East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.2023-062.170523

Vol. **14**, No. 1, pp. 104-123
February 2024

Less Emphasis on Hard Regions: Curriculum Learning of PINNs for Singularly Perturbed Convection-Diffusion-Reaction Problems

Yufeng Wang[1], Cong Xu[2], Min Yang[1,*] and Jin Zhang[3]

[1]*School of Mathematics and Information Sciences, Yantai University, Yantai, China.*
[2]*School of Computer Science and Technology, East China Normal University, Shanghai, China.*
[3]*Department of Mathematics, Shandong Normal University, Jinan, China.*

Received 18 February 2023; Accepted (in revised version) 17 May 2023.

Abstract. Although physics-informed neural networks (PINNs) have been successfully applied in a wide variety of science and engineering fields, they can fail to accurately predict the underlying solution in slightly challenging convection-diffusion-reaction problems. In this paper, we investigate the reason of this failure from a domain distribution perspective, and identify that learning multi-scale fields simultaneously makes the network unable to advance its training and easily get stuck in poor local minima. We show that the widespread experience of sampling more collocation points in high-loss regions hardly help optimize and may even worsen the results. These findings motivate the development of a novel curriculum learning method that encourages neural networks to prioritize learning on easier non-layer regions while downplaying learning on harder regions. The proposed method helps PINNs automatically adjust the learning emphasis and thereby facilitates the optimization procedure. Numerical results on typical benchmark equations show that the proposed curriculum learning approach mitigates the failure modes of PINNs and can produce accurate results for very sharp boundary and interior layers. Our work reveals that for equations whose solutions have large scale differences, paying less attention to high-loss regions can be an effective strategy for learning them accurately.

AMS subject classifications: 35Q68, 68T07, 68W25

Key words: Physics-informed neural network, convection-diffusion-reaction, boundary layer, interior layer, curriculum learning.

*Corresponding author. *Email addresses:* zytuyufengwang@163.com (Y. Wang), congxueric@gmail.com (C. Xu), yang@ytu.edu.cn (M. Yang), jinzhangalex@hotmail.com (J. Zhang)

1. Introduction

Convection-diffusion-reaction problems appear in the modeling of various modern complicated processes, such as fluid flow at high Reynolds numbers [16], drift diffusion in semiconductor device modeling [29], and chemical reactor theory [26]. Very often the size of diffusion is characterized by a parameter ϵ, which could be smaller by several orders of magnitude compared to the size of convection and/or reaction, resulting in narrow boundary or interior layers in which the solution changes extremely rapidly [31]. Classical numerical methods use layer-adapted meshes or introduce carefully designed artificial stability terms to solve these challenging problems [2, 4, 33, 37, 38].

In recent years, there has been a surge of interest in applying neural networks in traditional scientific modeling — e.g. partial differential equations, which yields the so-called physics-informed neural networks [5, 10, 11, 14, 17, 18, 21, 23, 30, 34, 35]. The main idea of PINNs is to include physical domain knowledge as soft constraints in the empirical loss function and then use existing machine learning methodologies such as stochastic optimization, to train the model. As an interesting alternative to traditional numerical solvers, PINN has the advantage of flexibility in dealing with high-dimensional PDEs in complicated geometry and easy incorporation of available data information. Moreover, well-trained PINNs can have good generalization ability and can quickly predict solutions outside the computational area.

However, as reflected in some recent studies on the failure modes of PINNs [1, 6–8, 21], it has been found that PINNs can fail to converge to the correct solution even for relatively simple convection-diffusion problems. Approaches to improve the accuracy of PINNs in solving convection-diffusion problems can be broadly classified into two categories. The first category borrows theories and concepts from conventional numerical methods. For example, Mojgani *et al*. [28] rewrote the original equation into a Lagrangian form on the characteristic curves and then applied a two-branch neural network to solve the reformulated form. However, the approach is only applicable to time-dependent problems and not to steady-state equations. Recently, inspired by the theory of singular perturbation and asymptotic expansions, Arzani *et al*. [1] used separate neural networks to learn the different levels on the inner and outer layer regions, respectively. The second category emphasizes machine learning techniques, such as the design of loss functions, sample selection, and learning strategies. He *et al*. [15] used a weighted sum of residual losses and showed that in order to obtain an accurate solution of the advection-dispersion equation, the weights of the initial and boundary conditions should be larger than the PDE residuals. Daw *et al*. [6] proposed an evolutionary sampling algorithm in which the collocation points evolve gradually with training to prioritize high-loss regions while maintaining a background distribution of uniformly sampled points. Krishnapriyan *et al*. [21] argued that the PDE-based soft constraints make the loss landscapes difficult to optimize, and proposed a curriculum approach that sets the PINN loss term starting with a simple equation regularization and progressively become more complex as the network gets trained, which suffers from complex training scheme and very long training phase when solving strong singular perturbation problems.

The existing studies mainly consider relatively simple cases where the viscosity/diffusivity is about a scale of 10^{-4}. Singularly perturbed problems containing extremely sharp layers (strong vanishing viscosity/diffusivity limit) remain an urgent target for PINNs. This paper aims to unravel the failure modes of PINNs from some new perspectives and to further advance the approximation performance of PINNs. We show that simultaneously learning multi-scale solutions in layer and non-layer regions makes the network difficult to advance its training and easily get stuck in poor local minima. We demonstrate that in such a case, prioritizing layer regions (sampling more collocation points in high-loss regions) can make the training more difficult and worsen the performance. This surprising finding is contrary to the majority of existing studies on PINNs. While most previous studies have emphasized high-loss regions, our investigation indicates that for problems containing samples with extreme scale differences, it seems not a good idea to emphasize high-loss regions. We argue that this is because collocation points from layer regions are significantly more challenging to learn than those from non-layer regions. To alleviate the learning difficulties, we propose a novel curriculum learning approach that can automatically adjust the sample weights to emphasize easier non-layer regions, thereby improving the approximation accuracy of the network for strongly singular perturbation problems. We empirically demonstrate the efficiency of the proposed approach in a variety of typical convection-diffusion-reaction problems. We show that the proposed curriculum learning algorithm can mitigate the failure modes of vanilla PINNs and well capture the sharp boundary or interior layers even in the cases of very small diffusivity ($\epsilon = 10^{-9}$). Our approach successfully learns solutions containing very sharp layers, using only one neural network, without learning any intermediate solutions. More importantly, we provide a new perspective to understand the failure modes of PINNs and reveal that for equations whose solutions have large scale differences, paying less attention to high-loss regions could be a feasible strategy for learning them accurately. The source code built on PyTorch is available at `https://github.com/WYu-Feng/CLPINN` to enable other researchers to reproduce and extend the results.

The remainder of the paper is organized as follows. Section 2 gives the problem under study and introduces the basic notation of PINNs. A toy example is used in Section 3 to explore the possible reason for the failure mode of PINNs in solving singularly perturbed equations. In Section 4, we design a curriculum learning approach to improve the performance of PINNs. Section 5 gives comprehensive experimental results to demonstrate the efficiency of the proposed method. Finally, the conclusion is drawn in Section 6.

2. Problem Setup

Consider the following singularly perturbed equation:

$$\mathscr{L}u := \epsilon\mathscr{L}_2 u + \mathscr{L}_1 u + \mathscr{L}_0 u = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega,$$

where Ω is a physical domain in $\mathbb{R}^d$, $\mathscr{L}_k$ represents a differential operator of order k, $k = 0, 1, 2$, $f(\boldsymbol{x})$ denotes the source term, and the diffusion coefficient satisfies $0 < \epsilon \leq 1$.

Further we assume that the solution $u(\boldsymbol{x})$ satisfies the following boundary condition:

$$\mathscr{B}u = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega,$$

where $\mathscr{B}$ is a well-defined differential operator determining the condition on the admissible boundary $\partial\Omega$. When the diffusion coefficient ϵ is very small, the latent solution of the equation changes rapidly within some thin layers, posing a great challenge to the numerical simulation [4,27].

For PINNs, the solution $u(\boldsymbol{x})$ is approximated by a neural network $u_\theta(\boldsymbol{x})$, where θ denotes the parameters of the network. Let

$$L_{phys}(\theta) = \frac{1}{N}\sum_{i=1}^{N} r_{phys}^2(\boldsymbol{x}_i;\theta) = \frac{1}{N}\sum_{i=1}^{N}\big[\mathscr{L}u_\theta(\boldsymbol{x}_i) - f(\boldsymbol{x}_i)\big]^2 \tag{2.1}$$

be the mean-squared physical residual loss of N training sample points in Ω, and

$$L_{bc}(\theta) = \frac{1}{M}\sum_{i=1}^{M} r_{bc}^2(\boldsymbol{x}_i;\theta) = \frac{1}{M}\sum_{i=1}^{M}\big[\mathscr{B}u_\theta(\boldsymbol{x}_i) - g(\boldsymbol{x}_i)\big]^2 \tag{2.2}$$

be the mean-squared boundary loss of M training sample points on $\partial\Omega$. All the samples constitute a training set X_{train}.

The neural network approximation $u_\theta(\boldsymbol{x})$ can be determined by solving the following optimization objective:

$$\min_\theta L_{phys}(\theta) + \lambda L_{bc}(\theta), \tag{2.3}$$

where λ is a hyperparameter to balance the weights of the two loss terms.

Although PINNs have been successfully applied in solving many types of differential equations, their performance for relatively simple convection-diffusion equations are far from satisfactory. In the next section, we are to analyze the dilemma encountered by PINNs.

3. Analysis of Failure Mode

Consider the following one-dimensional convection-diffusion problem:

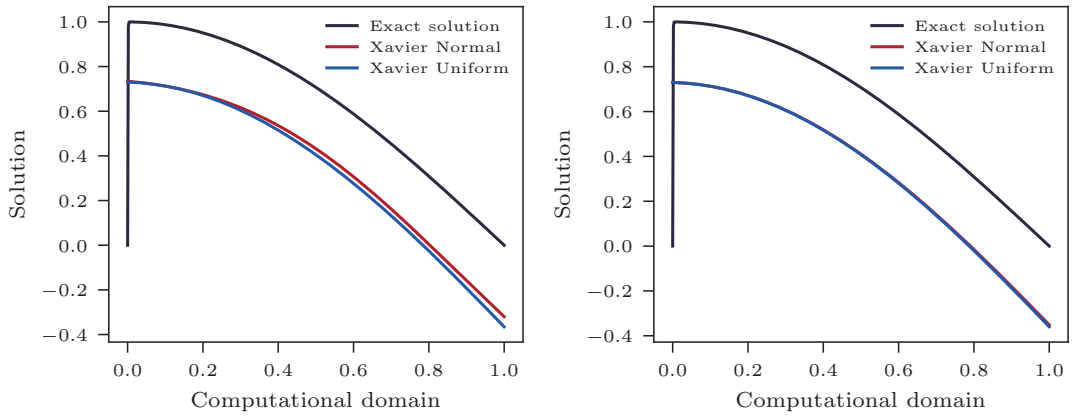
$$\begin{aligned} -\epsilon u_{xx} + (x-2)u_x &= f(x), \quad x \in (0,1), \\ u(0) = u(1) &= 0, \end{aligned} \tag{3.1}$$

where the diffusion coefficient ϵ is set as 10^{-3}, and the source term $f(x)$ is determined by the exact solution $u(x) = \cos(\pi x/2)(1-\exp(2x/\epsilon))$. This problem has a boundary layer at $x = 0$.

Consider a four-layer fully connected neural network $u_\theta(x)$, where each intermediate layer has 20 neurons and Tanh is used as the activation function. The training set X_{train} consists of 2500 points uniformly sampled from the domain $(0,1)$.

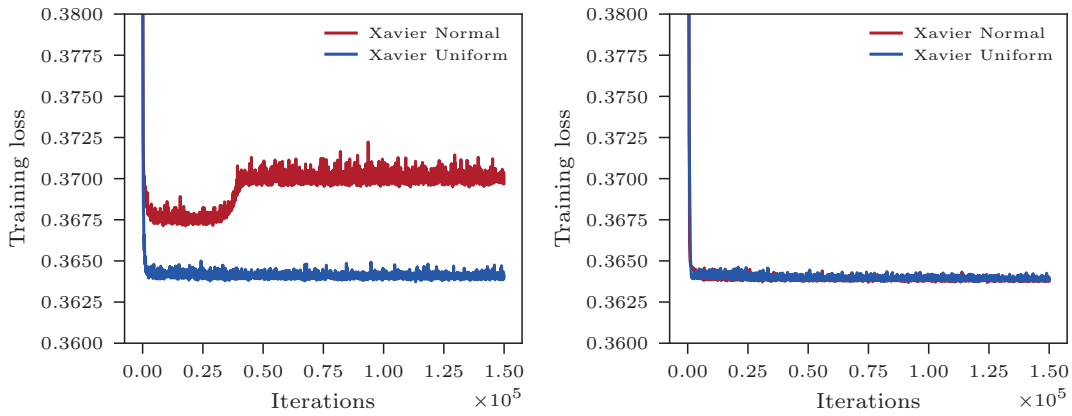
Initialization and optimization. The network parameters are initialized by normal Xavier or uniform Xavier methods [9]. Two mainstream optimizers, stochastic gradient descent (SGD) and Adam [20], are utilized to solve the optimization objective (2.3), where the balance parameter λ is set to 1.

It can be observed from Fig. 1 that the prediction $u_\theta(x)$ has very large errors throughout the computational domain, regardless of the initial or training methods used. When we further plot the corresponding training loss curves (Fig. 2), it is clear that the training loss of PINN fails to converge even after very long iterations. In particular, it can be seen that the training losses in the layer regions are much higher than those in the non-layer regions (Fig. 3).



(a) SGD (b) Adam

Figure 1: Predictions of PINN under various parameter initializations using SGD and Adam optimizers, respectively.



(a) SGD (b) Adam

Figure 2: Training loss curves of PINN under various parameter initializations using SGD and Adam optimizations, respectively.

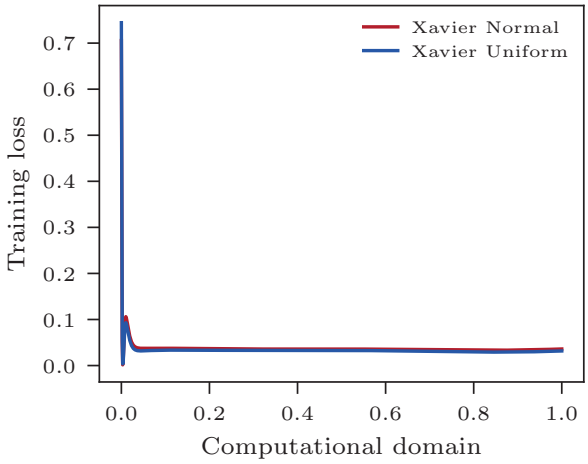


Figure 3: Training loss distribution of Eq. (3.1) under different initializations using the Adam optimizer.

Emphasizing high-loss layer regions? Note that there exists a widely accepted consensus that the performance of PINNs can be improved by sampling more collocation points in high-loss regions. We tried such a strategy, but unfortunately it can be found from Fig. 4 that instead of improving the approximations, the dense sampling in the high-loss layer region may lead to worse results.

The above experiments show that for singular perturbation equations, common PINNs cannot solve them well even with dense sampling in the high-loss layer regions. Such paradoxical phenomenon leads to the natural question of what is the cause of this undesirable performance.

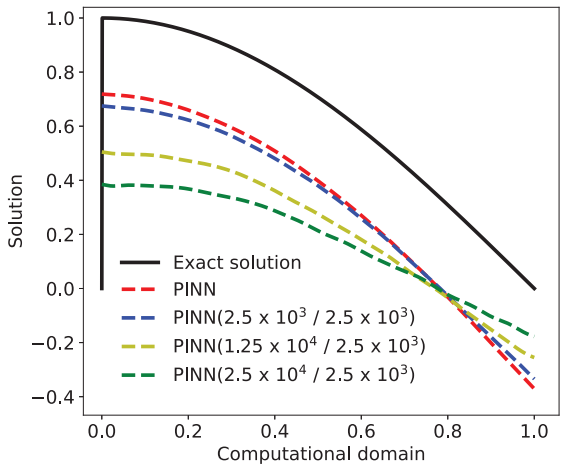
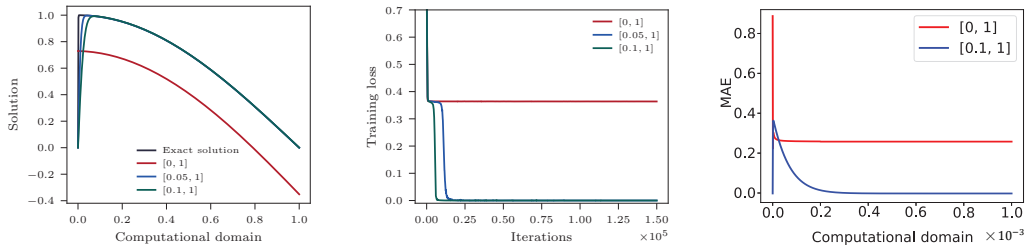


Figure 4: Predictions of PINN using a dense sampling in the layer domain, where 2500 points are sampled in the non-layer domain $(0.1, 1)$, and 2500, 12500, 25000 points are sampled in $(0, 0.1)$, respectively. For standard PINN, we apply a random sampling in $(0, 1)$.

Less emphasis on layer regions. We notice that compared with ordinary equations, the latent solutions of singular perturbation equations exhibit sharp scale variations in different regions. In the narrow layer region the solution transits very rapidly, while in the wide non-layer region the solution varies more flatly and slowly. We argue that such large scale differences make PINNs difficult to balance the learning of collocation points from the layer and non-layer regions. The final loss distribution in Fig. 3 shows the training losses for samples close to the boundary layer are much larger than those in the non-layer domain, which implies that the sharp layer domain may be too difficult for PINNs to learn.

In order to reduce the learning difficulty of PINN, we put forward the following experiment. We only select samples from non-layer regions to build the training composed of samples in $(a,1)$, where a is set to 0.05 and 0.1, respectively. It is surprising to observe from Fig. 5 that such a brutal discarding of layer samples can result in an obvious improvement for the training and prediction of PINN. Thereby, the above attempt inspires us that less emphasis on difficult layer regions may help to raise the performance of PINNs in solving singularly perturbed problems.

Of course, naively rejecting samples from the layer regions will inevitably result in the loss of important physical information, thus cannot guarantee the high accuracy of the prediction. Moreover, the location of the layers is usually not known in practice. Therefore, in the next section, we are to present a curriculum learning algorithm that dynamically estimates the location of layers and adaptively adjusts the importance of the samples close to the layers.



(a) (b) (c)

Figure 5: (a) Predictions of PINN after a rejection of the layer samples. (b) Training loss curves. (c) Absolute errors in the layer region $(0, 1e-3)$. Here $[a,1]$ denotes the result after ignoring the samples from the difficult region $(0,a)$.

4. The Proposed Curriculum Learning

So far, we have demonstrated that the failure mode of PINN is due to large discrepancy in sample difficulties between layer and non-layer regions. In this section, we are to provide a curriculum learning algorithm that encourages the network to prioritize learning easier non-layer regions. The complete process is illustrated in Fig. 6.

4.1. Surrogate for layer location

Since the learning difficulty in layer and non-layer regions differs significantly, the first

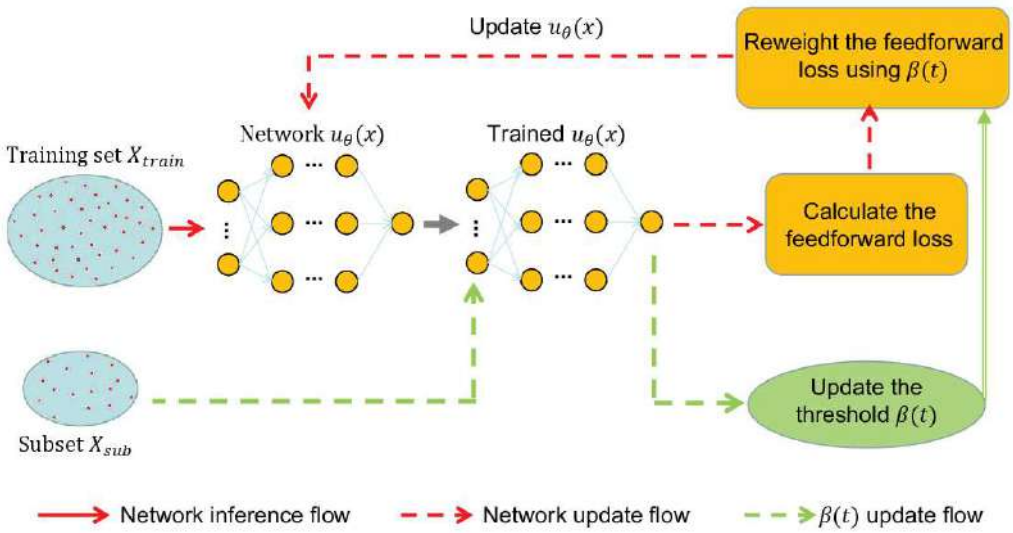


Figure 6: The proposed learning framework. A sub-training set X_{sub} is employed to update the threshold $\beta(t)$ at the t-th iteration step. Then, the threshold $\beta(t)$ is used to dynamically re-weight each training sample, especially lightening the importance of the samples close to the layer regions.

key step is to estimate the location of the layers. According to Fig. 3, it can be found that the layer region usually corresponds to a larger training loss. Therefore, we can take the feedforward training loss as a proxy to estimate the location of the layers. A larger loss implies that the corresponding sample is closer to the layer.

4.2. Importance reweighting

Recall that the optimization objective (2.1) is the average of the squared losses of all samples

$$L_{phys}(\theta) = \frac{1}{N}\sum_{i=1}^{N} r_{phys}^2(\boldsymbol{x}_i;\theta),$$

which means that samples from different regions are of equal importance for learning.

In order to make PINNs place less emphasis on the samples from the layer regions, we modify the optimization objective as follows:

$$L_{phys}(\theta) = \frac{1}{\sum_{i=1}^{N} w(\boldsymbol{x}_i)}\sum_{i=1}^{N} w(\boldsymbol{x}_i) r_{phys}^2(\boldsymbol{x}_i;\theta),$$

where $w(\boldsymbol{x}_i)$ represents the importance of the sample. The closer the sample is to the layer, the less weight it has.

As discussed in Section 4.1, we do not known the exact locations of the layers and shall estimate them using the training losses that vary dynamically with iterations. Therefore,

the weight of each sample should also be dynamically adjusted. To this end, we define

$$L_{phys}(\theta) = \frac{1}{\sum_{i=1}^{N} w(t, \boldsymbol{x}_i)} \sum_{i=1}^{N} w(t, \boldsymbol{x}_i) r_{phys}^2(\boldsymbol{x}_i; \theta), \tag{4.1}$$

where t denotes the iteration step. The sample weights in (4.1) can be determined by

$$w(t, \boldsymbol{x}_i) = \begin{cases} 1, & \text{if} \quad r_{phys}^2(\boldsymbol{x}_i) \le \beta(t), \\ \dfrac{\beta(t)}{r_{phys}^2(\boldsymbol{x}_i)}, & \text{if} \quad r_{phys}^2(\boldsymbol{x}_i) > \beta(t), \end{cases} \tag{4.2}$$

where $\beta(t)$ is a loss threshold to be updated adaptively with iterations.

Intuitively, the formula (4.2) indicates that if the training loss of a sample is greater than $\beta(t)$, which implies that the collocation point is close to the layer, then we give this sample a weight $\beta(t)/r_{phys}^2(\boldsymbol{x})$, which is less than 1. The larger the loss, the closer the sample is to the layer, and the smaller the corresponding weight. In this way, we not only emphasize the learning of easy non-layer region samples, but also maintain the necessary physical information of the high-loss layer regions.

4.3. Calculate the threshold by a sub-training set

Since the training loss of a desirable network model will gradually descend with iterations, then the threshold $\beta(t)$ cannot be predetermined, but should be updated adaptively with the training process. To save computational cost, this section will present a method to compute $\beta(t)$ based on the sub-training set.

First, notice that even with the same network structure, the amplitude of the training loss can vary greatly from equation to equation. It is hard to select a threshold that applies to all equations directly through training losses. However, we find that for singular perturbation equations, the gradient of the loss curve is extremely steep around the layer (Fig. 3). Therefore, we argue that indirectly determining the threshold $\beta(t)$ by the gradient of the loss curve can make the method have a better versatility.

More specifically, let X_{sub} be a randomly selected subset from the training set, which is fixed during the training process. Let G be a predefined hyperparameter. After the t-th iteration, for each $\boldsymbol{x} \in X_{sub}$, if $|\nabla_{\boldsymbol{x}} r_{phys}^2(\boldsymbol{x}; \theta)| < G$, which means that the collocation point is on the outside of the layer region, then we store its training loss in a memory bank M. Finally, the maximum loss value in M is chosen as the threshold $\beta(t)$. In this way, $\beta(t)$ can be considered as an upper bound of the training losses of all non-layer samples. If the loss of a sample exceeds this threshold, the sample is considered to be close to layer regions and its weight needs to be reduced in the next training iterations.

Remark 4.1. Since the training loss usually does not change quickly, especially in later training periods, to save computational cost, we employ an interval update strategy, where the threshold $\beta(t)$ is updated every K iterations. In our experiments, K is set as 50.

Remark 4.2. The proposed approach falls under the category of curriculum learning [3,13], which mimics human learning and suggests neural networks to prioritize learning easier tasks. Our algorithm incorporates the properties of singularly perturbed equations and therefore is distinctly different from those existing curriculum learning algorithms, which are mainly developed for computer vision [12,32] and natural language processing [22,36].

The pseudo-code of the proposed approach is summarized in Algorithm 4.1.

Algorithm 4.1 Pseudo-Code of Curriculum Learning for Singularly Perturbed Problems

Require: Training set X_{train}, subset $X_{sub} \subset X_{train}$, predefined constant G, balance parameter λ, and update frequency K.

```
Initialize the iteration step t = 0.
for each training step t do
    if t is divisible by K then
        Clean the memory bank M.
        for each collocation point x ∈ X_sub do
            if |∇_x r²_phys(x; θ)| < G then
                Store the corresponding training loss into M.
        Update the threshold β(t) by the maximum loss in the bank M.
    Update the sample weights by (4.2).
    Update the network parameters based on the loss functions (2.2) and (4.1).
    t++.
```

5. Experiments

5.1. Experimental setup

To evaluate the performance of the proposed method, six benchmark convection-diffusion-reaction equations, including one 1-dimensional example, three 2-dimensional examples, and one 3-dimensional example, are considered. In addition to the conventional PINN, we also compare our approach with the residual-based adaptive refinement method (RAR) [23], which updates the training dataset by refining collocation points with the largest residual values. We implement our approach with PyTorch and run the experiments on an Intel Xeon CPU E5-2650 v3 platform with 14GB ROM and an RTX 3060 GPU. The balance weight λ in the optimization objective (2.3) is set to 1, the update frequency $K = 50$, and the constant G is set to 10 in the one-dimensional case and to 50 in the multidimensional cases. The subset X_{sub} is $1/5$ of the size of the entire training set.

We utilize six fully connected feedforward neural networks to solve different equations, respectively. All networks employ the Tanh function as the activation unit. The training process is performed using the Adam optimizer [20]. The specific network structures as well as the training parameters are specified in Table 1.

For one dimensional equations, we employ a uniform sampling to construct a training set. For the multi-dimensional problems, to ensure that there are a number of training

Table 1: Structures of neural networks and learning parameters.

Equation	Network depth	Network width	Optimizer	Batch size	Learning rate	Iterations
5.1	3	20	Adam	50	0.001	1.5×10^5
5.2	5	20	Adam	200	0.01	1.5×10^6
5.3	3	20	Adam	200	0.01	1×10^6
5.4	3	20	Adam	200	0.01	1×10^6
5.5	3	20	Adam	200	0.005	1.5×10^6
5.6	5	20	Adam	500	0.01	1×10^6

Table 2: Number of training points for different equations.

Equation	Interior samples	Boundary samples
5.1	2.5×10^3	2
5.2–5.5	2×10^4	4×10^2
5.6	3×10^5	6×10^4

points belonging to the layer regions, we adopt a non-uniform sampling. Specifically, we first randomly sample half of the training points, and then add 0.05% of training samples around the points whose feedforward losses exceed the threshold $\beta(t)$ at every 50 iterations, until the training set reaches the predefined size. The size of the training set for each equation is listed in Table 2.

If the exact solution is known, we quantify the performance of the prediction by using the normalized root-mean-squared error (NRMSE)

$$\text{NRMSE}=\frac{\sqrt{\sum_{i=1}^{n}|u_\theta(x_i)-u(x_i)|^2}}{\sqrt{\sum_{i=1}^{n}|u(x_i)|^2}},$$

where $u_\theta(x)$ and $u(x)$ represent the predicted and the exact solution, respectively, and n denotes the number of uniformly sampled test points, which is set to 1000 for one-dimensional equation, and 5000 for multi-dimensional equations.

5.2. One-dimensional convection-diffusion equation

Consider the following two point problem:

$$\begin{aligned}&-\epsilon u_{xx}+(x-2)u_x=f(x),\quad x\in(0,1),\\&u(0)=u(1)=0,\end{aligned}\tag{5.1}$$

where the source term $f(x)$ is chosen such that the exact solution

$$u(x)=\cos\left(\frac{\pi x}{2}\right)\left(1-\exp\left(-\frac{2x}{\epsilon}\right)\right).$$

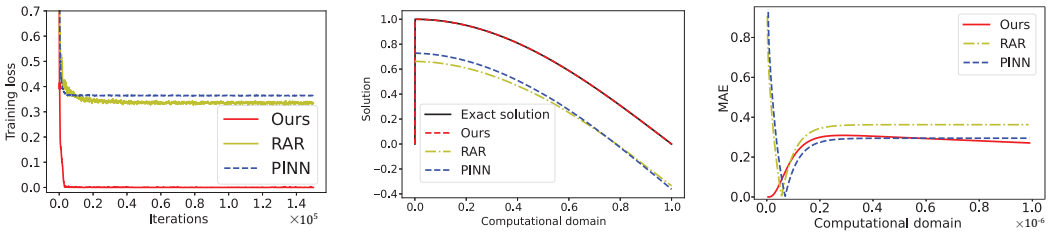
The solution of (5.1) is characterized by a boundary layer at $x=0$.

We first plot the training loss curves of our approach, RAR and the common PINN in the case of $\epsilon = 1e-6$. As demonstrated in Fig. 7(a), our method descends much faster than PINN and RAR in the early stage of training, and the corresponding training loss approaches 0 after about 4000 iterations. In contrast, PINN and RAR fail to converge even after a long period of iterations. It can be further observed from Fig. 7(b) that the prediction of our approach captures the boundary layer well and fits the exact solution much better over the entire computational domain, while the results of PINN and RAR differ significantly. Moreover, it is obvious from Fig. 7(c) that our method is superior to the other two methods in the difficult region, especially near the boundary layer point $x = 0$.

Further, we compare the normalized root-mean-squared errors of the two methods with more diffusion coefficients. It is obvious from Table 3 that for non-singularly perturbed case ($\epsilon = 1$), all methods can produce satisfactory results of the same order of accuracy. However, for singularly-perturbed cases, the errors of our method are 3 orders of magnitude lower than those of PINN and RAR.

Table 3: Normalized root-mean-squared errors between predicted and exact solutions of Eq. (5.1) for various diffusion coefficients.

Diffusion coefficient	Ours	RAR	PINN
$\epsilon = 1$	1.81×10^{-4}	1.77×10^{-4}	1.83×10^{-4}
$\epsilon = 1e-3$	1.41×10^{-4}	4.89×10^{-1}	4.45×10^{-1}
$\epsilon = 1e-6$	1.44×10^{-4}	4.92×10^{-1}	4.54×10^{-1}
$\epsilon = 1e-9$	1.47×10^{-4}	4.98×10^{-1}	4.47×10^{-1}



(a) Training loss curves (b) Predictions (c) Absolute errors in $(0, \epsilon)$

Figure 7: Comparison of RAR, PINN, and our approach for 1D equation (5.1), $\epsilon = 1e-6$.

5.3. Two-dimensional convection-diffusion-reaction equation with boundary layers

Consider the following two-dimensional problem [37]:

$$\begin{aligned} -\epsilon\Delta u + (3 - x_1 - x_2)u_{x_1} + 1.5u &= f, \quad && \boldsymbol{x} \in \Omega = (0,1)^2, \\ u &= 0, && \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{5.2}$$

where $f(x)$ is chosen such that the exact solution

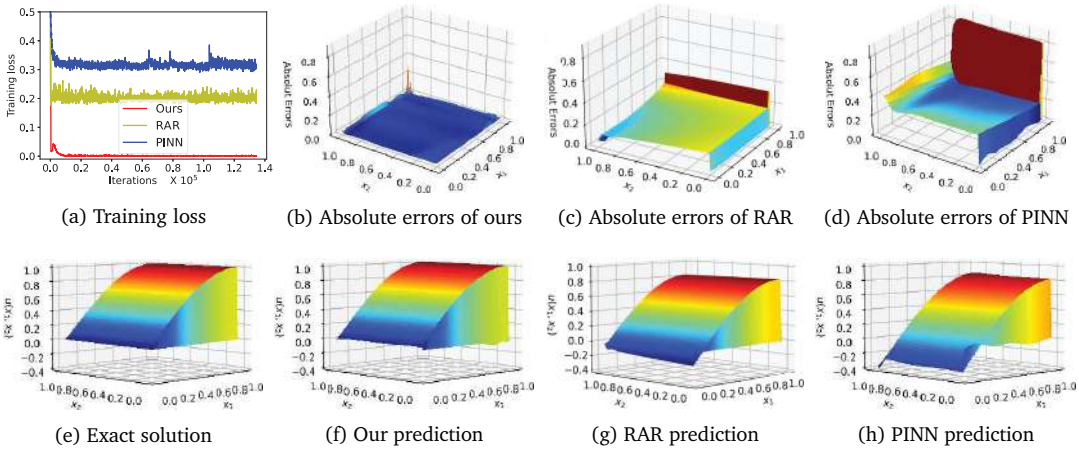
$$u = \left(\sin \frac{\pi x_1}{2} - \frac{e^{-(1-x_1)/\epsilon} - e^{-1/\epsilon}}{1 - e^{-1/\epsilon}} \right) \frac{(1 - e^{-x_2/\sqrt{\epsilon}})(1 - e^{-(1-x_2)/\sqrt{\epsilon}})}{1 - e^{-1/\sqrt{\epsilon}}}.$$

The solution of (5.2) is characterized by the presence of three boundary layers, one at $x_1 = 1$, and two at $x_2 = 0$ and $x_2 = 1$.

It can been observed from Fig. 8 and Table 4 that our method still performs well in capturing the behavior of the layers. The predictions only have a little oscillation at the boundary layer location. In contrast, the RAR shows certain deviations from the truth in layer regions, and the PINN deviates even more.

Table 4: Normalized root-mean-squared errors between predicted and exact solutions of Eq. (5.2) for various diffusion coefficients.

Diffusion coefficient	Ours	RAR	PINN
$\epsilon = 1e-3$	4.67×10^{-4}	8.58×10^{-3}	3.37×10^{-2}
$\epsilon = 1e-6$	5.38×10^{-4}	2.42×10^{-2}	5.31×10^{-1}
$\epsilon = 1e-9$	5.35×10^{-4}	2.87×10^{-2}	5.30×10^{-1}



(a) Training loss (b) Absolute errors of ours (c) Absolute errors of RAR (d) Absolute errors of PINN

(e) Exact solution (f) Our prediction (g) RAR prediction (h) PINN prediction

Figure 8: Comparison of RAR, PINN, and our approach for Eq. (5.2), $\epsilon = 1e-9$.

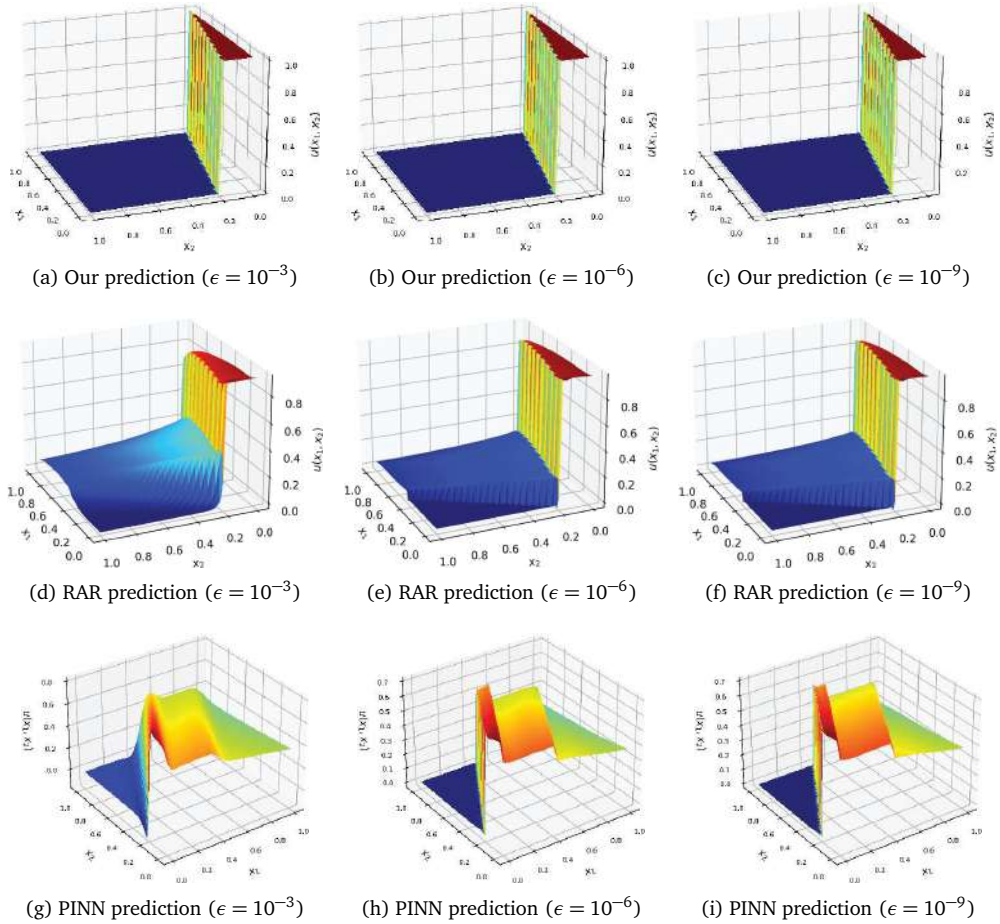
5.4. Two-dimensional convection-diffusion equation with interior layers

This section is devoted to assessing the performance of the proposed approach in the presence of interior layers. To this end, consider

$$-\epsilon \Delta u + \boldsymbol{b} \cdot \nabla u = 0, \quad \boldsymbol{x} \in \Omega = (0,1)^2,$$
$$u = \begin{cases} 1, & \text{if} \quad x_2 = 0, \\ 1, & \text{if} \quad x_1 = 0, \quad x_2 \le 1/5, \\ 0, & \text{elsewhere on} \ \partial\Omega, \end{cases} \tag{5.3}$$

where the convection coefficient $\boldsymbol{b} = (1/2, \sqrt{3}/2)^T$ [2]. The latent solution of Eq. (5.3) presents both internal and external boundary layers. For most traditional numerical methods, non-physical oscillations are often observed near the interior layer caused by the joints of the conflicting discontinuous boundary conditions.

As Fig. 9 shows, the internal layers are sharply captured by our approach with almost no overshooting/undershooting, while RAR shows slight overshooting/undershooting over there. In contrast, common PINN performs poorly and its predictions are highly oscillatory. Moreover, in this example, our method is stable with respect to various ϵ. When ϵ changes from $1e-3$ to $1e-9$, there is no obvious oscillation appearing in the prediction results.



(a) Our prediction ($\epsilon = 10^{-3}$) (b) Our prediction ($\epsilon = 10^{-6}$) (c) Our prediction ($\epsilon = 10^{-9}$)

(d) RAR prediction ($\epsilon = 10^{-3}$) (e) RAR prediction ($\epsilon = 10^{-6}$) (f) RAR prediction ($\epsilon = 10^{-9}$)

(g) PINN prediction ($\epsilon = 10^{-3}$) (h) PINN prediction ($\epsilon = 10^{-6}$) (i) PINN prediction ($\epsilon = 10^{-9}$)

Figure 9: Comparison between our approach, RAR and PINN for Eq. (5.3) under various diffusion coefficients.

5.5. Rotational flow

Consider the following rotational flow problem:

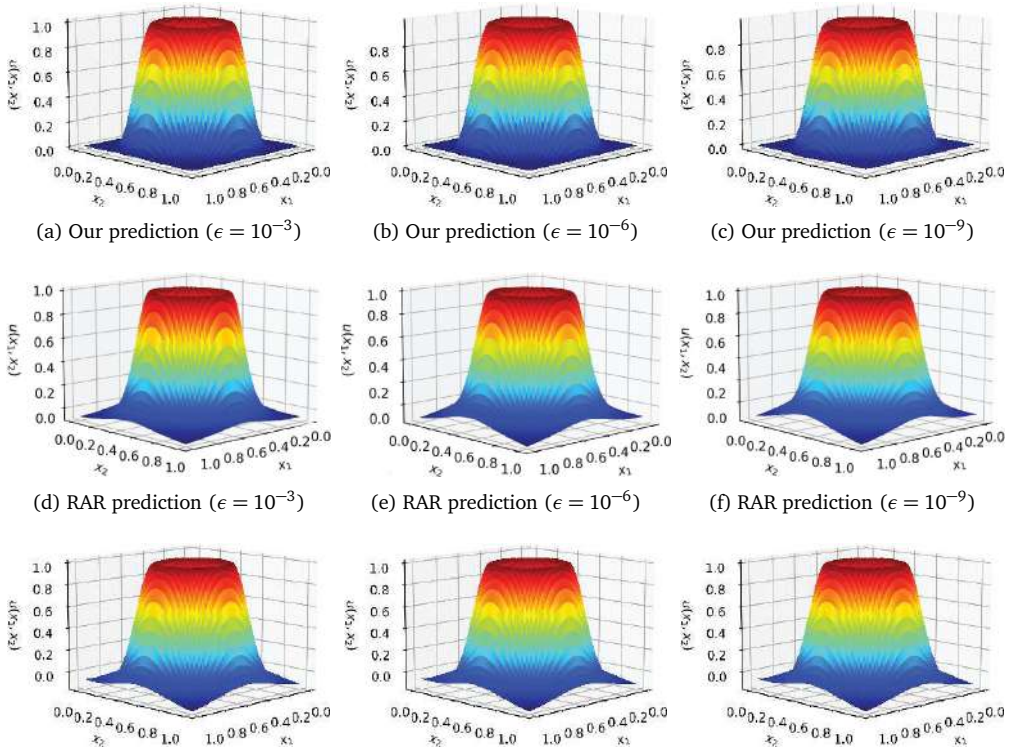
$$-\epsilon \Delta u + \nabla \cdot (\boldsymbol{b} u) = 0, \quad \boldsymbol{x} \in \Omega = (0,1)^2, \tag{5.4}$$

where the convection coefficient $\boldsymbol{b} = (1/2 - x_2, x_1 - 1/2)^T$, and the solution is prescribed along the slit $1/2 \times [0, 1/2]$ as follows:

$$u(1/2, x_2) = \sin^2(2\pi x_2), \quad x_2 \in [0, 1/2],$$

cf. Ref. [19]. The above equation describes the convection of a single component in a rotating flow field, where the axis of rotation passes through the center of the square domain.

Fig. 10 shows that our method yields satisfactory predictions, while the results of PINN and RAR have unreasonably negative values near the boundary corners.



(a) Our prediction ($\epsilon = 10^{-3}$) (b) Our prediction ($\epsilon = 10^{-6}$) (c) Our prediction ($\epsilon = 10^{-9}$)

(d) RAR prediction ($\epsilon = 10^{-3}$) (e) RAR prediction ($\epsilon = 10^{-6}$) (f) RAR prediction ($\epsilon = 10^{-9}$)

(g) PINN prediction ($\epsilon = 10^{-3}$) (h) PINN prediction ($\epsilon = 10^{-6}$) (i) PINN prediction ($\epsilon = 10^{-9}$)

Figure 10: Comparison between our approach, RAR and PINN for rotational flow (5.4) under various diffusion coefficients.

5.6. L-shaped domain

Consider the convection-diffusion-reaction problem

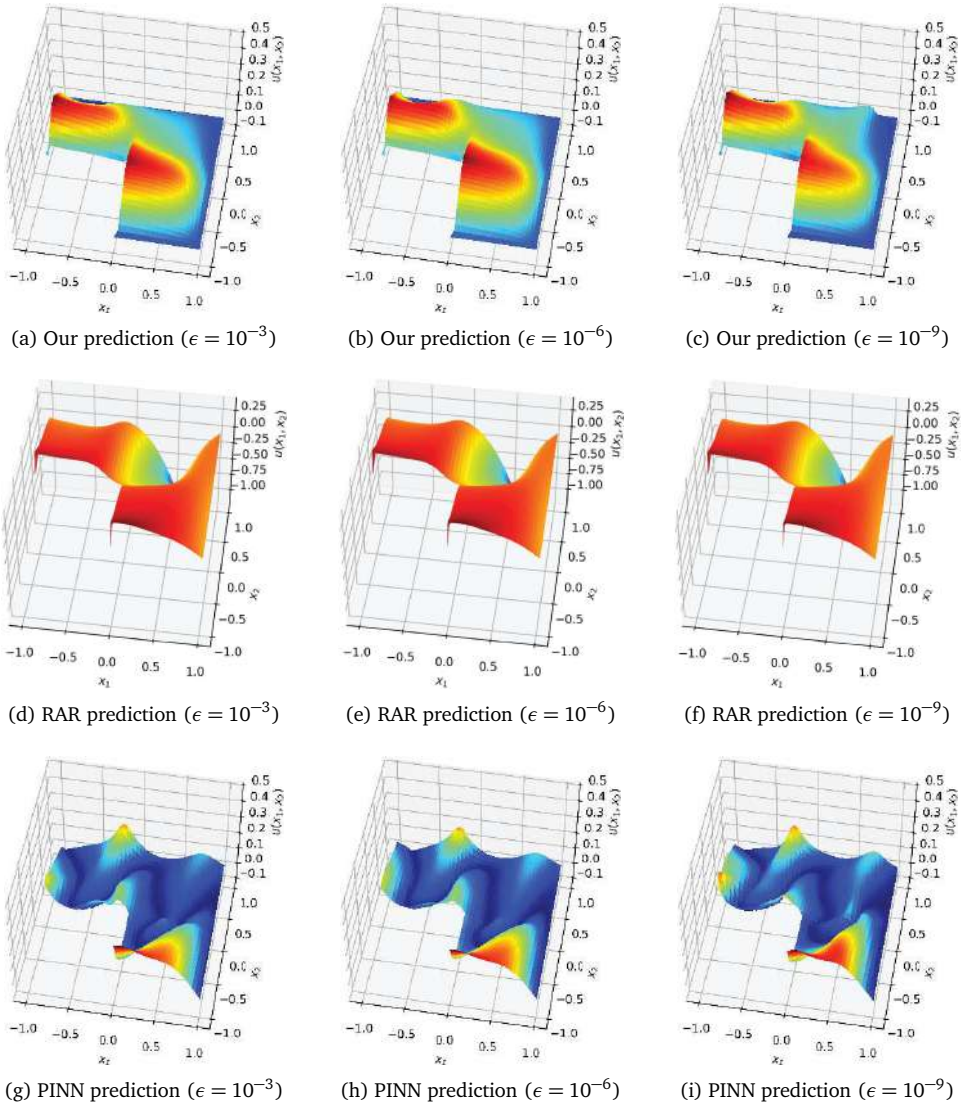
$$\begin{aligned} -\epsilon\Delta u + \boldsymbol{b}\cdot\nabla u + \big(3 + \sin(2\pi x_1 x_2)\big)u &= 1 - (x_1 + x_2)/2, && \boldsymbol{x} \in \Omega, \\ u &= 0, && \boldsymbol{x} \in \partial\Omega \end{aligned} \tag{5.5}$$

on the L-shaped domain $\Omega = (-1, 1)^2/(-1, 0)^2$ with

$$\boldsymbol{b} = -\big(1 + 1/2\sin(2\pi x_1), 2 - \cos(2\pi x_2)\big)^T,$$

cf. Ref. [24]. It results in boundary layers occurring at $x_1 = 0, -1$ and $x_2 = 0, -1$.

From Fig. 11, we can find that the boundary layers are well captured by our approach, with very slight overshooting/undershooting. In contrast, RAR and regular PINN perform very poorly and their predictions are highly oscillatory. It is also noticeable that in this example, our method seems to slightly degrade in performance as ϵ gets smaller.



(a) Our prediction ($\epsilon = 10^{-3}$) (b) Our prediction ($\epsilon = 10^{-6}$) (c) Our prediction ($\epsilon = 10^{-9}$)

(d) RAR prediction ($\epsilon = 10^{-3}$) (e) RAR prediction ($\epsilon = 10^{-6}$) (f) RAR prediction ($\epsilon = 10^{-9}$)

(g) PINN prediction ($\epsilon = 10^{-3}$) (h) PINN prediction ($\epsilon = 10^{-6}$) (i) PINN prediction ($\epsilon = 10^{-9}$)

Figure 11: Comparison of RAR [23], PINN, and our approach for Eq. (5.5) with various diffusion coefficients on L-shaped domain.

5.7. Three-dimensional singularly perturbed convection-diffusion problem

For traditional numerical methods, the singular perturbation equations in three spatial dimensions are difficult to solve due to the huge computational cost. On the contrary,

neural network are more powerful in dealing with high-dimensional problems. To this end, consider the following three-dimensional convection-diffusion problem:

$$\begin{aligned} -\epsilon\Delta u + \boldsymbol{b}\cdot\nabla u = f, \quad & \boldsymbol{x}\in\Omega, \\ u = 0, \quad & \boldsymbol{x}\in\partial\Omega, \end{aligned} \tag{5.6}$$

where $\Omega=(0,1)^3$, $\boldsymbol{b}=[1,2,1]^T$, and $f(x)$ is chosen such that the exact solution is given by

$$u=\sin(x_1)(1-e^{-(1-x_1)/\epsilon})(1-x_2)^2(1-e^{-x_2/\epsilon})(1-x_3)(1-e^{-x_3/\epsilon}).$$

The solution of (5.6) has three exponential layers at $x_1=1$, $x_2=0$ and $x_3=0$, respectively.

We compare the errors of our approach with RAR and PINN for solving a three-dimensional equation (5.6). As can be seen from Table 5, our method obtains about two orders of magnitude lower error than the normal PINN under various ϵ.

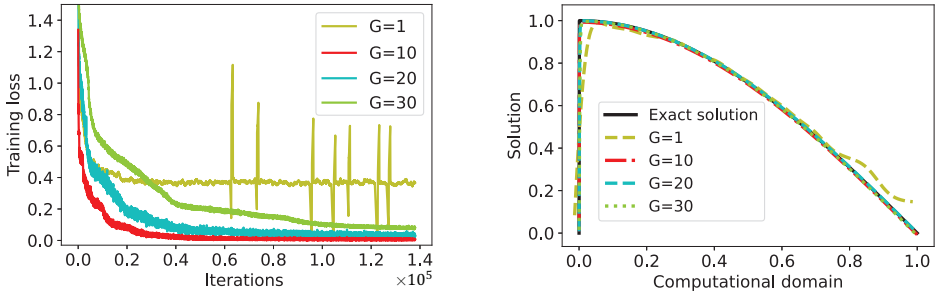
Table 5: Normalized root mean squared error and computational time of our approach for Eq. (5.6).

Diffusion coefficient	Ours	RAR	PINN
$\epsilon=1e-3$	4.37×10^{-3}	8.91×10^{-2}	2.58×10^{-1}
$\epsilon=1e-6$	4.46×10^{-3}	1.16×10^{-1}	4.68×10^{-1}
$\epsilon=1e-9$	4.43×10^{-3}	1.12×10^{-1}	4.72×10^{-1}

5.8. Sensitivity analysis

In our approach, there is an important hyperparameter G, which is used to quantify the magnitude of the gradients of the samples in the subset X_{sub}, and further helps to determine the threshold $\beta(t)$ for reweighting. In this subsection, we will study the effect of this hyperparameter. To this end, we take the Eq. (5.1) with $\epsilon=1e-9$ as an example, and then apply our approach using $G=1,10,20,30$, respectively.

From Fig. 12, we can find that the approach is stable with respect to a large parameter G. Whether $G=10,20$ or 30, the training process converges well, and the corresponding



(a) Training loss curves (b) Predictions

Figure 12: Sensitivity studies for the hyperparameter G.

predictions are almost identical. On the contrary, $G = 1$ results in an unstable training and larger prediction errors, which implies that a too small G cannot well distinguish the gradients from layer or non-layer regions.

6. Conclusion

PINNs fail to learn accurate approximations when dealing with singularly perturbed convection-diffusion-reaction problems whose solutions contain sharp boundary/interior layers. We studied this failure mode from a regional distribution perspective and revealed that the network fails to converge due to the extreme multiscale discrepancy in the underlying solutions between regions. We demonstrated that the widely used approach that prioritizing high-loss regions does not help in training. A curriculum learning approach was then developed that emphasizes learning of easier non-layer regions, thereby significantly improving the prediction accuracy of PINNs. Our study indicates for the first time that paying less attention to high-loss regions can be a feasible strategy for accurately learning the difficult equations with strong multiscale characteristics.

Acknowledgments

This research is partially supported by the National Natural Science Foundation of China (Grant 11771257) and by the Natural Science Foundation of Shandong Province (Grant ZR2021MA010).

References

[1] A. Arzani, K.W. Cassel and R.M. D'Souza, *Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation*, J. Comput. Phys. **473**, 111768 (2023).

[2] B. Ayuso and L. Marini, *Discontinuous Galerkin methods for advection-diffusion-reaction problems*, SIAM J. Numer. Anal. **47**, 1391–1420 (2009).

[3] Y. Bengio, J. Louradour, R. Collobert and J.Weston, *Curriculum learning*, in: *International Conference on Machine Learning*, International Machine Learning Society, 41–48 (2009).

[4] A. Brooks and T. Hughes, *Streamline upwind/Petrov-Galerkin methods for advection dominated flows*, in: *Proceeding of the Third International Confference on Finite Element Methods in Fluid Flow*, 283–292 (1980).

[5] L.W. Colby and J. Zhao, *Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks*, Commun. Comput. Phys. **29**, 930–954 (2021).

[6] A. Daw, J. Bu, S. Wang, P. Perdikaris and A. Karpatne, *Rethinking the importance of sampling in physics-informed neural networks*, arXiv:2207.02338 (2022).

[7] Z. Gao, L. Yan, T. Tang and T. Zhou, *Failure-informed adaptive sampling for PINNs, Part II: Combining with re-sampling and subset simulation*, arXiv:2302.01529 (2023).

[8] Z.Gao, L. Yan and T. Zhou, *Failure-informed adaptive sampling for PINNs*, SIAM J. Sci. Comput. **45**, A1971–A1994 (2023).

[9] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks*, in: *International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, 249–256 (2010).

[10] H. Guo and X. Yang, *Deep unfitted Nitsche method for elliptic interface problems*, Commun. Comput. Phys. **31**, 1162–1179 (2022).
[11] L. Guo, H. Wu and T. Zhou, *Normalizing field flows: Solving forward and inverse stochastic differential equations using physics-informed flow models*, J. Comput. Phys. **461**, 111202 (2022).
[12] S. Guo, W. Huang, H. Zhang, C. Zhuang, D. Dong, M.R. Scott and D. Huang, *Curriculumnet: Weakly supervised learning from large-scale web images*, in: *Proceedings of the European Conference on Computer Vision*, 135–150 (2018).
[13] G. Hacohen and D. Weinshall, *On the power of curriculum learning in training deep network*, in: *International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2535–2544 (2019).
[14] J. Han, A. Jentzen and W. E, *Solving high-dimensional partial differential equations using deep learning*, in: *Proceedings of the National Academy of Sciences*, 8505–8510 (2018).
[15] Q. He and A.M. Tartakovsky, *Physics-informed neural network method for forward and backward advection-dispersion equations*, Water Resour. Res. **57**, e2020WR029479 (2021).
[16] C. Hirsch, *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*, Butterworth-Heinemann, Elsvier (2007).
[17] J. Huang, C. Wang and H. Wang, *A deep learning method for elliptic hemivariational inequalities*, East Asian J. Appl. Math. **12**, 487–502 (2022).
[18] J. Huang , H. Wang and T. Zhou, *An augmented Lagrangian deep learning method for variational problems with essential boundary conditions*, Commun. Comput. Phys. **31**, 966–986 (2022).
[19] T. Hughes, G. Scovazzi, P. Bochev and A. Buffa, *A multiscale discontinuous Galerkin method with the computational structure of a continuous Galerkin method*, Comput. Methods Appl. Mech. Engrg. **195**, 2761–2787 (2006).
[20] D. Kingma and J. Ba, *ADAM: A method for stochastic optimization*, in: *International Conference on Learning Representations*, 6429–6462 (2015).
[21] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby and M. Mahoney, *Characterizing possible failure modes in physics-informed neural networks*, in: *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 26548–26560, (2021).
[22] X. Liu, H. Lai, D. Wong and L. Chao, *Norm-based curriculum learning for neural machine translation*, in: *Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 6358–6369 (2020).
[23] L. Lu, X. Meng, Z. Mao and G.E. Karniadakis, *DeepXDE: A deep learning library for solving differential equations*, SIAM Review **63**, 208–228 (2021).
[24] L. Ludwig and H.G. Roos, *Convergence and supercloseness of a finite element method for a singularly perturbed convection-diffusion problem on an L-shaped domain*, IMA J. Numer. Anal. **36**, 1261–1280 (2016).
[25] H. Ma, Y. Zhang, N. Thuerey, X. Hu and O.J. Haidn, *Physics-driven learning of the steady Navier-Stokes equations using deep convolutional neural networks*, Commun. Comput. Phys. **32**, 715–736 (2022).
[26] J. Miller, *Singular Perturbation Problem in Chemical Physics: Analytic and Computational Methods*, John Wiley and Sons (1997).
[27] J. Miller, E. O'Riordan and G. Shishkin, *Fitted Numerical Methods for Singular Perturbation Problems*, World Scientific (1996).
[28] R. Mojgani, M. Balajewicz and P. Hassanzadeh, *Kolmogorov n-width and Lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated PDEs*, Comput. Methods Appl. Mech. Engrg. **404**, 115810 (2023).
[29] S. Polak, C. Denheijer and W. Schilders, *Semiconductor device modelling from the numerical point of view*, Int. J. Numer. Methods Eng. **24**, 763–838 (1987).

[30] M. Raissi, P. Perdikaris and G.E. Karniadakis, *Physics informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, J. Comput. Phys. **378**, 686–707 (2019).

[31] H.G. Roos, M. Stynes and L. Tobiska, *Robust Numerical Methods for Singularly Perturbed Differential Equations*, Springer-Verlag (2008).

[32] N. Sarafianos, T. Giannakopoulos, C. Nikou and I. Kakadiaris, *Curriculum learning of visual attribute clusters for multi-task classification*, Pattern Recognit. **80**, 94–108 (2018).

[33] M. Stynes and L. Tobiska, *The SDFEM for a convection-diffusion problem with a boundary layer: Optimal error analysis and enhancement of accuracy*, SIAM J. Numer. Anal. **41**, 1620–1642 (2003).

[34] S. Wang, X. Yu and P. Perdikaris, *When and why pinns fail to train: A neural tangent kernel perspective*, J. Comput. Phys. **449**, 110768 (2021).

[35] S. Wu, A. Zhu, Y. Tang and B. Lu, *Convergence of physics-informed neural networks applied to linear second-order elliptic interface problems*, Commun. Comput. Phys. **33**, 596–627 (2023).

[36] B. Xu, L. Zhang, Z. Mao, Q. Wang, H. Xie and Y. Zhang, *Curriculum learning for natural language understanding*, in: *Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 6095–6104 (2020).

[37] J. Zhang and X. Liu, *Analysis of SDFEM on Shishkin triangular meshes and hybrid meshes for problems with characteristic layers*, J. Sci. Comput. **68**, 1299–1316 (2016).

[38] J. Zhang, X. Liu and M. Yang, *Optimal order L2 error estimate of SDFEM on Shishkin triangular meshes for singularly perturbed convection-diffusion equations*, SIAM J. Numer. Anal. **54**, 2060–2080 (2016).