INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 23, Number 2, Pages 163–183

doi: 10.4208/ijnam2026-1008

AN ASYMPTOTIC EXPANSION BASED DEEP NEURAL NETWORKS FOR SOLVING SINGULARLY PERTURBED PROBLEMS WITH EXPONENTIAL BOUNDARY LAYERS

SHUIQIAN XU[†], JIANGXING WANG[♮,*], YIDUO ZHANG[♮], AND XIAOFENG YANG[‡]

Abstract. Physics-informed neural networks (PINNs) have emerged as a powerful framework for solving partial differential equations (PDEs). However, their effectiveness deteriorates when applied to singularly perturbed problems, where solutions exhibit steep gradients and boundary layers confined to narrow regions features that standard PINNs architectures often fail to capture. To overcome these obstacles, we propose a novel, mesh-free deep neural networks (DNNs) method that incorporates asymptotic information through a systematic decomposition of the solution into smooth and sharp components. By capitalizing on the ability of DNNs to excel at approximating smooth functions, our approach constructs a series of moderately sized networks tailored to different elements of the solution. This strategy enables uniform approximation accuracy across a wide range of perturbation parameters, maintaining robustness and efficiency even in the extreme regime where the perturbation parameter is as small as 10^{-16}, near machine precision. Key advantages of the proposed method include its conceptual simplicity, full independence from mesh requirements, and ease of implementation. Extensive numerical experiments confirm that our approach delivers significantly improved accuracy and efficiency compared to standard PINNs, demonstrating its potential as a robust and versatile framework for tackling singularly perturbed problems.

Key words. Singular perturbed problem, Uniform convergence, Deep neural networks, Asymptotic expansion.

1. Introduction

Singularly perturbed problems arise frequently in a broad spectrum of disciplines, including fluid dynamics, optical control, financial mathematics, biology, chemistry, and engineering sciences. These problems are characterized by the presence of a small parameter (e.g., ϵ in (2)) within the governing equations, leading to solutions that exhibit thin boundary layers with rapid spatial variations. The presence of such layers poses substantial challenges to the design of effective numerical methods. Classical numerical schemes often fail to resolve these layers accurately when the perturbation parameter becomes sufficiently small, as they typically lack uniform convergence with respect to the parameter. As a result, the development of numerical methods that guarantee uniform convergence remains a fundamental and active task. Two widely adopted strategies for addressing these challenges are the fitted operator method and the fitted mesh method, both of which have inspired the development of a variety of uniformly convergent numerical schemes over the past decades; see, for instance, [2, 15, 26, 29, 28, 13, 30, 25].

Motivated by the limitations of traditional methods, recent years have witnessed growing interest in machine learning-based approaches–particularly deep neural networks (DNNs)–for solving PDEs. DNNs have achieved remarkable success in scientific computing, largely due to their universal approximation property. Unlike traditional numerical methods such as finite difference, finite element, spectral,

Received by the editors on July 2, 2025 and, accepted on September 1, 2025.
2000 *Mathematics Subject Classification.* 68T07, 65L11, 65C05.
*Corresponding author.

and finite volume methods, DNNs offer significant advantages when solving high-dimensional problems that suffer from the "curse of dimensionality", referring to the exponential growth in computational complexity as the number of dimensions increases [12]. Furthermore, DNNs have demonstrated notable capabilities in addressing low-dimensional problems with low regularity or sharp solutions. Widely adopted methods for solving PDEs with DNNs include physics-informed neural networks (PINNs), which minimize least-squares or variational loss functions [4, 19, 14]; the deep Ritz method (DRM), which employs energy functionals as loss functions [8, 27]; and the operator learning method, which seeks to learn solution operators directly [18], among others.

However, despite these successes, directly applying DNNs-based methods such as PINNs to singularly perturbed problems remains highly nontrivial. It is well-documented that neural networks struggle to accurately capture sharp variations in functions [23], making them ill-suited for resolving the thin boundary layers characteristic of such problems. Consequently, naive implementations of PINNs often suffer from large generalization errors in the singularly perturbed regime. To overcome these challenges, several enhancements have been proposed to improve the performance of PINNs on problems with sharp solution features. For instance, one common strategy is to increase the density of residual points in regions exhibiting rapid variation, thereby providing more training information where it is most needed [21, 22]. Another effective approach is adaptive sampling, which dynamically adjusts the distribution of training points based on error indicators or gradient magnitudes, so that computational effort is concentrated on regions with steep solution gradients [9, 20]. In addition, more tailored DNN-based approaches have been developed specifically for singular perturbation problems, including a boundary-layer PINNs based on singular perturbation theory [3], and an asymptotic parameter PINNs (PAPINNs), which first trains the network with large perturbation parameters to learn smooth solutions and then refines the network to resolve sharp features [5]. Other strategies combine deep operator networks with Shishkin mesh points for loss evaluation [7], or employ Galerkin neural network procedures to accurately capture boundary layer behaviors in stress functions [1].

In addition to these algorithmic improvements, a number of recent studies have systematically compared various PINNs-based methods for problems with sharp gradients and have introduced novel architectures that embed asymptotic knowledge directly into the learning process. For example, [6] presents a comparative study of several representative approaches, including Classic-PINNs, Deep-TFC, PIELM, and X-TFC. A novel semi-analytic SL-PINNs framework is proposed in [11], where explicit correctors capturing the singular behavior of stiff boundary-layer solutions are either embedded into the PINNs architecture or jointly trained with the network. The Component Fourier Neural Operator (ComFNO), an advanced operator learning method built upon the Fourier Neural Operator (FNO), further incorporates prior knowledge from asymptotic analysis to improve accuracy and generalization [17]. In addition, [10] develops a hybrid numerical approach for the singularly perturbed Robin parabolic convection-diffusion problem, combining a Shishkin mesh with central difference discretization in the inner region, an upwind midpoint scheme in the outer region, and the implicit Euler method for temporal discretization.

While the aforementioned methods have demonstrated success in various settings, they still face challenges in accurately capturing boundary layer behavior when the perturbation parameter becomes extremely small (e.g., $\epsilon = 10^{-16}$). In

such regimes, boundary layers become exceedingly thin, and conventional neural network architectures no matter how adaptive struggle to resolve these multiscale structures without excessive computational cost or overfitting. Moreover, many existing approaches require problem-specific tuning or complex sampling strategies to maintain stability and accuracy across different perturbation regimes. To address these limitations, in this paper, we propose a new construction of PINNs tailored specifically for singularly perturbed problems, with the goal of overcoming the bottlenecks associated with boundary layer resolution in traditional PINNs frameworks. Our approach is grounded in the asymptotic structure of the underlying equations: we first construct several sharp functions with thin layers by utilizing the asymptotic expansions of different singularly perturbed problems, and then combine these sharp functions with differential neural networks to form a deep neural network approximation for a particular singularly perturbed problem. The core idea of our method lies in the observation that DNNs are particularly effective at approximating smooth functions, while the asymptotic expansion of singularly perturbed problems naturally decomposes the solution into a smooth part and a sharp boundary layer part. Leveraging this structure, we construct a series of DNNs that utilize the asymptotic properties to accurately approximate each component, enabling efficient and robust resolution of the full solution, over a wide range of perturbation parameters.

To summarize, the proposed method enjoys several notable advantages:

- Significantly improved generalization ability compared with the traditional PINNs, particularly in resolving sharp boundary layers.
- No special sampling strategy is required; only randomly selected training points just as in standard PINNs are used during the learning process.
- Uniform convergence with respect to the perturbation parameter can be empirically observed.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of DNNs, PINNs, and the asymptotic expansions of singularly perturbed problems. In Section 3, we present the construction of our proposed method, asymptotic expansion-based physics-informed neural networks. Section 4 is devoted to numerical examples that demonstrate the accuracy and efficiency of the proposed scheme. Some concluding remarks are given in Section 5.

2. Preliminaries of DNNs, PINNs, and Singularly perturbed problems

In this section, we provide a brief introduction to the concepts of DNNs and PINNs for solving PDEs, and the singularly perturbed problems with asymptotic expansion properties.

2.1. Deep Neural Networks (DNNs). A fully connected neural networks is a sequential composition of affine transformations and nonlinear activation functions. Mathematically, an n-layer neural network $\mathcal{N}^n$ is defined as

- Input layer: $\mathcal{N}^0 = \boldsymbol{x}$,
- Hidden layers: $\mathcal{N}^l = \sigma(\mathbf{W}^l\mathcal{N}^{l-1} + \mathbf{b}^l), l = 1, 2, \cdots, n-1$,
- Output layer: $\mathcal{N}^n = \mathbf{W}^n\mathcal{N}^{n-1} + \mathbf{b}^n$,

where σ denotes the nonlinear activation function, $\mathbf{W}^l$ and $\mathbf{b}^l$ represent the weights and biases of the l-th layer, respectively. Commonly used activation functions include the sigmoid function $\sigma(t) = (1 + e^{-t})^{-1}$, the rectified linear unit (ReLU) $\sigma(t) = \max(0, t)$, and the hyperbolic tangent function $\tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}}$. For example, the output of a single hidden layer neural network with m neurons can be

defined as

$$u_h(\boldsymbol{x},\theta)=\sum_{j=1}^{m}c_j\sigma(W_j\cdot\boldsymbol{x}+b_j),\boldsymbol{x}\in\Omega,$$

where Ω denotes the input domain. Denote the set of all functions formulated by $u_h(x,\theta)$ as V_m^σ.

A key theoretical foundation for employing deep neural networks in function approximation tasks is their universal approximation capability [16]. Specifically, it has been proven that for any target function $f\in W^{s,p}(\Omega)(s>0,1\le p<\infty)$ and any given tolerance $\tau>0$, there exists a network of width m, and a function $\tilde{f}\in V_m^\sigma$, such that

$$\|f-\tilde{f}\|_{W^{s,p}}<\tau.$$

This result implies that deep neural networks, when equipped with suitable architecture and parameters, can approximate a wide class of functions including those with low regularity or sharp transitions arbitrarily well.

Fig. 1 provides a schematic illustration of a fully connected feedforward neural network, consisting of an input layer, multiple hidden layers, and an output layer. Each neuron in one layer is connected to every neuron in the next layer through a weighted linear transformation followed by a nonlinear activation. The input vector $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^T$ propagates forward through the network, ultimately producing the output vector $\boldsymbol{y}=(y_1,y_2,\cdots,y_m)^T$. This structure underlies the architecture used throughout this work.

2.2. Physics-informed neural networks (PINNs). Physics-informed neural networks have emerged as a powerful framework for solving a broad class of PDEs, particularly when analytical solutions are unavailable or traditional numerical methods become inefficient [24, 4]. In what follows, we briefly outline the general methodology of PINNs for solving PDEs.

To this end, consider the following prototypical PDE system:

$$\begin{aligned}&\mathcal{L}u(\boldsymbol{x})=0,\ \boldsymbol{x}\in\Omega;\\&\mathcal{B}u(\boldsymbol{x})=g(\boldsymbol{x}),\ \boldsymbol{x}\in\partial\Omega,\end{aligned}\tag{1}$$

where $\mathcal{L}$ denotes a differential operator, $\mathcal{B}$ represents a boundary operator, $\Omega\subset\mathbb{R}^d$ is a bounded domain with boundary $\partial\Omega$, and d denotes the dimension of Ω.

In the PINNs framework, the solution $u(\boldsymbol{x})$ is approximated by a neural network $u_h(\boldsymbol{x},\theta)$, where θ denotes the set of trainable parameters (weights and biases).

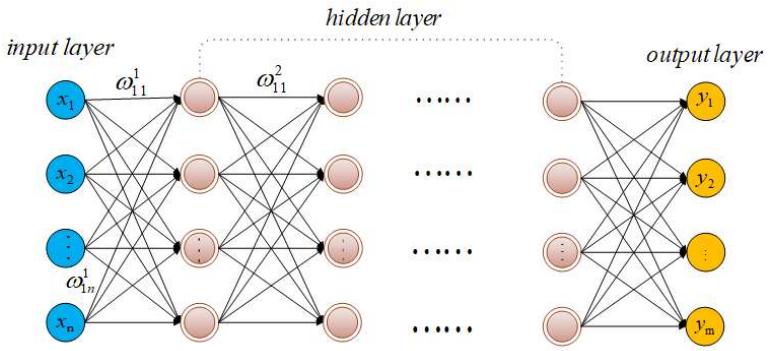


FIGURE 1. Multi-layer neural network structure.

These parameters are obtained by minimizing a composite loss function consisting of two terms: one enforcing the PDEs residual in the domain and the other enforcing the boundary conditions. Specifically, the total loss function is given by

$$\text{Loss} = \text{Loss}_{\text{PDE}} + \text{Loss}_{\text{BC}},$$

where

$$\text{Loss}_{\text{PDE}} = \left\|\mathcal{L}u_h(\boldsymbol{x}_i^{\Omega}, \theta)\right\|_2^2 = \frac{1}{N}\sum_{i=1}^{N} |\mathcal{L}u_h(\boldsymbol{x}_i^{\Omega}, \theta)|^2,$$

$$\text{Loss}_{\text{BC}} = \left\|\mathcal{L}u_h(\boldsymbol{x}_i^{BC}, \theta) - g(\boldsymbol{x}_i^{BC})\right\|_2^2 = \frac{1}{M}\sum_{i=1}^{M} |\mathcal{L}u_h(\boldsymbol{x}_i^{BC}, \theta) - g(\boldsymbol{x}_i^{BC})|^2.$$

Here, $\{\boldsymbol{x}_i^{\Omega}\}_{i=1}^{N}$ denotes the residual points sampled from the interior of the domain Ω, and $\{\boldsymbol{x}_i^{BC}\}_{i=1}^{M}$ denotes the residual points sampled from the boundary $\partial\Omega$. The optimization problem is then solved using gradient-based algorithms such as Adam or L-BFGS to determine the optimal parameters θ, thereby yielding the approximate solution u_h, see [24].

2.3. Singularly perturbed problems. In this subsection, we introduce singularly perturbed problems defined on a bounded domain $\Omega \subset \mathbb{R}^d$ with $d = 1, 2$. Let $0 < \epsilon \ll 1$ denotes the perturbation (or diffusion) parameter. The 1D singularly perturbed problem on $\Omega = [a, b]$ takes the form:

$$\begin{aligned} &-\epsilon u''(x) + \beta(x)u'(x) + c(x)u(x) = f(x),\ x \in (a, b), \\ &u(a) = u_0, \quad u(b) = u_1. \end{aligned} \tag{2}$$

where f is a source term, and $\beta(x)$, $c(x)$ are given coefficient functions. We remark that the domain $[a, b]$ is often rescaled to $[0, 1]$, see [25]; however, such a rescaling does not alter the form of the differential equation.

Under appropriate smoothness assumptions, the solution u to (2) admits the following asymptotic expansion:

Lemma 2.1. [25] *Assume that $\Omega = [0, 1]$, and that the coefficient functions and the right-hand side of the boundary problem* (2) *are sufficiently smooth. Suppose that $\beta(x) > \beta_0 > 0$ on $[0, 1]$, then its solution u admits a matched asymptotic expansion of the form*

$$u_{as}(x) = \sum_{n=0}^{m} \epsilon^n u_n(x) + \sum_{n=0}^{m} \epsilon^n v_n(\frac{1-x}{\epsilon}), \tag{3}$$

such that, for any sufficient small fixed constant ϵ_0, there holds

$$|u(x) - u_{as}(x)| \le C\epsilon^{m+1} \ \text{ for } x \in [0, 1] \text{ and } \epsilon \le \epsilon_0,$$

where the constant C is independent of x and ϵ and $u_n(x), v_n(x)$ denotes some smooth functions.

We now turn to the singularly perturbed problems in higher dimensions. Consider the following PDE on $\Omega = [a, b]^d \subset \mathbb{R}^d, d = 2, 3$:

$$\begin{aligned} &-\epsilon\Delta u(\boldsymbol{x}) + \beta(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x}) + c(\boldsymbol{x})u(\boldsymbol{x}) = f(\boldsymbol{x}), \\ &u(\boldsymbol{x})|_{\partial\Omega} = g(\boldsymbol{x}), \end{aligned} \tag{4}$$

with $0 < \epsilon \ll 1$. The solution u to (4) admits the following asymptotic expansion:

Lemma 2.2. [25] *Assume that $\Omega = [0,1]^2$, and the coefficient and the right-hand side of the boundary problem* (4) *are sufficiently smooth, $c(x) > 0$, and $\beta(\boldsymbol{x}) = (\beta_1, \beta_2)$ with $\beta_1 > 0, \beta_2 > 0$. Then the solution $u(x)$ of* (4) *has the following matched asymptotic expansion:*

$$\begin{aligned} u_{as}(\boldsymbol{x}) =& u_0(\boldsymbol{x}) - u_0(1,x_2)\exp\left(-\beta_1(1,x_2)\frac{1-x_1}{\epsilon}\right) \\ &- u_0(x_1,1)\exp\left(-\beta_2(x_1,1)\frac{1-x_2}{\epsilon}\right), \\ &+ u_0(1,1)\exp\left(-\beta_1(1,1)\frac{1-x_1}{\epsilon}\right)\exp\left(-\beta_2(1,1)\frac{1-x_2}{\epsilon}\right). \end{aligned} \tag{5}$$

Moreover, we have the following estimates:

- *If $u_0 \in C^2(\Omega) \cap C(\bar{\Omega})$, then*
$$\|u - u_{as}\|_\infty \le C\epsilon,$$
where $\|u\|_\infty$ denotes the maximum norm of u on Ω;
- *If $u_0 \in C^2(\Omega) \cap C(\bar{\Omega})$ is not satisfied, then*
$$\|u - u_{as}\|_\epsilon \le C\epsilon^{\frac{1}{2}},$$
where $\|u\|_\epsilon^2 = \epsilon|u|_{H^1}^2 + \|u\|_{L^2}$.

3. The proposed methodology: AE-PINNs

It is well recognized that approximating functions with sharp boundary layers, which typically arise when the perturbation parameter is extremely small, remains a fundamental challenge for standard neural networks [23]. In this section, we propose a novel deep learning framework asymptotic expansion-based physics-informed neural networks (AE-PINNs, for short) for solving singularly perturbed problems of the form (2) and (4). The core idea is to exploit the structural insights provided by asymptotic expansions to guide the construction and training of the neural network, thereby achieving accurate and robust resolution of solutions with sharp boundary layers. For clarity of exposition, we first present our approach in the 1D setting (2), followed by its extension to the 2D case (4).

3.1. AE-PINNs for 1D. We consider the 1D singularly perturbed model problem given in (2), which, according to Lemma 2.1, for the rescaled domain $[0,1]$, a boundary layer typically forms near $x = 1$ when $\beta(x) > 0$. However, as discussed in [25], when $\beta(x) = 0, c > c_0 > 0$, boundary layers may develop at both endpoints $x = 0$ and $x = 1$. This indicates that, in general, singularly perturbed problems may exhibit boundary layers at both $x = a$ and $x = b$. Motivated by the identified boundary layer behavior and the asymptotic expansion form given in (3), we formulate the AE-PINNs method as follows:

$$u_h(x;\theta) = NN_1(x,\theta_1) + NN_2(x,\theta_2)e^{-\frac{x-a}{\epsilon}} + NN_3(x,\theta_3)e^{-\frac{b-x}{\epsilon}}, \tag{6}$$

where $\theta = (\theta_1, \theta_2, \theta_3)$, and $NN_i(x,\theta_i)$ $(i = 1,2,3)$ denote three distinct neural networks. In this formulation: $NN_1(x,\theta_1)$ is designed to approximate the smooth interior part of the solution; $NN_2(x,\theta_2)$ is associated with the left boundary layer near $x = a$ through the exponential decay term $e^{-\frac{x-a}{\epsilon}}$; $NN_3(x,\theta_3)$ captures the right boundary layer near $x = b$ via the exponential term $e^{-\frac{b-x}{\epsilon}}$. This additive structure effectively decomposes the solution into its smooth and singular components, reflecting the typical asymptotic form of singularly perturbed problems and

allowing each neural subnetwork to specialize in learning a specific feature of the solution.

To compute the approximation $u_h(x;\theta)$, we minimize the residual of the governing equation using the following least-squares loss function:

$$\text{Loss}(\theta) = \text{Loss}_{eq}(\theta) + \rho \text{Loss}_{bc}(\theta), \tag{7}$$

where ρ is a penalty parameter that balances the interior and boundary contributions. Specifically,

$$\text{Loss}_{eq}(\theta) = \sum_{m=1}^{N} |f(x_m) + \epsilon u_h''(x_m,\theta) - \beta(x_m)u_h'(x_m,\theta) - c(x_m)u_h(x_m,\theta)|^2,$$
$$\text{Loss}_{bc}(\theta) = |u_h(a,\theta) - u_0|^2 + |u_h(b,\theta) - u_1|^2,$$

where $\{x_m\}_{m=1}^N$ are residual points sampled from the domain $[a,b]$.

3.2. AE-PINNs for 2D. We now extend the proposed AE-PINNs framework to 2D singularly perturbed problems governed by the model equation (4). In analogy to the one-dimensional case (6), the 2D neural network approximation must account for all potential boundary and corner layers. Guided by the structural form of the asymptotic expansion (5), we construct the neural network solution as a linear combination of nine neural subnetworks, each modulated by an appropriate exponential boundary or corner layer correction term.

More specifically, the approximation takes the form

$$\begin{aligned} u_h(\boldsymbol{x};\theta) = {} & NN_1(\boldsymbol{x},\theta_1) + NN_2(\boldsymbol{x},\theta_2)e^{-\frac{b-x_1}{\epsilon}} + NN_3(\boldsymbol{x},\theta_3)e^{-\frac{b-x_2}{\epsilon}} \\ & + NN_4(\boldsymbol{x},\theta_4)e^{-\frac{x_1-a}{\epsilon}} + NN_5(\boldsymbol{x},\theta_5)e^{-\frac{x_2-a}{\epsilon}} \\ & + NN_6(\boldsymbol{x},\theta_6)e^{-\frac{x_1-a}{\epsilon}}e^{-\frac{x_2-a}{\epsilon}} + NN_7(\boldsymbol{x},\theta_7)e^{-\frac{x_1-a}{\epsilon}}e^{-\frac{b-x_2}{\epsilon}} \\ & + NN_8(\boldsymbol{x},\theta_8)e^{-\frac{b-x_1}{\epsilon}}e^{-\frac{x_2-a}{\epsilon}} + NN_9(\boldsymbol{x},\theta_9)e^{-\frac{b-x_1}{\epsilon}}e^{-\frac{b-x_2}{\epsilon}} \end{aligned} \tag{8}$$

where $NN_i(\boldsymbol{x},\theta_i)$, $i = 1,\ldots,9$, denote nine independent neural networks, and $\theta = (\theta_1,\ldots,\theta_9)$ represents the total set of trainable parameters.

To determine the network parameters θ, we minimize a composite loss function that includes contributions from both the residual of the governing PDE and the mismatch with the boundary conditions. The total loss is defined as:

$$\text{Loss}(\theta) = \text{Loss}_{eq}(\theta) + \rho \text{Loss}_{bc}(\theta), \tag{9}$$

where ρ is a penalty parameter that balances the interior and boundary contributions, and

$$\text{Loss}_{eq}(\theta) = \sum_{m=1}^{N} |f(\boldsymbol{x}_m) + \epsilon \Delta u_h(\boldsymbol{x}_m,\theta) - \beta(\boldsymbol{x}_m)\cdot\nabla u_h(\boldsymbol{x}_m,\theta) - c(\boldsymbol{x}_m)u_h(\boldsymbol{x}_m,\theta)|^2,$$
$$\text{Loss}_{bc}(\theta) = \sum_{m=1}^{M} |u_h(\boldsymbol{x}_m,\theta) - g(\boldsymbol{x}_m)|^2,$$

where $\{\mathbf{x}_m\}_{m=1}^N \subset \Omega$ are the interior points used to evaluate the PDE residual, and $\{\mathbf{x}_m\}_{m=1}^M \subset \partial\Omega$ are the boundary points used to enforce the Dirichlet boundary condition.

Remark 3.1. *As shown in* [25]*, the corner layer correction term,*

$$u_0(1,1)\exp\Big(-\beta_1(1,1)\frac{1-x_1}{\epsilon}\Big)\exp\Big(-\beta_2(1,1)\frac{1-x_2}{\epsilon}\Big), \tag{10}$$

plays a critical role in the asymptotic error estimate. Therefore, it is essential to include all four corner-layer terms in the network representation, $e^{-\frac{x_1-a}{\epsilon}}e^{-\frac{x_2-a}{\epsilon}}$, $e^{-\frac{x_1-a}{\epsilon}}e^{-\frac{b-x_2}{\epsilon}}$, $e^{-\frac{b-x_1}{\epsilon}}e^{-\frac{x_2-a}{\epsilon}}$ *and* $e^{-\frac{b-x_1}{\epsilon}}e^{-\frac{b-x_2}{\epsilon}}$ *in* (8).

However, when the location of the boundary and corner layers is known a priori for instance, based on the sign and directionality of the convection coefficient certain exponential correction terms can be omitted without compromising accuracy. For example, if $\beta = (1,1), \Omega = [0,1]^2$*, boundary layers are expected only along the right and top boundaries, with a corner layer at the top-right vertex. In this case, the neural network approximation can be simplified as:*

$$u_h(\boldsymbol{x};\theta) = NN_1(\boldsymbol{x},\theta_1) + NN_2(\boldsymbol{x},\theta_2)e^{-\frac{1-x_1}{\epsilon}} + NN_3(\boldsymbol{x},\theta_3)e^{-\frac{1-x_2}{\epsilon}} + NN_4(\boldsymbol{x},\theta_4)e^{-\frac{1-x_1}{\epsilon}}e^{-\frac{1-x_2}{\epsilon}}.$$

Remark 3.2. *Although the theory and error analysis of asymptotic expansions for 3D singularly perturbed problems remain relatively underdeveloped, it is nonetheless feasible to construct 3D AE-PINNs by extending the 1D and 2D formulations given in* (6) *and* (8)*. In particular, the sharp boundary layer structures in 3D can still be captured by embedding appropriate anisotropic exponential terms aligned with the layer directions. Moreover, the modular nature of the AE-PINNs architecture makes it naturally extensible to higher dimensions without requiring significant modifications to the overall framework.*

4. Numerical Examples

In this section, we present a series of numerical experiments to assess the accuracy, robustness, and generalization performance of the proposed AE-PINNs framework on a variety of singularly perturbed problems in both 1D and 2D. These problems are carefully selected to include sharp boundary layers, multiscale structures, and variable coefficients, which pose significant challenges for standard PINNs approaches. To highlight the superiority of the proposed method, we also conduct direct comparisons with standard PINNs formulations, which often fail to resolve boundary layers when the perturbation parameter becomes small.

TABLE 1. Maxinum error for Example 1.

ϵ	10^{-1}	10^{-3}	10^{-5}	10^{-10}	10^{-15}
Maximum error	1.2734e−3	6.4863e−4	7.2827e−4	7.2848e−4	7.2848e−4

Example 1: Convection-diffusion problem in 1D. We consider the following 1D convection-diffusion problem:

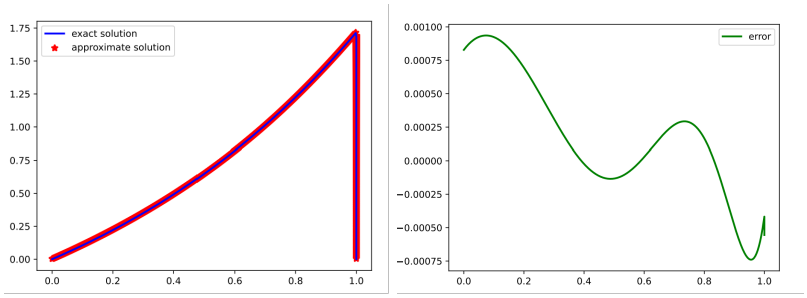
$$\begin{aligned} -\epsilon u'' + u' &= f(x), \quad x \in (0,1), \\ u(0) = 0,\ u(1) &= 0, \end{aligned} \tag{11}$$

where the forcing term is chosen as $f(x) = e^x$. The exact solution is given by

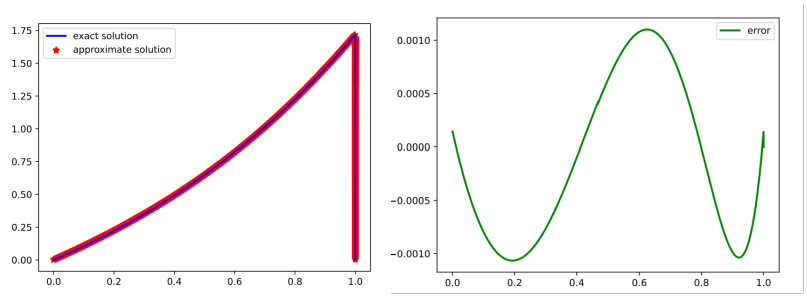
$$u(x) = \frac{e^x(1-e^{-\frac{1}{\epsilon}}) + e^{1-\frac{1}{\epsilon}} - 1 + (1-e)e^{\frac{x-1}{\epsilon}}}{(1-\epsilon)(1-e^{-\frac{1}{\epsilon}})}.$$

This solution exhibits a boundary layer near $x = 1$, and conventional numerical methods often require specialized treatments to accurately capture this behavior. To construct the AE-PINNs approximation, we exploit the known location of the boundary layer and propose the following ansatz:

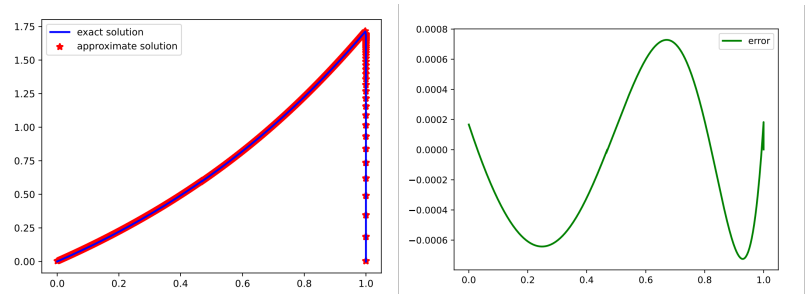
$$\tilde{u}(x;\theta) = NN_0(x,\theta_1) + NN_1(x,\theta_2)e^{\frac{x-1}{\epsilon}}, \tag{12}$$



(a) $\epsilon = 10^{-6}$.



(b) $\epsilon = 10^{-10}$.

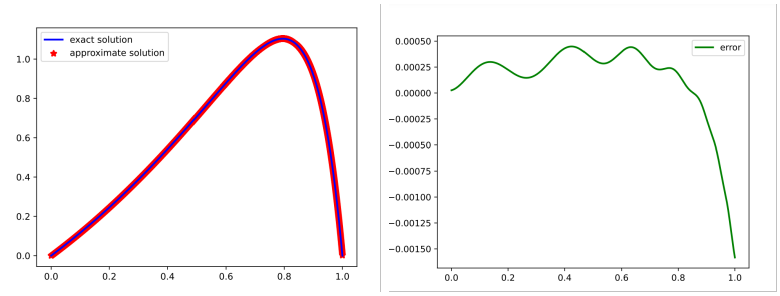


(c) $\epsilon = 10^{-15}$.

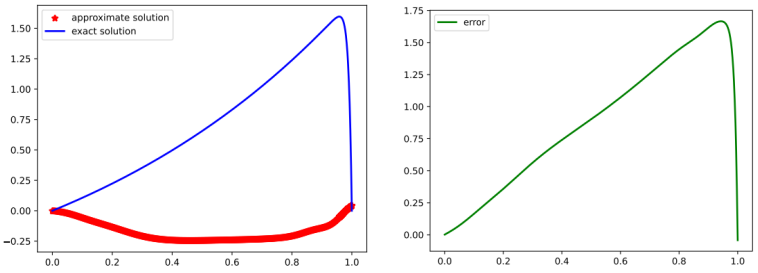
FIGURE 2. AE-PINNs results for Example 1. In each subfigure, the left panel shows the exact and approximate solutions; while the right panel presents the corresponding error distribution. Each subfigure corresponds to a different perturbation parameter ϵ.

where $\theta = (\theta_1, \theta_2)$, $NN_0(x, \theta_1)$ and $NN_1(x, \theta_2)$ are two independent neural networks with 8 hidden layers and 80 neurons per layer. During training, we randomly select 1600 points in $(0, 1)$ for each epoch and run a total of 800 epochs. To evaluate accuracy, we partition the interval into $[0, \tau]$ and $[\tau, 1]$ with $\tau = \max\{\frac{1}{2}, 1 - 10\epsilon\}$, and uniformly sample 1000 points in each subinterval.

Table 1 reports the maximum error across these sampled points for various values of ϵ , showing that the proposed AE-PINNs method achieves uniform accuracy even when $\epsilon = 10^{-15}$, which is close to machine precision. Fig. 2 visualizes the exact and numerical solutions, as well as their differences, for different values of ϵ. The results confirm that the AE-PINNs effectively resolves the boundary layer structure, even for extremely small perturbation parameters.



(a) $\epsilon = 0.1$.



(b) $\epsilon = 0.01$.

FIGURE 3. Standard PINN results for Example 1. In each subfigure, the left panel shows the exact and approximate solutions; while the right panel presents the corresponding error distribution. Each subfigure corresponds to a different perturbation parameter ϵ.

For comparison, we also apply the standard PINNs method to solve the same problem. The corresponding loss function is defined as

$$\text{Loss}(\theta) = \text{Loss}_{eq}(\theta) + \rho \text{Loss}_{bc}(\theta),$$

where

$$\text{Loss}_{eq}(\theta) = \sum_{m=1}^{M} |f(x_m) + \epsilon NN^{''}(x_m, \theta) - NN'(x_m, \theta)|^2,$$
$$\text{Loss}_{bc}(\theta) = |NN(0, \theta) - u_0|^2 + |NN(1, \theta) - u_1|^2,$$

with M denoting the number of the collocation points in the interval $[a, b]$ and $NN'(x_m, \theta)$, $NN^{''}(x_m, \theta)$ denote the first and second order derivatives of $NN(x_m, \theta)$ with respect to θ, respectively. In the implementation, we set $NN(x, \theta)$ as a fully connected neural network with 8 hidden layers and 80 neurons per layer for $\epsilon = 0.1$, and 12 hidden layers with 120 neurons per layer for $\epsilon = 0.01$. For both cases, we randomly select 1600 training points per epoch and run the training process for $10,000$ epochs. The numerical and exact solutions for $\epsilon = 0.1$ and $\epsilon = 0.01$ are shown in Fig. 3.

As shown in Fig. 3, the standard PINNs method performs reasonably well when the perturbation parameter is moderately large (e.g., $\epsilon = 0.1$), yielding accurate approximations with small residual errors. However, as ϵ decreases (e.g., $\epsilon = 0.01$), the method fails to resolve the boundary layer near $x = 1$, leading to significant

deviations from the exact profile. The corresponding error grows dramatically and exhibits a steep gradient near the boundary. This clearly highlights the advantage of our proposed AE-PINNs framework, which incorporates asymptotic layer information into the neural network architecture. By doing so, AE-PINNs are able to accurately capture the sharp gradients and boundary layer structure across a wide range of ϵ, maintaining both stability and accuracy in the singular perturbation regime.

TABLE 2. Maxinum error for Example 2.

ϵ	10^{-1}	10^{-3}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
Maximum error	3.5360e−4	7.9269e−4	1.4450e−3	7.1608e−4	5.3310e−4	5.3105e−4

Example 2: Reaction-diffusion problem in 1D. In this example, We consider the following singularly perturbed reaction-diffusion problem:

$$\begin{aligned} &-\epsilon u'' + u = 2\sqrt{\epsilon}e^{-\frac{x}{\sqrt{\epsilon}}} + 2\sqrt{\epsilon}e^{\frac{x-1}{\sqrt{\epsilon}}},\ 0 < x < 1,\\ &u(0) = -1,\ u(1) = -1. \end{aligned}$$

The exact solution to this problem is known to be:

$$u = (x-1)e^{-\frac{x}{\sqrt{\epsilon}}} - xe^{\frac{x-1}{\sqrt{\epsilon}}}.$$

This solution exhibits boundary layers near both endpoints, $x = 0$ and $x = 1$.

To construct the AE-PINNs approximation, we design the following ansatz:

$$\tilde{u}(x;\theta) = NN_1(x,\theta_1) + NN_2(x,\theta_2)\cdot e^{-\frac{\sqrt{c}}{\sqrt{\epsilon}}x} + NN_3(x,\theta_3)\cdot e^{\frac{\sqrt{c}}{\sqrt{\epsilon}}(x-1)},$$

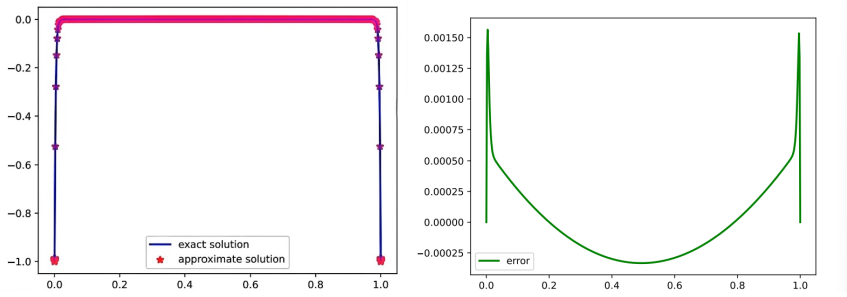
where $\theta = (\theta_1, \theta_2, \theta_3)$. To demonstrate the accuracy and efficiency of the proposed AE-PINNs method, we set each $NN_i(x,\theta_i)$ to be a fully connected neural network with 8 hidden layers and 80 neurons per layer. During training, we randomly sample 5000 collocation points per epoch and train the model for 800 epochs. We define $\tau = \max\{\frac{1}{2}, 1-10\epsilon\}$, and partition the interval $[0,1]$ into three subdomains $[0, 1-\tau]$, $[1-\tau, \tau]$ and $[\tau, 1]$. For evaluation, we adopt uniform sampling with 500 points per subinterval. The maximum error in the numerical solution for various ϵ values is reported in Table 2. Fig. 4 show the numerical solution, exact solution, and their difference for several values of ϵ, demonstrating excellent agreement between the numerical and exact solutions even for extremely small perturbation parameters.

TABLE 3. Maxinum error for the convection-dominated case in Example 3.

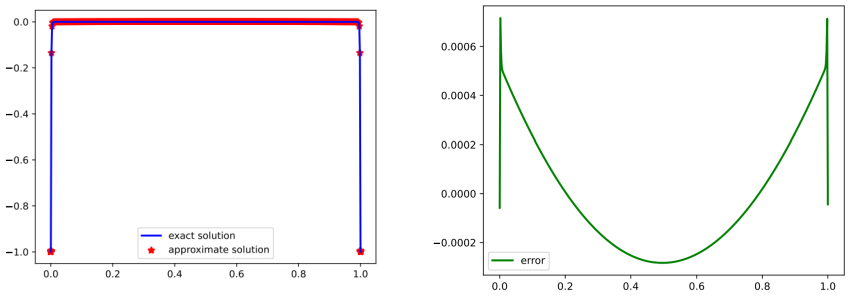
ϵ	10^{-1}	10^{-3}	10^{-6}	10^{-10}	10^{-15}
Maximum error	9.3801e−3	6.3431e−5	4.8734e−5	4.8733e−5	4.8734e−5

Example 3: Reaction-convection-diffusion problem with variable coefficients in 1D. In this example, we consider the following variable-coefficient singularly perturbed problem:

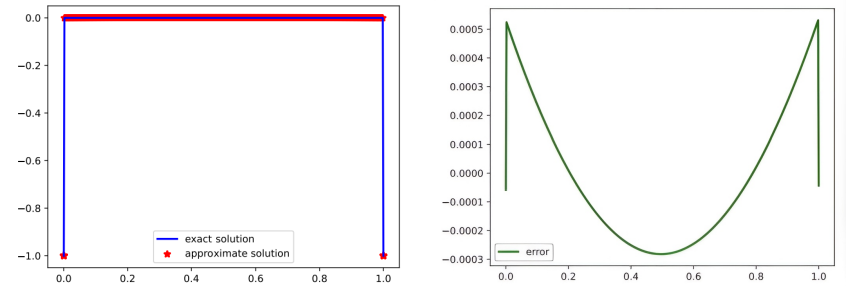
$$\begin{aligned} &-\epsilon u''(x) + b(x)u'(x) + c(x)u(x) = f(x),\ x \in (0,1),\\ &u(0) = u_0, u(1) = u_1. \end{aligned}$$



(a) $\epsilon = 10^{-5}$.



(b) $\epsilon = 10^{-6}$.



(c) $\epsilon = 10^{-8}$.

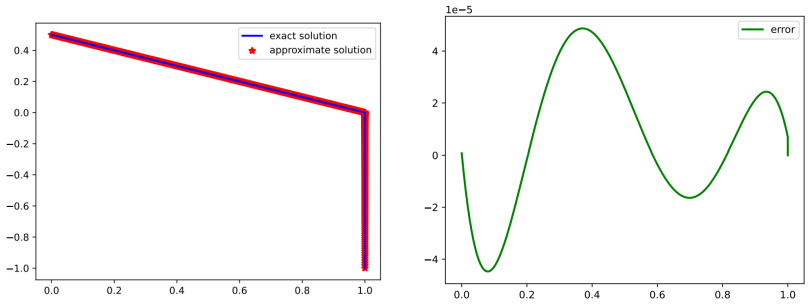
FIGURE 4. AE-PINN results for Example 2. In each subfigure, the left panel shows the exact and approximate solutions; while the right panel presents the corresponding error distribution. Each subfigure corresponds to a different perturbation parameter ϵ.

We first consider the convection-dominated case, where

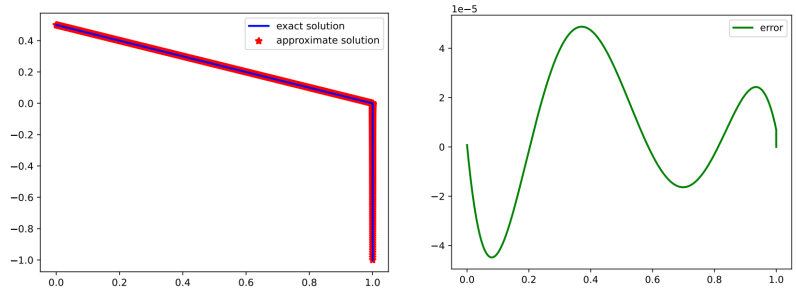
$$\begin{cases} b(x) = 1 + x^2, c(x) = 0, u_0 = \dfrac{1}{2}, u_1 = 0, \\ f(x) = \dfrac{2e^{\frac{2}{\epsilon}(x-1)}(x^2-1)}{\epsilon e^{-\frac{2}{\epsilon}} - 1} - \dfrac{1+x^2}{2}. \end{cases} \tag{13}$$

The exact solution to this problem is given by

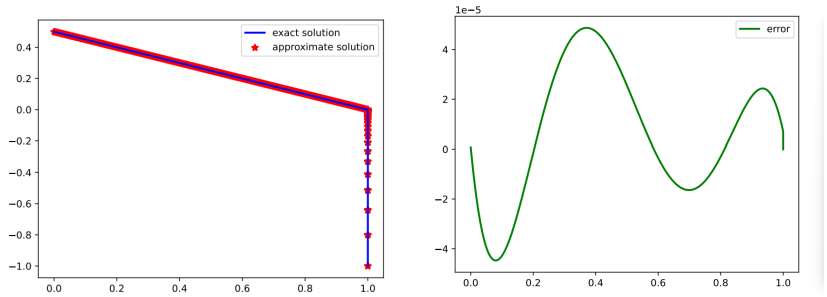
$$u = \frac{e^{\frac{2}{\epsilon}(x-1)} - 1}{e^{-\frac{2}{\epsilon}} - 1} - \frac{1+x}{2}.$$



(a) $\epsilon = 10^{-6}$.



(b) $\epsilon = 10^{-10}$.



(c) $\varepsilon = 10^{-15}$.

FIGURE 5. AE-PINN results for the convection-dominated case in Example 3. In each subfigure, the left panel shows the exact and approximate solutions; while the right panel presents the corresponding error distribution. Each subfigure corresponds to a different perturbation parameter ϵ.

It is evident that a boundary layer appears near $x = 1$. To construct the AE-PINNs approximation, we propose the following ansatz:

$$\tilde{u}(x;\theta) = NN_1(x,\theta_1) + NN_2(x,\theta_2)e^{\frac{b(1)(x-1)}{\epsilon}}, \text{ where } \theta = (\theta_1,\theta_2). \tag{14}$$

Here, $NN_1(x,\theta_1)$ and $NN_2(x,\theta_2)$ are independent neural networks, each with 8 hidden layers and 80 neurons per layer. During training, 1600 points are randomly selected in each epoch, and the training process runs for 800 epochs. To evaluate accuracy, we define $\tau = \max\{\frac{1}{2}, 1 - 10\epsilon\}$ and divide the interval $[0,1]$ into two subintervals: $[0,\tau]$ and $[\tau,1]$. We sample 500 uniform points from each subinterval.

The maximum errors for different values of ϵ are reported in Table 3, which demonstrate that the AE-PINNs method achieves uniform convergence. Fig. 5 compares the exact and numerical solutions, along with their differences, indicating excellent agreement even for very small ϵ.

TABLE 4. Maxinum error for the reaction-dominated case in Example 3.

ϵ	10^{-1}	10^{-3}	10^{-6}	10^{-7}	10^{-8}
Maximum error	4.4561e−4	1.1218e−2	7.5434e−2	6.1396e−3	6.0226e−3

We now turn to the reaction-dominated case:

$$\begin{cases} b(x) = 0, c(x) = 1 + x^2, u_0 = 0, u_1 = 0, \\ f(x) = (1+x)^2 + e^{-\frac{x}{\sqrt{\epsilon}}}(2\sqrt{\epsilon} + x(x^2+x-2)) + e^{\frac{2(x-1)}{\sqrt{\epsilon}}}(4\sqrt{\epsilon} - x(x^2+2x-3)). \end{cases} \tag{15}$$

The exact solution to this problem is given by

$$u = 1 + (x-1)e^{-\frac{x}{\sqrt{\epsilon}}} + xe^{\frac{2(x-1)}{\sqrt{\epsilon}}}.$$

This solution contains boundary layers near both $x = 0$ and $x = 1$. Hence, the AE-PINN ansatz is constructed as follows:

$$\tilde{u}(x;\theta) = NN_1(x,\theta_1) + NN_2(x,\theta_2) \cdot e^{-\frac{\sqrt{c}(1)}{\sqrt{\epsilon}}x} + NN_3(x,\theta_3) \cdot e^{\frac{\sqrt{c}(1)}{\sqrt{\epsilon}}(x-1)},$$

where $\theta = (\theta_1, \theta_2, \theta_3)$. Each $NN_i(x,\theta_i)(i = 1,2,3)$, is modeled by a neural network with 8 hidden layers and 80 neurons per layer. For training, we again use 1600 collocation points per epoch and run for 800 epochs. Let $\tau = \max\{\frac{1}{2}, 1-10\epsilon\}$. The interval $[0,1]$ is divided into three subintervals: $[0,1-\tau], [1-\tau,\tau]$ and $[\tau,1]$. We use 1500 uniform points in each subinterval. The maximum errors are shown in Table 4. The results indicate that the AE-PINNs approximation remains accurate even for very small values of ϵ. Fig. 6 further illustrate the excellent agreement between the numerical and exact solutions, confirming the robustness of the method.

TABLE 5. Maxinum error for Example 4.

ϵ	10^{-1}	10^{-7}	10^{-9}	10^{-14}	10^{-16}
Maxinum error	3.5405e−3	5.0368e−3	3.9208e−3	4.3344e−3	4.9273e−3

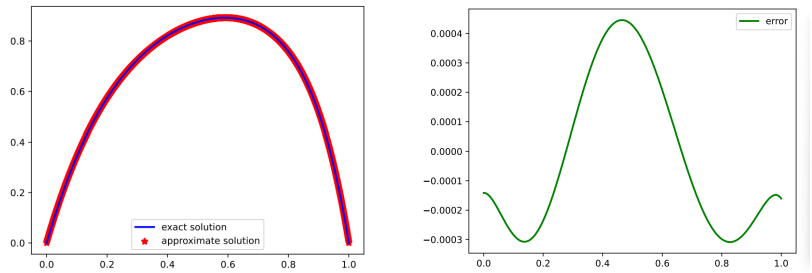
Example 4: Reaction-convection-diffusion problem in 2D. In this example, we consider the following reaction-convection-diffusion problem in 2D:

$$\begin{aligned} &-\epsilon\Delta u + \beta_1 \cdot u_x + \beta_2 \cdot u_y = f(x,y), \\ &u|_{\partial\Omega} = g, \end{aligned} \tag{16}$$

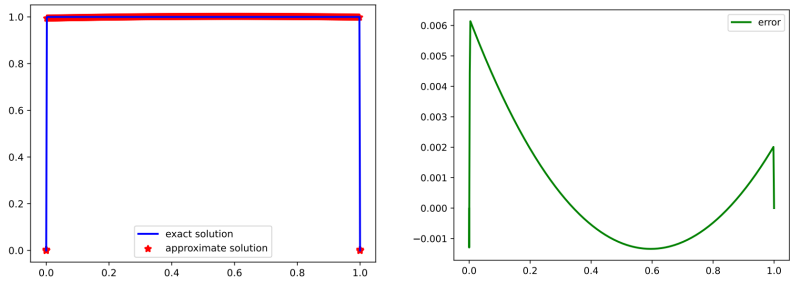
where $\Omega = [0,1]^2$ and we set $\beta_1 = 1, \beta_2 = 1$. The source term $f(x,y)$ is chosen such that the exact solution is given by

$$u(x,y) = \sin\frac{\pi x}{2} + \sin\frac{\pi y}{2}(1 - \sin\frac{\pi x}{2}) + \frac{e^{-\frac{1}{\epsilon}} - e^{-\frac{(1-x)(1-y)}{\epsilon}}}{1 - e^{-\frac{1}{\epsilon}}}.$$

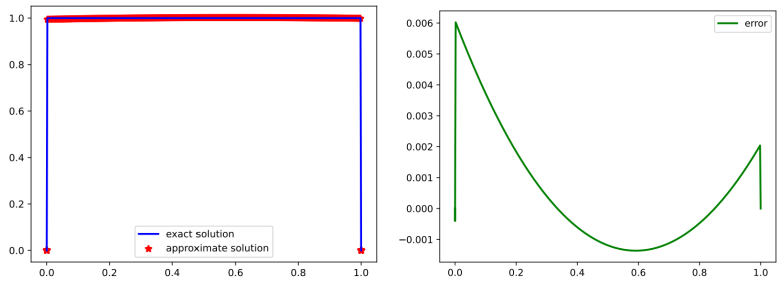
This solution exhibits boundary layers near the outflow corner $(x,y) = (1,1)$, with layer behavior becoming sharper as ϵ approaches 0.



(a) $\epsilon = 10^{-1}$.



(b) $\epsilon = 10^{-7}$.



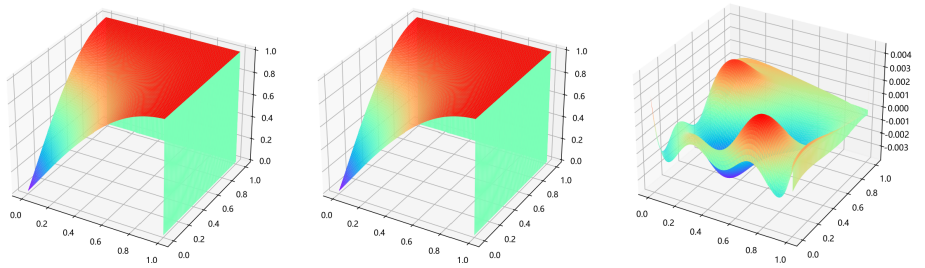
(c) $\epsilon = 10^{-8}$.

FIGURE 6. AE-PINN results for the reaction-dominated case in Example 3. In each subfigure, the left panel shows the exact and approximate solutions; while the right panel presents the corresponding error distribution. Each subfigure corresponds to a different perturbation parameter ϵ.

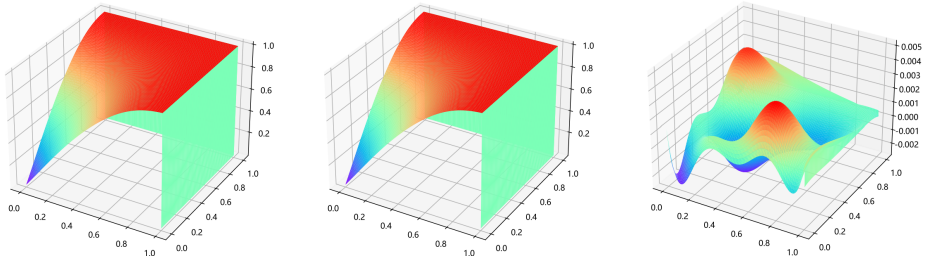
To approximate the solution, we adopt the AE-PINNs ansatz:

$$\begin{aligned}\tilde{u}(x,y;\theta) = {} & NN_1(x,y,\theta_1) + NN_2(x,y,\theta_2)\cdot e^{\frac{x-1}{\epsilon}} + NN_3(x,y,\theta_3)\cdot e^{\frac{y-1}{\epsilon}} \\ & + NN_4(x,y,\theta_4)\cdot e^{\frac{x-1}{\epsilon}} e^{\frac{y-1}{\epsilon}}, where\ \theta = (\theta_1,\theta_2,\theta_3,\theta 4).\end{aligned} \tag{17}$$

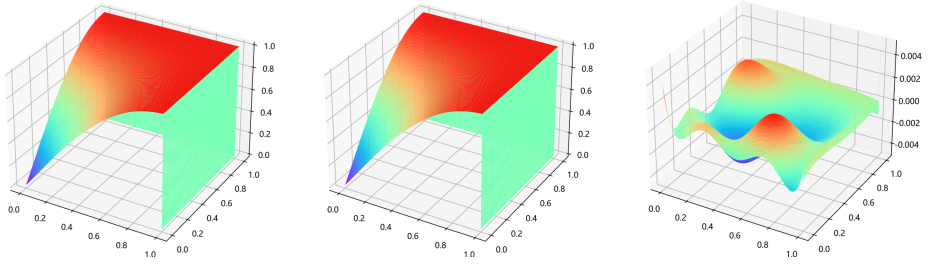
Each $NN_i(x,y,\theta_i)$ $(i = 1,2,3,4)$ is modeled by a neural network with 6 hidden layers and 60 neurons per layer. During training, $40,000$ collocation points in the interior and $4,000$ points on the boundary are randomly selected in each epoch, and the total number of epochs is set to $1,000$. To assess the accuracy of the proposed AE-PINN framework, we define $\tau = \max\{\frac{1}{2}, 1-10\epsilon\}$ and divide the domain into



(a) $\varepsilon = 10^{-9}$.



(b) $\varepsilon = 10^{-14}$.



(c) $\varepsilon = 10^{-16}$.

FIGURE 7. AE-PINN results for Example 4. In each subfigure, the left and middle panels show the exact and approximate solutions; while the right panel presents the corresponding error distribution. Each subfigure corresponds to a different perturbation parameter ϵ.

four subregions:

$$[0,\tau]\times[0,\tau],[0,\tau]\times[\tau,1],[\tau,1]\times[0,\tau],[\tau,1]\times[\tau,1].$$

In each subregion, we uniformly sample 1000×1000 grid points to evaluate the solution error. Table 5 reports the maximum errors for various values of ϵ, showing that the AE-PINNs method maintains high accuracy across a wide range of perturbation parameters. Fig. 7 displays the numerical and exact solutions, along with their pointwise differences. The results demonstrate that even with relatively small networks, the AE-PINNs approach yields excellent accuracy for highly singular problems.

Example 5: Convection-reaction problem in 2D. In this example, we consider the following 2D singularly perturbed convection-reaction problem on the

TABLE 6. Maxinum error for Example 5.

ϵ	10^{-1}	10^{-6}	10^{-10}	10^{-14}	10^{-16}
Maxinum error	4.1346e−3	2.2075e−2	1.5189e−2	2.7482e−2	1.8517e−2

domain $\Omega = [0,1]^2$:

$$\begin{aligned} &-\varepsilon \cdot \Delta u + \vec{\beta}(\boldsymbol{x}) \cdot \nabla u + c \cdot u = f(\boldsymbol{x}), \\ &u|_{\partial\Omega} = g, \end{aligned} \tag{18}$$

where we assume the convection field $\vec{\beta}(\boldsymbol{x}) = (1,1)$ and $c = 1$. The source term is chosen such that the exact solution is

$$u = xy \cdot (1 - e^{\frac{x-1}{\epsilon}}) \cdot (1 - e^{\frac{y-1}{\epsilon}}),$$

which clearly exhibits boundary layers near $x = 1$ and $y = 1$. The corresponding source term is given by

$$\begin{aligned} f(x,y) = -\epsilon \cdot \Big(x(2 + \frac{y}{\epsilon}) \cdot (e^{\frac{x-1}{\epsilon}} - 1) \cdot \frac{e^{\frac{y-1}{\varepsilon}}}{\epsilon} + y(2 + \frac{x}{\epsilon}) \cdot (e^{\frac{y-1}{\epsilon}} - 1) \cdot \frac{e^{\frac{x-1}{\epsilon}}}{\epsilon} \Big) \\ + (x+y+xy) \cdot (1 - e^{\frac{x-1}{\epsilon}}) \cdot (1 - e^{\frac{y-1}{\epsilon}}) - xy \cdot (e^{\frac{x-1}{\epsilon}} + e^{\frac{y-1}{\epsilon}} - 2e^{\frac{x-1}{\epsilon}} \cdot e^{\frac{y-1}{\epsilon}}). \end{aligned}$$

To solve this problem using the AE-PINNs framework, we design the following approximation:

$$\begin{aligned} \tilde{u}(x,y;\theta) =& NN_(x,y,\theta_1) + NN_2(x,y,\theta_2) \cdot e^{\frac{x-1}{\epsilon}} + NN_3(x,y,\theta_3) \cdot e^{\frac{y-1}{\epsilon}} \\ &+ NN_4(x,y,\theta_4) \cdot e^{\frac{x-1}{\epsilon}} e^{\frac{y-1}{\epsilon}} + NN_5(x,y,\theta_5) \cdot e^{\frac{-x}{\sqrt{\epsilon}}} \\ &+ NN_6(x,y,\theta_6) \cdot e^{\frac{-y}{\sqrt{\epsilon}}} + NN_7(x,y,\theta_7) \cdot e^{\frac{-x}{\sqrt{\epsilon}}} e^{\frac{-y}{\sqrt{\epsilon}}}, \end{aligned}$$

where $\theta = (\theta_1, \theta_2, \cdots, \theta_7)$, and each $NN_i(x,y,\theta_i)$ $(i = 1, 2, \cdots, 7)$ is a neural network with 6 hidden layers and 60 neurons per layer. During training, we randomly sample $40,000$ interior collocation points and $4,000$ boundary points per epoch, and train the model for $10,000$ epochs. To examine the accuracy in resolving boundary layers, we divide the domain into four subregions based on the threshold $\tau = \max\{\frac{1}{2}, 1 - 10\epsilon\}$:

$$[0,\tau] \times [0,\tau], [0,\tau] \times [\tau,1], [\tau,1] \times [0,\tau], [\tau,1] \times [\tau,1].$$

In each subregion, we evaluate the solution using a uniform grid of 1000×1000 points. The maximum pointwise errors for different values of ϵ are reported in Table 6. In addition, Fig. 8 presents visual comparisons between the exact and AE-PINNs approximated solutions, including their differences. As illustrated, the AE-PINNs framework accurately captures the solution behavior, even for very small ϵ, using relatively compact networks.

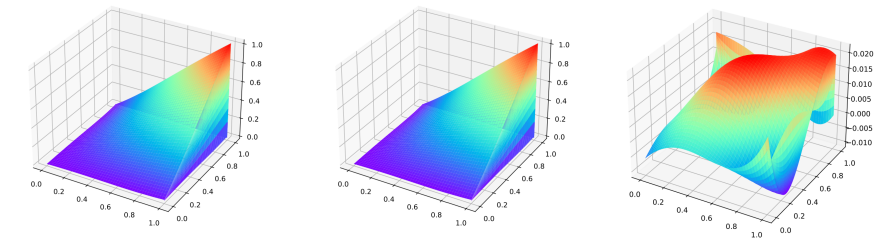
We also compare the performance of the standard PINNs method on the same problem. The loss function for standard PINNs is defined as:

$$\text{Loss}(\theta) = \text{Loss}_{eq}(\theta) + \rho \text{Loss}_{bc}(\theta), \tag{19}$$

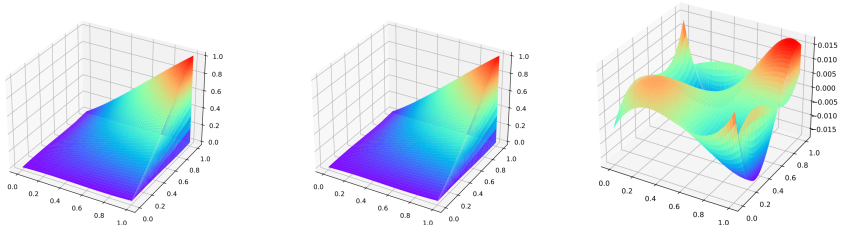
where ρ denotes a penalty parameter and

$$\text{Loss}_{eq}(\theta) = \sum_{m=1}^{N} \|f(x_m) + \epsilon \Delta NN(\boldsymbol{x}_m,\theta) - \beta(\boldsymbol{x}_m) \cdot \nabla NN(\boldsymbol{x}_m,\theta) + cNN(\boldsymbol{x}_m,\theta)\|^2,$$

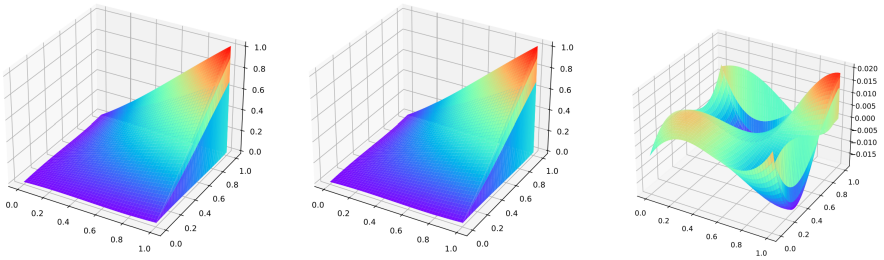
$$\text{Loss}_{bc}(\theta) = \sum_{m=1}^{M} \|NN(\boldsymbol{x}_m,\theta) - g(\boldsymbol{x}_m)\|^2.$$



(a) $\epsilon = 10^{-6}$.



(b) $\epsilon = 10^{-10}$



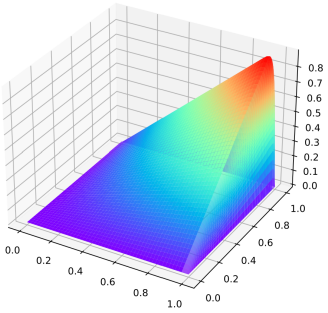
(c) $\epsilon = 10^{-16}$.

FIGURE 8. AE-PINN results for Example 5. In each subfigure, the left and middle panels show the exact and approximate solutions; while the right panel presents the corresponding error distribution. Each subfigure corresponds to a different perturbation parameter ϵ.

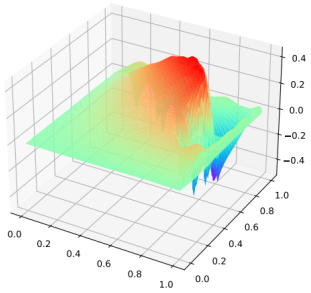
Here, N and M denote the number of interior and boundary points, respectively. We use a neural network architecture with 6 hidden layers and 60 neurons per layer, and test the method for $\epsilon = 0.01$. We again sample $40,000$ interior points per epoch and train for $15,000$ epochs. Fig. 9 shows the results for $\epsilon = 0.01$, revealing that the standard PINNs fails to resolve the boundary layers. The approximation is inaccurate and the interpolation errors remain large, highlighting the difficulty of standard PINNs in handling singular perturbation problems. This comparative study demonstrates the significant advantage of AE-PINNs over standard PINNs in resolving sharp boundary layers and achieving uniform accuracy across varying perturbation parameters.

5. Concluding Remarks

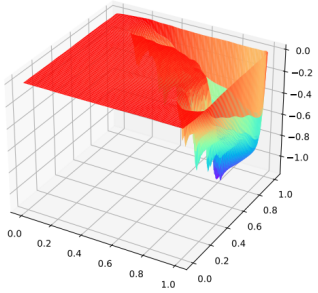
In this work, we have developed and systematically investigated a novel asymptotic expansion-based physics-informed neural networks (AE-PINNs) framework for efficiently solving singularly perturbed problems. By leveraging the multiscale



(a) Exact solution.



(b) Standard PINNs approximation.



(c) Errors.

FIGURE 9. Standard PINN results for Example 5. (a) exact solution; (b) standard PINNs approximation; and (c) corresponding pointwise error distribution. Results are shown for the case $\epsilon = 0.01$.

structure inherent in the asymptotic expansions of the exact solutions, we designed composite neural network architectures that explicitly incorporate boundary layer behavior into the solution ansatz. This strategy enables the AE-PINNs method to overcome the well-known limitations of standard PINNs, which often fail to capture sharp gradients or localized structures when the perturbation parameter ϵ becomes small. We present extensive numerical experiments on a variety of 1D and 2D reaction-diffusion, convection-diffusion, and reaction-convection-diffusion equations with both constant and variable coefficients. The results consistently demonstrate that AE-PINNs deliver uniformly accurate solutions across a wide range of ϵ, including values as small as $\epsilon = 10^{-16}$, while maintaining relatively simple network architectures and moderate computational cost. Comparisons with standard PINNs further highlight the robustness and superior resolution capabilities of the proposed approach in resolving boundary and internal layers.

Overall, this study provides a new perspective on incorporating analytical asymptotic structures into neural network design for solving multiscale PDEs. The AE-PINNs methodology not only enhances the approximation quality in the singular perturbation regime, but also lays a foundation for future developments in hybrid analytical machine learning frameworks designed for challenging PDE problems with multiscale structures or localized features.

Acknowledgments

The work of S. Xu and J. Wang were partially supported by the National Natural Science Foundation of China (grant No. 12371394, 52331002), Key Project of Hunan Provincial Department of Education (grant No. 22A0033).

References

[1] Mark Ainsworth and Justin Dong. Galerkin neural network approximation of singularly-perturbed elliptic systems. *Computer Methods in Applied Mechanics and Engineering*, 402:115169, 2022.

[2] María Gabriela Armentano, Ariel L Lombardi, and Cecilia Penessi. Robust estimates in balanced norms for singularly perturbed reaction diffusion equations using graded meshes. *Journal of Scientific Computing*, 96(1):18, 2023.

[3] Amirhossein Arzani, Kevin W. Cassel, and Roshan M. DSouze. Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation. *Journal of Computational Physics*, 473:111768.

[4] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[5] Fujun Cao, Fei Gao, Xiaobin Guo, and Dongfang Yuan. Physics-informed neural networks with parameter asymptotic strategy for learning singularly perturbed convection-dominated problem. *Computer & Mathematics with Applications*, 150:229–242.

[6] Mario De Florio, Enrico Schiassi, Francesco Calabr, and Roberto Furfaro. Physics-informed neural networks for 2nd order ODEs with sharp gradients. *Journal of Computational and Applied Mathematics*, 436:115396, 2024.

[7] Ting Du, Zhongyi Huang, and Ye Li. Approximation and generalization of deeponets for learning operators arising from a class of singularly perturbed problems. 2023.

[8] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349380, November 2017.

[9] Zhiwei Gao, Liang Yan, and Tao Zhou. Failure-informed adaptive sampling for PINNs. *SIAM Journal on Scientific Computing*, 45(4):A1971–A1994, 2023.

[10] Fasika Wondimu Gelu and Gemechis File Duressa. Hybrid method for singularly perturbed robin type parabolic convectiondiffusion problems on shishkin mesh. *Partial Differential Equations in Applied Mathematics*, 8:100586, 2023.

[11] Gung-Min Gie, Youngjoon Hong, Chang-Yeol Jung, and Tselmuun Munkhjin. Semi-analytic PINN methods for boundary layer problems in a rectangular domain. *Journal of Computational and Applied Mathematics*, 450:115989, 2024.

[12] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[13] Mohan K Kadalbajoo and Kailash C Patidar. ϵ-Uniformly convergent fitted mesh finite difference methods for general singular perturbation problems. *Applied mathematics and computation*, 179(1):248–266, 2006.

[14] George Em 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.

[15] Natalia Kopteva and Richard Rankin. Pointwise a posteriori error estimates for discontinuous Galerkin methods for singularly perturbed reaction-diffusion equations. *SIAM Journal on Numerical Analysis*, 61(4):1938–1961, 2023.

[16] Hornik Kurt. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[17] Ye Li, Ting Du, Yiwen Pang, and Zhongyi Huang. Component fourier neural operator for singularly perturbed differential equations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(12):13691–13699, Mar. 2024.

[18] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[19] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[20] Tao Luo and Qixuan Zhou. On residual minimization for pdes: Failure of PINN, modified equation, and implicit bias. *arXiv preprint arXiv:2310.18201*, 2023.
[21] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.
[22] Zhiping Mao and Xuhui Meng. Physics-informed neural networks with residual/gradient-based adaptive sampling methods for solving partial differential equations with sharp solutions. *Applied Mathematics and Mechanics*, 44(7):1069–1084, 2023.
[23] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.
[24] 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. In *Journal of Computational Physics*, 378:686-707, 2019.
[25] Hans-Görg Roos. *Robust numerical methods for singularly perturbed differential equations*. Springer, 2008.
[26] I Sykopetritou and C Xenophontos. An hp finite element method for a two-dimensional singularly perturbed boundary value problem with two small parameters. *Journal of Computational and Applied Mathematics*, 439:115620, 2024.
[27] E Weinan and Bing Yu. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.
[28] Ziqing Xie and Zhimin Zhang. Uniform superconvergence analysis of the discontinuous Galerkin method for a singularly perturbed problem in 1-d. *Mathematics of Computation*, 79(269):35–45, 2010.
[29] Wenchao Zheng, Jin Zhang, and Xiaoqi Ma. Uniform convergence of the LDG method for singularly perturbed problems. *Applicable Analysis*, pages 1–9, 2023.
[30] Huiqing Zhu and Zhimin Zhang. Uniform convergence of the LDG method for a singularly perturbed problem with the exponential boundary layer. *Mathematics of Computation*, 83(286):635–663, 2014.

†MOE-LCSM, School of Mathematics and Statistics, Hunan Normal University, Changsha, Hunan 410081, P. R. China.
E-mail: 3075532458@qq.com.

♮MOE-LCSM, School of Mathematics and Statistics, Hunan Normal University, Changsha, Hunan 410081, P. R. China.
E-mail: jxwang@hunnu.edu.cn.

♮Harrow International School, Hong Kong, P. R. China.
E-mail: yiduozhang79@gmail.com.

‡Corresponding author. Department of Mathematics, University of South Carolian, Columbia, SC 29208, USA.
E-mail: xfyang@math.sc.edu.