Asymptotic-Preserving Neural Networks for Multiscale Time-Dependent Linear Transport Equations

Shi Jin¹,²,⁴ · Zheng Ma¹,²,³,⁴ · Keke Wu¹,⁴

Received: 9 December 2021 / Revised: 23 November 2022 / Accepted: 5 January 2023 / Published online: 31 January 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
In this paper we develop a neural network for the numerical simulation of time-dependent linear transport equations with diffusive scaling and uncertainties. The goal of the network is to resolve the computational challenges of curse-of-dimensionality and multiple scales of the problem. We first show that a standard Physics-Informed Neural Network (PINN) fails to capture the multiscale nature of the problem, hence justifies the need to use Asymptotic-Preserving Neural Networks (APNNs). We show that not all classical AP formulations are directly fit for the neural network approach. We construct a micro-macro decomposition based neural network, and also build in a mass conservation mechanism into the loss function, in order to capture the dynamic and multiscale nature of the solutions. Numerical examples are used to demonstrate the effectiveness of this APNNs.

1 Introduction

In the category of multiscale modeling, kinetic equations are bridges between continuum and atomistic models [1]. There are two major challenges in kinetic modeling. First is the curse-of-dimensionality, since kinetic equations describe the evolution of probability density function of large number of particles, thus are defined in the phase space, typically a six-dimensional problem plus time. When uncertainties are considered the dimension can be much higher [2–6]. Another difficulty is the multiscale nature. Kinetic equations usually contain small or multiple space and/or temporal scales, characterized by the Knudsen number, which is the dimensionless mean free path or time. Often multiscale computations involve the coupling of models at different scales by different numerical schemes [7]. When the Knudsen number is small, kinetic equations can often be approximated by macroscopic hydrodynamic or diffusion equations [8]. In this regime, numerical simulations become...
prohibitively expensive, since one needs to numerically resolve the small physical scales. Asymptotic-preserving (AP) schemes are those that mimic the asymptotic transitions from the kinetic or hyperbolic equations to their macroscopic (hydrodynamic or diffusive) limits in discrete setting [9]. Since the mid-1990’s, the development of AP schemes for such problems has generated many interests [9–11]. The AP strategy has been proved to be a powerful and robust technique to address multiscale problems in many kinetic and hyperbolic problems. The main advantage of AP schemes is that they are very efficient in the hydrodynamic or diffusive regime, since they do not need to resolve the small physical parameters numerically and yet can still capture the macroscopic behavior governed by the hydrodynamic or diffusion equations. In fact, carefully constructed AP schemes avoid the difficulty of coupling a microscopic solver with a macroscopic one, as the micro solver automatically becomes a macro solver in the zero Knudsen number limit. It was proved, in the case of linear transport with a diffusive scaling, the AP scheme converges uniformly with respect to the scaling parameter [12].

Due to aforementioned computational challenges of kinetic modeling and computations, efficient computational methods that can efficiently deal with both curse-of-dimensionality and multiple scales are highly desirable. Classical tools to address such issues often use Monte-Carlo methods, which have low-order accuracy, and become exceedingly expensive when the Knudsen number becomes small [13–15]. In this paper we seek a machine learning based method to tackle these challenges.

The idea of using machine learning method, especially with deep neural networks (DNNs) to solve high-dimensional PDEs has been developed rapidly and achieved some success recently in many problems [16–19]. In all these approaches, the DNNs came down to minimize the loss function, a high dimensional non-convex optimization problem. The choices of losses usually make a difference for the PDEs; see [16, 17, 20–25] for reviews and references therein, and for applications in kinetic equations particularly, see [26–31]. Underlying this success is a new way to approximate functions, from an additive construction commonly used in approximation theory to a compositional construction used in DNNs. The compositional construction seems to be particularly powerful in high dimensions. This suggests that deep neural network based models can be of use in other contexts that involve constructing solution functions of PDEs [16]. The machine learning/DNNs approach might have several advantages. Firstly, it could break the curse of dimensionality [32, 33]. Secondly, compared to traditional mesh-based methods, such as the finite difference method (FDM) and the finite element method (FEM), deep learning could be a mesh-free approach by taking advantage of automatic differentiation. Thirdly, automatic differentiation could be used directly to avoid truncation errors and the numerical quadrature errors of variational forms [17]. Finally, one attractive feature of PINNs is that it can be used to solve inverse problems with minimum change of the code for forward problems [34–36].

However, the DNN approach also brings some drawbacks compared with classical numerical methods. The two well-known drawbacks are: lack of high accuracy and long training time due to large number of parameters in DNNs and the use of stochastic gradient method during the training step for high-dimensional non-convex functions [37]. Most importantly, as we find in this article, that the standard DeepRitz, PINNs, or NeuralOp approaches have difficulties in dealing with multiscales, especially for the unsteady problems. This is the main issue we want to address in this paper, and our main strategy to tackle the multiscale challenges is to design an Asymptotic-Preserving neural networks (APNNs).

In this paper, we present a framework of APNNs methods for linear transport equations that could exhibit diffusive behavior. The idea is based on the following observations when using PINNs-like framework to solve multiscale PDEs:
1. For the time-dependent/unsteady PDEs, PINNs converts it to a minimization problem of the population/empirical risk/loss in the least square formulation [17], which can only capture the leading order/single scale—depending on how to design the loss—behavior. This is not only due to poor selection of the risk/loss and commonly used first-order gradient based optimization algorithms, but also due to the underlying Frequency-principle [38] of DNN fitting function that is difficult to capture the small scale (high frequency) part of the solutions.

2. Unlike traditional AP schemes, the PINNs framework ignores the asymptotic property of the problem thus cannot capture the macroscopic behavior—which correspond to small physical parameters—efficiently or effectively. On the contrary, our strategy is to define the loss based on an AP strategy, which captures the correct asymptotic behavior when the physical scaling parameters become small, namely the loss has the AP property.

3. Beyond the AP property, in addition, the conservation or other physical constraints needed to be satisfied by the DNN solutions to get the correct solutions when dealing with small scaling parameters. We propose such a conservation mechanism in our loss.

We first define the APNNs:

Definition 1 Assume the solution is parameterized using DNN in certain way and trained by using a gradient-based method to minimize a Loss/Risk. Then we say it is APNN if, as the physical scaling parameter tends to zero, the loss of the microscopic equation converges to the loss of the macroscopic equation. That is, the loss when viewed as a numerical approximation of the original equation, has the AP property.

Our main contributions lie in the following aspects:

1. We propose the concept of Asymptotic-preserving neural networks (APNNs) which is a class of machine learning methods that can solve efficiently multiscale equations whose scaling parameter may differ in several orders of magnitude. By requiring loss to satisfy certain conditions (including AP and conservation), these methods can solve the equations accurately, efficiently and use number of neurons and layers independent of the scaling parameter. Moreover, it can automatically capture the limiting macroscopic behavior as the scaling parameter tends to zero.

2. A novel deep neural network with structure is constructed for the linear transport equations that exhibit multiscale or diffusive behavior. By using the micro-macro decomposition formulation in the loss and building in the mass-conservation structure, we obtain a machine learning approach to solve the equation which is APNN.

This paper is organized as follows. In Sect. 2, a brief introduction of the linear transport equation and its diffusion limit is given. The AP micro-macro decomposition is presented in Sect. 3. Section 4 is our main part where APNN is introduced with a detailed illustration. Specially, the loss function we choose is based on the micro-macro decomposition and we present our deep neural network that preserves both the asymptotic diffusion limit as well as mass conservation. Numerical problems for both multiscale and high-dimensional uncertainties are given in Sect. 5, which are solved by APNNs and compared with PINNs. The paper is concluded in Sect. 6.

2 The Transport Equation and its Diffusion Limit

Consider the multidimensional transport equation with the diffusive scaling. Let \( f(t, x, v) \) be the density distribution for particles at space point \( x \in D \subset \mathbb{R}^d \), time \( t \in [0, T] \) := \( T \subset \mathbb{R} \),
and traveling in direction $v \in \Omega \subset \mathbb{R}^d$, with $\int_{\Omega} dv = S$. Here $\Omega$ is symmetric in $v$, meaning that $\int_{\Omega} g(v) \, dv = 0$ for any function $g$ odd in $v$. Then $f$ solves the linear transport equation

$$\varepsilon \partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} \mathcal{L} f + \varepsilon Q,$$

with the collision operator

$$\mathcal{L} f = \frac{\sigma_S}{S} \int_{\Omega} f \, dv' - \sigma f,$$

where $\sigma = \sigma(x)$ is the total transport coefficient, $\sigma_S = \sigma_S(x)$ is the scattering coefficient, $Q = Q(x)$ is the source term, $\varepsilon > 0$ is the dimensionless mean free path (Knudsen number). Notice the time variable has been rescaled to the long-time scale $O(1/\varepsilon)$. Here,

$$\sigma_S = \sigma - \varepsilon^2 \sigma_A,$$

where $\sigma_A = \sigma_A(x)$ is the absorption coefficient. Such an equation arises in neutron transport, radiative transfer [39], wave propagation in random media [40], etc. In all these applications, the scaling gives rise to a diffusion equation as $\varepsilon \to 0$ [41], which is

$$\partial_t \rho = D \nabla_x \cdot \left( \frac{1}{\sigma} \nabla_x \rho \right) - \sigma_A \rho + Q,$$

where $\rho = \int_{\Omega} f(v) \, dv$. For different collision operators, the diffusion coefficient $D$ may be different. For examples, $D = 1/3$ in one-dimensional slab geometry and $D = 1/2$ when $\Omega$ is a unit sphere in two dimension. This diffusion approximation can be obtained by using the Hilbert expansion with the ansatz

$$f = f^{(0)} + \varepsilon f^{(1)} + \varepsilon^2 f^{(2)} + \cdots$$

and balance the terms by the order in $\varepsilon$. The diffusion approximation can also be derived using the so-called even- and odd-parities [42], or micro-macro decomposition technique [43] and various Asymptotic-preserving schemes can be constructed based on these methods [9].

For example in the 1-d case, by splitting Eq. (1) and define even- and odd-parities as

$$r(t, x, v) = \frac{1}{2} [f(t, x, v) + f(t, x, -v)],$$

$$j(t, x, v) = \frac{1}{2\varepsilon} [f(t, x, v) - f(t, x, -v)],$$

one can obtain

$$\partial_t r + v \partial_x j = \frac{\sigma_S}{\varepsilon^2} (\rho - r) - \sigma_A r + Q,$$

$$\partial_t j + \frac{v}{\varepsilon^2} \partial_x r = -\frac{\sigma_S}{\varepsilon^2} j - \sigma_A j.$$  

As $\varepsilon \to 0$, the first equation gives $r = \rho$ and the second gives $v \partial_x r = -\sigma_S j$. Substituting back to the first equation will result

$$\partial_t \rho = -v^2 \partial_x \left( \frac{1}{\sigma_S} \partial_x \rho \right) - \sigma_A \rho + Q,$$

which after integrating over $v$ yields to the diffusion limit Eq. (4).
method with the desired AP property. See Remark 3. This is an interesting phenomenon we want to emphasis in this paper. As will be seen in the sequel, the micro-macro decomposition based formulation will help us to construct the desired APNN.

3 The Micro-Macro Decomposition

In this section we describe the problem setup and then introduce the micro-macro decomposition. Consider the linear transport equation with initial and boundary conditions over a bounded domain $T \times D \times \Omega$:

$$\begin{cases}
\epsilon \partial_t f + v \cdot \nabla_x f = \frac{1}{\epsilon} \mathcal{L} f + \epsilon Q, \quad (t, x, v) \in T \times D \times \Omega, \\
B f = F_B, \quad (t, x, v) \in T \times \partial D \times \Omega, \\
I f = f_0, \quad (t, x, v) \in \{t = 0\} \times D \times \Omega,
\end{cases}$$

where $F_B, f_0$ are given functions; $\partial D$ is the boundary of $D$, and $B, I$ are boundary and initial operators, respectively. More precisely we consider the 1-d and 2-d cases in this paper.

3.1 One-dimensional case

Consider the one-dimensional transport equation in slab geometry

$$\epsilon \partial_t f + v \partial_x f = \frac{1}{\epsilon} \left( \frac{\sigma s}{2} \int_{-1}^{1} f \, dv' - \sigma f \right) + \epsilon Q, \quad x_L < x < x_R, \quad -1 \leq v \leq 1,$$

with in-flow boundary conditions as,

$$f(t, x_L, v) = F_L(v) \quad \text{for} \quad v > 0,$$
$$f(t, x_R, v) = F_R(v) \quad \text{for} \quad v < 0,$$

or periodic boundary condition

$$f(t, x_L, v) = f(t, x_R, v).$$

The initial function is given as a function of $x$ and $v$

$$f(0, x, v) = f_0(x, v).$$

3.2 Two-dimensional case

The two-dimensional case is very similar except the velocity/angular variables are constrained in the unit circle

$$\epsilon \partial_t f + v \cdot \partial_x f = \frac{1}{\epsilon} \left( \frac{\sigma s}{2\pi} \int_{|v|=1} f \, dv' - \sigma f \right) + \epsilon Q, \quad x \in \Gamma \subset \mathbb{R}^2, \quad |v| = 1,$$

with $v = (\xi, \eta), -1 \ll \xi, \eta \ll 1, \xi^2 + \eta^2 = 1$. The in-flow boundary condition is,

$$f(t, x, v) = F_B(x, v) \quad \text{for} \quad n \cdot v < 0, \quad x \in \partial \Gamma,$$

where $n$ is the outer normal of the boundary. The initial condition is

$$f(0, x, v) = f_0(x, v).$$
In this section, we describe the micro-macro decomposition formulation for the linear transport equation. It is a useful mechanism to build AP schemes [43]. The idea begins with the decomposition of $f$ into the equilibrium part $\rho$ and the non-equilibrium part $g$:

$$f = \rho + \epsilon g,$$

(17)

where

$$\rho = \frac{1}{S} \int_{\Omega} f \, dv'.$$

(18)

The non-equilibrium part $g$ clearly satisfies $\langle g \rangle = 0$, where

$$\langle g \rangle := \frac{1}{S} \int_{\Omega} g \, dv' = 0.$$

(19)

Applying Eq. (17) in (1) gives

$$\epsilon \partial_t \rho + \epsilon^2 \partial_t g + v \cdot \nabla_x \rho + \epsilon v \cdot \nabla_x g = Lg + \epsilon Q.$$

(20)

Integrating this equation with respect to $v$ one obtains the following continuity equation:

$$\partial_t \rho + \nabla_x \cdot \langle vg \rangle = Q.$$  

(21)

Define operator $\Pi : \Pi(\cdot)(v) = \langle \cdot \rangle$ and $I$ is the identity operator. Then an evolution equation on $g$ is found by applying the orthogonal projection $I - \Pi$ to Eq. (20):

$$\epsilon^2 \partial_t g + \epsilon (I - \Pi)(v \cdot \nabla_x g) + v \cdot \nabla_x \rho = Lg + (I - \Pi)\epsilon Q.$$  

(22)

Finally, (21) and (22) together with the constraint (18) and (19) constitute the micro-macro formulation of (1):

$$\begin{align*}
\partial_t \rho + \nabla_x \cdot \langle vg \rangle &= Q, \\
\epsilon^2 \partial_t g + \epsilon (I - \Pi)(v \cdot \nabla_x g) + v \cdot \nabla_x \rho &= Lg + (I - \Pi)\epsilon Q, \\
\langle g \rangle &= 0,
\end{align*}$$  

(23)

or the last equation can be replaced by enforcing it only at initial time

$$\begin{align*}
\partial_t \rho + \nabla_x \cdot \langle vg \rangle &= Q, \\
\epsilon^2 \partial_t g + \epsilon (I - \Pi)(v \cdot \nabla_x g) + v \cdot \nabla_x \rho &= Lg + (I - \Pi)\epsilon Q, \\
\langle g \rangle(0, x) &= 0.
\end{align*}$$  

(24)

This is due to the conservation of $\langle g \rangle$ with respect to $t$ and can be seen easily by integrating the second equation with respect to $v$,

$$\partial_t \langle g \rangle = 0.$$  

(25)

When $\epsilon \to 0$, the above system formally approaches

$$\begin{align*}
\partial_t \rho + \nabla_x \cdot \langle vg \rangle &= Q, \\
v \cdot \nabla_x \rho &= Lg.
\end{align*}$$  

(26)

The second equation yields

$$g = \mathcal{L}^{-1}(v \cdot \nabla_x \rho),$$  

(27)

which, when plugging into the first equation and integrating over $v$, gives the diffusion equation (4).
Fig. 1 Illustration of APNNs. \( \mathcal{F}^{-1} \) is the microscopic equation that depends on the small scale parameter \( \varepsilon \) and \( \mathcal{F}^{0} \) is its macroscopic limit as \( \varepsilon \to 0 \), which is independent of \( \varepsilon \). The latent solution of \( \mathcal{F}^{-1} \) is approximated by neural networks with its measure denoted by \( \mathcal{R}(\mathcal{F}^{-1}) \). The asymptotic limit of \( \mathcal{R}(\mathcal{F}^{0}) \) as \( \varepsilon \to 0 \), if exists, is denoted by \( \mathcal{R}(\mathcal{F}^{0}) \). If loss \((\mathcal{F}^{0})\) is a good measure of \( \mathcal{F}^{0} \), then it is called asymptotic-preserving (AP).

In our APNN framework, we are going to solve the micro-macro system (23) for Eq. (9) with boundary and initial conditions. Both \( \rho \) and \( g \) are parameterized by neural networks.

Remark 1 In (23) or (24) we singled out the condition \( \langle g \rangle = 0 \). This is not necessary in constructing classical AP schemes [43]. However, for DNN this condition is usually not automatically satisfied and one needs to impose this condition in the loss, as shown in the next section.

4 Asymptotic-Preserving Neural Networks

Unlike classical numerical schemes, DNN framework consists of three ingredients: a neural network parametrization of the solution, a population and empirical loss/risk and an optimization algorithm. In terms of proposed APNNs, the key component is to design a loss that has the AP property. The diagram in Fig. 1 illustrates the idea of APNNs.

The procedure of a deep neural network for solving a PDE problem consists of three parts: a neural network structure, a loss, and a method to minimize loss over the parameter space.

In what follows, conventional notations for deep neural networks (DNNs)\(^1\) are introduced. An \( L \)-layer feed forward neural network is defined recursively as,

\[
\begin{align*}
    f^{[0]}_\theta (x) &= x, \\
    f^{[l]}_\theta (x) &= \sigma \circ (W^{[l-1]} f^{[l-1]}_\theta (x) + b^{[l-1]}), \quad 1 \leq l \leq L - 1, \\
    f_\theta (x) &= f^{[L]}_\theta (x) = W^{[L-1]} f^{[L-1]}_\theta (x) + b^{[L-1]},
\end{align*}
\]

where \( W^{[l]} \in \mathbb{R}^{m_{l+1} \times m_l}, b^{[l]} \in \mathbb{R}^{m_{l+1}}, m_0 = d, m_L = d \) is the input dimension, \( m_L = d \) is the output dimension, \( \sigma \) is a scalar function and " \( \circ \) " means entry-wise operation. We denote the set of parameters by \( \theta \). For simplicity of neural network presentation, we denote the layers by a list, i.e., \([m_0, \ldots, m_L]\).

There are various choices to build the loss, when applying a DNN to solve a given PDE. In this paper we compare several results about different scale of \( \varepsilon \) in Eq. (9) with Physics-informed Neural Networks (PINNs) and our proposed Asymptotic Preserving Neural Networks (APNNs). PINNs has the loss as the mean-square error or the residual associated

\(^1\) BAAI.2020. Suggested Notation for Machine Learning. https://github.com/mazhengcn/suggested-notation-for-machine-learning.
to the given PDE. In contrast, APNNs rewrites the original PDE into a system in asymptotic-preserving form and takes their mean-square residual error as the loss. Boundary and initial conditions are treated as a regularization or penalty term with penalty parameters $\lambda_1, \lambda_2$ into the loss, which are selected for better performance.

For simplicity in the following discussion we set $\sigma = \sigma_S = 1$.

### 4.1 The Failure of PINNs to Resolve Small Scales

For PINNs, one neural network is applied to directly approximate the density function $f(t, x, v)$,

$$\text{NN}_\theta(t, x, v) \approx f(t, x, v).$$

(29)

The inputs of DNN are $(t, x, v)$, i.e., $m_0 = 3, 5$ for 1-d and 2-d respectively. The output is a scalar which represents the value of $f$ at $(t, x, v)$. Since $f$ is always non-negative, we put an exponential function at the last output layer of the DNN. To be precise, we use

$$f_{\bar{\theta}}^\text{NN}(t, x, v) := \exp(-\tilde{f}_{\bar{\theta}}^\text{NN}(t, x, v)) \approx f(t, x, v)$$

(30)

to represent the numerical solution of $f$. Then the least square of the residual of the original transport Eq. (1) is used as the target loss function, together with boundary and initial conditions as penalty terms, that is,

$$\mathcal{R}^\varepsilon_{\text{PINN}} = \frac{1}{|T \times D \times \Omega|} \int_T \int_D \int_{\Omega} \left| \varepsilon \partial_t f_{\bar{\theta}}^\text{NN} + \varepsilon v \cdot \nabla_x f_{\bar{\theta}}^\text{NN} - \mathcal{L} f_{\bar{\theta}}^\text{NN} - \varepsilon^2 Q \right|^2 d\nu d\sigma d\tau$$

$$+ \frac{\lambda_1}{|T \times \partial D \times \Omega|} \int_T \int_{\partial D} \int_{\Omega} |B f_{\bar{\theta}}^\text{NN} - F_B|^2 d\nu d\sigma d\tau$$

$$+ \frac{\lambda_2}{|D \times \Omega|} \int_D \int_{\Omega} |I f_{\bar{\theta}}^\text{NN} - f_0|^2 d\nu d\sigma,$$

(31)

where $\lambda_1$ and $\lambda_2$ are the penalty weights to be tuned. Then a standard stochastic gradient method (SGD) or Adam optimizer is used to find the global minimum of this loss. Notice that in order to approximate each integral in the loss we use randomly sampled batch at every step, to be detailed in section 5.

Now let us check whether this PINN method is AP. We only need to focus on the first term of (31)

$$\mathcal{R}^\varepsilon_{\text{PINN, residual}} := \frac{1}{|T \times D \times \Omega|} \int_T \int_D \int_{\Omega} \left| \varepsilon \partial_t f_{\bar{\theta}}^\text{NN} + \varepsilon v \cdot \nabla_x f_{\bar{\theta}}^\text{NN} - \mathcal{L} f_{\bar{\theta}}^\text{NN} - \varepsilon^2 Q \right|^2 d\nu d\sigma d\tau.$$ 

(32)

Taking $\varepsilon \to 0$, formally this will lead to

$$\mathcal{R}^\varepsilon_{\text{PINN, residual}} := \frac{1}{|T \times D \times \Omega|} \int_T \int_D \int_{\Omega} \left| -\mathcal{L} f_{\bar{\theta}}^\text{NN} \right|^2 d\nu d\sigma d\tau,$$

(33)

which can be viewed as the PINN loss of the equilibrium equation

$$\mathcal{L} f = 0.$$ 

(34)

This shows that when $\varepsilon$ is very small, to the leading order we are solving equation $\mathcal{L} f = 0$ which gives $f = \rho$. This does not give the desired diffusion Eq. (4). This explains why PINN will fail when $\varepsilon$ is small. We also conduct numerical experiments to verify this claim in Sect. 5.
4.2 The Micro-Macro Decomposition Based APNN Method

Now we present an APNN method based on the micro-macro decomposition method. The main idea is to use PINN to solve the micro-macro system (23) or (24) instead of the original Eq. (1).

First we need to use DNN to parametrize two functions $\rho(x, v)$ and $g(t, x, v)$. So here two networks are used. First

$$\rho^\text{NN}_\theta(t, x) := \exp(-\tilde{\rho}^\text{NN}_\theta(t, x)) \approx \rho(t, x),$$

(35)

Notice here $\rho$ is non-negative. Second,

$$g^\text{NN}_\theta(t, x, v) := \tilde{g}^\text{NN}_\theta(t, x, v) - \langle \bar{g}^\text{NN}_\theta \rangle(t, x) \approx g(t, x, v).$$

(36)

Here $\tilde{\rho}$ and $\tilde{g}$ are both fully-connected neural networks. Notice that by choosing $g^\text{NN}_\theta(t, x, v)$ as in (36) it will automatically satisfy the constraint (19) as,

$$\langle g^\text{NN}_\theta \rangle = \langle \bar{g}^\text{NN}_\theta \rangle - \langle \bar{g}^\text{NN}_\theta \rangle = 0, \ \forall \ t, x,$$

(37)

which automatically satisfies the third equation in the micro-macro system (23).

Then we propose the least square of the residual of the micro-macro system (23) as the APNN loss,

$$\mathcal{R}^\varepsilon_{\text{APNN}} = \frac{1}{|T \times D|} \int_T \int_D \left| \partial_t \rho^\text{NN}_\theta + \nabla_x \cdot \{v g^\text{NN}_\theta \} - Q \right|^2 \, dx \, dt$$

$$+ \frac{1}{|T \times D \times \Omega|} \int_T \int_D \int_\Omega \left| e^2 \partial_t \bar{g}^\text{NN}_\theta + \varepsilon (I - \Pi) (v \cdot \nabla_x g^\text{NN}_\theta) + v \cdot \nabla_x \rho^\text{NN}_\theta - \mathcal{L} g^\text{NN}_\theta - (I - \Pi) \varepsilon Q \right|^2 \, dv \, dx \, dt$$

$$+ \frac{\lambda_1}{|T \times \partial D \times \Omega|} \int_T \int_{\partial D} \int_{\partial \Omega} \left| \mathcal{B}(\rho^\text{NN}_\theta + \varepsilon g^\text{NN}_\theta) - F_B \right|^2 \, dv \, dx \, dt$$

$$+ \frac{\lambda_2}{|D \times \Omega|} \int_D \int_{\partial \Omega} \left| \mathcal{L} \rho^\text{NN}_\theta + \varepsilon g^\text{NN}_\theta - f_0 \right|^2 \, dv \, dx \, dt.$$

(38)

Now we show formally the AP property of this loss by considering its behavior for $\varepsilon$ small. We only need to focus on the first two terms of (38)

$$\mathcal{R}^\varepsilon_{\text{APNN, residual}} = \frac{1}{|T \times D|} \int_T \int_D \left| \partial_t \rho^\text{NN}_\theta + \nabla_x \cdot \{v g^\text{NN}_\theta \} - Q \right|^2 \, dx \, dt$$

$$+ \frac{1}{|T \times D \times \Omega|} \int_T \int_D \int_\Omega \left| e^2 \partial_t \bar{g}^\text{NN}_\theta + \varepsilon (I - \Pi) (v \cdot \nabla_x g^\text{NN}_\theta) + v \cdot \nabla_x \rho^\text{NN}_\theta - \mathcal{L} g^\text{NN}_\theta - (I - \Pi) \varepsilon Q \right|^2 \, dv \, dx \, dt$$

(39)

Taking $\varepsilon \to 0$, formally this will lead to

$$\mathcal{R}_{\text{APNN, residual}} = \frac{1}{|T \times D|} \int_T \int_D \left| \partial_t \rho^\text{NN}_\theta + \nabla_x \cdot \{v g^\text{NN}_\theta \} - Q \right|^2 \, dx \, dt$$

$$+ \frac{1}{|T \times D \times \Omega|} \int_T \int_D \int_\Omega \left| v \cdot \nabla_x \rho^\text{NN}_\theta - \mathcal{L} g^\text{NN}_\theta \right|^2 \, dv \, dx \, dt,$$

(40)

which is the least square loss of Eq. (26)

$$\begin{cases}
\partial_t \rho + \nabla_x \cdot \{v g\} = Q, \\
v \cdot \nabla_x \rho = \mathcal{L} g.
\end{cases}$$

(41)
Fig. 2 Schematic of APNNs for solving the linear transport equation with initial and boundary conditions

Same as in Section 3.1 the second equation yields \( g = \mathcal{L}^{-1}(v \cdot \nabla_x \rho) \), which, when plugging into the first equation and integrating over \( v \), gave the diffusion Eq. (4). Hence this proposed method is an APNN method. Finally we put a schematic plot of our method in Fig. 2.

**Remark 2** For the constraint \( \langle g \rangle = 0 \), one way is to construct a novel neural network for \( g \) such that it exactly satisfies \( \langle g \rangle = 0 \). The other way is to treat it as a soft constraint with parameter \( \lambda_3 \), i.e., without using (19), we use \( \hat{g}^{\text{NN}}_\theta \) and modifies the loss as

\[
\mathcal{R}_{\text{APNN, residual}} + \frac{\lambda_3}{|T \times D|} \int_T \int_D |\langle \hat{g}^{\text{NN}}_\theta \rangle - 0|^2 \, dx \, dt.
\] (42)

Our APNNs belongs to the first scenario. Specifically, in \( \mathcal{R}_{\text{APNN}} \), the integral is not exactly the micro-macro decomposition in (23). Instead we replaced \( g \) by \( g - \langle g \rangle \) since the later always satisfies \( \langle \cdot \rangle = 0 \), the desired conservation property! There are other terms in \( \mathcal{R}_{\text{APNN}} \) where such a change will not have any impact since the corresponding operators are invariant under such a transformation.

**Remark 3** The parity formulation (7) also allows one to construct AP schemes [42]. However if one uses it in the loss, it will not be an APNN. To see this, consider the case when \( \epsilon \to 0 \). Since the DNN will only pick up the leading term, thus one has, in the \( L^2 \) sense,

\[
0 = \rho - r,
\]

\[
j = -\frac{v}{\sigma_s} \partial_x r.
\] (43)

These two equations will not lead to the diffusion equation (4).
5 Numerical Examples

In this section, in order to verify and compare the performance of PINNs and APNNs, we present both 1D and 2D numerical results for several problems chosen from rarefied regimes ($\varepsilon \approx O(1)$) to hydrodynamic (diffusive) regimes ($\varepsilon \rightarrow 0$), including examples with high-dimensional uncertainties.

Since the losses of PINNs and APNNs are integrals, we approximate them by the Monte Carlo method by selecting small number of sub-domains randomly (batch size) and compute the operator $\langle \cdot \rangle$ and $\Pi(\cdot)$ with quadrature rule. The optimization problem is solved by the Adam version of the gradient descent method [44]. The initial value of parameters set $\theta$ in all numerical experiments are generated by Xavier initialization. The learning rate is set to be $0.001$ during training. All the hyper-parameters are chosen for best performance after trying these experiments. Specifically, in most of our experiments, we fix the $\lambda$ domain as $[0, 1]$ and approximate the problem at randomly selected points $\{(t_i, x_i, v_i)\}$. It should be pointed out that the integral of $v$ for operator $\mathcal{L}/\langle \cdot \rangle$ is computed by quadrature rule (the Gauss-Legendre Integration). In detail, assume $\{w_i, v_i\}_{i=1}^n$ are the nodes and weights, which $\{v_i\}_{i=1}^n$ are the roots of the Legendre polynomials with degree $n$ and $\{w_i\}_{i=1}^n$ are determined for accuracy. Then we can approximate an integral, for example, $\int_0^1 f(v) \, dv$ by the summation of linear combination of $f(v_i) : \sum_{i=1}^n w_i f(v_i)$.

The empirical risk for PINN is as follows

$$\mathcal{R}_{\text{PINN}} = \frac{1}{N_1} \sum_{i=1}^{N_1} \left| \varepsilon^2 \partial_t f_0^{\text{NN}}(t_i, x_i, v_i) + \varepsilon \mathbf{v} \cdot \nabla x f_0^{\text{NN}}(t_i, x_i, v_i) - \mathcal{L} f_0^{\text{NN}}(t_i, x_i) - \varepsilon^2 Q \right|^2$$

$$+ \frac{\lambda_1}{N_2} \sum_{i=1}^{N_2} |B f_0^{\text{NN}}(t_i, x_i, v_i) - F_B(t_i, x_i, v_i)|^2$$

$$+ \frac{\lambda_2}{N_3} \sum_{i=1}^{N_3} |I f_0^{\text{NN}}(t_i, x_i, v_i) - f_0(t_i, x_i, v_i)|^2,$$

where $N_1, N_2, N_3$ are the number of sample points of $T \times \mathcal{D} \times \Omega, T \times \partial \mathcal{D} \times \Omega, \mathcal{D} \times \Omega$.

Similarly, the empirical risk for APNN is as follows

$$\mathcal{R}_{\text{APNN}} = \frac{1}{N_1^{(1)}} \sum_{i=1}^{N_1^{(1)}} \left| \partial_t \rho_0^{\text{NN}}(t_i, x_i) + \nabla x \cdot \langle \mathbf{v} g_0^{\text{NN}} \rangle (t_i, x_i) - Q \right|^2$$

$$+ \frac{1}{N_1^{(2)}} \sum_{i=1}^{N_1^{(2)}} \left| \varepsilon^2 \partial_t g_0^{\text{NN}}(t_i, x_i, v_i) + \varepsilon (I - \Pi)(\mathbf{v} \cdot \nabla x g_0^{\text{NN}})(t_i, x_i) + \mathbf{v}_i \cdot \nabla x g_0^{\text{NN}}(t_i, x_i) - \mathcal{L} g_0^{\text{NN}}(t_i, x_i) - (I - \Pi)\varepsilon Q \right|^2$$

$$+ \frac{\lambda_1}{N_2} \sum_{i=1}^{N_2} |B(\rho_0^{\text{NN}}(t_i, x_i)) + \varepsilon g_0^{\text{NN}}(t_i, x_i, v_i)) - F_B(t_i, x_i, v_i)|^2$$

$$+ \frac{\lambda_2}{N_3} \sum_{i=1}^{N_3} |I(\rho_0^{\text{NN}}(t_i, x_i) + \varepsilon g_0^{\text{NN}}(t_i, x_i, v_i)) - f_0(t_i, x_i, v_i)|^2.$$

where $N_1^{(1)}, N_1^{(2)}, N_2, N_3$ are the number of sample points of $T \times \mathcal{D}, T \times \mathcal{D} \times \Omega, T \times \partial \mathcal{D} \times \Omega, \mathcal{D} \times \Omega.$
To investigate the influence of the Monte Carlo method in the integral with respect to \( v \), we conduct an extra experiment for the case \( \varepsilon = 10^{-8} \) of Problem II. Besides, we show why the conservation is important for the same problem.

The reference solutions are obtained by standard finite difference method. For most of the time we will check the relative \( \ell^2 \) error of the density \( \rho(x) \) between DNN methods and reference solutions, e.g. for 1d case,

\[
\text{error} := \sqrt{\frac{\sum_j |\rho_{NN, j} - \rho_{ref, j}|^2}{\sum_j |\rho_{ref, j}|^2}}. \tag{46}
\]

### 5.1 One-Dimensional Problems

We shall consider different problems in slab geometry from the rarefied regimes (\( \varepsilon \approx O(1) \)) to the diffusive regimes (\( \varepsilon \to 0 \)). Various boundary conditions and initial conditions will be used and both the transient and the steady state solutions will be presented with different \( \varepsilon \)'s.

#### Problem I. Smooth initial data with periodic BC

We start from the rarefied regimes where \( \varepsilon = 1 \) and consider periodic boundary condition with a smooth initial data as follows

\[
f_0(x, v) = \frac{\rho(x)}{\sqrt{2\pi}} e^{-\frac{v^2}{2}}, \tag{47}
\]

where

\[
\rho(x) = 1 + \cos(4\pi x). \tag{48}
\]

The source term, scattering and absorbing coefficients are set as

\[
\sigma_S = 1, \quad \sigma_A = 0, \quad Q = 0, \quad \varepsilon = 1. \tag{49}
\]

Enforcing exact periodic boundary is applied to improve the numerical performance. The ansatz is based on a Fourier basis and one can construct a transform \( T : x \rightarrow \{\sin(2\pi jx), \cos(2\pi jx)\}_{j=1}^{k} \) before the first layer of DNN [45, 46]. Here, we set \( k = 2 \).

The results are shown in Fig. 3 where we can find both PINN and APNN perform well.

#### Problem II. In-flow boundary condition

In the second example we consider the isotropic in-flow boundary conditions:

\[
x \in [0, 1], \quad F_L(v) = 1, \quad F_R(v) = 0, \tag{50}
\]

and initial condition \( f_0(x, v) = 0 \). The source term, scattering and absorbing coefficients are set as

\[
Q = 0, \quad \sigma_S = 1, \quad \sigma_A = 0, \quad \varepsilon = 1, 10^{-1}, 10^{-3}, 10^{-8}. \tag{51}
\]

The results are shown in Fig. 4 with exactly \( \langle g \rangle = 0 \). Table 1 records the relative \( \ell^2 \) errors of PINNs and APNNs in terms of penalty parameters \((\lambda_1, \lambda_2)\) with \( \varepsilon = 10^{-3} \). One can find that the approximate accuracy is not good if \( \lambda_1, \lambda_2 \) are too small or too large.

In the left of Fig. 5, this conservation condition is not exactly satisfied by treating it as a soft constraint, i.e., using (42). Clearly failure to conserve the mass gives poor result. In Fig. 4d, for small \( \varepsilon \), PINN does not give accurate results while APNN gives quite accurate...
Fig. 3 Problem I. Plot of density $\rho$ at $t = 0.0, 1.0$. Left: PINNs vs. Ref, Right: APNNs vs. Ref. Neural networks are [5, 128, 128, 128, 128, 1] for $\rho$ and [6, 256, 256, 256, 256, 1] for $g, f$. Batch size is 2000 in domain, 1000 with penalty $\lambda_2 = 1000$ for initial condition, the number of quadrature points is 30. Relative $\ell^2$ error of PINNs and APNNs are $2.41 \times 10^{-2}, 1.32 \times 10^{-2}$.

(a) Density $\rho$ at $t = 1.0$ with $\varepsilon = 1$.

(b) Density $\rho$ at $t = 0.5$ with $\varepsilon = 10^{-1}$.

(c) Density $\rho$ at $t = 0.1$ with $\varepsilon = 10^{-3}$.

(d) Density $\rho$ at $t = 0.05, 0.1$ with $\varepsilon = 10^{-8}$.

Fig. 4 Problem II. Plot of density $\rho$ for PINNs, APNNs and reference solutions. The number of quadrature points is 30. Neural networks are (a) to (d): [2, 128, 128, 128, 128, 1] for $\rho$ and [3, 256, 256, 256, 256, 1] for $g, f$. Batch size is 1000 for (a) to (d) in domain, (a) and (b): 400 $\times$ 2 with $\lambda_1 = 1$; (c) and (d): 400 $\times$ 2 with $\lambda_1 = 10$ for boundary condition, (a) and (b): 500 with $\lambda_2 = 1$; (c) and (d): 1000 with $\lambda_2 = 10$ for initial condition. Relative $\ell^2$ errors of PINNs and APNNs are (a): $4.01 \times 10^{-2}, 1.36 \times 10^{-2}$; (b): $1.17 \times 10^{-1}, 3.30 \times 10^{-2}$; (c): $2.17 \times 10^{-1}, 1.98 \times 10^{-2}$; (d): $9.40 \times 10^{-1}, 2.76 \times 10^{-2}$, respectively.
Table 1 Problem II. Relative $\ell^2$ errors of PINNs and APNNs in terms of penalty parameters $(\lambda_1, \lambda_2)$ with $\varepsilon = 10^{-3}$

| $(\lambda_1, \lambda_2)$ | Error PINNs | Error APNNs |
|---------------------------|-------------|-------------|
| (1, 10)                   | $7.09 \times 10^{-1}$ | $5.67 \times 10^{-2}$ |
| (100, 10)                 | $3.57 \times 10^{-1}$ | $4.72 \times 10^{-2}$ |
| (10, 10)                  | $2.17 \times 10^{-1}$ | $1.98 \times 10^{-2}$ |
| (10, 1)                   | $3.18 \times 10^{-1}$ | $1.11 \times 10^{-1}$ |
| (10, 100)                 | $5.52 \times 10^{-1}$ | $5.59 \times 10^{-2}$ |

Fig. 5 Problem II. Plot of density $\rho$ at $t = 0.1$. Both neural networks are $[2, 128, 256, 256, 128, 1]$ for $\rho$ and $[3, 128, 256, 512, 256, 128, 1]$ for $g$. Left: APNNs with soft constraint $\langle g \rangle = 0$ vs. reference solution. Batch size is 1000 in domain, $400 \times 2$ with penalty $\lambda_1 = 10$ for boundary condition, 1000 with penalty $\lambda_2 = 10$ for initial condition and penalty $\lambda_3 = 1$ for constrain $\langle g \rangle = 0$. Relative $\ell^2$ error of APNNs with constrain $\langle g \rangle = 0$ is $2.72 \times 10^{-1}$. Right: APNNs with integral by Monte Carlo method vs. reference solution. Batch size is 800 in domain, $500 \times 2$ with penalty $\lambda_1 = 1$ for boundary condition and 500 with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is 100. The number of sample is 100 for each iteration. The relative $\ell^2$ error of APNNs with constraint $\langle g \rangle = 0$ is $6.73 \times 10^{-2}$.

Problem III. A variable scattering coefficient

Let
\[ x \in [0, 1], \quad F_L(v) = 1, \quad F_R(v) = 0, \tag{52} \]
and initial condition $f_0(x, v) = 0$. The source term, scattering and absorbing coefficients are set as
\[ Q = 1, \quad \sigma_S = 1 + (10x)^2, \quad \sigma_A = 0, \quad \varepsilon = 10^{-2}. \tag{53} \]

In Fig. 6 we report the numerical solutions by APNN at time $t = 0.0, 0.1, 0.2$. In this problem we have a source term and the scattering cross section that depend on $x$, so the scaling term $\sigma_S/\varepsilon$ ranges from $1/\varepsilon \rightarrow O(1)$, a problem with mixing scales. The numerical results show reasonably good performance of APNN.

Problem IV. A problem with boundary layer

Let
\[ x \in [0, 1], \quad F_L(v) = 5 \sin(v), \quad F_R(v) = 0, \tag{54} \]
Fig. 6 Problem III. Plot of density $\rho$ with $\varepsilon = 10^{-2}$ at $t = 0.0, 0.1, 0.2$ for APNNs and reference solutions. Neural networks are $[2, 128, 128, 128, 128]$ for $\rho$ and $[3, 256, 256, 256, 256]$ for $g$. Batch size is 500 in domain, $200 \times 2$ with penalty $\lambda_1 = 1$ for boundary condition and 200 with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is 30.

Fig. 7 Problem IV. Plot of density $\rho$ with $\varepsilon = 5 \times 10^{-2}$ at $t = 0.05, 0.1$ for APNNs and reference solutions. Neural networks are $[2, 128, 128, 128, 128]$ for $\rho$ and $[3, 256, 256, 256, 256]$ for $g$. Batch size is 1000 in domain, $400 \times 2$ with penalty $\lambda_1 = 10$ for boundary condition and 1000 with penalty $\lambda_2 = 10$ for initial condition, the number of quadrature points is 30. Relative $\ell^2$ error of APNNs is $4.80 \times 10^{-3}$.

and initial condition $f_0(x, v) = 0$. The source term, scattering and absorbing coefficients are set as

$$Q = 0, \quad \sigma_S = 1, \quad \sigma_A = 0, \quad \varepsilon = 5 \times 10^{-2}. \quad (55)$$

Here since $F_L$ depends on $v$, there is a boundary layer near $x = 0$. Figure 7 shows that the boundary layer is well captured by APNN.
5.2 Two-Dimensional Problems

Problem V. Rarefied regime

Consider a two dimensional problem with

\[ \Gamma = [0, 1] \times [0, 1], \quad F_B(x, v) = 0, \quad n \cdot v < 0, \quad x \in \partial \Gamma, \] (56)

and initial condition \( f_0(x, v) = 0 \). The source term, scattering and absorbing coefficients are set as

\[ Q = 1, \quad \sigma_S = 1, \quad \sigma_A = 0, \quad \varepsilon = 1. \] (57)

Here \( n \) denotes the exterior unit normal vector on \( \partial \Gamma \).

Figure 8 shows the density \( \rho \) trained by APNNs and reference solution at time \( t = 1.0 \) and the residual between them. The results show the good performance of APNNs with absolute error about 0.06 and relative \( \ell^2 \) error \( 5.78 \times 10^{-2} \).

Problem VI. A diffusive regime

This test is a two dimensional test in the diffusive regime in which most of the setup are the same as previous example except \( \varepsilon = 10^{-8} \) and \( Q = 1, 50 \). Figure 9 show the density \( \rho \) trained by APNNs and reference at time \( t = 0.1 \) and corresponding residual error.

5.3 Uncertainty Quantification (UQ) Problems

For the uncertainty quantification problem we consider the linear transport equation with a sine scattering function \( (\varepsilon = 1, 10^{-3}) \) or Gaussian scattering function \( \sigma_S(z) (\varepsilon = 10^{-8}) \):

\[ \varepsilon \partial_t f + v \partial_x f = \frac{\sigma_S(z)}{\varepsilon} \left( \frac{1}{2} \int_{-1}^{1} f dv' - f \right), \quad x_L < x < x_R, \quad -1 \leq v \leq 1, \] (58)

with scattering coefficients

\[ \sigma_S(z) = 1 + 0.3 \prod_{i=1}^{10} \sin(\pi z_i), \quad z = (z_1, z_2, \ldots, z_{10}) \sim U([-1, 1]^{10}), \] (59)
Problem VI. Plot of density $\rho$ with $\varepsilon = 10^{-8}$ at $t = 0.1$ for APNNs and reference solution. Neural networks are $[3, 64, 128, 256, 512, 1]$ for $\rho$ and $[5, 64, 128, 256, 1024, 1]$ for $g$. Batch size is 800 in domain, $400 \times 2$ with penalty $\lambda_1 = 1$ for boundary condition and 400 with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is 50. Relative $\ell^2$ error of APNNs is $1.29 \times 10^{-2}$ for $Q = 1$ and $4.35 \times 10^{-2}$ for $Q = 50$

or

$$\sigma_S(z) = 1 + 0.3 \exp \left( -\frac{|z|^2}{2} \right), \quad z = (z_1, z_2, \cdots, z_{20}) \sim \mathcal{U}([-3, 3]^{20}),$$

(60)

initial condition $f_0(x, v) = 0$, and in-flow boundary condition as,

$$f(t, x_L = 0, v) = F_L(v) = 1 \quad \text{for} \quad v > 0,$$

$$f(t, x_R = 1, v) = F_R(v) = 0 \quad \text{for} \quad v < 0.$$  

(61)

In this problem the 10-dimensional or 20-dimensional vector $z$ represents the random input parameters in a typical uncertain problem setup [47]. Thus, $m_0 = 12, 13$ or 21, 22 for $\rho$ and $g$ respectively. To compare numerical results, we evaluate $\rho$ at $t = 0.05, 0.1$ by taking expectation on $10^4$ times simulations for $(z_1, \cdots, z_{10})$ or $(z_1, \cdots, z_{20})$. The goal of these examples is to show the ability of APNNs for high-dimensional problems.

Problem VII. UQ problem with $\varepsilon = 1$

Let $\varepsilon = 1$ and set scattering coefficients

$$\sigma_S(z) = 1 + 0.3 \prod_{i=1}^{10} \sin(\pi z_i).$$

(62)

Figure 10 shows the density $\rho$ trained by APNNs and reference at time $t = 0.05, 0.1$. APNNs gives good approximation results.
Fig. 10  Problem VII. Plot of density $\rho$ by taking expectation for $z$ at $t = 0.05, 0.1$ for APNNs and reference solution. $\varepsilon = 1$, $\sigma_S(z) = 1 + 0.3 \prod_{i=1}^{10} \sin(\pi z_i)$ and neural networks are $[12, 128, 256, 256, 512, 1]$ for $\rho$ and $[13, 128, 256, 512, 1024, 1]$ for $g$. Batch size is 1000 in domain, $500 \times 2$ with penalty $\lambda_1 = 1$ for boundary condition and $500$ with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is $30$. Relative $\ell^2$ error of APNNs is $8.78 \times 10^{-2}$.

Fig. 11  Problem VIII. Plot of density $\rho$ by taking expectation for $z$ at $t = 0.05, 0.1$ for APNNs and reference solutions. $\varepsilon = 10^{-3}$, $\sigma_S(z) = 1 + 0.3 \prod_{i=1}^{10} \sin(\pi z_i)$ and neural networks are $[12, 64, 128, 256, 512, 1]$ for $\rho$ and $[13, 64, 128, 256, 512, 1024, 1]$ for $g$. Batch size is 1000 in domain, $400 \times 2$ with penalty $\lambda_1 = 10$ for boundary condition and $1000$ with penalty $\lambda_2 = 10$ for initial condition, the number of quadrature points is $30$. Relative $\ell^2$ error of APNNs is $2.75 \times 10^{-2}$.

Problem VIII. UQ problem with $\varepsilon = 10^{-3}$

Let $\varepsilon = 10^{-3}$ and set scattering coefficients same as Problems VIII. Figure 11 shows the density $\rho$ trained by APNNs and reference at time $t = 0.05, 0.1$. APNN again gives quite good results.
Problem IX. Plot of density $\rho$ by taking expectation for $z$ at $t = 0.05, 0.1$ for APNNs and reference solutions. $\varepsilon = 10^{-8}$, $\sigma_S(z) = 1 + 0.3 \exp\left(-\frac{|z|^2}{2}\right)$ and neural networks are $[22, 64, 128, 256, 512, 1]$ for $\rho$ and $[23, 64, 128, 256, 512, 1024, 1]$ for $g$. Batch size is 1000 in domain, $400 \times 2$ with penalty $\lambda_1 = 10$ for boundary condition and 1000 with penalty $\lambda_2 = 10$ for initial condition, the number of quadrature points is 30. Relative $\ell^2$ error of APNNs is $1.51 \times 10^{-2}$.

Problem IX. UQ problem with Gaussian scattering function

Consider the linear transport equation with $\varepsilon = 10^{-8}$ and scattering coefficients

$$\sigma_S(z) = 1 + 0.3 \exp\left(-\frac{|z|^2}{2}\right), \quad z = (z_1, z_2, \ldots, z_{20}) \in [-3, 3]^{20}. \quad (63)$$

Figure 12 shows the density $\rho$ trained by APNNs and reference at time $t = 0.05, 0.1$. In this case one can also see good approximation results by APNN.

6 Conclusion

In this paper we propose a deep neural network (DNN) for computing the multiscale uncertain linear transport equation with diffusive scaling. Our work follows the framework of Asymptotic-Preserving (AP) schemes for multiscale kinetic equations. We first point out that not all AP schemes will have the desired asymptotic structure when implementing them in the DNN framework. We design an AP neural network (APNN) by using the micro-macro decomposition, together with a mass conservation constraint in the loss function, that will have the desired AP properties, as will be shown by various multiscale and high-dimensional uncertain examples.

Acknowledgements This work is partially supported by the National Key R&D Program of China Project No. 2020YFA0712000 and Shanghai Municipal of Science and Technology Major Project No. 2021SHZDZX0102. Shi Jin is also supported by NSFC Grant No. 11871297. Zheng Ma is also supported by NSFC Grant No. 12031013 and partially supported by Institute of Modern Analysis—A Shanghai Frontier Research Center.

Funding The work is supported by the National Key R&D Program of China Project No. 2020YFA0712000 and Shanghai Municipal of Science and Technology Major Project No. 2021SHZDZX0102. Shi Jin is also
supported by NSFC grant No. 11871297. Zheng Ma is also supported by NSFC Grant No.12031013 and partially supported by Institute of Modern Analysis – A Shanghai Frontier Research Center.

Data Availability None.

Declarations

Conflict of interest The authors have not disclosed any competing interests.

References

1. Cercignani, C.: The boltzmann equation. In: The Boltzmann Equation and its Applications, pp. 40–103. Springer (1988)
2. Jin, S., Pareschi, L.: Uncertainty Quantification for Hyperbolic and Kinetic Equations, vol. 14. Springer, Cham (2018)
3. Hu, J., Jin, S.: A stochastic Galerkin method for the Boltzmann equation with uncertainty. J. Comput. Phys. 315, 150–168 (2016)
4. Jin, S., Liu, J.-G., Ma, Z.: Uniform spectral convergence of the stochastic Galerkin method for the linear transport equations with random inputs in diffusive regime and a micro-macro decomposition-based asymptotic-preserving method. Res. Math. Sci. 4, 15–25 (2017)
5. Poëtte, G.: A GPC-intrusive Monte-Carlo scheme for the resolution of the uncertain linear Boltzmann equation. J. Comput. Phys. 385, 135–162 (2019)
6. Poëtte, G.: Numerical analysis of the Monte-Carlo noise for the resolution of the deterministic and uncertain linear Boltzmann equation (comparison of non-intrusive gpc and mc-gpc). J. Comput. Theor. Trans. 385, 135–162 (2022)
7. Weinan, E.: Principles of Multiscale Modeling. Cambridge University Press, Cambridge (2011)
8. Bouchut, F., Golse, F., Pulvirenti, M.: Kinetic Equations and Asymptotic Theory. Elsevier, London (2000)
9. Jin, S.: Asymptotic preserving (ap) schemes for multiscale kinetic and hyperbolic equations: a review. Lecture notes for summer school on methods and models of kinetic theory (M&MKT), Porto Ercole (Grosseto, Italy), pp. 177–216 (2010)
10. Degond, P., Deluzet, F.: Asymptotic-preserving methods and multiscale models for plasma physics. J. Comput. Phys. 336, 429–457 (2017)
11. Hu, J., Jin, S., Li, Q.: Asymptotic-preserving schemes for multiscale hyperbolic and kinetic equations. In Handbook of Numerical Analysis, vol 18, pp. 103–129. Elsevier (2017)
12. Golse, F., Jin, S., Levermore, C.D.: The convergence of numerical transfer schemes in diffusive regimes i: discrete-ordinate method. SIAM J. Numer. Anal. 36(5), 1333–1369 (1999)
13. Pareschi, L., Caflisch, R.E.: An implicit Monte Carlo method for rarefied gas dynamics: I. The space homogeneous case. J. Comput. Phys. 154(1), 90–116 (1999)
14. Dimarco, G., Pareschi, L.: Numerical methods for kinetic equations. Acta Numer. 23, 369–520 (2014)
15. Pareschi, L., Russo, Giovanni: Time relaxed Monte Carlo methods for the Boltzmann equation. SIAM J. Sci. Comput. 23(4), 1253–1273 (2001)
16. Weinan, E., Yu, B.: The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. Commun. Math. Stat. 6(1), 1–12 (2018)
17. 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)
18. Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. Nat. Mach. Intell. 3(3), 218–229 (2021)
19. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 (2020)
20. Beck, C., Hutzenthaler, M., Jentzen, A., Kuckuck, B.: An overview on deep learning-based approximation methods for partial differential equations. arXiv preprint arXiv:2012.12348 (2020)
21. Liao, Y., Ming, P.: Deep Nitsche method: deep Ritz method with essential boundary conditions. arXiv preprint arXiv:1912.01309 (2019)
22. Sirignano, J., Spiliopoulos, Konstantinos: DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. 375, 1339–1364 (2018)
23. Zang, Y., Bao, G., Ye, X., Zhou, H.: Weak adversarial networks for high-dimensional partial differential equations. J. Comput. Phys. p. 109409 (2020)
24. Cai, Z., Chen, J., Liu, M.: Least-squares Relu neural network (ISNN) method for linear advection-reaction equation. J. Comput. Phys. p 110514 (2021)
25. Lyu, L., Zhang, Z., Chen, M., Chen, J.: MIM: a deep mixed residual method for solving high-order partial differential equations. arXiv preprint arXiv:2006.04146 (2020)
26. 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)
27. Chen, Z., Liu, L., Mu, L.: Solving the linear transport equation by a deep neural network approach. arXiv preprint arXiv:2102.09157 (2021)
28. Li, L., Yang, C.: Asymptotic preserving scheme for anisotropic elliptic equations with deep neural network. arXiv preprint arXiv:2104.05337 (2021)
29. Delgadillo, R. A., Hu, J., Yang, H.: Multiscale and nonlocal learning for pdes using densely connected rnns. arXiv preprint arXiv:2109.02362 (2018)
30. Ryzhik, L., Papanicolaou, G., Keller, J. B.: Transport equations for elastic and other waves in random media. Wave Motion, 24(4):327–370 (1996)
31. Bardos, C., Santos, R., Sentis, Rémi.: Diffusion approximation and computation of the critical size. Trans. Am. Math. Soc. 318(2), 617–649 (1984)
32. Jin, S., Pareschi, L., Toscani, Giuseppe: Uniformly accurate diffusive relaxation schemes for multiscale transport equations. SIAM J. Numer. Anal. 39(3), 913–936 (2000)
33. Lemou, M., Mieussens, Luc: A new asymptotic preserving scheme based on micro-macro formulation for linear kinetic equations in the diffusion limit. SIAM J. Sci. Comput. 31(1), 334–368 (2008)
34. Le Maître, O. P., Knio, O. M.: Spectral Methods for Partial Differential Equations: With Applications in Science and Engineering (2010)
35. Wensch, J., Senf, H., Schaller, G.: The multilevel Monte Carlo method for partial differential equations. Acta Numer. 21, 517–638 (2012)
36. Giles, M. B.: Multilevel Monte Carlo path simulation. Monte Carlo Methods and Applications 14(3), 269–317 (2008)
37. Giles, M. B.: Multilevel Monte Carlo methods that scale logarithmically. Stochastics 83(5-6), 601–618 (2011)
38. Giles, M. B.: Multilevel Monte Carlo path simulation with non-smooth payoff. Monte Carlo Methods and Applications 18(1-2), 137–155 (2012)
39. Giles, M. B.: Multilevel Monte Carlo path simulation for stochastic differential equations. Monte Carlo Methods and Applications 19(1), 35–48 (2013)
40. Giles, M. B.: Multilevel Monte Carlo method for markovian systems of stochastic differential equations. Monte Carlo Methods and Applications 20(2), 129–145 (2014)
41. Giles, M. B.: Multilevel Monte Carlo method for markovian systems of stochastic differential equations. Monte Carlo Methods and Applications 20(2), 129–145 (2014)
42. Giles, M. B.: Multilevel Monte Carlo method for markovian systems of stochastic differential equations. Monte Carlo Methods and Applications 20(2), 129–145 (2014)
43. Giles, M. B.: Multilevel Monte Carlo method for markovian systems of stochastic differential equations. Monte Carlo Methods and Applications 20(2), 129–145 (2014)
44. Giles, M. B.: Multilevel Monte Carlo method for markovian systems of stochastic differential equations. Monte Carlo Methods and Applications 20(2), 129–145 (2014)
45. Giles, M. B.: Multilevel Monte Carlo method for markovian systems of stochastic differential equations. Monte Carlo Methods and Applications 20(2), 129–145 (2014)
46. Giles, M. B.: Multilevel Monte Carlo method for markovian systems of stochastic differential equations. Monte Carlo Methods and Applications 20(2), 129–145 (2014)
47. Giles, M. B.: Multilevel Monte Carlo method for markovian systems of stochastic differential equations. Monte Carlo Methods and Applications 20(2), 129–145 (2014)