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

Shi Jin[1, 2], Zheng Ma[1, 2, 3], and Keke Wu[1, *]

[1]School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai, 200240, P. R. China.
[2] Institute of Natural Sciences, MOE-LSC, Shanghai Jiao Tong University, Shanghai, 200240, P. R. China.
[3]Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai, 200240, China

October 31, 2021

Contents

1 Introduction **2**

2 The transport equation and its diffusion limit **4**

3 The micro-macro decomposition **5**

4 Asymptotic-Preserving Neural Networks **7**
4.1 The failure of PINNs to resolve small scales 8
4.2 The micro-macro decomposition based APNN method 9

5 Numerical examples **11**
5.1 One-dimensional problems 13
5.2 Two-dimensional problems 15
5.3 Uncertainty quantification (UQ) problems 17

6 Conclusion **18**

Abstract

In this paper we develop a neural network for the numerical simulaton 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

*Corresponding author: wukekever@sjtu.edu.cn

the problem. We first show that a standard Physics-Informed Neural Network (PINNs) 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 fit for the neural network approach. We construct a micro-macro decomposition based neutral 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 [5]. 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 [18, 13, 17]. Another difficulty is the multiscale nature. Kinetic equations usually contain small or multiple space and/or temporal scales, characterized by the Knudesen 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 [34]. When the Knudsen number is small, kinetic equations can often be approximated by macroscopic hydrodynamic or diffusion equations [3]. 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* [16]. Since the mid-1990's, the development of AP schemes for such problems has generated many interests [8, 16, 14]. 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 [29, 10, 30]. 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 rencely in many problems [11, 31, 27, 24]. 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 [2, 11, 25, 33, 31, 36, 4, 28, 24] for reviews and references therein. For applications in kinetic type equations see [15, 7, 23, 9]. The machine learning/DNNs approach has several advantages over the traditional numerical methods. First, it could deal with

high-dimensional, highly nonlinear PDEs due to the strong capacity/representativeness of DNNs. Second, it is a mesh-free method so it can deal with problems in complex domain and geometry. Third, it is user friendly to implement even for very complicated equations and boundary conditions. One does not need to construct sophisticated numerical schemes and design delicate numerical implementations of boundary conditions.

However, the DNN approach also brings some drawbacks compared with classicial nuemrical methods. The two well-known drawbacks are: lack of high accuracy and time consuming training time due to large number parameters in DNNs and the use of stochastic gradient method during the training step for high-dimensional non-convex functions. Most importantly, as we find in this article, that *the standard DeepRitz, PINNs, or NeuralOp approaches have di culties 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 [31], 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 [35] 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. Beyong 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 networks 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 Section 2, a brief introduction of the linear transport equation and its diffusion limit is given. The AP micro-macro decomposition is presented in Section 3. Section 4 is our main part where APNNs 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 multiscales and high-dimensional uncertainties are given in Section 5, which are solved by APNNs and compared with PINNs. The paper is concluded in Section 6.

2 The transport equation and its diffusion limit

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

$$\varepsilon \partial_t f + \mathsf{v} \quad \mathsf{r}_x f = \frac{1}{\varepsilon} \mathsf{L} f + \varepsilon Q, \tag{1}$$

with the collision operator

$$\mathsf{L} f = \frac{\sigma_S}{S} \int f \,\mathrm{d}\mathsf{v}' \quad \sigma f, \tag{2}$$

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

$$\sigma_S = \sigma \quad \varepsilon^2 \sigma_A, \tag{3}$$

where $\sigma_A = \sigma_A(\mathsf{x})$ is the absorption coefficient. Such an equation arises in neutron transport, radiative transfer [6], wave propagation in random media [32], etc. In all these applications, the scaling gives rise to a diffusion equation as $\varepsilon \; ! \; 0$ [1], which is

$$\partial_t \rho = D\, \mathsf{r}_x \quad \left(\frac{1}{\sigma} \mathsf{r}_x \rho \right) \quad \sigma_A \rho + Q\,, \tag{4}$$

where $\rho = \int f(\mathsf{v})\,\mathrm{d}\mathsf{v}$. 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 Ω is a unit sphere in two dimension. This diffusion approximation can be obtained by using the Hilbert expansion with the ansatz

$$f = f^{(0)} + \varepsilon f^{(1)} + \varepsilon^2 f^{(2)} + \tag{5}$$

and balance the terms by the order in ε. The diffusion approximation can also be derived using the so-called even- and odd-parities [19], or micro-macro decomposition technique [22] and various Asymptotic-preserving schemes can be constructed based on these methods [16].

For example in the 1-d case, by splitting equation (1) and define evne- and odd-parities as

$$\begin{aligned} r(t,x,v) &= \frac{1}{2}[f(t,x,v) + f(t,x, \quad v)], \\ j(t,x,v) &= \frac{1}{2\varepsilon}[f(t,x,v) \quad f(t,x, \quad v)], \end{aligned} \tag{6}$$

one can obtain

$$\begin{aligned} \partial_t r + v\partial_x j &= \frac{\sigma_S}{\varepsilon^2}(\rho \quad r) \quad \sigma_A r + Q, \\ \partial_t j + \frac{v}{\varepsilon^2}\partial_x r &= \quad \frac{\sigma_S}{\varepsilon^2} j \quad \sigma_A j. \end{aligned} \tag{7}$$

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

$$\partial_t \rho = \quad v^2 \partial_x (\frac{1}{\sigma_S}\partial_x \rho) \quad \sigma_A \rho + Q, \tag{8}$$

which after integrating over v yields to the diffusion limit equation (4).

One thing we would like to mention here is that not all classical AP formulations, based on which AP schemes have been constructed, can be used to construct the correct APNN 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 $T \quad D \quad \Omega$:

$$\begin{cases} \varepsilon\partial_t f + v \quad r_x f = \frac{1}{\varepsilon} L f + \varepsilon Q, & (t,x,v) \; 2 \; T \quad D \quad \Omega, \\ Bf = F_{\mathsf{B}}, & (t,x,v) \; 2 \; T \quad \partial D \quad \Omega, \\ I f = f_0, & (t,x,v) \; 2 \; ft = 0g \quad D \quad \Omega, \end{cases} \tag{9}$$

where F_{B}, f_0 are given functions; ∂D is the boundary of D, and B, I are initial and boundary operators, respectively. More precisely we consider the 1-d and 2-d cases in this paper.

One-dimensional case Consider the one-dimensional transport equation in slab geometry

$$\varepsilon\partial_t f + v\partial_x f = \frac{1}{\varepsilon}\left(\frac{\sigma_S}{2}\int_1^1 f \; \mathrm{d}v^{\emptyset} \quad \sigma f\right) + \varepsilon Q, \quad x_L < x < x_R, \qquad 1 \quad v \quad 1, \tag{10}$$

with in-flow boundary conditions as,

$$\begin{aligned} f(t,x_L,v) &= F_L(v) \quad \text{for} \quad v > 0, \\ f(t,x_R,v) &= F_R(v) \quad \text{for} \quad v < 0, \end{aligned} \tag{11}$$

or periodic boundary condition

$$f(t, x_L, v) = f(t, x_R, v). \tag{12}$$

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

$$f(0, x, v) = f_0(x, v). \tag{13}$$

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

$$\varepsilon \partial_t f + \mathbf{v} \quad \partial_{\mathbf{x}} f = \frac{1}{\varepsilon} \left(\frac{\sigma_S}{2\pi} \int_{|\mathbf{v}|=1} f \, \mathrm{d}\mathbf{v}' \quad \sigma f \right) + \varepsilon Q, \quad \mathbf{x} \; 2 \; \Gamma \quad \mathbb{R}^2, \quad |\mathbf{v}| = 1, \tag{14}$$

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

$$f(t, \mathbf{x}, \mathbf{v}) = F_B(\mathbf{x}, \mathbf{v}) \quad \text{for} \quad \mathbf{n} \quad \mathbf{v} < 0, \quad \mathbf{x} \; 2 \; \partial\Gamma, \tag{15}$$

where $\mathbf{n}$ is the outer normal of the boundary. The initial condition is

$$f(0, \mathbf{x}, \mathbf{v}) = f_0(\mathbf{x}, \mathbf{v}). \tag{16}$$

In this section, we describe the micro-macro decomposition formulation for the linear transport equation. It is a useful mechanism to build AP schemes [22]. The idea begins with the decomposition of f into the equilibrium part ρ and the non-equilibrium part g:

$$f = \rho + \varepsilon g, \tag{17}$$

where

$$\rho = \frac{1}{S} \int f \, \mathrm{d}\mathbf{v}'. \tag{18}$$

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

$$\langle g \rangle := \frac{1}{S} \int g \, \mathrm{d}\mathbf{v}' = 0. \tag{19}$$

Applying equation (17) in (1) gives

$$\varepsilon \partial_t \rho + \varepsilon^2 \partial_t g + \mathbf{v} \quad \nabla_{\mathbf{x}} \rho + \varepsilon \mathbf{v} \quad \nabla_{\mathbf{x}} g = \mathcal{L} g + \varepsilon Q. \tag{20}$$

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

$$\partial_t \rho + \nabla_{\mathbf{x}} \quad \langle \mathbf{v} g \rangle = Q. \tag{21}$$

Define operator $\Pi : \Pi(\,)(\mathbf{v}) = \langle \, \rangle$ and I is the identity operator. Then an evolution equation on g is found by applying the orthogonal projection $I \quad \Pi$ to equation (20):

$$\varepsilon^2 \partial_t g + \varepsilon (I \quad \Pi)(\mathbf{v} \quad \nabla_{\mathbf{x}} g) + \mathbf{v} \quad \nabla_{\mathbf{x}} \rho = \mathcal{L} g + (I \quad \Pi) \varepsilon Q. \tag{22}$$

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

$$\begin{cases} \partial_t \rho + \nabla_x \cdot \langle v g \rangle = Q, \\ \varepsilon^2 \partial_t g + \varepsilon (I - \Pi)(v \cdot \nabla_x g) + v \cdot \nabla_x \rho = L g + (I - \Pi)\varepsilon Q, \\ \langle g \rangle = 0. \end{cases} \tag{23}$$

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

$$\begin{cases} \partial_t \rho + \nabla_x \cdot \langle v g \rangle = Q, \\ \varepsilon^2 \partial_t g + \varepsilon (I - \Pi)(v \cdot \nabla_x g) + v \cdot \nabla_x \rho = L g + (I - \Pi)\varepsilon Q, \\ \langle g \rangle(0, x) = 0. \end{cases} \tag{24}$$

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,

$$\partial_t \langle g \rangle = 0. \tag{25}$$

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

$$\begin{cases} \partial_t \rho + \nabla_x \cdot \langle v g \rangle = Q, \\ v \cdot \nabla_x \rho = L g. \end{cases} \tag{26}$$

The second equation yields

$$g = L^{-1}(v \cdot \nabla_x \rho) \tag{27}$$

which, when plugging into the first equation and integrating over v, gives the diffusion equation (4).

Remark 1. *In (23) or (24) we singled out the condition $\langle g \rangle = 0$. This is not necessary in constructing classical AP schemes [22]. 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,

$$\begin{aligned} f_\theta^{[0]}(x) &= x, \\ f_\theta^{[l]}(x) &= \sigma\left(W^{[l-1]} f_\theta^{[l-1]}(x) + b^{[l-1]}\right), \quad 1 \le l \le L-1, \\ f_\theta(x) &= f_\theta^{[L]}(x) = W^{[L-1]} f_\theta^{[L-1]}(x) + b^{[L-1]}, \end{aligned} \tag{28}$$

[1] BAAI.2020. Suggested Notation for Machine Learning. https://github.com/mazhengcn/suggested-notation-for-machine-learning.

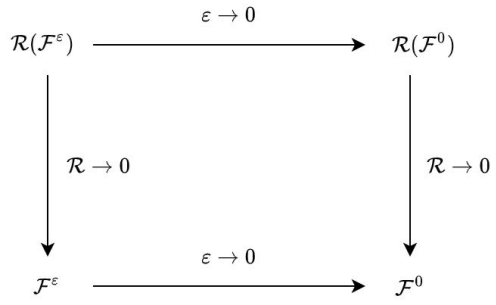

Figure 1: Illustration of APNNs. F^ε is the microscopic equation that depends on the small scale parameter ε and F^0 is its macroscopic limit as $\varepsilon \; ! \; 0$, which is independent of ε. The latent solution of F^ε is approximated by neural networks with its measure denoted by $R(F^\varepsilon)$. The asymptotic limit of $R(F^\varepsilon)$ as $\varepsilon \; ! \; 0$, if exists, is denoted by $R(F^0)$. If $loss(F^0)$ is a good measure of F^0, then it is called asymptotic-preserving (AP).

where $W^{[l]} \; 2 \; \mathsf{R}^{m_{l+1} \;\; m_l}, b^l \; 2 \; \mathsf{R}^{m_{l+1}}, m_0 = d_{in} = d$ is the input dimension, $m_L = d_0$ is the output dimension, σ is a scalar function and " " means entry-wise operation. We denote the set of parameters by θ. For simplicity of neural network presentation, we denote the layers by a list, i.e., $[m_0, \quad , m_L]$.

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 ε in equation (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 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 λ_1, λ_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 $f(t, x, v)$,

$$\mathrm{NN}_\theta(t, \mathsf{x}, \mathsf{v}) \quad f(t, \mathsf{x}, \mathsf{v}). \tag{29}$$

The inputs of DNN are $(t, \mathsf{x}, \mathsf{v})$, 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 $(t, \mathsf{x}, \mathsf{v})$. Since f is always non-negative, we put an exponential function at the last output layer of the DNN. To be precise, we use

$$f_\theta^{\mathsf{NN}}(t, \mathsf{x}, \mathsf{v}) := \exp\Big(\quad \tilde{f}_\theta^{\mathsf{NN}}(t, \mathsf{x}, \mathsf{v})\Big) \quad f(t, \mathsf{x}, \mathsf{v}) \tag{30}$$

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

conditions as penalty terms, that is,

$$
\begin{aligned}
R^{\varepsilon}_{\mathsf{PINN}} =& \frac{1}{|T| \cdot |D| \cdot |\Omega|} \int_T \int_D \int \left| \varepsilon^2 \partial_t f_\theta^{\mathsf{NN}} + \varepsilon v \cdot \nabla_x f_\theta^{\mathsf{NN}} - L f_\theta^{\mathsf{NN}} - \varepsilon^2 Q \right|^2 \mathrm{d}v \, \mathrm{d}x \, \mathrm{d}t \\
&+ \frac{\lambda_1}{|T| \cdot |\partial D| \cdot |\Omega|} \int_T \int_{\partial D} \int |B f_\theta^{\mathsf{NN}} - F_{\mathsf{B}}|^2 \, \mathrm{d}v \, \mathrm{d}x \, \mathrm{d}t \\
&+ \frac{\lambda_2}{|D| \cdot |\Omega|} \int_D \int |I f_\theta^{\mathsf{NN}} - f_0|^2 \, \mathrm{d}v \, \mathrm{d}x,
\end{aligned} \tag{31}
$$

where λ_1 and λ_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)

$$
R^{\varepsilon}_{\mathsf{PINN,\ residual}} := \frac{1}{|T| \cdot |D| \cdot |\Omega|} \int_T \int_D \int \left| \varepsilon^2 \partial_t f_\theta^{\mathsf{NN}} + \varepsilon v \cdot \nabla_x f_\theta^{\mathsf{NN}} - L f_\theta^{\mathsf{NN}} - \varepsilon^2 Q \right|^2 \mathrm{d}v \, \mathrm{d}x \, \mathrm{d}t. \tag{32}
$$

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

$$
R_{\mathsf{PINN,\ residual}} := \frac{1}{|T| \cdot |D| \cdot |\Omega|} \int_T \int_D \int \left| L f_\theta^{\mathsf{NN}} \right|^2 \mathrm{d}v \, \mathrm{d}x \, \mathrm{d}t, \tag{33}
$$

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

$$
L f = 0. \tag{34}
$$

This shows that when ε is very small, to the leading order we are solving equation $Lf = 0$ which gives $f = \rho$. This does not give the desired diffusion equation (4). This explains why PINN will fail when ε is small. We also conduct numerical experiments to verify this claim in Section 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) insteal of the original equation (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

$$
\rho_\theta^{\mathsf{NN}}(t, x) := \exp\left(\tilde{\rho}_\theta^{\mathsf{NN}}(t, x) \right) \cdot \rho(t, x), \tag{35}
$$

Notice here ρ is non-negative. Second,

$$
g_\theta^{\mathsf{NN}}(t, x, v) := \tilde{g}_\theta^{\mathsf{NN}}(t, x, v) - \langle \tilde{g}_\theta^{\mathsf{NN}} \rangle(t, x) \cdot g(t, x, v). \tag{36}
$$

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

$$
\langle g_\theta^{\mathsf{NN}} \rangle = \langle \tilde{g}_\theta^{\mathsf{NN}} \rangle - \langle \tilde{g}_\theta^{\mathsf{NN}} \rangle = 0, \quad \forall\, t, x, \tag{37}
$$

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,

$$\begin{aligned} R^{\varepsilon}_{\text{APNN}} =& \frac{1}{|T|\,|D|} \int_T \int_D |\partial_t \rho_\theta^{\text{NN}} + \nabla_x \cdot \langle v g_\theta^{\text{NN}} \rangle|^2 \,\mathrm{d}x\,\mathrm{d}t \\ &+ \frac{1}{|T|\,|D|\,|\Omega|} \int_T \int_D \int |\varepsilon^2 \partial_t g_\theta^{\text{NN}} + \varepsilon (I - \Pi)(v \cdot \nabla_x g_\theta^{\text{NN}}) \\ &\quad + v \cdot \nabla_x \rho_\theta^{\text{NN}} - L g_\theta^{\text{NN}} - (I - \Pi)\varepsilon Q|^2 \,\mathrm{d}v\,\mathrm{d}x\,\mathrm{d}t \\ &+ \frac{\lambda_1}{|T|\,|\partial D|\,|\Omega|} \int_T \int_{\partial D} \int |B(\rho_\theta^{\text{NN}} + \varepsilon g_\theta^{\text{NN}}) - F_B|^2 \,\mathrm{d}v\,\mathrm{d}x\,\mathrm{d}t \\ &+ \frac{\lambda_2}{|D|\,|\Omega|} \int_D \int |I(\rho_\theta^{\text{NN}} + \varepsilon g_\theta^{\text{NN}}) - f_0|^2 \,\mathrm{d}v\,\mathrm{d}x. \end{aligned} \tag{38}$$

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

$$\begin{aligned} R^{\varepsilon}_{\text{APNN, residual}} =& \frac{1}{|T|\,|D|} \int_T \int_D |\partial_t \rho_\theta^{\text{NN}} + \nabla_x \cdot \langle v g_\theta^{\text{NN}} \rangle|^2 \,\mathrm{d}x\,\mathrm{d}t \\ &+ \frac{1}{|T|\,|D|\,|\Omega|} \int_T \int_D \int \Big| \varepsilon^2 \partial_t g_\theta^{\text{NN}} + \varepsilon (I - \Pi)(v \cdot \nabla_x g_\theta^{\text{NN}}) + v \cdot \nabla_x \rho_\theta^{\text{NN}} \\ &\qquad - L g_\theta^{\text{NN}} - (I - \Pi)\varepsilon Q \Big|^2 \,\mathrm{d}v\,\mathrm{d}x\,\mathrm{d}t \end{aligned} \tag{39}$$

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

$$\begin{aligned} R_{\text{APNN, residual}} =& \frac{1}{|T|\,|D|} \int_T \int_D |\partial_t \rho_\theta^{\text{NN}} + \nabla_x \cdot \langle v g_\theta^{\text{NN}} \rangle|^2 \,\mathrm{d}x\,\mathrm{d}t \\ &+ \frac{1}{|T|\,|D|\,|\Omega|} \int_T \int_D \int \Big| v \cdot \nabla_x \rho_\theta^{\text{NN}} - L g_\theta^{\text{NN}} \Big|^2 \,\mathrm{d}v\,\mathrm{d}x\,\mathrm{d}t, \end{aligned} \tag{40}$$

which is the least square loss of equations (26)

$$\begin{cases} \partial_t \rho + \nabla_x \cdot \langle v g \rangle = Q, \\ v \cdot \nabla_x \rho = L g. \end{cases} \tag{41}$$

Same as in Section 3.1 the second equation yields $g = L^{-1}(v \cdot \nabla_x \rho)$, which, when plugging into the first equation and integrating over v, gived the diffusion equation (4). Hence this proposed method is an APNN method. Finally we put a schematic plot of our method in Figure 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 λ_3, i.e., without using (19), we use $\hat{g}_\theta^{NN}$ and modifies the loss as*

$$R_{APNN,\ residual} + \frac{\lambda_3}{|T|\,|D|} \int_T \int_D |\langle \tilde{g}_\theta^{NN} \rangle - 0|^2 \,\mathrm{d}x\,\mathrm{d}t. \tag{42}$$

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

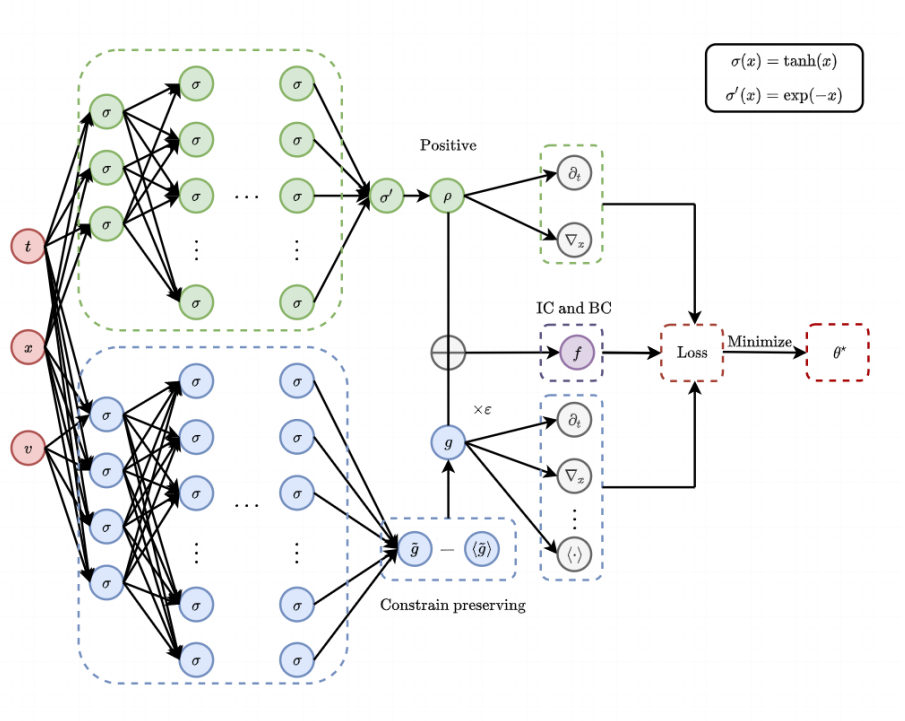

Figure 2: Schematic of APNNs for solving the linear transport equation with initial and boundary conditons.

Remark 3. *The parity formulation (7) also allows one to construct AP schemes [19]. However if one uses it in the loss, it will not be an APNN. To see this, consider the case when* $\varepsilon \to 0$. *Since the DNN will only pick up the leading term, thus one has, in the* L^2 *sense,*

$$\begin{aligned} r &= \rho \quad r \\ j &= \quad \frac{v}{\sigma_S}\partial_x r\,. \end{aligned} \tag{43}$$

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

Remark 4. *A uniform in* ε *convergence of APNNs, as the number of unknowns in* θ *goes to infinity, can be proved along the lines of [15, 7], combined with the uniform stability for the micro-macro system (24) (see [26], even with uncertainty[17]). We omit the details.*

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 ($\varepsilon \quad O(1)$) to hydrodynamic (diffusive) regimes ($\varepsilon \to 0$), 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 $\langle\, \rangle$ and $\Pi(\,)$ with quadrature rule. The optimization problem is solved by the Adam version of the gradient descent method [21]. Specifically, in most of our experiments, we fix the x 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 $L/\langle\, \rangle$ is computed by quadrature rule (the Gauss-Legendre Integration). In detail, assume $\{w_i, v_i^0\}_{i=1}^n$ are the nodes and weights, which $\{v_i^0\}_{i=1}^n$ 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, $\int_{-1}^{1} f(v)\,\mathrm{d}v$ by the summation of linear combination of $f(v_i^0) : \sum_{i=1}^n w_i f(v_i^0)$.

The empirical risk for PINN is as follows

$$\begin{aligned} R^{\varepsilon}_{\mathsf{PINN}} =& \frac{1}{N_1}\sum_{i=1}^{N_1} \left|\varepsilon^2 \partial_t f_\theta^{\mathsf{NN}}(t_i,x_i,v_i) + \varepsilon v \cdot \nabla_x f_\theta^{\mathsf{NN}}(t_i,x_i,v_i) - L f_\theta^{\mathsf{NN}}(t_i,x_i) - \varepsilon^2 Q\right|^2 \\ &+ \frac{\lambda_1}{N_2}\sum_{i=1}^{N_2} |B f_\theta^{\mathsf{NN}}(t_i,x_i,v_i) - F_{\mathsf{B}}(t_i,x_i,v_i)|^2 \\ &+ \frac{\lambda_2}{N_3}\sum_{i=1}^{N_3} |I f_\theta^{\mathsf{NN}}(t_i,x_i,v_i) - f_0(t_i,x_i,v_i)|^2, \end{aligned} \tag{44}$$

where N_1, N_2, N_3 are the number of sample points of $\Omega_T \times D \times \Omega$, $\partial\Omega_T \times D \times \Omega$, $\Omega_T \times \partial D \times \Omega$, $\Omega_T \times D \times \Omega$.

Similarly, the empirical risk for APNN is as follows

$$\begin{aligned} R^{\varepsilon}_{\mathsf{APNN}} =& \frac{1}{N_1^{(1)}}\sum_{i=1}^{N_1^{(1)}} |\partial_t \rho_\theta^{\mathsf{NN}}(t_i,x_i) + \nabla_x \cdot \langle v g_\theta^{\mathsf{NN}}\rangle(t_i,x_i) - Q|^2 \\ &+ \frac{1}{N_1^{(2)}}\sum_{i=1}^{N_1^{(2)}} |\varepsilon^2 \partial_t g_\theta^{\mathsf{NN}}(t_i,x_i,v_i) + \varepsilon(I - \Pi)(v \cdot \nabla_x g_\theta^{\mathsf{NN}})(t_i,x_i) \\ &\quad + v_i \cdot \nabla_x \rho_\theta^{\mathsf{NN}}(t_i,x_i) - L g_\theta^{\mathsf{NN}}(t_i,x_i) - (I - \Pi)\varepsilon Q|^2 \\ &+ \frac{\lambda_1}{N_2}\sum_{i=1}^{N_2} |B(\rho_\theta^{\mathsf{NN}}(t_i,x_i) + \varepsilon g_\theta^{\mathsf{NN}}(t_i,x_i,v_i)) - F_{\mathsf{B}}(t_i,x_i,v_i)|^2 \\ &+ \frac{\lambda_2}{N_3}\sum_{i=1}^{N_3} |I(\rho_\theta^{\mathsf{NN}}(t_i,x_i) + \varepsilon g_\theta^{\mathsf{NN}}(t_i,x_i,v_i)) - f_0(t_i,x_i,v_i)|^2. \end{aligned} \tag{45}$$

where $N_1^{(1)}, N_1^{(2)}, N_2, N_3$ are the number of sample points of $\Omega_T \times D, \Omega_T \times D \times \Omega$, $\partial\Omega_T \times D \times \Omega$, $\Omega_T \times \partial D \times \Omega$, $\Omega_T \times D \times \Omega$.

To investigate the influence of the Monte Carlo method in the integral, 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 methods. For most of the time we will check the relative ℓ^2 error of the density $\rho(x)$ between DNN methods and reference

solutions, e.g. for 1d case,

$$\text{error} := \sqrt{\frac{\sum_j j\rho_{\theta,j}^{\mathsf{NN}} \quad \rho_j^{\mathsf{ref}} j^2}{\sum_j j\rho_j^{\mathsf{ref}} j^2}} \tag{46}$$

5.1 One-dimensional problems

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

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

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

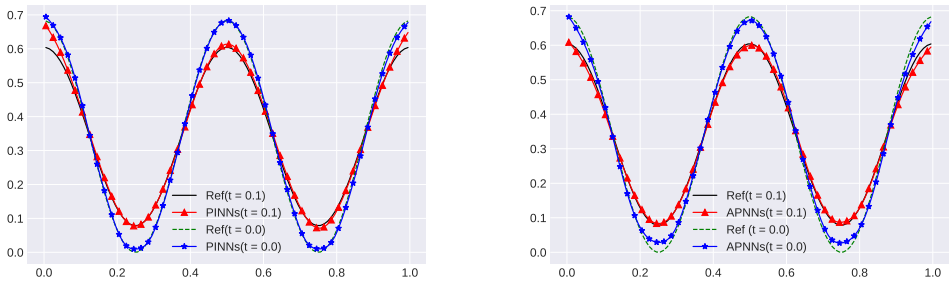

Figure 3: Problems I. Plot of density ρ at $t = 0.0, 1.0$. Left: PINNs vs. Ref, Right: APNNs vs. Ref. $\varepsilon = 1$ and neural networks are $[2, 128, 128, 128, 128, 1]$ for ρ and $[3, 256, 256, 256, 256, 1]$ for g, f. Batch size is 2000 in domain, 500 2 with penalty $\lambda_1 = 1$ for boundary condition and 1000 with penalty $\lambda_2 = 1000$ for initial condition, the number of quadrature points is 30. Relative ℓ^2 error of PINNs and APNNs are 4.09 10 ^{2}, 2.66 10 ^{2}.

Problem II. In-flow boundary condition In the second example we consider the isotropic in-flow boundary conditions:

$$x \; 2 \; [0,1], \quad F_L(v) = 1, \quad F_R(v) = 0. \tag{50}$$

The source term, scattering and absorbing coefficients are set as

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

The results are shown in Figure 4 to Fig. 7 with exactly $\hbar g i = 0$, and in Figure 9 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. Clearly for small ε PINN does not give accurate results while APNN gives quite accurate results for all ε tested. Figure 10 shows the result where the integral is computed by the Monte Carlo method. The training process about loss for PINNs and APNNs with case $\varepsilon = 10^{-8}$ are plotted in Figure 8, which shows that APNN has a much smaller training error.

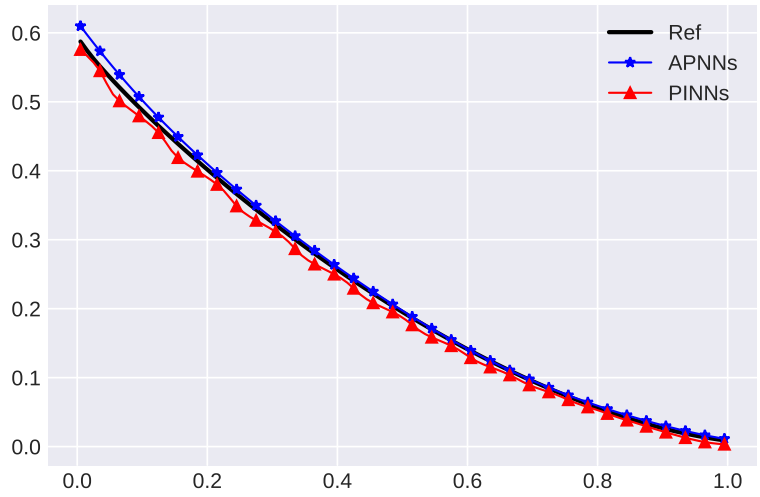

Figure 4: Problems II. Plot of density ρ at $t = 1.0$. For simplicity of neural network presentation, we denote the layers of neural network by a list: $[m_0, \quad , m_L]$. Left: PINNs vs. Ref, Right: APNNs vs. Ref. $\varepsilon = 1$ and neural networks are $[2, 128, 128, 128, 128, 1]$ for ρ and $[3, 256, 256, 256, 256, 1]$ for g, f. Batch size is 1000 in domain, 400 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 ℓ^2 errors of PINNs and APNNs are 4.01 10^{-2}, 1.36 10^{-2} respectively.

Problem III. A variable scattering coefficient Let

$$x \in [0, 1], \quad F_L(v) = 1, \quad F_R(v) = 0. \tag{52}$$

The source term, scattering and absorbing coefficients are set as

$$\sigma_S = 1 + (10x)^2, \quad \sigma_A = 0, \quad Q = 1, \quad \varepsilon = 0.01. \tag{53}$$

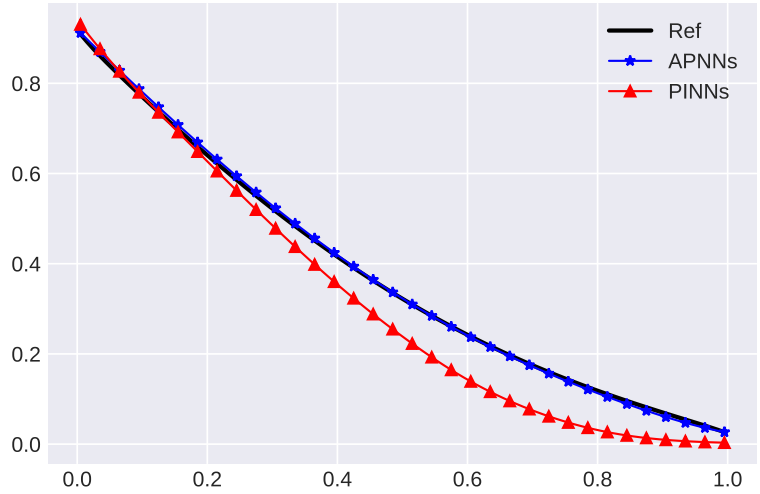

Figure 5: Problems II. Plot of density ρ at $t = 0.5$. Left: PINNs vs. Ref, Right: APNNs vs. Ref. $\varepsilon = 10^{-1}$ and neural networks are $[2, 128, 128, 128, 128, 1]$ for ρ and $[3, 256, 256, 256, 256, 1]$ for g, f. Batch size is 1000 in domain, 400 $\partial\Omega$ 2 with penalty $\lambda_1 = 1$ for boundary condition and 500 with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is 30.Relative ℓ^2 errors of PINNs and APNNs are $1.17 \times 10^{-1}, 3.30 \times 10^{-2}$ respectively.

In Figure 11 we report the numerical solution 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 σ_S/ε ranges from $1/\varepsilon$! $O(1)$, a problem with mixing scales. The numerical results how reasonably good performance of APNN.

Problem IV. A problem with boundary layer Let

$$x \in [0, 1], \quad F_L(v) = 5\sin(v), \quad F_R(v) = 0. \tag{54}$$

The source term, scattering and absorbing coefficients are set as

$$\sigma_S = 1, \quad \sigma_A = 0, \quad Q = 0, \quad \varepsilon = 0.05. \tag{55}$$

Here since F_l depends on v, there is a boundary layer near $x = 0$. Fig. 11 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

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

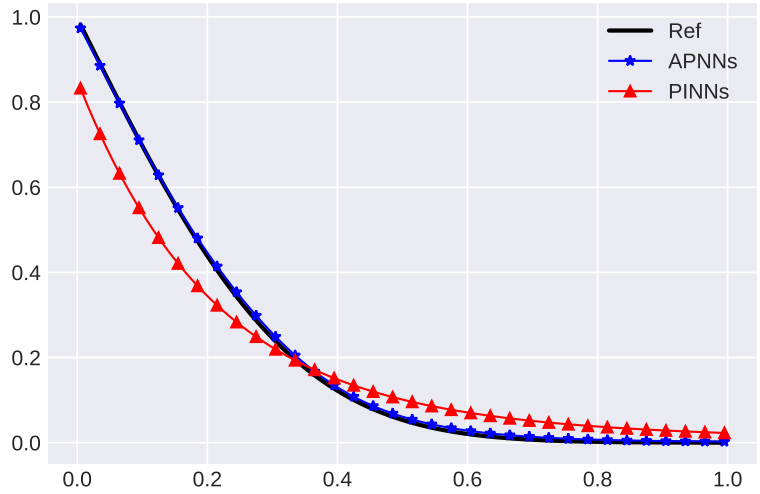

Figure 6: Problems II. Plot of density ρ at $t = 0.1$. Left: PINNs vs. Ref, Right: APNNs vs. Ref. $\varepsilon = 10^{-3}$ and neural networks are $[2, 128, 128, 128, 128, 1]$ for ρ and $[3, 256, 256, 256, 256, 1]$ for g, f. Batch size is 1000 in domain, 400×2 with penalty $\lambda_1 = 10$ for boundary condition and 1000 with penalty $\lambda_2 = 10$ for initial condition, the number of quadrature points is 30. Relative ℓ^2 errors of PINNs and APNNs are $2.17 \times 10^{-1}, 1.98 \times 10^{-2}$ respectively.

The source term, scattering and absorbing coefficients are set as

$$\sigma_S = 1, \quad \sigma_A = 0, \quad Q = 1, \quad \varepsilon = 1. \tag{57}$$

Here $\boldsymbol{n}$ denotes the exterior unit normal vector on $\partial\Gamma$.

Figure 13 shows the density ρ trained by APNNs and reference at time $t = 1.0$. The residual between them are plotted in Figure 14. The results show the good performance of APNNs with absolute error about 0.06 and relative ℓ^2 error 5.78×10^{-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 $\varepsilon = 10^{-8}$. Figures 15 and 16 show the density ρ trained by APNNs and reference at time $t = 0.1$ and corresponding residual error.

Problem VII. Another diffusive regime The last two dimensional case is the same as Problem VI except $Q = 50$. Figures 17 and 18 show the density ρ trained by APNNs and reference at time $t = 0.1$ and corresponding residual error, which shows a good approximation by APNN.

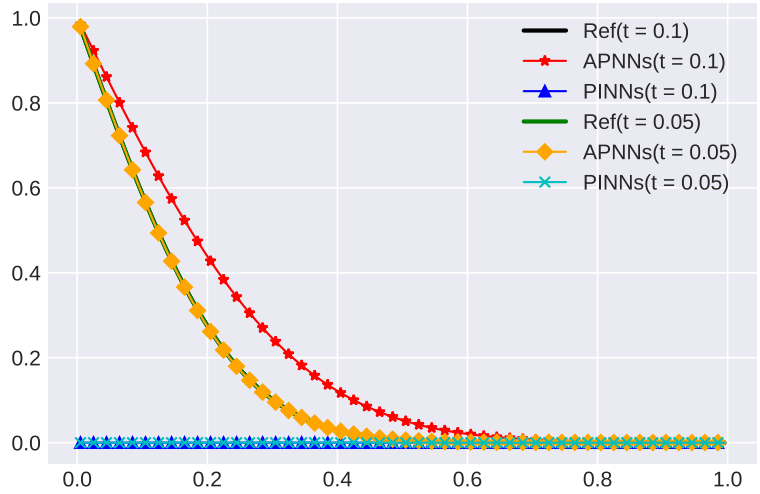

Figure 7: Problems II. Plot of density ρ at $t = 0.05, 0.1$. Left: PINNs vs. Ref, Right: APNNs vs. Ref. $\varepsilon = 10^{-8}$ and neural networks are $[2, 128, 128, 128, 128, 1]$ for ρ and $[3, 256, 256, 256, 256, 1]$ for g, f. Batch size is 1000 in domain, 400×2 with penalty $\lambda_1 = 10$ for boundary condition and 1000 with penalty $\lambda_2 = 10$ for initial condition, the number of quadrature points is 30. Relative ℓ^2 errors of PINNs and APNNs are $9.40 \times 10^{-1}, 2.76 \times 10^{-3}$ respectively.

5.3 Uncertainty quantification (UQ) problems

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

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

with scattering coefficients

$$\sigma_S(\boldsymbol{z}) = 1 + 0.3 \prod_{i=1}^{10} \sin(\pi z_i), \ \boldsymbol{z} = (z_1, z_2, \ldots, z_{10}) \sim U([-1, 1]^{10}) \tag{59}$$

or

$$\sigma_S(\boldsymbol{z}) = 1 + 0.3 \exp\left(-\frac{|\boldsymbol{z}|^2}{2} \right), \ \boldsymbol{z} = (z_1, z_2, \ldots, z_{20}) \sim U([-3, 3]^{20}) \tag{60}$$

and in-flow boundary condition as,

$$\begin{aligned} f(t, x_L = 0, v) &= F_L(v) = 1 \quad \text{for} \quad v > 0, \\ f(t, x_R = 1, v) &= F_R(v) = 0 \quad \text{for} \quad v < 0. \end{aligned} \tag{61}$$

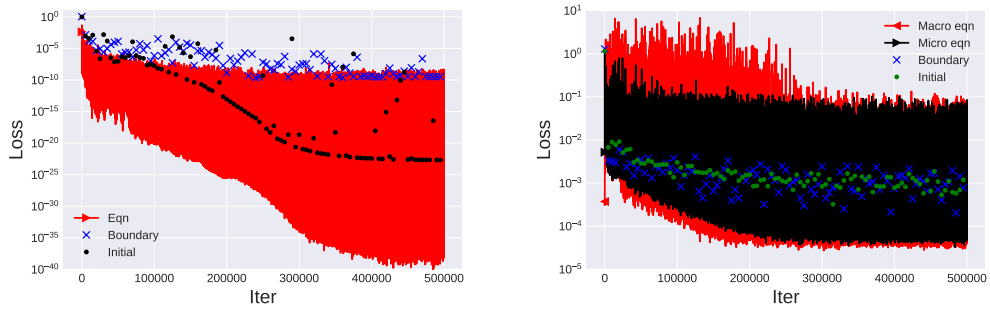

Figure 8: Plot of loss for PINNs and APNNs with case $\varepsilon = 10^{-8}$. Relative ℓ^2 errors of PINNs and APNNs are $9.40 \quad 10^{-1}, 2.76 \quad 10^{-3}$ respectively.

In this problem the 10-dimensional or 20-dimensional vector $\boldsymbol{z}$ represents the random input parameters in a typical uncertain problem setup [20]. Thus, $m_0 = 12, 13$ or $21, 22$ for ρ and g respectively. To compare numerical results, we evaluate ρ at $t = 0.05, 0.1$ by taking expectation on 10^4 times simulations for $(z_1, \quad, z_{10})$ or $(z_1, \quad, z_{20})$.

Problem VIII. UQ problem with $\varepsilon = 10^{-8}$ Let $\varepsilon = 10^{-8}$ and set scattering coefficients

$$\sigma_S(\boldsymbol{z}) = 1 + 0.3 \prod_{i=1}^{10} \sin(\pi z_i). \tag{62}$$

Figure 19 shows the density ρ trained by APNNs and reference at time $t = 0.05, 0.1$, for $\varepsilon = 10^{-8}$. As one can see, even for this high dimensional problem and a very small ε. APNN gives quite good results.

Problem IX. UQ problem with $\varepsilon = 10^{-3}$ Let $\varepsilon = 10^{-3}$ and set scattering coefficients same as Problems VIII. Figure 20 shows the density ρ train by APNNs and reference at time $t = 0.05, 0.1$. APNN gives good results.

Problem X. UQ problem with Gaussian scattering function Consider the linear transport equation with $\varepsilon = 10^{-8}$ and scattering coefficients

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

Figure 21 shows the density ρ train by APNNs and reference at time $t = 0.05, 0.1$.

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

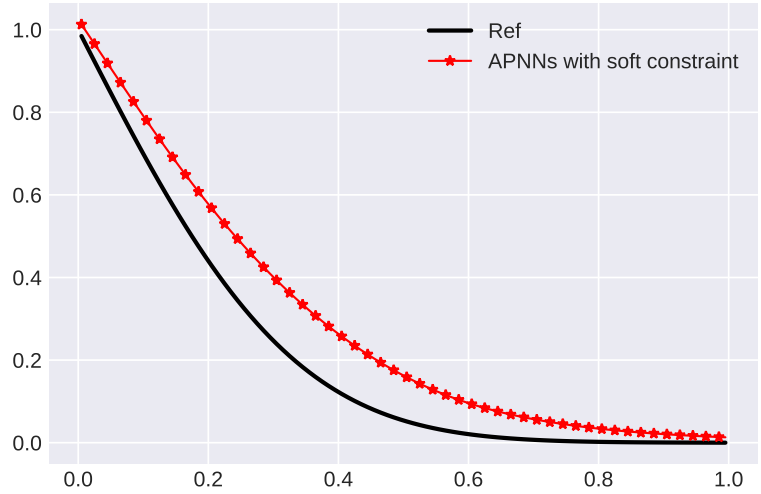

Figure 9: Problems II. Plot of density ρ at $t = 0.1$: APNNs with soft constraint $hgi = 0$ vs. Ref. $\varepsilon = 10^{-8}$ and neural networks are $[2, 128, 256, 256, 128, 1]$ for ρ and $[3, 128, 256, 512, 256, 128, 1]$ for g. Batch size is 1000 in domain, 400×2 with penalty $\lambda_1 = 10$ for boundary condition, 1000 with penalty $\lambda_2 = 10$ for initial condition and penalty $\lambda_3 = 1$ for constrain $hgi = 0$. Relative ℓ^2 error of APNNs with constrain $hgi = 0$ is 2.72×10^{-1}.

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.

Acknowledgement

This work is partially supported by the National Key R&D Program of China, Project Number 2020YFA0712000 and Shanghai Municipal of Science and Technology Major Project No. 2021SHZDZX0102. Shi Jin is also supported by NSFC grant No. 11871297 and Zheng Ma by NSFC Grant No. 12031013.

References

[1] Claude Bardos, Rafael Santos, and Rémi Sentis. Diffusion approximation and computation of the critical size. *Transactions of the american mathematical society*, 284(2):617–649, 1984.

[2] Christian Beck, Martin Hutzenthaler, Arnulf Jentzen, and Benno Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. *arXiv preprint arXiv:2012.12348*, 2020.

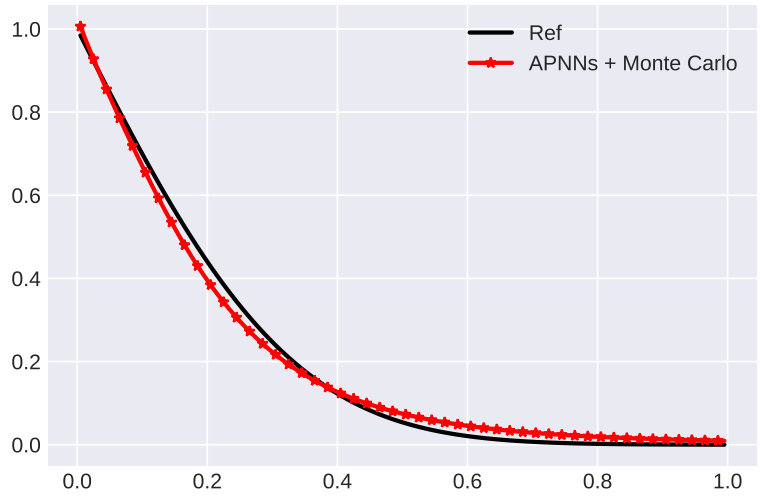

Figure 10: Problems II. Plot of density ρ at $t = 0.1$: APNNs with integral by Monte Carlo method vs. Ref. $\varepsilon = 10^{8}$ and neural networks are $[2, 128, 256, 256, 128, 1]$ for ρ and $[3, 128, 256, 512, 256, 128, 1]$ for g. Batch size is 800 in domain, 500 2 with penalty $\lambda_1 = 1$ for 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 ℓ^2 error of APNNs with constraint $hgi = 0$ is 6.73 10^{2}.

[3] François Bouchut, François Golse, and Mario Pulvirenti. *Kinetic equations and asymptotic theory*. Elsevier, 2000.

[4] Zhiqiang Cai, Jingshuang Chen, and Min Liu. Least-squares relu neural network (lsnn) method for linear advection-reaction equation. *Journal of Computational Physics*, page 110514, 2021.

[5] Carlo Cercignani. The boltzmann equation. In *The Boltzmann equation and its applications*, pages 40–103. Springer, 1988.

[6] Subrahmanyan Chandrasekhar. *Radiative transfer*. Courier Corporation, 2013.

[7] Zheng Chen, Liu Liu, and Lin Mu. Solving the linear transport equation by a deep neural network approach. *arXiv preprint arXiv:2102.09157*, 2021.

[8] Pierre Degond and Fabrice Deluzet. Asymptotic-preserving methods and multiscale models for plasma physics. *Journal of Computational Physics*, 336:429–457, 2017.

[9] Ricardo A Delgadillo, Jingwei Hu, and Haizhao Yang. Multiscale and nonlocal learning for pdes using densely connected rnns. *arXiv preprint arXiv:2109.01790*, 2021.

[10] Giacomo Dimarco and Lorenzo Pareschi. Numerical methods for kinetic equations. *Acta Numerica*, 23:369–520, 2014.

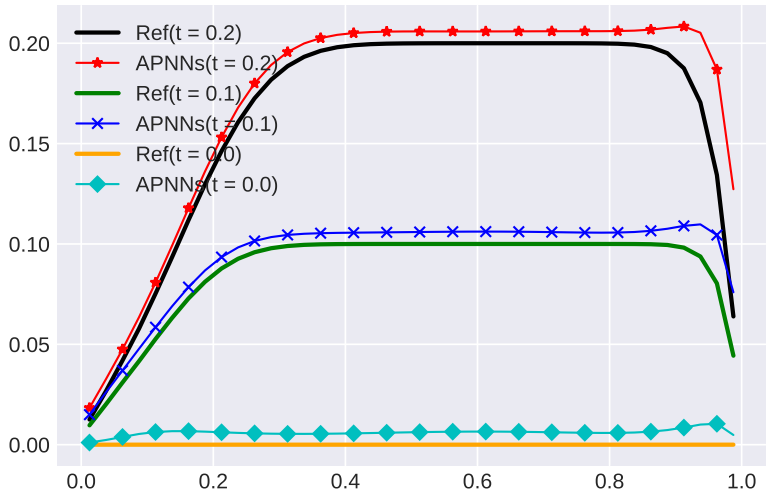

Figure 11: Problems III. Plot of density ρ at $t = 0.0, 0.1, 0.2$: APNNs vs. Ref. $\varepsilon = 10^{\ 2}$ and neural networks are $[2, 128, 128, 128, 128, 1]$ for ρ and $[3, 256, 256, 256, 256, 1]$ for g. Batch size is 500 in domain, 200 $\ 2$ with penalty $\lambda_1 = 1$ for boundary condition and 200 with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is 30.

[11] Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[12] François Golse, Shi Jin, and C David Levermore. The convergence of numerical transfer schemes in diffusive regimes i: Discrete-ordinate method. *SIAM journal on numerical analysis*, 36(5):1333–1369, 1999.

[13] Jingwei Hu and Shi Jin. A stochastic galerkin method for the boltzmann equation with uncertainty. *Journal of Computational Physics*, 315:150–168, 2016.

[14] Jingwei Hu, Shi Jin, and Qin Li. Asymptotic-preserving schemes for multiscale hyperbolic and kinetic equations. In *Handbook of Numerical Analysis*, volume 18, pages 103–129. Elsevier, 2017.

[15] Hyung Ju Hwang, Jin Woo Jang, Hyeontae Jo, and Jae Yong Lee. Trend to equilibrium for the kinetic fokker-planck equation via the neural network approach. *Journal of Computational Physics*, 419:109665, 2020.

[16] Shi Jin. 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)*, pages 177–216, 2010.

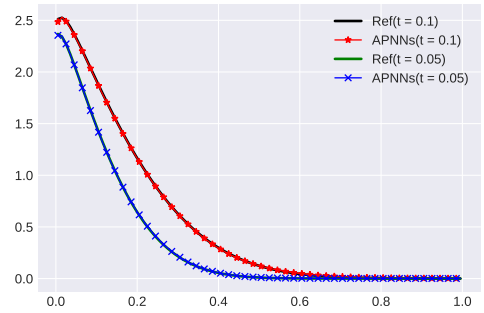

Figure 12: Problems IV. Plot of density ρ at $t = 0.05, 0.1$: APNNs vs. Ref. $\varepsilon = 5 \quad 10^{-2}$ and neural networks are $[2, 128, 128, 128, 128, 1]$ for ρ and $[3, 256, 256, 256, 256, 1]$ for g. Batch size is 1000 in domain, $400 \quad 2$ with penalty $\lambda_1 = 10$ for boundary condition and 1000 with penalty $\lambda_2 = 10$ for initial condition, the number of quadrature points is 30. Relative ℓ^2 error of APNNs is $4.80 \quad 10^{-3}$.

[17] Shi Jin, Jian-Guo Liu, and Zheng Ma. 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:Paper No. 15, 25, 2017.

[18] Shi Jin and Lorenzo Pareschi. *Uncertainty quanti cation for hyperbolic and kinetic equations*, volume 14. Springer, 2018.

[19] Shi Jin, Lorenzo Pareschi, and Giuseppe Toscani. Uniformly accurate diffusive relaxation schemes for multiscale transport equations. *SIAM Journal on Numerical Analysis*, 38(3):913–936, 2000.

[20] Shi Jin, Dongbin Xiu, and Xueyu Zhu. Asymptotic-preserving methods for hyperbolic and transport equations with random inputs and diffusive scalings. *Journal of Computational Physics*, 289:35–52, 2015.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[22] Mohammed Lemou and Luc Mieussens. A new asymptotic preserving scheme based on micro-macro formulation for linear kinetic equations in the diffusion limit. *SIAM Journal on Scienti c Computing*, 31(1):334–368, 2008.

[23] Long Li and Chang Yang. Asymptotic preserving scheme for anisotropic elliptic equations with deep neural network. *arXiv preprint arXiv:2104.05337*, 2021.

[24] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

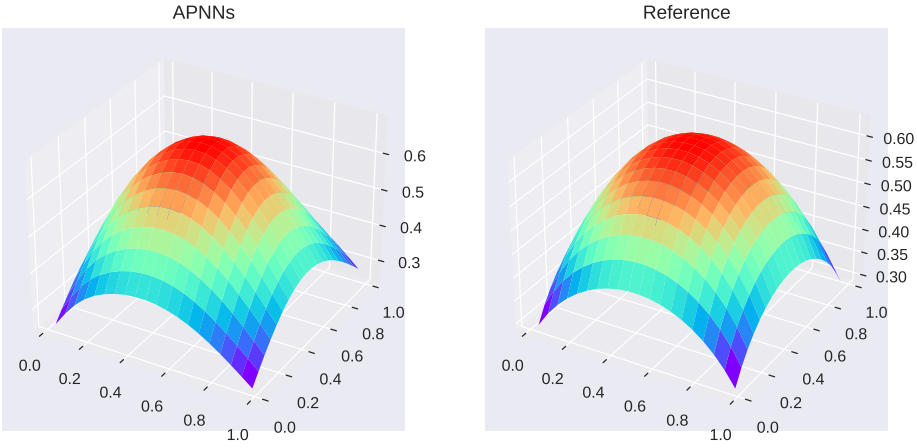

Figure 13: Problems V. Plot of density ρ at $t = 1.0$: APNNs vs. Ref. $\varepsilon = 1$ and neural networks are $[3, 64, 128, 256, 1]$ for ρ and $[5, 64, 128, 256, 512, 1]$ for g. Batch size is 800 in domain, 400 2 with penalty $\lambda_1 = 1$ for boundary condition and 400 with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is 50. Relative ℓ^2 error of APNNs is 5.78 10^{2}.

[25] Yulei Liao and Pingbing Ming. Deep Nitsche method: Deep Ritz method with essential boundary conditions. *arXiv preprint arXiv:1912.01309*, 2019.

[26] Jian-Guo Liu and Luc Mieussens. Analysis of an asymptotic preserving scheme for linear kinetic equations in the diffusion limit. *SIAM J. Numer. Anal.*, 48(4):1474–1491, 2010.

[27] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[28] Liyao Lyu, Zhen Zhang, Minxin Chen, and Jingrun Chen. MIM: A deep mixed residual method for solving high-order partial differential equations. *arXiv preprint arXiv:2006.04146*, 2020.

[29] Lorenzo Pareschi and Russel E Caflisch. An implicit monte carlo method for rarefied gas dynamics: I. the space homogeneous case. *Journal of Computational Physics*, 154(1):90–116, 1999.

[30] Lorenzo Pareschi and Giovanni Russo. Time relaxed monte carlo methods for the boltzmann equation. *SIAM Journal on Scienti c Computing*, 23(4):1253–1273, 2001.

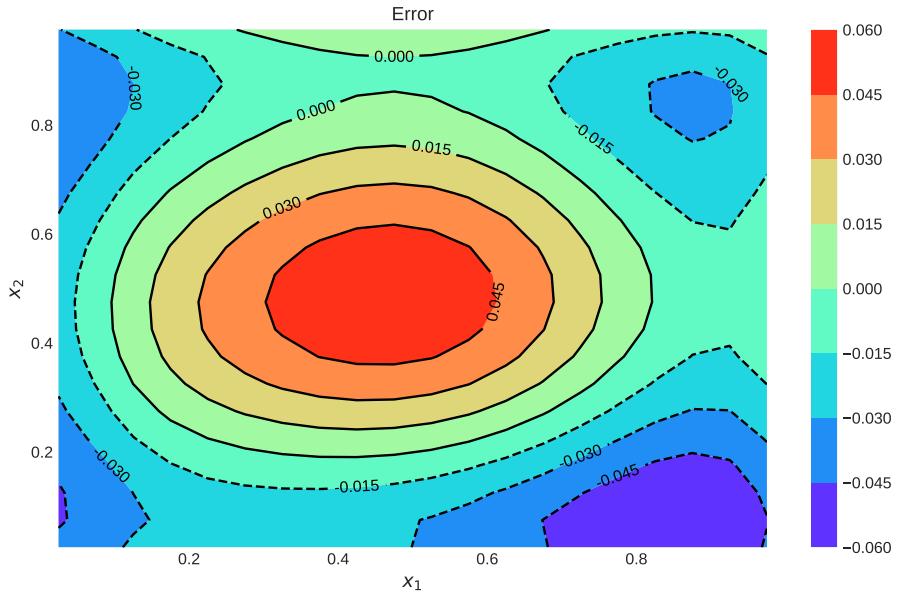

Figure 14: Problems VI. Plot of the error of density ρ at $t = 1.0$ between APNNs and Reference.

[31] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[32] Leonid Ryzhik, George Papanicolaou, and Joseph B Keller. Transport equations for elastic and other waves in random media. *Wave motion*, 24(4):327–370, 1996.

[33] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[34] E Weinan. *Principles of multiscale modeling*. Cambridge University Press, 2011.

[35] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.

[36] Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, page 109409, 2020.

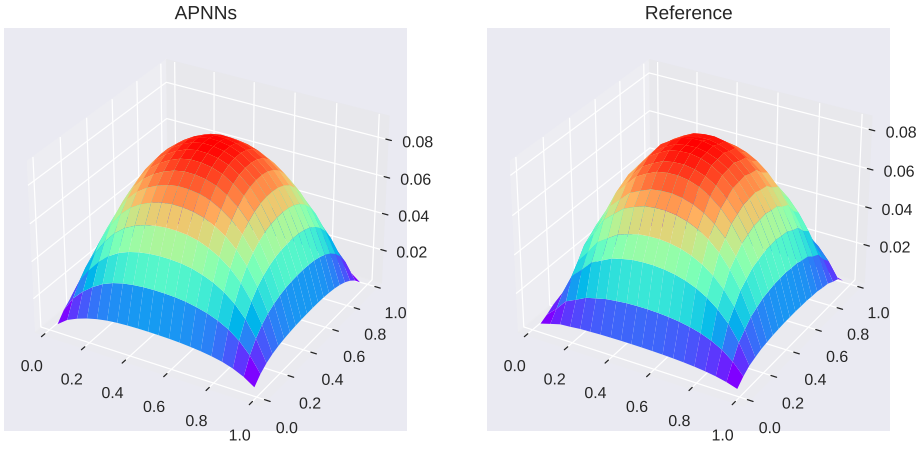

Figure 15: Problems VI. Plot of density ρ at $t = 0.1$: APNNs vs. Ref. $\varepsilon = 10^{-8}$ and neural networks are $[3, 64, 128, 256, 1]$ for ρ and $[5, 64, 128, 256, 512, 1]$ for g. Batch size is 800 in domain, 400×2 with penalty $\lambda_1 = 1$ for boundary condition and 400 with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is 50. Relative ℓ^2 error of APNNs is 1.29×10^{-1}.

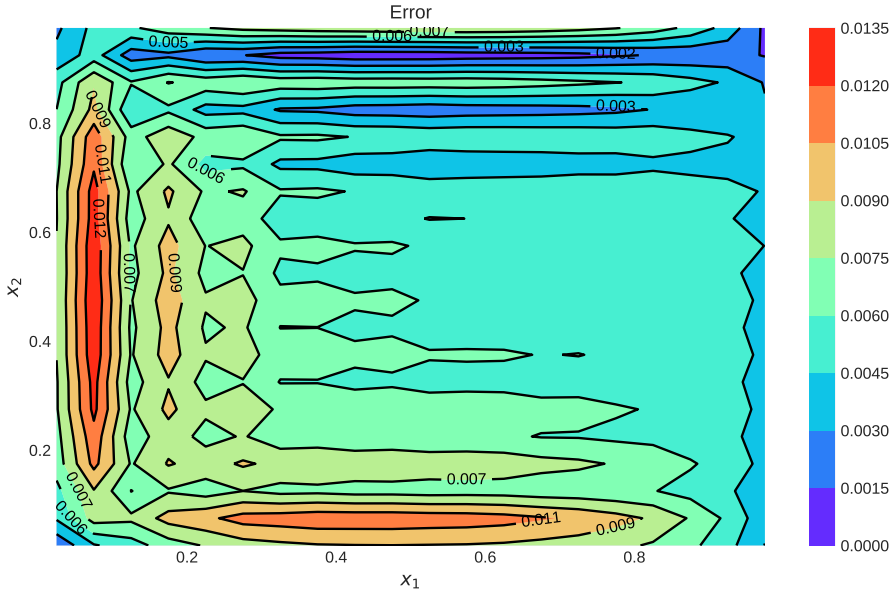

Figure 16: Problems VI. Plot of the error of density ρ at $t = 0.1$ between APNNs and Reference.

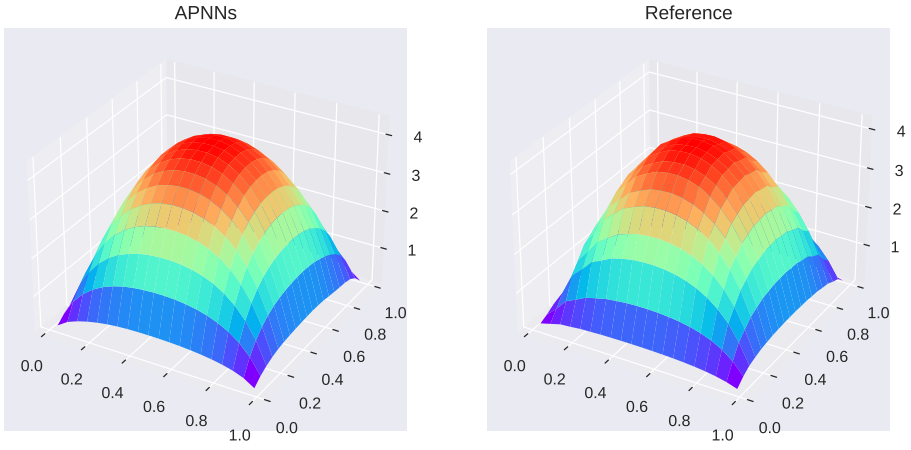

Figure 17: Problems VII. Plot of density ρ at $t = 0.1$: APNNs vs. Ref. $\varepsilon = 10^{-8}$ and neural networks are $[3, 64, 128, 256, 512, 1]$ for ρ and $[5, 64, 128, 256, 512, 1024, 1]$ for g. Batch size is 800 in domain, 400 $\times$ 2 with penalty $\lambda_1 = 1$ for boundary condition and 400 with penalty $\lambda_2 = 1$ for initial condition, the number of quadrature points is 50.

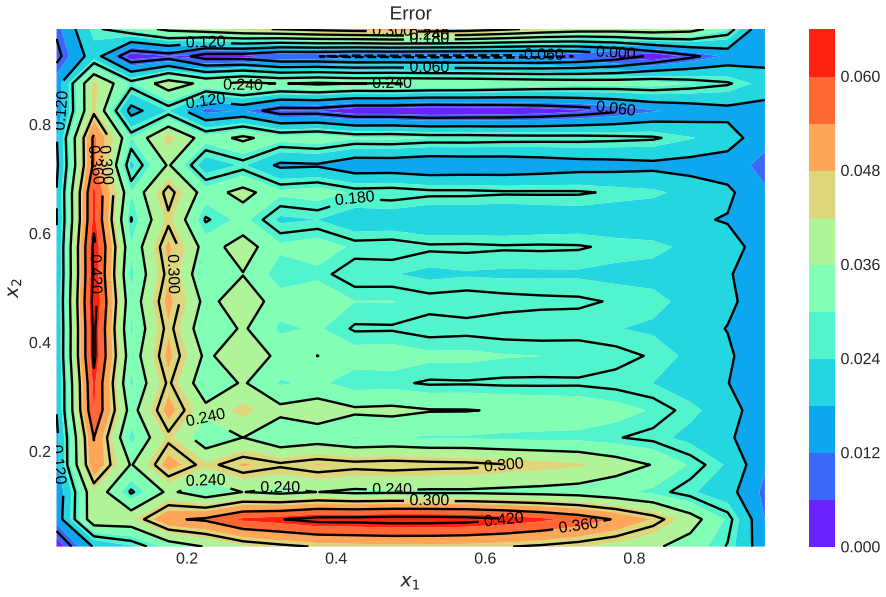

Figure 18: Problems VII. Plot of the error of density ρ at $t = 0.1$ between APNNs and Reference. The relative ℓ^2 error of APNNs is 4.35 10 ^{2}.

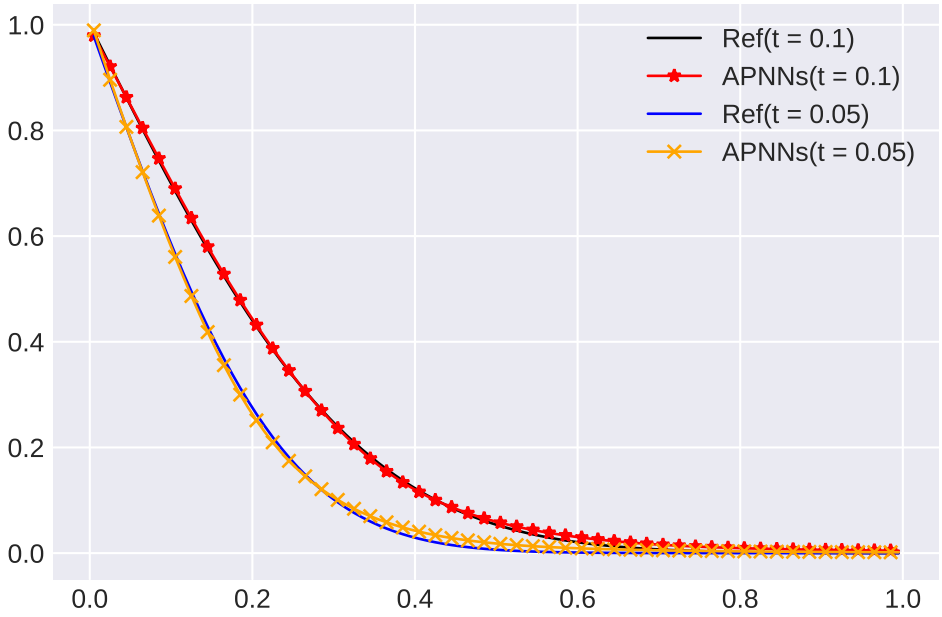

Figure 19: Problems VIII. Plot of density ρ by taking expectation for $\boldsymbol{z}$ at $t = 0.05, 0.1$: APNNs vs. Ref. $\varepsilon = 10^{-8}, \sigma_S(\boldsymbol{z}) = 1 + 0.3\prod_{i=1}^{10}\sin(\pi z_i)$ and neural networks are $[12, 64, 128, 256, 512, 1]$ for ρ and $[13, 64, 128, 256, 512, 1024, 1]$ for g. Batch size is 1000 in domain, 400×2 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 ℓ^2 error of APNNs is 2.47×10^{-2}.

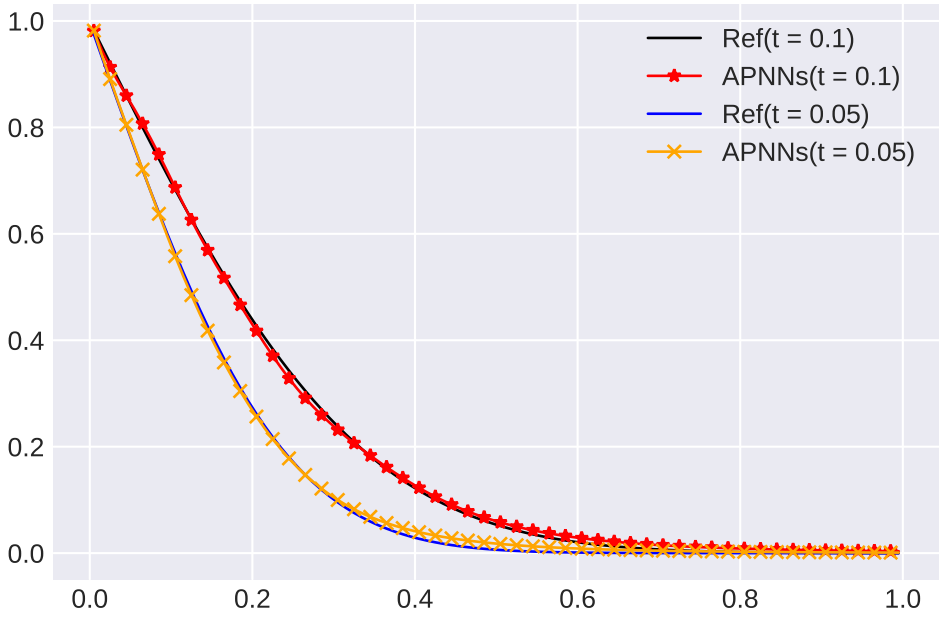

Figure 20: Problems IX. Plot of density ρ by taking expectation for $\boldsymbol{z}$ at $t = 0.05, 0.1$: APNNs vs. Ref. $\varepsilon = 10^{-3}, \sigma_S(\boldsymbol{z}) = 1 + 0.3\prod_{i=1}^{10}\sin(\pi z_i)$ and neural networks are $[12, 64, 128, 256, 512, 1]$ for ρ and $[13, 64, 128, 256, 512, 1024, 1]$ for g. Batch size is 1000 in domain, 400^2 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 ℓ^2 error of APNNs is 2.75×10^{-2}.

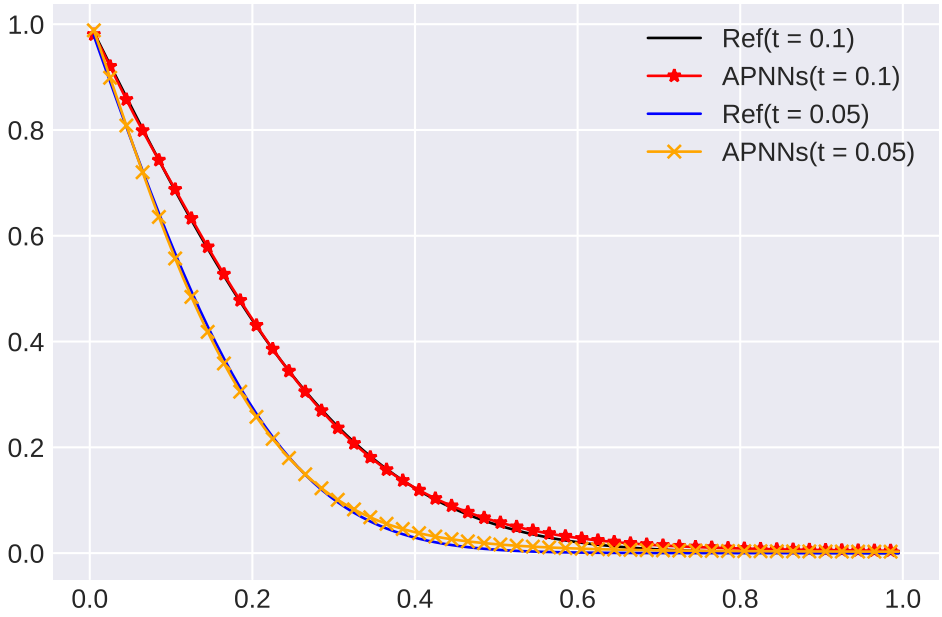

Figure 21: Problems X. Plot of density ρ by taking expectation for z at $t = 0.05, 0.1$: APNNs vs. Ref. $\varepsilon = 10^{-8}, \sigma_S(z) = 1 + 0.3\exp\left(-|z|^2/2\right)$ and neural networks are $[22, 64, 128, 256, 512, 1]$ for ρ and $[23, 64, 128, 256, 512, 1024, 1]$ for g. Batch size is 1000 in domain, 400×2 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 ℓ^2 error of APNNs is 1.51×10^{-2}.