Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation

Qin Lou[a,b,1], Xuhui Meng[c,1], George Em Karniadakis[c,d,2]

[a]*School of Energy and Power Engineering, University of Shanghai for Science and Technology, Shanghai 200093, PR China*
[b]*Shanghai Key Laboratory of Multiphase Flow and Heat Transfer in Power Engineering, University of Shanghai for Science and Technology, Shanghai 200093, PR China*
[c]*Division of Applied Mathematics, Brown University, Providence, RI 02906, USA*
[d]*Pacific Northwest National Laboratory, Richland, WA 99354, USA*

Abstract

The Boltzmann equation with the Bhatnagar-Gross-Krook collision model (Boltzmann-BGK equation) has been widely employed to describe multiscale flows, i.e., from the hydrodynamic limit to free molecular flow. In this study, we employ physics-informed neural networks (PINNs) to solve forward and inverse problems via the Boltzmann-BGK formulation (PINN-BGK), enabling PINNs to model flows in both the continuum and rarefied regimes. In particular, the PINN-BGK is composed of three sub-networks, i.e., the first for approximating the equilibrium distribution function, the second for approximating the non-equilibrium distribution function, and the third one for encoding the Boltzmann-BGK equation as well as the corresponding boundary/initial conditions. By minimizing the residuals of the governing equations and the mismatch between the predicted and provided boundary/initial conditions, we can approximate the Boltzmann-BGK equation for both continuous and rarefied flows. For forward problems, the PINN-BGK is utilized to solve various benchmark flows given boundary/initial conditions, e.g., Kovasznay flow, Taylor-Green flow, cavity flow, and micro Couette flow for Knudsen number up to 5. For inverse problems, we focus on rarefied flows in which accurate boundary conditions are difficult to obtain. We employ the PINN-BGK to infer the flow field in the entire computational domain given a limited number of interior scattered measurements on the velocity without using the (unknown) boundary conditions. Results for the two-dimensional micro Couette and micro cavity flows with Knudsen numbers ranging

[1]The first two authors contributed equally to this work.
[2]Corresponding author: george_karniadakis@brown.edu (George Em Karniadakis).

from 0.1 to 10 indicate that the PINN-BGK can infer the velocity field in the entire domain with good accuracy. Finally, we also present some results on using transfer learning to accelerate the training process. Specifically, we can obtain a three-fold speedup compared to the standard training process (e.g., Adam plus L-BFGS-B) for the two-dimensional flow problems considered in our work.

Keywords: Boltzmann-BGK equation, physics-informed neural networks, rarefied flows, velocity slip, inverse problems, transfer learning

1. Introduction

Multiscale flows spanning the rarefied and continuum regimes are present in diverse applications, such as shale gas flow in porous media [1, 2], vacuum technology [3, 4], and microfluidics [5, 6, 7]. The Boltzmann equation with the Bhatnagar-Gross-Krook collision model (Boltzmann-BGK equation) [8] is a widely used model for descriptions of both rarefied and continuous flows [9, 10]. Several numerical methods have been developed to solve the Boltzmann-BGK equation in recent years since analytical solutions are difficult to obtain, such as finite difference based methods, e.g., lattice Boltzmann method (LBM, in the low-speed limit at the continuum regime) [11, 12, 13, 14, 15, 16, 17, 18, 19, 20], finite volume based methods [21], e.g., the uniform gas kinetic scheme (UGKS) [22, 23, 24], the discrete ordinate method (DOM) [25], the discrete uniform gas kinetic scheme (DUGKS) [26, 27], and so on. Successful applications of the aforementioned approaches include multiscale heat transfer [28], reactive transport in porous media [29], and non-equilibrium flows [24], to name just a few.

In addition to the conventional numerical methods, deep learning algorithms have emerged recently as an alternative for solving partial differential equations (PDEs), especially in conjunction with sparse data [30, 31, 32, 33]. In particular, we focus on the physics-informed neural networks (PINNs) proposed in [33] due to their effectiveness in solving both forward and inverse PDE problems as well as the straightforward implementation. Different from the classical numerical methods in which the differential operators are approximated by the data on certain discrete lattices (meshes), PINNs compute all the differential operators of a PDE using the automatic differentiation technique [34] involved in the backward propagation [33]. Consequently, no mesh (structured mesh or unstructured mesh used in the classical numerical methods) is required for the PINN to solve PDEs, which saves a lot of in grid generation [35, 36]. Another attractive feature is that PINNs are capable of solving the inverse PDE problems effectively [33, 37] and with the same code that is used for forward problems. Specifically, PINNs can infer the unknown parameters in

a PDE and reconstruct the solution given partial observations on the solution to the PDE. Successful examples of using PINNs for forward and inverse flow problems are: (1) modeling both laminar and turbulent channel flows [38], (2) learning the density, velocity, and pressure fields based on partial observations on the density gradient in high-speed flows [39], and (3) inferring the velocity and pressure fields given measurements from spatio-temporal visualizations of a passive scalar, e.g., temperature or solute concentration [40].

Despite the great progress of PINNs for solving such flow problems, so far only flows in the continuum regime described by the Navier-Stokes or Euler equations have been considered, to the best of our knowledge. However, for flow problems in the rarefied or transitional regimes such as gas flow in shale nanopores [41, 42], no existing PINNs or their variants can be applied directly. In addition to PINNs, Han *et al.* proposed to solve the Boltzmann-BGK equation using the deep convolution neural networks [43], which works well for multiscale flows with a Knudsen number ranging from 10^{-3} to 10. However, only forward problems are considered in [43], i.e., solving the Boltzmann-BGK equation given boundary/initial conditions. In practical applications, accurate boundary conditions can be quite difficult to obtain, especially for (1) multiscale flows across different flow regimes [44, 45, 46, 47, 48], and (2) rarefied flows within complicated geometries, e.g., porous media [49, 50, 51, 52]. An interesting question arising here is: can we predict the velocity field when we only have access to a limited number of scattered interior measurements on the velocity rather than the boundary conditions? We refer to this problem as an "inverse problem" in the present study. To leverage the merits of PINNs for both forward and inverse problems, we propose herein to employ PINNs for solving both the forward and inverse problems via Boltzmann-BGK formulation (PINN-BGK), hence developing a new PINN capability in simulating seamlessly both the continuum and rarefied regimes.

The remainder of the paper is organized as follows. In Section 2, we describe the details of PINNs for solving the Boltzmann-BGK equation. In Section 3, we present the results for modeling some benchmark two-dimensional flows in both the continuum and rarefied regimes. In Section 4, we apply the PINN-BGK to inverse problems, i.e., inferring the flow field in the entire computational domain given a limited number of scattered measurements on the velocity. A brief summary is then presented in Sec. 5. We finally present more details on the PINN-BGK and a systematic study on the parameters in PINN-BGK in Appendix A and Appendix B, respectively.

2. Methodology

2.1. From Boltzmann-BGK equation to discrete velocity Boltzmann-BGK model

The Boltzmann equation with the simplified collision model, i.e., Bhatnagar-Gross-Krook (BGK) model [8], is considered here, which can be expressed as

$$\begin{aligned} \frac{\partial \boldsymbol{f}}{\partial t} + \boldsymbol{\xi} \cdot \nabla \boldsymbol{f} &= -\frac{1}{\tau} \left(\boldsymbol{f} - \boldsymbol{f}^{eq} \right) \\ \boldsymbol{f}^{neq} &= \left(\boldsymbol{f} - \boldsymbol{f}^{eq} \right), \end{aligned} \tag{1}$$

where $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\xi}, t)$ is the particle distribution function moving with a velocity $\boldsymbol{\xi}$ at position $\boldsymbol{x}$ and time t, τ is the relaxation time related to the dynamic viscosity μ and the fluid pressure p as $\tau = \mu/p$, and $\boldsymbol{f}^{eq}$ is the equilibrium distribution function given by the Maxwellian equilibrium distribution function,

$$\boldsymbol{f}^{eq} = \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\frac{|\boldsymbol{\xi} - \boldsymbol{u}|^2}{2RT} \right), \tag{2}$$

where R is the gas constant, T is the temperature, D is the spatial dimension, ρ is the density related to the fluid pressure as $\rho = p/RT$, $\boldsymbol{u}$ is the fluid velocity; $\boldsymbol{f}^{neq}$ denotes the non-equilibrium part of the distribution function. The conservative flow variables, i.e., ρ and $\boldsymbol{u}$, are the moments of the distribution function and can be computed as follows [53, 54],

$$\boldsymbol{\phi} = \int \boldsymbol{\psi} f d\boldsymbol{\xi}, \ \boldsymbol{\psi} = (1, \boldsymbol{\xi}), \tag{3}$$

where

$$\boldsymbol{\phi} = (\rho, \ \rho\boldsymbol{u}), \ \rho = \int f d\boldsymbol{\xi}, \ \rho\boldsymbol{u} = \int \boldsymbol{\xi} f d\boldsymbol{\xi}. \tag{4}$$

Note that the aforementioned particle velocity $\boldsymbol{\xi}$ is continuous, which makes it computationally inefficient to solve Eq. (1). To reduce the computational cost, the discrete velocity model is generally used while keeping the mass and momentum conserved [21, 53]. In particular, one can project Eq. (1) onto the Hermite basis by expanding $f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ in terms of the Hermite orthonormal polynomials in the velocity space $\boldsymbol{\xi}$. In addition, by truncating the Hermite polynomials up to Nth order, the following discrete velocity Boltzmann-BGK model (DVB) can be obtained (See [54] for more details on the derivations):

$$\frac{\partial f_i}{\partial t} + \boldsymbol{\xi_i} \cdot \nabla f_i = -\frac{1}{\tau} \left(f_i - f_i^{eq} \right), \ i = 0, ..., Q-1, \tag{5}$$

where $\boldsymbol{\xi}_i$ is the ith discrete velocity, and Q here is the number of discrete velocities. Meanwhile, Eq. (5) serves as the governing equation and will be solved in this study.

For the low-speed isothermal flows (i.e., the Mach number $Ma \approx |\boldsymbol{u}|/\sqrt{RT} \ll 1.0$) considered in this study, f_i^{eq} is the Hermite expansion up to second order of Ma for the Maxwellian equilibrium distribution function (i.e., Eq. (2)), which reads as [26, 54]

$$f_i^{eq}(\rho, \boldsymbol{u}) = w_i \rho \left[1 + \frac{\boldsymbol{\xi}_i \cdot \boldsymbol{u}}{RT} + \frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})^2}{2(RT)^2} - \frac{\boldsymbol{u} \cdot \boldsymbol{u}}{2RT}\right], \tag{6}$$

where ω_i is a set of constant weights for different $\boldsymbol{\xi}_i$. In addition, the conservative flow variables, i.e., ρ and $\boldsymbol{u}$, are computed as

$$\rho = \sum_{i=0}^{Q-1} f_i = \sum_{i=0}^{Q-1} f_i^{eq}, \ \rho \boldsymbol{u} = \sum_{i=0}^{Q-1} \boldsymbol{\xi}_i f_i = \sum_{i=0}^{Q-1} \boldsymbol{\xi}_i f_i^{eq}, \tag{7}$$

based on the property of the collision operation, i.e., the collision operator conserves the mass as well as the moment [54]. The employed discrete velocity model will be introduced for each specific case in Secs. 3 and 4.

2.2. Physics-informed neural networks for solving DVB model

In this section, we first discuss how to extend PINNs for solving both forward and inverse problems via the Boltzmann-BGK formulation (i.e., PINN-BGK). Then we present the boundary conditions employed in our study.

2.2.1. PINN-BGK

As shown in Fig. 1, the PINN-BGK is composed of three sub-networks, i.e., one to approximate the equilibrium distribution function (i.e., $\mathcal{NN}_{eq}$), the second one for the non-equilibrium distribution function (i.e., $\mathcal{NN}_{neq}$), and the last one to encode the Boltzmann-BGK equation as well as the corresponding boundary/initial conditions. In particular, $\mathcal{NN}_{eq}$ takes $(\boldsymbol{x}, t)$ as input, and outputs the macroscopic variables, i.e., ρ, and $\boldsymbol{u}$. $f_i^{eq}(\rho, \boldsymbol{u})$ can then be obtained based on Eq. (6). Meanwhile, $\mathcal{NN}_{neq}$ takes in $(\boldsymbol{x}, t)$ and outputs the non-equilibrium part of f_i (i.e., f_i^{neq}) for each $\boldsymbol{\xi}_i$. More details on the PINN-BGK can be found in Appendix A.

Note that the distribution function $f_i(\boldsymbol{x}, t)$ is decomposed into an equilibrium part (i.e., f_i^{eq}) and a non-equilibrium part (i.e., f_i^{neq}), i.e., $f_i(\boldsymbol{x}, t) = f_i^{eq} + f_i^{neq}$, in the PINN-BGK. Generally, the magnitudes of f_i^{eq} and f_i^{neq} can be quite different. For continuous flows, we can conduct the following multiscale analysis for Eq. (5) since the relaxation time τ is a small parameter (e.g., $\tau < 10^{-2}$) [15, 55]. Specifically,

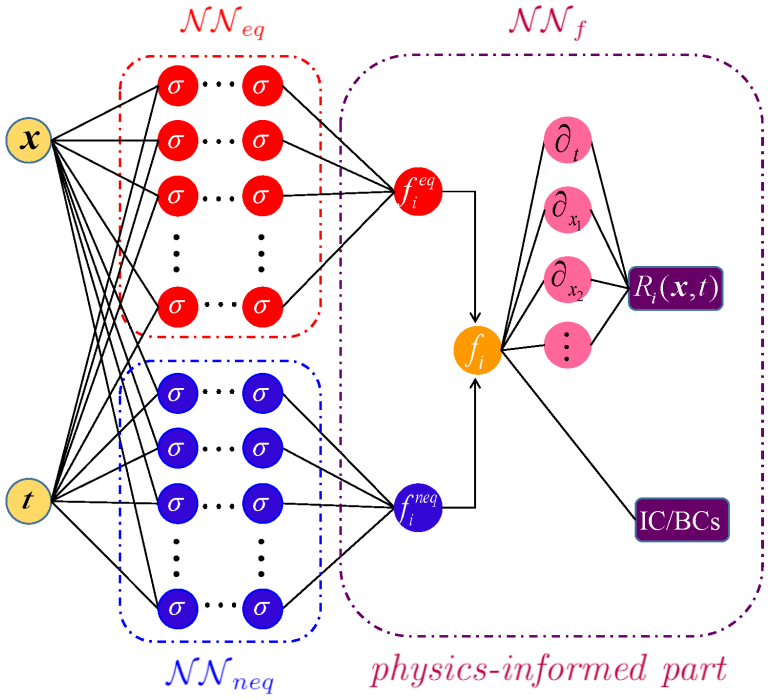


Figure 1: Schematic of PINNs for solving the discrete velocity Boltzmann-BGK equation (PINN-BGK). $\mathcal{NN}_{eq}$ and $\mathcal{NN}_{neq}$ are employed to approximate the equilibrium (f_i^{eq}) and non-equilibrium (f_i^{neq}) parts of the distribution functions, respectively. σ is the activation function, which is set as the hyperbolic tangent function in this work. $R_i(\boldsymbol{x},t) = \partial_t f_i + \xi_{ix}\partial_x f_i + (f_i - f_i^{eq})/\tau$ is the residual of the Boltzmann-BGK equation. IC/BCs represent the initial and boundary conditions.

we can expand f_i in terms of τ as [56]

$$f_i = f_i^{(0)} + \tau f_i^{(1)} + \tau^2 f_i^{(2)} + ... \tag{8}$$

Substituting Eq. (8) into Eq. (5), we can obtain the following equations [56]:

$$\frac{1}{\tau}: \ f_i^{(0)} = f_i^{eq}, \tag{9}$$

$$\tau^0: \ f_i^1 = -\left(\partial_t f_i^{(0)} + \boldsymbol{\xi}_i \cdot \nabla f_i^{(0)}\right). \tag{10}$$

As we truncate f_i to $O(\tau^2)$, the following equation can then be obtained

$$f_i = f_i^{(eq)} + \tau f_i^{(1)} + O(\tau^2). \tag{11}$$

With the aid of Eq. (10), we can rewrite Eq. (11) as

$$f_i \approx f_i^{eq} - \tau \left(\partial_t f_i^{(0)} + \boldsymbol{\xi}_i \cdot \nabla f_i^{(0)} \right), \tag{12}$$

which can be further rewritten as

$$f_i^{neq} = f_i - f_i^{eq} \approx -\tau \left(\partial_t f_i^{eq} + \boldsymbol{\xi}_i \cdot \nabla f_i^{eq} \right), \tag{13}$$

where f_i^{neq} is the same as in Fig. 1. As we can see in Eqs. (9) and (13), f_i^{eq} can be more than 100 times larger than the f_i^{neq} in continuous flows [57, 58]. If we approximate f_i with one deep neural network (DNN) directly, it is probable that f_i^{neq} will not be resolved accurately since f_i^{eq} dominates over f_i. However, f_i^{neq} has a strong influence on the convergence of the governing equation in Eq. (5) due to the fact that τ is also a small parameter, e.g., $\tau \sim O(10^{-4})$ in Sec. 3.1 (Numerical results for the effect of f_i^{neq} on predicted accuracy are presented in Sec. 3.1). We therefore apply two DNNs to approximate f_i^{eq} and f_i^{neq}, respectively.

In forward problems, we assume that both the initial/boundary conditions and the governing equations are known, and hence we can solve the Boltzmann-BGK equation by minimizing the following loss function,

$$\begin{aligned} L &= L_{Eq} + L_{IC} + L_{BC}, \\ L_{Eq} &= \frac{1}{N_e} \sum_{i=0}^{Q-1} \sum_{j=0}^{N_e-1} |R_i(\boldsymbol{x}_j, t_j)|^2, \\ L_{IC} &= \frac{1}{N_i} \sum_{j=0}^{N_i-1} |\phi_{\mathcal{NN}_{eq}}(\boldsymbol{x}_j, 0) - \phi^*(\boldsymbol{x}_j, 0)|^2, \end{aligned} \tag{14}$$

where $R_i(\boldsymbol{x}, t)$ is residual of the governing equation that is defined as

$$R_i := \frac{\partial f_i}{\partial t} + \boldsymbol{\xi}_i \cdot \nabla f_i + \frac{1}{\tau} \left(f_i - f_i^{eq} \right). \tag{15}$$

In addition, L_{Eq}, L_{IC}, and L_{BC} represent the loss function for the residual of the DVB equation, the initial condition as well as the boundary condition. Specifically, ϕ denote the macroscopic variables, i.e., $\rho/\boldsymbol{u}$, and $\phi_{\mathcal{NN}_{eq}}$ and ϕ^* are the output of $\mathcal{NN}_{eq}$ and the given initial conditions, respectively. N_e and N_i are the number of the training data for governing equations and initial condition, respectively. Note that the initial conditions used here are similar as that is often used in the lattice Boltzmann method [15, 26]. Furthermore, we will present the details on L_{BC} in Sec. 2.2.2.

For inverse problems, we assume that we know the equations as well as a small set of interior measurements on the velocity, but the initial/boundary conditions are unknown. We then infer the velocity field by minimizing the following loss function:

$$\begin{aligned} L &= L_{Eq} + L_u, \\ L_u &= \frac{1}{N_u} \sum_{j=0}^{N_u-1} |u(\boldsymbol{x}_{u_j}, t_{u_j}) - u^*(\boldsymbol{x}_{u_j}, t_{u_j})|^2, \end{aligned} \tag{16}$$

where L_{Eq} is the same as in Eq. (14), and L_u denotes the mismatch between the predictions and the measurements, $(\boldsymbol{x}_u, t_u)$ are the locations of measurements, and N_u is the number of measurements.

2.2.2. Boundary conditions in PINN-BGK

In general, the pressure or velocity at a boundary is provided for the boundary condition in a flow problem. To solve the discrete velocity Boltzmann-BGK equation, we need to specify appropriate boundary conditions for the discrete distribution functions based on the given pressure/velocity, i.e., kinetic boundary conditions. In the present study, two different kinetic boundary conditions, i.e., Eq. (13) and the diffuse Maxwell boundary condition, are employed for the continuous and rarefied flows, respectively.

For continuous flows, we only consider cases with velocity boundary conditions in this study. Generally, the density at each boundary is unknown. As reported in [59], the density fluctuation in the incompressible limit is of order $O(Ma^2)$, where $Ma \approx |u|/\sqrt{RT} \ll 1.0$. The density field is initialized as 1 in our computations, we can thus approximate the density at the boundary as $\rho_w \approx 1$ [26]. Consequently, the unknown f_i^{eq} and f_i^{neq} at the boundary condition can be computed based on Eq. (6) and Eq. (13), respectively. The loss function for the velocity boundary condition in PINN-BGK is then expressed as

$$\begin{aligned} L_{BC} &= L_{m_{BC}} + L_{f_{BC}} + L_{E_{BC}}, \\ L_{m_{BC}} &= \frac{1}{N_b} \sum_{j=0}^{N_b-1} |\phi_{\mathcal{NN}_{eq}}(\boldsymbol{x}_{b_j}, t_j) - \phi^*(\boldsymbol{x}_{b_j}, t_j)|^2, \\ L_{f_{BC}} &= \frac{1}{N_b} \sum_{j=0}^{N_b-1} \sum_{\boldsymbol{\xi}_i \cdot \boldsymbol{n} > 0} |f^{neq}_{i,\mathcal{NN}_{neq}}(\boldsymbol{x}_{b_j}, t_j) - f_i^{neq,*}(\boldsymbol{x}_{b_j}, t_j)|^2, \\ L_{E_{BC}} &= \frac{1}{N_b} \sum_{j=0}^{N_b-1} \sum_{\boldsymbol{\xi}_i \cdot \boldsymbol{n} < 0} |R_i(\boldsymbol{x}_{b_j}, t_j)|^2, \end{aligned} \tag{17}$$

where N_b is the number of points used to compute the loss function for boundary conditions, $\phi_{\mathcal{NN}_{eq}}$ and ϕ^* are the output of $\mathcal{NN}_{eq}$ and the given boundary conditions, respectively; $f^{neq}_{i,\mathcal{NN}_{neq}}(\boldsymbol{x}_{b_j}, t_j)$ is the output of $\mathcal{NN}_{neq}$, and $f_i^{neq,*}(\boldsymbol{x}_{b_j}, t_j)$ is computed using Eq. (13); $\boldsymbol{n}$ is the unit inward-wall vector normal to the boundary. We would like to mention that $L_{m_{BC}}$ and $L_{f_{BC}}$ can be viewed as the boundary conditions for the equilibrium (f_i^{eq}) and non-equilibrium part (f_i^{neq}) in f_i since $f_i = f_i^{eq} + f_i^{neq}$.

For rarefied flows, the gas-surface interaction is implemented via the diffuse Maxwell boundary condition as follows [26]:

$$f_i(\boldsymbol{x}_b, t) = \rho_w f_i^{eq}(1, \boldsymbol{u}_w), \quad (\boldsymbol{\xi}_i \cdot \boldsymbol{n} > 0), \tag{18}$$

where $\boldsymbol{u}_w$ is the wall velocity, and ρ_w is the density at the wall which is determined based on the condition of impermeable wall as [26]

$$\rho_w = -\frac{\sum_{\boldsymbol{\xi}_i \cdot \boldsymbol{n} < 0} (\boldsymbol{\xi}_i \cdot \boldsymbol{n}) f_i(\boldsymbol{x}_b, t)}{\sum_{\boldsymbol{\xi}_i \cdot \boldsymbol{n} > 0} (\boldsymbol{\xi}_i \cdot \boldsymbol{n}) f_i^{eq}(1, \boldsymbol{u}_w)}. \tag{19}$$

Note that Eqs. (18) and (19) are used to determine the unknown distribution functions with velocities pointing to the flow. For particles leaving the wall, we again impose them to satisfy Eq. (5), which is the same as in continuous flows. We then express the boundary conditions in the loss function as

$$\begin{aligned}
L_{BC} &= L_{f_{BC}} + L_{E_{BC}}, \\
L_{f_{BC}} &= L_{f_{BC,1}} + L_{f_{BC,2}}, \\
L_{f_{BC,1}} &= \frac{1}{N_b} \sum_{j=0}^{N_b - 1} \sum_{\boldsymbol{\xi}_i \cdot \boldsymbol{n} > 0} |f_{i,\mathcal{NN}}(\boldsymbol{x}_{b_j}, t_j) - f_i^*(\boldsymbol{x}_{b_j}, t_j)|^2, \\
L_{f_{BC,2}} &= \frac{1}{N_b} \sum_{j=0}^{N_b - 1} \left[|\sum_{i=0}^{Q-1} f^{neq}_{i,\mathcal{NN}_{neq}}(\boldsymbol{x}_{b_j}, t_j)|^2 + |\sum_{i=0}^{Q-1} \boldsymbol{\xi}_i f^{neq}_{i,\mathcal{NN}_{neq}}(\boldsymbol{x}_{b_j}, t_j)|^2 \right], \\
L_{E_{BC}} &= \frac{1}{N_b} \sum_{j=0}^{N_b - 1} \sum_{\boldsymbol{\xi}_i \cdot \boldsymbol{n} < 0} |R_i(\boldsymbol{x}_{b_j}, t_j)|^2,
\end{aligned} \tag{20}$$

where $f_{i,\mathcal{NN}} = f^{eq}_{i,\mathcal{NN}_{eq}} + f^{neq}_{i,\mathcal{NN}_{neq}}$ is the predicted distribution function using the DNNs, $f_i^*(\boldsymbol{x}_{b_j}, t_j)$ is obtained using Eqs. (18) and (19), and R_i is predicted residual in Eq. (15). It is worth mentioning that only $L_{f_{BC,1}}$ and $L_{E_{BC}}$ are needed to determine the unknown distribution function $f_i(\boldsymbol{x}_b, t)$ at the boundary. However, here we add one more constraint for $f^{neq}_{i,\mathcal{NN}_{neq}}$ based on Eq. (7), i.e., $L_{f_{BC,2}}$, since $f_i(\boldsymbol{x}_b, t)$ is

decomposed into an equilibrium (f_i^{eq}) and a non-equilibrium part (f_i^{neq}), which is similar as in the continuous flows discussed above.

We would like to point out that various kinetic initial/boundary conditions have been developed during the past few decades, and interested readers can refer to [15, 14, 26, 56] for more details.

3. Forward problems

In this section, we will apply the PINN-BGK to model various two-dimensional flow problems given boundary/initial conditions, i.e., (1) continuous flows including time-independent/dependent cases, e.g., Kovasznay flow and Taylor-Green flow; and (2) rarefied flows, i.e., micro-Couette flows with Knudsen numbers ranging from 0.01 to 5.

3.1. Kovasznay flow

We first employ the PINN-BGK to simulate the two-dimensional steady incompressible Kovasznay flow in a two-dimensional rectangular domain with a length $-\frac{1}{2} \leq x \leq 2$ and height $-\frac{1}{2} \leq y \leq \frac{3}{2}$. The exact solutions for the flow considered here are [60]:

$$\begin{aligned} u(x,\ y) &= u_0[1 - \exp(\lambda x)\cos(2\pi y)], \\ v(x,\ y) &= u_0[\frac{\lambda}{2\pi}\exp(\lambda x)\sin(2\pi y)], \\ p(x,\ y) &= p_0[1 - \frac{1}{2}\exp(2\lambda x)] + C, \end{aligned} \tag{21}$$

where u_0 and p_0 are reference velocity and reference pressure, respectively, C is a constant, L is half the height of the computational domain, $\boldsymbol{u} = (u, v)$ represents the fluid velocity, p is the fluid pressure, and

$$\lambda = \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}, \quad Re = \frac{u_0 L}{\nu}, \tag{22}$$

where ν is the kinematic viscosity. In addition, Dirichlet boundary conditions which can be derived from Eq. (21) are imposed on all boundaries.

Here we employ the two-dimensional-nine-speed (D2Q9) model [61], which is

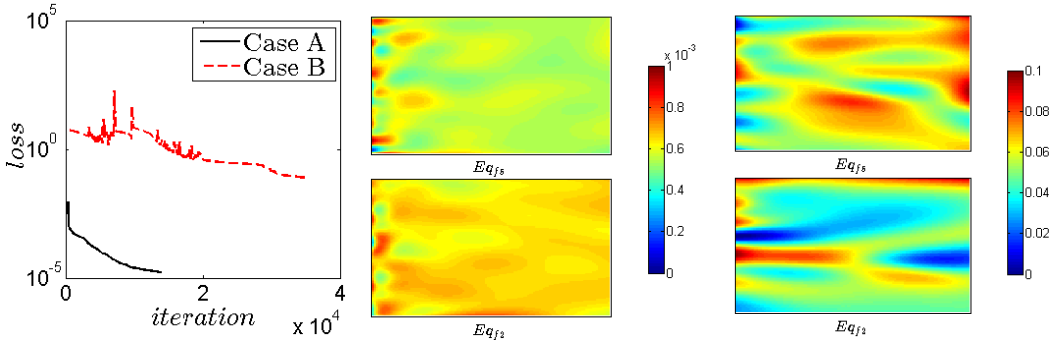
widely used for continuous flows. Specifically, D2Q9 is defined as $\boldsymbol{\xi}_i = c\boldsymbol{e}_i$ in which,

$$\begin{aligned} c = \sqrt{3RT}, w_0 = 4/9, \\ w_{1\sim4} = 1/9, w_{5\sim8} = 1/36, \\ \boldsymbol{e}_0 = (0,\ 0), \\ \boldsymbol{e}_1 = -\boldsymbol{e}_3 = (1,\ 0),\ \boldsymbol{e}_2 = -\boldsymbol{e}_4 = (0,\ 1), \\ \boldsymbol{e}_5 = -\boldsymbol{e}_7 = (1,\ 1),\ \boldsymbol{e}_6 = -\boldsymbol{e}_8 = (-1, 1). \end{aligned} \tag{23}$$

The D2Q9 model will be used in all continuous flow cases in this study, i.e., Secs. 3.1, 3.2, and 3.3. In our simulations, Re = 10, $u_0 = 0.1581$, $p_0 = 0.05$, $RT = 100$, $C = RT$, and $L = 1$. The kinematic viscosity and the relaxation time can then be computed as $\nu = Lu_0/Re = 0.0158$ and $\tau = \mu/p = \nu/RT = 1.58 \times 10^{-4}$. In addition, we randomly sample $17,000$ points for the evaluation of residuals, and 300 points are used on each boundary to provide Dirichlet boundary conditions. As for the optimization, we first employ the Adam [62] to train the PINN-BGK until the loss is less than 10^{-3}, then we switch to the L-BFGS-B [63] until convergence.

We note that $\tau \sim O(10^{-4})$ in the present case, leading to $f_i^{neq} \sim O(10^{-4})$ according to Eq. (13). Based on our experience, the optimization algorithm works well as the output of a NN is of the order $O(1)$. Such a small value of f_i^{neq} in the present case will make it difficult to be approximated by DNNs. We then scale the output of $\mathcal{NN}_{neq}$ as $f^{neq}_{i,\mathcal{NN}_{neq}} = 10,000 \times f_i^{neq}$ to make it of the order $O(1)$, where $f^{neq}_{i,\mathcal{NN}_{neq}}$ is the output of the $\mathcal{NN}_{neq}$, and f_i^{neq} is the predicted non-equilibrium distribution function used in Eq. (5).

To study the effect of scaling of f_i^{neq} on the predicted accuracy, we compare the results for two different cases, i.e., with and without scaling for f_i^{neq}, in Fig. 2. In both cases, we employ 5 hidden layers with 40 neurons per layer for $\mathcal{NN}_{eq}$ and $\mathcal{NN}_{neq}$. We see that the training loss for the case with scaling (Case A) is about three orders smaller than that without scaling (Case B). Furthermore, we also present the predicted residuals on a uniform lattice, i.e., $x \times y = 125 \times 100$, for two representative equations, i.e., f_2 and f_5, in Figs. 2(b)- 2(c). We observe that the computed residuals for Case B are about two orders larger than those in Case A, which suggests that the scaling of f_i^{neq} can significantly improve the predicted accuracy.



(a) (b) (c)

Figure 2: PINN-BGK for Kovasznay flows: (a) Time histories of the total loss function obtained by different cases. Case A: $f^{neq}_{i,\mathcal{NN}_{neq}} \approx 10000 f^{neq}_i$; Case B: $f^{neq}_{i,\mathcal{NN}_{neq}} \approx f^{neq}_i$. (b) Predicted residuals for f_2 and f_5 for Case A. (c) Predicted residuals for f_2 and f_5 for Case B.

We further compare the relative L_2 errors (err_ψ) for u, v and f^{neq}_i for these two cases, where err_ψ is defined as

$$err_\psi = \frac{\sqrt{\sum_{x,y} |\psi(x,y,t) - \psi_e(x,y,t)|^2}}{\sqrt{\sum_{x,y} |\psi_e(x,y,t)|^2}}, \ \psi = u, \ v, \ \text{and} \ f^{neq}_i, \tag{24}$$

and ψ_e is the exact solution here. In particular, the exact solutions for u and v are in Eq. (21), and the reference solution for f^{neq}_i is computed from Eq. (13). For brevity, we only present the relative errors for f^{neq}_2 and f^{neq}_5 as representatives since the results for the remaining f^{neq}_i are similar as these two. As displayed in Table 1, the predictions in Case A are all in good agreement with the exact solutions and much better than the results from Case B. These results again indicate that the scaling of f^{neq}_i is able to substantially improve the predicted accuracy.

We note that the errors for f^{neq}_i are about one order greater than those for u/v, which may be attributed to the roundoff in the computation of f^{neq}_i, i.e., $f^{neq}_i = f^{neq}_{i,\mathcal{NN}_{neq}}/10,000$. Another test case (Case C in Table 1) is then conducted here, i.e., we replace the $\mathcal{NN}_{eq}$ with the exact solutions for $\rho/u/v$, and only train $\mathcal{NN}_{neq}$ to satisfy the governing equation. Other parameters (e.g., size of $\mathcal{NN}_{eq}$, scaling of f^{neq}_i, etc.) as well as the optimization method are all the same as in Case A. As shown, the errors for f^{neq}_i in Case A are comparable to those in Case C, which validates the present assumption.

To accelerate the training process, we employ the transfer learning strategy, which has been widely used to enhance the convergence of training for diverse deep learning

Table 1: PINN-BGK for Kovasznay flows: Relative L_2 errors of velocity (u, v) and two representative non-equilibrium distribution functions. Case A: $f^{neq}_{i,\mathcal{NN}_{neq}} \approx 10,000 f^{neq}_i$; Case B: $f^{neq}_{i,\mathcal{NN}_{neq}} \approx f^{neq}_i$; Case C: $f^{neq}_{i,\mathcal{NN}_{neq}} \approx 10,000 f^{neq}_i$, ρ, u, and v are from the exact solutions.

	err_u	err_v	$err_{f_2^{neq}}$	$err_{f_5^{neq}}$
Case A	0.21%	0.60%	3.49%	4.56%
Case B	65.19%	58.66%	89.41%	82.14%
Case C	-	-	1.81%	2.23%

Table 2: PINN-BGK for Kovasznay flows: Relative L_2 errors and computational cost at different Reynolds numbers for different cases. L-BFGS-B (TL): L-BFGS-B optimization with transfer learning. L-BFGS-B only: L-BFGS-B optimization with random initialization. All the computations are performed on a workstation with one Quadro RTX 6000 GPU and Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz CPU.

Re	Case	err_u	err_v	Walltime (min)
20	L-BFGS-B (TL)	0.2%	0.75%	20.84
	Adam + L-BFGS-B	0.36%	0.98%	57.05
	L-BFGS-B only	0.40%	1.12%	32.15
40	L-BFGS-B (TL)	0.22%	0.89%	36.14
	Adam + L-BFGS-B	0.26%	0.91%	63.65
	L-BFGS-B only	2.63%	108.78%	31.47
60	L-BFGS-B (TL)	0.43%	1.07%	28.18
	Adam + L-BFGS-B	0.27%	1.06%	75.87
	L-BFGS-B only	3.08%	119.43%	32.76

problems [64, 65]. In particular, we employ the PINN-BGK to model the flows for three different Reynolds numbers, i.e., Re = 20, 40, and 60. The weights/biases for each case are initialized with those for the case Re = 10 on its completion of training. Note that only the L-BFGS-B algorithm is used for optimizing the loss function in the transfer learning cases. Here, we also present the results for using the Adam plus L-BFGS-B as used in the previous case for comparison. As we can see, comparable accuracy can be obtained by using the L-BFGS-B with transfer learning and the standard training process used in this work, i.e., first Adam and then L-BFGS-B. However, we can obtain up to a three-fold speedup with transfer learning as compared to the latter case.

It is worth mentioning that the L-BFGS-B algorithm is based on the Newton's method, hence the computational accuracy strongly depends on the initial guess. In other words, it is easy for the L-BFGS-B to converge to a local optimal without good initialization of the weights/biases in DNNs. For demonstration, we then present the results for using the L-BFGS-B with random initialization for the DNNs (which is referred to as L-BFGS-B only in Table 2). As shown, the computational time for the L-BFGS-B (TL) and L-BFGS-B only is similar for all the three test cases. However, the L-BFGS-B with transfer learning can achieve much better accuracy than the L-BFGS-B with the random initialization. In particular, the predicted accuracy for the L-BFGS-B (TL) is about two orders better than the L-BFGS-B only for the vertical velocity in the last two cases, i.e., Re = 40, and 60.

Finally, (1) we present a systematic study on the effects of architectures of DNNs and the number of residual/boundary points on the predicted accuracy in Appendix B, (2) we note that the multiscale analysis in Sec. 2.2 only holds for continuous flows in which τ is a small parameter. In the present study, we assume f_i^{neq} has a similar magnitude as $O(\tau)$ in all test cases, and scale $f_{i,\mathcal{NN}_{neq}}^{neq}$ to $O(1)$ based on τ used in the corresponding case to improve the predicted accuracy, and (3) we employ the same scaling factor for $f_{i,\mathcal{NN}_{neq}}^{neq}$ in the entire computational domain for all test cases considered in the present study. For cases where the magnitude of f_i^{neq} varies in different part of the domain, a promising way to obtain the proper scaling factor is to employ a space-dependent weight for $f_{i,\mathcal{NN}_{neq}}^{neq}$ whose value can be optimized in a similar way for optimizing the hyperparameters in DNNs. Interested readers can refer to the self-adaptive PINNs [66] for more details.

3.2. Taylor-Green flow

We proceed to model a time-dependent flow problem, i.e., the Taylor-Green flow, using the PINN-BGK. The computational domain is defined as $x \in [-\pi, \pi]$, $y \in$

$[-\pi, \pi]$. The exact solution is set as follows

$$\begin{aligned} u(x,\ y,\ t) &= -\cos(x)\sin(y)e^{-2t\nu}, \\ v(x,\ y,\ t) &= \sin(x)\cos(y)e^{-2t\nu}, \\ p(x,\ y,\ t) &= -\frac{1}{4}[\cos(2x)+\cos(2y)]e^{-4t\nu}, \end{aligned} \tag{25}$$

where u, and v are the velocity components in the horizontal and vertical directions, respectively, and p denotes the fluid pressure. Finally, all the initial and boundary conditions are derived from Eq. (25).

In simulations, we set $RT = 100$, $\nu = 0.01$, and $t \in [0, 10]$. The D2Q9 velocity model is also employed here. As for the training data, 300 random points are employed on each boundary while $160,000$ random points in the spatial-temporal domain are used to compute the residuals of Eq. (5). Both $\mathcal{NN}_{eq}$ and $\mathcal{NN}_{neq}$ contain 6 hidden layers with 80 neurons per layer. Similar as the first test case, two optimization algorithms, i.e., Adam and L-BFGS-B are used. Specifically, we first train the DNNs using the Adam optimizer until the loss is less than 1.0×10^{-4} with an initial learning rate of 5×10^{-4}, and then we switch to the L-BFGS-B optimization until convergence.

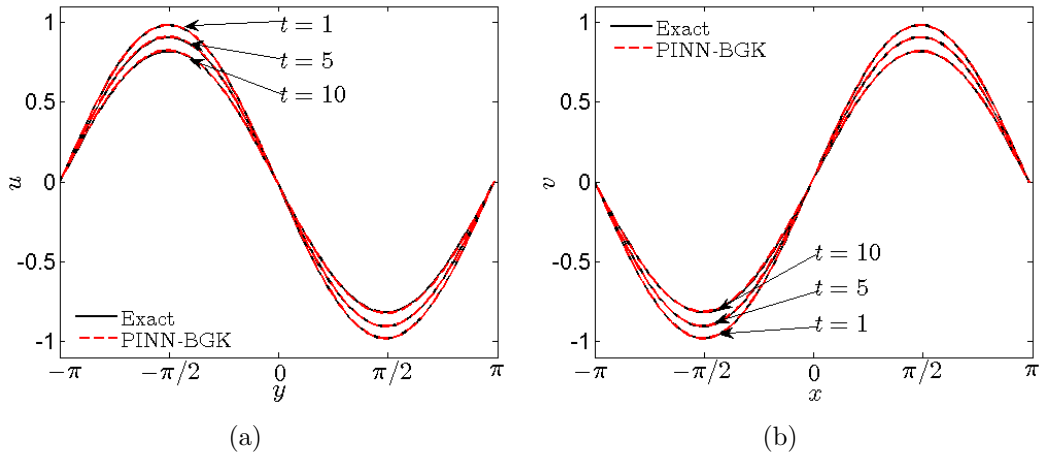


Figure 3: PINN-BGK for Taylor-Green flows: Predicted velocity profiles. (a) u component for $x = 0$ at $t = 1$, 5, and 10. (b) v component for for $y = 0$ at $t = 1$, 5, and 10.

The predicted velocity profiles at three representative times, i.e., $t = 1$, 5 and 10, are illustrated in Fig. 3. We see that the predicted velocities agree well with the exact solutions. Furthermore, we present the relative L_2 errors for the predicted $u/v/p$ in

more detail in Table 3, which demonstrates the good accuracy of the PINN-BGK for modeling unsteady flows.

Table 3: PINNs for Taylor-Green flows: Relative L_2 errors of pressure (p) and velocity (u, v) at different times.

Time	$t = 0.0$	$t = 2.0$	$t = 4.0$	$t = 6.0$	$t = 8.0$	$t = 10.0$
err_p	0.00092%	0.0011%	0.0012%	0.0015%	0.002%	0.0037%
err_u	0.0058%	0.16%	0.24%	0.30%	0.35%	0.48%
err_v	0.0051%	0.16%	0.26%	0.35%	0.44%	0.55%

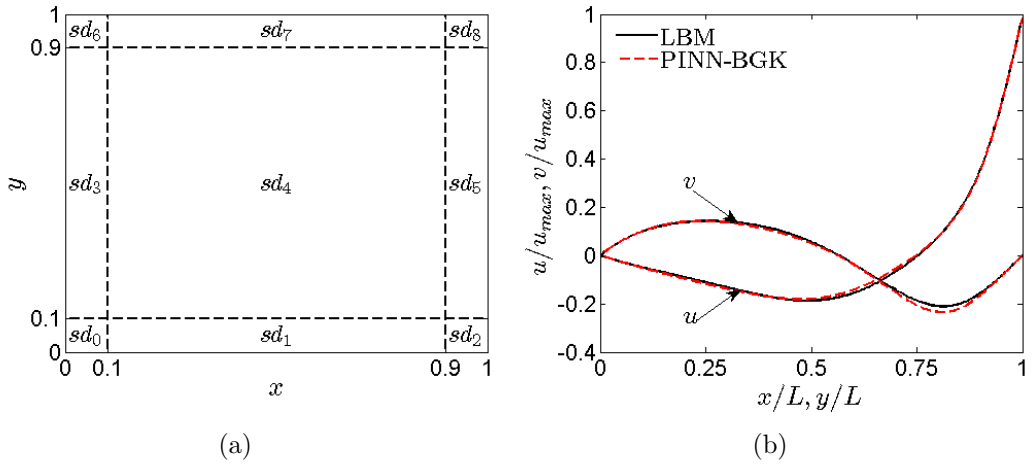
3.3. Regularized Cavity flow

Here we model the cavity flow using the PINN-BGK. The computational domain is set as $x \in [0,\ 1]$, and $y \in [0,\ 1]$. The upper boundary is moving from the left to right with a velocity u_w defined as

$$u_w = u_{max}\left(1 - \frac{\cosh[r(x - L/2)]}{\cosh(rL/2)}\right), \tag{26}$$

where $r = 10$ is a constant, $u_{max} = 1.0$ is the maximum velocity at the top wall, and $L = 1$ is the characteristic length of the cavity. Note that the velocity is continuous at the two upper corners, which is less physical but can alleviate the effect of singularity on the computational accuracy. A similar strategy has also been employed in [67, 68].

In simulations, $Re = Lu_{max}/\nu = 100$, where ν is the kinematic viscosity. We employ 4 hidden layers with 50 neurons per layer for $\mathcal{NN}_{eq}$, and 8 hidden layers with 60 neurons per layer for $\mathcal{NN}_{neq}$. As for the training data, 257 uniformly distributed points on each boundary are used to provide Dirichlet boundary conditions, and $45,000$ random points are employed to compute the residuals of the equations. To accurately predict the boundary layers, we divide the whole computational domain into nine subdomains ($sd_0,\ sd_1, \cdots, sd_8$) as shown in Fig. 4(a), which is similar as the nonuniform meshes used in the conventional numerical methods [69, 70]. In each subdomain, 5000 random points are employed to evaluate the residuals of the governing equations. For the optimization, we first train the DNNs using the Adam optimization with an initial learning rate 1.0×10^{-4} for $50,000$ iterations, and then train the DNNs for 50,000 steps using Adam with a smaller learning rate 1.0×10^{-5}. Finally, we switch to the L-BFGS-B until convergence.



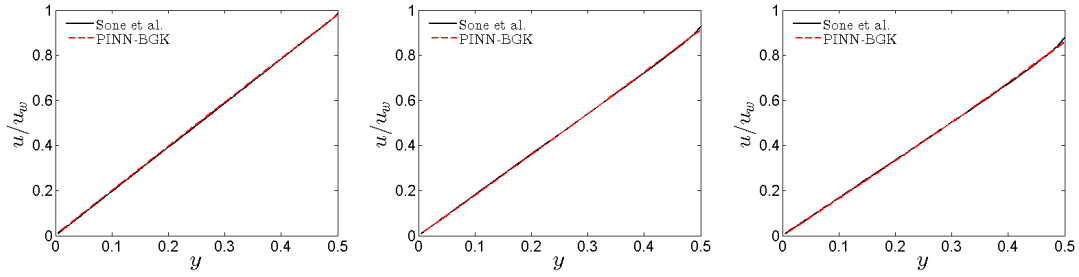
(a) (b)

Figure 4: PINN-BGK for cavity flows: (a) Schematic of the subdomains for cavity flow. (b) Predicted u- and v-velocity profiles along $y = 0.5$ and $x = 0.5$, respectively.

The predicted velocity profiles along the central lines ($x = 0.5$ and $y = 0.5$) are illustrated in Fig. 4(b). In addition, the results from the lattice Boltzmann method (LBM) [55] with 256×256 uniform grid are also displayed in Fig. 4 as reference. As we can see, the predictions using the PINN-BGK agree well with the results from the LBM.

3.4. Micro Couette flow

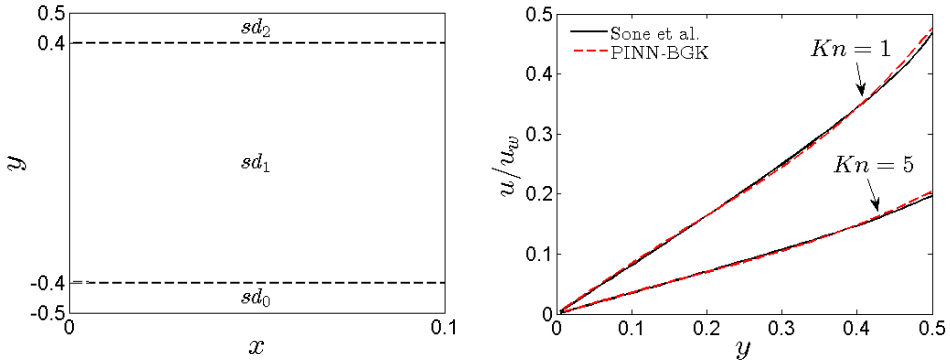
The PINN-BGK is now applied to simulate the micro Couette flow between two parallel plates. The computational domain is set as $0 \leq x \leq 0.1$, $-0.5 \leq y \leq 0.5$. The top (i.e., $y = 0.5$) and bottom walls (i.e., $y = -0.5$) move with a constant velocity u_w in an opposite direction. For rarefied flows, the Knudsen number Kn is defined as $Kn = \lambda/L$, where λ is the mean free path, and L is the characteristic length of the system. Note that λ is related to the relaxation time τ in the Boltzmann-BGK equation as $\lambda = \tau\sqrt{\pi RT/2}$ [26]. Based on Kn, gas flows can be divided into four regimes: the hydrodynamic regime ($Kn < 0.001$), the slip regime ($0.001 < Kn < 0.1$), the transition regime ($0.1 < Kn < 10$), and the free molecular flow regime ($Kn > 10$) [10]. Here, we employ the PINN-BGK to model the flows in two regimes, i.e., the slip regime ($0.001 < Kn < 0.1$), and the transition regime ($0.1 < Kn < 10$). In addition, the two-dimensional-sixteen-velocity (i.e., D2Q16) model is employed for the slip regime while a two-dimensional 28×28 velocity model (28×28-DVB) is used for the flows in transition regime. Interested readers can refer to [54, 71] for more details on these two discrete velocity models.



(a) $Kn = 0.01$ (b) $Kn = 0.05$ (c) $Kn = 0.09$

Figure 5: PINN-BGK for micro-Couette flows: Predicted horizontal velocity profiles in the slip regime using the D2Q16 model.

We first apply the PINN-BGK to simulate the Couette flow in the slip regime. In particular, three different cases, i.e., $Kn = 0.01,\ 0.05$, and 0.09, are considered. For all the test cases, we set $u_w = 0.1$, $RT = 1.0$. In addition, both $\mathcal{NN}_{eq}$ and $\mathcal{NN}_{neq}$ have 6 hidden layers with 30 neurons per layer. In our simulations, 10 uniformly distributed points are used on each boundary to provide the Dirichlet boundary conditions and $10,000$ random points are employed to compute the residuals of the equations. Similarly, both the Adam and L-BFGS-B are used for optimization as follows: the Adam optimizer with an initial learning rate 10^{-3} is first employed until the loss is less than 10^{-3}, and then L-BFGS-B is applied until convergence.



(a) (b)

Figure 6: PINN-BGK for micro-Couette flows in transition regime: (a) Schematic of subdomains. (b) Predicted horizontal velocity profiles.

The horizontal velocity profiles predicted by PINN-BGK are displayed in Fig. 5. In addition, the solutions to the linearized Boltzmann equation [72] are presented

as reference. We observe velocity slip at the upper boundary in all test cases, which is well captured by the PINN-BGK. Specifically, the relative L_2 errors between the PINN-BGK and the reference solutions are 0.55%, 0.59%, and 0.85% for $Kn = 0.01,\ 0.05$, and 0.09, respectively, suggesting that the PINN-BGK is capable of providing accurate predictions for the flows in the slip regime.

We now simulate the Couette flow in the transition regime. Here two different cases, i.e., $Kn = 1$ and 5 are considered. In both cases, $\mathcal{NN}_{eq}$ has 2 hidden layers with 30 neurons per layer, and $\mathcal{NN}_{neq}$ has 2 hidden layers with 40 neurons per layer. Here, the 28×28 discrete velocity model is employed, in which RT is set to be $1/2$. Note that the model is more computationally expensive than the previous cases, so we utilize the minibatch approach [73, 74] with a batch size of 800 for both cases to accelerate the training. In addition, 10 uniformly distributed points on each boundary are employed to enforce the boundary conditions. To accurately capture the slip boundary layers near the upper and bottom walls, we divide the computational domain into 3 subdomains as shown in Fig. 6(a). In sd_0 and sd_2, 800 uniformly distributed residual points are employed, while $1,600$ uniformly distributed points are employed to compute the residuals of equations in sd_1. The Adam optimizer with an initial learning rate 10^{-3} is used for optimization. The predicted horizontal velocity profiles from the PINN-BGK for both cases are depicted in Fig. 6(b); we observe that they agree well with the reference solutions from the linearized Boltzmann equation in [72].

4. Inverse problems

In this section, we test the performance of the PINN-BGK for inverse problems. Specifically, we focus on rarefied flows with unknown velocity boundary conditions. We assume that we only have partial measurements on the velocity inside the computational domain. The objective is to infer the entire flow field based on a limited number of scattered observations on velocity with the aid of the Boltzmann-BGK equation.

4.1. Micro Couette flows

We first use the PINN-BGK to infer the velocity field for the micro Couette flow based on a small set of measurements on velocity inside the computational domain. The computational domain is set as $0 \leq x \leq 0.1$, $-0.5 \leq y \leq 0.5$. The top and bottom walls move with a constant velocity (i.e., $u_w = 0.1$) in an opposite direction, which is the same as the case in Sec. 3.4. Three different cases, i.e., $Kn = 1$, 5, and 10, are considered. We therefore employ the 28×28-discrete velocity model as in

Sec. 3.4 [54, 71]. For all the test cases, RT is set to be 1/2. We then assume that we have 10 sensors for the velocity, which are randomly located inside the computational domain, as shown in Fig. 7(a). In addition, we utilize 4 hidden layers with 30 neurons per layer for both $\mathcal{NN}_{eq}$ and $\mathcal{NN}_{neq}$, and $5,000$ ($N_e = 5,000$) residual points are randomly selected inside the computational domain. To train the NNs, the Adam optimizer with an initial learning rate 10^{-3} is first used until the loss is less than 10^{-4}, and then the LBFGS-B optimizer is employed until convergence.

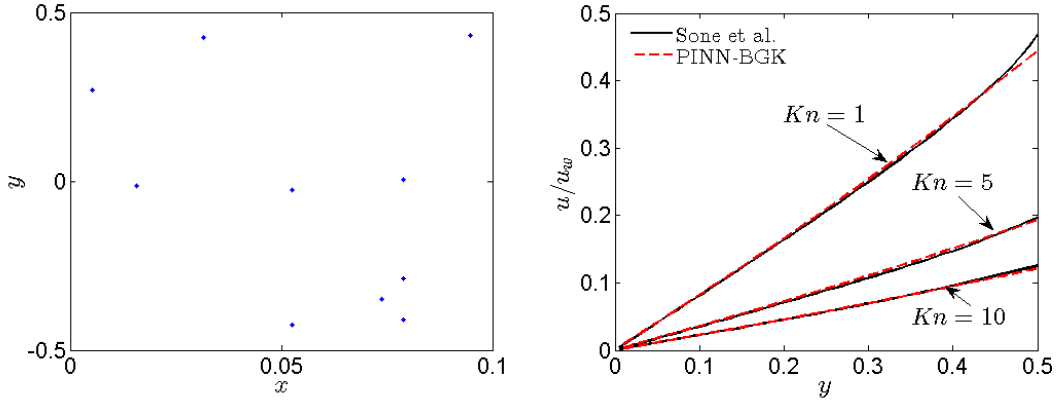


Figure 7: PINN-BGK for inverse problems in micro Couette flow: (a) Locations of the sensors for velocity. (b) Predicted velocity profiles along the vertical central lines for different Knudsen numbers.

The predicted velocity profiles using the PINN-BGK are displayed in Fig. 7(b), and the solutions from the linearized Boltzmann equation [72] are presented as reference. As shown, the predictions from the PINN-BGK agree well with the reference solutions for all cases. Furthermore, the relative L_2 errors between the PINN-BGK and the reference are 2.04%, 2.04%, and 1.44% for $Kn = 1$, 5, and $Kn = 10$, respectively, which demonstrates that the PINN-BGK is capable of inferring the velocity field based on partial interior scattered measurements on velocity without the explicit knowledge of boundary conditions.

In addition to estimate the velocity field based on partial interior observations on the velocity, we can also employ the PINN-BGK to predict the slip velocity at the boundary for different Knudsen numbers. Here, we use the PINN-BGK to infer the fluid velocity at the upper boundary for Kn ranging from 0.01 to 10. As shown in Fig. 8, the predicted fluid velocities at the upper boundary u_{fluid} are in good agreement with the reference solutions in [72], which clearly demonstrates the capability of the

PINN-BGK for modeling multiscale flows. Also, we would like to point out that the predicted velocities at the boundary can serve as Dirichlet boundary conditions for other numerical methods, which need explicit boundary conditions to solve the Boltzmann-BGK equations, such as the discrete unified gas kinetic scheme [26], the unified gas kinetic scheme [75], etc. The obtained results from numerical solvers can be employed to validate the flow field predicted by PINNs. This scenario may happen in experiments in which we only have a few interior sensors for velocity, and we have no information on the velocity or temperature at the wall.

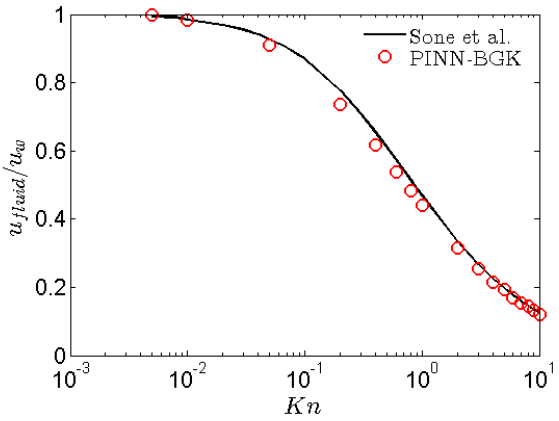


Figure 8: PINN-BGK for inverse problems in micro Couette flow: normalized fluid velocity at the upper boundary u_{fluid} for different Kn. u_w: moving velocity of upper wall.

4.2. Micro cavity flows

We now employ the PINN-BGK to infer the velocity field for the micro cavity flow based on partial interior measurements on velocity, in which the computational domain is set as $0 \leq x \leq 1$, $0 \leq y \leq 1$. The top wall of the cavity moves along the x direction with the velocity defined in Eq. (26) in which $u_{max} = 0.15$ and $r = 10$ while the other walls are stationary. Again, we assume that only a few sensors for the fluid velocity inside the domain are available while the boundary conditions are unknown.

Here we test three different cases, i.e., $Kn = 0.1$, 1, and 10. In addition, the discrete unified gas kinetic scheme (DUGKS) [26], which is a finite-volume solver for the Boltzmann-BGK equation, is used to generate training data as well as provide reference solutions. In the DUGKS simulations, the computational domain is discretized into 60×60 uniform grids. Similar as in [26, 75], the Newton-Cotes quadrature is used to discrete the velocity space, and an 81×81 uniformly distributed velocities in $[-4\sqrt{RT}, 4\sqrt{RT}] \times [-4\sqrt{RT}, 4\sqrt{RT}]$ are employed. RT is set to be $1/2$, ν and τ

can then be computed using Kn and RT. In the PINN-BGK, we employ the same discrete velocity model as in DUGKS. In addition, we first assume that we have 100 sensors for the velocity, which are randomly distributed inside the computational domain, see Fig. 9(a). Two hidden layers with 30 neurons per layer are used for $\mathcal{NN}_{eq}$ while 6 hidden layers with 30 neurons per layer are employed for $\mathcal{NN}_{neq}$. Also, $12,000$ random points are employed to compute the residuals for the equation in the loss function. Similarly, the minibatch approach with a batch size of 128 is employed to accelerate the training.

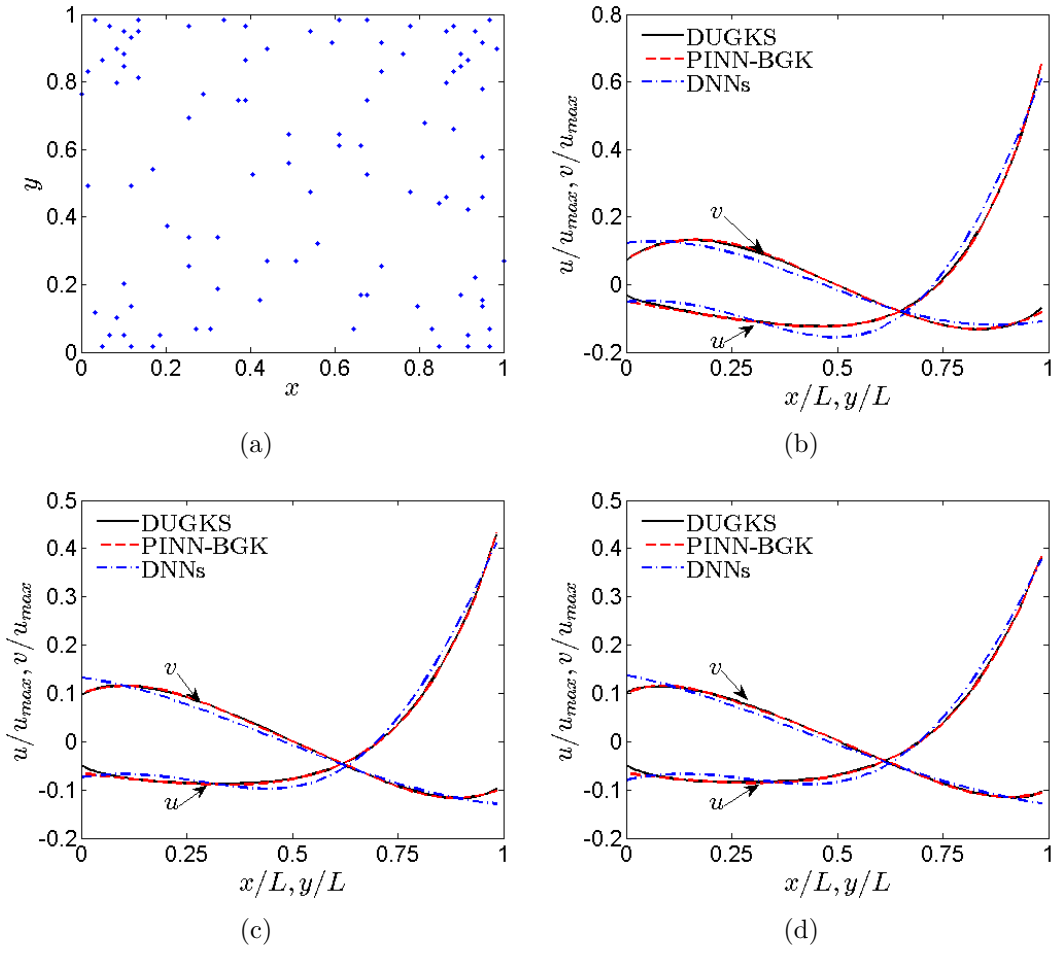


Figure 9: PINN-BGK for inverse problems in micro cavity flows with random training data: (a) Locations of sensors for $\boldsymbol{u}$; (b - d): Predicted velocity profiles across the cavity center for (b) $Kn = 0.1$, (c) $Kn = 1.0$, (d) $Kn = 10$.

Table 4: PINN-BGK for inverse problems in micro cavity flows with random training data: Relative L_2 relative errors for velocity (u, v) at different Knudsen numbers. DNNs: results from regression without the constraint of PDEs. Reference solutions: DUGKS with 60×60 uniform grids.

Kn	err_u		err_v	
	PINN-BGK	DNNs	PINN-BGK	DNNs
0.1	3.36%	18.68%	4.27%	19.65%
1	3.39%	16.02%	3.36%	15.71%
10	3.66%	15.13%	2.99%	18.81%

The predicted velocity profiles from the PINN-BGK across the cavity center for different Knudsen numbers are presented in Figs. 9(b)-9(d). In addition, the results from DUGKS are also presented as reference solutions in Figs. 9(b)-9(d). We can see that the PINN-BGK predictions are in good agreement with the results from the DUGKS at the three Knudsen numbers, demonstrating the capability of the PINN-BGK to model multiscale transitional flows with unknown velocity boundary conditions.

To further demonstrate the effectiveness of the PINN-BGK, we also present the results for regression (denoted as DNNs in Figs. 9(b)-9(d)), i.e., we employ $\mathcal{NN}_{eq}$ to predict the velocity profiles based on the same measurements on velocity without the constraint of the governing equation. We observe that the predictions from the PINN-BGK are better than those from DNN regression, especially for areas near the boundaries. We further compute the relative L_2 errors for the PINN-BGK and regression, in which the results from the DUGKS are used as reference solutions. As shown in Table 4, the errors for the PINN-BGK are less than 5% in all cases, suggesting that the PINN-BGK can accurately reconstruct the velocity field even without the use of boundary conditions. In addition, the errors for regression are larger than 15% in all cases, which means that enforcing the constraint of the governing equations in the loss function helps greatly to improve the accuracy of the predictions.

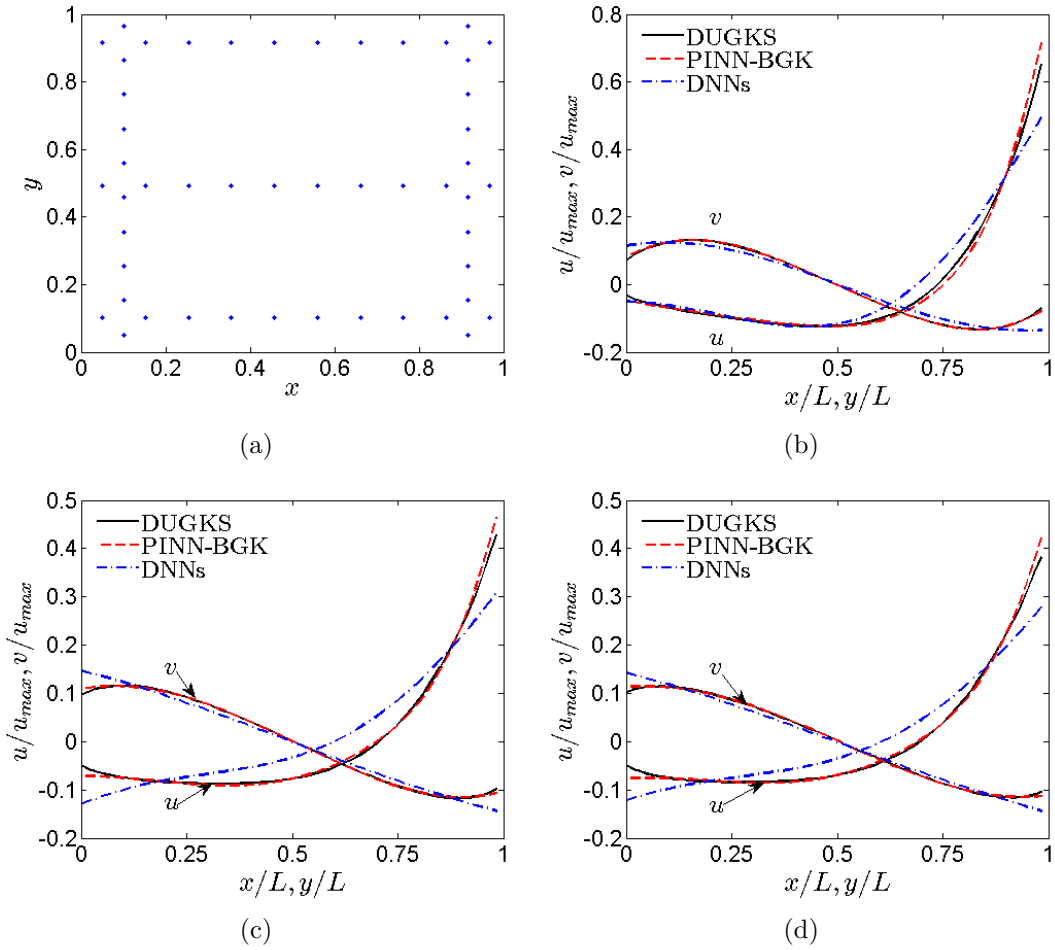


Figure 10: PINN-BGK for inverse problems in micro cavity flows with uniform training data (Case I): (a) Locations of sensors for $\boldsymbol{u}$. The 50 training samples are uniformly distributed along 5 lines. (b - d): Predicted velocity profiles across the cavity center for (b) $Kn = 0.1$, (c) $Kn = 1.0$, (d) $Kn = 10$.

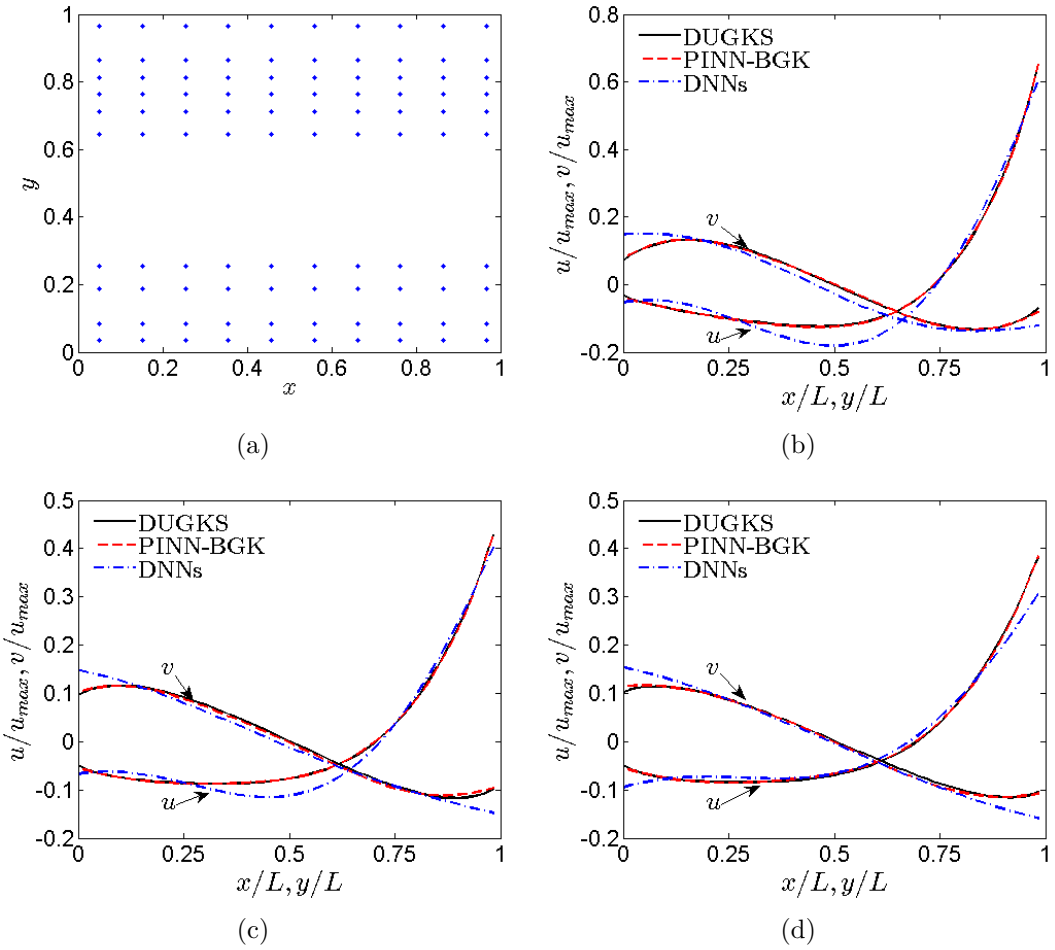


Figure 11: PINN-BGK for inverse problems in micro cavity flows with uniform training data (Case II): (a) Locations of sensors for $\boldsymbol{u}$. The 100 training samples are uniformly distributed along 10 lines. (b - d): Predicted velocity profiles across the cavity center for (b) $Kn = 0.1$, (c) $Kn = 1.0$, (d) $Kn = 10$.

Due to the fact that the measurements at arbitrary locations may be difficult to obtain in practical applications, next we first test the case in which we assume that 50 measurements on $\boldsymbol{u}$ are uniformly distributed along 5 lines (Fig. 10(a), Case I). In simulations of the present case, the parameters (e.g., the architectures of DNNs, optimizer, etc.) are kept the same as those in the previous one. The predicted velocity profiles for $Kn = 0.1$, 1, and 10 across the cavity center are presented in Fig. 10. As observed, the results predicted by the PINN-BGK here are not as accurate as those in Fig. 9, especially for areas near the boundaries. Specifically, the

Table 5: PINN-BGK for inverse problems in micro cavity flows with uniformly sampled training data: Relative L_2 errors for velocity $(u,\ v)$ at different Knudsen numbers. DNNs: results from regression without the constraint of PDEs. Reference solutions: DUGKS with 60×60 uniform grids. Case I: 50 measurements on $\boldsymbol{u}$ are uniformly distributed along 5 lines. Case II: 100 measurements on $\boldsymbol{u}$ are uniformly distributed along 10 lines.

Case	Kn	err_u		err_v	
		PINN-BGK	DNNs	PINN-BGK	DNNs
Case I	0.1	9.37%	25.45%	36.58%	21.52%
	1	8.51%	35.29%	21.39%	23.07%
	10	8.56%	31.87%	17.46%	20.62%
Case II	0.1	3.26%	20.22%	5.0%	19.47%
	1	3.5%	17.99%	4.49%	15.99%
	10	3.62%	29.10%	4.04%	21.69%

relative L_2 errors for velocity (Case I in Table 5) are much larger than the results from PINN-BGK in Table 4. We then increase the number of training samples to 100, which are uniformly distributed along 10 lines (Fig. 11(a), Case II). Specifically, more measurements are added near the vortex zone in the present case. Similarly, we employ the same parameters (e.g., the architectures of DNNs, optimizer, etc.) here as in the previous cases. As displayed in Fig. 11, better accuracy is then achieved, which can also be seen in Table 5.

Based on the results in Figs. 9-11, we can further conclude that: (1) the predicted accuracy can be enhanced by increasing the number of training data as we compare the predictions in Figs. 9 and 11 with the results in Fig. 10, (2) the predictions of PINN-BGK are more accurate than those from regression, which again demonstrates the physics-informed constraint is able to improve the predicted accuracy, and (3) we can obtain better accuracy as we add more measurements near the vortex zone as well as the boundaries when comparing the results in Fig. 11 with those in Fig. 10.

Note that the boundary condition for the upper wall in Eq. (26) is also used in the DUGKS simulations here, aiming to reduce the effect of the singularities at the two upper corners. This boundary condition may be less physical than the constant velocity. However, the focus here is to test the performance of the PINNs for reconstructing the velocity field given partial interior measurements with unknown boundary conditions. We therefore utilize the simplified boundary condition here and will leave investigations on the effect of different boundary conditions on the computational accuracy in the future.

5. Summary

The Boltzmann equation with the Bhatnagar-Gross-Krook collision model (Boltzmann-BGK equation) is a well known model for describing multiscale flows from the hydrodynamic limit to free molecular flows. The recently proposed physics-informed neural networks (PINNs) have shown expressive power for solving both forward and inverse PDE problems in fluid mechanics and beyond. By leveraging the merits of the Boltzmann-BGK equation and PINNs, we have solved forward and inverse problems in multiscale flows via Boltzmann-BGK formulation using PINNs (PINN-BGK). In particular, the proposed PINN-BGK contains three sub-networks, i.e., one for the equilibrium distribution function, a second one to approximate the non-equilibrium distribution function, and a third one to encode the Boltzmann-BGK equation as well as the corresponding boundary/initial conditions. By minimizing the residuals of all governing equations and the mismatch between the predicted and provided boundary/initial conditions, we can then solve both forward and inverse problems in multiscale flows with the same formulation and the same code. We note that the PINN-BGK may not be as efficient as other kinetic-based methods (e.g., LBM) for forward problems, especially for flows with low Knudsen numbers. The attractive advantages for PINN-BGK are: (1) It is very efficient in solving inverse problems, even ill-posed inverse problems. (2) It is easy to implement and thus can serve as an educational tool to be used in the classroom as well as a research tool for solving problems in computational science and engineering as mentioned in [33, 35]. (3) It is an unified method, which can be used for both forward and inverse problems in a wide range of flow regimes as shown in the present study.

We first test the PINN-BGK for solving forward problems in multiscale flows. Specifically, the PINN-BGK is applied to model both continuous (e.g., Taylor-Green flows, cavity flows, etc.) and rarefied flows with Knudsen number ranging from 0.01 to 5. The results demonstrate that the PINN-BGK is capable of approximating accurately the solution to the Boltzmann-BGK equation given boundary/initial conditions. We then apply the PINN-BGK to solve inverse problems, i.e., inferring the velocity field based on a limited number of scattered interior observations on the velocity without any information on the boundary conditions. In particular, we test the performance of the PINN-BGK for micro Couette and cavity flows with a Knudsen number ranging from 0.01 to 10. Our results demonstrate that the PINN-BGK is capable of reconstructing the velocity field with good accuracy based on a small set of interior measurements on velocities. This is quite promising for multiscale flows within complicated geometries in which accurate boundary conditions are difficult to obtain, such as shale gas flow in pores at scales varying from nm to mm. Finally, we demonstrate that we can achieve three-fold speedup by using transfer learning

as compared to the standard training, i.e., Adam optimizer plus L-BFGS-B, for the specific Kovasznay flow case we studied; Generally, a larger neural network as well as more training points should be employed for a higher dimensional problem, suggesting that more hyperparameters need to be optimized, and that more terms need to be computed in the loss function. We therefore expect even greater speedup for three-dimensional time-dependent flows.

Acknowledgement

Q. Lou would like to acknowledge the support of the National Natural Science Foundation of China (Grant No.51976128) and the Natural Science Foundation of Shanghai (Grant No. 19ZR1435700). X. Meng and G. E. Karniadakis would like to acknowledge the support of PhILMS grant DE-SC0019453. The authors would like to thank Dr. C. Zhang for providing the DUGKS results for inverse problems.

Appendix A. On the training of PINNs

Here we present more details on how to use the PINN-BGK to solve the Boltzmann-BGK equation. Let $\mathcal{NN}_{eq}^{M}$ (Fig. 1) be an M-layer neural network and the $\mathcal{NN}_{neq}^{N}$ (Fig. 1) be an N-layer neural network. Then f_i^{eq} is approximated by the output of the neural network $\mathcal{NN}_{eq}^{M}$ as follows,

$$\begin{aligned}
\text{input layer}: &\quad \mathcal{NN}_{eq}^{0}(\boldsymbol{x},t) = (\boldsymbol{x},t),\\
\text{hidden layer}: &\quad \mathcal{NN}_{eq}^{l}(\boldsymbol{x},t) = \sigma(\boldsymbol{W}_{eq}^{l}\mathcal{NN}_{eq}^{l-1}(\boldsymbol{x},t) + \boldsymbol{b}_{eq}^{l}) \quad \text{for} \quad 1 \le l \le M-1,\\
\text{output layer}: &\quad \mathcal{NN}_{eq}^{M}(\boldsymbol{x},t) = \boldsymbol{W}_{eq}^{M}\mathcal{NN}_{eq}^{M-1}(\boldsymbol{x},t) + \boldsymbol{b}_{eq}^{M}),
\end{aligned} \tag{A.1}$$

and f_i^{neq} is approximated by the output of the neural network $\mathcal{NN}_{neq}^{N}$ as follows,

$$\begin{aligned}
\text{input layer}: &\quad \mathcal{NN}_{neq}^{0}(\boldsymbol{x},t) = (\boldsymbol{x},t),\\
\text{hidden layer}: &\quad \mathcal{NN}_{neq}^{l}(\boldsymbol{x},t) = \sigma(\boldsymbol{W}_{neq}^{l}\mathcal{NN}_{neq}^{l-1}(\boldsymbol{x},t) + \boldsymbol{b}_{neq}^{l}) \quad \text{for} \quad 1 \le l \le N-1,\\
\text{output layer}: &\quad \mathcal{NN}_{neq}^{N}(\boldsymbol{x},t) = \boldsymbol{W}_{neq}^{N}\mathcal{NN}_{neq}^{N-1}(\boldsymbol{x},t) + \boldsymbol{b}_{neq}^{N}),
\end{aligned} \tag{A.2}$$

where σ is a nonlinear activation function, which is set as the hyperbolic tangent function in the present work; the superscript l demotes the lth layer, $\boldsymbol{W}_{eq}$ and $\boldsymbol{W}_{neq}$ are the weight matrixes for $\mathcal{NN}_{eq}^{M}$ and $\mathcal{NN}_{neq}^{N}$, respectively, $\boldsymbol{b}_{eq}$ and $\boldsymbol{b}_{neq}$ are bias

vectors for $\mathcal{NN}_{eq}^{M}$ and $\mathcal{NN}_{neq}^{N}$, respectively. To this end, we can express the distribution function f_i as a function of $(\boldsymbol{x}, t)$ as well as the weights and biases in these two neural networks. Meanwhile, we can tune the hyperparameter (i.e., weights/biases) in the neural networks to fit a wide range of f_i by minimizing a certain loss function using the optimization algorithms.

In PINNs, the DNNs, which are universal approximators for any continuous functions [76], are imposed to satisfy the PDE as well as certain boundary/initial conditions. This is implemented by minimizing the mean square errors (MSE) of the residuals for a PDE and the mismatches between the predicted and exact boundary/initial conditions using the optimization algorithms (e.g., Adam [62], L-BFGS-B [63]). In practice, the MSE is evaluated on scattered points which can be either randomly or specially selected. Generally, randomly selected training points work well for cases with smooth solutions, while for solutions with sharp gradients, more points near the sharpness are required to fully resolve the solution [35]. Specifically, the temporal and spatial derivatives in PDEs are computed using automatic differentiation [34], in which the chain rule is applied for compositions of functions in neural networks. Thus, no meshes are needed for the computation of the differential operators in PDEs.

Appendix B. Systematic study on parameters in PINNs

In this section, we conduct a systematic study on the effect of different architectures and the number of residual/boundary points on the computational cost as well as on accuracy using the example of Kovasznay flow in Sec. 3.1. All computations in this section are performed on a workstation with one GeForce RTX 2080 GPU and Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz CPU.

We first test the effect of the architectures for DNNs. The residual/boundary points in all test cases are kept the same as in Sec. 3.1. Although the DNNs have been justified theoretically to be a universal approximator for any continuous function [76], the optimal architecture for approximating a specified function is problem-dependent and also empirical. Based on our experience, a DNN with 2 to 6 hidden layers with 20-60 neurons per layer is able to approximate most of the continuous functions well. We therefore demonstrate the performance of 12 DNNs with different architectures in Table B.6. As shown, the DNNs with 4 hidden layers and 20-60 neurons per layer have the most accurate predictions among all the tested architectures. In addition, the computational cost for DNNs with 4 hidden layers and different number of neurons per layers is comparable, which is around 60 minutes for the three test cases. Generally, a deep and wide DNN is more expressive than a shallow and narrow

Table B.6: PINN-BGK for Kovasznay flows: Relative L_2 errors of velocity (u, v) with different architectures of neural networks. The architecture is presented in the form of depth $\times$ width, i.e., number of hidden layers $\times$ number of neurons per layer.

Architecture	err_u	err_v	Walltime (min)
2×20	0.62%	3.57%	54.42
2×40	1.12%	6.02%	56.02
2×60	0.43%	1.62%	65.18
4×20	0.44%	1.86%	59.11
4×40	0.58%	1.23%	59.38
4×60	0.38%	1.16%	60.35
6×20	0.49%	3.38%	49.97
6×40	0.48%	1.46%	55.23
6×60	0.32%	2.44%	68.30
8×20	0.79%	3.15%	81.39
8×40	0.47%	1.76%	67.81
8×60	0.23%	1.24%	78.43

one, but it will requite higher computational cost and may also cause problems of overfitting. A possible solution to this problem is the utilization of neural architecture search (NAS) [77], which is able to obtain a neural network with optimal architecture in a specified search space. We will leave investigations on this advanced technique for future work.

We proceed to test the effect of the number of residual points. Similarly, here the architecture of DNN as well as the boundary points are kept the same as in Sec. 3.1. In all test cases, the residual points are randomly selected in the entire domain. As demonstrated in Table B.7, the computational errors are similar as the number of residual points is larger than 1,000. Meanwhile, the computational cost will increase with the number of residual points.

We now test the effect of number of boundary points. The architecture of DNN and the residual points are kept the same as in Sec. 3.1 here. In all test cases, the equidistant boundary points are employed on each boundary. As displayed in Table B.8, the computational errors are similar as the number of boundary points is larger than 200, suggesting that 200 points are fine enough to resolve the boundary conditions. In addition, the computational cost is similar for all test cases, which is reasonable since the number of boundary points is much smaller than the residual points.

We note that the single-precision floating point format is utilized in the com-

Table B.7: PINN-BGK for Kovasznay flows: Relative L_2 errors of velocity (u, v) using different number of residual points. The computational errors and time are the averages over 5 runs with different random residual points.

N_{rp}	err_u	err_v	Walltime (min)
100	0.55%	3.54%	40.55
500	1.18%	1.73%	51.29
1000	0.28%	1.37%	46.04
5000	0.3%	1.82%	49.55
10000	0.28%	1.58%	56.07
15000	0.32%	1.37%	68.43
20000	0.31%	1.01%	63.92

Table B.8: PINN-BGK for Kovasznay flows: Relative L_2 errors of velocity (u, v) using different numbers of boundary points.

N_{bp}	err_u	err_v	Walltime (min)
10	0.69%	4.01%	45.56
100	0.43%	2.73%	55.56
200	0.28%	1.8%	56.78
300	0.3%	1.82%	49.55
400	0.24%	1.34%	48.90
500	0.28%	1.83%	55.82
600	0.32%	1.90%	52.15

putations of this study, which is similar as in [33, 36, 37]. To further demonstrate the effect of the precision of floating-point numbers on predicted accuracy as well as the computational cost, we present the results using double-precision floating point numbers in Table B.9. As shown, the predicted accuracy using the single- and double-precision floating point format are similar, but the single-precision floating point format is computationally cheaper on GPUs (compared to results in Table B.7). We therefore employ the single-precision floating point numbers in our computations.

Finally, we would like to mention that the optimal architecture of the DNN as well as the number of residual/boundary points in PINNs are problem-dependent. To obtain good accuracy, one can tune the architecture of the DNN and the number of residual/boundary points for PINNs based on the predicted residuals after training adaptively, see [35] for more details.

Table B.9: PINN-BGK for Kovasznay flows with the double-precision floating point format: Relative L_2 errors of velocity (u, v) using different number of residual points. The computational errors and time are the averages over 5 runs with different random residual points.

N_{rp}	err_u	err_v	Walltime (min)
1000	0.30%	1.34%	59.76
5000	0.26%	1.25%	62.03
10000	0.30%	1.36%	59.72

References

[1] I. Y. Akkutlu, Y. Efendiev, M. Vasilyeva, Y. Wang, Multiscale model reduction for shale gas transport in poroelastic fractured media, J. Comput. Phys. 353 (2018) 356–376.

[2] Z. Jin, A. Firoozabadi, Flow of methane in shale nanopores at low and high pressure by molecular dynamics simulations, J. Chem. Phys. 143 (10) (2015) 104315.

[3] J. Greene, Transitioning from the art to the science of thin films: 1964 to 2003, J. Vac. Sci. Technol. A 21 (5) (2003) S71–S73.

[4] A. L. Redman, H. Bailleres, P. Perré, E. Carr, I. Turner, A relevant and robust vacuum-drying model applied to hardwoods, Wood Sci. Technol. 51 (4) (2017) 701–719.

[5] G. E. Karniadakis, A. Beskok, N. Aluru, Microflows and nanoflows: fundamentals and simulation, Springer, New York, 2005.

[6] R. Qiao, P. He, Modulation of electroosmotic flow by neutral polymers, Langmuir 23 (10) (2007) 5810–5816.

[7] S. Succi, R. Benzi, L. Biferale, M. Sbragaglia, F. Toschi, Lattice kinetic theory as a form of supra-molecular dynamics for computational microfluidics, B. Pol. Acad. Sci-Tech. (2007) 151–158.

[8] P. L. Bhatnagar, E. P. Gross, M. Krook, A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems, Phys. Rev. 94 (3) (1954) 511.

[9] H. Chen, S. A. Orszag, I. Staroselsky, Macroscopic description of arbitrary Knudsen number flow using Boltzmann-BGK kinetic theory, J. Fluid Mech. 574 (2007) 495.

[10] J. Meng, Y. Zhang, Accuracy analysis of high-order lattice Boltzmann models for rarefied gas flows, J. Comput. Phys. 230 (3) (2011) 835–849.

[11] L. Luo, Some recent results on discrete velocity models and ramifications for lattice Boltzmann equation, Comput. Phys. Commun. 129 (2000) 63–74.

[12] P. Lallemand, L. Luo, Lattice Boltzmann equation with Overset method for moving objects in two-dimensional flows, J. Comput. Phys. 407 (2020) 109223.

[13] L. Luo, Theory of the lattice Boltzmann method: Lattice Boltzmann models for nonideal gases, Phys. Rev. E 62 (4) (2000) 4982.

[14] Z. Guo, C. Shu, Lattice Boltzmann method and its applications in engineering, World Scientific Publishing, 2013.

[15] S. Succi, The lattice Boltzmann equation: for fluid dynamics and beyond, Oxford University Press, New York, 2001.

[16] C. K. Aidun, J. R. Clausen, Lattice Boltzmann method for complex flows, Annu. Rev. Fluid Mech. 42 (2010) 439–472.

[17] M. Geier, S. Lenz, M. Schönherr, M. Krafczyk, Under-resolved and large eddy simulations of a decaying Taylor-Green vortex with the cumulant lattice Boltzmann method, Theor. Comp. Fluid Dyn. 35 (2) (2021) 169–208.

[18] M. Geier, A. Pasquali, M. Schönherr, Parametrization of the cumulant lattice Boltzmann method for fourth order accurate diffusion part I: Derivation and validation, J. Comput. Phys. 348 (2017) 862–888.

[19] M. Geier, A. Pasquali, M. Schönherr, Parametrization of the cumulant lattice Boltzmann method for fourth order accurate diffusion part II: Application to flow around a sphere at drag crisis, J. Comput. Phys. 348 (2017) 889–898.

[20] P. Lallemand, L. Luo, M. Krafczyk, W. Yong, The lattice Boltzmann method for nearly incompressible flows, J. Comput. Phys. 431 (2021) 109713.

[21] L. Mieussens, Discrete velocity model and implicit scheme for the BGK equation of rarefied gas dynamics, Math. Mod. Meth. Appl. S. 10 (8) (2000) 1121–1149.

[22] K. Xu, J. Huang, A unified gas-kinetic scheme for continuum and rarefied flows, J. Comput. Phys. 229 (20) (2010) 7747–7764.

[23] T. Xiao, C. Liu, K. Xu, Q. Cai, A velocity-space adaptive unified gas kinetic scheme for continuum and rarefied flows, J. Comput. Phys. 415 (2020) 109535.

[24] K. Xu, J. Huang, An improved unified gas-kinetic scheme and the study of shock structures, IMA J. Appl. Math. 76 (5) (2011) 698–711.

[25] V. A. Titarev, Numerical method for computing two-dimensional unsteady rarefied gas flows in arbitrarily shaped domains, Comput. Math. Math. Phys. 49 (7) (2009) 1197–1211.

[26] Z. Guo, K. Xu, R. Wang, Discrete unified gas kinetic scheme for all Knudsen number flows: Low-speed isothermal case, Phys. Rev. E 88 (3) (2013) 033305.

[27] C. Zhang, Z. Guo, Discrete unified gas kinetic scheme for multiscale heat transfer with arbitrary temperature difference, Int. J. Heat Mass Tran. 134 (2019) 1127–1136.

[28] C. Zhang, Z. Guo, S. Chen, An implicit kinetic scheme for multiscale heat transfer problem accounting for phonon dispersion and polarization, Int. J. Heat Mass Tran. 130 (2019) 1366–1376.

[29] X. Meng, H. Sun, Z. Guo, X. Yang, A multiscale study of density-driven flow with dissolution in porous media, Adv. Water Resour. 142 (2020) 103640.

[30] M. Dissanayake, N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, Commun. Numer. Meth. En. 10 (3) (1994) 195–201.

[31] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.

[32] K. Wu, D. Xiu, Data-driven deep learning of partial differential equations in modal space, J. Comput. Phys. 408 (2020) 109307.

[33] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.

[34] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018) 5595–5637.

[35] L. Lu, X. Meng, Z. Mao, G. E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.

[36] X. Meng, Z. Li, D. Zhang, G. E. Karniadakis, PPINN: Parareal physics-informed neural network for time-dependent PDEs, Comput. Method. Appl. M. 370 (2020) 113250.

[37] X. Meng, G. E. Karniadakis, A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems, J. Comput. Phys. 401 (2020) 109020.

[38] X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951.

[39] Z. Mao, A. D. Jagtap, G. E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Method. Appl. M. 360 (2020) 112789.

[40] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.

[41] H. Yu, J. Chen, Y. Zhu, F. Wang, H. Wu, Multiscale transport mechanism of shale gas in micro/nano-pores, Int. J. Heat Mass Tran. 111 (2017) 1172–1180.

[42] X. Yang, W. Zhou, X. Liu, Y. Yan, A multiscale approach for simulation of shale gas transport in organic nanopores, Energy 210 (2020) 118547.

[43] J. Han, C. Ma, Z. Ma, W. E, Uniformly accurate machine learning-based hydrodynamic models for kinetic equations, Proc. Natl. Acad. Sci. U.S.A. 116 (44) (2019) 21983–21991.

[44] P. R. Bandyopadhyay, Rough-wall turbulent boundary layers in the transition regime, J. Fluid Mech. 180 (1987) 231–266.

[45] X. Gu, D. R. Emerson, A high-order moment approach for capturing non-equilibrium phenomena in the transition regime, J. Fluid Mech. 636 (2009) 177.

[46] D. K. Bhattacharya, G. C. Lie, Nonequilibrium gas flow in the transition regime: A molecular-dynamics study, Phys. Rev. A 43 (2) (1991) 761.

[47] M. Barisik, A. Beskok, Surface–gas interaction effects on nanoscale gas flows, Microfluid. Nanofluid. 13 (5) (2012) 789–798.

[48] A. T. Celebi, A. Beskok, Molecular and continuum transport perspectives on electroosmotic slip flows, J. Phys. Chem. C 122 (17) (2018) 9699–9709.

[49] S. Tao, Z. Guo, Boundary condition for lattice Boltzmann modeling of microscale gas flows with curved walls in the slip regime, Phys. Rev. E 91 (4) (2015) 043305.

[50] S. Singh, F. Jiang, T. Tsuji, Impact of the kinetic boundary condition on porous media flow in the lattice Boltzmann formulation, Phys. Rev. E 96 (1) (2017) 013303.

[51] Z. Liu, Z. Mu, H. Wu, A new curved boundary treatment for LBM modeling of thermal gaseous microflow in the slip regime, Microfluid. Nanofluid. 23 (2) (2019) 27.

[52] W. Fang, Y. Tang, C. Ban, Q. Kang, R. Qiao, W. Tao, Atomic layer deposition in porous electrodes: a pore-scale modeling study, Chem. Eng. J. 378 (2019) 122099.

[53] X. Shan, X. He, Discretization of the velocity space in the solution of the Boltzmann equation, Phys. Rev. Lett. 80 (1) (1998) 65.

[54] X. Shan, X. Yuan, H. Chen, Kinetic theory representation of hydrodynamics: a way beyond the Navier–Stokes equation, J. Fluid Mech. 550 (2006) 413–441.

[55] Z. Guo, B. Shi, N. Wang, Lattice BGK model for incompressible Navier–Stokes equation, J. Comput. Phys. 165 (1) (2000) 288–306.

[56] S. Chen, K. Xu, A comparative study of an asymptotic preserving scheme and unified gas-kinetic scheme in continuum flow limit, J. Comput. Phys. 288 (2015) 52–65.

[57] X. Meng, Z. Guo, Localized lattice Boltzmann equation model for simulating miscible viscous displacement in porous media, Int. J. Heat Mass Tran. 100 (2016) 767–778.

[58] Z. Guo, T. S. Zhao, Lattice Boltzmann model for incompressible flows through porous media, Phys. Rev. E 66 (3) (2002) 036304.

[59] Z. Guo, C. Zheng, B. Shi, An extrapolation method for boundary conditions in lattice boltzmann method, Phys. Fluids 14 (6) (2002) 2007–2010.

[60] L. Kovasznay, Laminar flow behind a two-dimensional grid, Proc. Camb. Philos. Soc. 44 (1) (1948) 58–62.

[61] Y. Qian, D. d'Humières, P. Lallemand, Lattice BGK models for Navier-Stokes equation, Europhys. Lett. 17 (6) (1992) 479.

[62] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[63] R. H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. 16 (5) (1995) 1190–1208.

[64] L. Yang, S. Hanneke, J. Carbonell, A theory of transfer learning with applications to active learning, Mach. Learn. 90 (2) (2013) 161–189.

[65] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, G. Zhang, Transfer learning using computational intelligence: A survey, Knowl.-Based Syst. 80 (2015) 14–23.

[66] L. McClenny, U. Braga-Neto, Self-adaptive physics-informed neural networks using a soft attention mechanism, arXiv preprint arXiv:2009.04544 (2020).

[67] J. Shen, Dynamics of regularized cavity flow at high reynolds numbers, Appl. Math. Lett. 2 (4) (1989) 381–384.

[68] J. Shen, Hopf bifurcation of the unsteady regularized driven cavity flow, J. Comput. Phys. 95 (1) (1991) 228–245.

[69] T. Lee, C. Lin, A characteristic Galerkin method for discrete Boltzmann equation, J. Comput. Phys. 171 (1) (2001) 336–356.

[70] L. Yang, C. Shu, W. Yang, J. Wu, An improved three-dimensional implicit discrete velocity method on unstructured meshes for all Knudsen number flows, J. Comput. Phys. 396 (2019) 738–760.

[71] D. Galant, Gauss Quadrature rules for the evaluation of $2\pi^{-1/2}\int_0^\infty \exp(-x^2)f(x)dx$, Math. Comput. 23 (1969) 674, s31–s39.

[72] Y. Sone, S. Takata, T. Ohwada, Numerical analysis of the plane Couette flow of a rarefied gas on the basis of the linearized Boltzmann equation for hard sphere molecules, Eur. J. Mech. B-Fluid. 9 (1990) 273–288.

[73] G. Hinton, N. Srivastava, K. Swersky, Neural networks for machine learning lecture 6a overview of mini-batch gradient descent, Cited on 14 (8) (2012).

[74] M. Li, T. Zhang, Y. Chen, A. J. Smola, Efficient mini-batch training for stochastic optimization, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 661–670.

[75] S. Chen, C. Zhang, L. Zhu, Z. Guo, A unified implicit scheme for kinetic model equations. Part I. Memory reduction technique, Sci. Bull. 62 (2) (2017) 119–129.

[76] F. Scarselli, A. C. Tsoi, Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results, Neural networks 11 (1) (1998) 15–37.

[77] T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: A survey, J. Mach. Learn. Res. 20 (55) (2019) 1–21.