Improving Boundary Layer Predictions Using Parametric Physics-Aware Neural Networks

Antônio Tadeu Azevedo Gomes(✉), Larissa Miguez da Silva, and Frédéric Valentin

Laboratório Nacional de Computação Científica (LNCC), Av. Getúlio Vargas 333, Quitandinha, Petrópolis-RJ, Brazil
{atagomes,lamiguez,valentin}@lncc.br

Abstract. *Physics-Informed Neural Networks* (PINNs) are machine learning tools that approximate the solution of general partial differential equations (PDEs) by adding them in some form as terms of the loss/cost function of a Neural Network. Most pieces of work in the area of PINNs tackle *non-linear* PDEs. Nevertheless, many interesting problems involving linear PDEs may benefit from PINNs; these include parametric studies, multi-query problems, and parabolic (transient) PDEs. The purpose of this paper is to explore PINNs for linear PDEs whose solutions may present one or more *boundary layers*. More specifically, we analyze the steady-state reaction-advection-diffusion equation in regimes in which the diffusive coefficient is small in comparison with the reactive or advective coefficients. We show that adding information about these coefficients as predictor variables in a PINN results in better prediction models than a PINN that only uses spatial information as predictor variables. Even though using these coefficients when training a PINN model is a common strategy for inverse problems, to the best of our knowledge we are the first to consider these coefficients for parametric direct problems. This finding may be instrumental in multiscale problems where the coefficients of the PDEs present high variability in small spatiotemporal regions of the domain, and therefore PINNs may be employed together with domain decomposition techniques to efficiently approximate the PDEs locally at each partition of the spatiotemporal domain, without resorting to different learned PINN models at each of these partitions.

Keywords: Physics-informed neural networks · Boundary layer problems · Multiscale methods

1 Introduction

Physics-Aware Neural Networks (NNs) are machine learning tools that approximate the solution of general partial differential equations (PDEs) by adding the

The authors are presented in alphabetical order.

physical laws these equations represent to some component of the neural network. The PINNs [19] are likely to be the most well-known of these NNs; the defining characteristic of a PINN is the inclusion of the strong form of a PDE (including its boundary and initial conditions) as terms of the loss/cost function.

Most pieces of work in the area of physics-aware NNs tackle *non-linear* PDEs [6,10,11,14,17]. Nevertheless, many interesting problems involving linear PDEs may benefit from physics-aware NNs; these include parametric studies, multi-query problems, and parabolic (transient) PDEs.

We are mostly interested in the solution of linear PDEs whose coefficients present high variability in small spatiotemporal regions of the physical domain. In this case, we say that the solution has a multiscale behavior. Standard numerical methods often present difficulties in approximating the solution to such PDEs with combined quality and computational affordability. Multiscale numerical methods (e.g. [9]) have emerged as an attractive option for dealing with such difficulties by rewriting the original formulation of the PDE in terms of: (i) local problems living each one in a partition of the physical domain; and (ii) a global problem that "glues together" the solution of the local problems. The price to pay is a potentially large number of local problems. Although said local problems are independent from one another, thus benefiting from massive parallel computations, they may still be computationally demanding.

The purpose of this paper is to investigate the potential of physics-aware NNs in general, and PINNs specifically, for efficiently solving local problems in multiscale numerical methods. We explore the particular case of linear PDEs whose solutions may present one or more *boundary layers*. More specifically, we analyze the steady-state reaction-advection-diffusion equation in regimes in which the diffusive coefficient is small in comparison with the reactive or advective coefficients. We verify that adding information about these coefficients as predictor variables in a PINN results in better prediction models than a PINN that only uses spatial information as predictor variables. Even though using these coefficients when training a PINN model is a common strategy for *inverse* problems, to the best of our knowledge we are the first to consider these coefficients for parametric direct problems. We believe this finding may be instrumental in multiscale problems, because it opens the path for PINNs to be employed together with domain decomposition techniques to efficiently approximate the PDEs locally at each partition of the spatiotemporal domain, without resorting to different learned PINN models at each of these partitions.

The remainder of this paper is structured as follows. In Sect. 2, we quickly review the related literature. In Sect. 3, we present the problem and the methodology for the proposed model. In Sect. 4, we examine two different cases of the target equation and the effectiveness of the proposed model. Finally, in Sect. 5, we report the conclusions of this work along with a discussion of future directions.

2 Related Work

In recent years, the use of algorithms that "learn" from data has caused great impact and change in several areas of science. Algorithms using NNs have been used in many problems governed by PDEs and presented satisfactory results [2,15,20,21]. In particular, in [19] the methodology of PINNs was first proposed, combining the properties of universal approximation of NNs and the knowledge of physical laws described by PDEs. Since then, many pieces of work have been published on this topic [3,16,18]. However, problems with complex geometry domains have led to other methodologies based on domain decomposition methods and PINNs, including Extended PINNs (XPINNs) [12], Conservative PINNs (cPINNs) [11] and Variational PINNs (VPINNs) [14].

Although the aforementioned pieces of work have shown excellent results, there are still many theoretical gaps that need to be filled. The techniques are new and do not have a trivial application in the solution of physical problems. For instance, a particularly complex step in formulating deep learning problems and PINNs is the definition of the loss functional to be minimized. Additionally, there are many hyperparameters to be configured and, although the automatic selection of hyperparameters is possible, there is usually a large computational cost. Interestingly, this last problem also exists somehow in a handful of multiscale methods with regard to their configuration parameters (e.g. [7,8]).

There is a growing number of papers relating multiscale methods and data-driven approaches [4,5,13,22]. To the best of our knowledge, none of these pieces of work tackle the problem the way we do, which is by training a single machine learning model that may be parameterized for approximating the solution of a PDE with highly variable coefficients in the spatial domain.

3 Methodology

3.1 Boundary Layer Problem

The boundary layer problem can appear in many applications, including fluid dynamics, meteorology, atomic reactors, among others. This phenomenon occurs when the gradient is high in the region close to the boundary and can bring instability to the discrete solution of the problem. Next, we present an example for this case.

Consider the case in which the reaction-advection-diffusion problem has an exact solution which contains boundary layers. This happens when the reactive or advective coefficient dominates the diffusive one. We consider the following reaction-advection-diffusion problem: *Find* $u \in H^1(\Omega)$ *such that*:

$$\begin{cases} \nabla \cdot (-\mathcal{K}\nabla u + \alpha u) + \sigma u = 1 \text{ in } \Omega, \\ \hfill u = 0 \text{ on } \partial\Omega_D, \\ \hfill \nabla u \cdot \mathbf{n} = 0 \text{ on } \partial\Omega_N, \end{cases} \tag{1}$$

where Ω is a unit square domain, $\partial\Omega_D$ corresponds to the boundaries $x = (0, y)$ and $x = (1, y)$, with $y \in (0, 1)$, where homogeneous Dirichlet conditions are to be enforced, and $\partial\Omega_N = \partial\Omega \backslash \partial\Omega_D$ corresponds to the boundaries where homogeneous Neumann conditions are to be enforced. The coefficients are such that $\alpha := (a, 0)^T$, $\mathcal{K} = k\mathcal{I}$, where $\mathcal{I}$ is the identity matrix and a, k, $\sigma \in \mathbb{R}$. If $\sigma > 0$, the analytical solution to this equation is:

$$u(x,y) = \frac{\sinh(\frac{\sqrt{4k\sigma}}{2k}(x-1)) - \sinh(\frac{\sqrt{4k\sigma}}{2k}x)}{\sinh(\frac{\sqrt{a^2+4k\sigma}}{2k})} + 1\,,$$

Otherwise, if $\sigma = 0$ and $a > 0$, then the exact solution becomes

$$u(x,y) = \frac{1}{a}\left(x - \frac{\sinh(\frac{\sqrt{a}}{2k}x)}{\sinh(\frac{\sqrt{a}}{2k})}e^{\frac{\sqrt{a}}{2k}(x-1)}\right)$$

We consider two experimental settings. First, in Subsect. 4.1 we will assume $\sigma = 1$ and $a = 0$. So, we will explore the case in which the reactive coefficient dominates the diffusive one. Next, in Subsect. 4.2 we will consider the case in which the advective coefficient dominates by assuming that $\sigma = 0$ and $a = 1$.

3.2 Architecture Design

To solve the problem presented in Subsect. 3.1, we use the PINN depicted in Fig. 1. A key feature of this PINN is the use of the diffusion coefficient k as a predictor variable together with the spatial data (x, y). This way, we aim at getting a model capable of making predictions for different diffusion coefficients.

We assume a feed-forward NN with the following structure: 4 fully connected layers each containing 24 neurons and each followed by a hyperbolic tangent activation function. Furthermore, we use one output layer of size 1 and a linear activation function. These hyperparameters as well as all other configurations not explicitly explained in the remainder of the text have been determined empirically.

For the sake of comparison, we establish two scenarios for the input layer, as will be further explained in the following section: (i) *Scenario 1*, with only 2 neurons, input k being taken out; (ii) *Scenario 2* with 3 neurons, exactly as shown in Fig. 1.

Also, we considered the loss function

$$\phi_\theta(X) := c_1\phi_\theta^{bd}(X^{bd}) + c_2\phi_\theta^{bn}(X^{bn}) + c_3\phi_\theta^{r}(X^{r}), \tag{2}$$

as a function of the training data, as also explained in the following section.

4 Experimental Results

In this section, we present some numerical results that show the performance of PINNs to solve the boundary layer problem.[1] In our simulations, we consider the following scenarios:

[1] The experiments presented in this paper can be reproduced in Google Colaboratory: https://colab.research.google.com/drive/1dzzK41xIrmi5ozzO4IzBnkktGjI90j_-.

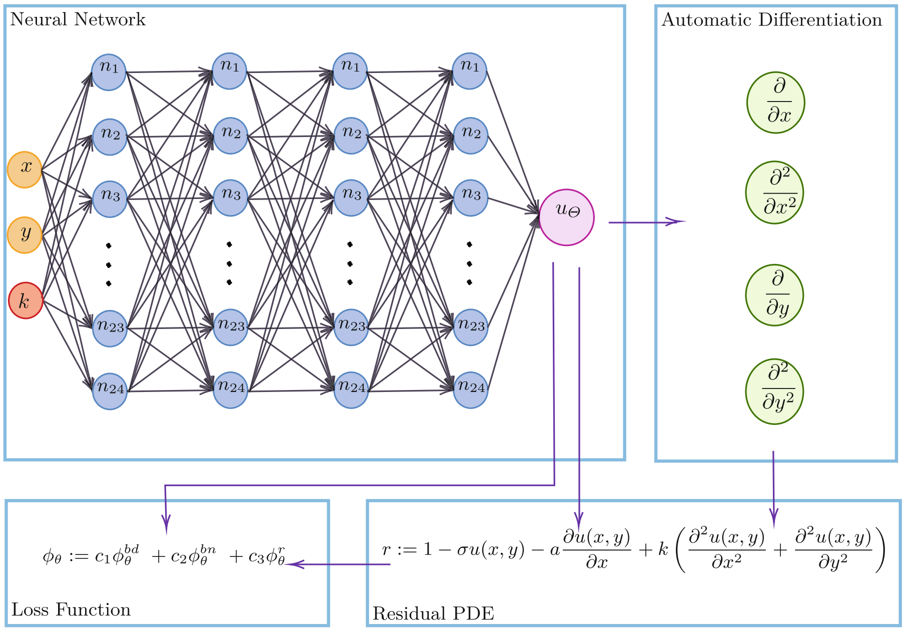

Neural Network
x
y
k
n_1
n_2
n_3
n_{23}
n_{24}
u_Θ
Automatic Differentiation
$\frac{\partial}{\partial x}$
$\frac{\partial^2}{\partial x^2}$
$\frac{\partial}{\partial y}$
$\frac{\partial^2}{\partial y^2}$
$\phi_\theta := c_1\phi_\theta^{bd} + c_2\phi_\theta^{bn} + c_3\phi_\theta^{r}$
Loss Function
$r := 1 - \sigma u(x,y) - a\frac{\partial u(x,y)}{\partial x} + k\left(\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2}\right)$
Residual PDE

Fig. 1. PINN architecture for Reaction-Advection-Diffusion problem.

Scenario 1: We fix a diffusion coefficient k, train the network for some collocation and boundary points and predict others. Therefore, the input parameters of the network are (x, y) and the output is the solution u.

Scenario 2: We vary the diffusion coefficient k and train the network for some k to predict. Therefore, the input parameters of the network are (x, y, k) and the output is the solution u.

The collocation points are given by X_r, the boundary data is in X_{bd} and X_{bn}, where X_{bd} represents the data on the Dirichlet boundary Ω_D and X_{bn} the data on the Neumann boundary Ω_N. Respectively, on those boundary points, we have the solutions u_{bd} and u_{bn}. Additionally, the coefficients c_1, c_2 and c_3 representing the weights of each loss term are hyperparameters of the model, and their values have been chosen empirically based on the knowledge of the authors about the behavior of Eq. 1 for different values of its coefficients.

We assume that the collocation points X_r as well as the points for the boundary data X_{bd} and X_{bn} are randomly sampled from a uniform distribution.

4.1 First Setting: Reaction-Diffusion Problem

For this first problem, we began with a training data of size $N_{bd} = N_{bn} = 200$ and $N_r = 1000$, where N_{bd} is when we apply the Dirichlet boundary condition and N_{bn} when we apply the Neumann boundary condition. We illustrate this setting in Fig. 2.

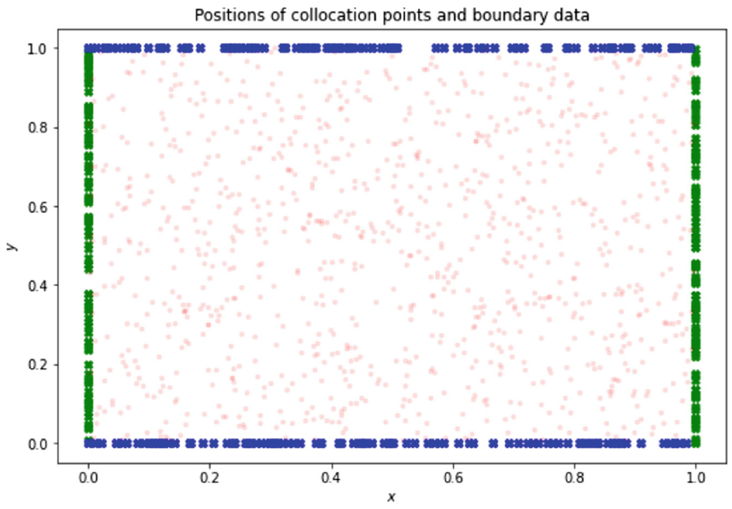
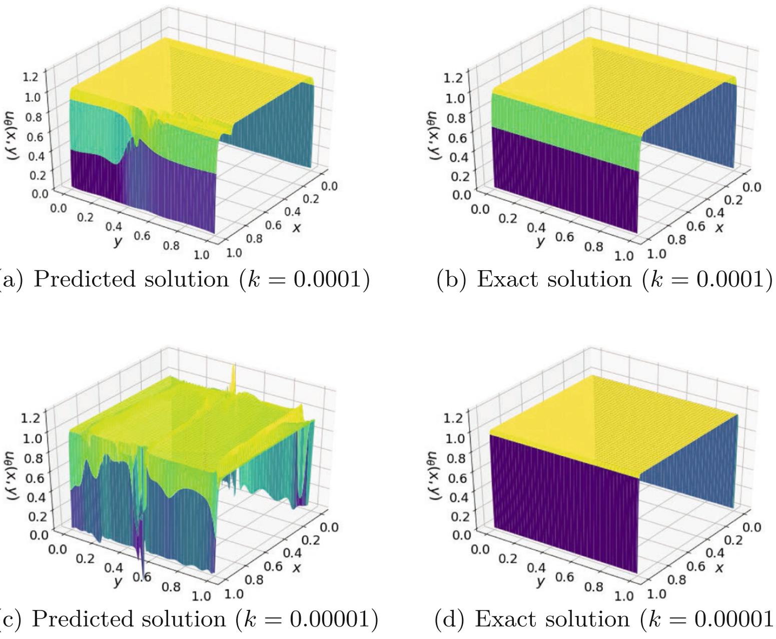


Fig. 2. The collocation points (red circles) and the positions where the Dirichlet boundary condition (blue cross marks) and the Neumann boundary condition (green cross marks) will be weakly imposed. (Color figure online)

Figure 3 depicts some experimental results for Scenario 1. We observe that as we shrink the diffusive coefficient k, the solution gets worse, with some undesired features when $k = 0.0001$ and $k = 0.00001$. (Nonetheless, we see in Fig. 6 that the PINN is able to approximate the solution with a small error when $k = 1.0$, $k = 0.1$, and $k = 0.01$.) Besides the difficulty of approximating the solution in the case where we have a very small k, another disadvantage of this approach is that the model needs to be retrained for each new k.

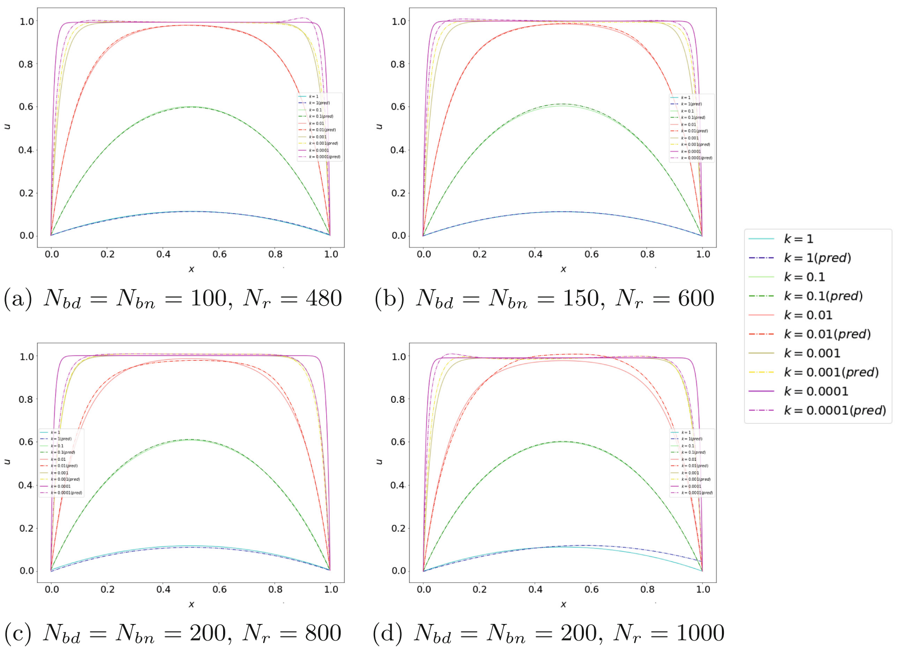
For Scenario 2, we add 20 different, randomly sampled $k \in (0.0001, 1.0)$ to the input data. First, we investigate the sensitivity of the PINN with respect to the amount and dispersion of collocation training points in Scenario 2. In Fig. 4, we show the exact solution and the solution field generated by a PINN with decreasing values of k, for different amounts of collocation and boundary points. We plot a cut for a fixed y; the exact solutions are represented by the solid lines and the predicted PINN solutions are represented by the dotted lines.

We can observe that the proposed PINN architecture for Scenario 2 interpolates quite well for values of k greater than or equal to 0.001, but for $k = 0.0001$ we have significant errors. Once more we see the impact of the boundary layer problem on the predictions.


(a) Predicted solution ($k = 0.0001$) (b) Exact solution ($k = 0.0001$)

(c) Predicted solution ($k = 0.00001$) (d) Exact solution ($k = 0.00001$)

Fig. 3. Scenario 1, reaction-diffusion setting: predicted solution vs exact solution with respect to parameter k.

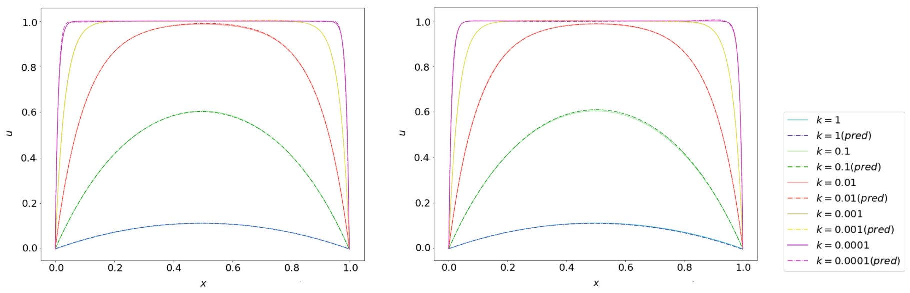


(a) $N_{bd} = N_{bn} = 100,\ N_r = 480$ (b) $N_{bd} = N_{bn} = 150,\ N_r = 600$

(c) $N_{bd} = N_{bn} = 200,\ N_r = 800$ (d) $N_{bd} = N_{bn} = 200,\ N_r = 1000$

Fig. 4. Scenario 2, reaction-diffusion setting: sensitivity analysis with respect to the size of the training data.

In the above experiments we trained the PINNs with $c_1 = 2$, $c_2 = 1$ and $c_3 = 0.01$ as the weights of the loss terms. In Fig. 5 we show the sensitivity of the model for Scenario 2 with respect to the loss weights.



(a) $c_3 = 0.2$ (b) $c_3 = 0.4$

Fig. 5. Scenario 2, reaction-diffusion setting: sensitivity analysis with respect to loss weights.

For the cases presented in Fig. 5 we used $N_{bd} = N_{bn} = 100$, $N_r = 240$. The results are even more impressive because the PINN algorithm in Scenario 2 is able to reconstruct the solution field with high precision from a small number of points used for the training, even for the case where we have a very small k. Therefore, the results clearly show the impact of the loss weights on the training.

Finally, we compare the errors originating from Scenario 1 and Scenario 2. We use the Relative Mean Square Error (RMSE) to compare the results of the different scenarios, as presented in Fig. 6. We observe a significant error increase with a decaying k for Scenario 1, whereas for Scenario 2 this increase is much slower, specially for $k \in (0.001, 1.0)$. However, the much higher errors for Scenario 2 with larger values of k are still largely unexplained and motivates a series of investigations as part of our future work.

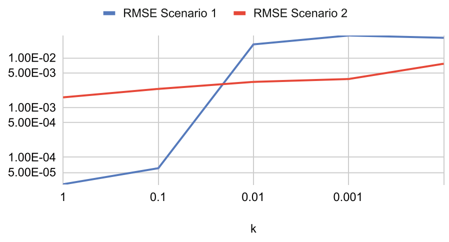
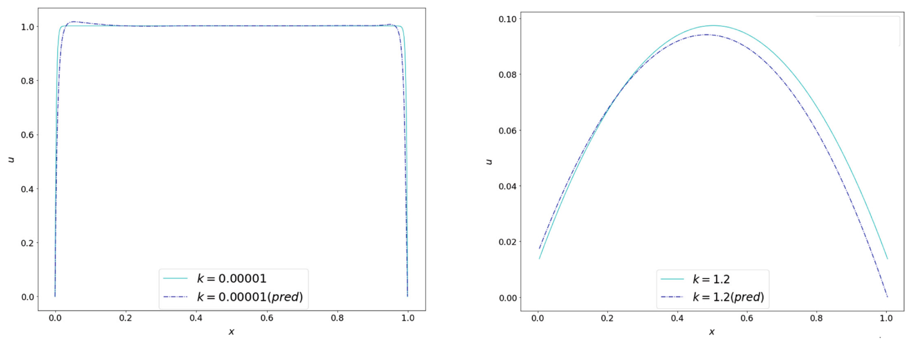


Fig. 6. Quality of prediction for the reaction dominant case (log-log scale).

Now, we take advantage of the fact that we can predict for different values of k in Scenario 2 and try to perform an extrapolation. The results are presented in Fig. 7, in which we plot two different cases. In Fig. 7(a), we extrapolate for an even smaller k, where the boundary layer problem is more evident. Even so, we still get a good approximation; we believe this is due to the fact that the diffusion coefficient is close to the range of k used for the training. The same does not occur in Fig. 7(b), when we try to extrapolate to a larger value of k. (Mind, however, that we have plotted Fig. 7(b) in a different scale, to emphasize the prediction error.) The curve for the exact solution is far from the curve that represents the predicted solution although this case is a completely boundary layer free problem. This result is likely an indication that the proposed method still does not work well for extrapolations far from the set used for the training. Also note that the error is particularly high near the Dirichlet boundary $(1, y)$, with $y \in (0, 1)$, which shows that imposing the boundary condition weakly by means of a loss term may be tricky.



(a) Extrapolating to $k = 10^{-5}$ (b) Extrapolating to $k = 1.2$

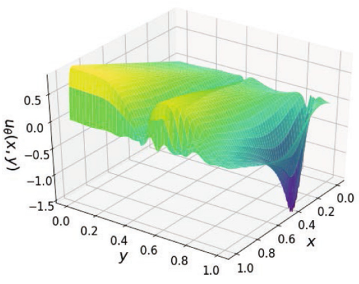
Fig. 7. Scenario 2, reaction-diffusion setting: extrapolating for different values of k.

4.2 Second Setting: Advection-Diffusion Problem

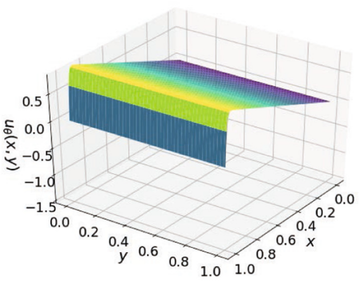
Here, similarly to Subsect. 4.1, we will consider the same two scenarios, as well as the same PINN architecture, and the same number of collocation points already described. What we will change are the loss weights, now set to $c_1 = 1$, $c_2 = 1.2$ and $c_3 = 1$.

Figure 8 depicts some experimental results for Scenario 1. As in the reaction-diffusion setting, we observe that as we shrink the diffusive coefficient k, the solution gets worse. The advection-diffusion problem is nevertheless much tougher to approximate well than the reaction-diffusion problem. The figure clearly shows this, with completely wrong solutions when $k = 0.01$ and $k = 0.001$.

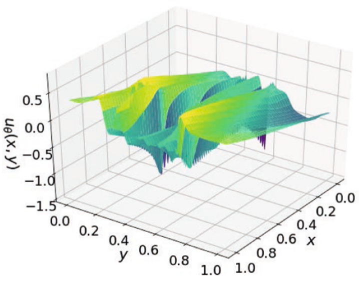
For Scenario 2, we add 20 different, randomly sampled $k \in (0.001, 1.0)$ to the input data. In Fig. 9, we show the exact solution and the solution field generated by a PINN with decreasing values of k for this scenario. The results are again



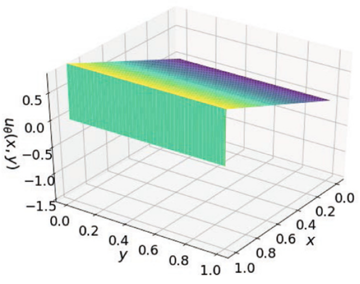
(a) Predicted solution ($k = 0.01$)



(b) Exact solution ($k = 0.01$)



(c) Predicted solution ($k = 0.001$)



(d) Exact solution ($k = 0.001$)

Fig. 8. Scenario 1, advection-diffusion setting: predicted solution vs exact solution with respect to parameter k.

impressive; the PINN algorithm in Scenario 2 is able to reconstruct the solution field with high precision from a small number of points used for the training, even for the case where we have a very small k.

Lastly, Fig. 10 presents the RMSE originating from Scenario 1 and Scenario 2. Again, for Scenario 2 the increase in the error is much slower than for Scenario 1. Nevertheless, as in the reaction-diffusion case, the much higher errors for Scenario 2 with larger values of k are still largely unexplained and motivates a series of investigations as part of our future work.

5 Summary and Outlook

The results presented herein clearly show the potential of PINNs for predicting the solution of PDEs with complex geometries and highly variable coefficients. Bringing physical coefficients into the training stage is key to avoiding the discrete solution's spurious oscillatory behavior in singularly perturbed regimes. In addition, it allows obtaining an accurate parameterization of the discrete solution concerning the physical coefficient in the interpolation scenario. However, when we extrapolate, we can observe that this methodology will hardly overcome the numerical methods to solve direct linear problems and, while automatic hyper-parameter selection is possible, it can be expensive.

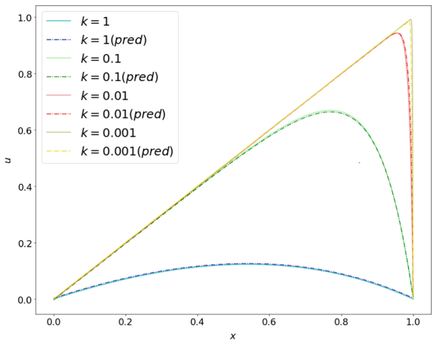


Fig. 9. Scenario 2, advection-diffusion setting: predicted solution vs exact solution with respect to parameter k.

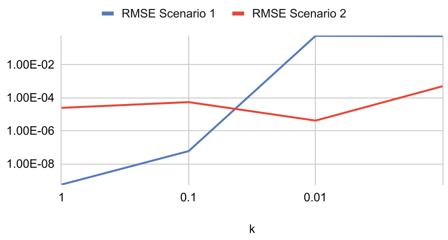


Fig. 10. Quality of prediction for the advection dominant case (log-log scale).

One way to solve this problem is to combine the strengths of numerical methods and data science by creating hybrid combinations of theory-based and data science models. We will focus on the family of Multiscale Hybrid-Mixed (MHM) methods [1,9] and their interaction with PINNs. The MHM methods are attractive because of their approximation properties and massive parallelization capability, which allows the physical properties of the model to be treated locally and efficiently, thanks to the concept of local multiscale functions. So, we envision PINNs being used within MHM as surrogate models to predict the shape of the multiscale basis functions in parallel, among other possibilities. This combination will be the subject for future work.

Other topics for future work include: (i) to explore alternatives to impose boundary conditions strongly (e.g. as in [6]); (ii) to apply the technique to other parametric studies, such as solutions with oscillatory behavior arising in oscillatory coefficient models or wave equation propagation problems; (iii) to consider the use of PINNs for other expensive linear problems, such as in multi-query scenarios.

Acknowledgment. The authors were supported by CAPES/Brazil under Project EOLIS (No. 88881.520197/2020-01). Frédéric Valentin was supported by Inria/France under the Inria International Chair, by CNPq/Brazil under Project No. 309173/2020-5, and by FAPERJ/Brazil under Project No. E-26/201.182/2021.

References

1. Barrenechea, G., Jaillet, F., Paredes, D., Valentin, F.: The multiscale hybrid mixed method in general polygonal meshes. Numerische Mathematik **145**(1), 197–237 (2020). https://doi.org/10.1007/s00211-020-01103-5
2. Berg, J., Nyström, K.: A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing **317**, 28–41 (2018)
3. Cai, S., Wang, Z., Wang, S., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks for heat transfer problems. J. Heat Transfer **143**(6), 060801 (2021)
4. Capuano, G., Rimoli, J.J.: Smart finite elements: a novel machine learning application. Comput. Methods Appl. Mech. Eng. **345**, 363–381 (2019)
5. Chan, S., Elsheikh, A.H.: A machine learning approach for efficient uncertainty quantification using multiscale methods. J. Comput. Phys. **354**, 493–511 (2018)
6. E, W., Yu, B.: The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**(1), 1–12 (2018). https://doi.org/10.1007/s40304-018-0127-z
7. Fabian, J.H.L., Gomes, A.T.A., Ogasawara, E.: Estimating the execution time of fully-online multiscale numerical simulations. In: Proceedings of the XXI Brazilian Symposium on High-Performance Computing Systems (WSCAD), pp. 191–202. Sociedade Brasileira de Computação (2020)
8. Fabian, J.H.L., Gomes, A.T.A., Ogasawara, E.: Estimating the execution time of the coupled stage in multiscale numerical simulations. In: Nesmachnow, S., Castro, H., Tchernykh, A. (eds.) CARLA 2020. CCIS, vol. 1327, pp. 86–100. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68035-0_7
9. Harder, C., Paredes, D., Valentin, F.: On a multiscale hybrid-mixed method for advective-reactive dominated problems with heterogeneous coefficients. Multiscale Model. Simul. **13**(2), 491–518 (2015)
10. Jagtap, A., Karniadakis, G.: Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Commun. Comput. Phys. **28**(5), 2002–2041 (2020). https://global-sci.org/intro/article_detail/cicp/18403.html
11. Jagtap, A.D., Kharazmi, E., Karniadakis, G.E.: Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. Comput. Methods Appl. Mech. Eng. **365**, 113028 (2020). https://www.sciencedirect.com/science/article/pii/S0045782520302127
12. Jagtap, A.D., Karniadakis, G.E.: Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. In: AAAI Spring Symposium: MLPS (2021)
13. Karpatne, A., Watkins, W., Read, J., Kumar, V.: Physics-guided neural networks (PGNN): an application in lake temperature modeling. arXiv preprint arXiv:1710.11431 (2017)
14. Kharazmi, E., Zhang, Z., Karniadakis, G.E.: Variational physics-informed neural networks for solving partial differential equations (2019). https://arxiv.org/abs/1912.00873

15. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. **9**(5), 987–1000 (1998)
16. Mao, Z., Jagtap, A.D., Karniadakis, G.E.: Physics-informed neural networks for high-speed flows. Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020)
17. Liao, Y., Ming, P.: Deep Nitsche method: deep Ritz method with essential boundary conditions. Commun. Comput. Phys. **29**(5), 1365–1384 (2021). https://doi.org/10.4208/cicp.OA-2020-0219
18. Misyris, G.S., Venzke, A., Chatzivasileiadis, S.: Physics-informed neural networks for power systems. In: 2020 IEEE Power & Energy Society General Meeting (PESGM), pp. 1–5. IEEE (2020)
19. Raissi, M., Perdikaris, P., Karniadakis, G.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019). https://www.sciencedirect.com/science/article/pii/S0021999118307125
20. Raissi, M., Karniadakis, G.E.: Hidden physics models: machine learning of nonlinear partial differential equations. J. Comput. Phys. **357**, 125–141 (2018)
21. Sirignano, J., Spiliopoulos, K.: DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018)
22. Yeung, T.S.A., Chung, E.T., See, S.: A deep learning based nonlinear upscaling method for transport equations. arXiv preprint arXiv:2007.03432 (2020)