Lu et al. *Res Math Sci* (2022) 9:45
https://doi.org/10.1007/s40687-022-00345-z

Research in the Mathematical Sciences

RESEARCH

Solving multiscale steady radiative transfer equation using neural networks with uniform stability

Yulong Lu[1], Li Wang[2] and Wuzhe Xu[2*]

*Correspondence:
xu000355@umn.edu
[2]School of Mathematics, University of Minnesota, Twin Cities, MN 55455, USA
Full list of author information is available at the end of the article

Abstract

This paper concerns solving the steady radiative transfer equation with diffusive scaling, using the physics informed neural networks (PINNs). The idea of PINNs is to minimize a least-square loss function, that consists of the residual from the governing equation, the mismatch from the boundary conditions, and other physical constraints such as conservation. It is advantageous of being flexible and easy to execute, and brings the potential for high dimensional problems. Nevertheless, due the presence of small scales, the vanilla PINNs can be extremely unstable for solving multiscale steady transfer equations. In this paper, we propose a new formulation of the loss based on the macro-micro decomposition. We prove that, the new loss function is uniformly stable with respect to the small Knudsen number in the sense that the L^2-error of the neural network solution is uniformly controlled by the loss. When the boundary condition is an-isotropic, a boundary layer emerges in the diffusion limit and therefore brings an additional difficulty in training the neural network. To resolve this issue, we include a boundary layer corrector that carries over the sharp transition part of the solution and leaves the rest easy to be approximated. The effectiveness of the new methodology is demonstrated in extensive numerical examples.

Keywords: Radiative transfer equation, Diffusion limit, Boundary layer, PINN, Uniform stability

Mathematics Subject Classification: 68T07, 65N12, 82B40, 76R50

1 Introduction

Developing efficient and robust numerical scheme for multiscale kinetic equation has always been a challenging yet important subject of research, and has attracted a lot of attention in the past decade. The main difficulty comes from the stiffness raised by multiple scales of the equation, which generically requires fine spatial mesh grid and short time step to guarantee both accuracy and stability. A large number of numerical schemes has been devoted to relaxing such a requirement, in the traditional grid-based framework, including the finite difference method, finite volume method, discrete Galerkin method, and etc [5,12,22]. Recently, deep learning method has emerged as a competitive mesh-free method for solving partial differential equations (PDEs). The idea is to represent

solutions of PDEs by (deep) neural networks to take advantage of the rich expressiveness of neural networks representation. The parameters of neural networks are chosen by training or optimizing some loss functions associated with the PDE. It is advantageous of being intuitive and easy to execute, and also offers an innovational approach for solving high dimensional problems.

Many deep learning methods, based on optimizing different loss functions, have been developed for solving PDEs. To the best of our knowledge, the first neural network method PDE solver dates back to [18] and builds on minimizing the L^2-residual of the PDE and that of the boundary/initial conditions. The now-days popular physical informed neural network (PINN) [32] and deep Galerkin method (DGM) [34] fail into the same residual minimization framework. Another method called Deep Ritz Method [39] is designed to solve some PDE problems with variational structures by exploiting the Ritz formulation of PDEs. The deep BSDE method [10] was developed for solving some parabolic PDEs based on the stochastic representation of the solutions. For discussions of other machine learning methods for PDEs, we refer the interested reader to the excellent review article [9].

Recently, several works [4,11,26] proposed neural network methods for solving kinetic equations by employing the framework of PINNs. However, the error bounds proved in those works for vanilla PINNs deteriorate in the diffusive regime where the Knudsen number is small. More specifically, the stability estimates proved for the vanilla PINN loss functions blow up as the Knudsen number tends to zero. The purpose of the present paper is to build a new loss function which satisfies a stability estimate that is uniform with respect to the Knudsen number in the diffusive regime. Consider the steady radiative transfer equation (RTE), which takes the following general form:

$$\begin{cases} \varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{v}) = \sigma_s(\boldsymbol{x}) \mathcal{L} f(\boldsymbol{x}, \boldsymbol{v}) - \varepsilon^2 \sigma_a(\boldsymbol{x}) f + \varepsilon^2 G(\boldsymbol{x}), (\boldsymbol{x}, \boldsymbol{v}) \in \Omega := \Omega_x \times \boldsymbol{S}^{d-1}, \\ f(\boldsymbol{x}, \boldsymbol{v}) = \phi(\boldsymbol{x}, \boldsymbol{v}), (\boldsymbol{x}, \boldsymbol{v}) \in \Gamma_-. \end{cases} \tag{1.1}$$

Here $f(t, \boldsymbol{x}, \boldsymbol{v})$ is the distribution of particles at time t and location $\boldsymbol{x}$ with velocity $\boldsymbol{v}$, $G(\boldsymbol{x})$ is source function and $\Gamma_- := \{(\boldsymbol{x}, \boldsymbol{v}) \in \partial\Omega_x \times \boldsymbol{S}^{d-1} | \ \boldsymbol{v} \cdot \boldsymbol{n}_x < 0\}$ is the inflow boundary. Assume that Ω_x is bounded and Lipschitz on $\mathbb{R}^d$. We also assume that the inflow boundary value $\phi(\boldsymbol{x}, \boldsymbol{v}) \in L^2(\Gamma_-)$. The parameter $\varepsilon > 0$, often termed as Knudsen number, is a dimensionless parameter that governs the regime of the equation. In particular, $\varepsilon \sim \mathcal{O}(1)$ refers to kinetic regime, and $\varepsilon \ll 1$ corresponds to the diffusive regime. The scattering operator $\mathcal{L}$ is defined by

$$\mathcal{L} f = \frac{1}{|\boldsymbol{S}^{d-1}|} \int_{\boldsymbol{S}^{d-1}} K(\boldsymbol{v}, \boldsymbol{v}')(f(\boldsymbol{v}') - f(\boldsymbol{v})) \mathrm{d}\boldsymbol{v}',$$

where $K : \boldsymbol{S}^{d-1} \times \boldsymbol{S}^{d-1} \to \mathbb{R}$ is a nonnegative kernel. The functions σ_s and σ_a are the scattering coefficient and absorption coefficient respectively. In addition, we assume the following assumption is valid.

Assumption 1 There exist positive constants $\sigma_{\min}$ and $\sigma_{\max}$ such that

$$0 < \sigma_{\min} \leq \sigma_s(\boldsymbol{x}) \leq \sigma_{\max} \text{ and } 0 \leq \sigma_a(\boldsymbol{x}) \leq \sigma_{\max}.$$

Throughout the paper, we also make the following assumption on the scattering operator $\mathcal{L}$, which will play an essential role in obtaining a stability estimate for our new PINN loss function.

Assumption 2 The scattering operator $\mathcal{L}$ satisfies

1) $\langle \mathcal{L}f \rangle := \frac{1}{|\boldsymbol{S}^{d-1}|} \int_{\boldsymbol{S}^{d-1}} \mathcal{L}f \, \mathrm{d}\boldsymbol{v} = 0$ for any $f(\boldsymbol{v}) \in L^2(\boldsymbol{S}^{d-1})$;
2) the null space of $\mathcal{L}$ is $\mathcal{N}(\mathcal{L}) = \{f = \langle f \rangle\}$;
3) $\mathcal{L}$ is non-positive self-adjoint in $L^2(\boldsymbol{S}^{d-1})$, and moreover, $\langle f \mathcal{L}f \rangle \leq -c \langle f^2 \rangle$ for every $f \in \mathcal{N}^{\perp}(\mathcal{L})$ and some constant $c > 0$;
4) $\mathcal{L}$ admits a pseudo-inverse from $\mathcal{N}^{\perp}(\mathcal{L})$ to $\mathcal{N}^{\perp}(\mathcal{L})$.
5) There exists $C_K > 0$ such that $\|\mathcal{L}f\|_{L^2(\Omega)} \leq C_K \|f\|_{L^2(\Omega)}$.

Under Assumptions 1 and 2, it is well-known that RTE (1.1) is well-posedness as shown in the theorem below. To state the theorem, let us first define the function space $\mathcal{X}$ by setting

$$\mathcal{X} := \{f \in L^2(\Omega) \,|\boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f \in L^2(\Omega)\}.$$

Theorem 1 *Suppose that Assumptions 1 and 2 hold. There exists a unique solution f to (1.1) such that $f \in \mathcal{X}$ and*

$$\|f\|_{L^2(\Omega)} + \|\boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f\|_{L^2(\Omega)} \leq C(\|\phi\|_{L^2(\Gamma_I)} + \|G\|_{L^2(\Omega_x)}),$$

where the constant C depends on σ_a, σ_s, Ω and ε.

Proof Thanks to [6, Theorem 1.1], problem (1.1) has a unique solution in $L^2(\Omega)$. Moreover,

$$\|f\|_{L^2(\Omega)} \leq C(\|\phi\|_{L^2(\Gamma_I)} + \|G\|_{L^2(\Omega_x)}).$$

Notice that the stability bound in [6, Theorem 1.1] is slightly stronger than the one stated above since the L^2-bound there is weighted against $\ell(\boldsymbol{x}, \boldsymbol{v})$, which is the length of line segment through $\boldsymbol{x}$ in direction $\boldsymbol{v}$ completely contained in $\Omega_{\boldsymbol{x}}$. The gradient bound on $\|\boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f\|$ follows directly by taking L^2-norm on both sides of (1.1) and the estimate above. □

In the diffusive regime ($\varepsilon \ll 1$), problem (1.1) is well approximated by the elliptic equation:

$$\left\langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \mathcal{L}^{-1} \left(\frac{1}{\sigma_s} \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \rho_0 \right) \right\rangle = -\sigma_a \rho_0 + G, \quad \rho_0|_{\partial\Omega} = \xi(x),$$

where ξ is obtained through the boundary layer analysis [1]. In many applied problems, the magnitude of ε can vary significantly across different regions; in this case it is desirable to have a solver that can deal with both kinetic ($\varepsilon \sim \mathcal{O}(1)$) and diffusion ($\varepsilon \ll 1$) regimes. Methods that fulfill this task fall into two categories, the domain-decomposition method [7,23] and the asymptotic preserving method [3,14,15,21,24,31,35,36]. The former one solves different equations in different regimes and constructs an interface condition to connect them, whereas the latter seeks a unified solver that works in both regimes and therefore avoids the complication in identifying the interface location and designing interface condition. For time dependent RTE, there has been a vast literature on developing asymptotic preserving methods [14], with the focus on resolving the stability issue by way of an implicit-explicit time discretization.

For stationary problems, on the other hand, specific challenge arises due to the presence of boundary layer. In general, generic numerical method may induce a numerical boundary

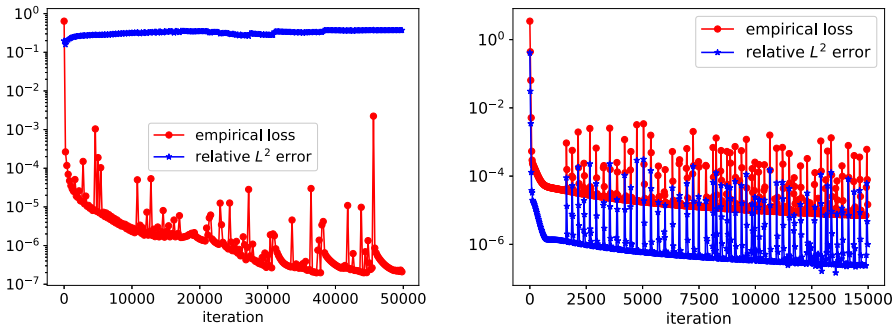

Fig. 1 Comparison of results solving (3.3) with $\varepsilon = 10^{-3}$. Left is obtained with vanilla loss and right is with loss from macro-micro decomposition

condition in the zero ε limit that does not match the theoretical boundary condition, and then introduces errors not only on the boundary but also inside the computational domain. To this end, several efforts have been made to incorporate part of the boundary layer information into the scheme. For instance, Klar [17] constructed a boundary condition for the diffusion equation by approximating a Milne problem. Han, Tang and Ying [8] developed a tailored finite volume scheme that is uniform accurate up to boundary by freezing the coefficient in each cell and use special solutions to the constant coefficient equation as local basis functions. Lemou and Mehats [20] proposed a new macro-micro decomposition by choosing the macro part such that its incoming velocity moments coincide with that of the distribution function, and therefore directly injected the exact boundary condition into the macro-micro system. When the collision kernel K is isotropic, boundary layer can also be resolved by the Chandrasekhar H-function [14,29,40]. For general collision kernel, Li, Lu and Sun [23] have proposed a half space solver, which then leads to an interface condition to connect different regimes.

The primary goal in this paper is to develop a neural network method that is uniformly stable and accurate for solving (1.1) in both the kinetic ($\varepsilon \sim 1$) and the diffusive regimes ($\varepsilon \ll 1$). It is important to emphasize that the vanilla PINN is not able to resolve the solution when $\varepsilon \ll 1$. In fact, one can construct examples where the neural network solution differs much from the exact solution whereas the vanilla PINN loss is small; see Sect. 3.1 for such an example. To overcome the instability issue, we propose a new loss function to train the neural network using the idea of macro-micro decomposition that underlies many asymptotic preserving methods. We shall show later that under some assumptions the new loss satisfies a uniform stability estimate. As a illustration, let us compare in Fig. 1 the errors of solutions computed using two loss functions for the toy example in Sect. 3.1 (see Eq. (3.3)). One sees that if the vanila PINN loss is used the relative L^2-error remains $O(1)$ even when the (empirical) PINN loss already decreases to below 10^{-7}. Whereas our new PINN loss yields that the relative L^2-error decreases along with the decreasing empirical loss.

Another issue—the boundary layer arises when the boundary data ϕ is variant in the $\boldsymbol{v}$-direction. Its presence can significantly slow down the training of neural networks. To deal with this issue, we construct a boundary layer corrector that mitigates the sharp transition in the solution and therefore eases the training process significantly. In comparison with the recent work in the same vein [19,26], our contributions are highlighted as follows:

- We design a new least-square-type loss function based on the idea of macro-micro decomposition of (1.1) and prove that the new loss satisfies a stability estimate that is uniform with respect to small Knudsen number.
- When boundary layer is present in (1.1), we modify the macro-micro decomposition by incorporating a boundary layer corrector that can capture the sharp transition of the solution near the boundary.
- We demonstrate the accuracy and robustness of the proposed methodologies in a wide range of numerical experiments.

Parallel to our work here, we would like to mention a recent manuscript [13] that considers the time dependent case; it shares similar ideas of macro-micro decomposition, but with many details differently.

The rest of the paper is organized as follows. In the Sect. 2, we recall a formal derivation of the diffusion limit of (1.1) and summarize the half space problem for boundary layer in multiple dimensions. In Sect. 3, we first discuss the pitfalls of the vanilla PINN loss and then introduce new loss functions based on the macro-micro decomposition (with and without boundary layer corrector). Theoretical stability estimates of the new loss function are proved in Sect. 4. Finally, we illustrate the accuracy and efficiency of our method by presenting several numerical examples in Sect. 5.

2 The diffusion approximation for the radiative transfer equation

In this section, we collect some preliminary information regarding the diffusion approximation of the radiative transfer equation, both inside the domain and near the boundary. In particular, we have the following theorem.

Theorem 2 *Suppose f solves (1.1). Then as $\varepsilon \to 0$, $f(\boldsymbol{x}, \boldsymbol{v})$ converges to $\rho_0(\boldsymbol{v})$, which solves*

$$\left\langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \mathcal{L}^{-1} \left(\frac{1}{\sigma_s} \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \rho_0 \right) \right\rangle = -\sigma_a \rho_0 + G, \qquad \rho_0 \big|_{\partial\Omega} = \zeta(\boldsymbol{x}). \tag{2.1}$$

Here $\zeta(\boldsymbol{x})$ at any point $\boldsymbol{x_b} \in \partial_x \Omega$ is determined by

$$\zeta(\boldsymbol{x_b}) = \lim_{z \to \infty} f_{BL}(z, \boldsymbol{v}; \boldsymbol{x_b}),$$

where $f_{BL}(z, \boldsymbol{v}; \boldsymbol{x_b})$ solves the half space problem:

$$(-\boldsymbol{v} \cdot \boldsymbol{n_b}) \partial_z f_{BL} = \mathcal{L}(f_{BL}), \quad f_{BL}(0, \boldsymbol{v}) = \phi(\boldsymbol{x_b}, \boldsymbol{v}), \quad \boldsymbol{v} \cdot \boldsymbol{n}_b < 0. \tag{2.2}$$

Proof Here we provide a formal derivation. Rigorous proof can be found in [1]. Away from the boundary, consider the Hilbert expansion of $f(x, v)$:

$$f(\boldsymbol{x}, \boldsymbol{v}) = f_0(\boldsymbol{x}, \boldsymbol{v}) + \varepsilon f_1(\boldsymbol{x}, \boldsymbol{v}) + \varepsilon f_2(\boldsymbol{x}, \boldsymbol{v}) + \cdots$$

which inserting into (1.1) leads to the following equations with like powers:

$$\mathcal{O}(1): \qquad \mathcal{L}(f_0) = 0, \tag{2.3}$$

$$\mathcal{O}(\varepsilon): \qquad \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f_0 = \sigma_s \mathcal{L} f_1, \tag{2.4}$$

$$\mathcal{O}(\varepsilon^2): \qquad \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f_1 = \sigma_s \mathcal{L} f_2 - \sigma_a f_0 + G. \tag{2.5}$$

First (2.3) implies $f_0(x, v) = \langle f_0 \rangle := \rho_0(x)$. From (2.4), according to the property of $\mathcal{L}$, since $\boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \rho_0 \in \mathcal{N}(\mathcal{L})^{\perp}$, we can write $f_1 = \mathcal{L}^{-1} \left(\frac{1}{\sigma_s} \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \rho_0 \right)$. Then plugging this relation to

(2.5) and taking average in v, one can show that ρ_0 satisfies the elliptic equation in (2.1). Consequently, we obtain that f converges to ρ_0 as $\varepsilon \to 0$, with ρ_0 solving (2.1).

In general, the boundary condition for ρ_0 is different from f due to the presence of boundary layer. Therefore, we need to conduct the matched asymptotic boundary layer analysis to obtain the correct boundary data for ρ_0. Our derivation follows [2], see also [16,17]. For a given point $\boldsymbol{x_b}$ on the boundary, i.e., $\boldsymbol{x_b} \in \partial\Omega_x$, let $\boldsymbol{n_b}$ be the outer normal direction at $\boldsymbol{x_b}$, then we define locally a stretching variable $z = z(\boldsymbol{x}; \boldsymbol{x_b}) \in [0, \infty)$ in a way such that

$$\sigma_s(\boldsymbol{x})\boldsymbol{v} \cdot \boldsymbol{n_b} d_z = -\varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}}, \ d_z = \frac{\mathrm{d}}{\mathrm{d}z}. \tag{2.6}$$

For instance, when σ_s is independent of x, $z = \frac{(\boldsymbol{x_b} - \boldsymbol{x}) \cdot \boldsymbol{n_b}}{\varepsilon}$; when $d = 1$, at the left boundary $x = x_L$, $z = \frac{1}{\varepsilon} \int_{x_L}^{x} \sigma_s(t)\mathrm{d}t$. It is obvious that when $\boldsymbol{x} = \boldsymbol{x_b}$, $z = 0$; when $\boldsymbol{x}$ is away from $\boldsymbol{x_b}$, $z \to \infty$ as $\varepsilon \to 0$. Then along z direction, (1.1) reads, in the leading order of ε,

$$\begin{cases} (-\boldsymbol{v} \cdot \boldsymbol{n_b}) \partial_z f = \mathcal{L}(f), \\ f(0, \boldsymbol{v}) = \phi(\boldsymbol{x_b}, \boldsymbol{v}), \ \boldsymbol{v} \cdot \boldsymbol{n}_b < 0. \end{cases}$$

The well-posedness of the above problem can be found in [7]. In particular, if $\phi(\boldsymbol{x_b}, \boldsymbol{v}) \in L^2(\boldsymbol{S}^-_{\boldsymbol{n_b}}, |\boldsymbol{v} \cdot \boldsymbol{n_b}|\mathrm{d}\boldsymbol{v})$, where $\boldsymbol{S}^-_{\boldsymbol{n_b}} = \{\boldsymbol{v} \in \boldsymbol{S}^{d-1} | \boldsymbol{v} \cdot \boldsymbol{n_b} < 0\}$, (2.2) has a unique solution in $L^\infty(\mathbb{R}_+; L^2(\boldsymbol{S}^-_{\boldsymbol{n_b}}, |\boldsymbol{v} \cdot \boldsymbol{n_b}|\mathrm{d}\boldsymbol{v}))$. In addition, denote its solution as $f_{BL}(z, \boldsymbol{v}; \boldsymbol{x_b})$, then it can be shown that

$$f_{BL}(z, \boldsymbol{v}; \boldsymbol{x_b}) \to f^\infty_{BL}(\boldsymbol{x_b}), \quad as\ z \to \infty, \tag{2.7}$$

where $f^\infty_{BL}(\boldsymbol{x_b})$ is a function independent of $\boldsymbol{v}$, which gives the condition for ρ_0 at $\boldsymbol{x_b}$, i.e., $\zeta(\boldsymbol{x_b}) = f^\infty_{BL}(\boldsymbol{x_b})$. The same procedure can be carried out at each point on the boundary $\partial\Omega_x$ and we therefore obtain the boundary condition $\zeta[\phi]$ for ρ_0. □

Note that, when the collision kernel is isotropic, i.e., $K(\boldsymbol{v}, \boldsymbol{v}') = 1$, there is an explicit relationship between $f^\infty_{BL}(\boldsymbol{x_b})$ and $\phi(\boldsymbol{x_b}, \boldsymbol{v})$, through the Chandrasekhar's H-function, see Sect. 2 (for dimension one) and Appendix B (for multi dimension) in [7].

Additionally, when the boundary condition is independent of $\boldsymbol{v}$, that is, $\phi(\boldsymbol{x}, \boldsymbol{v}) = \phi(\boldsymbol{x})$, $\boldsymbol{x} \in \partial\Omega_x$, then there is no boundary layer and hence $\zeta(\boldsymbol{x}) = \phi(\boldsymbol{x})$. This is seen from the fact that $f_{BL}(z, \boldsymbol{v}; \boldsymbol{x_b}) \equiv \phi(\boldsymbol{x_b})$ is a solution to (2.2).

3 Approximation by physics informed neural networks (PINNs)

We aim to approximate the solution of (1.1) with functions that are parameterized by neural networks. Denote $f^{nn}(\theta; \boldsymbol{x}, \boldsymbol{v})$ the neural network function where θ represents the set of neural network parameters including weights and biases in the neurons. In the framework of PINNs and other neural network-based approaches, one seeks the approximation $f^{nn}(\theta; \boldsymbol{x}, \boldsymbol{v})$ by minimizing a loss function that is defined by the PDE problem. The vanilla PINN (population) loss is defined as the sum of the L^2-misfit of the PDE and that of the boundary values:

$$\begin{aligned} \mathcal{E}_0(f) = &\int_\Omega \left| \varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{v}) - \sigma_s(\boldsymbol{x}) \mathcal{L} f(\boldsymbol{x}, \boldsymbol{v}) + \varepsilon^2 \sigma_a(\boldsymbol{x}) f - \varepsilon^2 G(\boldsymbol{x}) \right|^2 \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{v} \\ &+ \int_{\Gamma_-} |f - \phi|^2 \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{v}. \end{aligned} \tag{3.1}$$

In practice, we need to approximate the integrations above and this leads to the definition of the empirical PINN loss:

$$\mathcal{E}_0^N(f) = \sum_{i=1}^{N_x^r}\sum_{j=1}^{N_v^r}\Big|\varepsilon \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_i^r,\boldsymbol{v}_j^r) - \sigma_s(\boldsymbol{x}_i^r)\mathcal{L}f(\boldsymbol{x}_i^r,\boldsymbol{v}_j^r) + \varepsilon^2\sigma_a(\boldsymbol{x}_i^r)f(\boldsymbol{x}_i^r,\boldsymbol{v}_j^r) - \varepsilon^2 G(\boldsymbol{x}_i^r)\Big|^2 w_{ij}^q + \sum_{m=1}^{N_x^b}\sum_{n=1}^{N_v^b}|f(\boldsymbol{x}_i^b,\boldsymbol{v}_j^b)-\phi(\boldsymbol{x}_i^b,\boldsymbol{v}_j^b)|^2 w_{ij}^b. \tag{3.2}$$

Here $\{\boldsymbol{x}_i^r\}_{i=1}^{N_x^r}$, $\{\boldsymbol{v}_j^r\}_{j=1}^{N_v^r}$ and $\{w_j^r\}_{j=1}^{N_v^r}$, and $\{\boldsymbol{x}_m^b\}_{m=1}^{N_x^b}$, $\{\boldsymbol{v}_n^b\}_{n=1}^{N_v^b}$ and $\{w_j^b\}_{j=1}^{N_v^b}$ are the interior and boundary quadrature points and weights, respectively. See concrete choices of quadrature points and weights in Sect. 5. Then a neural network approximation $f^{nn}(\theta;\boldsymbol{x},\boldsymbol{v})$ is obtained by solving the minimization problem:

$$\min_\theta \mathcal{E}_0^N(f^{nn}(\theta;\boldsymbol{x},\boldsymbol{v})).$$

3.1 Pitfall of vanilla PINN with small ε

In this section, we would like to point out one pitfall of vanilla PINN loss (3.1) (or (3.2)) in the case where the Knudsen number ε is small; this is illustrated in the following simple example. Consider the one dimensional boundary value problem:

$$\begin{cases} \varepsilon v\partial_x f = \langle f\rangle - f - \varepsilon v \quad x\in[0,1], v\in[-1,1],\\ f(0,v>0)=1,\quad f(1,v<0)=0, \end{cases} \tag{3.3}$$

whose analytic solution is given by $f^*(x,v)=1-x$. Its vanilla PINN loss takes the form

$$\mathcal{E}_v(f) = \|\varepsilon v\partial_x f - \langle f\rangle + f + \varepsilon v\|_{L^2(\Omega)}^2 + \|f(0,\cdot)-1\|_{L^2([0,1])}^2 + \|f(1,\cdot)\|_{L^2([-1,0])}^2. \tag{3.4}$$

Then it is obvious that when $\varepsilon \ll 1$, any v-independent function f that satisfies the boundary condition, such as $f(x,v)=(1-x)^2$, and $(1-x)^3$ leads to $\mathcal{E}(f)=\mathcal{O}(\varepsilon^2)$. However, for those f we have $\|f-f^*\|_{L^2(\Omega)}=\mathcal{O}(1)$. This shows that the vanilla PINN loss does not provide a good error indicator for the kinetic equation (1.1) when ε is small. Consequently, training neural networks with the vanilla PINN loss can potentially lead to inaccurate estimation of the solution; see Fig. 2. Here we use a fully connected neural network with 4 hidden layers and 50 neurons within each hidden layer. In computing (3.2), the parameters are chosen as $N_v^b=60$, $N_x^r=80$ and $N_v^r=60$. In fact, when $\varepsilon=1$, as shown in Fig. 2, PINN is able to give an accurate approximation to the analytic solution. However, when $\varepsilon=10^{-3}$, we observed in Fig. 2 that even the empirical loss decreases to as small as 10^{-8}, the prediction is far away from the analytic solution. This observation motivates us to consider the macro-micro decomposition technique, which has become a standard numerical technique in solving multiscale problems.

3.2 PINN based on macro-micro decomposition

In order to resolve the issue mentioned in the last section, we propose a new loss function based on a macro-micro decomposition. Write

$$f=\rho(\boldsymbol{x})+\varepsilon g(\boldsymbol{x},\boldsymbol{v}),\quad \text{with } \rho=\langle f\rangle,\ \langle g\rangle=0, \tag{3.5}$$

then (1.1) can be decomposed into

$$\begin{cases} \langle \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} g\rangle = -\sigma_a\rho + G,\\ \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}}(\rho+\varepsilon g) - \varepsilon\langle \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} g\rangle = \sigma_s\mathcal{L}g - \varepsilon^2\sigma_a g,\\ \rho+\varepsilon g|_{\Gamma_-} = \phi. \end{cases} \tag{3.6}$$

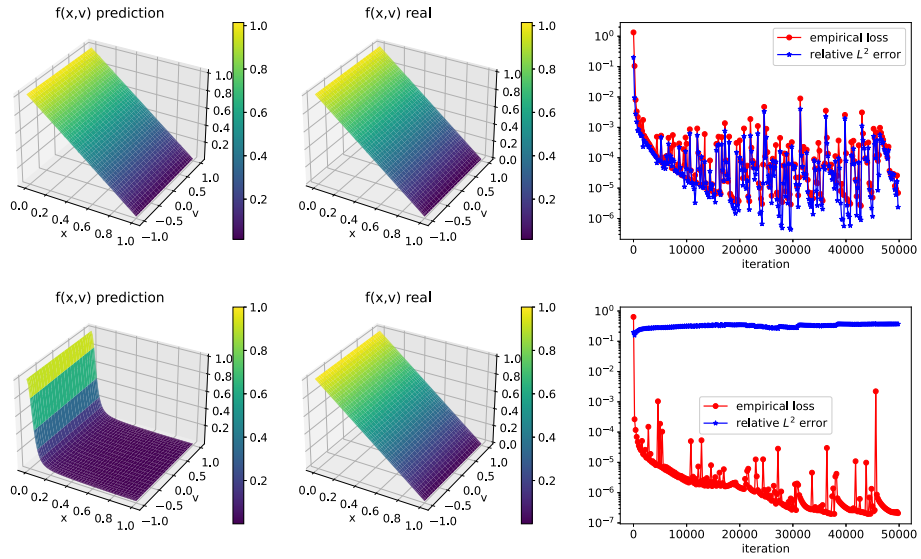

Fig. 2 Computation of (3.3) with vanilla loss (3.4). The top row is for $\varepsilon = 1$ and bottom row is for $\varepsilon = 0.001$. Left column is the predicted $f(x, v)$ from neural network, middle column is the analytic solution, and the right column is the error versus iterations with Adam optimizer

Instead of (3.1), we propose the following loss function:

$$\begin{aligned}\mathcal{E}(f) &:= \mathcal{E}(\rho, g)\\ &= \| \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} g \rangle + \sigma_a \rho - G \|^2_{L^2(\Omega)} + \| \langle g \rangle \|^2_{L^2(\Omega)} + \| \rho + \varepsilon g - \phi \|^2_{L^2(\Gamma_-)}\\ &\quad + \| \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} (\rho + \varepsilon g) - \varepsilon \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} g \rangle - \sigma_s \mathcal{L} g + \varepsilon^2 \sigma_a g - \varepsilon G \|^2_{L^2(\Omega)}.\end{aligned} \tag{3.7}$$

Now let us revisit the example (3.3). Applying the decomposition (3.6) leads to

$$\begin{cases} \langle v \partial_x g \rangle = 0, \\ v \partial_x (\rho + \varepsilon g) = -g - v, \\ \rho(0) + \varepsilon g(0, v > 0) = 1, \ \rho(1) + \varepsilon g(1, v < 0) = 0, \end{cases} \tag{3.8}$$

which gives the following loss function

$$\begin{aligned}\mathcal{E}(f) &= \| \langle v \partial_x g \rangle \|^2_{L^2(\Omega)} + \| v \partial_x (\rho + \varepsilon g) + g + v \|^2_{L^2(\Omega)}\\ &\quad + \int_0^1 (\rho(0) + \varepsilon g(0, v) - 1)^2 \mathrm{d}v + \int_{-1}^0 (\rho(1) + \varepsilon g(1, v))^2 \mathrm{d}v.\end{aligned} \tag{3.9}$$

Note that we have eliminated the term $\| \langle g \rangle \|^2_{L^2(\Omega)}$ as $\langle g \rangle = 0$ is guaranteed by the second equation in (3.8). With the new loss function (3.9), we get a good approximation to the analytic solution for $\varepsilon = 10^{-3}$, see Fig. 3.

3.3 Boundary layer corrector

The example (3.3) we have mentioned thus far is with homogeneous in $\boldsymbol{v}$ boundary condition, and the PINN loss (3.7) works just fine. However, when boundary value ϕ depends on $\boldsymbol{v}$, the boundary layer will arise, which brings in additional challenge as one needs to approximate a fast varying function.

We illustrate this difficulty through an example. Consider

$$\begin{cases} \varepsilon v \partial_x f = \langle f \rangle - f, \\ f(0, v > 0) = 5 \sin(v), \quad f(1, v < 0) = 0 \end{cases} \tag{3.10}$$

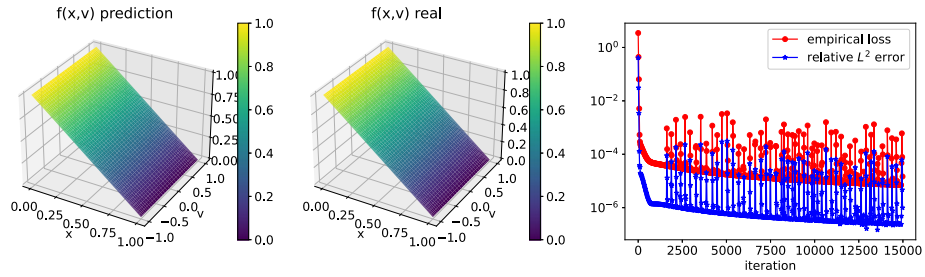

Fig. 3 Computation of (3.3) with new loss (3.9) for $\varepsilon = 10^{-3}$. The left figure is the predicted $f(x, v)$ from neural network, the middle is the analytic solution, and the right is the error versus iterations with Adam optimizer

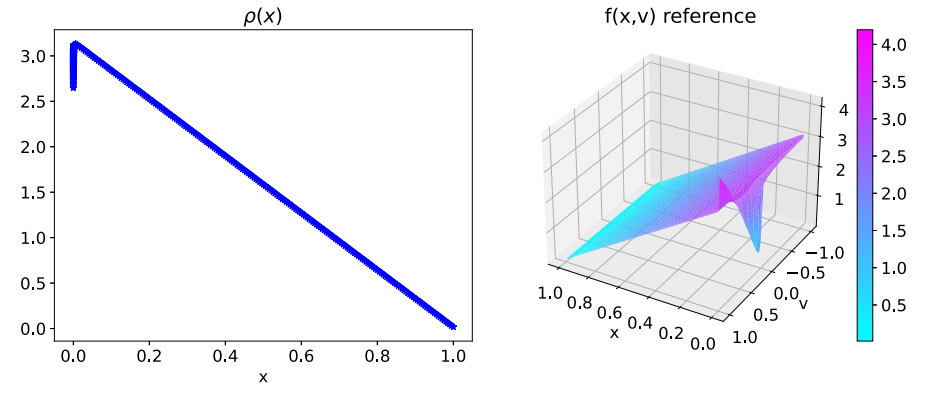

Fig. 4 Reference solution of (3.10) computed by a finite difference method with nonuniform mesh

with $\varepsilon = 10^{-3}$. Its macro-micro decomposition has the form

$$\begin{cases} \langle v\partial_x g\rangle = 0\,, \\ v\partial_x(\rho+\varepsilon g) = -g\,, \\ \rho(0)+\varepsilon g(0, v>0) = 5\sin(v), \quad \rho(1)+\varepsilon g(1, v<0) = 0\,. \end{cases} \tag{3.11}$$

On the left boundary at $x = 0$, one sees that, $\rho(0)$ takes a value independent of v, and leaves $\varepsilon g(0, v > 0)$ of $\mathcal{O}(1)$ magnitude, and therefore $g(0, v > 0)$ is of $\mathcal{O}(1/\varepsilon)$ magnitude. However, inside the domain, g is of order $\mathcal{O}(1)$ according to the second equation in (3.11), thus a sharp transition on g at the left boundary is expected (Fig. 4).

To see how such a sharp transition affects the neural network approximation, we now apply the loss function (3.7) to (3.11) to get

$$\begin{aligned} \mathcal{E}(f) = &\| \langle v\partial_x g\rangle \|^2_{L^2(\Omega)} + \|v\partial_x(\rho+\varepsilon g)+g\|^2_{L^2(\Omega)} \\ &+ B_{w,0}\int_0^1 (\rho(0)+\varepsilon g(0,v)-5\sin(v))^2 \mathrm{d}v + B_{w,1}\int_{-1}^0 (\rho(1)+\varepsilon g(1,v))^2\mathrm{d}v\,, \end{aligned} \tag{3.12}$$

and collect the results in Fig. 5 with $B_{w,0} = B_{w,1} = 1$. As displayed, the empirical loss remains large even after 50,000 iterations, and the f prediction is still far off the reference solution, which is plotted in Fig. 4 with finite difference method on a non-uniform mesh.

We noticed that, the dominated loss that hinders the convergence is the boundary loss in (3.12), due to the presence of boundary layer, which is intrinsically harder to approximate.

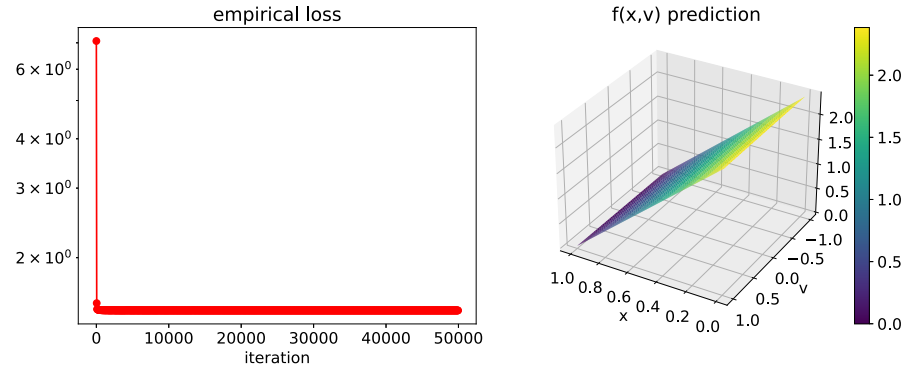

Fig. 5 Computation of (3.10) with (3.12) and $B_{w,0} = B_{w,1} = 1$ for $\varepsilon = 10^{-3}$. Here we use a fully connected neural network with 4 hidden layers and 50 neurons within each hidden layer, and $N_v^b = 60$, $N_x^r = 80$ and $N_v^r = 60$ in computing the empirical loss. The left is the empirical loss versus iteration, and right is the prediction of $f(x, v)$

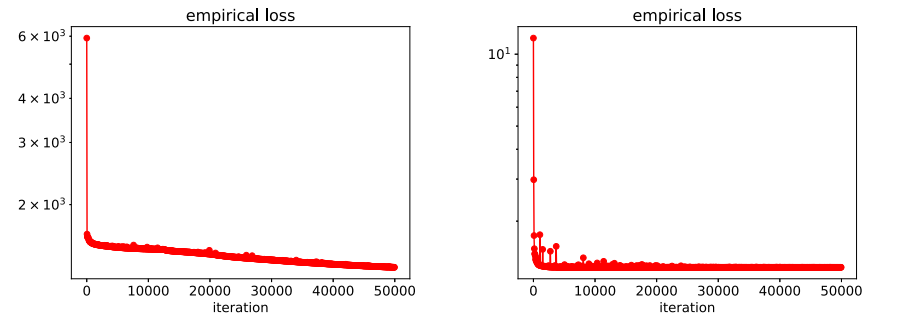

Fig. 6 Computation of (3.10) with $\varepsilon = 10^{-3}$, using neural network approximation with 4 hidden layers and 50 neurons within each hidden layer. Left: using the loss function (3.12) with $B_{w,0} = 10^3$ and $B_{w,1} = 1$. And $N_v^b = 60$, $N_x^r = 80$ and $N_v^r = 60$ in computing the empirical loss. Right: using the loss function (3.12) with $B_{w,0} = B_{w,1} = 1$. And $N_v^b = 60$, $N_{x1}^r = 150$ in $(0, \varepsilon)$, $N_{x2}^r = 50$ in $(\varepsilon, 1)$ and $N_v^r = 60$ in computing the empirical loss

Therefore, we tried to put more emphasize on the boundary term by increasing the weight $B_{w,0}$ from 1 to $1/\varepsilon = 10^3$, but the result is unfortunately barely improved, see the left plot of Fig. 6. A more sophisticated dynamics re-weighting [37,38] might improve the performance, but adjusting the weight appropriately seems to be very artificial and nontrivial. Another typical way of dealing with functions with sharp transition is to use non-uniform mesh and put more points near the fast transition region. This technique works well for grid based method, but not for our case. In fact, we have tried to assign 150 uniform points inside the boundary layer $[0, \varepsilon]$ (from the reference solution, we observe that the thickness of the boundary layer is ε in this example) and 50 points in the rest of the domain $(\varepsilon, 1]$, but the result is still unsatisfactory, see the right plot of Fig. 6.

Therefore, we propose a new decomposition that includes a boundary layer corrector. In particular, we decompose f as

$$f(\boldsymbol{x}, \boldsymbol{v}) = \tilde{\rho}(\boldsymbol{x}) + \varepsilon g(\boldsymbol{x}, \boldsymbol{v}) + \Gamma(\boldsymbol{x}, \boldsymbol{v}), \quad \text{with } \langle g \rangle = 0,\ \tilde{\rho}(\boldsymbol{x}) = \langle f(\boldsymbol{x}, \boldsymbol{v}) - \Gamma(\boldsymbol{x}, \boldsymbol{v}) \rangle .$$

Compared to (3.5), the main difference lies in the boundary layer corrector $\Gamma(\boldsymbol{x}, \boldsymbol{v})$, which is obtained by solving the half space problem. More precisely, consider a change of variable

Ψ_ε:

$$\Psi_\varepsilon(\boldsymbol{x}) : \boldsymbol{x} \in \Omega_x \subset \mathbb{R}^d \mapsto (z, \boldsymbol{x_b}),\ z \in [0, \infty),\ \boldsymbol{x_b} \in \partial\Omega_x,$$

where z is chosen according to (2.6), and thus Ψ_ε depends on ε. For instance, when $\varepsilon \to 0$ and $\boldsymbol{x} \notin \partial\Omega_x$, $z = \infty$. Let $f_{BL}(z, \boldsymbol{x_b}, \boldsymbol{v})$ be the solution to (2.2), then $\Gamma(\boldsymbol{x}, \boldsymbol{v})$ is obtained by

$$\Gamma(\boldsymbol{x}, \boldsymbol{v}) = f_{BL}(\Psi_\varepsilon(\boldsymbol{x}), \boldsymbol{v}) - f_{BL}(\Psi_{\varepsilon=0}(\boldsymbol{x}), \boldsymbol{v}).$$

Consequently, Γ carries over the sharp transition part of f, and leaves the remaining $\tilde{\rho}$ and g smooth and easily approximated by neural networks. In particular, $\tilde{\rho}$ and g solve

$$\begin{cases} \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} g \rangle + \frac{1}{\varepsilon} \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \Gamma \rangle = -\sigma_a(\tilde{\rho} + \langle \Gamma \rangle) + G, \\ \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}}(\tilde{\rho} + \Gamma + \varepsilon g) - \varepsilon \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} g \rangle - \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \Gamma \rangle \\ \quad = \sigma_s \mathcal{L} g + \frac{\sigma_s}{\varepsilon} \mathcal{L}\Gamma - \varepsilon^2 \sigma_a g - \varepsilon \sigma_a(\Gamma - \langle \Gamma \rangle), \\ \tilde{\rho} + \Gamma + \varepsilon g\big|_{\Gamma_-} = \phi. \end{cases} \tag{3.13}$$

In practice, Ψ_ε has an explicit form for domains with special geometry. Below we list special cases both in 1D and 2D domain.

3.3.1 $\Gamma(x, v)$ in 1D

Consider (1.1) in one dimensional domain with $x \in [0, 1]$ and $v \in [-1, 1]$:

$$\begin{cases} \varepsilon v \partial_x f = \sigma_s(x)\mathcal{L}(f) - \varepsilon^2 \sigma_a(x) f + \varepsilon^2 G(x), \\ f(0, v > 0) = \phi_L(v), \quad f(1, v < 0) = 0. \end{cases} \tag{3.14}$$

Without loss of generality, we only let boundary layer appears on the left, as the one on the right shall be treated in exactly the same way. Define the stretch variable $z = \frac{1}{\varepsilon}\int_0^x \sigma_s(s)\mathrm{d}s \in [0, \infty)$ and let $f_{BL}(z, v)$ solves

$$\begin{cases} v \partial_z f_{BL}(z, v) = \mathcal{L}(f_{BL}), \\ f_{BL}(0, v) = \phi_L(v),\ v \in (0, 1]. \end{cases} \tag{3.15}$$

Then Γ is obtained via

$$\Gamma(x, v) = f_{BL}\left(\frac{1}{\varepsilon}\int_0^x \sigma_s(s)\mathrm{d}s, v\right) - f_{BL}^\infty, \tag{3.16}$$

where $f_{BL}^\infty = \lim_{x\to\infty} f_{BL}(x, v)$.

In practice, we cannot solve (3.15) on an infinite domain. Instead, according to the Lemma 2.1 in [7], $f_{BL}(x, v)$ converges to the f_{BL}^∞ exponentially fast, it's sufficient to pick a large enough number Z and enforce (3.15) on $[0, Z] \times [-1, 1]$. Additionally, we impose the following condition

$$\langle v f_{BL}(z, \cdot) \rangle = 0, \quad z \in [0, Z]. \tag{3.17}$$

Indeed, taking average of (3.15), one gets $\partial_z \langle v f_{BL} \rangle = 0$, which implies that $\langle v f_{BL} \rangle$ is a constant. Noting from (2.7), f_{BL}^∞ is independent of v, hence $\langle v f_{BL}(\infty, v) \rangle = 0$. Therefore (3.17) generally holds. In sum, we utilize the following loss function to obtain the solution to (3.15):

$$\mathcal{E}(f_{BL}) = \|v\partial_z f_{BL}(z, v) - \mathcal{L}(f_{BL})\|^2_{L^2(\Omega)} + \|\langle v f_{BL} \rangle\|^2_{L^2(\Omega_z)} + \|f(0, v) - \phi_L(v)\|^2_{L^2(\Gamma_-)},$$

where $\Omega := [0, Z] \times [-1, 1]$ and $\Gamma_- = (0, 1]$.

Note that since f_{BL}^{∞} is a constant and f_{BL} solves (3.15), Γ obtained from (3.16) should satisfy

$$\varepsilon v \partial_x \Gamma = \sigma_s(x)\mathcal{L}(\Gamma),$$

and therefore the $(\tilde{\rho}, g)$ system (3.13) simplifies to

$$\begin{cases} \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} g \rangle = -\sigma_a(\tilde{\rho} + \langle \Gamma \rangle) + G, \\ \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}}(\tilde{\rho} + \varepsilon g) - \varepsilon \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} g \rangle = \sigma_s \mathcal{L} g - \varepsilon^2 \sigma_a g - \varepsilon \sigma_a(\Gamma - \langle \Gamma \rangle), \\ \tilde{\rho} + \Gamma + \varepsilon g|_{\Gamma_-} = \phi. \end{cases} \tag{3.18}$$

3.3.2 $\Gamma(x, v)$ in 2D square domain

As a second example, we consider a two dimensional square domain with $\boldsymbol{x} = (x, y) \in [-1, 1]^2$, $\boldsymbol{v} = (\cos\alpha, \sin\alpha)$, $\alpha \in [0, 2\pi]$. We assume that only the boundary $x = -1$ has a boundary layer and choose $\sigma_s(\boldsymbol{x}) = 1$, then the boundary value problem reads:

$$\begin{cases} \varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f = \mathcal{L}(f) - \varepsilon^2 \sigma_a(\boldsymbol{x}) f + \varepsilon^2 G(\boldsymbol{x}), \\ f(-1, y, \alpha) = \phi_L(y, \alpha), \ \alpha \in [0, \pi/2] \cup [3\pi/2, 2\pi], \\ f(1, y, \alpha) = 0, \ \alpha \in [\pi/2, 3\pi/2], \\ f(x, -1, \alpha) = 0, \ \alpha \in [0, \pi], \\ f(x, 1, \alpha) = 0, \ \alpha \in [\pi, 2\pi], \end{cases}$$

where $\mathcal{L}(f) = \langle f \rangle - f$ with $\langle f \rangle = \frac{1}{2\pi} \int_0^{2\pi} f \, \mathrm{d}\alpha$. In this case, we define the stretch variable z as

$$z = \frac{x + 1}{\varepsilon},$$

and solve $f_{BL}(z, y, \alpha)$ from

$$\begin{cases} \cos\alpha \, \partial_z f_{BL} = \mathcal{L}(f_{BL}), \\ f_{BL}(0, y, \alpha) = \phi_L(y, \alpha), \ \cos\alpha > 0. \end{cases} \tag{3.19}$$

Then the boundary layer corrector can be obtained as

$$\Gamma(x, y, \alpha) = f_{BL}(\frac{x + 1}{\varepsilon}, y, \alpha) - f_{BL}^{\infty}(y),$$

where $f_{BL}^{\infty}(y) = \lim_{x \to \infty} f_{BL}(x, y, \alpha)$.

As in the previous case, we do not solve (3.19) on infinite domain. Instead, we impose the zero flux condition

$$\langle \cos\alpha f_{BL}(z, y, \alpha) \rangle = \frac{1}{2\pi} \int_0^{2\pi} \cos\alpha f_{BL}(z, y, \alpha) \mathrm{d}\alpha = 0.$$

To implement, we place N_y grid points on y and denote them by y_j, $j = 1, \cdots, N_y$. Then for each fixed y_j, we use the following loss function to obtain $f_{BL}(z, y_j, \alpha)$:

$$\begin{aligned} \mathcal{E}(f_{BL}(\cdot, y_j, \cdot)) = & \| \cos\alpha \, \partial_z f_{BL}(\cdot, y_j, \cdot) - \mathcal{L}(f_{BL}(\cdot, y_j, \cdot)) \|_{L^2(\Omega)}^2 \\ & + \| \langle \cos\alpha f_{BL}(\cdot, y_j, \alpha) \rangle \|_{L^2(\Omega_z)}^2 + \| f_{BL}(0, y_j, \cdot) - \phi_L(y_j, \cdot) \|_{L^2(\Gamma_-)}^2, \end{aligned}$$

where $\Omega := \Omega_z \times [0, 2\pi] = [0, Z] \times [0, 2\pi]$ and $\Gamma_- = [0, \pi/2] \cup [3\pi/2, 2\pi]$. Consequently, since $\varepsilon \cos\alpha \, \partial_x \Gamma = \mathcal{L}(\Gamma)$, (3.13) in 2D is simplified to

$$\begin{cases} \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} g \rangle + \frac{1}{\varepsilon} \langle \sin\alpha \, \partial_y \Gamma \rangle = -\sigma_a(\tilde{\rho} + \langle \Gamma \rangle) + G, \\ \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}}(\tilde{\rho} + \varepsilon g) + \sin\alpha \, \partial_y \Gamma = \mathcal{L} g - \varepsilon \sigma_a(\tilde{\rho} + \varepsilon g + \Gamma) + \varepsilon G, \\ \tilde{\rho} + \Gamma + \varepsilon g|_{\Gamma_-} = \phi. \end{cases}$$

3.4 Algorithm

In this section, we include implementation details of our algorithm. For expository simplicity, we will describe it for one dimensional case, i.e., (3.14). The generalization to two dimensions is straightforward.

The first step is to obtain a neural network approximation $f_{BL}^{nn}(\theta; z, v)$ to the half space problem (3.15). To this end, we first generate the training set. For residual loss, given a sufficiently large number Z, assign N_z^r uniform grid points on $z \in [0, Z)$, i.e. $z_i^r = (i-1)\Delta z$ with $\Delta z = Z/N_z^r$. Since the physical dimension of v is at most two, to numerically evaluate the integration we choose N_v^r Gaussian quadrature points, which requires relatively smaller number of points and provide better accuracy than the uniform grid points and the randomly sample points. Denoted the Gaussian quadrature points as $\{v_j^r\}_{j=1}^{N_v^r}$, with corresponding weights $\{w_j^r\}_{j=1}^{N_v^r}$. For boundary loss function, we randomly sample N_v^b velocity points and denote them as $\{v_j^b\}_{j=1}^{N_v^b}$. Then the empirical loss function becomes

$$\begin{aligned}\mathcal{E}_{BL}^N(f_{BL}^{nn}(\theta)) = &\sum_{i=1}^{N_z^r}\sum_{j=1}^{N_v^r}(\partial_z f_{BL}^{nn}(\theta; z_i^r, v_j^r) - \mathcal{L} f_{BL}^{nn}(\theta; z_i^r, v_j^r))^2 w_j^r \Delta z \\ &+ \sum_{i=1}^{N_z^r}(\sum_{j=1}^{N_v^r} w_j^r v_j^r f_{BL}^{nn}(\theta; z_i^r, v_j^r))^2 \Delta z + \frac{1}{N_b}\sum_{j=1}^{N_v^b}(f_{BL}^{nn}(\theta; 0, v_j^b) - \phi_L(v_j^b))^2 .\end{aligned}$$

Here the second term on the right hand side corresponds to the condition (3.17). Minimizing over θ of $\mathcal{E}_{BL}^N(f_{BL}^{nn}(\theta))$, one gets

$$\theta_* = \arg\min_\theta \mathcal{E}_{BL}^N(f_{BL}^{nn}(\theta)), \tag{3.20}$$

and thus obtains $f_{BL}^{nn}(\theta_*; z, v)$. Here we summarize the procedure of training the neural network in Algorithm 1.

Algorithm 1: Algorithm for (3.20)

Input: Training set $\{z_i^r\}_{i=1}^{N_z^r}$, $\{v_j^r\}_{j=1}^{N_v^r}$, $\{w_j^r\}_{j=1}^{N_v^r}$, $\{v_j^b\}_{j=1}^{N_v^b}$; neural network parameters: number of hidden layer n_l, number of neurons in each layer n_r and activation function; two max iteration numbers I_{max1}, I_{max2}.
Output: θ_*
Initialize neural network;
Set $k_1 = 0, k_2 = 0$
while $k_1 < I_{max1}$ *and* $\mathcal{E}_{BL}^N(\theta^{k_1}) < \delta_1$ **do**
  Update θ^{k_1} by applying Adam to problem (3.20), $k_1 = k_1 + 1$;
end
Let $\theta^{k_2=0} = \zeta^{k_1}$;
while $k_2 < I_{max2}$ *and* $\nabla_\theta \mathcal{E}_{BL}^N(\theta^{k_2}) < \delta_2$ **do**
  Update θ^{k_2} by applying LBFGS to problem (3.20), $k_2 = k_2 + 1$;
end
$\theta_* = \theta^{k_2}$

After this, we denote $f_{BL}^\infty \approx f_{BL}^{nn}(\theta; Z, 0)$ as it is homogeneous in v and extend the function value of $f_{BL}^{nn}(\theta_*; z, v)$ as

$$f_{BL}^{nn}(\theta_*; z, v) = \begin{cases} f_{BL}^{nn}(\theta_*; z, v), & 0 \le z \le Z, \\ f_{BL}^\infty, & z > Z, \end{cases}$$

and compute the boundary layer corrector as follows:

$$\Gamma(x,v) = f_{BL}^{nn}(\theta_*; \frac{x}{\varepsilon}, v) - f_{BL}^{\infty}.$$

To proceed, we solve the following macro-micro system, which is tailored from (3.18) to adapt the specific boundary condition here

$$\begin{cases} \langle v\partial_x g\rangle = -\sigma_a\tilde{\rho} - \sigma_a\langle\Gamma\rangle + G, \\ v\partial_x(\tilde{\rho}+\varepsilon g) = \sigma_s\mathcal{L}(g) - \varepsilon\sigma_a(\tilde{\rho}+\varepsilon g+\Gamma) + \varepsilon G, \\ \tilde{\rho}(0)+\varepsilon g(0,v>0)+\Gamma(0,v>0) = \phi_L(v), \\ \tilde{\rho}(1)+\varepsilon g(1,v<0)+\Gamma(1,v<0) = 0. \end{cases}$$

As before, we first generate the training set, which consists of uniform grids in x and Gaussian quadrature point in v for residual loss, and random sample points in v for boundary loss. Once the empirical loss function is formed, applying the same optimization procedure, we obtain the predicted solution $\tilde{\rho}^{nn}$ and g^{nn}.

4 Theoretical analysis

We hereby provide a theoretical justification of our neural network formulation. In particular, we intend to show that the L^2 error of the predicted solution by neural network is *uniformly* bounded by the loss function. Let us first state a theorem that justifies the well-posedness of the (ρ, g)-system (3.6).

Theorem 3 *Let Assumptions 1 and 2 hold. Then the system (3.6) has a unique solution $(\rho, g) \in \mathcal{X}\times\mathcal{X}$ with $\langle g\rangle = 0$.*

Proof The existence of $(\rho, g) \in \mathcal{X}\times\mathcal{X}$ follows from Theorem 1. Indeed, let $f \in \mathcal{X}$ be the unique solution of (1.1). Then by construction, the pair $(\rho, g) := (\langle f\rangle, f - \langle f\rangle) \in \mathcal{X}\times\mathcal{X}$ solves (3.6) with $\langle g\rangle = 0$. Moreover, the uniqueness follows by tracking the proof of Lemma 1 (see the bound (4.7)). □

Next we proceed to show that our new (population) loss function $\mathcal{E}(f)$ defined in (3.7) satisfies a stability estimate, namely the L^2-error between the neural networks solution f and the exact solution f^* can be bounded above by $\mathcal{E}(f)$. Let $(\rho^*, g^*) \in \mathcal{X}\times\mathcal{X}$ be the solution to the macro-micro system (3.6) and $f^* = \rho^* + \varepsilon g^*$ be the exact solution to (1.1). Let $(\rho, g) \in \mathcal{X}\times\mathcal{X}$ be a neural network approximation to (ρ^*, g^*) and let $f = \rho + \varepsilon g$.

The the main theoretical result is as follows.

Theorem 4 *Let $(\rho, g) \in \mathcal{X}\times\mathcal{X}$. Then there exists a constant $C_\varepsilon > 0$ such that $\lim_{\varepsilon\downarrow 0} C_\varepsilon < \infty$ and that*

$$\|f - f^*\|^2_{L^2(\Omega)} \le \frac{C_\varepsilon}{\varepsilon}\mathcal{E}(f), \tag{4.1}$$

where $\mathcal{E}(f)$ is defined in (3.7). If in addition $\phi = \phi(x) \in H^{\frac{1}{2}}(\partial\Omega_{\boldsymbol{x}})$ and $\rho \in H^1(\Omega_{\boldsymbol{x}})$, then

$$\|f - f^*\|^2_{L^2(\Omega)} \le C_\varepsilon\mathcal{E}(f), \tag{4.2}$$

where again C_ε satisfies that $\lim_{\varepsilon\downarrow 0} C_\varepsilon < \infty$.

Remark 1 The estimate (4.2) of Theorem 4 shows that if the boundary data $\phi \in H^{\frac{1}{2}}(\partial\Omega_{\boldsymbol{x}})$ and the approximate solution $\rho \in H^1(\Omega_{\boldsymbol{x}})$, then the L^2-error between f and f^* can be

bounded by the loss $\mathcal{E}(f)$ uniformly in the regime where ε is small. It is worth to comment on the role of the above stability estimate in the numerical analysis of neural network methods for solving PDEs. In fact, from a practical perspective, an approximate solution is parameterized by neural networks f_θ^N and is obtained by minimizing the empirical loss $\mathcal{E}^N(f^{nn})$ (instead of $\mathcal{E}(f^{nn})$) with respect to the neural network parameters θ. Thanks to the well-established generalization theory of statistical learning [33], the difference between the population loss $\mathcal{E}(f_\theta^N)$ and the empirical loss $\mathcal{E}^N(f_\theta^N)$ (also known as the generalization gap) can be made arbitrarily small as both the number of quadrature points and the complexity of the neural network increase to infinity. As a result, the population loss $\mathcal{E}(f_\theta^N)$, and equivalently the L^2-error $\|f - f^*\|_{L^2(\Omega)}$ (thanks to Theorem 4), can be made small through minimizing the empirical loss $\mathcal{E}^N(f_\theta^N)$ via training. In another word, the stability bounds enable us to transfer the bound on trainable loss function to the solution. In the present paper, we only focus on the stability estimate and leave the complete generalization error analysis to the interested readers; such generalization analysis for neural networks has been carried out in the context of PDEs, see e.g. [27,28,30].

Proof of Theorem 4 Let us define $\tilde{\rho} = \rho - \rho^*, \tilde{g} = g - g^*$ and $\tilde{f} = f - f^*$. Then it is easy to verify that $(\tilde{\rho}, \tilde{g})$ solves the boundary value problems

$$\begin{aligned} \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \tilde{g} \rangle + \sigma_a \tilde{\rho} &=: r_1 && \text{in } \Omega, \\ \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} (\tilde{\rho} + \varepsilon \tilde{g}) - \varepsilon \langle \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \tilde{g} \rangle - \sigma_s \mathcal{L} \tilde{g} + \varepsilon^2 \sigma_a \tilde{g} &=: r_2 && \text{in } \Omega, \\ \tilde{\rho} + \varepsilon \tilde{g} &=: r_3 && \text{on } \Gamma_-. \end{aligned}$$

Then $\tilde{f}$ satisfies that

$$\begin{aligned} \varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} \tilde{f} &= \sigma_s(\boldsymbol{x}) \mathcal{L} \tilde{f}(\boldsymbol{x}, \boldsymbol{v}) - \varepsilon^2 \sigma_a \tilde{f} + \varepsilon^2 r_1(\boldsymbol{x}) + \varepsilon r_2(\boldsymbol{x}, \boldsymbol{v}) && \text{on } \Omega, \\ \tilde{f} &= r_3 && \text{on } \Gamma_-. \end{aligned}$$

Observe that by definition $\langle r_2 \rangle = \varepsilon^2 \sigma_a \langle g \rangle$. Then an application of Lemma 1 with $\eta = r_2 - \langle r_2 \rangle$ and $\xi = r_1 + \varepsilon^{-1} \langle r_2 \rangle = r_1 + \varepsilon \sigma_a \langle g \rangle$, the estimate (4.1) follows from (4.4). Furthermore, if $\phi = \phi(x) \in H^{\frac{1}{2}}(\partial \Omega_x)$ and $\rho \in H^1(\Omega)$, then on Γ_- one has $f(\boldsymbol{x}) = \rho(\boldsymbol{x}) - \phi(\boldsymbol{x}) + \varepsilon g(\boldsymbol{x}, \boldsymbol{x})$ with $\rho(\boldsymbol{x}) - \phi(\boldsymbol{x}) \in H^{\frac{1}{2}}(\partial \Omega_x)$. Therefore applying the estimate (4.5) of Lemma 1 leads to (4.2). □

Lemma 1 *Let $f \in H^1(\Omega)$ solve the problem*

$$\begin{aligned} \varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{v}) &= \sigma_s \mathcal{L} f(\boldsymbol{x}, \boldsymbol{v}) - \varepsilon^2 \sigma_a f + \varepsilon^2 \xi(\boldsymbol{x}) + \varepsilon \eta(\boldsymbol{x}, \boldsymbol{v}) && \text{on } \Omega, \\ f(\boldsymbol{x}, \boldsymbol{v}) &= \zeta(\boldsymbol{x}, \boldsymbol{v}) && \text{on } \Gamma_-, \end{aligned} \tag{4.3}$$

where $\xi \in L^2(\Omega_x)$, $\eta \in L^2(\Omega)$ with $\langle \eta \rangle = 0$ and $\zeta \in L^2(\Gamma_-)$. Then there exists a constant $C_\varepsilon > 0$ depending on $\varepsilon, \sigma_a, \sigma_s, C_K$ such that $\lim_{\varepsilon \downarrow 0} C_\varepsilon < \infty$ and that

$$\|f\|_{L^2(\Omega)}^2 \le C_\varepsilon (\|\xi\|_{L^2(\Omega_x)}^2 + \|\eta\|_{L^2(\Omega)}^2 + \varepsilon^{-1} \|\zeta\|_{L^2(\Gamma_-)}^2). \tag{4.4}$$

If in addition $\zeta = \zeta_1(\boldsymbol{x}) + \varepsilon \zeta_2(\boldsymbol{x}, \boldsymbol{v})$ where $\zeta_1 \in H^{\frac{1}{2}}(\partial \Omega_x)$ and $\zeta_2 \in L^2(\Gamma_-)$, then

$$\|f\|_{L^2(\Omega)}^2 \le C_\varepsilon (\|\xi\|_{L^2(\Omega_x)}^2 + \|\eta\|_{L^2(\Omega)}^2 + \|\zeta_1\|_{H^{\frac{1}{2}}(\partial \Omega_x)}^2 + \varepsilon \|\zeta_2\|_{L^2(\Gamma_-)}^2). \tag{4.5}$$

In particular, the stability constants in (4.5) are uniformly bounded in ε as $\varepsilon \downarrow 0$.

Proof Let us first prove the estimate (4.4). Multiplying (4.3) with f and the integrating on Ω leads to

$$\varepsilon\int_{\partial\Omega}(\boldsymbol{v},0)\cdot\boldsymbol{n}f^2\mathrm{d}s-\int_\Omega\sigma_s\mathcal{L}ff\,\mathrm{d}x\,\mathrm{d}\nu+\varepsilon^2\int_\Omega\sigma_a f^2\mathrm{d}x\,\mathrm{d}\nu=\varepsilon^2\int_\Omega\xi f\,\mathrm{d}x\,\mathrm{d}\nu$$
$$+\varepsilon\int_\Omega\eta f\,\mathrm{d}x\,\mathrm{d}\nu. \tag{4.6}$$

Notice from the definition of Γ_- that

$$(\boldsymbol{v},0)\cdot\boldsymbol{n}=\begin{cases}-|(\boldsymbol{v},0)\cdot\boldsymbol{n}| & \text{on }\Gamma_-,\\ |(\boldsymbol{v},0)\cdot\boldsymbol{n}| & \text{on }\Gamma_+.\end{cases}$$

Therefore we have from (4.6) that

$$\varepsilon\int_{\Gamma_+}|(\boldsymbol{v},0)\cdot\boldsymbol{n}|f^2\mathrm{d}s-\int_\Omega\sigma_s\mathcal{L}ff\,\mathrm{d}x\,\mathrm{d}\nu+\varepsilon^2\int_\Omega\sigma_a f^2\mathrm{d}x\,\mathrm{d}\nu$$
$$=\varepsilon\int_{\Gamma_-}|(\boldsymbol{v},0)\cdot\boldsymbol{n}|f^2\mathrm{d}s+\varepsilon^2\int_\Omega\xi f\,\mathrm{d}x\,\mathrm{d}\nu+\varepsilon\int_\Omega\eta f\,\mathrm{d}x\,\mathrm{d}\nu.$$

Thanks to part (3) of Assumption (2), the positivity of σ_s and the non-negativity of σ_a we have from above that

$$c\sigma_{\min}\|f-\langle f\rangle\|^2_{L^2(\Omega)}+\varepsilon^2\int_\Omega\sigma_a f^2\mathrm{d}x\,\mathrm{d}\nu\le\varepsilon\int_{\Gamma_-}|(\boldsymbol{v},0)\cdot\boldsymbol{n}|f^2\mathrm{d}s+\varepsilon^2\int_\Omega\xi f\,\mathrm{d}x\,\mathrm{d}\nu$$
$$+\varepsilon\int_\Omega\eta f\,\mathrm{d}x\,\mathrm{d}\nu. \tag{4.7}$$

Now let us write $f(\boldsymbol{x},\boldsymbol{v})=\rho(\boldsymbol{x})+\varepsilon g(\boldsymbol{x},\boldsymbol{v})$ with $\rho=\langle f\rangle$ and $g=\frac{1}{\varepsilon}(f-\langle f\rangle)$. Then (ρ,g) satisfy

$$\begin{aligned}\langle\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}}g\rangle&=-\sigma_a\rho+\xi && \text{on }\Omega,\\ \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}}(\rho+\varepsilon g)-\varepsilon\langle\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}}g\rangle-\sigma_s\mathcal{L}g&=\varepsilon^2\sigma_a g+\eta && \text{on }\Omega,\\ \rho+\varepsilon g&=\zeta && \text{on }\Gamma_-.\end{aligned} \tag{4.8}$$

By the assumption that $\langle\eta\rangle=0$, one has

$$\int_\Omega\eta f\,\mathrm{d}x\,\mathrm{d}\nu=\varepsilon\int_\Omega\eta g\mathrm{d}x\,\mathrm{d}\nu. \tag{4.9}$$

It follows from (4.7), (4.9) and Young's inequality that for $\alpha>0$,

$$c\sigma_{\min}\|g\|^2_{L^2(\Omega)}+\int_\Omega\sigma_a f^2\mathrm{d}x\,\mathrm{d}\nu\le\frac{1}{\varepsilon}\int_{\Gamma_-}|(\boldsymbol{v},0)\cdot\boldsymbol{n}|f^2\mathrm{d}s$$
$$+\frac{\|\xi\|^2_{L^2(\Omega)}+\|\eta\|^2_{L^2(\Omega)}}{4\alpha}+\alpha\|f\|^2_{L^2(\Omega)}$$
$$+\alpha\|g\|^2_{L^2(\Omega)}.$$

In particular, for any $\alpha\le\frac{c\sigma_{\min}}{2}$, we have

$$\frac{c\sigma_{\min}}{2}\|g\|^2_{L^2(\Omega)}+\int_\Omega\sigma_a f^2\mathrm{d}x\,\mathrm{d}\nu\le\frac{1}{\varepsilon}\int_{\Gamma_-}|(\boldsymbol{v},0)\cdot\boldsymbol{n}|f^2\mathrm{d}s$$
$$+\frac{\|\xi\|^2_{L^2(\Omega)}+\|\eta\|^2_{L^2(\Omega)}}{4\alpha}+\alpha\|f\|^2_{L^2(\Omega)}.$$

Now taking L^2-norm on the second line of (4.8) and applying Lemma 2, one obtains that

$$\begin{aligned}\|f\|^2_{L^2(\Omega)} &\le C_P\left(\|\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f\|^2_{L^2(\Omega)} + \int_{\Gamma_-}|(\boldsymbol{v},0)\cdot\boldsymbol{n}|f^2\mathrm{d}s\right)\\ &\le C_P\left(\varepsilon^2\|\xi-\sigma_a\rho\|^2_{L^2(\Omega)} + \|\varepsilon^2\sigma_a g+\eta\|^2_{L^2(\Omega)} + \|\sigma_s\mathcal{L}g\|^2_{L^2(\Omega)} + \|\zeta\|^2_{L^2(\Gamma_-)}\right)\\ &\le C_P\Big(2\varepsilon^2\|\xi\|^2_{L^2(\Omega)} + 2\varepsilon^2\|\sigma_a f\|^2_{L^2(\Omega)} + 4\varepsilon^4\|\sigma_a g\|^2_{L^2(\Omega)} + 2\|\eta\|^2_{L^2(\Omega)}\\ &\quad + \sigma^2_{\max}C_K^2\|g\|^2_{L^2(\Omega)} + \|\zeta\|^2_{L^2(\Gamma_-)}\Big)\end{aligned}$$

where in the second inequality we used the fact that $(\boldsymbol{v},0)\cdot\boldsymbol{n}\le 1$ since $|\boldsymbol{v}|=1$, and in the last inequality we have used part (5) of Assumption 2 and the fact that $\mathcal{L}\langle g\rangle = 0$. Combining the last two inequality and using the fact that $0\le\sigma_a\le\sigma_{\max}$, we obtain that

$$\begin{aligned}\|f\|^2_{L^2(\Omega)} \le C_P\Big(&2\varepsilon^2\|\xi\|^2_{L^2(\Omega)} + \Big(2\varepsilon^2\sigma_{\max} + \frac{2\sigma^2_{\max}(4\varepsilon^4+C_K^2)}{c\sigma_{\min}}\Big)\\ &\times\Big(\varepsilon^{-1}\|\zeta\|^2_{L^2(\Gamma_-)} + \frac{\|\xi\|^2_{L^2(\Omega)}+\|\eta\|^2_{L^2(\Omega)}}{4\alpha} + \alpha\|f\|^2_{L^2(\Omega)}\Big)\\ &+ 2\|\eta\|^2_{L^2(\Omega)} + \|\zeta\|^2_{L^2(\Gamma_-)}\Big).\end{aligned}$$

Setting

$$\alpha = \alpha^* := \Big(4\varepsilon^2\sigma_{\max} + \frac{4\sigma^2_{\max}(4\varepsilon^4+C_K^2)}{c\sigma_{\min}}\Big)^{-1}\wedge\frac{c\sigma_{\min}}{2}$$

in the above leads to

$$\begin{aligned}\|f\|^2_{L^2(\Omega)} \le\ & 2C_P\left(2\varepsilon^2 + \frac{1}{4\alpha^*}\Big(2\varepsilon^2\sigma_{\max} + \frac{2\sigma^2_{\max}(4\varepsilon^4+C_K^2)}{c\sigma_{\min}}\Big)\right)\|\xi\|^2_{L^2(\Omega)}\\ &+ 2C_P\left(2 + \frac{1}{4\alpha^*}\Big(2\varepsilon^2\sigma_{\max} + \frac{2\sigma^2_{\max}(4\varepsilon^4+C_K^2)}{c\sigma_{\min}}\Big)\right)\|\eta\|^2_{L^2(\Omega)}\\ &+ 2C_P\left(1 + \frac{1}{4\alpha^*}\Big(2\varepsilon^2\sigma_{\max} + \frac{2\sigma^2_{\max}(4\varepsilon^4+C_K^2)}{c\sigma_{\min}}\Big)\varepsilon^{-1}\right)\|\zeta\|^2_{L^2(\Gamma_-)}.\end{aligned}$$

This in particular implies (4.4).

Next we prove the improved estimate (4.5) when $\zeta = \zeta_1(\boldsymbol{x}) + \varepsilon\zeta_2(\boldsymbol{x},\boldsymbol{v})$ where $\zeta_1\in H^{\frac{1}{2}}(\partial\Omega_x)$ and $\zeta_2\in L^2(\Gamma_-)$. In fact, let us first decompose the solution as $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x},\boldsymbol{v})$, where f_1 solves the Laplace problem

$$\begin{aligned}\Delta f_1 &= 0, \qquad \text{on } \Omega_{\boldsymbol{x}},\\ f_1 &= \zeta_1(\boldsymbol{x}), \quad \text{on } \partial\Omega_{\boldsymbol{x}},\end{aligned}$$

where by the standard regularity estimate $\|f_1\|_{H^1(\Omega_{\boldsymbol{x}})}\le C_1\|\zeta_1\|_{H^{\frac{1}{2}}(\partial\Omega_x)}$ for some $C_1>0$, and $f_2 = f - f_1$ solves

$$\begin{aligned}\varepsilon\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f_2(\boldsymbol{x},\boldsymbol{v}) = \sigma_s\mathcal{L}f_2(\boldsymbol{x},\boldsymbol{v}) - \varepsilon^2\sigma_a(\boldsymbol{x})f_2(\boldsymbol{x},\boldsymbol{v}) - \varepsilon^2\sigma_a(\boldsymbol{x})f_1(\boldsymbol{x})\\ + \varepsilon^2\xi(\boldsymbol{x}) + \varepsilon\eta(\boldsymbol{x},\boldsymbol{v}) - \varepsilon\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f_1(\boldsymbol{x}) \qquad \text{on } \Omega,\\ f_2(\boldsymbol{x},\boldsymbol{v}) = \varepsilon\zeta_2(\boldsymbol{x},\boldsymbol{v}) \qquad \text{on } \Gamma_-.\end{aligned}$$

Applying the estimate (4.4) to the problem above and noticing that $\langle\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f_1(x)\rangle = 0$, we have

$$\begin{aligned}\|f_2\|^2_{L^2(\Omega)} &\le C_{2,\varepsilon}(\|\xi\|^2_{L^2(\Omega_x)} + \|\sigma_a f_1\|^2_{L^2(\Omega_x)} + \|\eta\|^2_{L^2(\Omega)}\\ &\quad + \|\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f_1\|^2_{L^2(\Omega)} + \varepsilon\|\zeta_2\|^2_{L^2(\Gamma_-)})\\ &\le \tilde{C}_{2,\varepsilon}(\|\xi\|^2_{L^2(\Omega_x)} + \|\eta\|^2_{L^2(\Omega)} + \|\zeta_1\|^2_{H^{\frac{1}{2}}(\partial\Omega_x)} + \varepsilon\|\zeta_2\|^2_{L^2(\Gamma_-)}).\end{aligned}$$

□

Let us recall the following directional Poincaré inequality from [29].

Lemma 2 (Directional Poincaré inequality) *There exists a constant C_P depending only on Ω such that*

$$\|f\|^2_{L^2(\Omega)} \le C_P \left(\|\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f\|^2_{L^2(\Omega)} + \int_{\Gamma_-} |(\boldsymbol{v},0)\cdot\boldsymbol{n}| f^2 \mathrm{d}s \right).$$

5 Numerical examples

In this section, we conduct extensive numerical experiments to verify the efficiency and accuracy of our neural network formulation based on macro-micro-(boundary layer) decomposition. For the structure of the neural network, we always use a fully connected network with n_l layers and n_r number of neurons within each layer. In the following examples, we use $\sigma^l(z) = \tanh(z)$ as the activation function of the hidden layer. For the activation function of the output layer, we use $\sigma^o_\rho(z) = \ln(1+e^z)$, $\sigma^o_g(z) = z$, and $\sigma^o_{f_{BL}}(z) = C_a/(1+e^{-z})$ for the macro part ρ, micro part g and boundary layer f_{BL}, respectively. Here C_a is tuned according to the L_∞ norm of the incoming boundary condition. For instance, $C_a = \|\phi_L(v)\|_\infty$ in solving (3.15). When training the neural network, as introduced in Algorithm 1, I_{max1}, δ_1, I_{max2}, δ_2 are the stopping parameters for Adam and LBFGS step, respectively. In 1D case, we choose $I_{max1} = 1.2\times10^4, \delta_1 = 0.005, I_{max2} = 10^4$, $\delta_2 = 10^{-6}$; in 2D case, we choose $I_{max1} = 2\times10^4$, $\delta_1 = 0.01$, $I_{max2} = 10^4$, $\delta_2 = 10^{-6}$. Unless otherwise specified, the learning rate for Adam step is fixed to be 10^{-3}.

Upon obtaining the neural network prediction $f^{nn} := \rho^{nn}(\boldsymbol{x}) + \varepsilon g^{nn}(\boldsymbol{x},\boldsymbol{v})$ or $f^{nn} := \tilde{\rho}^{nn}(\boldsymbol{x}) + \varepsilon g^{nn}(\boldsymbol{x},\boldsymbol{v}) + \Gamma^{nn}(\boldsymbol{x},\boldsymbol{v})$, we calculate its L^2 error to the reference solution as

$$error = \frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_v}(f^{nn}(\boldsymbol{x}_i,\boldsymbol{v}_j) - f^{ref}(\boldsymbol{x}_j,\boldsymbol{v}_j))^2 w_i w_j}{\sum_{i=1}^{N_x}\sum_{j=1}^{N_v}(f^{ref}(\boldsymbol{x}_j,\boldsymbol{v}_j))^2 w_i w_j}. \tag{5.1}$$

Here $\{\boldsymbol{x}_i, w_i\}$ and $\{\boldsymbol{v}_j, w_j\}$ are the test set and corresponding weight we use to calculate the reference solution, and therefore will be more refined than the training set. In particular, we again use the Gaussian quadrature for $\boldsymbol{v}$ and uniform mesh in $\boldsymbol{x}$ without boundary layer, or two sets of uniform mesh with boundary layer.

5.1 RTEs without boundary layers

In this subsection we consider RTEs in one and two dimensions where the solutions do not have boundary layers.

Example 1 1D problem with spatially homogeneous scattering:

$$\begin{cases} \varepsilon v\partial_x f = \langle f\rangle - f, \\ f(0, v>0) = 1, \quad f(1, v<0) = 0. \end{cases}$$

Using the loss function (3.7), we collect the results of $\varepsilon = 1$ and $\varepsilon = 10^{-3}$ in Fig. 7. Here we use $n_l = 4$, $n_r = 50$, $N^r_x = 80$, $N^r_v = 60$ and $N^b_v = 60$ for training. The reference solution is obtained by finite difference method on test set with $N_x = 200$, $N_v = 80$. It is evident that in both cases, the prediction obtained by the neural networks matches well with the reference solution, which is provided by a finite difference solver. Additionally, the relative L^2 error (5.1) is well controlled by the loss function.

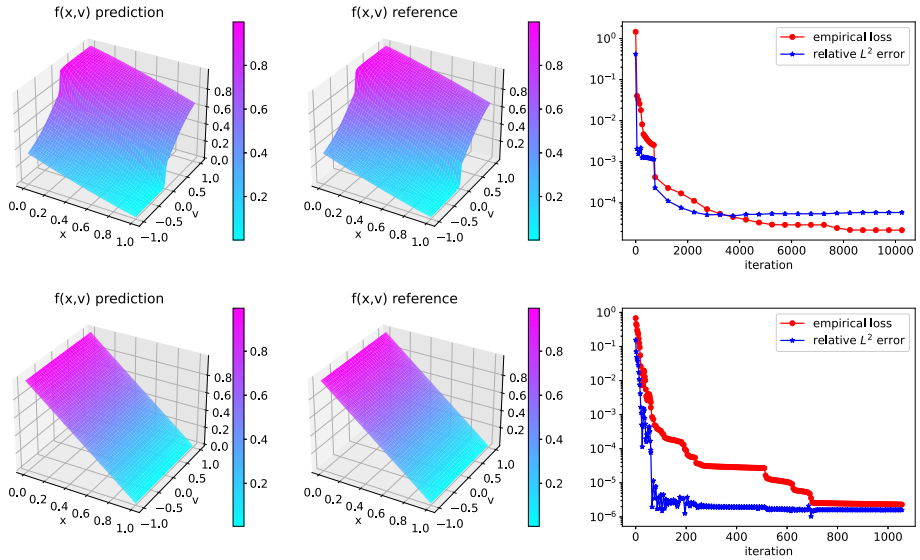

Fig. 7 Example 1 with $\varepsilon = 1$ in top row and $\varepsilon = 0.001$ in bottom row. The left column is $f(x, v)$ prediction, the middle column is the reference $f(x, v)$, the right column is the empirical loss and relative L^2 error to reference solution

Example 2 2D problem with $\boldsymbol{x} \in [-1, 1]^2$, $\boldsymbol{v} = (\cos\alpha, \sin\alpha)$:

$$\begin{cases} \varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f = \frac{1}{2\pi} \int_{|\boldsymbol{v}|=1} f(\boldsymbol{x}, \boldsymbol{v}) \mathrm{d}\boldsymbol{v}' - f + \varepsilon^2 G(\boldsymbol{x}, \boldsymbol{v}), \\ f(-1, y, \alpha) = e^{1-y}, \ \alpha \in [0, \pi/2] \cup [3\pi/2, 2\pi], \\ f(1, y, \alpha) = e^{-1-y}, \ \alpha \in [\pi/2, 3\pi/2], \\ f(x, -1, \alpha) = e^{1-x}, \ \alpha \in [0, \pi], \\ f(x, 1, \alpha) = e^{-1-x}, \ \alpha \in [\pi, 2\pi], \end{cases}$$

where $G(x, y, \alpha) = \frac{1}{\varepsilon}(-\cos\alpha - \sin\alpha)e^{-x-y}$. This problem has an analytic solution $f(x, y, \alpha) = e^{-x-y}$. The numerical solutions for $\varepsilon = 1$ and $\varepsilon = 10^{-3}$ are presented in Fig. 8, where the reference solution is the above analytic form. In both cases, the numerical parameters we use are: $n_l = 4$, $n_r = 30$, $N_x^r = 40$, $N_y^r = 40$, $N_v^r = 40$, $N_v^b = 40$, $N_x^b = 40$, $N_y^b = 40$ and $N_v^b = 40$ for training.

Example 3 1D problem with spatially heterogeneous scattering:

$$\begin{cases} v\partial_x f = \sigma(x)(\langle f \rangle - f), \\ f(0, v > 0) = 5, \qquad f(1, v < 0) = 0, \end{cases}$$

where σ is a smooth varying function $\sigma(x) = 1 + b/e^{-a(x-0.5)}$.

In practice, we rewrite the equation as

$$\varepsilon(x) v \partial_x f = \langle f \rangle - f, \quad \text{with} \quad \varepsilon(x) = \frac{1 + e^{-a(x-0.5)}}{b + 1 + e^{-a(x-0.5)}}.$$

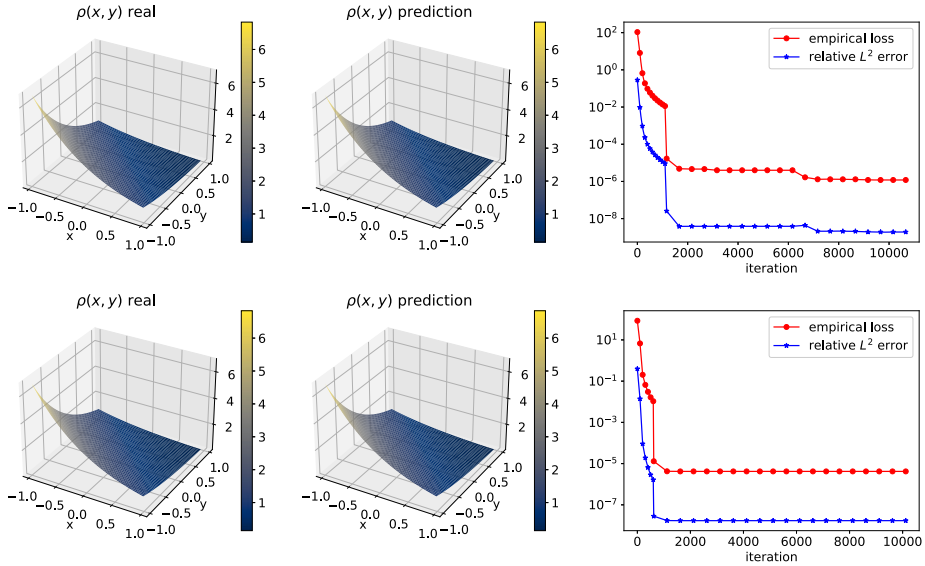

Fig. 8 Example 2 with $\varepsilon = 1$ in top row and $\varepsilon = 0.001$ in bottom row. The left column is prediction of $\rho(x, y)$. The middle column is analytic $\rho(x, y)$. Right column is empirical loss and relative L^2 error

Accordingly, our macro-micro decomposition reads $f(x, v) = \rho(x) + \varepsilon(x)g(x, v)$, where ρ and g solve

$$\begin{cases} \langle v\partial_x(\varepsilon(x)g)\rangle = 0, \\ v\partial_x(\rho + \varepsilon(x)g) + g = 0, \\ \rho(0) + \varepsilon(0)g(0, v > 0) = 5, \quad \rho(1) + \varepsilon(1)g(1, v < 0) = 0. \end{cases}$$

Choosing $a = 10$ and $b = 20$, we plot the shape of $\sigma(x)$ and gather the corresponding numerical solutions in Fig. 9. Here the neural network is constructed using $n_l = 4, n_r = 50$, $N_x^r = 80$, $N_v^r = 60$, $N_v^b = 60$; and trained with initial learning rate 0.001 for Adam and decrease by a factor of 0.95 after every 2000 steps. The reference solution is obtained by a finite difference method with $N_x = 200$ and $N_v = 80$. As expected, a good match to the reference solution is observed, and a good control of relative L^2 error is obtained.

Example 4 2D RTE with an-isotropic scattering:

$$\begin{cases} \varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f = \int_{|\boldsymbol{v}|=1} K(\boldsymbol{v}, \boldsymbol{v}') f(\boldsymbol{x}, \boldsymbol{v}) \mathrm{d}\boldsymbol{v}' - f, \quad \boldsymbol{x} \in [-1, 1]^2, \boldsymbol{v} = (\cos\alpha, \sin\alpha) \\ f(-1, y, \alpha) = (1 - y^2), \ \alpha \in [0, \pi/2] \cup [3\pi/2, 2\pi] \\ f(1, y, \alpha) = 0, \ \alpha \in [\pi/2, 3\pi/2] \\ f(x, -1, \alpha) = 0, \ \alpha \in [0, \pi] \\ f(x, 1, \alpha) = 0, \ \alpha \in [\pi, 2\pi], \end{cases}$$

with Henyey-Greenstein scattering kernel:

$$K(\boldsymbol{v}, \boldsymbol{v}') = \frac{1 - h^2}{2\pi(1 + h^2 - 2h\boldsymbol{v} \cdot \boldsymbol{v}')}, \quad h \in (0, 1).$$

The numerical solutions with $\varepsilon = 1$ and $\varepsilon = 0.001$ are presented in Fig. 10. Here the neural network is constructed with $n_l = 4$, $n_r = 30$, $N_x^r = 40$, $N_y^r = 40$, $N_v^r = 40$,

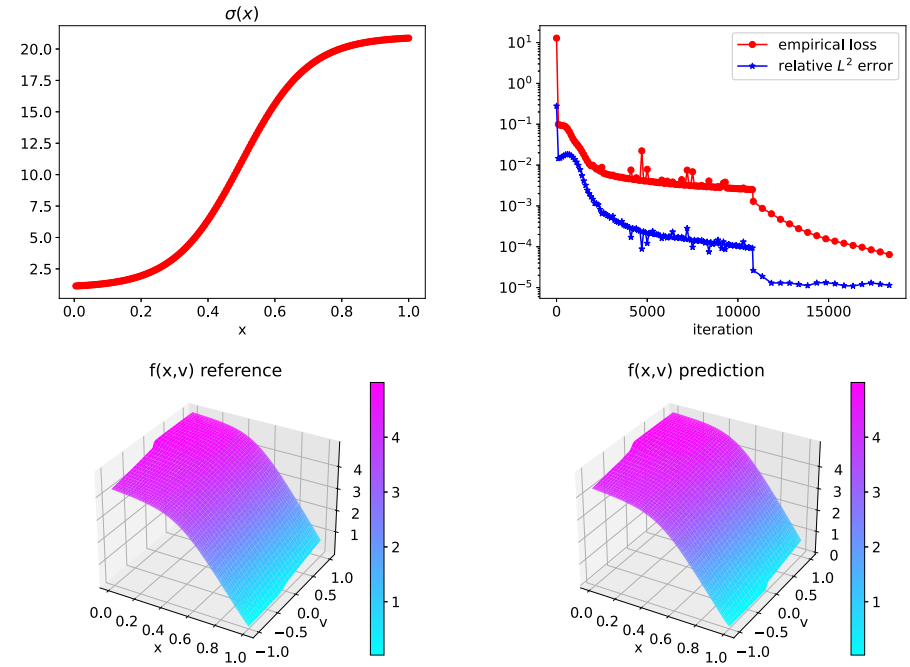

Fig. 9 Example 3 with $a = 10$ and $b = 20$. The top left is the plot of σ, top right is the empirical loss and relative L^2 error, bottom left is referenced $f(x, v)$, and bottom right is the predicted $f(x, v)$ by neural network

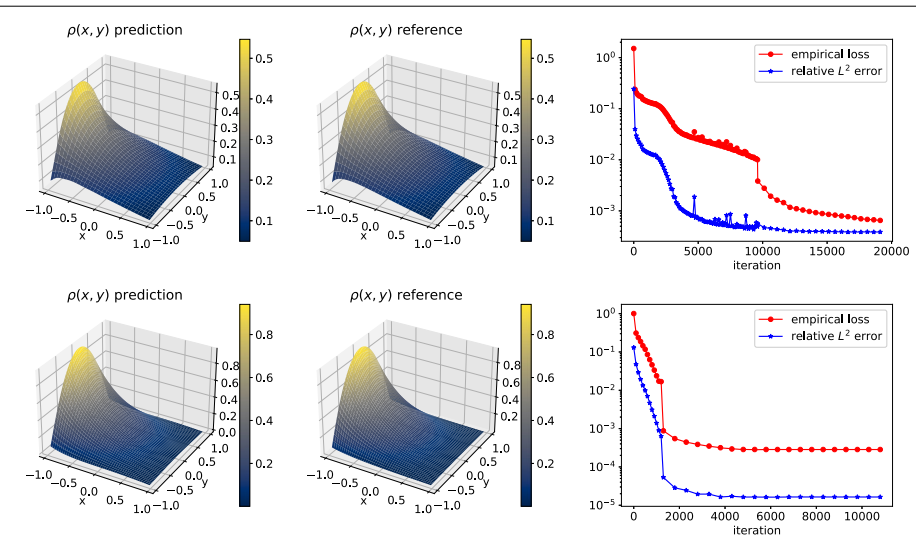

Fig. 10 Top and bottom are Example 4 with $\varepsilon = 1$ and $\varepsilon = 0.001$ respectively

$N_v^b = 40$, $N_x^b = 40$, $N_y^b = 40$ and $N_v^b = 40$; and the reference solution is obtained by a finite difference method with $N_x = 60$, $N_y = 60$, $N_v = 40$.

5.2 1D RTE with boundary layer

Here we consider a one dimensional example with velocity dependent boundary condition. With a velocity-dependent boundary term, a boundary layer is present in the solution of RTE when ε is small. To find such solution, we first solve the half space problem.

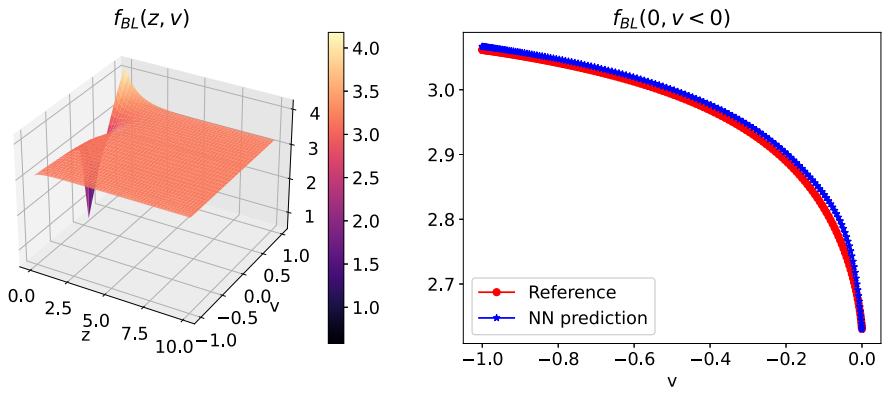

Fig. 11 Example 5, here we use $nl = 4$, $nr = 50$, $N_z^r = 400$, $N_v^r = 40$ and $N_v^b = 60$. The left is $f_{BL}(x, v)$ prediction, and the right is the comparison between (5.4) and the prediction by Neural network. The numerical approximation of the limit constant $f_{BL}^{nn}(10, 0) \approx 3.1919$ and the exact $f_{BL}^{\infty} \approx 3.1889$

Example 5 1D half space problem:

$$\begin{cases} v\partial_z f_{BL}(z, v) = \langle f_{BL} \rangle - f_{BL}, \\ f_{BL}(0, v) = 5\sin(v). \end{cases} \tag{5.2}$$

For this problem, we know that its solution $f_{BL}(z, v)$ admits an analytical limit

$$f_{BL}^{\infty} = \frac{\sqrt{3}}{2} \int_0^1 5\sin(v) H(v) v \mathrm{d}v \tag{5.3}$$

from the Chandrasekhar H-function, which satisfies

$$\frac{1}{H(v)} = \int_0^1 \frac{H(w)}{2(v + w)} w \mathrm{d}w.$$

Additionally, the reflection boundary condition has the form

$$f_{BL}(0, v < 0) = \frac{1}{2} H(v) \int_0^1 5\sin(w) \frac{H(w)}{w + v} w \mathrm{d}w. \tag{5.4}$$

In Fig. 11, we plot the numerical solution to (5.2), and compare its reflection boundary with (5.4) with good agreement.

Example 6 We then proceed to solve the transport equation:

$$\begin{cases} \varepsilon v \partial_x f = \langle f \rangle - f, \\ f(0, v > 0) = 5\sin v, \quad f(1, v < 0) = 0. \end{cases} \tag{5.5}$$

When $\varepsilon = 1$, we obtain the neural network prediction by using the macro-micro decomposition. The results are collected in Fig. 12, where the reference solution is obtained by solving (5.5) with a finite difference method on a uniform mesh. On the other hand, when $\varepsilon = 10^{-3}$, we obtain the neural network prediction by the macro-micro-boundary layer decomposition. For comparison, we construct two reference solution. One is obtained by the same finite difference method but on a non-uniform mesh in x, with 150 points in $[0, \varepsilon)$ and 50 points in $[\varepsilon, 1]$. The other is obtained by solving the diffusion limit, whose boundary condition is computed via the H-function (5.3). In this specific example, we have $f_{BL}^{\infty} = 3.188$, and therefore the limit density is $\rho_0(x) = 3.188(1 - x)$. The comparisons with good agreement are displayed in Fig. 13.

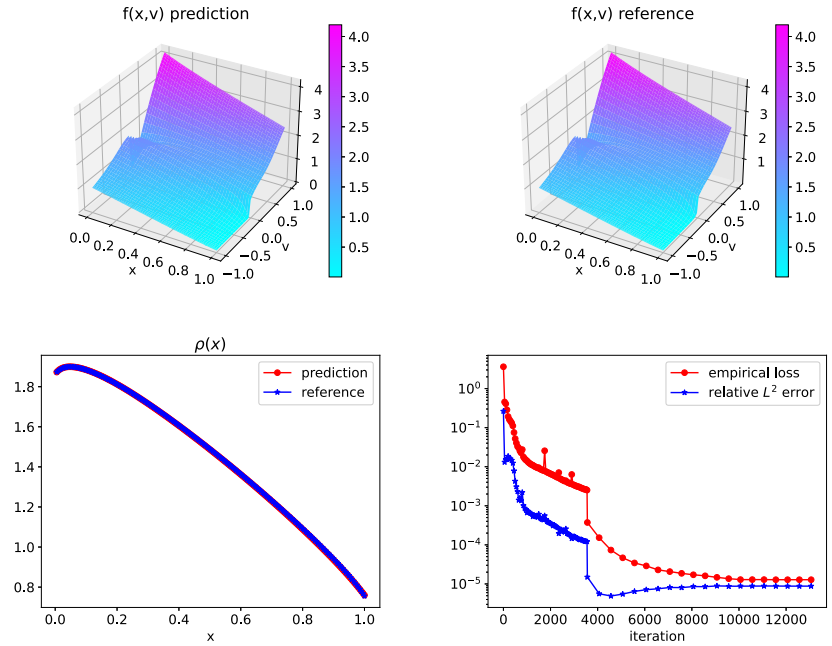

Fig. 12 Example 6 with $\varepsilon = 1$. we use $nl = 4$, $nr = 50$, $N_x^r = 80$, $N_v^r = 60$ and $N_v^b = 60$ for training. Compute the reference solution by finite difference method on test set with $N_x = 200$, $N_v = 80$. The top left is $f(x, v)$ prediction, the top right is the reference $f(x, v)$, bottom left is comparison of $\rho(x)$ and bottom right is the empirical loss and relative L^2 error vs number of iterations

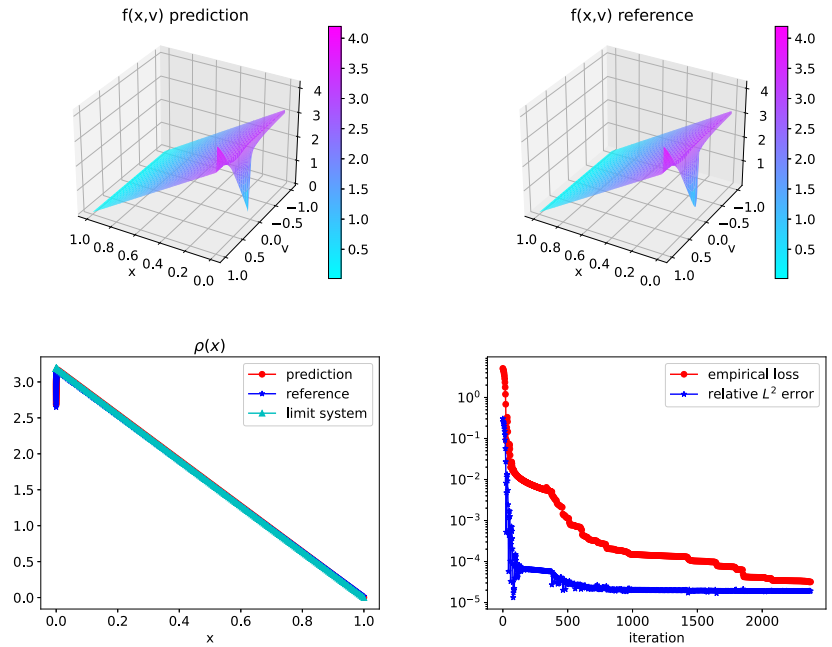

Fig. 13 Example 6 with $\varepsilon = 10^{-3}$, we use $n_l = 4$, $n_r = 50$, $N_x^r = 80$, $N_v^r = 60$ and $N_v^b = 60$ for training. The top left is $f(x, v)$ prediction, top right is the reference $f(x, v)$, bottom left is comparison of $\rho(x)$ and bottom right is the empirical loss and relative L^2 error vs iteration number

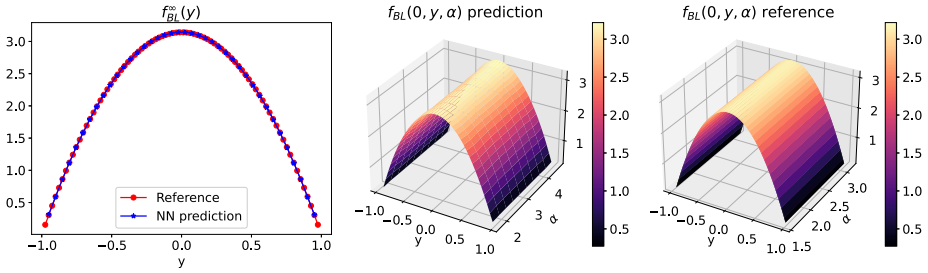

Fig. 14 Example 7. Left: plot of $f_{BL}^{\infty}(y)$. Middle and right are plots of $f_{BL}(0, y, \alpha)$ for $\alpha \in [0, 2\pi]$.The reference solution are obtained from the formula (5.6) and (5.7)

5.3 RTE in two dimensions with boundary layer

As with Sect. 5.2, we first solve the half space problem and then the corresponding RTE.

Example 7 2D half space problem: for $z \in [0, \infty)$ and $y \in [-1, 1]$

$$\begin{cases} \cos\alpha\, \partial_z f_{BL}(z, y, \alpha) = \langle f_{BL} \rangle - f_{BL}\,, \quad \langle f_{BL} \rangle = \frac{1}{2\pi} \int_0^{2\pi} f_{BL} \mathrm{d}\alpha\,, \\ f_{BL}(0, y, \alpha) = (1 - y^2)\alpha, \quad \cos\alpha < 0\,. \end{cases}$$

Then it admits a limit (see formula (7.3) in [7]):

$$f_{BL}^{\infty}(y) := \lim_{z \to \infty} f_{BL}(z, y, \alpha) = \frac{1}{\sqrt{\pi}} \int_{\Gamma_-} (1 - y^2)\alpha \cos\alpha H(\alpha) \mathrm{d}\alpha\,, \tag{5.6}$$

where H is the Chandrasekhar H-function that satisfies

$$\frac{1}{H(\alpha)} = \int_{\Gamma_-} \frac{H(\xi)}{\cos\alpha + \cos\xi} \cos\xi \mathrm{d}\xi\,,$$

and $\Gamma_- = [0, \pi/2] \cup [3\pi/2, 2\pi]$.

Additionally, we can get the reflection boundary condition at $x = 0$. Since the reflected velocity $\tilde{\boldsymbol{v}}$ at boundary is

$$\tilde{\boldsymbol{v}} = \boldsymbol{v} - 2(\boldsymbol{v} \cdot \boldsymbol{n_x})\boldsymbol{n_x}\,, \quad \boldsymbol{v} = (\cos\alpha, \sin\alpha)\,,$$

and in our case $\boldsymbol{n_x} = (-1, 0)$, we have $\tilde{\boldsymbol{v}} = (-\cos\alpha, \sin\alpha)$. Then

$$\begin{cases} f_{BL}(0, y, \pi - \alpha) \\ \quad = \int_{\Gamma_-} \xi(1 - y^2) \cos\xi H(\xi) H(\alpha)/(\cos\alpha + \cos\xi) \mathrm{d}\xi\,, \quad \alpha \in [0, \pi/2]\,, \\ f_{BL}(0, y, 3\pi - \alpha) \\ \quad = \int_{\Gamma_-} \xi(1 - y^2) \cos\xi H(\xi) H(\alpha)/(\cos\alpha + \cos\xi) \mathrm{d}\xi\,, \quad \alpha \in [3\pi/2, 2\pi]\,. \end{cases} \tag{5.7}$$

In Fig. 14, we plot the numerical prediction from the neural network approximation with parameters $n_l = 3$, $n_r = 50$, $N_x = 200$, $N_y = 50$ and $N_v = 40$, and compared it with the reference solution (5.6) and (5.7).

Example 8 We then move on to solve the 2D transport equation, $\boldsymbol{x} \in [-1, 1]^2$, $\boldsymbol{v} = (\cos\alpha, \sin\alpha)$, $\alpha \in [0, 2\pi]$:

$$\begin{cases} \varepsilon \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f = \frac{1}{2\pi} \int_{|\boldsymbol{v}|=1} f(\boldsymbol{x}, \boldsymbol{v}) \mathrm{d}\boldsymbol{v}' - f\,, \\ f(-1, y, \alpha) = (1 - y^2)\alpha, \ \alpha \in [0, \pi/2] \cup [3\pi/2, 2\pi]\,, \\ f(1, y, \alpha) = 0, \ \alpha \in [\pi/2, 3\pi/2]\,, \\ f(x, -1, \alpha) = 0, \ \alpha \in [0, \pi]\,, \\ f(x, 1, \alpha) = 0, \ \alpha \in [\pi, 2\pi]. \end{cases}$$

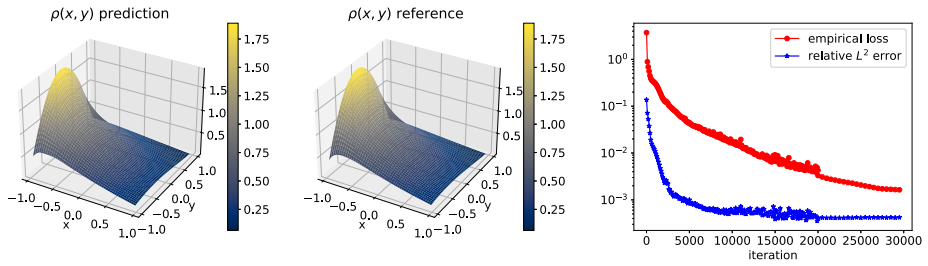

Fig. 15 Example 8 with $\varepsilon = 1$

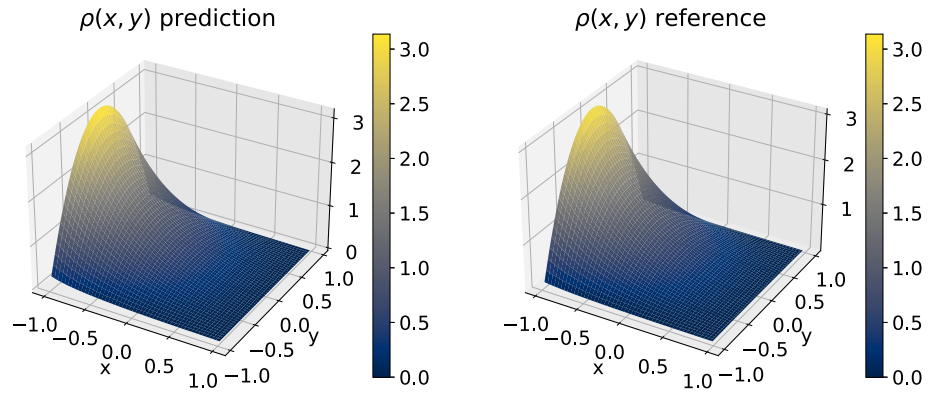

Fig. 16 Example 8 with $\varepsilon = 10^{-3}$

When $\varepsilon = 1$, we use the macro-micro decomposition based PINN, and when $\varepsilon = 10^{-3}$, we include a boundary layer corrector which is computed in Example 7. In both cases, the numerical parameters are $n_l = 4$, $n_r = 30$, $N_x^r = 40$, $N_y^r = 40$, $N_v^r = 40$, $N_v^b = 40$, $N_x^b = 40$, $N_y^b = 40$ and $N_v^b = 40$ for training. As a comparison, we use a finite difference method with uniform grid $N_x^r = 60$, $N_y^r = 60$, $N_v^r = 60$, for $\varepsilon = 1$. For $\varepsilon = 10^{-3}$, we solve the diffusion limit

$$\begin{cases} \Delta\rho = 0, \\ \rho(-1, y) = \pi(1 - y^2), \\ \rho(1, y) = \rho(x, -1) = \rho(x, 1) = 0. \end{cases}$$

Here the boundary condition $\rho(-1, y)$ is obtained from (5.6). The numerical results are presented in Figs. 15 and 16.

5.4 Nonlinear RTE

At last, we consider an example of nonlinear RTE in one dimension.

Example 9 For $x \in [0, 1]$, $v \in [-1, 1]$

$$\begin{cases} \varepsilon v \partial_x I(x, v) = \sigma(acT^4(x) - I(x, v)), \\ \varepsilon^2 \partial_{xx} T(x) = \sigma(acT^4(x) - \langle I(x, v) \rangle), \\ I(0, v > 0) = 1, \ I(1, v < 0) = 0, \\ T(0) = 1, \ T(1) = 0, \end{cases} \tag{5.8}$$

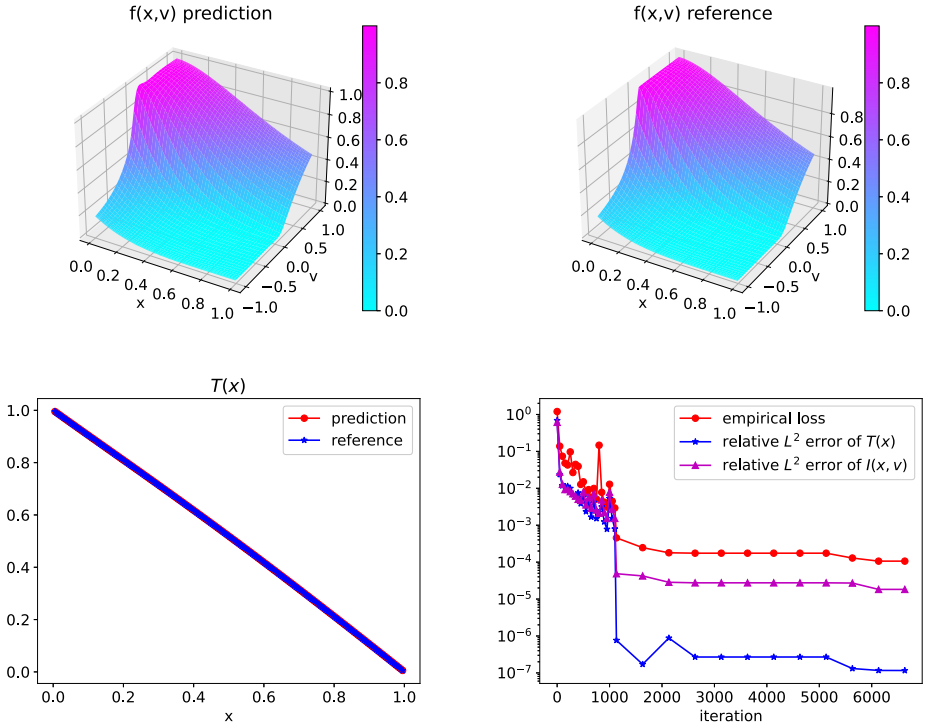

Fig. 17 Example 9 with $\varepsilon = 1$. Top left: $I(x, v)$ prediction. Top right: reference $I(x, v)$. Bottom left is comparison of $T(x)$. Bottom: the empirical loss and relative L^2 errors to reference solutions

where a, c and σ are three constants.

For more details on nonlinear RTE, please refer to [25]. When $\varepsilon \to 0, I$ and T will converge to I_0 and T_0, which satisfy

$$\begin{cases} \frac{ac}{3\sigma}\partial_{xx}T_0^4 + \partial_{xx}T_0 = 0, \\ T_0(0) = 1,\ T_0(1) = 0. \end{cases} \tag{5.9}$$

To solve (5.8), we again conduct the macro-micro decomposition for I:

$$I = \rho(x) + \varepsilon g(x, v), \quad \text{where } \rho(x) = \langle I \rangle,\ \langle g \rangle = 0.$$

Then the corresponding decomposed system reads:

$$\begin{cases} \langle v\partial_x g\rangle = \partial_{xx}T, \\ v\partial_x(\rho + \varepsilon g) - \varepsilon\partial_{xx}T = -\sigma g, \\ \varepsilon^2\partial_{xx}T = \sigma acT^4 - \sigma\rho, \\ \rho(0) + \varepsilon g(0, v > 0) = 1, \quad \rho(1) + \varepsilon g(1, v < 0) = 0, \\ T(0) = 1, \quad T(1) = 0. \end{cases}$$

As a result, the loss function has the form:

$$\begin{aligned} \mathcal{E}(I, g, T) = &\| \langle v\partial_x g\rangle - \partial_{xx}T\|^2_{L^2(\Omega_x)} + \|v\partial_x(\rho + \varepsilon g) - \varepsilon\partial_{xx}T + \sigma g\|^2_{L^2(\Omega)} \\ &+ (T(0) - 1)^2 + T(1)^2 + \|\varepsilon^2\partial_{xx}T - \sigma acT^4 + \sigma\rho\|^2_{L^2(\Omega_x)} + \int_0^1 (\rho(0) \\ &+ \varepsilon g(0, v) - 1)^2 \mathrm{d}v + \int_{-1}^0 (\rho(1) + \varepsilon g(1, v))^2 \mathrm{d}v. \end{aligned}$$

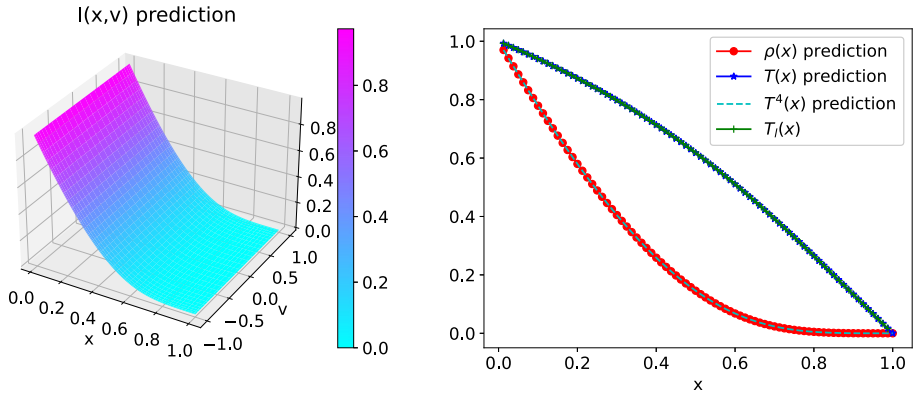

Fig. 18 Example 9 with $\varepsilon = 10^{-3}$

We then train the neural network with $n_l = 4$ and $n_r = 50$, using $N_x^r = 80$, $N_v^r = 60$ and $N_v^b = 60$ to generate training set. When $\varepsilon = 1$, we compute the reference solution by a finite difference method on a uniform grid with $N_x = 200$ and $N_v = 80$. When $\varepsilon = 10^{-3}$, we solve the limit system (5.9) instead to get the reference solution. The results are collected in Figs. 17 and 18, respectively.

6 Conclusion

In this paper, we develop a numerical scheme based on PINNs for steady RTE with diffusive scaling. As illustrated in Sect. 3.1, vanilla PINNs suffer from the instability issue when ε is small, and our major contribution is to resolve this issue by proposing an novel empirical loss function based on the micro macro decomposition. More importantly, a rigorous uniform stability result is established. We prove that the L^2-error of the PINNs prediction can be bounded by the aforementioned new empirical loss function uniformly in ε. When ε is small and an an-isotropic boundary condition is considered, a boundary layer is expected and the neural network is hence hard to converge. We construct a boundary layer corrector based on the solution of the associated half space problem, which encodes the sharp transition information within the boundary layer and leaves the rest part of solution smooth and thus can be easily approximated. Extensive numerical results demonstrate the effectiveness of our novel numerical methods.

Acknowledgements
Y.L. thanks the US National Science Foundation for its support through the award DMS-2107934. L.W. and W.X. thank the National Science foundation for its support through the award DMS-1846854. The authors also acknowledge the Minnesota Supercomputing Institute (MSI) at the University of Minnesota for providing resources that contributed to the research results reported within this paper.

Data availability statement
Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Author details
[1]Department of Mathematics and Statistics, Lederle Graduate Research Tower, University of Massachusetts, 710 N. Pleasant Street, Amherst, MA 01003, USA, [2]School of Mathematics, University of Minnesota, Twin Cities, MN 55455, USA.

Received: 10 December 2021 Accepted: 20 June 2022 Published online: 15 July 2022

References

1. Bardos, C., Santos, R., Sentis, R.: Diffusion approximation and computation of the critical size. Trans. Am. Math. Soc. **284**, 617–649 (1984)
2. Bensoussan, A., Lions, P.-L., Papanicolaou, G.C.: Boundary layers and homogenizatlon of transport processes. Publ. Res. Inst. Math. Sci. **15**, 53–157 (1979)
3. Boscarino, S., Pareschi, L., Russo, G.: Implicit-explicit Runge–Kutta scheme for hyperbolic systems and kinetic equations in the diffusion limit. SIAM J. Sci. Comput. **35**, 22–51 (2013)
4. Chen, Z., Liu, L., Mu, L.: Solving the linear transport equation by a deep neural network approach, arXiv preprint arXiv:2102.09157 (2021)
5. Dimarco, G., Pareschi, L.: Numerical methods for kinetic equations. Acta Numer. **23**, 369–520 (2014)
6. Egger, H., Schlottbom, M.: An lp theory for stationary radiative transfer. Appl. Anal. **93**, 1283–1296 (2014)
7. Golse, F., Jin, S., Levermore, C.D.: A domain decomposition analysis for a two-scale linear transport problem. ESAIM: Math. Model. Numer. Anal. **37**, 869–892 (2003)
8. Han, H., Tang, M., Ying, W.: Two uniform tailored finite point schemes for the two dimensional discrete ordinates transport equations with boundary and interface layers. Commun. Comput. Phys. **15**, 797–826 (2014)
9. Han, J., Jentzen, A., et al.: Algorithms for solving high dimensional pdes: from nonlinear monte carlo to machine learning. arXiv preprint arXiv:2008.13333 (2020)
10. Han, J., Jentzen, A., Weinan, E.: Solving high-dimensional partial differential equations using deep learning. Proc. Natl. Acad. Sci. **115**, 8505–8510 (2018)
11. Hwang, H.J., Jang, J.W., Jo, H., Lee, J.Y.: Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network approach. J. Comput. Phys. **419**, 109665 (2020)
12. Jin, S.: Asymptotic preserving (AP) schemes for multiscale kinetic and hyperbolic equations: a review. Riv. Mat. Univ. Parma **2**, 177–216 (2012)
13. Jin, S., Ma, Z., Wu, K.: Asymptotic-preserving neural networks for multiscale time-dependent linear transport equations, arXiv preprint arXiv:2111.02541 (2021)
14. Jin, S., Pareschi, L., Toscani, G.: Uniformly accurate diffusive relaxation schemes for multiscale transport equations. SIAM J. Numer. Anal. **38**, 913–936 (2000)
15. Jin, S., Tang, M., Han, H.: A uniformly second order numerical method for the one-dimensional discrete-ordinate transport equation and its diffusion limit with interface. Netw. Heterogeneous Media **4**, 35–65 (2009)
16. Klar, A.: Asymptotic-induced domain decomposition methods for kinetic and drift diffusion semiconductor equations. SIAM J. Sci. Comput. **19**, 2032–2050 (1998)
17. Klar, A.: An asymptotic-induced scheme for nonstationary transport equations in the diffusive limit. SIAM J. Numer. Anal. **35**, 1073–1094 (1998)
18. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. **9**, 987–1000 (1998)
19. Lee, J. Y., Jang, J. W., Hwang, H. J.: The model reduction of the Vlasov-Poisson-Fokker-Planck system to the Poisson-Nernst-Planck system via the deep neural network approach, arXiv preprint arXiv:2009.13280 (2020)
20. Lemou, M., Méhats, F.: Micro-macro schemes for kinetic equations including boundary layers. SIAM J. Sci. Comput. **34**, B734–B760 (2012)
21. Lemou, M., Mieussens, L.: New asymptotic preserving scheme based on micro-macro formulation for linear kinetic equations in the diffusion limit. SIAM J. Sci. Comput. **31**, 334–368 (2008)
22. Lewis, E., Miller, W., Jr.: Computational Methods of Neutron Transport. Wiley, London (1983)
23. Li, Q., Lu, J., Sun, W.: Diffusion approximations and domain decomposition method of linear transport equations: asymptotics and numerics. J. Comput. Phys. **292**, 141–167 (2015)
24. Li, Q., Wang, L.: Implicit asymptotic preserving method for linear transport equations. Commun. Comput. Phys. **22**, 157–181 (2017)
25. Li, W., Song, P., Wang, Y.: An asymptotic-preserving imex method for nonlinear radiative transfer equation, arXiv preprint arXiv:2008.06730 (2020)
26. Liu, L., Zeng, T., Zhang, Z.: A deep neural network approach on solving the linear transport model under diffusive scaling, arXiv preprint arXiv:2102.12408 (2021)
27. Lu, J., Lu, Y.: A priori generalization error analysis of two-layer neural networks for solving high dimensional schrödinger eigenvalue problems, arXiv preprint arXiv:2105.01228 (2021)
28. Lu, Y., Lu, J., Wang, M.: A priori generalization analysis of the deep ritz method for solving high dimensional elliptic partial differential equations. In: Conference on Learning Theory, PMLR, pp. 3196–3241 (2021)
29. Manteuffel, T.A., Ressel, K.J., Starke, G.: A boundary functional for the least-squares finite-element solution of neutron transport problems. SIAM J. Numer. Anal. **37**, 556–586 (1999)
30. Mishra, S., Molinaro, R.: Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs (2020). arXiv preprint arXiv:2006.16144
31. Peng, Z., Cheng, Y., Qiu, J.-M., Li, F.: Stability-enhanced ap imex-ldg schemes for linear kinetic transport equations under a diffusive scaling. J. Comput. Phys. **415**, 109485 (2020)
32. Raissi, M., Perdikaris, P., Karniadakis, G.E.: 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)
33. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, Cambridge (2014)
34. Sirignano, J., Spiliopoulos, K.: Dgm: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018)
35. Sun, W., Jiang, S., Xu, K.: An implicit unified gas kinetic scheme for radiative transfer with equilibrium and non-equilibrium diffusive limits. Commun. Comput. Phys. **22**, 889–912 (2017)
36. Tang, M., Wang, L., Zhang, X.: Accurate front capturing asymptotic preserving scheme for nonlinear gray radiative transfer equation. SIAM J. Sci. Comput. **43**, B759–B783 (2021)

37. Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient pathologies in physics-informed neural networks, arXiv preprint arXiv:2001.04536 (2020)
38. Wang, S., Yu, X., Perdikaris, P.: When and why pinns fail to train: a neural tangent kernel perspective, arXiv preprint arXiv:2007.14527 (2020)
39. Weinan, E., Yu, B.: The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. **6**, 1–12 (2018)
40. Yang, X., Golse, F., Huang, Z., Jin, S.: Numerical study of a domain decomposition method for a two-scale linear transport equation. Netw. Heterogeneous Media **1**, 143 (2006)

Publisher's Note