon \Delta u + \boldsymbol{b} \cdot \nabla u + \sigma u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where $\varepsilon > 0$ is the constant diffusion coefficient, $\boldsymbol{b}$ is the convection field, and $\sigma \geq 0$ is the reaction field. The sources are prescribed by the right-hand side f and, for simplicity of presentation, the problem is equipped with Dirichlet conditions g on $\partial\Omega$.

From the mathematical point of view it is known that solutions of (1) satisfy maximum principles for appropriate source terms, e.g., see [5, Sec. 6.4]. In practice, convection-diffusion-reaction problems model the behavior of quantities like temperature (energy balance) or concentrations. In many situations, the range $[u_{\min}, u_{\max}]$ of admissible values of the solution is known a priori, e.g., for concentrations this range is usually $[0, 1]$. For such situations methods are of most interest that satisfy the discrete counterpart of maximum principles, so-called discrete maximum principles (DMPs). But also methods that preserve the given bounds $u_{\min}$ and $u_{\max}$ and that are more sophisticated and more accurate than a simple cut off strategy are already very helpful. Such methods are called bound-preserving, to distinguish them from methods that satisfy DMPs without using a priori information on the range of admissible solution values.

Although (1) is a linear boundary value problem, its numerical solution might be challenging, in particular in the so-called convection-dominated regime. In this regime, the convective term $\boldsymbol{b} \cdot \nabla u$ dominates the equation, in particular the diffusive term $-\varepsilon \Delta u$. There is no precise mathematical definition of what is a convection-dominated regime. In practice, the size of convection, $\|\boldsymbol{b}\|_{L^\infty(\Omega)}$ in the dimensionless equations is often larger than the size of diffusion ε by five or more orders of magnitude. Solutions of (1) in the convection-dominated regime possess typically layers, which are thin regions with steep gradients. Depending on the type, the layer width is of order $\mathcal{O}(\varepsilon)$ or $\mathcal{O}(\sqrt{\varepsilon})$ if the convection field is of order $\mathcal{O}(1)$. Consequently, grid-based discretizations like finite element methods, finite difference methods, and finite volume methods cannot resolve the layers. The situation that a main feature of a solution cannot be resolved by a numerical method is the typical feature of multiscale problems. In this sense, the boundary value problem (1) constitutes from the numerical point of view a multiscale problem. It is well known that the development of physically consistent and accurate numerical methods

DOI 10.20347/WIAS.PREPRINT.3134 Berlin 2024

for multiscale problems is challenging. In fact, the recent survey [2] and monograph [3] reveal that there are only rather few finite element methods for which the satisfaction of DMPs can be proven. And many of the most accurate ones were proposed in the last decade.

Utilizing techniques from machine learning for supporting the numerical solution of partial differential equations or for directly computing a numerical approximation of the solution is currently an active topic of research. The current paper will pursue the latter approach, using physics-informed neural networks (PINNs) and hp-variational PINN (hp-VPINNs). In a nutshell, the solution of a boundary value problem for a partial differential equation is approximated by learning a neural network that minimizes a loss functional that contains information about the problem, [14]. A commonly used loss functional is an appropriate norm of the residual.

There are only rather few papers that explore the application of PINNs or hp-VPINNs to convection-dominated convection-diffusion-reaction problems. However, many of them consider only one-dimensional problems (two-point boundary value problems), see the introduction of [6] for a survey. Two-point boundary value problems are notably simpler than problems in higher-dimensional domains. For the common situation that convection does not change sign, the solution does not possess interior layers. Of course, corner singularities cannot occur. Although PINNs have demonstrated successful applications in diverse fields, they can still face difficulties when approximating the solution to singularly perturbed problems. It was demonstrated in [16] that PINNs can fail to learn the solution of convection-diffusion-reaction problems due to difficulties in optimizing the loss landscape, even outside the convection-dominated regime as considered in [16]. The authors of [21] examined advection-dispersion equations (ADE) as an example of parabolic problems with sharply perturbed initial conditions where various measures were taken to improve PINNs training. A spatially two-dimensional example demonstrated that the choice of the loss functional weights significantly influences the PINN solution's quality and thus criteria for choosing the weights of the loss functionals were proposed. Further, a normalized form of the ADE was used in PINNs to tackle the initial condition perturbation along with using an adaptive sampling scheme. The proposed measures significantly reduced the PINN approximation error for the ADE problem. Reference [15] proposed a variational form of the loss functional, which was tested on time-dependent convection-diffusion-reaction problems with diffusion coefficients ranging between 0.1 and 0.003. The study examined a test case with a boundary layer and showed that the trained PINNs can reasonably capture the solution behavior, but only one-dimensional domain test cases were considered. The emphasis of [6] was on studying different loss functionals. It was found that using others than the standard residual loss might improve the results considerably, with respect to the $L^2(\Omega)$ norm of the error. The authors of [20] investigated reasons for PINNs' failure in convection-dominated convection-diffusion-reaction problems from a domain distribution perspective. Problems in one-, two-, and three-dimensional domains with boundary and interior layers were examined and surprisingly found that selecting collocation points away from layer regions yields better results than increasing the number of points within the layers. A similar observation was made in [7], where a priori adapted distributions of the collocation points were investigated and the points were concentrated in regions with layers. This approach did not show any clear advantage compared with a uniform distribution of the collocation points. None of the mentioned papers tries to enforce a DMP for the solution obtained with the neural networks. In fact, some of the solutions presented in more detail in [6, 7] show spurious oscillations, especially in examples where the solution has very steep layers.

The paper is organized as follows. Section 2 presents the setup of the used PINNs and hp-VPINNs. It introduces in particular the approaches for preserving prescribed bounds. Details of the training process are provided in Section 3. Our numerical studies will be presented in Section 4. The paper closes with a summary and outlook.

2 Setup of the PINNs and hp-VPINNs

The simulations presented in this paper were performed with a standard multilayer perceptron (MLP) model or feedforward neural network (FNN). There are meanwhile numerous descriptions of such neural networks and it is refereed to the corresponding literature. More precisely, the used MLP is an extension of the one used in [6], where also a detailed description can be found. Here, we like to concentrate on the presentation of the necessary information concerning the loss functionals and of the extensions, namely possible techniques for obtaining bound-preserving numerical solutions of (1).

This paper studies PINNs and hp-VPINNs. Both approaches differ by the type of loss functional that should be minimized in the training process. The loss functional of PINNs is based on the residual of the strong form of the partial differential equation. In general, one distinguishes loss terms for the interior of the domain, for the Dirichlet boundary conditions, and for the Neumann boundary conditions. Since in the studied examples there are no Neumann conditions and we applied a hard-constrained imposition of the Dirichlet boundary conditions, see below, only the term for the interior of the domain is

present. Let $\{\boldsymbol{x}_i \in \Omega\}_{i=1}^N$ be the set of collocation points, then the residual loss is defined by

$$\mathcal{L}_{\mathsf{hd}}^{\mathsf{st}}(u_N) := \frac{1}{N}\sum_{i=1}^{N}\Big[\left(-\varepsilon\Delta u_N + \boldsymbol{b}\cdot\nabla u_N + \sigma u_N - f\right)(\boldsymbol{x}_i)\Big]^2, \tag{2}$$

where u_N is the solution predicted by the neural network. Note that the right-hand side of (2) can be considered as an approximation, more precisely as a Monte-Carlo approximation, of

$$\int_\Omega \Big[\left(-\varepsilon\Delta u + \boldsymbol{b}\cdot\nabla u + \sigma u - f\right)(\boldsymbol{x})\Big]^2 \,\mathrm{d}\boldsymbol{x}.$$

The definition of the residual loss in hp-VPINNs starts with a triangulation $\mathcal{T}_h$ of $\overline{\Omega}$ into mesh cells $\{K\}$. In the examples studied in this paper, these cells are squares. Then a polynomial space $P_p(K)$ can be defined with an affine transformation to a reference cell $\hat{K} = [-1,1]^2$ and the space

$$P_p(\hat{K}) := \{\varphi_j(x)\varphi_k(y) \,:\, \varphi_j, \varphi_k \in L_p([-1,1]),\quad j,k = 1,2,\cdots,p-1\},$$

where

$$L_p([-1,1]) := \{\phi_{k+1}(x) - \phi_{k-1}(x) \,:\, k = 1,2,\cdots,p-1\},$$

and ϕ_k is the Legendre polynomial of order k. The set of test functions $P_p(K)$ on the physical cell $K \in \mathcal{T}_h$ is given by

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

where F_K^{-1} is the inverse of the reference transform. Then, the set of global polynomial test functions $P_p(\mathcal{T}_h)$ is defined as

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

so that the support of all test functions $v \in P_p(\mathcal{T}_h)$ is just one mesh cell K and they satisfy $v|_{\partial\Omega} = 0$. This approach can be extended to three dimensions in a straightforward way. Now, the variational residual loss for the interior of Ω is defined by

$$\mathcal{L}_{\mathsf{hd}}^{v}(u_N) := \sum_{K\in\mathcal{T}_h}\frac{1}{|P_p(K)|}\sum_{v\in P_p(K)}\left(\int_K(-\varepsilon\Delta u_N + \boldsymbol{b}\cdot\nabla u_N + \sigma u_N - f)v\,\mathrm{d}\boldsymbol{x}\right)^2, \tag{3}$$

where $|P_p(K)|$ is the dimension of $P_p(K)$. The integrals in (3) are approximated by numerical quadrature. In our simulations, $p = 6$ was used, i.e., polynomials of degree 5, and a Gauss–Legendre quadrature with 10×10 nodes for each K was applied.

As already mentioned, the technique of hard-constrained Dirichlet boundary conditions was used. We think that this approach, instead of extending the loss functional by a term that measures the nonsatisfaction of weakly imposed Dirichlet condition, is suitable for convection-diffusion-reaction problems. First, the solution in Ω is determined in many problems by some inlet boundary condition that is transported in the domain, as in Example 4.3 below. To compute an accurate solution, it is inevitable that the boundary condition at the inlet is represented accurately. Second, using weakly imposed Dirichlet boundary conditions at outlets, we could observe in preliminary numerical studies that then often the layers at the outflow boundary, which are important features of the solution, are not present in the numerical approximation, compare also the experience reported in [17]. And third, in our opinion, Dirichlet boundary conditions are given data of the problem, that should be directly used in the training process and not be learnt.

Imposing hard-considered Dirichlet boundary conditions in PINNs and hp-VPINNs requires the use of a continuous extension $\tilde{g} \,:\, \overline{\Omega}\to\mathbb{R}$ of the Dirichlet data g and an indicator function $\ell \,:\, \overline{\Omega}\to\mathbb{R}$ satisfying

$$\ell(\boldsymbol{x}) = 0, \text{ if } \boldsymbol{x}\in\partial\Omega,\quad \ell(\boldsymbol{x}) > 0, \text{ if } \boldsymbol{x}\in\Omega.$$

As it will be discussed in Section 4, a good choice of the indicator function depends on the concrete problem that is studied.

Next, several techniques for achieving bound-preserving numerical solutions will be introduced. The first approach consists in extending the loss functional by a term that penalizes the violation of the bound-preservation, like

$$P(u_N) = \sum_i\left[\left(\max\{0, u_N(\boldsymbol{x}_i) - u_{\max}\}\right)^2 + \left(\max\{0, u_{\min} - u_N(\boldsymbol{x}_i)\}\right)^2\right],\quad i\in\mathbb{N}.$$

Then, the augmented loss functional for PINNs has the form

$$\mathcal{L}_{\text{hd}}^{\text{st,pen}}(u_N) = \lambda^{\text{st}} \mathcal{L}_{\text{hd}}^{\text{st}}(u_N) + \lambda P(u_N), \tag{4}$$

where $\mathcal{L}_{\text{hd}}^{\text{st}}(u_N)$ is given in (2) and $\lambda^{\text{st}}, \lambda > 0$ are weights that have to be chosen by the user. Similarly, the augmented loss functional for hp-VPINNs is

$$\mathcal{L}_{\text{hd}}^{\text{v,pen}}(u_N) = \lambda^{v} \mathcal{L}_{\text{hd}}^{v}(u_N) + \lambda P(u_N), \tag{5}$$

with $\mathcal{L}_{\text{hd}}^{v}(u_N)$ defined in (3). Note that this approach does not guarantee that the numerical solution takes values only in $[u_{\min}, u_{\max}]$. Whether or not this is the case depends certainly on the chosen weights.

A different class of approaches applies a condition for bound-preservation as a post-processing step. Let u_N^* be the function computed by the neural network, then the computation of the numerical solution of the convection-diffusion-reaction problem requires the incorporation of the Dirichlet boundary conditions, and this solution is given by

$$\tilde{u}_N := \tilde{g} + \ell u_N^*.$$

An easy way to obtain a bound-preserving numerical solution consists in setting

$$u_N := \max\left\{u_{\min}, \min\left\{u_{\max}, \tilde{u}_N\right\}\right\}. \tag{6}$$

This way corresponds to the classical technique of cutting off undesired values. For PINNs, (6) is applied for all collocation points and for hp-VPINNs, this approach is applied for all quadrature nodes.

A more sophisticated approach consists in applying the post-processing in such a way that it can be incorporated in a natural way in the neural network. To this end, an appropriate activation function is used in the output layer of the network. We studied this approach with two activation functions. It is assumed for this approach that the indicator function ℓ is defined such that it takes only values in $(0, 1]$ in Ω, preferably the value 1 is taken in large parts of the domain. In addition, the extension of the Dirichlet boundary condition should have the value zero in all collocation points for PINNs and all quadrature nodes for hp-VPINNs.

Let $\hat{u}_N$ be the prediction of the neural network in the layer before the output layer. The first method that we studied uses the hyperbolic tangent function as activation function for the last layer. Applying this function gives

$$u_N^* = \frac{1}{2}\left(\tanh(\hat{u}_N) + 1\right). \tag{7}$$

This function possesses values in $(0, 1)$. Finally, the function u_N^* is scaled, shifted, and the Dirichlet conditions are incorporated to give the numerical solution

$$u_N = \tilde{g} + \ell\left(u_N^* \cdot (u_{\max} - u_{\min}) + u_{\min}\right). \tag{8}$$

With the assumptions on $\tilde{g}$ and ℓ, one finds that the term in parentheses takes values in $(u_{\min}, u_{\max})$ in the collocation points (PINNs) or quadrature nodes (hp-VPINNs).

A drawback of the previous method, with respect to its accuracy, might be that u_N cannot attain the values $u_{\min}$ and $u_{\max}$ in Ω. To mitigate this issue, we pursued the same strategy with another activation function, namely the sinus function. Let $\tilde{u}_N$ be the output of the FNN, then the values of this function are mapped first to the interval $[-\pi/2, \pi/2]$ by means of the transform

$$\hat{u}_N = \alpha \tilde{u}_N + \beta, \quad \text{with } \alpha = \frac{\pi}{\tilde{u}_{N_{\text{max}}} - \tilde{u}_{N_{\text{min}}}},\ \beta = -\frac{\pi}{2}\frac{\tilde{u}_{N_{\text{min}}} + \tilde{u}_{N_{\text{max}}}}{\tilde{u}_{N_{\text{max}}} - \tilde{u}_{N_{\text{min}}}},$$

where $\tilde{u}_{N_{\text{min}}}$ and $\tilde{u}_{N_{\text{max}}}$ are the extremal values of $\tilde{u}_N$. This function is normalized to $[-1, 1]$ by

$$u_N^* = \frac{1}{2}(\sin(\hat{u}_N) + 1). \tag{9}$$

And finally, u_N^* is scaled, shifted, the Dirichlet values are set to give the numerical solution

$$u_N = \tilde{g} + \ell\left(u_N^* \cdot (u_{\max} - u_{\min}) + u_{\min}\right). \tag{10}$$

Both transformations, $\tanh$ and $\sin$, are directly embedded into the neural network as part of the activation functions. This means that each time the network processes a batch of data, the output passes through these activation functions. The $\tanh$ or $\sin$ transformation is applied to the network's output during the forward pass, before the loss functional is computed. This guarantees that the network learns to fit its predictions within the desired range throughout training. These activations control the network's output, so there is no need for additional steps after training to correct the range. Since these transformations are incorporated as part of the neural network's structure, they also influence the gradients of the loss functional. During backpropagation, the algorithm computes gradients through the $\tanh$ or $\sin$ functions and updates the weights to ensure optimal performance. The network learns how to correctly map inputs to outputs while respecting the constraints imposed by the activation functions.

Table 1: Hyperparameters chosen for PINNs' and hp-VPINNs' model training. In total, this produces 225 different combinations of hyperparameters.

Hyperparameter	Values/Approach
Learning Rate	$0.01 \cdot 3^k$ for $k = -8, -7, \ldots, 0$
Number of Hidden Layers	$5k$ for $k = 1, \ldots, 5$
Activation Functions	GELU, $\tanh$, $\sin$, Mish, Swish
Batch Size	32
Nodes per Hidden Layer	20

3 The Training Process

This section provides information on the used software and on the training process.

The programming language selected was Python 3.10, primarily due to its extensive support of prominent neural network libraries. Among the available libraries, TensorFlow was chosen as the primary machine learning framework for its proven capabilities in managing the complexities associated with large-scale neural networks [1], which are common in physics-informed modeling. Another tool that is used in the implementation and especially in evaluation of PINNs is Ray Tune [18]. Ray Tune is employed for hyperparameter tuning, leveraging its capability to explore various configurations systematically.

An appropriate combination of hyperparameters can significantly enhance the accuracy of the model's prediction. In our numerical studies, the hyperparameters number of epochs, activation function, learning rate, and number of hidden layers were examined.

According to [8] and [4], the optimal number of training epochs is identified at the point where the training error continues to decrease, but the validation error begins to increase, indicating the onset of model overfitting. In our studies, the method used to prevent overfitting is early stopping, which stops training when the model begins to diverge from optimal generalization. The application of this method in our implementation of PINNs and hp-VPINNs showed that performing 10,000 epochs is an appropriate approach.

The learning rate regulates weight adjustments based on error estimations during weight updates. Adjusting the learning rate is crucial for the performance of the model's training, as a too high rate may prevent convergence, and a too low rate may cause the model to get trapped in local minima. Following established best practices [8, 4], one approach is to initiate training with a relatively high learning rate and then gradually reduce it. Starting with a learning rate of $0.01 \cdot 3^0$ and scaling it down by a factor of three in each adjustment phase, this strategy was applied until the learning rate reached a minimal threshold of $0.01 \cdot 3^{-8}$.

Another vital hyperparameter is the number of hidden layers within the neural network. The selected values for our studies were 5, 10, 15, 20, and 25 hidden layers. This decision was influenced by the literature specific to PINNs, e.g., [19]. The choice of different numbers of hidden layers aims to understand the impact of the depth of the network on the ability of the model to learn complex patterns and to improve prediction accuracy in contexts informed by physics.

The batch size, which specifies the number of training samples processed per iteration, was set to a default value of 32. This setting was chosen since it aligns with the learning rate, following the guidance of [4]. Additionally, each hidden layer within the neural network contained 20 nodes, thus establishing a consistent structure across the network to facilitate a controlled evaluation of the model's performance and the impact of other varied hyperparameters. The networks were initialized with the Glorot (or uniform Xavier) initialization.

Finally, the selection of the activation functions for the hidden layers was evaluated. The activation functions considered include GELU (Gaussian Error Linear Unit), $\tanh$, $\sin$, Mish, and Swish, where

$$\begin{aligned}
\mathsf{GELU}(x) &= 0.5x\left(1+\tanh\left(\sqrt{\frac{2}{\pi}}\left(x+0.044715x^3\right)\right)\right),\\
\mathsf{Mish}(x) &= x\,\tanh\left(\mathsf{softplus}(x)\right), \quad \mathsf{softplus}(x)=\log\left(1+\mathrm{e}^x\right),\\
\mathsf{Swish}(x) &= x\,\mathsf{sigmoid}(\beta x), \quad \mathsf{sigmoid}(\beta x)=\frac{1}{1+\mathrm{e}^{-\beta x}}, \quad \beta=1.
\end{aligned}$$

These choices are inspired by their effectiveness in various PINN contexts, aiming to optimize the model's ability to capture and represent the underlying physics of the problem being solved, [19].

The studied hyperparameters are summarized in Table 1. Through Ray Tune, parallel simulations was conducted across

the 225 different hyperparameter configurations. This methodology allows for an efficient identification of superior hyperparameters.

Finally, the evaluation metrics for assessing the performance of the models will be discussed. Examples 4.1 and 4.2 possess prescribed analytic solutions. In this situation it is possible to compute errors. A discussion on different norms and numerical experience with them can be found in [6]. Like in [6, 7], a discrete approximation of the error in $L^2(\Omega)$ is monitored. This approximation is achieved by numerical quadrature, where the domain was divided into $10,000$ squares of equal size and a Gauss–Legendre quadrature rule with ten points in each coordinate direction was applied, leading to $1,000,000$ weights and points. Example 4.3 does not possess a known analytic solution. For this example, two strategies for assessing the numerical results were pursued, which will be explained in the presentation of the example.

4 Numerical Studies

The numerical studies consider three convection-dominated convection-diffusion(-reaction) problems. Examples 4.1 and 4.2 possess a known analytic solution. They were already studied in [6]. Example 4.3 is a popular benchmark problem, without known analytic solution.

For each example, ten methods to compute a numerical solution were studied. These methods comprise, on the one hand, the use of PINN and hp-VPINN, and on the other hand, the standard residual loss and the four approaches for enforcing bound preservation. For all methods, 255 different network configurations were applied over $1,000$ epochs. Following this, the best 10 configurations for each model and functional, e.g., defined as those that produce the lowest $L^2(\Omega)$ error for examples with an analytic solution, were selected. These models were started from the beginning and then trained for $10,000$ epochs, and the best model for each example was determined based on the chosen metric of evaluating the solutions.

The methods with respect to the bound preservation are abbreviated below by *wo_bound* for the approaches (2) and (3) that do not care about a bound preservation, *bound_pen* for the approaches (4) and (5) that contain a penalty term in the loss functional, *bound_cut* for just cutting off undesired values, see (6), *bound_tanh* for applying the hyperbolic tangent in the output layer (7), (8), and *bound_sin* for using the sinus in this layer (9), (10). The weights in *bound_pen* were chosen to be $\lambda^{\text{st}} = \lambda^v = \lambda = 1$.

4.1 Solution with a Circular Interior Layer

The coefficients of (1) for this problem are given by $\Omega = (0,1)^2$, $\varepsilon := 10^{-8}$, $\boldsymbol{b} = (2,3)^T$, and $\sigma = 2$. The right-hand side and the Dirichlet boundary condition are chosen such that

$$u(x,y) = 16x(1-x)y(1-y)\left(\frac{1}{2} + \frac{\arctan\left(200(r_0^2 - (x-x_0)^2 - (y-y_0)^2)\right)}{\pi}\right), \tag{11}$$

with $r_0 = 0.25$ and $x_0 = y_0 = 0.5$ is the solution of (1), see Figure 1. The solution has an interior layer of circular shape. Its extremal values are $u_{\min} = 0$ and $u_{\max} = 0.974$.

The hard-constrained Dirichlet boundary conditions were imposed with the extension $\tilde{g}(x,y) = 0$ and the indicator function

$$\ell(x,y) = \left(1 - \mathrm{e}^{-\kappa_1 x}\right)\left(1 - \mathrm{e}^{-\kappa_2 y}\right)\left(1 - \mathrm{e}^{-\kappa_3(1-x)}\right)\left(1 - \mathrm{e}^{-\kappa_4(1-y)}\right), \tag{12}$$

with $\kappa_i = 30$ for $i = 1,\ldots,4$, see Figure 1. Since the layers of the solution are away from $\partial\Omega$, the choice of the parameters in (12) is not of big importance of this example, as long as the parameters are not too small. Note that ℓ satisfies the assumptions from Section 2.

As already mentioned, the accuracy of the computed solutions was assessed by computing the error in $L^2(\Omega)$ to the prescribed solution (11). Training errors for each of the ten methods and each of the 255 network configurations after $1,000$ epochs are presented in Figure 2. It can be seen that the lowest $L^2(\Omega)$ error in almost every configuration is obtained by the method *bound_sin*. In the case of hp-VPINNs, the model that performed worse in almost every configuration is *wo_bound*, with an average error of 0.4435. The hyperparameters that yielded the best model performance consisted of ten hidden layers and the learning rate that provided the best results was established at $0.01 \cdot 3^{-3}$. Concerning the activation function, the GELU function was identified as the most effective one.

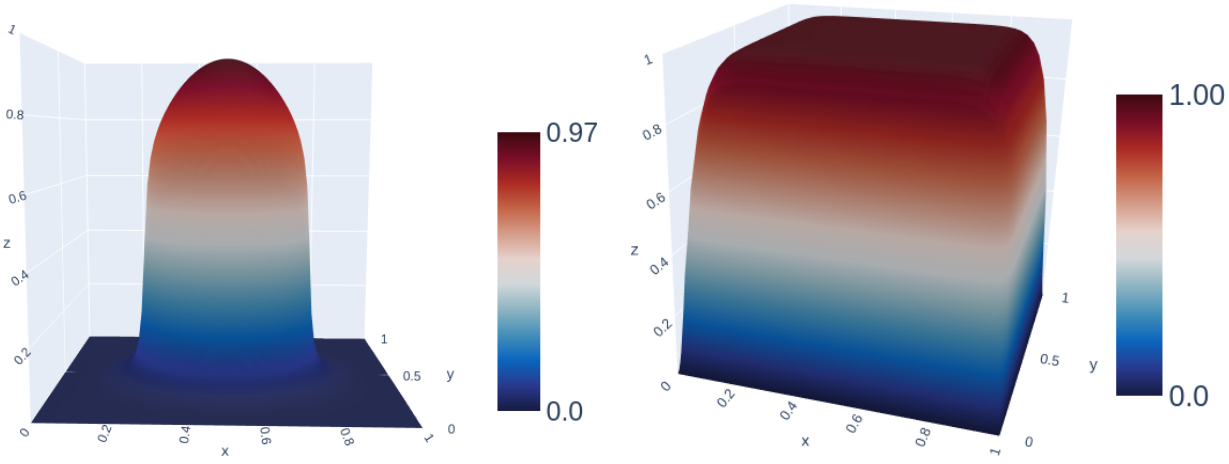

Figure 1: Example 4.1. Prescribed solution (left) and indicator function for the hard-constrained Dirichlet boundary conditions (right).

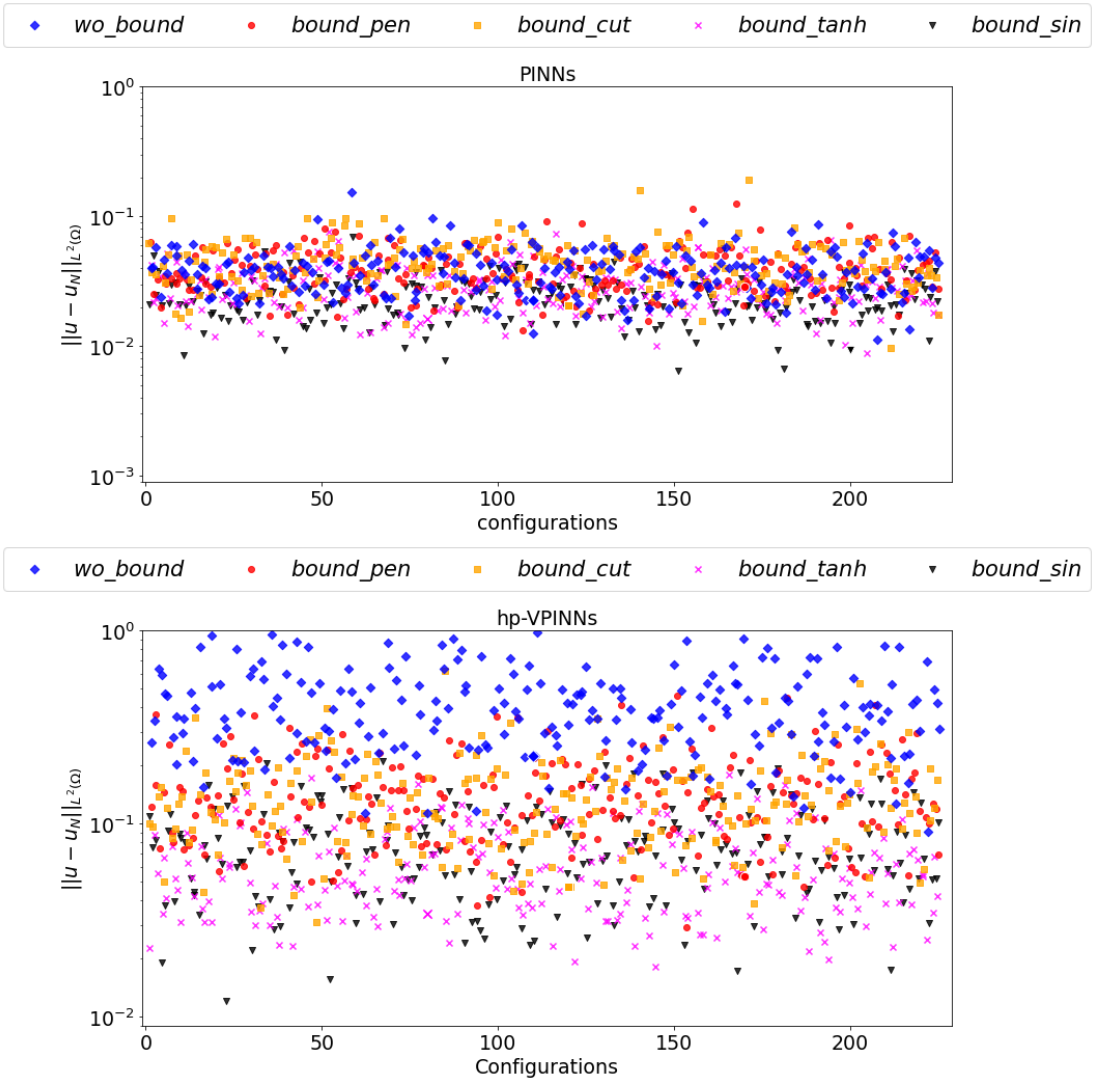

Figure 2: Example 4.1. Errors $\|u - u_N\|_{L^2(\Omega)}$ after $1,000$ epochs for all 225 configurations of hard-constrained PINNs (top) and hp-VPINNs (bottom).

Table 2: Example 4.1. Best errors $\|u - u_N\|_{L^2(\Omega)}$ after $1,000$ epochs.

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.0226	0.0288	0.0207	0.0161	0.0094
hp-VPINNs	0.2235	0.0533	0.0471	0.0340	0.0208

Table 3: Example 4.1. Best errors $\|u-u_N\|_{L^2(\Omega)}$ after $10,000$ epochs. The results with *wo_bound* after $100,000$ epochs in [6] are 0.00156 (PINNs) and 0.07573 (hp-VPINNs).

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.0092	0.0086	0.0229	0.0023	0.0010
hp-VPINNs	0.1183	0.0344	0.0820	0.0201	0.0178

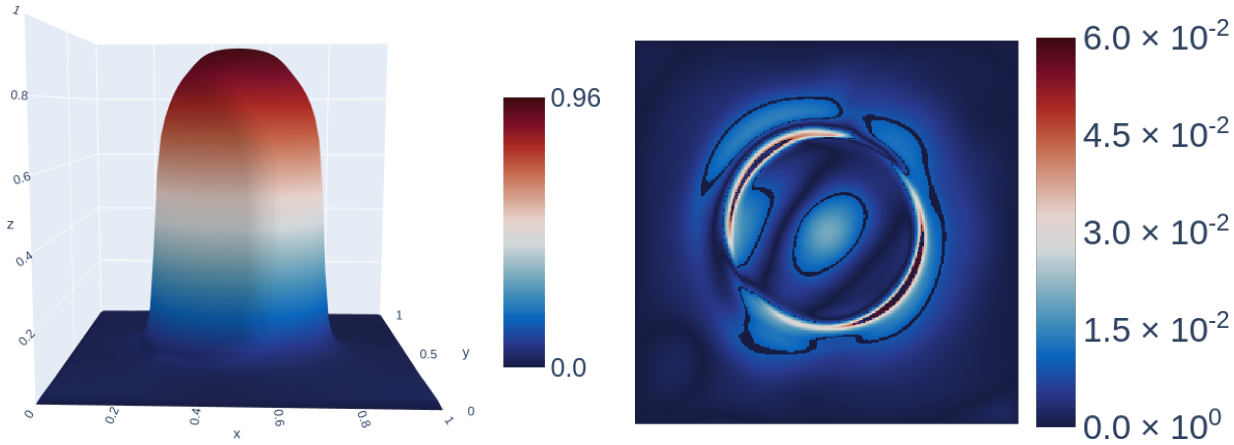

Figure 3: Example 4.1. PINN with *bound_sin* after $10,000$ epochs, solution (left) and pointwise error (right).

Table 2 presents the best results for the ten methods after a training period of $1,000$ epochs. It shows that using *bound_sin* outperforms the other models. The method *bound_tanh* is next with respect to accuracy, followed by *bound_cut*. In contrast, *wo_bound* and *bound_pen* have the highest errors.

The results obtained after training the top configurations for $10,000$ epochs are presented in Table 3. The most effective method is still *bound_sin*. Comparing with the result after $1,000$ epochs, an error reduction by a factor of about nine is achieved for PINNs. Also for the other methods, but *bound_cut*, noticeable reductions of the errors are achieved. Concerning the error reduction, the situation is somewhat different for hp-VPINNs, for which usually only a rather small reduction could be observed, again save for *bound_cut*. Altogether, considerably more accurate solutions could be computed with PINNs than with hp-VPINNs. In the solutions computed with *wo_bound* we could observe some overshoots.

Comparing the obtained results with those from [6], where different architectures with respect to the number of hidden layers and number of nodes per hidden layer than in our simulations were used, one can see that the results for PINNs with *wo_bound* after having performed $100,000$ epochs are much better than the results obtained in our simulations with *wo_bound* after $10,000$ epochs. But the solutions computed with *bound_sin* after $10,000$ epochs are even somewhat more accurate. For hp-VPINNs, a similar comment can be made, but here also *bound_pen* and *bound_tanh* gave more accurate results than the one from [6].

The computed solutions and the corresponding pointwise errors for the best methods from Table 3 are presented in Figures 3 and 4. It can be seen that for the PINN solution the maximal value is quite close to $u_{\max}$ and the largest errors are committed in the layer regions. The second statement is also true for the hp-VPINN solution. But the maximal value of this solution is noticeable smaller than $u_{\max}$.

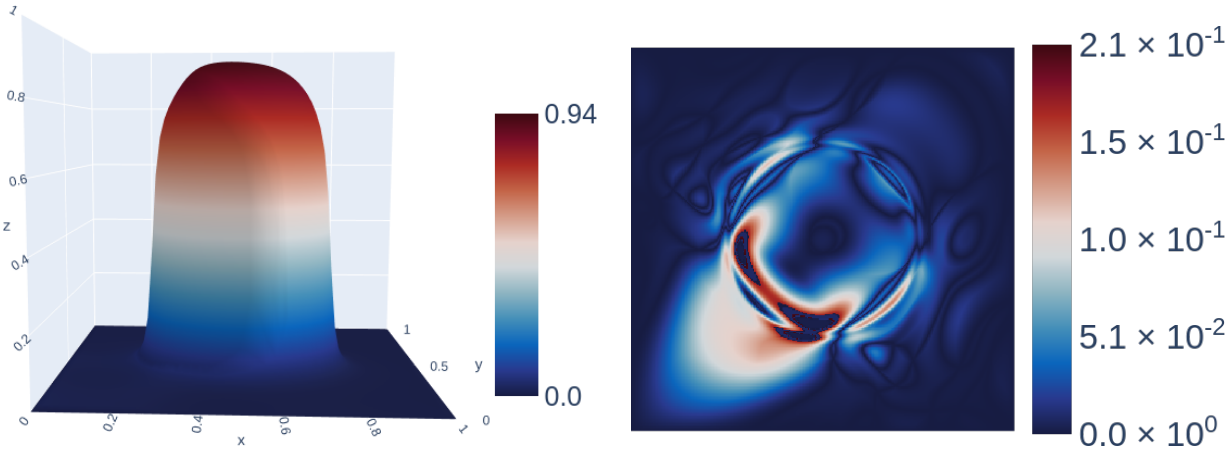

Figure 4: Example 4.1. hp-VPINN with *bound_sin* after $10,000$ epochs, solution (left) and pointwise error (right).

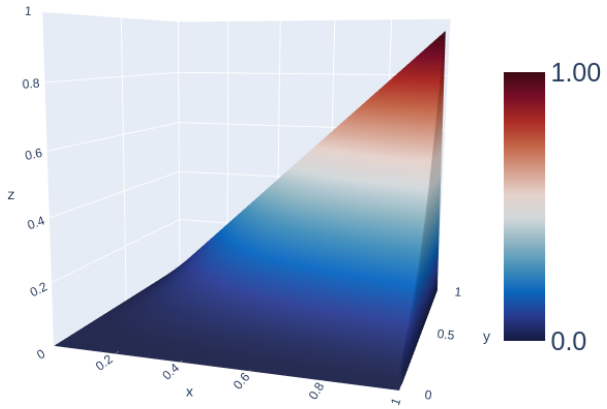

Figure 5: Example 4.2. Prescribed solution.

4.2 Solution with Layers at the Outflow Boundary

This problem, proposed in [12], is defined by $\Omega = (0,1)^2$ and by the coefficients $\varepsilon = 10^{-8}$, $\boldsymbol{b} = (2,3)^T$, and $\sigma = 1$ in (1). The prescribed solution has the form

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

see Figure 5. The right-hand side f and the Dirichlet boundary conditions g on $\partial\Omega$ are determined with the known solution. This solution exhibits boundary layers at the outflow boundaries $x = 1$ and $y = 1$ and a corner singularity at the upper right corner of the domain. The maximum value of the solution is very close to 1, we took $u_{\max} = 1$, and its minimum is very close to 0, so that we took $u_{\min} = 0$.

The boundary conditions are basically homogeneous. To impose the hard-constrained Dirichlet conditions, $\tilde{g}(x,y) = 0$ and an indicator function of type (12) were used. The parameters $\kappa_i = {}^1\!/_{10\varepsilon} = 10^7$, $i = 1,\dots,4$, were chosen to account for the steep boundary layers. This is the similar approach as in [6, 7], where the parameter was 10^9. The indicator function has the same principal shape as the indicator function presented in Figure 1 and thus it satisfies the assumptions from Section 2.

Results with respect to the $L^2(\Omega)$ error for all ten methods and all 255 network configurations are depicted in Figure 6. It can be seen that the model with the lowest average error for PINNs (0.0394) and for hp-VPINNs (0.0408) is *bound_sin*. The method with the highest average error is *bound_pen*, with an average error of 0.1135 for PINNs and 0.2331 for hp-VPINNs. It turned out that the best learning rate in this example was $0.01 \cdot 3^{-3}$, the most effective number of hidden layers was ten, and the best activation function for PINNs was $\tanh$ and for hp-VPINNs Mish.

Tables 4 and 5 present the best results for each method after $1,000$ and $10,000$ epochs, respectively. Again, the smallest errors were obtained in all cases with *bound_sin*, but also the results computed with *bound_tanh* are quite good. At any rate, all results with these methods are noticeably better than the corresponding ones obtained with *wo_bound*. In

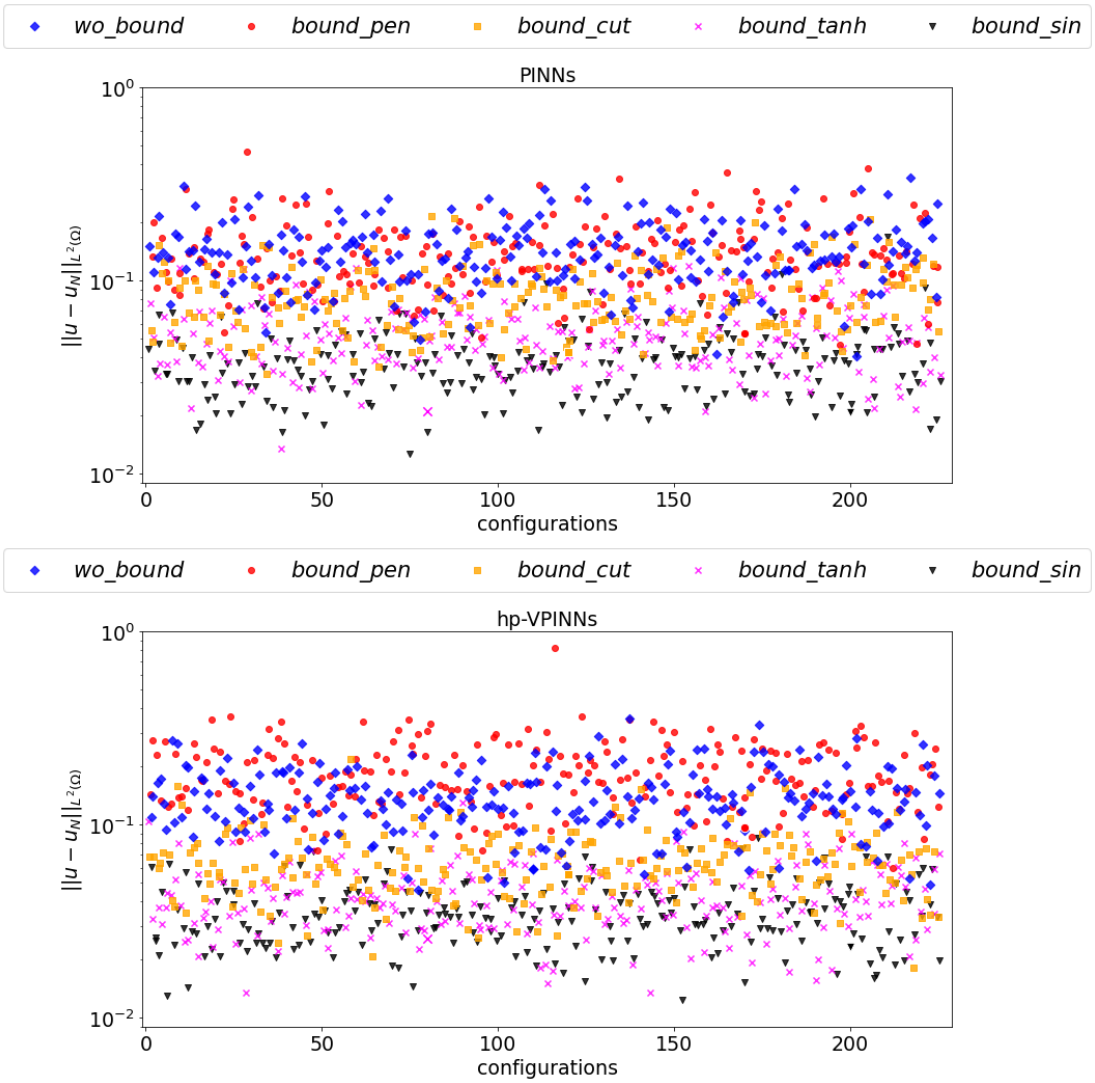

Figure 6: Example 4.2, $\kappa_i = 1/10\varepsilon = 10^7$, $i = 1, \ldots, 4$. Errors $\|u - u_N\|_{L^2(\Omega)}$ after $1,000$ epochs for all 225 configurations of hard-constrained PINNs (top) and hp-VPINNs (bottom).

Table 4: Example 4.2, $\kappa_i = 1/10\varepsilon = 10^7$, $i = 1, \ldots, 4$. Best errors $\|u - u_N\|_{L^2(\Omega)}$ after $1,000$ epochs.

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.0712	0.0533	0.0429	0.0212	0.0208
hp-VPINNs	0.0589	0.0962	0.0511	0.0258	0.0233

Table 5: Example 4.2, $\kappa_i = 1/10\varepsilon = 10^7$, $i = 1, \ldots, 4$. Best errors $\|u - u_N\|_{L^2(\Omega)}$ after $10,000$ epochs. The results with *wo_bound* after $100,000$ epochs in [6] are 0.06457 (PINNs) and 0.03619 (hp-VPINNs).

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.0348	0.0335	0.0298	0.0202	0.0131
hp-VPINNs	0.0475	0.0416	0.0219	0.0221	0.0215

addition, they are also better than those from [6] with $100,000$ epochs, where however, a somewhat different network architecture was used in [6]. We could observe some undershoots in the solutions from *wo_bound*. The results computed with *bound_pen* are unsatisfactory, compared with those from *bound_sin* and *bound_tanh*. For this example, using PINNs was usually better than hp-VPINNs, but the differences are not that large as in Example 4.1.

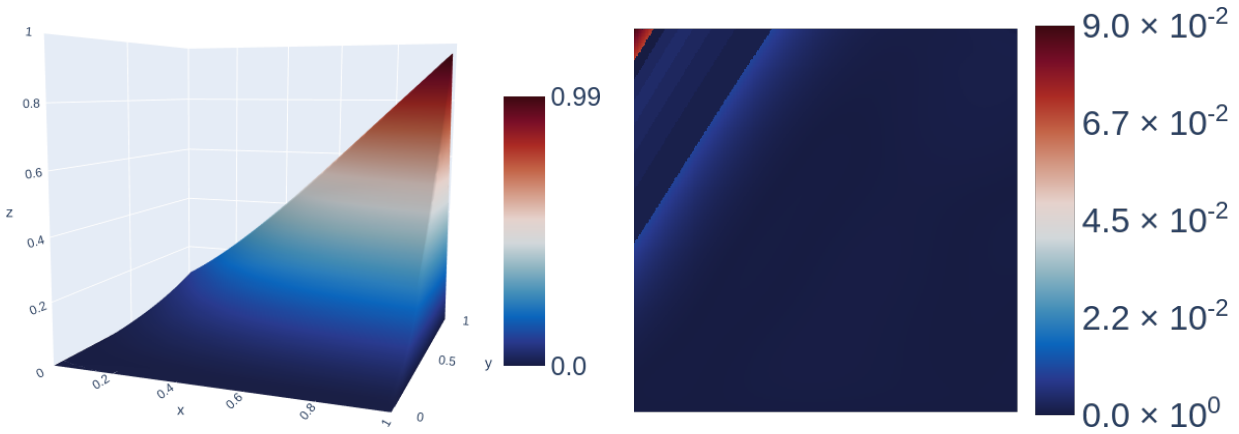

Figure 7: Example 4.2, $\kappa_i = {}^1/_{10\varepsilon} = 10^7$, $i = 1, \ldots, 4$. PINN with *bound_sin* after $10,000$ epochs, solution (left) and pointwise error (right).

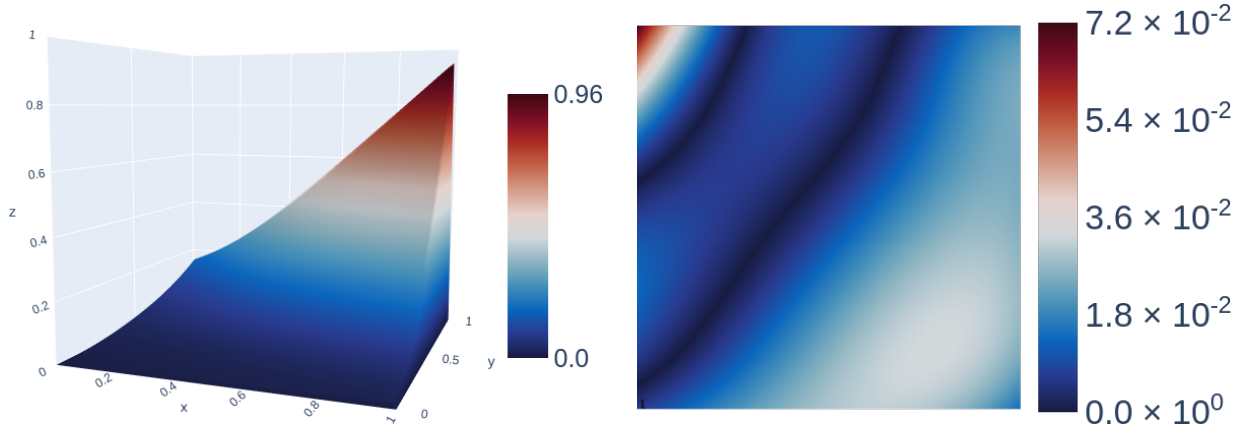

Figure 8: Example 4.2, $\kappa_i = {}^1/_{10\varepsilon} = 10^7$, $i = 1, \ldots, 4$. hp-VPINN with *bound_sin* after $10,000$ epochs, solution (left) and pointwise error (right).

Table 6: Example 4.2, $\kappa_1 = \kappa_2 = 50$, $\kappa_3 = \kappa_4 = {}^1/_{10\varepsilon}$. Best errors $\|u - u_N\|_{L^2(\Omega)}$ after $1,000$ epochs.

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.0032	0.0032	0.0020	0.0011	0.0010
hp-VPINNs	0.0190	0.0214	0.0195	0.0187	0.0185

Figures 7 and 8 present the solutions with the smallest errors for PINNs and hp-VPINNs, respectively. It can be seen that the maximal value of the PINN solution is very close to $u_{\max}$, which is a big improvement compared with the best result in [6], where this value is around 0.88. The maximal value of the best hp-VPINN solution is somewhat smaller than for the PINN solution.

A striking observation in Figures 7 and 8 is that the largest errors occur at the left upper corner of the domain and not in the boundary layer regions. This feature was already observed in the papers [6, 7], which were used to compare our results with. We suspected that the concrete choice of the indicator function $\ell(x, y)$ is responsible for this behavior. To verify this conjecture, we performed studies where at the inlet boundaries a much smaller value of the parameter in $\ell(x, y)$ was used than at the outflow boundaries. With such parameters, the steepness of the layers of the indicator function is larger at the outflow boundaries, where the solution has boundary layers, than at the inlet boundaries. More precisely, the values $\kappa_1 = \kappa_2 = 50$, $\kappa_3 = \kappa_4 = {}^1/_{10\varepsilon}$ were chosen.

Results obtained with the modified indicator functions are presented in Tables 6 and 7 and Figures 9 and 10. In fact, one can observe a tremendous increase of the accuracy with the modified indicator function. Whereas the best results with the former approach have a $L^2(\Omega)$ error of about 10^{-2} after $10,000$ epochs, now the best errors are 10^{-3} after $1,000$ epochs and of order 10^{-4} after $10,000$ epochs. The largest error for the best PINN solution is now committed at the

Table 7: Example 4.2, $\kappa_1 = \kappa_2 = 50$, $\kappa_3 = \kappa_4 = 1/10\varepsilon$. Best errors $\|u - u_N\|_{L^2(\Omega)}$ after $10,000$ epochs.

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.00029	0.00021	0.00021	0.00020	0.00015
hp-VPINNs	0.00415	0.00424	0.00391	0.00223	0.00220

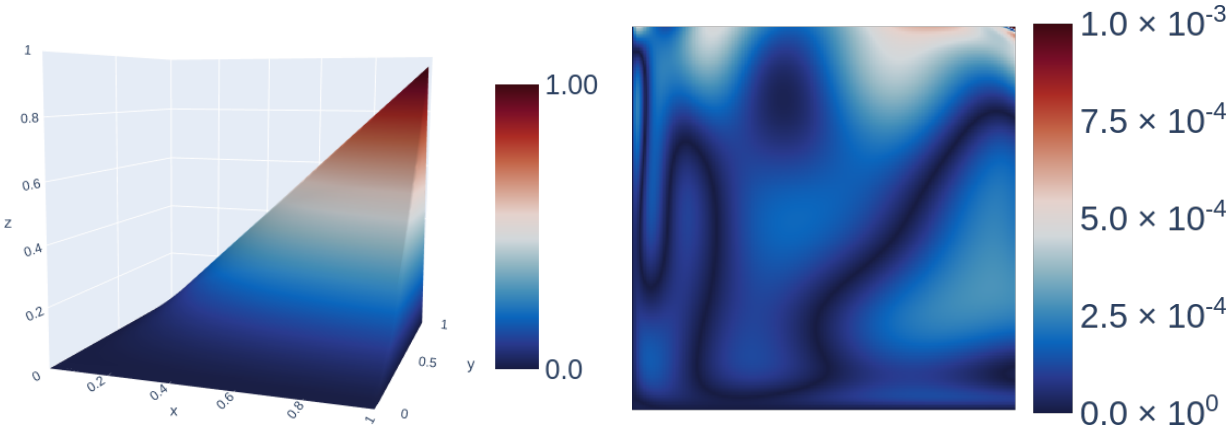

Figure 9: Example 4.2, $\kappa_1 = \kappa_2 = 50$, $\kappa_3 = \kappa_4 = 1/10\varepsilon$. PINN with *bound_sin* after $10,000$ epochs, solution (left) and pointwise error (right).

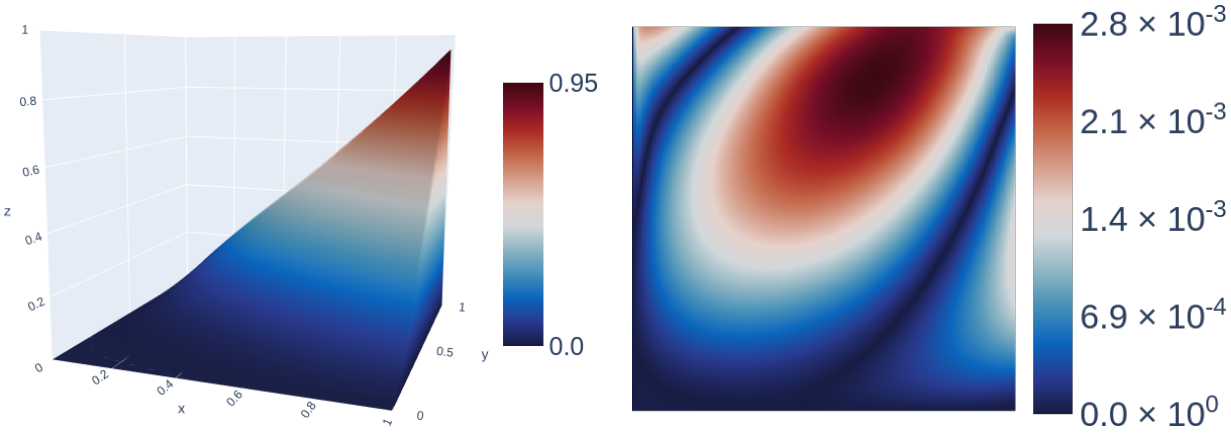

Figure 10: Example 4.2, $\kappa_1 = \kappa_2 = 50$, $\kappa_3 = \kappa_4 = 1/10\varepsilon$. hp-VPINN with *bound_sin* after $10,000$ epochs, solution (left) and pointwise error (right).

corner singularity in the right upper corner of the domain, see Figure 9.

Comparing the different methods, the same principal behavior can be observed as in the studies with the former indicator function. Using PINNs leads now to much more accurate results than applying hp-VPINNs. The latter methods have some difficulties to predict the correct gradient of the solution in Ω, compare Figure 10. And *bound_sin* was again the most accurate method.

4.3 Benchmark Problem with a Convection Field Skew to the Mesh

This type of example is proposed in [9]. It is given by $\Omega = (0,1)^2$, $\varepsilon = 10^{-8}$, $\boldsymbol{b} = (\cos(-\pi/3), \sin(-\pi/3))^T$, $\sigma = f = 0$. The Dirichlet conditions at $\partial\Omega$ are prescribed as follows

$$g = \begin{cases} 1 & \text{if } (y = 1 \wedge x > 0) \text{ or } (x = 0 \wedge y > 0.75), \\ 0 & \text{elsewhere on } \partial\Omega. \end{cases}$$

Note that this definition is slightly different than in [9], where the jump in the boundary condition is prescribed for $y = 0.7$. The problem simulates a scenario where a quantity is transported through a square domain. The solution, visualized in

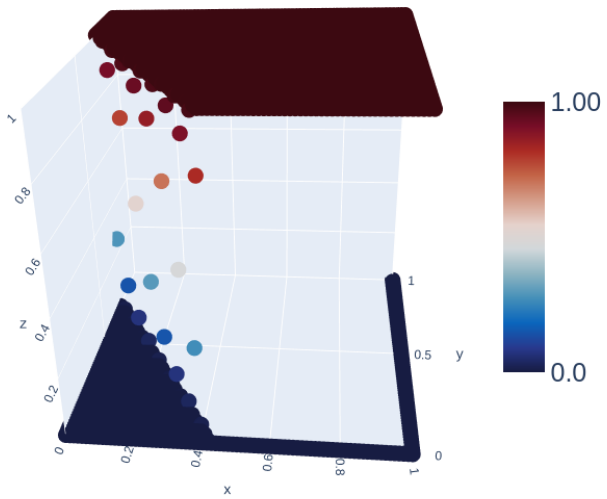

Figure 11: Example 4.3. Sketch of the solution computed with the AFC MUAS method from [11].

Figure 11, spans values between 0 and 1 and features several distinct layers: an interior layer aligned with the direction of convection beginning at the jump of the boundary condition and two boundary layers at the outflow boundary. An analytic solution for this problem is not known.

The extension of the Dirichlet data for imposing the boundary conditions[1] was defined by

$$\tilde{g}(x,y) = \frac{1}{1+\mathrm{e}^{-1000(y-0.75)}} - \frac{1}{1+\mathrm{e}^{-1000(y-0.75)}} \left(1-\mathrm{e}^{1000(y-1)}\right)\left(1-\mathrm{e}^{-1000x}\right). \tag{13}$$

Again, we used an indicator function of form (12). Based on the experience from the second study of Example 4.2, we tried different values of the parameters in this function. We could not find a selection of parameters where all results are better than for other choices. The results presented below were obtained for $\kappa_1 = \kappa_4 = 50$ and $\kappa_2 = \kappa_3 = 1/10\varepsilon = 10^7$, which belonged to the best sets of parameters in our preliminary studies. The indicator function has a very similar shape as the indicator function depicted in Figure 1. Both functions $\tilde{g}$ and ℓ satisfy the assumptions from Section 2

For this example, one has to find a different metric for evaluating the numerical results than computing errors to a prescribed solution. We pursued two approaches.

The first one consists in studying a quantity of interest, which is here the width of the interior layer. The process of determining the layer width involves a detailed examination along a certain cross-section and evaluating the value transitions of the numerical solutions to calculate the width. To this end, the line at $y = 0.3$ for $x \in [0.1, 1]$ was studied. This line is decomposed into $100,000$ equidistant intervals. At each node on this line, the numerical solutions were evaluated. The focus was on identifying two critical points: x_{start}, the first coordinate x where $u_N(x, 0.3)$ exceeds the value 0.1, indicating the onset of the layer region, and x_{end}, the point where $u_N(x, 0.3)$ exceeds a threshold of 0.9, marking the end of the layer. These points were calculated via interpolating the values at the nodes of the line. The width of the layer is calculated as the difference $x_{\text{end}} - x_{\text{start}}$. Since the used resolutions for the PINNs and hp-VPINNs were quite coarse, our expectation is that the layer width of the computed solutions is always larger than the actual layer width, and so the results with small layer width are more accurate than those with large layer width. The whole approach follows a proposal from [10].

In Figure 12, the layer widths after $1,000$ epochs are presented for all 225 configurations. The approach yielding the smallest average layer width is *bound_sin*, achieving the widths 0.031 and 0.044 for PINNs and hp-VPINNs, respectively. The method that results in the highest average layer width for PINNs is *wo_bound*, with the value 0.072, and in the case of hp-VPINNs, it is *bound_pen* with the value 0.084. The smallest layer widths are presented in Table 8, revealing that the best results are obtained with *bound_sin* and *bound_tanh*. Once more, the use of PINNs was better than using hp-VPINNs. Concerning the networks, it turned out that the best learning rate was again $0.01 \cdot 3^{-3}$, the most effective activation function was GELU, and the number of hidden layers that worked best was 10.

Table 9 presents the best layer widths after $10,000$ epochs. It can be seen that there are improvements compared with the results after $1,000$ epochs. In particular, the methods with inaccurate results after $1,000$ epochs improved noticeable. Now, the differences between the methods are small, but still it can be seen that some approaches that impose bound preservation, like *bound_sin* and *bound_tanh*, lead to somewhat steeper layers. The results with PINNs are still more

[1] In our implementation of the methods we used in fact the function given in (13). It is probably sufficient just to provide a routine that gives the value of the Dirichlet condition if $\boldsymbol{x} \in \partial\Omega$ and otherwise the value zero.

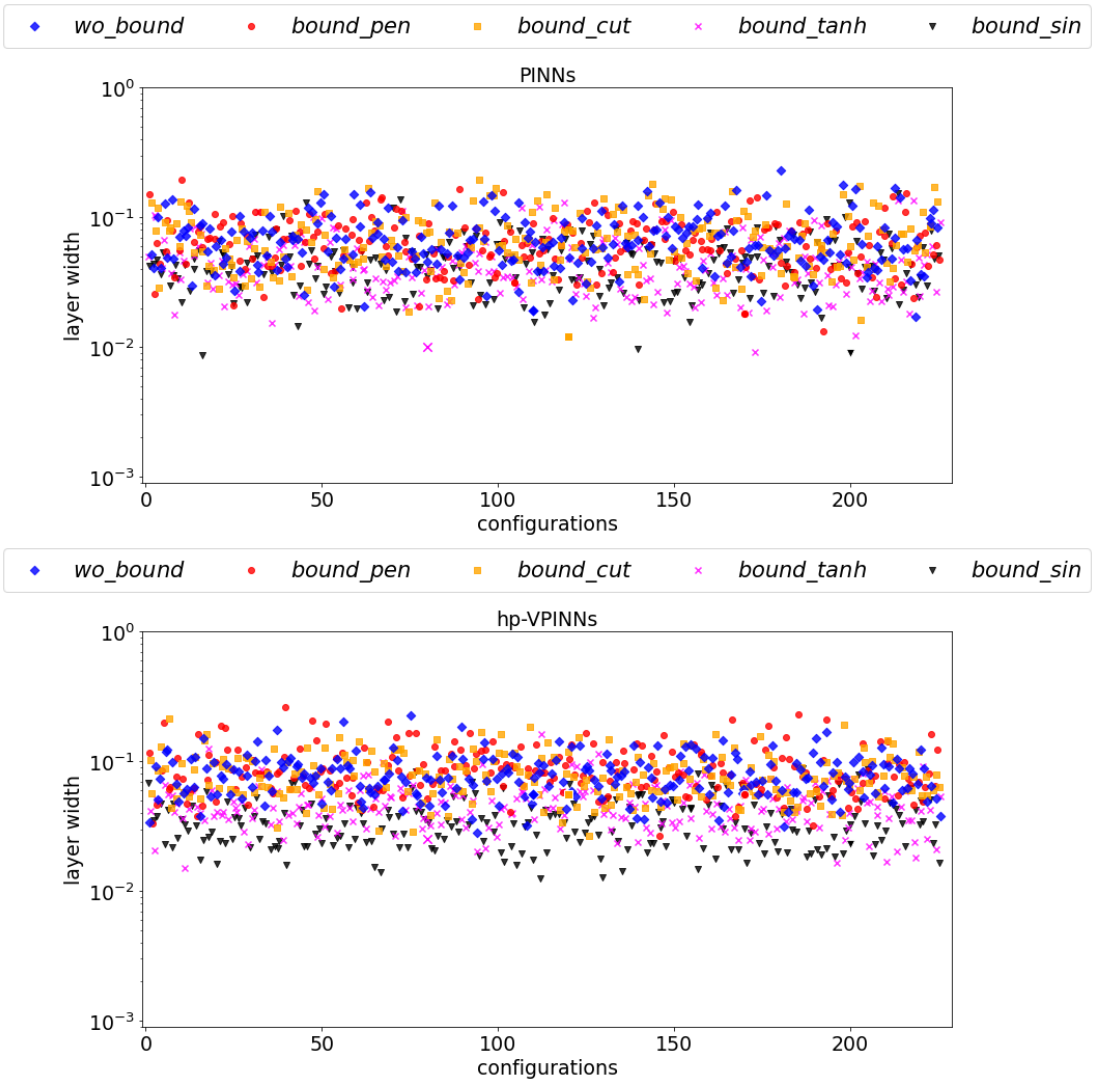

Figure 12: Example 4.3. Layer widths after $1,000$ epochs for all 225 configurations of hard-constrained PINNs (top) and hp-VPINNs (bottom).

Table 8: Example 4.3. Best layer widths after $1,000$ epochs.

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.019	0.018	0.012	0.010	0.009
hp-VPINNs	0.056	0.056	0.056	0.025	0.023

Table 9: Example 4.3. Best layer widths after $10,000$ epochs.

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.009	0.007	0.007	0.006	0.006
hp-VPINNs	0.018	0.018	0.017	0.016	0.016

accurate than those with hp-VPINNs. For this example, we observed overshoots of the order of 10^{-2} in the case of *wo_bound*. During training, both the methods *wo_bound* and *bound_pen* exhibited over- and undershoots of the same order of magnitude, but these deviations diminished by the end of the training period.

The second metric that was utilized to assess the numerical solutions used values of an accurate numerical solution as

Table 10: Example 4.3. Best errors $\|u - u_N\|_{L^2(\Omega)}$ after $10,000$ epochs, the error calculation was based on the values of the MUAS solution.

	wo_bound	*bound_pen*	*bound_cut*	*bound_tanh*	*bound_sin*
PINNs	0.059	0.053	0.047	0.022	0.020
hp-VPINNs	0.061	0.060	0.050	0.044	0.041

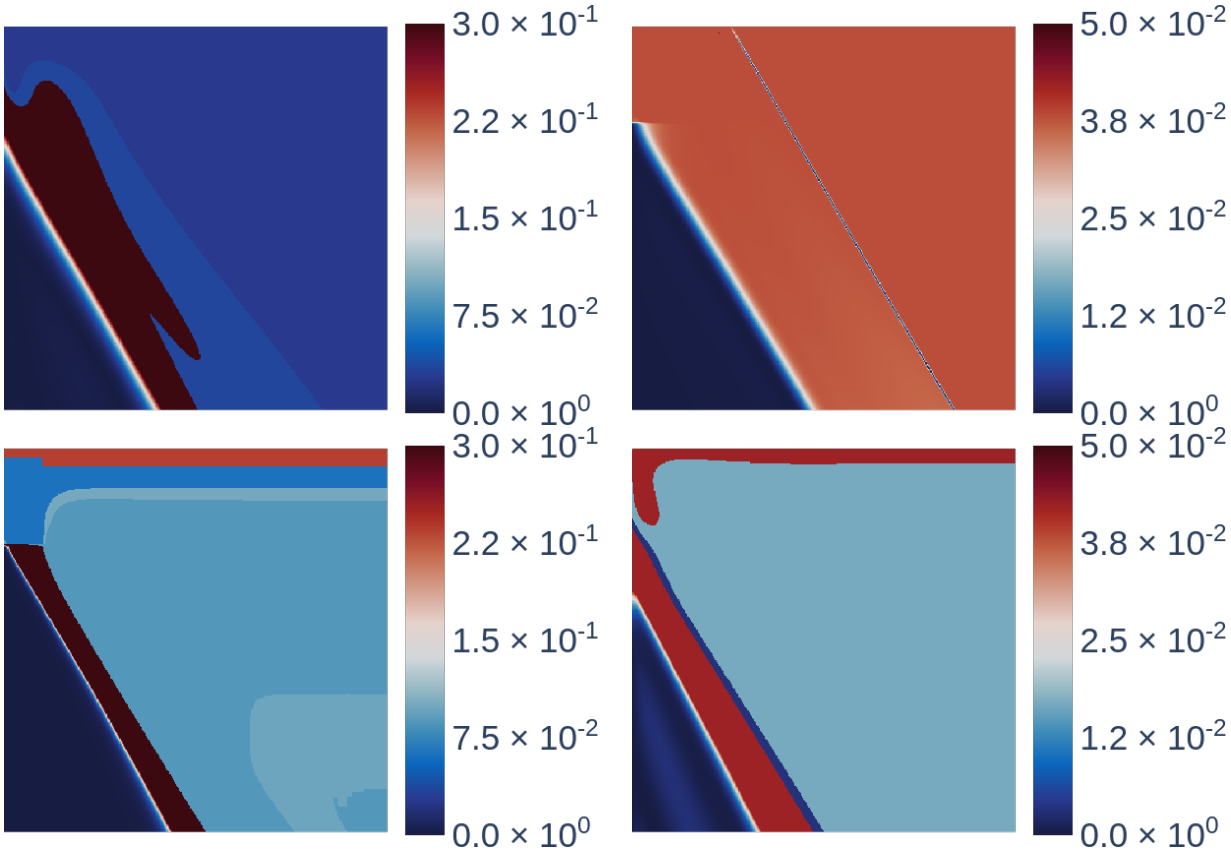

Figure 13: Example 4.3. Pointwise errors for the best solutions obtained with *wo_bound* (left) and *bound_sin* (right); PINNs (top), hp-VPINNs (bottom).

reference values. To this end, the problem was solved with an algebraically stabilized method, the MUAS method from [11], on a uniform triangulation with $1,682,209$ degrees of freedom. Algebraically stabilized methods are currently the most promising methods for convection-dominated problems if the satisfaction of discrete maximum principles is of importance, see [2]. Then, the values in the collocation points were used as reference values to calculate an approximation of the $L^2(\Omega)$ error, which, for simplicity of notation, will be denoted with the same symbol as before. For the sake of brevity, only the results obtained after $10,000$ epochs are presented, see Table 10. This table reveals that *bound_sin* and *bound_tanh* remain the best methods, both for PINNs and hp-VPINNs. Again, the application of PINNs led to more accurate results.

Error plots for the solutions from Table 10 obtained with *wo_bound* and *bound_sin* are presented in Figure 13. It can be seen that the largest errors are committed at the interior layer. Also the shortcomings of the solutions computed with hp-VPINNs are clearly visible, with a large error in the vicinity of the interior layer and a noticeable error at the upper boundary for *bound_sin*.

5 Summary and Outlook

This paper presented approaches that aimed to improve the accuracy of numerical solutions computed with PINNs and hp-VPINNs for steady-state convection-diffusion-reaction problems. To this end, several methods for enforcing the preservation of a priori known bounds were proposed and studied numerically for convection-dominated problems. First, it turned

out that, with the setups and hyperparameters we had studied, PINNs gave always noticeable more accurate results than hp-VPINNs. And second, the most accurate solutions were generally computed with the method *bound_sin* defined in (9), (10). In particular, these solutions are often considerably more accurate than the solutions obtained with the standard method *wo_bound* given in (2) and (3), respectively. Another important observation is that a carefully chosen indicator function for the hard-constrained Dirichlet boundary conditions, which distinguishes between inlet and outlet boundaries, might improve the solution substantially. Altogether, the results presented for Examples 4.1 and 4.2 are considerably more accurate than those in [6, 7]

Despite the encouraging results presented in this paper, we think that it is still an open problem whether PINNs or hp-VPINNs are appropriate methods for solving convection-dominated convection-diffusion-reaction problems. There are still the long training times, in our simulations similarly to those reported in [6]. This issue might be solved by utilizing GPU servers for the training. In addition, reasonably accurate solutions for more complicated problems, like for [7, Problem 1], could not be obtained so far with PINNs or hp-VPINNs. It could be observed that the choice of the indicator function for the hard-constrained Dirichlet boundary conditions possesses a great impact on the accuracy of the computed solutions. Appropriate functions were defined for the considered examples based on previous experience. However, it would be preferable that good indicator functions are determined automatically, i.e., if they are learnt by the network. And finally, the development of strategies for evaluating numerical solutions for problems where an analytic solution is not known is important. If networks with several hyperparameters are investigated, like in our studies, then the accuracy of a large set of numerical solutions has to be evaluated. And using the numerical solution of a different method for comparison, e.g., as obtained with the MUAS method for Example 4.3, then it is not possible to see whether the PINN or hp-VPINN solutions are more accurate than the solution of the other method.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, *Tensorflow: A system for large-scale machine learning*, 2016.

[2] G. R. Barrenechea, V. John, and P. Knobloch, *Finite element methods respecting the discrete maximum principle for convection-diffusion equations*, SIAM Rev., 66 (2024), pp. 3–88.

[3] G. R. Barrenechea, V. John, and P. Knobloch, *Monotone Discretizations for Elliptic Second Order Partial Differential Equations*, Springer Series in Computational Mathematics, Springer, Cham, 2025. to appear.

[4] Y. Bengio, *Practical recommendations for gradient-based training of deep architectures*, in Neural Networks: Tricks of the Trade: Second Edition, G. Montavon, G. B. Orr, and K.-R. Müller, eds., vol. 7700 of Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2012, pp. 437–478.

[5] L. C. Evans, *Partial differential equations*, vol. 19 of Graduate Studies in Mathematics, American Mathematical Society, Providence, RI, second ed., 2010.

[6] D. Frerichs-Mihov, L. Henning, and V. John, *On loss functionals for physics-informed neural networks for steady-state convection-dominated convection-diffusion problems*, Communications on Applied Mathematics and Computation, (2024). accepted.

[7] D. Frerichs-Mihov, V. John, and M. Zainelabdeen, *On collocation points for physics-informed neural networks applied to convection-dominated convection-diffusion problems*, in Proceedings of ENUMATH 2023, Springer, 2024. accepted.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] T. J. R. Hughes, M. Mallet, and A. Mizukami, *A new finite element formulation for computational fluid dynamics. II. Beyond SUPG*, Comput. Methods Appl. Mech. Engrg., 54 (1986), pp. 341–355.

[10] V. John and P. Knobloch, *On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations. I. A review*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 2197–2215.

[11] V. John and P. Knobloch, *On algebraically stabilized schemes for convection-diffusion-reaction problems*, Numer. Math., 152 (2022), pp. 553–585.

[12] V. JOHN, J. M. MAUBACH, AND L. TOBISKA, *Nonconforming streamline-diffusion-finite-element-methods for convection-diffusion problems*, Numer. Math., 78 (1997), pp. 165–188.

[13] V. JOHN, T. MITKOVA, M. ROLAND, K. SUNDMACHER, L. TOBISKA, AND A. VOIGT, *Simulations of population balance systems with one internal coordinate using finite element methods*, Chemical Engineering Science, 64 (2009), pp. 733–741.

[14] G. E. KARNIADAKIS, I. G. KEVREKIDIS, L. LU, P. PERDIKARIS, S. WANG, AND L. YANG, *Physics-informed machine learning*, Nature Reviews Physics, 3 (2021), pp. 422–440.

[15] R. KHODAYI-MEHR AND M. ZAVLANOS, *VarNet: Variational Neural Networks for the Solution of Partial Differential Equations*, in Proceedings of the 2nd Conference on Learning for Dynamics and Control, Virtual, Online, 2020, PMLR, pp. 298–307.

[16] A. KRISHNAPRIYAN, A. GHOLAMI, S. ZHE, R. KIRBY, AND M. W. MAHONEY, *Characterizing possible failure modes in physics-informed neural networks*, in Advances in Neural Information Processing Systems, vol. 34, Virtual, Online, 2021, Curran Associates, Inc., pp. 26548–26560.

[17] J. NOVO AND E. TERRÉS, *Can neural networks learn finite elements?*, J. Comput. Appl. Math., 453 (2025), p. 8. Id/No 116168.

[18] RAY PROJECT CONTRIBUTORS, *Ray: A fast and simple framework for building and running distributed applications*. `https://docs.ray.io/en/latest/ray-overview/index.html`, 2023. Accessed: 2023-03-17.

[19] P. SHARMA, L. E., M. TINDALL, AND P. NITHIARASU, *Hyperparameter selection for physics-informed neural networks (pinns) – application to discontinuous heat conduction problems*, Numerical Heat Transfer Fundamentals, (2023).

[20] Y. WANG, C. XU, M. YANG, AND J. ZHANG, *Less emphasis on hard regions: curriculum learning of PINNs for singularly perturbed convection-diffusion-reaction problems*, East Asian J. Appl. Math., 14 (2024), pp. 104–123.

[21] Y. ZONG, Q. HE, AND A. M. TARTAKOVSKY, *Improved training of physics-informed neural networks for parabolic differential equations with sharply perturbed initial conditions*, Comput. Methods Appl. Mech. Engrg., 414 (2023), pp. Paper No. 116125, 28.