Weierstraß-Institut
für Angewandte Analysis und Stochastik

Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

On collocation points for physics-informed neural networks applied to convection-dominated convection-diffusion problems

Derk Frerichs-Mihov[1], Marwa Zainelabdeen[1], Volker John[1,2]

submitted: December 20, 2023

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

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

No. 3074
Berlin 2023

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

Key words and phrases. Convection-diffusion problems, convection-dominated regime, hard-constrained physics-informed neural networks, layer-adapted collocation points.

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

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

On collocation points for physics-informed neural networks applied to convection-dominated convection-diffusion problems

Derk Frerichs-Mihov, Marwa Zainelabdeen, Volker John

Abstract

In recent years physics-informed neural networks (PINNs) for approximating the solution to (initial-)boundary value problems gained a lot of interest. PINNs are trained to minimize several residuals of the problem in collocation points. In this work we tackle convection-dominated convection-diffusion problems, whose solutions usually possess layers, which are small regions where the solution has a steep gradient. Inspired by classical Shishkin meshes, we compare hard-constrained PINNs trained with layer-adapted collocation points with ones trained with equispaced and uniformly randomly chosen points. We observe that layer-adapted points work the best for a problem with an interior layer and the worst for a problem with boundary layers. For both problems at most acceptable solutions can be obtained with PINNs.

1 Introduction

A fundamental class of problems in computational fluid dynamics are convection-diffusion-reaction problems that model the distribution of a scalar quantity like energy or mass inside a flowing medium. In their strong form, we are seeking a smooth enough function u such that

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

where $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, is a domain with a polyhedral Lipschitz boundary $\partial\Omega$, $0 < \varepsilon \in \mathbb{R}$ is the diffusion coefficient, $\boldsymbol{b} \in [W^{1,\infty}(\Omega)]^d$ models the convection, $c \in L^\infty(\Omega)$ describes a reaction term, $f \in L^2(\Omega)$ denotes exterior forces, and $g \in H^{1/2}(\partial\Omega)$ the Dirichlet boundary conditions. It is well known that, if convection dominates diffusion, then the solution to these problems exhibit usually very thin regions where the solution has a steep gradient, so-called *layers*; see, e.g., [13, 7].

Physics-informed neural networks (PINNs) as presented in [2, 12] became popular for approximating the solution to (initial-)boundary value problems (IBVPs). The approximation is performed by minimizing some form of the residual of the governing equations and the corresponding boundary and possibly initial conditions in so-called *collocation points* [8].

In the past few years, it was recognized that the choice of the collocation points significantly influences the quality of the PINN approximation; see, e.g., [1, 11, 14, 5, 16]. In these references the collocation points are adaptively generated or redistributed during the training. Inspired by Shishkin meshes for classical numerical methods and in contrast to the former references, in this work, we use a fixed distribution of points that is chosen accordingly to the challenging regions of the solution, the layers. We compare the performance of PINNs trained with these points to PINNs trained with a random choice and an equispaced choice of the collocation points. The goal consists of further exploring the benefits but also the limitations of PINNs.

DOI 10.20347/WIAS.PREPRINT.3074 Berlin 2023

2 Physics-informed neural networks

PINNs are data-driven methods that approximate the solution of a given (I)BVP by minimizing a loss that incorporates the strong form of the residual of the differential equation, the boundary and possibly initial conditions [8]. In our numerical studies we deploy *hard-constrained PINNs* as introduced, e.g., in [9]. They use $u_{\mathcal{N}} \coloneqq \widetilde{g}_{\mathrm{D}} + h_{\mathrm{ind}}\widetilde{u}_{\mathcal{N}}$ as ansatz, where $h_{\mathrm{ind}} : \overline{\Omega} \to \mathbb{R}$ is an indicator function satisfying $h_{\mathrm{ind}}(\boldsymbol{x}) = 0$, if $\boldsymbol{x} \in \partial\Omega$, and $h_{\mathrm{ind}}(\boldsymbol{x}) > 0$ else, $\widetilde{g}_{\mathrm{D}} \in C(\overline{\Omega})$ is a continuous extension of the Dirichlet boundary condition g_{D} to $\overline{\Omega}$, and $\widetilde{u}_{\mathcal{N}}$ is a multilayer perceptron model (MLP). With this approach the Dirichlet conditions are satisfied exactly if $\widetilde{g}_{\mathrm{D}}|_{\partial\Omega} = g_{\mathrm{D}}$. MLPs are made of $L \in \mathbb{N}$ layers, which consist of $n_i \in \mathbb{N}$, $i = 1, 2, \ldots, L$, nodes each representing a real value. For the basic variant of PINNs it holds $n_1 \coloneqq \dim(\Omega)$ and $n_L \coloneqq \dim(\mathrm{Im}(u))$ since PINNs try to approximate an exact solution u. We collect the nodes of the ith layer in a vector $\widehat{\boldsymbol{y}}_i \in \mathbb{R}^{n_i}$. Then, for a given $\boldsymbol{x} \in \Omega$, we define $\widetilde{u}_{\mathcal{N}}(\boldsymbol{x}) \coloneqq \widehat{\boldsymbol{y}}_L$ as the value(s) of the last layer that is recursively computed by $\widehat{\boldsymbol{y}}_1 \coloneqq \boldsymbol{x}$, $\widehat{\boldsymbol{y}}_i \coloneqq \boldsymbol{\sigma}_i(W_i\widehat{\boldsymbol{y}}_{i-1} + \widehat{\boldsymbol{b}}_i)$, $i = 2, 3, \ldots, L$, where $\boldsymbol{\sigma}_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is a component-wise defined non-linear *activation function*, $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ is the *weight matrix* and $\widehat{\boldsymbol{b}}_i \in \mathbb{R}^{n_i}$ is the *bias vector*. Examples for activation functions are the hyperbolic tangent $\tanh$ and mish from [10] that are, for a given $x \in \mathbb{R}$, defined as

$$\tanh(x) \coloneqq \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \mathrm{mish}(x) \coloneqq x \cdot \tanh\left(\ln\left(1 + e^x\right)\right). \tag{2}$$

The entries of the weight matrices and the bias vectors are called *parameters*, which we denote by $\boldsymbol{p} \in \mathbb{R}^{d_{\boldsymbol{p}}}$, where $d_{\boldsymbol{p}} \coloneqq \sum_{i=2}^{L}(n_i + n_i \cdot n_{i-1})$. These are the degrees of freedom of the network that will be adapted to the problem. All remaining quantities to define a MLP are called *hyperparameters*. They are specified by the user and are fixed. The parameters are adapted during the so-called *training*. Let $u_{\mathcal{N};\boldsymbol{p}}$ be the ansatz whose underlying MLP $\widetilde{u}_{\mathcal{N}}$ has a certain value of the parameters vector $\boldsymbol{p}$. During the training we search for the optimal parameters $\boldsymbol{p}^* \in \mathbb{R}^{d_{\boldsymbol{p}}}$ such that

$$\boldsymbol{p}^* \in \operatorname*{arg\,min}_{\boldsymbol{p} \in \mathbb{R}^{d_{\boldsymbol{p}}}} \mathcal{L}(u_{\mathcal{N};\boldsymbol{p}}), \tag{3}$$

i.e., the parameters are adapted to minimize a certain *loss functional* $\mathcal{L} : \mathbb{R}^{d_{\boldsymbol{p}}} \to \mathbb{R}$ that depends on the problem. To solve the minimization problem (3) often a variant of stochastic gradient descent is applied; see also [6].

The idea of PINNs is to decode information about the (I)BVP in $\mathcal{L}$. Omitting the dependency of $u_{\mathcal{N}}$ on its MLP parameters $\boldsymbol{p}$, for the convection-diffusion problem (1) the *standard loss functional* is given by

$$\mathcal{L}^{\mathrm{st}}(u_{\mathcal{N}}) \coloneqq \frac{|\Omega|}{N_{\mathrm{I}}} \sum_{i=1}^{N_{\mathrm{I}}} \left(\mathrm{Res}(u_{\mathcal{N}})(\boldsymbol{x}_{i,\mathrm{I}})\right)^2, \tag{4}$$

where $\boldsymbol{x}_{i,\mathrm{I}} \in \Omega$, $i = 1, 2, \ldots, N_{\mathrm{I}}$, denote N_{I} *interior collocation points*, and

$$\mathrm{Res}(u_{\mathcal{N}}) \coloneqq -\varepsilon\Delta u_{\mathcal{N}} + \boldsymbol{b} \cdot \nabla u_{\mathcal{N}} + c u_{\mathcal{N}} - f.$$

In [3, 4] two more loss functionals are introduced that are especially designed for convection-diffusion problems; see [3, 4] for a derivation. The *limited residual* $\mathcal{L}^{\mathrm{lr}}_{t_0}$ loss functional and the *limited residual with crosswind* $\mathcal{L}^{\mathrm{lrcw}}_{t_0}$ loss are defined as

$$\mathcal{L}^{\mathrm{lr}}_{t_0}\left(u_{\mathcal{N}}\right) \coloneqq \sum_{i=1}^{N_{\mathrm{I}}} \xi\left(\frac{|\Omega|}{N_{\mathrm{I}}\, t_0}\left(\mathrm{Res}(u_{\mathcal{N}})\left(\boldsymbol{x}_{i,\mathrm{I}}\right)\right)^2\right), \tag{5}$$

$$\mathcal{L}_{t_0}^{\mathrm{lrcw}}\left(u_{\mathcal{N}}\right) \coloneqq \sum_{i=1}^{N_{\mathrm{I}}} \xi\left(\frac{|\Omega|}{N_{\mathrm{I}}\, t_0}\left(\mathrm{Res}(u_{\mathcal{N}})\left(\boldsymbol{x}_{i,\mathrm{I}}\right)\right)^2\right) + \frac{|\Omega|}{N_{\mathrm{I}}} \sum_{i=1}^{N_{\mathrm{I}}} \Phi\left(\left|\boldsymbol{b}^{\perp}(\boldsymbol{x}_{i,\mathrm{I}}) \cdot \nabla u_{\mathcal{N}}(\boldsymbol{x}_{i,\mathrm{I}})\right|\right), \tag{6}$$

depending on a positive user-chosen constant $t_0 \in \mathbb{R}$, and with, in two dimensions,

$$\boldsymbol{b}^{\perp}(\boldsymbol{x}) \coloneqq \begin{cases} \frac{(b_2(\boldsymbol{x}), -b_1(\boldsymbol{x}))^T}{\sqrt{b_1(x)^2 + b_2(x)^2}}, & \text{if } \boldsymbol{b}(\boldsymbol{x}) \neq \boldsymbol{0}, \\ \boldsymbol{0}, & \text{else}, \end{cases}$$

and $\xi : \mathbb{R} \to \mathbb{R}, \Phi : \mathbb{R} \to \mathbb{R}$ are given, for any $x \in \mathbb{R}$, by

$$\Phi(x) \coloneqq \begin{cases} \sqrt{x}, & \text{if } x \geq 1, \\ 0.5\left(5x^2 - 3x^3\right), & \text{else}, \end{cases}, \qquad \xi(x) \coloneqq \begin{cases} \frac{1}{2}x^4 - x^3 - \frac{1}{2}x^2 + 2x, & \text{if } x \leq 1, \\ 1, & \text{else}. \end{cases}$$

3 Set-up of numerical studies

In our numerical studies we train PINNs with three different sets of $N_{\mathrm{I}} \coloneqq 4,096$ collocation points: Randomly chosen points drawn from a uniform distribution, equispaced points and what we call *layer-adapted points* inspired by classical Shishkin meshes. For the latter, for each problem below we define corresponding layer regions and generate $N_{\mathrm{I}}/2$ equispaced points in these particular regions and $N_{\mathrm{I}}/2$ equispaced points in the remaining parts of the domain.

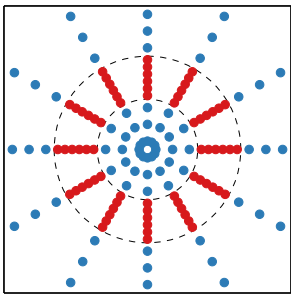
We deploy hard-constrained PINNs, for which we define $\widetilde{g}_{\mathrm{D}} \coloneqq 0$ and

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

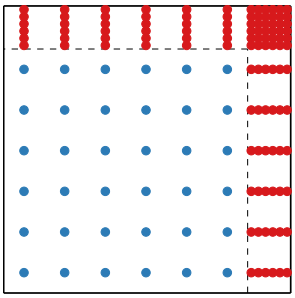
with $\kappa \coloneqq (10\varepsilon)^{-1}$, which is in correspondence with Problems 1 and 2 below. The PINNs are implemented in TensorFlow [15] and the Adam algorithm is used for optimization with TensorFlow's default values, except for the learning rate. During the training, the parameters are adapted to minimize either one of the losses given in equations (4) to (6) plus an L^2-weight decay regularization term $\lambda_{\mathrm{wd}} n_{\mathrm{bs}}/(2N_{\mathrm{I}}) \sum_j w_j^2$, where $\lambda_{\mathrm{wd}} \in \mathbb{R}$ is a positive constant, $n_{\mathrm{bs}} \coloneqq 32$, and w_j denotes the entries of all weight matrices. For $\mathcal{L}_{t_0}^{\mathrm{lr}}, \mathcal{L}_{t_0}^{\mathrm{lrcw}}$ we vary $t_0 \in \{10^1, 10^0, 10^{-1}, 10^{-2}\}$. We deploy networks with $n_1 \coloneqq 2$ and $n_L \coloneqq 1$ nodes in the first and the last layer, resp., and seven intermediate layers with 30 nodes each. While in the last layer the identity is deployed as activation function, in the intermediate layers either $\tanh$ or mish are applied. The weights are initialized using the Glorot initialization based on the seeds given in Table 1 and the biases are initially set to zero. We train in total 144 networks with the hyperparameters given in Table 1 with each loss functional and every set of collocation points for $7,500$ epochs. Afterwards, we compute a mean over the seeds, assess the L^2 error with respect to the exact solution and train for each loss and each set of collocation points the seven best networks until $100,000$ epochs are reached. Afterwards, again the mean over the seeds is taken and the L^2 error is measured. To compute the error, Ω is divided into $10,000$ equally-sized squares in which a Gauss–Legendre quadrature rule with ten points per coordinate direction is utilized.

Table 1: Set of hyperparameters leading to 144 different PINN architectures.

activation function	$\tanh, \mathrm{mish}$
learning rate	$0.01 \cdot 3^{-1}, 0.01 \cdot 3^{-2}, 0.01 \cdot 3^{-3}, 0.01 \cdot 3^{-4}, 0.01 \cdot 3^{-5}, 0.01 \cdot 3^{-6}$
initialization seed	$42, 43, 44$
weight decay λ_{wd}	$10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$



(a) Sketch of points for Problem 1.



(b) Sketch of points for Problem 2.

Figure 1: Sketch of layer-adapted points for Problems 1 and 2 on the unit square. Red points lie within the layer region(s) and blue points away from them. Dashed lines separate the layer regions from the non-layer regions.

4 Numerical studies with different sets of collocation points

We investigate two problems whose solutions show different kinds of layers.

Problem 1 (Circular internal layer)**.** Let $\Omega := (0,1)^2$, $\varepsilon := 10^{-8}$, $\boldsymbol{b} := (2,3)^T$, and $c := 2$. The other data of the problem are derived from the prescribed solution

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

where $r_0 := 0.25$ and $x_0 := y_0 := 0.5$. The solution possess an interior layer at the circle with radius r_0 around $(0.5, 0.5)$; cf. Fig. 6(a) in [4].

To generate layer-adapted points, we place $\sqrt{N_\mathrm{I}}$ points along $\sqrt{N_\mathrm{I}}$ rays. Let $\alpha_\ell := 2\pi\ell/\sqrt{N_\mathrm{I}}$ for $\ell \in \{0, 1, \ldots, \sqrt{N_\mathrm{I}} - 1\}$. The ℓth ray is given by $(0.5, 0.5) + t(\cos(\alpha_\ell), \sin(\alpha_\ell))$, where $0 \leq t \in \mathbb{R}$. Let $D := 100\sqrt{\varepsilon}$ be the layer width. Along each ray, we put $\sqrt{N_\mathrm{I}}/4$ equispaced points between $(0.5, 0.5)$ and the intersection point of the ray with a circle of radius $r_0 - D/2$ around $(0.5, 0.5)$, $\sqrt{N_\mathrm{I}}/2$ equispaced points between $r_0 - D/2$ and $r_0 + D/2$, and $\sqrt{N_\mathrm{I}}/4$ equispaced points between $r_0 + D/2$ and the intersection point of the ray with $\partial\Omega$; see Figure 1a for a visualization with 144 points and a layer width of 0.15.

From Table 2, we observe that layer-adapted points work worse for the standard loss but the best for seven out of eight of the novel functionals. The overall best error is achieved with $\mathcal{L}^{\mathrm{lr}}_{1.0}$ trained on the layer-adapted points, which is roughly 36.5% smaller than the best result obtained with the standard loss functional.

The best PINN solution that is obtained with $\mathcal{L}^{\mathrm{lr}}_{1.0}$ trained on the layer-adapted points is shown in Figure 2a. It can be seen that the hump structure can be recognized and that the solution has a steep gradient in the layer region. However, the solution's maximum is only roughly half as large as it should be. Furthermore, the solution possesses significant positive values away from the layer, where it should be close to zero. A reason might be that too many points lie in the layer region and hence the network focusses too much on this region during the training. On the other hand, with random and equispaced points it might be too difficult for the network to catch the steep gradient in the layer. Overall, due to the huge difference between the numerical and the exact solution we consider the MLP solution as non-acceptable.

Table 2: Minimal value of the error $\|u - u_{\mathcal{N}}\|_{L^2}$ after $100,000$ epochs of the best PINNs that approximate the solution to Problem 1 for all distributions. The smallest value in each row is marked with bold font. The blue number is the overall best value and the corresponding solution is plotted in Figure 2a.

loss functional	random	equispaced	layer-adapted
$\mathcal{L}^{\mathrm{st}}$	$3.885 \cdot 10^{-1}$	$\mathbf{3.488 \cdot 10^{-1}}$	$1.971 \cdot 10^{0}$
$\mathcal{L}^{\mathrm{lr}}_{0.01}$	$2.690 \cdot 10^{-1}$	$2.763 \cdot 10^{-1}$	$\mathbf{2.379 \cdot 10^{-1}}$
$\mathcal{L}^{\mathrm{lr}}_{0.1}$	$2.633 \cdot 10^{-1}$	$2.808 \cdot 10^{-1}$	$\mathbf{2.490 \cdot 10^{-1}}$
$\mathcal{L}^{\mathrm{lr}}_{1.0}$	$2.723 \cdot 10^{-1}$	$2.565 \cdot 10^{-1}$	$\mathbf{2.213 \cdot 10^{-1}}$
$\mathcal{L}^{\mathrm{lr}}_{10.0}$	$2.704 \cdot 10^{-1}$	$2.464 \cdot 10^{-1}$	$\mathbf{2.399 \cdot 10^{-1}}$
$\mathcal{L}^{\mathrm{lrcw}}_{0.01}$	$2.799 \cdot 10^{-1}$	$2.731 \cdot 10^{-1}$	$\mathbf{2.434 \cdot 10^{-1}}$
$\mathcal{L}^{\mathrm{lrcw}}_{0.1}$	$2.667 \cdot 10^{-1}$	$2.794 \cdot 10^{-1}$	$\mathbf{2.284 \cdot 10^{-1}}$
$\mathcal{L}^{\mathrm{lrcw}}_{1.0}$	$2.570 \cdot 10^{-1}$	$2.481 \cdot 10^{-1}$	$\mathbf{2.393 \cdot 10^{-1}}$
$\mathcal{L}^{\mathrm{lrcw}}_{10.0}$	$2.759 \cdot 10^{-1}$	$\mathbf{2.583 \cdot 10^{-1}}$	$2.716 \cdot 10^{-1}$

Problem 2 (Outflow layers)**.** Let $\Omega := (0,1)^2$, $\varepsilon := 10^{-8}$, $\boldsymbol{b} := (2,3)^T$, $c := 1$ and the other data of the problem are given by the prescribed solution

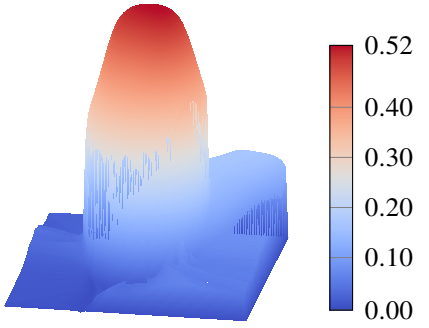
$$u(x,y) := xy^2 - y^2 \exp\left(\frac{2(x-1)}{\varepsilon}\right) - x \exp\left(\frac{3(y-1)}{\varepsilon}\right) + \exp\left(\frac{2(x-1)+3(y-1)}{\varepsilon}\right),$$

which is depicted in Fig. 6(b) in [4].

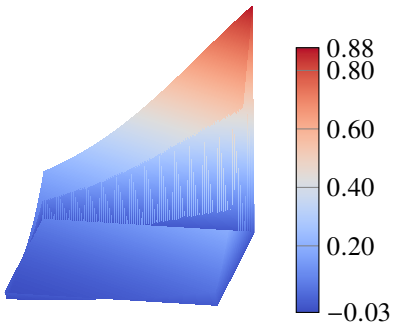
The layer-adapted mesh is given as follows: Let $D := 10\varepsilon$ be the layer width. We choose $\sqrt{N_{\mathrm{I}}}/2$ equispaced points in $[0, 1-D]$, and $\sqrt{N_{\mathrm{I}}}/2$ equispaced points in $[1-D, 1]$. Finally, a tensor product of these points is computed to get two-dimensional collocation points; see Figure 1b for a sketch with $N_{\mathrm{I}} = 144$ and $D = 0.15$.

From Table 3 we can see that layer-adapted points lead to bad results for all loss functionals. Random points work the best for five functionals, and for the remaining four functionals equispaced points lead to the best results. Overall the best result is obtained with $\mathcal{L}^{\mathrm{lrcw}}_{10.0}$ followed by $\mathcal{L}^{\mathrm{lr}}_{10.0}$ both trained on equispaced points. Their error is approximately half as large as the best result obtained with the standard functional.

The best PINN approximation trained with $\mathcal{L}^{\mathrm{lrcw}}_{10.0}$ on equispaced points is depicted in Figure 2b. It has roughly the shape it should have, but it's maximum (0.88) is significantly smaller than the maximum



(a) PINN solution to Problem 1.



(b) PINN solution to Problem 2.

Figure 2: PINN solutions that lead to the smallest errors, cf. Tables 2 and 3. The left one is obtained using $\mathcal{L}^{\mathrm{lr}}_{1.0}$ and layer-adapted points and the right one by training PINNs with the $\mathcal{L}^{\mathrm{lrcw}}_{10.0}$ on equispaced points.

Table 3: Minimal value of the error $\|u - u_{\mathcal{N}}\|_{L^2}$ after $100,000$ epochs of the best PINNs that approximate the solution to Problem 2 for all distributions. The smallest value in each row is marked with bold font. The blue number is the overall best value and the corresponding solution is plotted in Figure 2b.

loss functional	random	equispaced	layer-adapted
$\mathcal{L}^{\text{st}}$	$\mathbf{6.517 \cdot 10^{-2}}$	$6.951 \cdot 10^{-2}$	$2.582 \cdot 10^{-1}$
$\mathcal{L}^{\text{lr}}_{0.01}$	$\mathbf{1.196 \cdot 10^{-1}}$	$1.795 \cdot 10^{-1}$	$2.005 \cdot 10^{-1}$
$\mathcal{L}^{\text{lr}}_{0.1}$	$\mathbf{9.590 \cdot 10^{-2}}$	$3.122 \cdot 10^{-1}$	$1.547 \cdot 10^{-1}$
$\mathcal{L}^{\text{lr}}_{1.0}$	$1.242 \cdot 10^{-1}$	$\mathbf{7.605 \cdot 10^{-2}}$	$1.740 \cdot 10^{-1}$
$\mathcal{L}^{\text{lr}}_{10.0}$	$1.375 \cdot 10^{-1}$	$\mathbf{3.462 \cdot 10^{-2}}$	$1.908 \cdot 10^{-1}$
$\mathcal{L}^{\text{lrcw}}_{0.01}$	$\mathbf{1.675 \cdot 10^{-1}}$	$5.206 \cdot 10^{-1}$	$2.580 \cdot 10^{-1}$
$\mathcal{L}^{\text{lrcw}}_{0.1}$	$\mathbf{1.655 \cdot 10^{-1}}$	$1.960 \cdot 10^{-1}$	$2.540 \cdot 10^{-1}$
$\mathcal{L}^{\text{lrcw}}_{1.0}$	$1.781 \cdot 10^{-1}$	$\mathbf{6.694 \cdot 10^{-2}}$	$2.497 \cdot 10^{-1}$
$\mathcal{L}^{\text{lrcw}}_{10.0}$	$2.006 \cdot 10^{-1}$	$\mathbf{3.418 \cdot 10^{-2}}$	$2.570 \cdot 10^{-1}$

of the exact solution (≈ 1.0). Moreover, it shows negative values, which the exact solution does not have, and the network has difficulties to approximate the solution in the upper left corner of Ω. Overall, we think that this is a somewhat unsatisfactory but still an acceptable solution. The reason why layer-adapted points lead to worse solutions compared to the other choices of points might be again that too much focus is spend on the layer regions. The numerical layer of the discrete solution is mainly determined by the choice of the indicator function h_{ind}. With layer-adapted points and a non-optimal choice of h_{ind} huge deviations between the discrete and the exact solution can occur in the layers such that the networks have to spend too much training to counteract these errors. Moreover, when the numerical layer is completely determined by the indicator function, both random and equidistant points work comparably well since the solution in the rest of the domain is rather smooth and therefore comparably easy to approximate.

5 Conclusion

We numerically compared the approximation quality of PINNs for convection-dominated convection-diffusion problems trained with randomly drawn, equispaced and layer-adapted chosen collocation points that are inspired by Shishkin meshes. It was observed that for a problem with an interior layer the layer-adapted points worked the best, but the solution itself was quite inaccurate. For the problem with boundary layers, layer-adapted points led to the worst results. An explanation for the behavior might be that too much effort is put onto the layer region which makes it difficult for the network to approximate the solution reasonably well in the whole domain. For problems with boundary layers we suggest to control the layers by the indicator function of hard-constrained PINNs and remove any points from the layer regions. In contrast, it seems to be useful to have points inside interior layers, but a suitable proportion of these points compared to non-layer points is an open question.

References

[1] Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling, 2022.

[2] Gamini Dissanayake and Nhan Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Comm. Numer. Methods Engrg.*, 10(3):195–201, 1994.

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

[4] Derk Frerichs-Mihov, Linus Henning, and Volker John. On loss functionals for physics-informed neural networks for convection-dominated convection-diffusion problems, 2023. WIAS preprint.

[5] John M. Hanna, José V. Aguado, Sebastien Comas-Cardona, Ramzi Askri, and Domenico Borzacchiello. Residual-based adaptivity for two-phase flow simulation in porous media using Physics-informed Neural Networks. *Comput. Methods Appl. Mech. Engrg.*, 396:115100, 2022.

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

[7] Volker John, Petr Knobloch, and Julia Novo. Finite elements for scalar convection-dominated equations and incompressible flow problems: A never ending story? *Comput. Vis. Sci.*, 19(5-6):47–63, 2018.

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

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

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

[11] Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, 2021.

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

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

[14] Shashank Subramanian, Robert M. Kirby, Michael W. Mahoney, and Amir Gholami. Adaptive Self-Supervision Algorithms for Physics-Informed Neural Networks. In *ECAI 2023*, volume 372 of *Frontiers in AI and Appl.*, pages 2234–2241, Kraków, Poland, 2023. IOS Press.

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

[16] Colby L. Wight & Jia Zhao. Solving Allen-Cahn and Cahn-Hilliard Equations using the Adaptive Physics Informed Neural Networks. *Communications in Computational Physics*, 29(3):930–954, 2021.