

# Solve Kinetic Equations with Deep Learning

Zheng Ma

Shanghai Jiao Tong University

Joint work with Shi Jin, Keke Wu, Tianai Zhang, Han Wang, Jingrun Chen

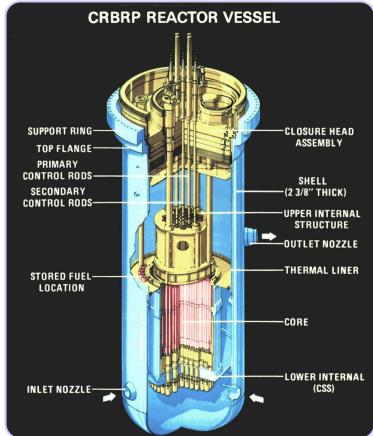


July 10, 2025

# Introduction

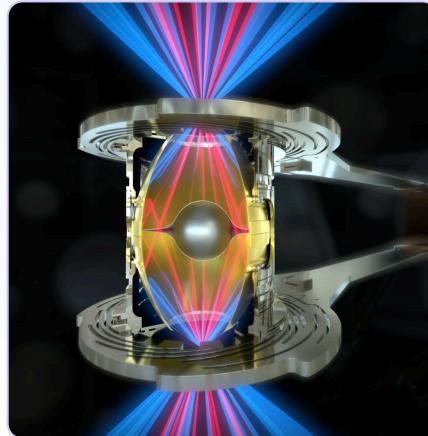
Kinetic equations are important in many areas

Neutron transport



Fission reactor

Radiative transfer



ICF

Rarefied gas



Reentry

Key problem: numerical simulation of **kinetic equations**

# Multiscale Kinetic Equations

$$\partial_t f + v \cdot \nabla_x f = Q(f)$$

- $f(t, x, v)$ : distribution function at time  $t$  in phase space  $(x, v)$
- $Q(f)$ : collision operator
- $\varepsilon$ : Knudsen number (ratio between mean free path and characteristic length)

Neutron transport equation

$$\varepsilon \partial_t f + v \cdot \nabla_x f + \Sigma_t f = \frac{1}{\varepsilon} \int \Sigma_s(v, v') f \, dv' + q$$

BGK equation

$$\partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} (M_{\text{eq}}[f] - f)$$

Multiscale problem:  $\varepsilon$  can vary from  $O(1)$  **kinetic regime** to 0 **hydrodynamic regime**

# Numerical Chanllenges

Curse of dimensionality

## Dimension

- Phase space + time:  $6 + 1 = 7$
- Collision operator is a 5-fold integral
- Need to evaluate collision at every phase point

## Collision

- Hard to maintain conservation at discrete level
- Highly nonlinear for Boltzmann collision
- Ray effect

## Multiscale

- Stability issues for small  $\varepsilon$  (stiffness)
- Consistency of the scheme with limiting model as  $\varepsilon \rightarrow 0$
- Automatically capture the transition across regimes

# Conventional Approaches

## Probabilistic approaches

- Monte Carlo methods for linear transport
  - MCNP
  - COG (LLNL, criticality safety analysis, general radiation)
  - Mercury
- Direct simulation Monte Carlo (DSMC) methods
  - Bird, Nanbu, ...
  - Sparta (ORNL, rarefied gas dynamics)

### 👍 Pros

- ✓ Easy implementation
- ✓ Relatively efficient

### 👎 Cons

- ⚠ Only half-order accuracy
- ⚠ Converge slow
- ⚠ Random fluctuations

# Conventional Approaches

Deterministic approaches

- Discrete velocity/ordinate methods (DVM)
  - Ardra (LLNL)
  - NEWT (ORNL)
  - DORT
  - Kit-RT
- Spectral methods

## 👍 Pros

- ✓ Spectral accuracy
- ✓ Relatively expensive

## 👎 Cons

- ⚠ Do not main conservation

# **Deep learning methods may become new approach**

Overcome the curse of dimensionality

# Solve PDEs with Deep Learning

Key components

## Constraints as the loss of minimization problem

- Model: PDE / physical information needed (e.g., PINNs, DeepRitz, DeepGalerkin)
- Data: pure supervised or as a priori information
- IC (initial conditions) and BC (boundary conditions)
- Other constraints: **conservation**, symmetry, etc.

## Architecture build a deep neural network (function class) as the trial function

- Approximate solution: PINNs, DeepRitz, DeepGalerkin, etc.
- Approximate solution operator: DeepONet, FNO, etc.
- Mapping from equations (as a computational graph) to solutions (PDEFormer)

## Optimization

- Minimize loss over the parameter space, usually SGD, Adam, LBFGS, etc.

# Constraints for multiscale kinetic equations

PINNs, DeepRitz, etc.

# Linear Transport Equation

1-D for illustration

$$\varepsilon \partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} \left( \frac{1}{2} \int_{-1}^1 f \, dv' - f \right)$$

PINN for example:

- **Architecture:** approximate the **solution** by a deep neural network

$$f_\theta^{\text{NN}}(t, x, v) \approx f(t, x, v)$$

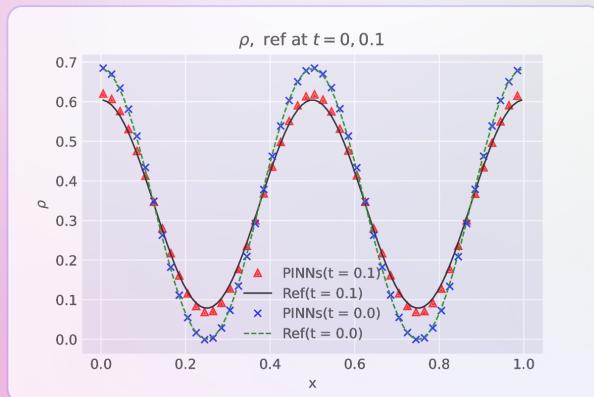
- **Model:** take the **least square** of the linear transport equation residual (strong form) as loss

$$\mathcal{R}_{\text{PINN}}^\varepsilon = \int \left( \varepsilon^2 \partial_t f_\theta^{\text{NN}} + \varepsilon v \cdot \nabla_x f_\theta^{\text{NN}} - \left( \frac{1}{2} \int_{-1}^1 f_\theta^{\text{NN}} \, dv' - f_\theta^{\text{NN}} \right) \right)^2 \, dv \, dx \, dt$$

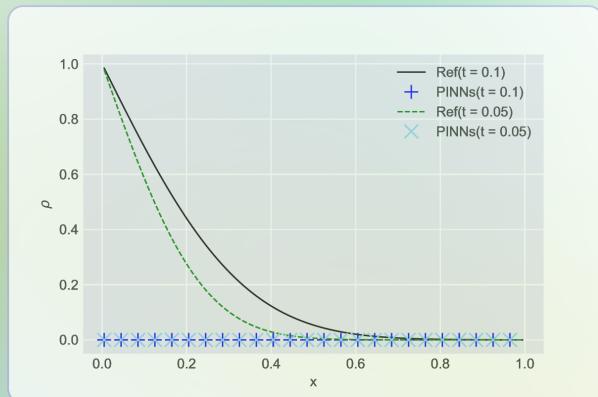
# Result of PINN

$$\varepsilon \partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} (\rho(t, x) - f), \quad \rho(t, x) = \frac{1}{2} \int_{-1}^1 f \, dv'$$

Kinetic regime:  $\varepsilon = 1$



Diffusive regime:  $\varepsilon = 10^{-8}$



$$f_0(x, v) = [1 + \cos(4\pi x)] e^{-\frac{v^2}{2}} / \sqrt{2\pi}$$

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

# Failure of PINN Loss

PINN can not resolve small scale

$$\mathcal{R}_{\text{PINN}}^{\varepsilon} = \int \left( \varepsilon^2 \partial_t f_{\theta}^{\text{NN}} + \varepsilon v \cdot \nabla_x f_{\theta}^{\text{NN}} - \left( \frac{1}{2} \int_{-1}^1 f_{\theta}^{\text{NN}} dv' - f_{\theta}^{\text{NN}} \right) \right)^2 dv dx dt$$

Let  $\varepsilon \rightarrow 0$

$$\mathcal{R}_{\text{PINN}}^0 = \int \left( \frac{1}{2} \int_{-1}^1 f_{\theta}^{\text{NN}} dv' - f_{\theta}^{\text{NN}} \right)^2 dv dx dt$$

which can be viewed as the **PINN loss of the steady equation**

$$f_{\theta}^{\text{NN}} = \frac{1}{2} \int_{-1}^1 f_{\theta}^{\text{NN}} dv'$$

PINN loss is not AP (Asymptotic-preserving), i.e., it does not converge to the **correct macroscopic limit** as  $\varepsilon \rightarrow 0$ .

# Diffusion Limit

$$\varepsilon \partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} (\rho(t, x) - f), \quad \rho(t, x) = \langle f \rangle := \frac{1}{2} \int_{-1}^1 f dv'.$$

Decompose  $f$  into the equilibrium  $\rho(t, x)$  and the non-equilibrium part  $g(t, x, v)$ :

$$f(t, x, v) = \rho(t, x) + \varepsilon g(t, x, v),$$

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

Subsititing  $f = \rho + \varepsilon g$  into the linear transport equation yields

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

where  $I$  is the identity operator and  $\Pi(\cdot)(v) := \langle \cdot \rangle$  is the projection operator.

# Micro-macro System

Classical numerical schemes based on this system are usually AP schemes

The micro-macro system for the linear trasport equation

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

Sending  $\varepsilon \rightarrow 0$ , the above system formally approaches

$$\begin{cases} \partial_t \rho + \langle v \cdot \nabla_x g \rangle = 0, \\ -v \cdot \nabla_x \rho = g. \end{cases}$$

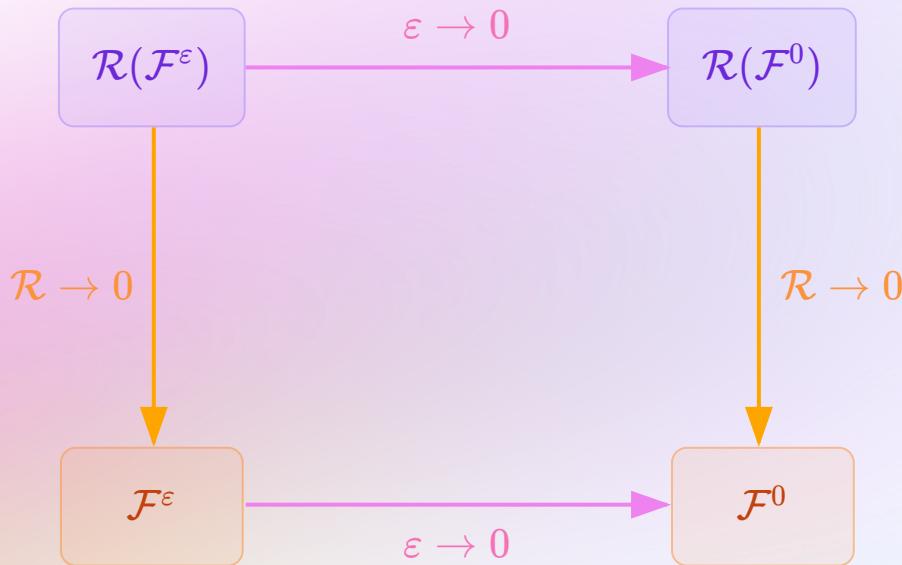
Plugging the second equation into the first equation gives the diffusion equation

$$\rho_t - \frac{1}{3} \rho_{xx} = 0.$$

# What is "good" loss for multiscale kinetic equations?

Conservation, symmetry, parity, etc.

# Asymptotic-Preserving Neural Networks



$\mathcal{F}^\varepsilon, \mathcal{F}^0$

Microscopic and macroscopic  
models

$\mathcal{R}(\mathcal{F}^\varepsilon), \mathcal{R}(\mathcal{F}^0)$

Loss of microscopic  
and macroscopic models

# Micro-macro System

Classical numerical schemes based on this system are usually AP schemes

The micro-macro system for the linear trasport equation

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

Sending  $\varepsilon \rightarrow 0$ , the above system formally approaches

$$\begin{cases} \partial_t \rho + \langle v \cdot \nabla_x g \rangle = 0, \\ -v \cdot \nabla_x \rho = g. \end{cases}$$

Plugging the second equation into the first equation gives the diffusion equation

$$\rho_t - \frac{1}{3} \rho_{xx} = 0.$$

# APNN v1: based on Micro-macro decomposition

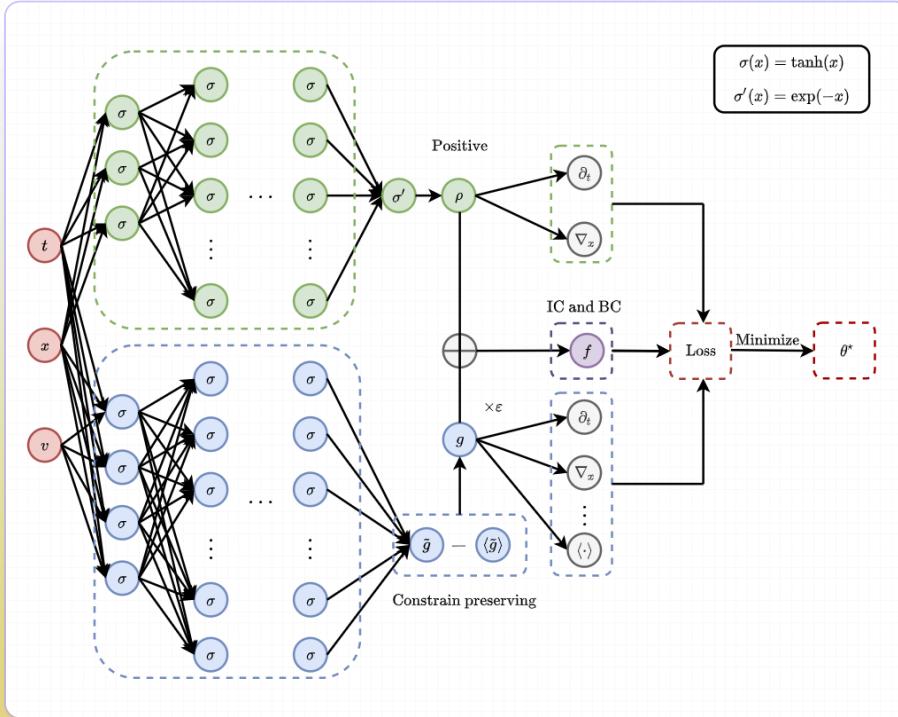
Re-design of loss

$$\begin{aligned}\mathcal{R}_{\text{APNN}}^\varepsilon &= \int (\partial_t \rho_\theta^{\text{NN}} + \nabla_x \cdot \langle v g_\theta^{\text{NN}} \rangle)^2 dx dt \\ &\quad + \int |\varepsilon^2 \partial_t g_\theta^{\text{NN}} + \varepsilon(I - \Pi)(v \cdot \nabla_x g_\theta^{\text{NN}}) + v \cdot \nabla_x \rho_\theta^{\text{NN}} + g_\theta^{\text{NN}}|^2 dv dx dt\end{aligned}$$

Let  $\varepsilon \rightarrow 0$ , the loss converges to

$$\mathcal{R}_{\text{APNN}}^0 = \int (\partial_t \rho_\theta^{\text{NN}} + \nabla_x \cdot \langle v g_\theta^{\text{NN}} \rangle)^2 dx dt + \int (v \cdot \nabla_x \rho_\theta^{\text{NN}} + g_\theta^{\text{NN}})^2 dv dx dt.$$

# APNN v1: based on Micro-macro decomposition

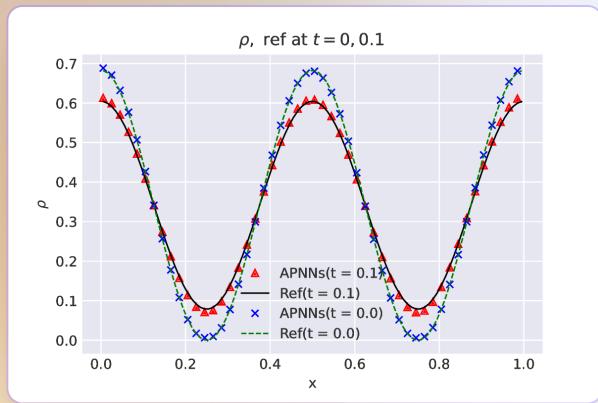


Mass conservation mechanism  $g_{\theta}^{\text{NN}} = \tilde{g}_{\theta}^{\text{NN}} - \langle \tilde{g}_{\theta}^{\text{NN}} \rangle$  is also important!

# Test examples

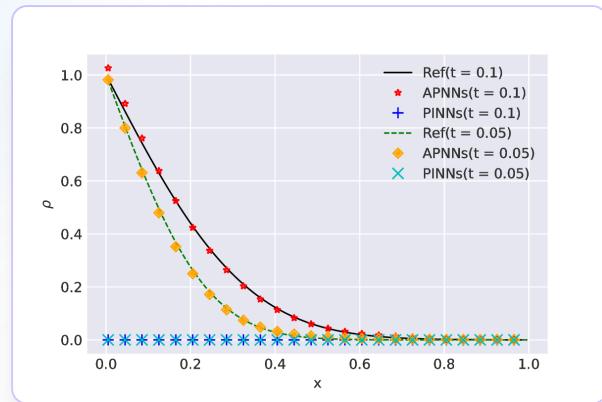
Periodic boundary condition:  $\varepsilon = 1$

$$f(t, x_L, v) = f(t, x_R, v),$$
$$f_0(x, v) = \frac{1 + \cos(4\pi x)}{\sqrt{2\pi}} e^{-\frac{v^2}{2}}.$$



Inflow boundary condition  $\varepsilon = 10^{-8}$

$$f(t, x_L, v) = 1 \text{ for } v > 0 \text{ and } 1 \text{ for } v < 0,$$
$$f_0(x, v) = 0.$$



One can observe that APNN works for both  $\varepsilon = 1$  and  $\varepsilon = 10^{-8}$ .

# Mass conservation mechanism

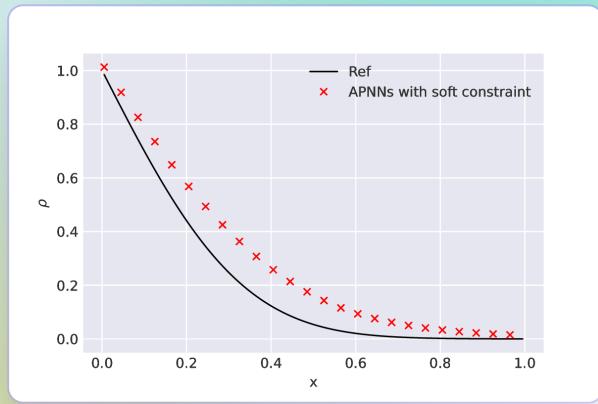
Ex 3: Inflow boundary condition ( $\varepsilon = 10^{-8}$ )

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  $\lambda_3$ , we use  $\hat{g}_\theta^{\text{NN}}$  and modifies the loss as

$$\mathcal{R}_{\text{APNN}} + \lambda_3 \int |\langle \hat{g}_\theta^{\text{NN}} \rangle - 0|^2 dx dt.$$

Plot of density  $\rho$  at  $t = 0.1$ : APNNs with soft constraint (marker) vs. Ref (line).



Mass conservation mechanism  $g_\theta^{\text{NN}} = \tilde{g}_\theta^{\text{NN}} - \langle \tilde{g}_\theta^{\text{NN}} \rangle$  is important!

# Even and odd parity method

Alternative method for constructing AP schemes in classical numerical methods

$$\varepsilon \partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} \left( \frac{1}{2} \int_{-1}^1 f dv' - f \right), \quad -1 \leq v \leq 1$$

By splitting equation and define even- and odd-parities as

$$r(t, x, v) = \frac{1}{2}[f(t, x, v) + f(t, x, -v)], \quad 0 \leq v \leq 1,$$

$$j(t, x, v) = \frac{1}{2\varepsilon}[f(t, x, v) - f(t, x, -v)], \quad 0 \leq v \leq 1,$$

one can obtain

$$\begin{cases} \partial_t r + v \partial_x j = \frac{1}{\varepsilon^2}(\rho - r), \\ \partial_t j + \frac{1}{\varepsilon^2} v \partial_x r = -\frac{1}{\varepsilon^2} j, \end{cases}$$

where  $\rho = \langle r \rangle := \int_0^1 r(t, x, v) dv$ .

# Even and odd parity method

Not all traditional AP schemes work for neural networks!

$$\begin{cases} \partial_t r + v \partial_x j = \frac{1}{\varepsilon^2} (\rho - r), \\ \partial_t j + \frac{1}{\varepsilon^2} v \partial_x r = -\frac{1}{\varepsilon^2} j. \end{cases}$$

So far, we've got the even-odd system, however, it is not AP when applying neural networks for  $r$  and  $j$ .

To make it AP, we next introduce  $\rho$  into this system as a bridge between  $r$  and  $j$ .

By integrating over  $v$ , the first equation gives

$$\partial_t \langle r \rangle + \int_0^1 v \partial_x j dv = \frac{1}{\varepsilon^2} (\rho - \langle r \rangle),$$

and due to  $\rho = \langle r \rangle$ , one can write as follows

$$\partial_t \rho + \langle v \partial_x j \rangle = 0.$$

# Even and odd parity method

The even-odd parity system for the linear transport equation

$$\begin{cases} \partial_t r + v \partial_x j = \frac{1}{\varepsilon^2} (\rho - r), \\ \partial_t j + \frac{1}{\varepsilon^2} v \partial_x r = -\frac{1}{\varepsilon^2} j, \\ \partial_t \rho + \langle v \partial_x j \rangle = 0. \end{cases}$$

Sending  $\varepsilon \rightarrow 0$ , the above system formally approaches

$$\begin{cases} \rho = r, \\ v \partial_x r = -j, \\ \partial_t \rho + \langle v \partial_x j \rangle = 0. \end{cases}$$

Plugging the first two equations into the third equation gives the diffusion equation  $\rho_t - \frac{1}{3} \rho_{xx} = 0$ .

# APNN v2: based on even and odd parity

More general approach

The equation of local conservation law  $\partial_t \rho + \langle v \partial_x j \rangle = 0$  is important in constructing the APNN loss. By coupling these equations of  $r$ ,  $j$  and  $\rho$ , one can obtain the loss for the diffusion limit equation.

For solving the linear transport equation by deep neural networks, we need to use DNN to parametrize three functions  $\rho(t, x)$ ,  $r(t, x, v)$  and  $j(t, x, v)$ .

So here three networks are used:

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

$$r_\theta^{\text{NN}}(t, x, v) := \exp\left(-\frac{1}{2}(\tilde{r}_\theta^{\text{NN}}(t, x, v) + \tilde{r}_\theta^{\text{NN}}(t, x, -v))\right) \approx r(t, x, v),$$

$$j_\theta^{\text{NN}}(t, x, v) := \tilde{j}_\theta^{\text{NN}}(t, x, v) - \tilde{j}_\theta^{\text{NN}}(t, x, -v) \approx j(t, x, v).$$

# APNN v2: based on even- and odd- parity

$$\begin{aligned}\mathcal{R}_{\text{APNN}}^\varepsilon &= \lambda_1 \int |\varepsilon^2 \partial_t r_\theta^{\text{NN}} + \varepsilon^2 v \partial_x j_\theta^{\text{NN}} - (\rho_\theta^{\text{NN}} - r_\theta^{\text{NN}})|^2 dv dx dt \\ &\quad + \lambda_2 \int |\varepsilon^2 \partial_t j_\theta^{\text{NN}} + v \partial_x r_\theta^{\text{NN}} - (-j_\theta^{\text{NN}})|^2 dv dx dt \\ &\quad + \lambda_3 \int |\partial_t \rho_\theta^{\text{NN}} + \langle v \partial_x j_\theta^{\text{NN}} \rangle|^2 dx dt \\ &\quad + \lambda_4 \int |\rho_\theta^{\text{NN}} - \langle r_\theta^{\text{NN}} \rangle|^2 dx dt\end{aligned}$$

Notice that the constraint  $\rho = \langle r \rangle$  is also added into the APNN loss.

# Test examples - Case I

Inflow boundary condition ( $\varepsilon = 10^{-3}$ )

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

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

$$\rho_0(x) = 0,$$

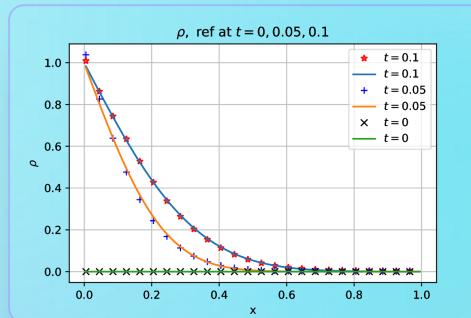
$$f_0(x, v) = 0.$$

Note that the function  $f$  has a jump at  $t = 0$  since  $F_L(v) = 1, F_R(v) = 0$  but  $f_0(x, v) = 0$ .

For better numerical performance,  $\rho_\theta^{\text{NN}}$  can be further constructed to automatically satisfies initial condition:

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

Plot of density  $\rho$  at  $t = 0, 0.05, 0.1$ : APNNs (marker) vs. Ref (line).

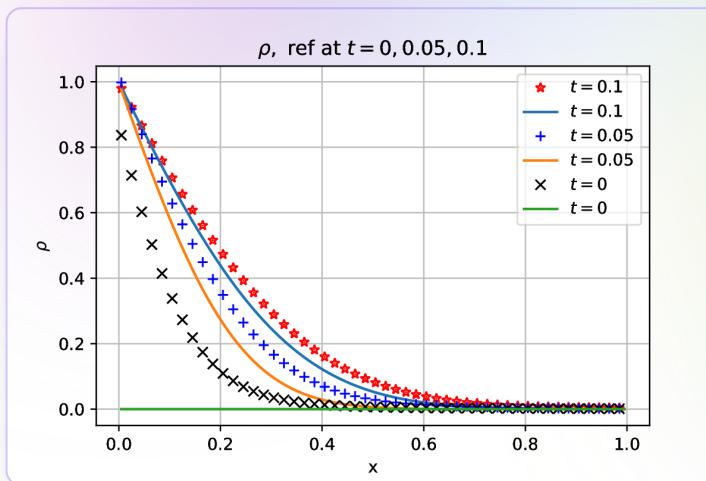


Relative  $\ell^2$  error of APNNs is  $9.87 \times 10^{-3}$ .

The numerical performance of enforcement of initial condition and the soft constraint  $\rho = \langle r \rangle$  are discussed as follows.

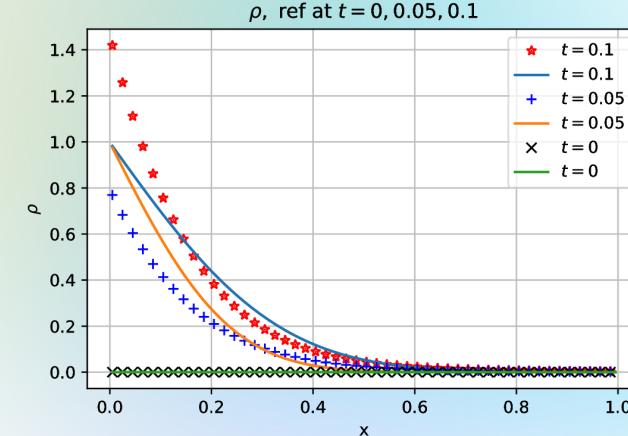
Plot of density  $\rho$  at  $t = 0, 0.05, 0.1$ : APNNs (marker) vs. Ref (line).

Figure 1: without the enforcement of initial condition



Due to the poor approximate of initial and boundary layer effect, it gives wrong solution at time  $t = 0, 0.05, 0.1$ .

Figure 2: without the constraint  $\rho = \langle r \rangle$



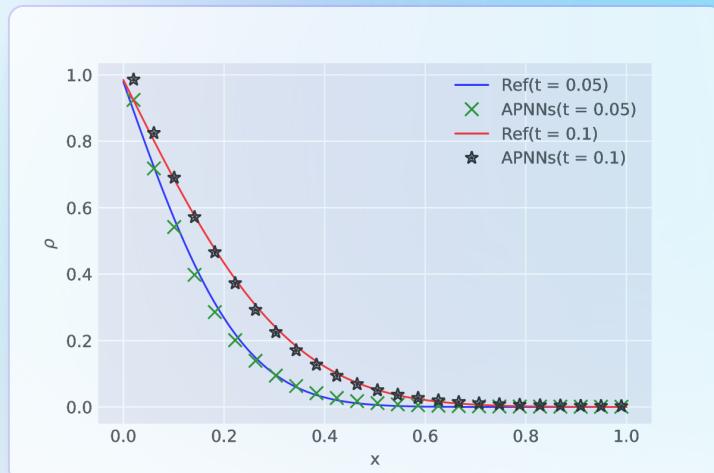
The solutions are also wrong at time  $t = 0, 0.05, 0.1$ , therefore, we consider this constraint into our APNN loss.

# Test examples - Case II

UQ problems with inflow condition ( $\varepsilon = 10^{-5}$ )

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

$$\sigma_S(\mathbf{z}) = 1 + \frac{1}{10} \prod_{i=1}^{20} \sin(\pi z^i), \quad \mathbf{z} = (z^1, z^2, \dots, z^{20}) \sim \mathcal{U}([-1, 1]^{20}),$$



# Bhatnagar-Gross-Krook (BGK) equation

$$\partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} (M(U) - f), \quad v \in \mathbb{R},$$

where  $M(U)$  denotes the local Maxwellian distribution function given by

$$M(U) = \frac{\rho}{(2\pi T)^{\frac{1}{2}}} \exp\left(-\frac{|v - u|^2}{2T}\right),$$

and  $\rho(t, x)$ ,  $u(t, x)$  and  $T(t, x)$  are the density, macroscopic velocity and temperature which are related with the moments of  $f$ :

$$U := \begin{pmatrix} \rho \\ \rho u \\ \frac{1}{2}\rho|u|^2 + \frac{1}{2}\rho T \end{pmatrix} = \int_{\mathbb{R}} m f dv, \quad m = \left(1, v, \frac{1}{2}v^2\right)^T.$$

Notice that the Boltzmann-BGK equation is an integro-differential equation with its nonlinear and non-local collision operator.

# Density, macroscopic velocity and temperature

- density

$$\rho = \int f(t, x, v) dv.$$

- velocity

$$u = \int v \cdot \frac{f(t, x, v)}{\int f(t, x, v) dv} dv = \frac{1}{\rho} \int v f dv.$$

- thermodynamics energy

$$\int \frac{1}{2} (v - u)^2 \cdot \frac{f(t, x, v)}{\int f(t, x, v) dv} dv = \frac{1}{\rho} \left[ \frac{1}{2} \int v^2 f dv - \frac{1}{2} \rho u^2 \right],$$

$$\Rightarrow \frac{1}{2} \rho T = \frac{1}{2} \int_{\mathbb{R}} v^2 f dv - \frac{1}{2} \rho |u|^2 \text{ (The ideal gas law).}$$

# Local conservation laws

$$\partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} (M(U) - f), \quad v \in \mathbb{R},$$

$$U := \begin{pmatrix} \rho \\ \rho u \\ \frac{1}{2}\rho|u|^2 + \frac{1}{2}\rho T \end{pmatrix} = \int_{\mathbb{R}} m f dv, \quad m = \left(1, v, \frac{1}{2}v^2\right)^T.$$

Due to the properties of conserving mass, momentum and energy of collision operator, one can multiply the BGK equation by  $m(v)$  and integrate with respect to  $v$  to obtain the equations of local conservation laws

$$\partial_t \langle mf \rangle + \nabla_x \cdot \langle vmf \rangle = 0, \text{ where } \langle g \rangle = \int_{\mathbb{R}} g(v) dv,$$

i.e.,

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ \frac{1}{2}\rho|u|^2 + \frac{1}{2}\rho T \end{pmatrix} + \nabla_x \cdot \langle vmf \rangle = 0.$$

# APNN System

The systems of Boltzmann-BGK model

$$\begin{cases} \varepsilon (\partial_t f + v \partial_x f) = M(U) - f, \\ \partial_t \begin{pmatrix} \rho \\ \rho u \\ \frac{1}{2}\rho|u|^2 + \frac{1}{2}\rho T \end{pmatrix} + \nabla_x \cdot \langle vmf \rangle = 0, \\ \begin{pmatrix} \rho \\ \rho u \\ \frac{1}{2}\rho|u|^2 + \frac{1}{2}\rho T \end{pmatrix} = \int_{\mathbb{R}} mf dv. \end{cases}$$

Sending  $\varepsilon \rightarrow 0$ , one have  $f = M(U)$  and the local conservation laws becomes compressible Euler equation:

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ \frac{1}{2}\rho|u|^2 + \frac{1}{2}\rho T \end{pmatrix} + \nabla_x \cdot \langle vmM \rangle = 0.$$

# Boundary and initial conditions

The boundary conditions of  $\rho, u, T$  are set as constants:

$$\begin{aligned}\rho(t, x_L) &= \rho_L, \rho(t, x_R) = \rho_R, \\ u(t, x_L) &= u_L = 0, u(t, x_R) = u_R = 0, \\ T(t, x_L) &= T_L, T(t, x_R) = T_R.\end{aligned}$$

The initial condition of  $f$  is computed by the initial functions  $\rho_0(x), u_0(x), T_0(x)$ :

$$f(0, x, v) = \frac{\rho_0}{(2\pi T_0)^{\frac{1}{2}}} \exp\left(-\frac{|v - u_0|^2}{2T_0}\right) := f_0(x, v).$$

Here, time  $t \in \mathcal{T} := [0, T]$ , space point  $x \in \mathcal{D} := [x_L, x_R]$  and we restrict the range of velocity to a bounded symmetrical domain  $\Omega = [-V, V]$  with  $V = 10$  since this assumption might be realistic in many studies.

# APNN v2 for Boltzmann-BGK equation

First we parametrize four functions  $f(t, x, v)$ ,  $\rho(t, x)$ ,  $u(t, x)$  and  $T(t, x)$  with four networks. The time and velocity variable  $t, v$  are normalized into  $[0, 1]$  and  $[-1, 1]$  with scaling  $\bar{t} = t/T$ ,  $\bar{v} = v/V$  and we construct four DNNs as follows:

$$f_\theta^{\text{NN}}(t, x, v) := \ln \left( 1 + \exp(\tilde{f}_\theta^{\text{NN}}(\bar{t}, x, \bar{v})) \right) > 0,$$

$$\rho_\theta^{\text{NN}}(t, x) := \exp \left( (x - x_L)(x_R - x) \cdot \tilde{\rho}_\theta^{\text{NN}}(\bar{t}, x) + \log(\rho_L) \frac{x_R - x}{x_R - x_L} + \log(\rho_R) \frac{x - x_L}{x_R - x_L} \right) > 0,$$

$$u_\theta^{\text{NN}}(t, x) := \sqrt{(x - x_L)(x_R - x)} \cdot \tilde{u}_\theta^{\text{NN}}(\bar{t}, x),$$

$$T_\theta^{\text{NN}}(t, x) := \exp \left( (x - x_L)(x_R - x) \cdot \tilde{T}_\theta^{\text{NN}}(\bar{t}, x) + \log(T_L) \frac{x_R - x}{x_R - x_L} + \log(T_R) \frac{x - x_L}{x_R - x_L} \right) > 0,$$

which  $\rho_\theta^{\text{NN}}$ ,  $u_\theta^{\text{NN}}$ ,  $T_\theta^{\text{NN}}$  automatically satisfy the boundary conditions. In this problem, to keep  $f$  positive,  $\ln(1 + \exp(\cdot))$  is applied for constructing  $f_\theta^{\text{NN}}$ . The benefit of this construction is that the values of  $f_\theta^{\text{NN}}$  and  $\tilde{f}_\theta^{\text{NN}}$  are at the same level.

# APNN loss for Boltzmann-BGK

$$\begin{aligned}\mathcal{R}_{\text{APNN, BGK}}^\varepsilon = & \frac{\lambda_1}{N_1^{(1)}} \sum_{i=1}^{N_1^{(1)}} |\varepsilon(\partial_t f_\theta^{\text{NN}}(t_i, x_i, v_i) + v \nabla_x f_\theta^{\text{NN}}(t_i, x_i, v_i)) - (M(U_\theta^{\text{NN}}) - f_\theta^{\text{NN}})(t_i, x_i, v_i)|^2 \\ & + \frac{\lambda_2}{N_1^{(2)}} \sum_{i=1}^{N_1^{(2)}} |\partial_t \rho_\theta^{\text{NN}}(t_i, x_i) + \nabla_x \langle v f_\theta^{\text{NN}} \rangle(t_i, x_i)|^2 \\ & + \frac{\lambda_3}{N_1^{(3)}} \sum_{i=1}^{N_1^{(3)}} |\partial_t (\rho_\theta^{\text{NN}}(t_i, x_i) u_\theta^{\text{NN}}(t_i, x_i)) + \nabla_x \langle v^2 f_\theta^{\text{NN}} \rangle(t_i, x_i)|^2 \\ & + \frac{\lambda_4}{N_1^{(4)}} \sum_{i=1}^{N_1^{(4)}} |\partial_t \left( \frac{1}{2} \rho_\theta^{\text{NN}}(t_i, x_i) (u_\theta^{\text{NN}}(t_i, x_i))^2 + \frac{1}{2} \rho_\theta^{\text{NN}}(t_i, x_i) T_\theta^{\text{NN}}(t_i, x_i) \right) + \\ & \quad \nabla_x \left\langle \frac{1}{2} v^3 f_\theta^{\text{NN}} \right\rangle(t_i, x_i)|^2\end{aligned}$$

# Test example

Initial conditions ( $\varepsilon = 10^{-3}$ )

$$\rho_0(x) = 1.5 + (0.625 - 1.5) \cdot \frac{\sin(\pi x) + 1}{2},$$

$$u_0(x) = 0,$$

$$T_0(x) = 1.5 + (0.75 - 1.5) \cdot \frac{\sin(\pi x) + 1}{2},$$

$$f_0(x, v) = \frac{\rho_0}{(2\pi T_0)^{\frac{1}{2}}} \exp\left(-\frac{|v - u_0|^2}{2T_0}\right).$$

The reference solutions are the density, momentum and energy:

$$\rho, \rho u, \frac{1}{2} \rho u^2 + \frac{1}{2} \rho T.$$

Setting: ResNet with units

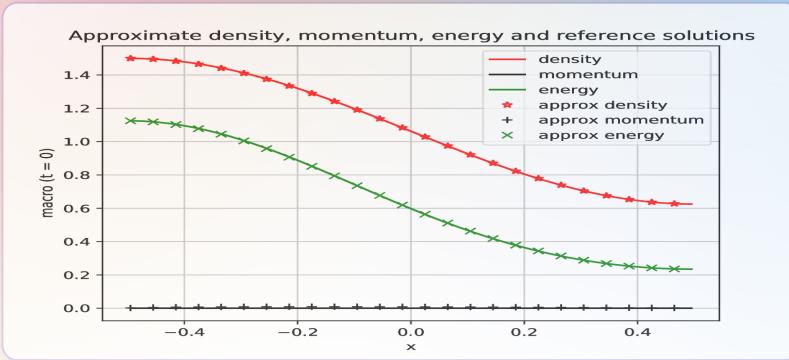
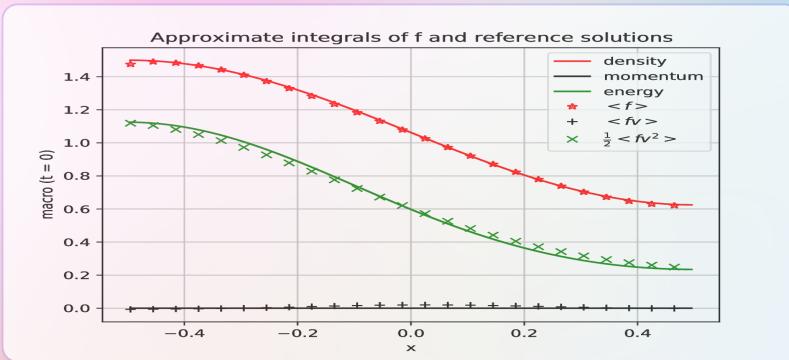
[3, 128, 128, 128, 128, 128, 128, 1] for  $f$  and [2, 64, 64, 64, 64, 64, 64, 1] both for  $\rho, u$  and  $T$ .  
Batch size is 512 in domain, and 256 on initial, the number of quadrature points is 64.

$\lambda_9 = 0.1, \lambda_{11} = \lambda_{13} = \lambda_{14} = 10$  and others are set to be 1.

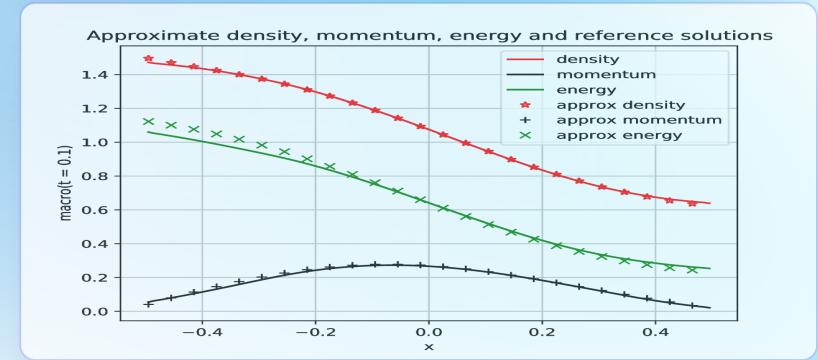
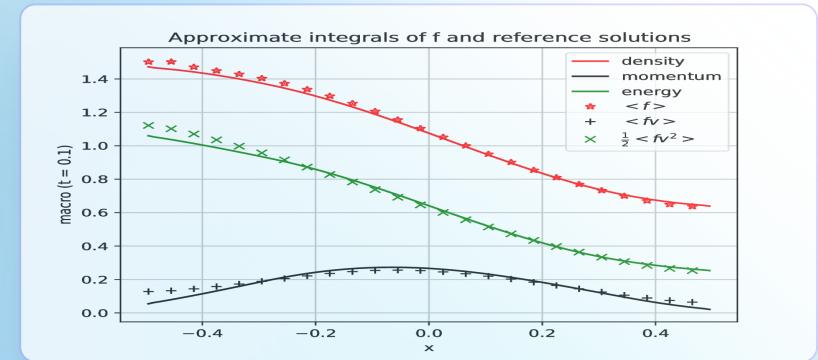
For  $t = 0$  : mean square error of density, momentum and energy are  
7.16e-8, 4.50e-6, 8.13e-7.

For  $t = 0.1$  : relative  $l^2$  error of density, momentum and energy are  
5.43e-3, 6.35e-3, 4.47e-2.

The integrals of approximate  $f$  and approximate density, momentum and energy at time  $t = 0$



The integrals of approximate  $f$  and approximate density, momentum and energy at time  $t = 0.1$



# APNN for More General Systems

VPFP, Gray-RTE, etc.

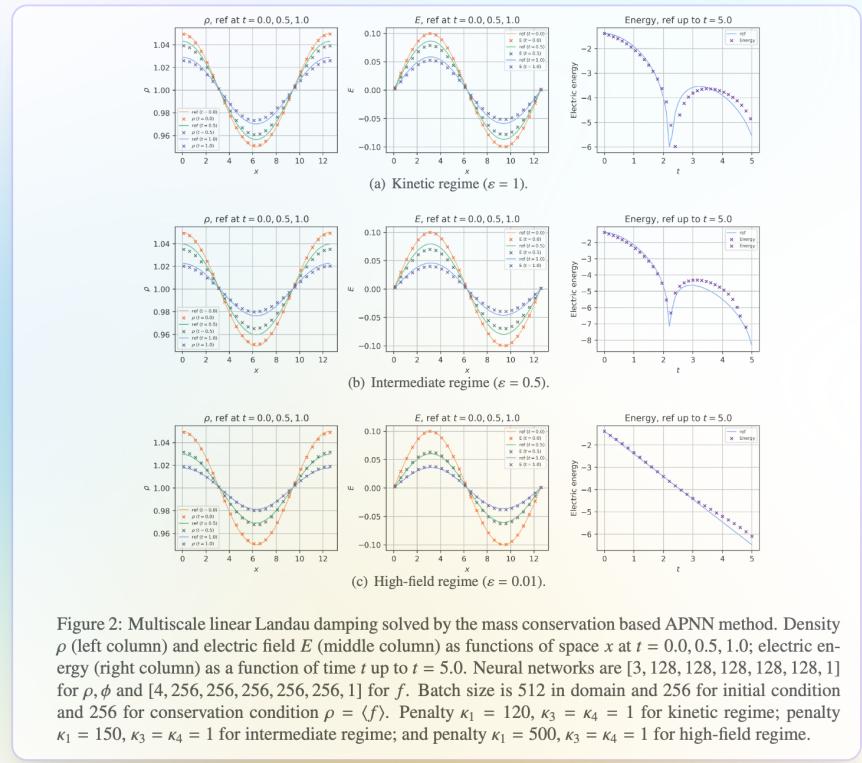
# APNNs for Vlasov-Poisson-Fokker-Planck system

## VPFP System

$$\begin{aligned} \partial_t f + v \cdot \nabla_x f - \frac{1}{\varepsilon} \nabla_x \phi \cdot \nabla_v f &= \frac{1}{\varepsilon} \nabla_v \cdot [v f + \nabla_v f], \\ -\nabla_x \phi(t, x) &= \rho(t, x) - h(x), \end{aligned}$$

where

$$\rho(t, x) = \int_{\mathbb{R}^N} f(t, x, v) dv$$



# APNNs for GRTE

APNNs for Multiscale Gray Radiative Transfer Equations

Original system

$$\begin{cases} \frac{\varepsilon^2}{c} \frac{\partial I}{\partial t} + \varepsilon \Omega \cdot \nabla I = \sigma \left( \frac{1}{4\pi} acT^4 - I \right), \\ \varepsilon^2 C_v \frac{\partial T}{\partial t} = \sigma \left( \int_{S^2} I d\Omega - acT^4 \right), \\ \mathcal{B}I = 0, \\ I(t=0, x, \Omega) = I_0(x, \Omega), \\ T(t=0, x) = T_0(x), \end{cases}$$

APNN based on even-odd decomposition

$$\begin{cases} \frac{\varepsilon^2}{c} \partial_t r + \frac{\varepsilon^2}{\sqrt{\sigma_0}} \cdot \mu \partial_x j = \sigma \left( \frac{1}{2} acT^4 - r \right), \\ \frac{\varepsilon^2}{c\sqrt{\sigma_0}} \partial_t j + \mu \partial_x r = -\frac{\sigma}{\sqrt{\sigma_0}} j, \\ \frac{1}{c} \partial_t \rho + \frac{1}{\sqrt{\sigma_0}} \cdot \langle \mu \partial_x j \rangle = -\frac{1}{2} C_v \frac{\partial T}{\partial t}, \\ \varepsilon^2 C_v \frac{\partial T}{\partial t} = \sigma (2\rho - acT^4), \\ \rho = \langle r \rangle. \end{cases}$$

# APNNs for GRTE

Numerics and schema

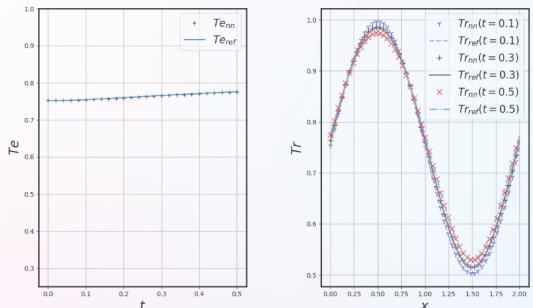
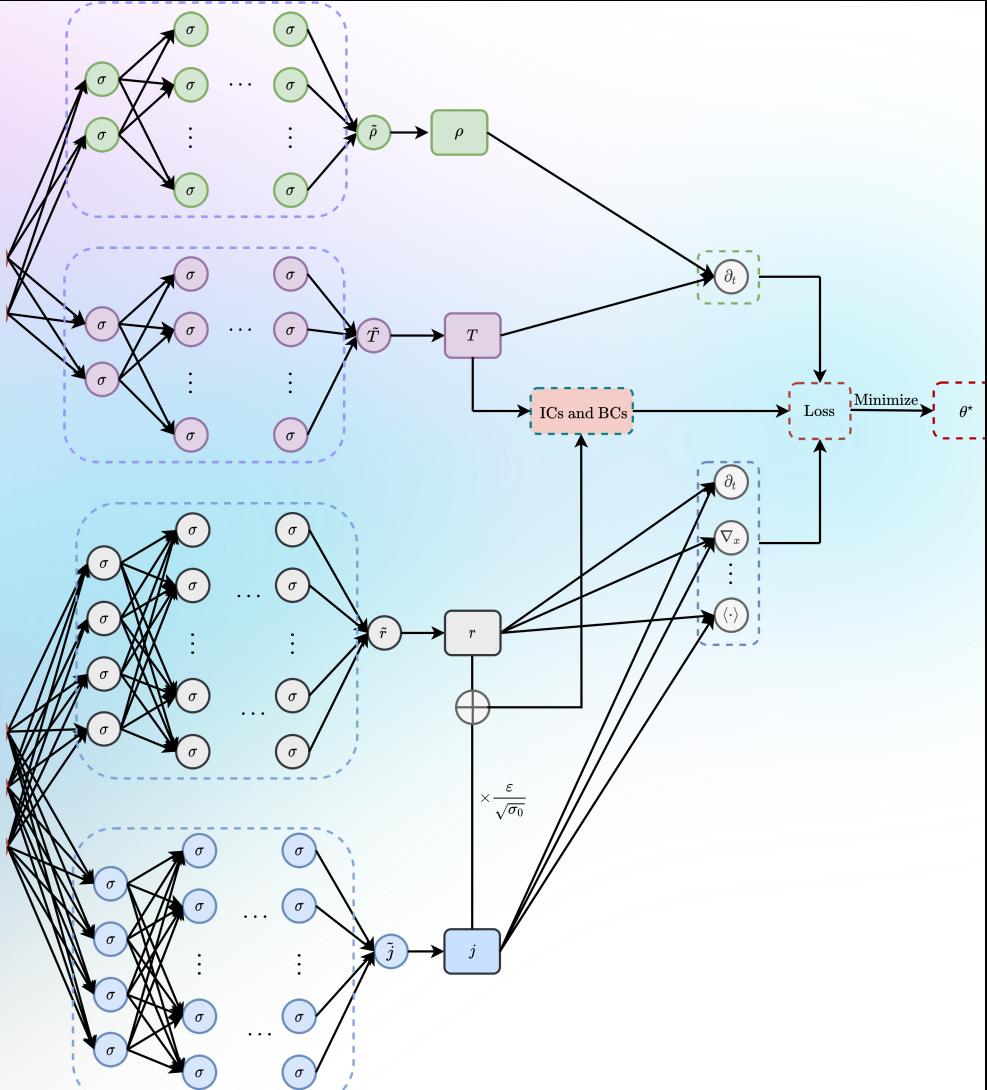
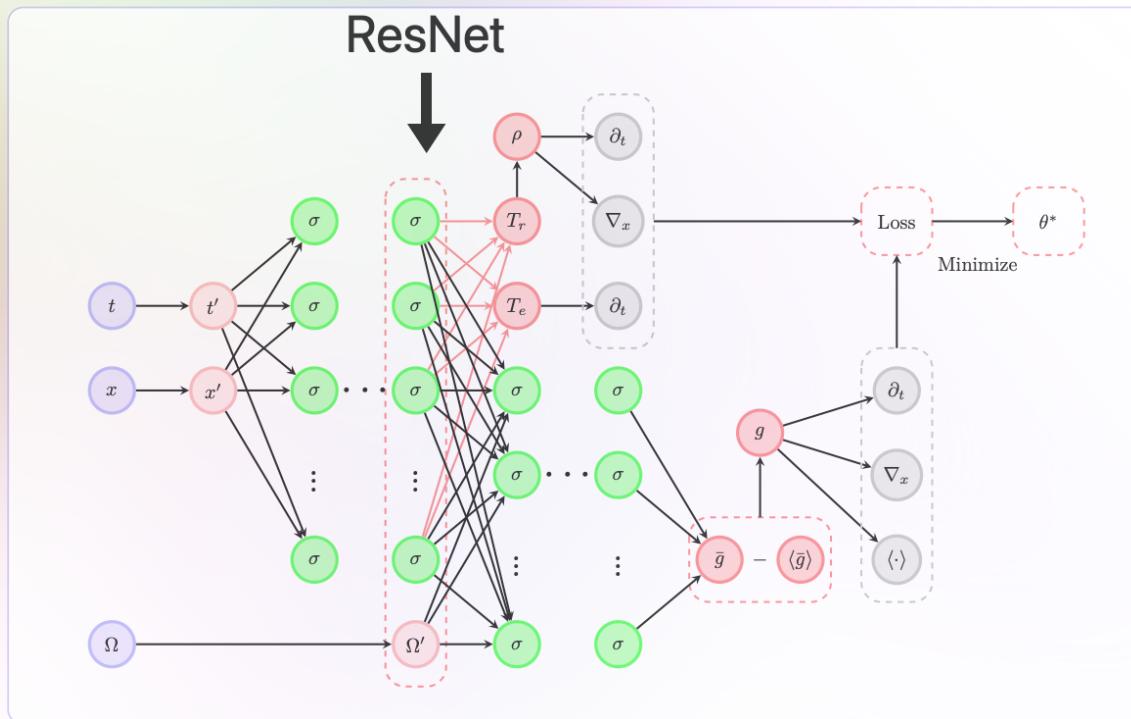


Figure 5: Plot of the approximated material temperature  $T_e = T$  at space  $x = 0.0025$  and the radiation temperature  $T_r$  at time  $t = 0.1, 0.3, 0.5$  with APNN based on micro-macro decomposition under the kinetic regime.



# RT-APNNs: ResNet

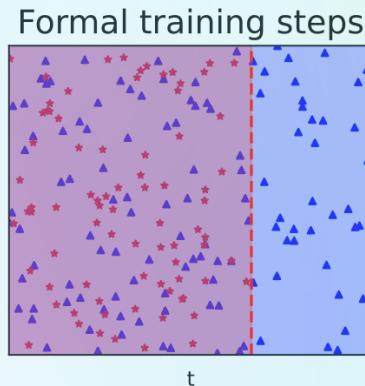
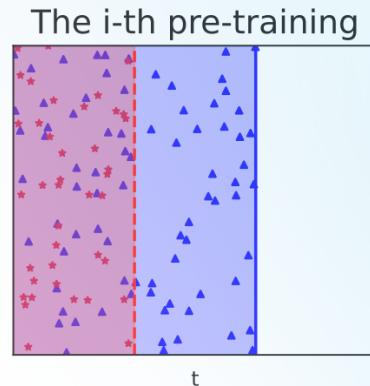
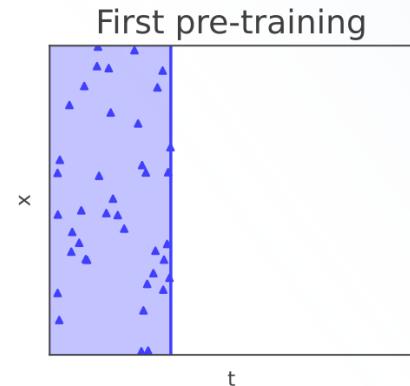
With 3 key improvements to APNNs



# RT-APNNs: Pre-training

With 3 key improvements to APNNs

## Pre-training steps



- ▲ Residual points      ━━ Previous training interval      □ Training region
- ★ Extra supervision points      ── Training interval      ■ Previous training region

Improve the **long time** stability

# RT-APNNs: MCMC

With 3 key improvements to APNNs

---

**Algorithm 1** Adaptive sampling in MCMC [54].

---

**Require:** Initial points  $\{(t_j^{(0)}, x_j^{(0)})\}_{j=1}^{N_r}$ , upper bound  $U_b$ , lower bound  $L_b$ , Update function  $U$ , Indicator function  $\pi$ , step size  $\epsilon$ , steps  $N_{\text{mcmc}}$ .

- 1: **for**  $j = 1$  to  $N_r$  **do**
- 2:   **for**  $i = 1$  to  $N_{\text{mcmc}}$  **do**
- 3:     Generate  $u$  from  $\mathcal{U}[0, 1]$ .
- 4:     Sample new intermediate state  $(t', x') \sim \mathcal{N}((t_j^{(i-1)}, x_j^{(i-1)}), \Sigma)$ ,
- 5:     Map to new proposal state:  $(t^*, x^*) = U(t', x')$ .
- 6:     Calculate acceptance probability:

$$\alpha(t_j^{(i-1)}, x_j^{(i-1)}; t^*, x^*) = \min \left\{ 1, \frac{\pi(t_j^{(i-1)}, x_j^{(i-1)})}{\pi(t^*, x^*)} \right\} \quad (3.3.3)$$

- 7:     **if**  $u < \alpha(t_j^{(i-1)}, x_j^{(i-1)}; t^*, x^*)$  **then**
- 8:       Accept proposal:  $(t_j^{(i)}, x_j^{(i)}) = (t^*, x^*)$ .
- 9:     **else**
- 10:       Reject proposal:  $(t_j^{(i)}, x_j^{(i)}) = (t_j^{(i-1)}, x_j^{(i-1)})$ .
- 11:     **end if**
- 12:   **end for**
- 13:   Set  $(t_j, x_j) = (t_j^{(N_{\text{mcmc}})}, x_j^{(N_{\text{mcmc}})})$ .
- 14: **end for**

**Ensure:** The newly obtained collocation points

---

Improve the **Sampling** efficiency

# RT-APNNs Result: Marshak Wave

ResNet + Pre-training + MCMC

Table 2: The relative  $L^2$  error of  $\rho$  at time  $t = 0.04, 0.1, 0.3, 2.0$  for the diffusion regime ( $\varepsilon = 10^{-8}$ ) in Ex 1 [26].

$L^2$ error	$t = 0.04$	$t = 0.1$	$t = 0.3$	$t = 2.0$
PINNs	7.05e - 01	7.57e - 01	7.49e - 01	3.63e - 01
APNNs	4.54e - 02	1.62e - 02	5.31e - 03	5.78e - 03
RT-APNN	1.61e - 02	4.73e - 03	3.64e - 03	1.55e - 03

Table 3: The relative  $L^2$  error of  $\rho$  at times  $t = 0.04, 0.1, 0.3, 2.0$  with Adam training 10,000 steps in the diffusion regime ( $\varepsilon = 10^{-8}$ ) in Ex 1.

$L^2$ error	$t = 0.04$	$t = 0.1$	$t = 0.3$	$t = 2.0$
APNNs	2.99e - 02	9.14e - 03	3.84e - 03	4.54e - 04
APNNs+①	1.24e - 02	6.17e - 03	4.02e - 03	2.98e - 03
APNNs+①②	1.34e - 02	6.04e - 03	6.10e - 03	1.94e - 03
APNNs+①②③ (RT-APNN)	1.55e - 02	6.89e - 03	2.33e - 03	9.89e - 04

Obtain higher accuracy and efficiency, especially for long time simulation

# **APNN with Different Architectures**

APRFMs, APCONs, etc.

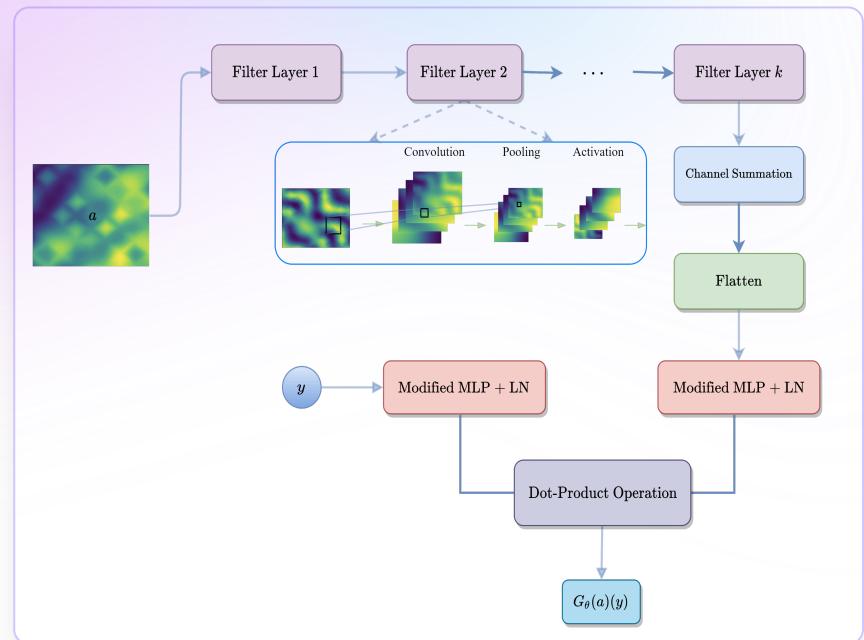
# APCONs: Operator Learning

APNN + Modified DeepONets

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

Our goal is to learn the mapping  $\mathcal{G}$  from

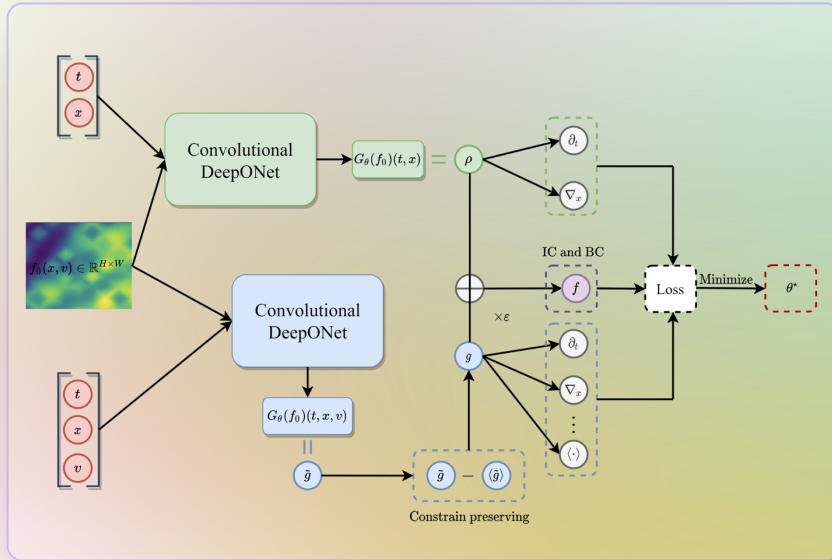
$$\mathcal{G} : f_0(x, v) \rightarrow f(t, x, v)$$



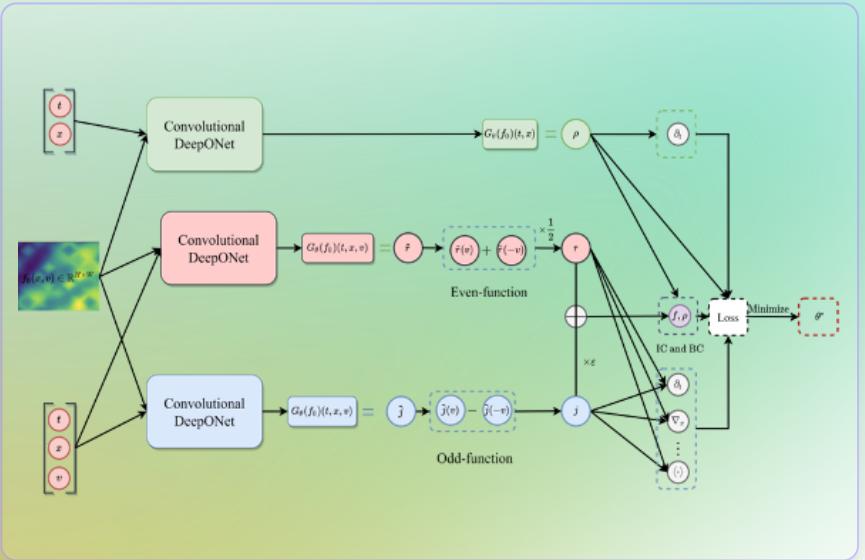
Convolutional DeepONets

# Schematic of APCONs

APCON based on Micro-macro decomposition



APCON based on even-odd decomposition



# APCONs Performance

## Accuracy and parameters

Table 1: Comparison in kinetic regime ( $\varepsilon = 1$ ) of Problem I.

QoIs \ Method	PI		AP v1		AP v2	
	DON	CON	DON	CON	DON	CON
Relative $\ell^2$ error	1.88 e-2	1.74 e-2	3.32 e-2	1.59 e-2	1.48 e-2	1.44 e-2
# params	423 296	43 288	846 400	86 384	1 269 696	129 672

Table 2: Comparison in the diffusion regime ( $\varepsilon = 10^{-4}$ ) of Problem I.

QoI \ Method	PI		AP v1		AP v2	
	DON	CON	DON	CON	DON	CON
Relative $\ell^2$ error	—	—	2.63e-2	1.58e-2	4.09e-2	1.55 e-2

## APCON efficiency

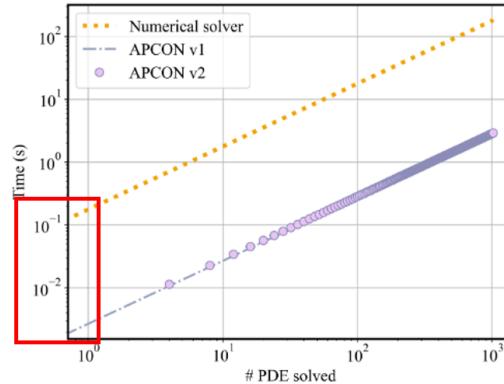


Figure 10: Computational cost (second) for performing inference with pre-trained APCONs, along with the corresponding time required to solve a PDE with a conventional fast spectral method which encompasses a scalable GPU implementation utilizing cupy. Reported timings are obtained on a single NVIDIA RTX 3090 graphics processing unit (GPU). The average total time costs for the classical numerical solver, APCON-v1, and APCON-v2 are 176.9 ms, 2.69 ms, and 2.85 ms, respectively.

With less parameters than PIDON, 100X faster than conventional numerical methods

# Conclusions

- We proposed APNNs framework for solving multiscale kinetic equations by deep learning:
  - Based on micro-macro decomposition: APNN v1
  - Based on odd-even parity method: APNN v1
  - Based on local conservation laws: APNN v2
  - Combined with ResNet, pre-training and MCMC: RT-APNNs
  - Combined with random feature methods: APRFMs
  - Combined with **operator learning** : APCONs
- APNNs can be applied to various multiscale kinetic equations, such as:
  - Linear transport equations
  - Boltzmann-BGK equation
  - Vlasov-Poisson-Fokker-Planck system
  - Gray radiative transfer equations
  - and more: **full Boltzmann equations**

# References

- Keke Wu, Xizhe Xie, Wengu Chen, Han Wang and Zheng Ma, RT-APNN for Solving Gray Radiative Transfer Equations, *preprint*, 2025
- Jingru Chen, Zheng Ma and Keke Wu, A micro-macro decomposition-based asymptotic-preserving random feature method for multiscale radiative transfer equations, *Jounral of Computational Physics*, 2025
- Shi Jin, Zheng Ma and Keke Wu, Asymptotic-Preserving Neural Networks for Multiscale Kinetic Equations, *Communication in Computational Physics*, 2024
- Shi Jin, Zheng Ma and Tian-ai Zhang, Asymptotic-preserving neural networks for multiscale Vlasov-Poisson-Fokker-Planck system in the high-field, *Jounral of Scientific Computing*, 2024
- Keke Wu, Xiong-bin Yan, Shi Jin and Zheng Ma, Capturing the Diffusive Behavior of the Multiscale Linear Transport Equations by Asymptotic-Preserving Convolutional DeepONets, *Computer Methods in Applied Mechanics and Engineering* , 2024
- Shi Jin, Zheng Ma and Keke Wu, Asymptotic-Preserving Neural Networks for Multiscale Time-Dependent Linear Transport Equations, *Journal of Scientific Computing*, 2023

# Thank You!

Slides can be found [here](#)