Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2024-0040

Vol. **38**, No. 3, pp. 603-629
September 2025

Two-Scale Neural Networks for Partial Differential Equations with Small Parameters

Qiao Zhuang[1,†], Chris Ziyi Yao[2], Zhongqiang Zhang[1,*] and George Em Karniadakis[3]

[1] *Department of Mathematical Sciences, Worcester Polytechnic Institute, Worcester, MA 01609, USA.*
[2] *Department of Aeronautics, Imperial College London, London, SW7 2AZ, UK.*
[3] *Division of Applied Mathematics, Brown University, Providence, RI 02912, USA. and Pacific Northwest National Laboratory, P.O. Box 999, Richland, 99352, WA, USA.*

Received 27 February 2024; Accepted (in revised version) 8 October 2024

Abstract. We propose a two-scale neural network method for solving partial differential equations (PDEs) with small parameters using physics-informed neural networks (PINNs). We directly incorporate the small parameters into the architecture of neural networks. The proposed method enables solving PDEs with small parameters in a simple fashion, without adding Fourier features or other computationally taxing searches of truncation parameters. Various numerical examples demonstrate reasonable accuracy in capturing features of large derivatives in the solutions caused by small parameters.

AMS subject classifications: 65N35, 35B25

Key words: Two-scale neural networks, partial differential equations, small parameters, successive training.

1 Introduction

In this work, we consider physics-informed neural networks (PINNs) for the following equation

$$p\partial_t u - \epsilon L_2 u + L_0 u = f, \quad \mathbf{x} \in D \subset \mathbb{R}^d, \quad t \in (0,T], \tag{1.1}$$

[†]Current address: School of Science and Engineering, University of Missouri-Kansas City, Kansas City, MO 64110, USA.

[*]Corresponding author. *Email addresses:* qzhuang@umkc.edu (Q. Zhuang), chris.yao20@imperial.ac.uk (C. Z. Yao), zzhang7@wpi.edu (Z. Zhang) george_karniadakis@brown.edu (G. E. Karniadakis)

where some proper boundary conditions and initial conditions are imposed. Here $p=0$ or 1, $\epsilon>0$, D is the spatial domain, $d\in\mathbb{N}^+$ and $T>0$. Also, L_2u consists of the leading-order differential operator and L_0u consists of lower-order linear or nonlinear differential operators. For example, consider a singular perturbation problem where $p=0$, $L_2u=\operatorname{div}(a\nabla u)$ is second-order and $L_0u=b\cdot\nabla u+cu$ is first-order.

Small parameters in the equation often pose extra difficulties for numerical methods, see e.g., [29]. The difficulties come from one or more sharp transitions in regions of small volumes, which implies large first-order derivatives or even large high-order derivatives. When $p=0$, $\epsilon>0$ is very small, $L_2u=\Delta u$ and $L_0u=\mathbf{1}\cdot\nabla u$, then at least one boundary layer arises.

When deep feedforward neural networks are used, they are usually trained with stochastic gradient descent methods but do not resolve the issues above as they learn functions with low frequency and small first-order derivatives, see e.g., in [3,28,40].

1.1 Literature review

To deal with functions with high-frequency components such as in singular perturbation problems, at least four approaches have been proposed to address this issue:

- Adding features in the neural networks: Adding random Fourier features is probably the most non-intrusive approach. Adding $\cos(\omega_i^\top x)$,$\sin(\omega_i^\top x)$'s to deep neural networks as approximations of target functions or solutions. With an explicitly specified range for the random frequencies, one can learn a large class of functions. In [5,20,22,33], frequency ranges are scheduled to represent complicated solutions to partial differential equations. See also [21,41,44] for more elliptic type multi-scale PDEs. The Fourier feature networks are also used in [36,37], and see [35] for a review.

- Enhanced loss by adding first-order and higher-order derivatives: Another approach is to include the gradient information in the loss function, e.g., in [6,13,19]. This approach has been applied to solve PDEs, e.g., in [30,42].

- Adaptive weights: In the loss function, adaptive weights are assigned to have a better balance of each squared term in the least-squares formulation, e.g., self-adaptive [26], attention-based weights [2], binary weights [10]. See also [25,39,43].

- Resampling: Sampling points in the loss function can be made adaptive based on the residuals at sampling points during the training process such as in [8,9,23,38]. Also, in [27,31,32], the density function for sampling during the training is computed using the idea of importance sampling. The density function for re-sampling is approximated by Gaussian mixtures and formulated by finding the min-max of the loss function via fixed-point iterations.