Journal of Computational Physics 443 (2021) 110521

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Using neural networks to accelerate the solution of the Boltzmann equation

Tianbai Xiao *, Martin Frank

Karlsruhe Institute of Technology, Karlsruhe, Germany

ARTICLE INFO

Article history:
Available online 17 June 2021

Keywords:
Boltzmann equation
Kinetic theory
Non-equilibrium flow
Deep learning
Neural network

ABSTRACT

One of the biggest challenges for simulating the Boltzmann equation is the evaluation of fivefold collision integral. Given the recent successes of deep learning and the availability of efficient tools, it is an obvious idea to try to substitute the calculation of the collision operator by the evaluation of a neural network. However, it is unlcear whether this preserves key properties of the Boltzmann equation, such as conservation, invariances, the H-theorem, and fluid-dynamic limits.

In this paper, we present an approach that guarantees the conservation properties and the correct fluid dynamic limit at leading order. The concept originates from a recently developed scientific machine learning strategy which has been named "universal differential equations". It proposes a hybridization that fuses the deep physical insights from classical Boltzmann modeling and the desirable computational efficiency from neural network surrogates. The construction of the method and the training strategy are demonstrated in detail. We conduct an asymptotic analysis and illustrate the multi-scale applicability of the method. The numerical algorithm for solving the neural network-enhanced Boltzmann equation is presented as well. Several numerical test cases are investigated. The results of numerical experiments show that the time-series modeling strategy enjoys the training efficiency on this supervised learning task.

1. Introduction

Modern data-driven techniques widen the possibility of solving the problems that seemed beset with difficulties in the past, e.g., computer vision [1] and natural language processing [2]. The same momentum is building in computational sciences, leading to the so-called scientific machine learning (SciML) [3]. As typical classification and regression tasks in classical machine learning applications mostly handle discrete and localized data, in the SciML, information at different locations is expected to be connected by mathematical and physical constraints, i.e., ordinary and partial differential equations (PDEs). Besides, given the high expense of conducting experiments and numerical simulations, e.g., for the research of fluid dynamics and particle physics, it is challenging to establish an all-round data base. While the generalization performance of neural networks based on small training sets is questionable, quantitative interpretability lies at the core of scientific modeling and simulation, which more or less collides with the blackbox nature of multi-layer artificial neural networks (NNs) employed in the deep learning.

* Corresponding author.
E-mail addresses: tianbaixiao@gmail.com (T. Xiao), martin.frank@kit.edu (M. Frank).

https://doi.org/10.1016/j.jcp.2021.110521
0021-9991/

Several approaches that try to fuse the advantages of differential equations and machine learning have emerged recently. One intuitive strategy falls into the search of NN surrogates for high-dimensional PDE solutions [4–9]. For example, the physics-informed neural networks (PINNs) encode prior knowledge from differential equations or existing data into the deep learning models. By minimizing a cost function that represents the residual of differential equations, the numerical solutions can be iterated along with the training process. PINNs have led to a wide range of applications in fluid mechanics [10–14], biology [15,16], materials science [17–19], and uncertainty quantification [20,21].

The approximation approaches of direct data-to-solution mapping of physical systems have also been discovered in the SciML community [22–25]. For example, Raissi et al. proposed a method that is called hidden fluid mechanics to learn velocity and pressure fields directly from flow visualizations [26]. Thanks to the capability of interpolating data by NNs, such a method is robust with respect to low resolution and noises in the observation data. In general, these strategies are mostly data driven and have no much correlation with underlying physics.

Another direction denotes the data-driven discovery of governing equations [27–30]. For example, the sparse identification of nonlinear dynamical systems (SINDy) employs sparse regression to choose most probable equations from data [31]. It provides an efficient tool for identifying partial differential equations, e.g. the transport coefficients used in the Navier-Stokes equations, and require a relatively small training dataset. SINDy is able to incorporate existing knowledge into an unknown system to be studied but the automatic combination of alternatives relies on a simple recombination of differential terms.

As we look into multi-scale fluid mechanics with the upscaling effects from atomistic level, more complex dynamical system, e.g. the phase space evolution, could emerge. A typical example is the Boltzmann equation, which describes the evolution of the one-particle probability density function $f(t, \mathbf{x}, \mathbf{u})$, which describes the probability of finding a particle with a certain location $\mathbf{x}$ and speed $\mathbf{u}$. In the absence of external force field, the Boltzmann equation reads as follows,

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla_{\mathbf{x}} f = Q(f) = \int_{\mathcal{R}^3} \int_{\mathcal{S}^2} \mathcal{B}(\cos\beta, g) \left[f(\mathbf{u}') f(\mathbf{u}_*') - f(\mathbf{u}) f(\mathbf{u}_*) \right] d\mathbf{\Omega} d\mathbf{u}_*, \tag{1}$$

where $\{\mathbf{u}, \mathbf{u}_*\}$ are the pre-collision velocities of two colliding particles, and $\{\mathbf{u}', \mathbf{u}_*'\}$ are the corresponding post-collision velocities. The collision kernel $\mathcal{B}(\cos\beta, g)$ measures the strength of collisions in different directions, where $g = |\mathbf{g}| = |\mathbf{u} - \mathbf{u}_*|$ is the magnitude of relative pre-collision velocity, $\mathbf{\Omega}$ is the unit vector along the relative post-collision velocity $\mathbf{u}' - \mathbf{u}_*'$, and the deflection angle β satisfies the relation $\cos\beta = \mathbf{\Omega} \cdot \mathbf{g}/g$.

The Boltzmann equation serves as the basis of many high-level theories, e.g. non-equilibrium thermodynamics and extended hydrodynamics [32]. As is shown, the Boltzmann equation is an integro-differential equation, with its right-hand side being a fivefold integral over phase space. This convolution-type collision operator brings tremendous difficulty to the application of PINN since a direct differentiable structure is absent. On the other hand, despite the development of classical numerical solvers, the computational cost of solving the Boltzmann collision integral can be prohibitive. Consider the fast spectral method [33], an efficient Boltzmann solution algorithm which employs fast Fourier transform to compute convolutions within spectral space. The computational cost of it is $O(M^2 N_x^D N_u^D \log N_u)$, where N_x, N_u and M are the numbers of grids in physical, velocity and angular space with dimension D [34]. This makes it unrealistic to perform a direct numerical simulation for a real-world application in aerospace industry. Moreover, due to the high-dimensional nature of intermolecular interactions, it is sometimes cumbersome to adopt the Boltzmann solver if we are interested in one-dimensional distribution of solutions only, e.g. the profiles of macroscopic variables inside a shock tube.

The goal of current work is to use deep neural networks as building blocks in a numerical method to solve the Boltzmann equation. Besides the complicated high-dimensional integro-differential structure, the Boltzmann equation as a physical model at the same time possesses an intricate structure of many-particle system that a numerical method needs to preserve. Therefore, as a universal function approximator in the latent space [35], artificial neural networks can be beneficial, but cannot be used out-of-the-box. In this paper, we consider the idea of hybridizing mechanical and neural dynamics into a learnable framework, which originates from a recently developed scientific machine learning strategy named as "universal differential equations" [36]. The universality refers to the differential equations defined in part by universal approximators. Specifically, we keep the particle transport model of original Boltzmann equation, and build the NN-enhanced collision model on the basis of kinetic relaxation model. The proposed neural network-enhanced Boltzmann equation maintains a well balance of interpretability and flexibility, and we prove that the current approach guarantees the conservation properties and the correct fluid dynamic limit at leading order.

The paper is organized as follows. In Sec. 2 we introduce some fundamental concepts in the kinetic theory of gases. Sec. 3 presents the main idea of this work, and Sec. 4 details the numerical solution algorithm. Sec. 5 contains the numerical experiments for both spatially homogeneous and inhomogeneous cases to validate the current method. The last section is the conclusion.

2. Kinetic theory of gases

Kinetic theory depicts the evolution of a many-particle system at the scale of mean free path and collision time. It provides a one-to-one correspondence with the macroscopic description, which can be regarded as its limiting case. Taking moments through particle velocity space, we get the macroscopic mass, momentum and energy density,

$$\mathbf{W}(t,\mathbf{x}) = \begin{pmatrix} \rho \\ \rho\mathbf{U} \\ \rho E \end{pmatrix} = \int f\psi d\mathbf{u}, \tag{2}$$

where $\psi = \left(1, \mathbf{u}, \frac{1}{2}\mathbf{u}^2\right)^T$ is a vector of collision invariants. The collision operator satisfies the compatibility condition for conservative variables, i.e.,

$$\int Q(f)\psi d\mathbf{u} = 0. \tag{3}$$

Substituting the H function,

$$H(t,\mathbf{x}) = -\int f \ln f d\mathbf{u},$$

into the Boltzmann equation we have

$$\frac{\partial H}{\partial t} = -\int (1+\ln f)\frac{\partial f}{\partial t} d\mathbf{u} = -\iiint (1+\ln f)\left(f' f'_* - f f_*\right) \mathcal{B} d\Omega d\mathbf{u} d\mathbf{u}_*. \tag{4}$$

From the H-theorem [37] we know that the physical entropy is locally maximal when f is a Maxwellian,

$$\mathcal{M}(t,\mathbf{x},\mathbf{u},\mathbf{z}) = \rho\left(\frac{\lambda}{\pi}\right)^{\frac{3}{2}} e^{-\lambda(\mathbf{u}-\mathbf{U})^2}, \tag{5}$$

where $\lambda = m/(2kT)$, m is molecule mass and k is the Boltzmann constant.

Since intermolecular collisions drive the system towards Maxwellian, simplified relaxation models, e.g. the Bhatnagar-Gross-Krook (BGK) [38] and Shakhov [39], have been constructed. It writes

$$\frac{\partial f}{\partial t} + \mathbf{u}\cdot\nabla_{\mathbf{x}} f = Q(f) = \nu(f^+ - f), \tag{6}$$

where ν is collision frequency. For the BGK model, the equilibrium state is Maxwellian $f^+ = \mathcal{M}$, while in the Shakhov model it takes the form

$$f^+ = \mathcal{M}\left[1 + (1-\mathrm{Pr})(\mathbf{u}-\mathbf{U})\cdot\mathbf{q}\left(\frac{(\mathbf{u}-\mathbf{U})^2}{RT} - 5\right)/(5pRT)\right], \tag{7}$$

where Pr is the Prandtl number, $\mathbf{q}$ is heat flux, p is pressure and R is gas constant. The relaxation models avoid the complicated fivefold Boltzmann integral. They still possess some key properties of the original Boltzmann equation, e.g. the H-theorem, but fail to provide exactly equivalent Boltzmann solutions as the distribution function deviates far from the Maxwellian.

3. Neural network-enhanced Boltzmann equation

3.1. Idea

One intuitive strategy is to build a neural network surrogate of the nonlinear function described by the Boltzmann equation. Instead of modeling the data-to-solution mapping explicitly, one can lean on the structure of Boltzmann equation and incorporate the neural network function as the derivative term. It follows the form of a newly developed family of deep neural network models, i.e. the neural ordinary differential equation (ODE) [40], and the nomenclature can be inherited here as the neural Boltzmann equation (NBE),

$$f_t = \mathrm{NN}_\theta(f,t), \tag{8}$$

where θ denotes the collection of all the parameters inside neural network (NN).

The idea of Neural ODEs originates from the structure of some state-of-the-art neural networks, e.g. residual neural network, which updates the hidden state with the strategy

$$\mathbf{h}^{n+1} = \mathbf{h}^n + \mathcal{F}(\mathbf{h}^n, \theta^n), \tag{9}$$

where $n \in \{0, 1, \ldots, N\}$ is the index of hidden layers. Such an iterative stepping can be regarded equivalently as N forward Euler steps (with a fixed step size $\delta t = 1$) for solving differential equations. Therefore, in the limiting case with infinite number of layers, the discrete iteration can be concluded by an ordinary differential equation in terms of neural network, i.e. the so-called neural ODE,

$$\mathbf{h}_t = \mathcal{F}(\mathbf{h}, \theta, t), \quad t \in (0, N], \tag{10}$$

where $t \in (0, N]$ is introduced as an artificial time [41]. Eq. (10) forms an initial value problem (IVP) for the neural network. Since the derivatives of hidden layers are parameterized with continuous dynamics, the parameters of original discrete sequence layers in Eq. (9) can be regarded as seamlessly coupled. As a result, for a typical supervised learning task, the required number of parameters drops correspondingly [40]. Modern ODE solvers can be employed to solve the IVPs with monitoring of desirable accuracy and efficiency. No intermediate quantities of forward pass need to be stored, leading to a constant memory cost as a function of depth. Also, the continuous modeling make it much easier to perform interpolation and extrapolation beyond the training data. It is worth mentioning that the dependence on the parameters is typically non-smooth and therefore the optimization is still challenging with the continuous formulation in time.

In spite of the advantages, due to the blackbox nature of neural networks, this approach does not guarantee any property of the Boltzmann equation under the optimization and estimation errors within training and computing processes. One rather general approach to enforce physical constraints has been presented under the name universal differential equations (UDE) [36], in which the model is constructed cooperatively by mechanical formulations and neural networks as universal function approximators. Continuing with the example above, we rewrite the kinetic model and call it universal Boltzmann equation (UBE)

$$f_t = Q\left(f, t, \mathrm{NN}_\theta(f, t)\right), \tag{11}$$

where Q is the particle collision term, and $\mathrm{NN}_\theta(f, t)$ denotes the neural network model that plays a portion necessary for the self-contained physical description but missed from the mechanical modeling.

The key idea to go beyond the mere approximation of the right-hand side (and thus to obtain a neural differential equation) is to split the right-hand side into a mechanistic part and a part to be approximated. In this paper, we employ the BGK equation as the mechanical part of the UBE, leaving the difference to the full Boltzmann collision operator to be approximated by a neural network. The concrete UBE is designed as follows,

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla_{\mathbf{x}} f = \nu(\mathcal{M} - f) + \mathrm{NN}_\theta(\mathcal{M} - f), \tag{12}$$

where NN_θ can be a concrete type of neural network, with the input set as the difference between the Maxwellian and current particle distribution function.

The proposed UBE has the following benefits. First, the BGK model has a similar structure as the Boltzmann equation, while the computational cost is of $O(N^D)$, where N is the number of discrete velocity grids and D is dimension. Therefore, it is significantly more efficient than solving the full Boltzmann collision integral.

Second, it automatically ensures the asymptotic limits. Let us consider the Chapman-Enskog method for solving Boltzmann equation [42], where the distribution function is expanded with respect to collision frequency ν,

$$f \simeq \sum_{i=0} \nu^i f^{(i)}, \quad f^{(0)} = \mathcal{M}. \tag{13}$$

Take the zeroth order truncation, and consider an illustrative L-layer perceptron network free of biases,

$$\mathrm{NN}_\theta(x) = \mathrm{layer}_L(\ldots \mathrm{layer}_2(\mathcal{A}(\mathrm{layer}_1(\mathbf{x})))), \quad \mathrm{layer}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}, \tag{14}$$

where each layer is a matrix multiplication followed by activation function $\mathcal{A}$ that can adpot sigmoid, tanh, ReLU, etc. Given the zero input from $(\mathcal{M} - f)$, the contribution from the collision term is removed naturally. Taking moments with respect to collision invariants,

$$\int \begin{pmatrix} 1 \\ \mathbf{u} \\ \frac{1}{2}\mathbf{u}^2 \end{pmatrix} (\mathcal{M}_t + \mathbf{u} \cdot \nabla_{\mathbf{x}} \mathcal{M})\, d\mathbf{u} = 0, \tag{15}$$

we arrive at the corresponding Euler equations,

$$\frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho\mathbf{U} \\ \rho E \end{pmatrix} + \nabla_{\mathbf{x}} \cdot \begin{pmatrix} \rho\mathbf{U} \\ \rho\mathbf{U} \otimes \mathbf{U} \\ \mathbf{U}(\rho E + p) \end{pmatrix} = 0. \tag{16}$$

As is shown, the asymptotic property of UBE in the hydrodynamic limit is preserved independent of the training parameters θ. The above analysis illustrates the design idea of the current model. It is widely believed that neural model performs better if the architecture is designed to be consistent with the solution structure. With the splitting of BGK relaxation and its deviation from the exact Boltzmann collision integral, the relaxation term can be regarded as pre-conditioning of the neural differential equation, and the inevitable optimization and prediction errors will not affect the continuum limit from such a biases-free neural network.

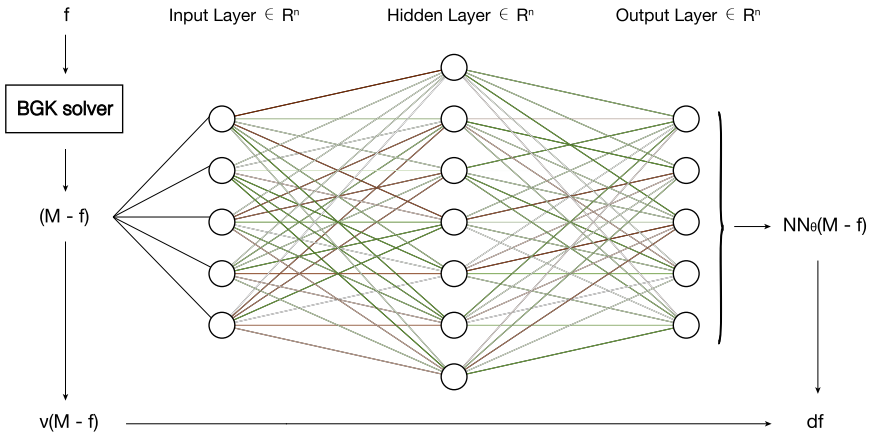


Fig. 1. An illustrative flow chart for the collision term evaluation in the universal Boltzmann equation.

Another advantage from current strategy is the training efficiency. Since the BGK relaxation term provides a qualitative mechanism to describe gas evolution, as analyzed in [43–45], after several collisions from initial non-equilibrium distribution, the difference between Boltzmann integral and BGK model becomes minor. Therefore, now the task left becomes to train a neural network that approximates values close to zero, which will significantly accelerate the convergence of θ. Fig. 1 provides an illustration for the collision term evaluation in the universal Boltzmann equation, and the detailed training strategy will be presented in the next subsection.

3.2. Training strategy

Training NBE and UBE with datasets consisting of exact or reference solutions is a typical supervised learning task. It amounts to an optimization problem which minimizes the difference between the current predictions and ground-truth solutions. For example, a cost function can be defined based on the Euclidean distance along discrete grid points,

$$C(\theta) = \sum_{i,j,n} ||f_\theta - f_{\text{ref}}||^2(t^n, \mathbf{x}_i, \mathbf{u}_j) + \zeta \sum_{l=1}^{L} ||\theta^{(l)}||^2. \tag{17}$$

The latter plays as an regularization term which sums over the squared weight parameters of the network to mitigate overfitting, where $\theta^{(l)}$ denotes the weight parameters of l-th layer, L is the total number of layers, and the regularization strength is chosen as $\zeta = 1.0 \times 10^{-6}$.

In general, the optimization algorithms can be classified into gradient-free and gradient-required methods. Thanks to the rapid development of automatic differentiation (AD), the latter one becomes prevalent in machine learning community. There are two modes of AD, i.e. the forward-mode and the reverse-mode, which differ from the direction of evaluating the chain rules, and here we focus on the latter. Consider a smooth function $y = \mathcal{F}(x)$, the reverse-mode AD computes the dual (conjugate-transpose) matrix of Jacobian $\mathcal{J} = \nabla\mathcal{F}$ at $x = x_0$ with the chain rule,

$$(\mathcal{J}(\mathcal{F})(x_0))^* = (\mathcal{J}(G_1)(x_0))^* \times \cdots \times \left(\mathcal{J}(G_k)\left(x_{k-1}\right)\right)^*, \tag{18}$$

with $x_i := G_i(x_{i-1})$ for $i = 1, \ldots, k-1$.

As the reverse-mode AD can naturally be expressed using pullbacks and differential one-forms from geometric perspective, in this work we employ open-source package Zygote.jl [46], which utilizes pullback functions to perform reverse-mode AD. Different from the tracing methods used in Tensorflow [47] and PyTorch [48], it employs the source-to-source mode via differentiable programming, i.e. generates derivative directly from pullback functions. Such an approach enjoys the benefits of, e.g. low overhead, efficient support for control flow and user-defined data types and dynamism.

When the derivatives of cost function have been evaluated by automatic differentiation, gradient-descent-type optimizers can be employed, e.g. the standard stochastic gradient decent method, ADAM [49], Nesterov [50], Broyden–Fletcher–Goldfarb–Shanno (BFGS) [51], or its limited-memory version (L-BFGS). In this paper, the dataset for training neural networks is produced by the fast spectral method [52] with respect to different initial value problems of homogeneous Boltzmann equation

$$f_t = \int_{\mathcal{R}^3}\int_{\mathcal{S}^2} \mathcal{B}(\cos\beta, g)\left[f(\mathbf{u}')f(\mathbf{u}_*') - f(\mathbf{u})f(\mathbf{u}_*)\right] d\mathbf{\Omega} d\mathbf{u}_*. \tag{19}$$

The open-source solution algorithm is implemented in Kinetic.jl [53], which works together with DifferentialEquations.jl ecosystem [54] and generates time-series data with desirable orders of accuracy along evolution trajectories.

4. Solution algorithm

4.1. Update algorithm

We construct the numerical algorithm within finite volume framework. The notation of cell-averaged particle distribution function in a control volume is adopted,

$$f(t^n, \mathbf{x}_i, \mathbf{u}_j) = f_{i,j}^n = \frac{1}{\Omega_i(\mathbf{x})\Omega_j(\mathbf{u})} \int_{\Omega_i} \int_{\Omega_j} f(t^n, \mathbf{x}, \mathbf{u}) d\mathbf{x} d\mathbf{u}, \tag{20}$$

where Ω_i and Ω_j are the cell area in the discrete physical and velocity space. The update of distribution function can be formulated as

$$f_{i,j}^{n+1} = f_{i,j}^n + \frac{1}{\Omega_i} \int_{t^n}^{t^{n+1}} \sum_{r=1}^{n_f} F_r \Delta S_r dt + \int_{t^n}^{t^{n+1}} Q(f_{i,j}) dt, \tag{21}$$

where F_r is the time-dependent flux function of distribution function at cell interface, ΔS_r is the interface area and n_r is the number of interfaces per cell.

4.2. Interface flux

For the numerical flux evaluation, we first reconstruct the particle distribution function around the cell interface, e.g. around $\mathbf{x}_{i+1/2}$,

$$\begin{aligned} f_{i+1/2,j}^L &= f_{i,j}, \\ f_{i+1/2,j}^R &= f_{i+1,j}, \end{aligned} \tag{22}$$

with first-order accuracy and

$$\begin{aligned} f_{i+1/2,j}^L &= f_{i,j} + \nabla_{\mathbf{x}} f_{i,j} \cdot (\mathbf{x}_{i+1/2} - \mathbf{x}_i), \\ f_{i+1/2,j}^R &= f_{i+1,j} + \nabla_{\mathbf{x}} f_{i+1,j} \cdot (\mathbf{x}_{i+1/2} - \mathbf{x}_{i+1}), \end{aligned} \tag{23}$$

with second-order accuracy, where $\nabla_{\mathbf{x}} f$ is the reconstructed gradient with limiters.

The interface distribution function is defined in an upwind way, i.e.,

$$f_{i+1/2,j} = f_{i+1/2,j}^L H[\mathbf{u}_j] + f_{i+1/2,j}^R (1 - H[\mathbf{u}_j]), \tag{24}$$

where $H[x]$ is the Heaviside step function. The corresponding numerical flux of particle distribution function can be evaluated via

$$F_{i+1/2,j} = f_{i+1/2,j} \mathbf{n}_{i+1/2} \cdot \mathbf{u}_j, \tag{25}$$

where $\mathbf{n}_{i+1/2}$ is the unit normal vector of cell interface and $\mathbf{u}_j$ denotes discrete velocity at j-th quadrature point.

4.3. Collision term

The neural network-enhanced collision term inside each cell is formulated as

$$Q(f_{i,j}) = \nu_i(\mathcal{M}_{i,j} - f_{i,j}) + \mathrm{NN}_\theta(\mathcal{M}_{i,j} - f_{i,j}). \tag{26}$$

The collision frequency is defined as,

$$\nu = p/\mu, \tag{27}$$

where p is pressure., and μ is viscosity coefficient. It follows the variational hard-sphere (VHS) model's rule,

$$\mu = \mu_{\mathrm{ref}} \left(\frac{T}{T_{\mathrm{ref}}} \right)^{\omega}, \tag{28}$$

where μ_{ref} and T_{ref} are the viscosity and temperature in the reference state, and ω is the viscosity index.

Table 1
Computational setup for training set production in homogeneous relaxation.

t	Δt	u	v	w	N_u	N_v	N_w
$[0, 2]$	0.2	$[-5, 5]$	$[-5, 5]$	$[-5, 5]$	80	28	28
Quadrature	Kn	Pr	μ_{ref}	α	ω	Integrator	
rectangular	1	2/3	0.554	1.0	0.5	Tsitouras' 5/4	

Once the interface fluxes are defined, the solution algorithm in Eq. (21) becomes

$$f_{i,j}^{n+1} = f_{i,j}^{n} + \frac{1}{\Omega_i} \int_{t^n}^{t^{n+1}} \sum_{r=1}^{n_f} F_r \Delta S_r dt + \int_{t^n}^{t^{n+1}} \left[\nu_i (\mathcal{M}_{i,j} - f_{i,j}) + \text{NN}_\theta (\mathcal{M}_{i,j} - f_{i,j}) \right] dt. \tag{29}$$

The most straightforward time-integral algorithm for the above equation is the forward Euler method. If the time step is much larger than mean collision time $\tau = 1/\nu$, or more accurate solutions are requested, higher-order methods, e.g. the Forward Euler method, the midpoint rule, the Rosenbrock approach [55], and Tsitouras's 5/4 Runge-Kutta method [56], can be employed to ensure a balance of accuracy and stability.

5. Numerical experiments

In this section, we will introduce the detailed methodology for conducting numerical experiments and the solutions to validate the current model and scheme. Both spatially uniform and non-uniform Boltzmann equations will be considered. For convenience, dimensionless variables will be introduced in the simulations,

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{L_0},\ \tilde{\rho} = \frac{\rho}{\rho_0},\ \tilde{T} = \frac{T}{T_0},\ \tilde{\mathbf{u}} = \frac{\mathbf{u}}{(2RT_0)^{1/2}},\ \tilde{\mathbf{U}} = \frac{\mathbf{U}}{(2RT_0)^{1/2}},$$
$$\tilde{f} = \frac{f}{\rho_0 (2RT_0)^{3/2}},\ \tilde{\mathbf{T}} = \frac{\mathbf{T}}{\rho_0 (2RT_0)},\ \tilde{\mathbf{q}} = \frac{\mathbf{q}}{\rho_0 (2RT_0)^{3/2}},$$

where R is the gas constant, $\mathbf{T}$ is stress tensor, and $\mathbf{q}$ is heat flux. The denominators with subscript zero are characteristic variables in the reference state. For brevity, the tilde notation for dimensionless variables will be removed henceforth.

5.1. Homogeneous relaxation

First we consider the homogeneous relaxation of particles from an initial non-equilibrium distribution, i.e.

$$f(t = 0, u, v, w) = \frac{1}{2\pi^{2/3}} (\exp(-(u - 0.99)^2) + \exp(-(u + 0.99)^2)) \exp(-v^2) \exp(-w^2).$$

The training set is produced by the fast spectral method, which consists a series of discrete particle distribution functions from every time step $\Delta t = 0.2$. The detailed computational setup can be found in Table 1. Notice that the viscosity coefficient in the reference state is connected with the Knudsen number,

$$\mu_0 = \frac{5(\alpha + 1)(\alpha + 2)\sqrt{\pi}}{4\alpha(5 - 2\omega)(7 - 2\omega)} \text{Kn},$$

where $\{\alpha, \omega\}$ are parameters for the VHS model.

As the initial particle distribution along y and z is set as equilibrium, we are mostly concerned about the evolution in x direction. We employ neural network to conduct dimension reduction, and the corresponding universal Boltzmann equation writes,

$$h_t(t, u) = \nu(\mathcal{M} - h) + \text{NN}_\theta(\mathcal{M} - h).$$

The reduced distribution function here is defined as

$$h(t, u) = \iint f(t, u, v, w) dv dw,$$

and the training data is projected into one-dimensional profiles in the same way. The neural network chain consists of two hidden dense layers with $(N_u \times 16)$ neurons each, and the tanh plays as the activation function. Tsitouras' 5/4 Runge-Kutta method [56] is again used to solve the UBE and produces the data h_θ at same instants as training set, and the loss function is evaluated through mean squared error, i.e.

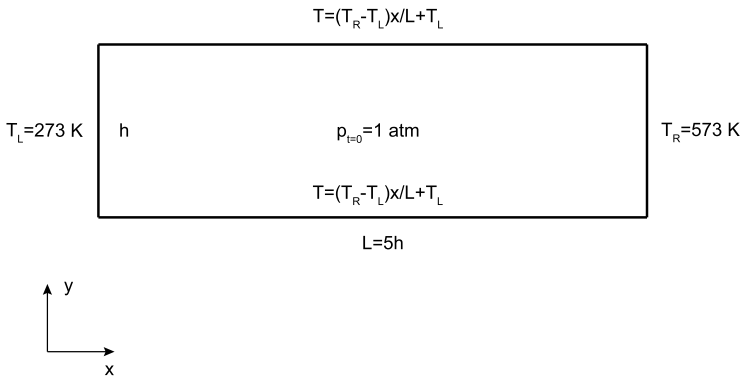
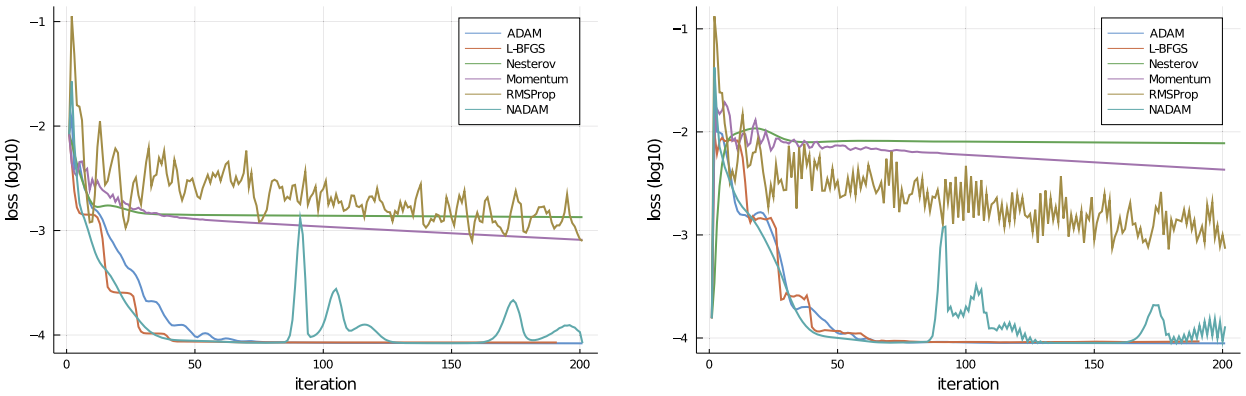


Fig. 2. Schematic of thermal creep problem.



(a) Training set (b) Test set

Fig. 3. Loss functions of universal Boltzmann equation versus iterations under different optimizers in the homogeneous relaxation problem. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$L(\theta) = \sum_{j,n} (h_\theta - h_{\text{train}})^2 (t^n, u_j) + \zeta \sum_{l=1}^{2} ||\theta^{(l)}||^2,$$

where the regularization parameter takes $\zeta = 10^{-6}$.

The variation of loss function with respect to iterations in both training and test sets is shown in Fig. 3. Some commonly used optimizers are compared for the training of UBE. As a second-order gradient method, L-BFGS enjoys the fastest convergence speed in such a supervised learning problem with relatively small amounts of data. Thanks to the usage of gradient momentum and the self-adaptive learning rate, ADAM provides a robust and fast gradient descent process. Simple momentum-based methods, i.e. Nesterov and RMSProp, however, provide a much slower convergence due to the prefixed total learning rate. The hybrid Nesterov and ADAM algorithm, i.e. NADAM, presents equivalent convergence speed as L-BFGS in the beginning, but suffers from some fluctuations as the training proceeds.

Once the training process finishes, we get the universal differential equation we need. Since we are modeling continuous dynamics, instead of keeping the same approaches for producing training set, we can choose different algorithms with respect to accuracy requirement. For instance, now we use the midpoint rule to solve the obtained UBE. Fig. 4 presents the particle distribution profile along u direction at the same time instants as training set. Different from the time-series training used in the UBE, we also plot the results with conventional discrete training strategy based on the same initial condition. As can be seen, significant differences exist between Boltzmann and BGK solutions. Thanks to the continuous time-series training, the current UBE holds much better training performance than the direct training of Boltzmann integral on discrete time instants. With the mechanical part being BGK model, the UBE provides perfectly equivalent solutions as fast spectral method (FSM) of Boltzmann equation at a much lower computational cost. Table 2 shows the detailed memory usage, allocation numbers, and running time of the two methods for solving the right-hand side of the Boltzmann equation with the midpoint rule. As can be seen, the current method saves 97% memory load and achieves 33 times faster computational efficiency.

An effective neural network as function approximator should be able to conduct interpolation and extrapolation beyond the training set. To test the performance of trained UBE, we recalculate the time-series solution within $t \in [0, 9]$ and save

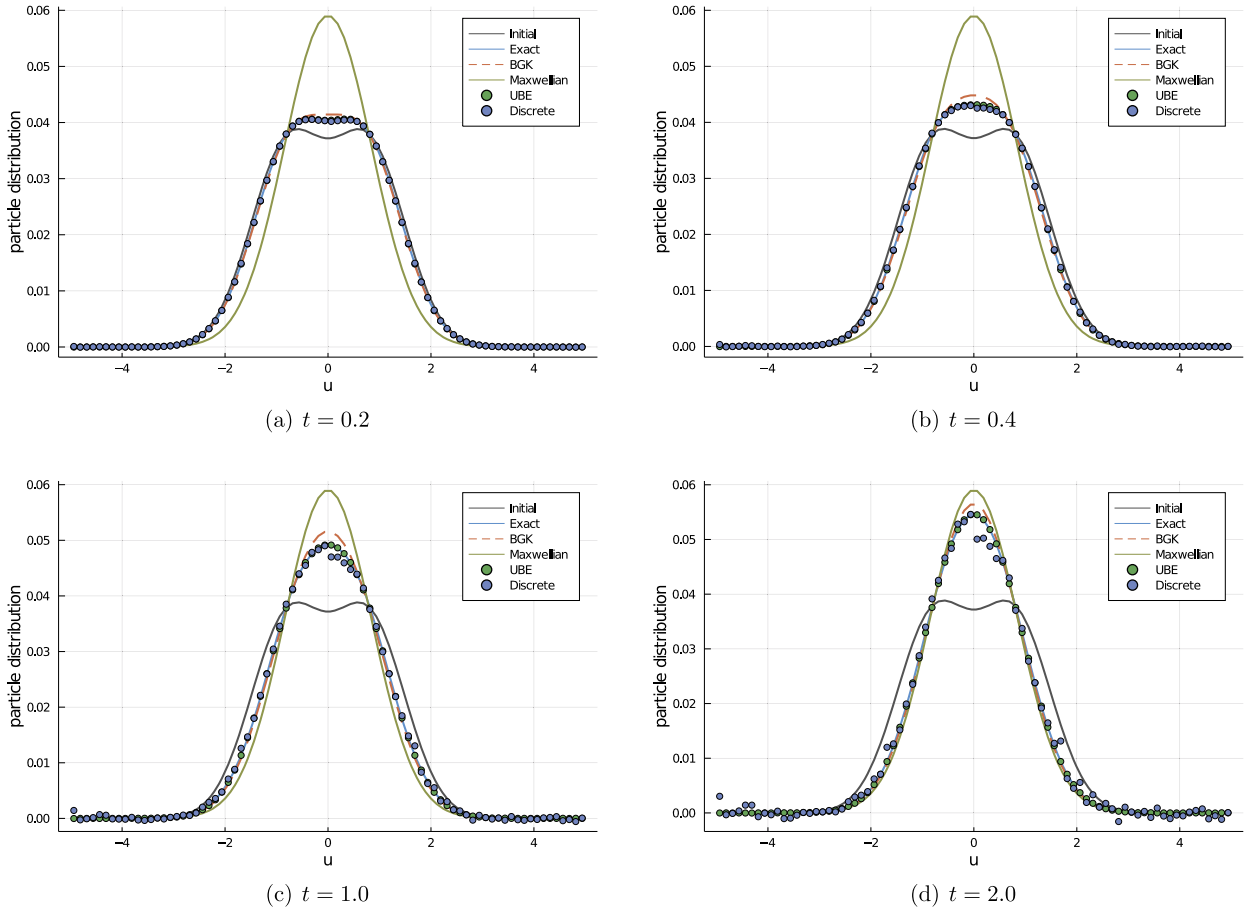


Fig. 4. Particle distribution functions at different time instants within the training set of the homogeneous relaxation problem.

Table 2
Computational cost for solving the right-hand side of the Boltzmann equation with midpoint rule in homogeneous relaxation.

	MEM	n_{allocs}	t_{min}	t_{median}	t_{mean}	t_{max}	n_{samples}
UBE	242.25 MB	5638	89.30 ms	142.43 ms	144.87 ms	221.96 ms	35
FSM	7.42 GB	194610	4.71 s	4.83 s	4.83 s	4.94 s	2

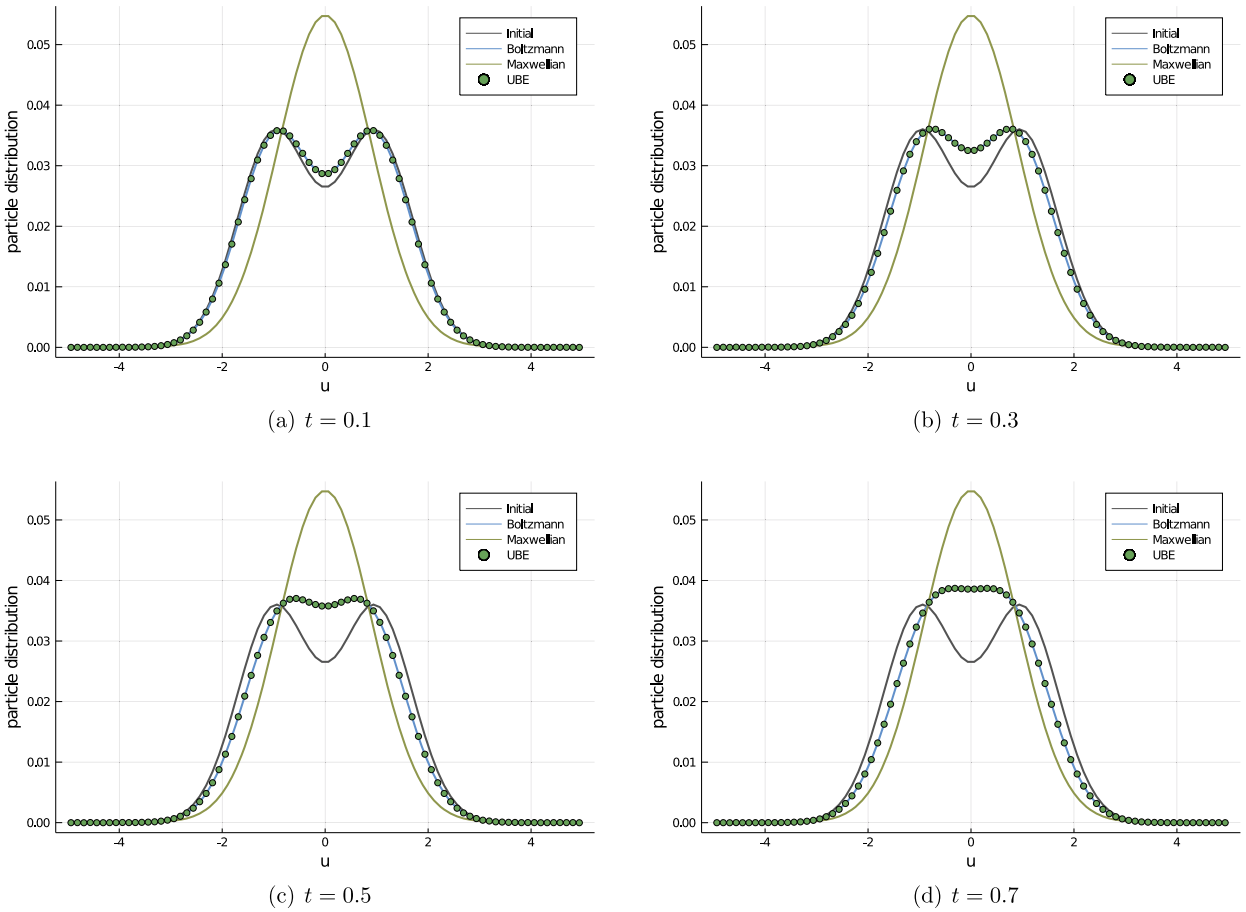
at every $\Delta t = 0.1$. Obviously, there exist solutions off and beyond the original training data. Fig. 5 presents the particle distribution profile along u direction at the offset data points, and Fig. 6 shows the profile at extrapolation points.

The benchmark solutions are provided by FSM with the same computational setup. As is shown, the UBE provides equivalent solutions as FSM at interpolated and extrapolated time instants. Fig. 7 plots the particle distribution function and collision term in the phase space throughout the evolution. Fig. 8 demonstrates that the entropy inequality is satisfied precisely by the UBE. This case serves as a benchmark validation of the universal model and numerical scheme to provide Boltzmann solutions efficiently.

5.2. Normal shock structure

Then let us turn to spatially inhomogeneous case. The normal shock wave structure is an ideal case to validate theoretical modeling and numerical algorithm in case of highly dissipative flow organizations and strong non-equilibrium effects. Built on the reference frame of shock wave, the stationary upstream and downstream status can be described via the well-known Rankine-Hugoniot relation,

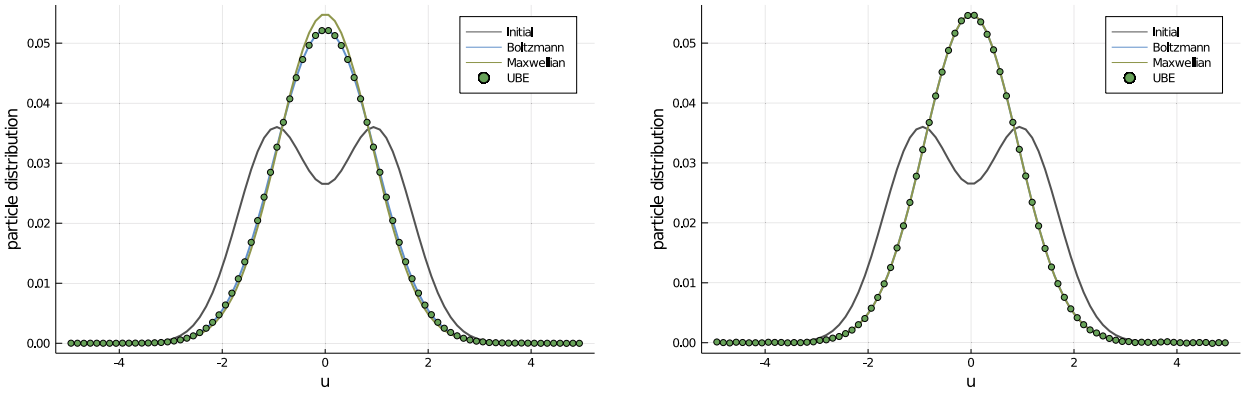
$$\frac{\rho_+}{\rho_-} = \frac{(\gamma+1)\text{Ma}^2}{(\gamma-1)\text{Ma}^2+2},$$



(a) $t = 0.1$ (b) $t = 0.3$

(c) $t = 0.5$ (d) $t = 0.7$

Fig. 5. Particle distribution functions at interpolating time instants outside the training set of the homogeneous relaxation problem.

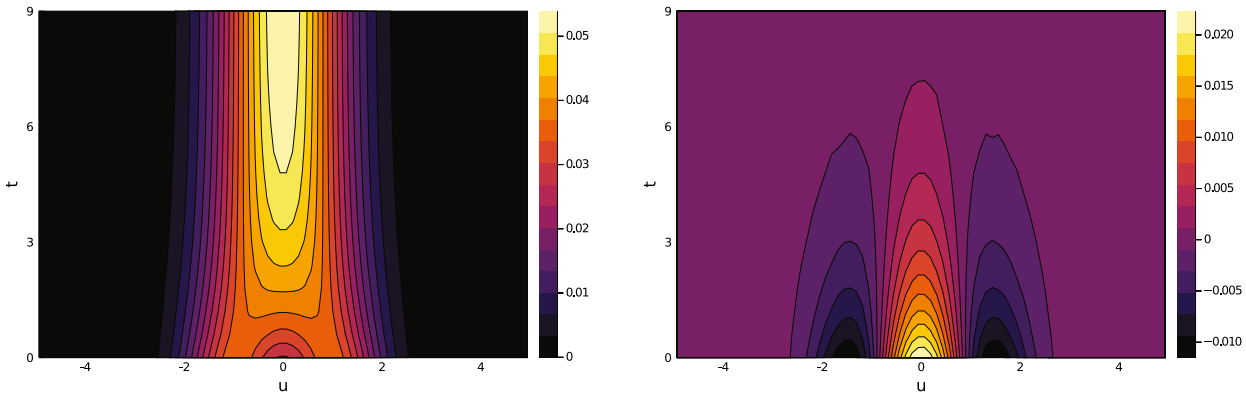


(a) $t = 6.0$ (b) $t = 9.0$

Fig. 6. Particle distribution functions at extrapolating time instants outside the training set of the homogeneous relaxation problem.

$$\frac{U_+}{U_-} = \frac{(\gamma-1)\mathrm{Ma}^2+2}{(\gamma+1)\mathrm{Ma}^2},$$
$$\frac{T_+}{T_-} = \frac{((\gamma-1)\mathrm{Ma}^2+2)(2\gamma\mathrm{Ma}^2-\gamma+1)}{(\gamma+1)^2\mathrm{Ma}^2},$$

where γ is the ratio of specific heat. The upstream and downstream density, velocity and temperature are denoted with $\{\rho_-, U_-, T_-\}$ and $\{\rho_+, U_+, T_+\}$. The computational setup for this case is presented in Table 3.



(a) Particle distribution function (b) Collision term

Fig. 7. Particle distribution functions and collision terms over phase space $\{t, u\}$ in the homogeneous relaxation problem.

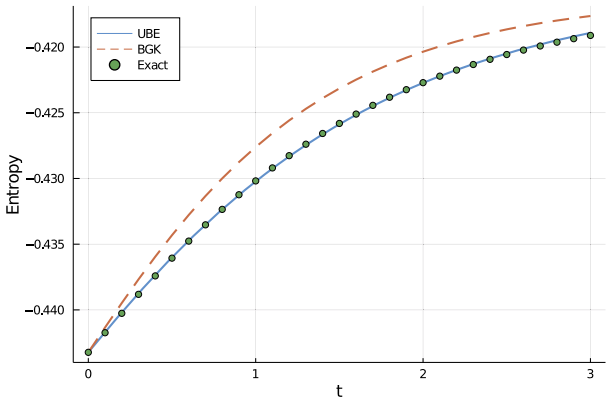


Fig. 8. Time evolution of entropy in the homogeneous relaxation problem.

Table 3
Computational setup in normal shock structure.

x	N_x	u	N_u	Quadrature	Kn	Pr
$[-25, 25]$	80	$[-5, 5]$	80	rectangle	1	2/3
μ_{ref}	α	ω	CFL	Integral	Layer	Optimizer
0.554	1.0	0.5	0.7	Midpoint	Dense	ADAM

We are only concerned about the one-dimensional profile of flow variables. Similar as the homogeneous relaxation problem, two reduced distribution functions can be introduced to conduct dimension reduction,

$$h(t, x, u) = \iint f(t, x, u, v, w) dv dw,$$
$$b(t, x, u) = \iint (v^2 + w^2) f(t, x, u, v, w) dv dw,$$

and the corresponding universal Boltzmann equations become,

$$\frac{\partial h}{\partial t} + u \frac{\partial h}{\partial x} = \nu(\mathcal{M}_h - h) + \text{NN}_\theta(\mathcal{M}_h - h),$$
$$\frac{\partial b}{\partial t} + u \frac{\partial b}{\partial x} = \nu(\mathcal{M}_b - h) + \text{NN}_\theta(\mathcal{M}_b - h).$$

The neural network chain consists of an input layer which accepts h and b with $(N_u \times 2)$ neurons, three hidden dense layers with $(N_u \times 2 \times 16)$ neurons each, tanh as the activation function and the output layer that is of the same shape with input. For the current steady-state problem, BGK equation is employed first to evolve the flow field from initial jump condition, from which we extract particle distribution functions at different time instants. The fast spectral method with midpoint rule

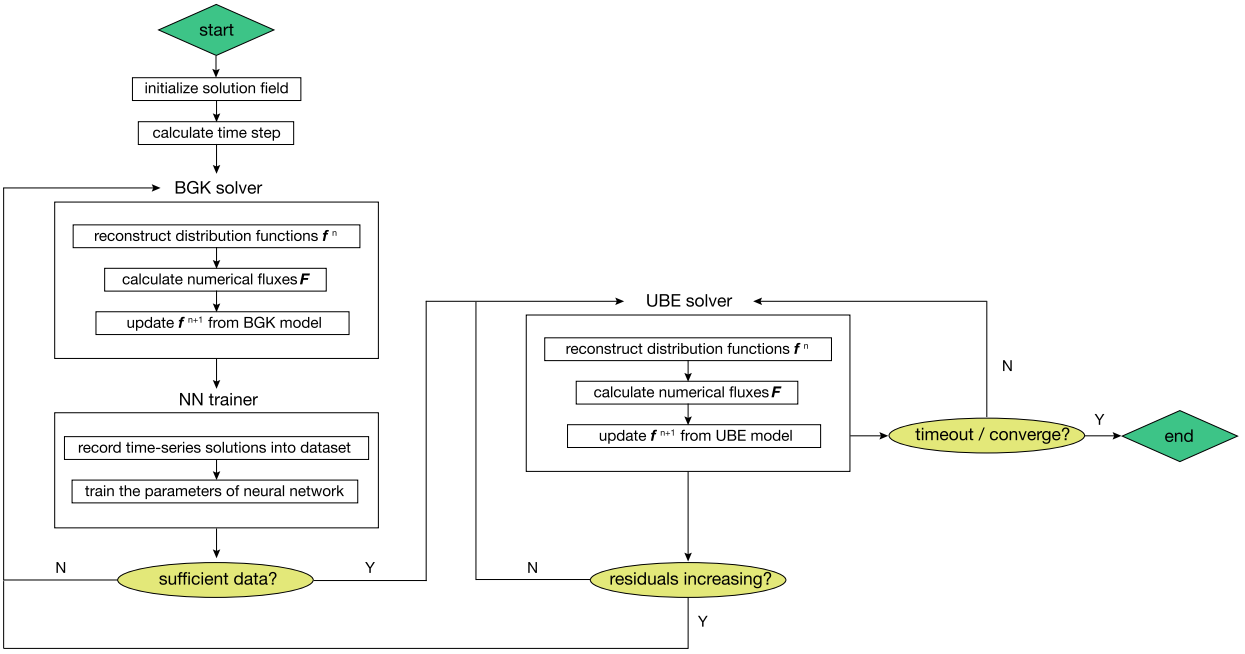


Fig. 9. Training process and solution algorithm of the universal Boltzmann equation.

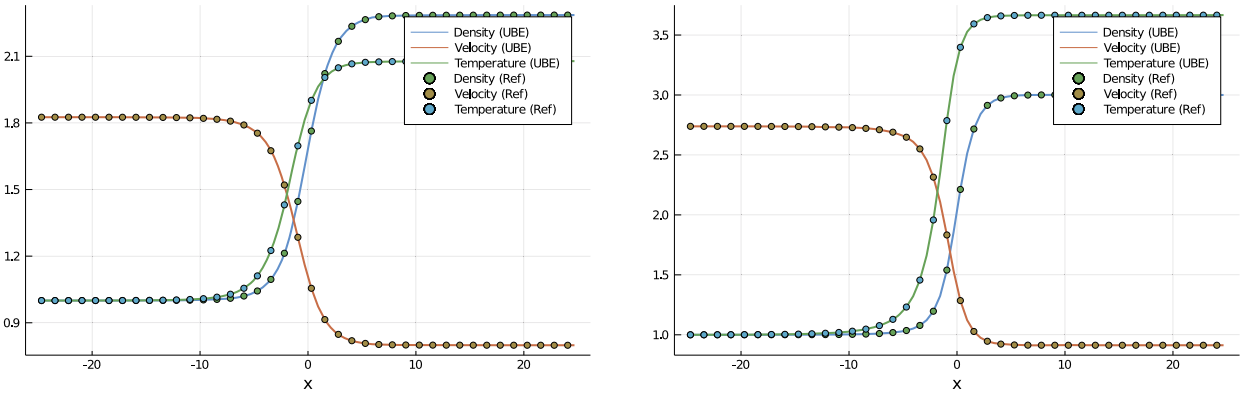


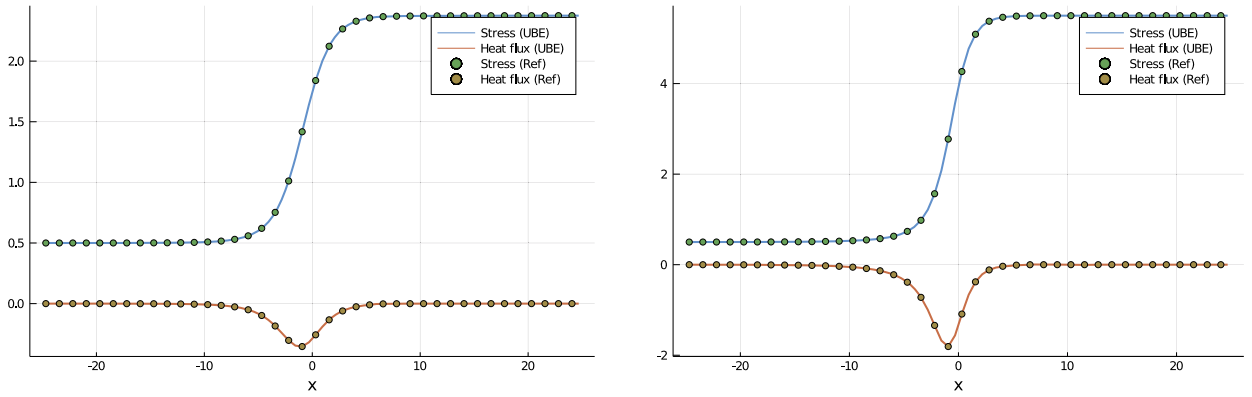
Fig. 10. Gas density, velocity and temperature profiles at different Mach numbers in the normal shock wave problem.

is employed to solve the Boltzmann collision integral and generates training data in time series. In this case five tracing solution points are recorded within each time step. Thereafter, the same algorithm is used to solve the UBE and produces data points at the same time instants as training set. The loss function is evaluated through mean squared error,

$$L(\theta) = \sum_{i,j,n}(h_\theta - h_{\text{train}})^2(t^n, x_i, u_j) + \sum_{i,j,n}(b_\theta - b_{\text{train}})^2(t^n, x_i, u_j) + \zeta \sum_{l=1}^{3} ||\theta^{(l)}||^2.$$

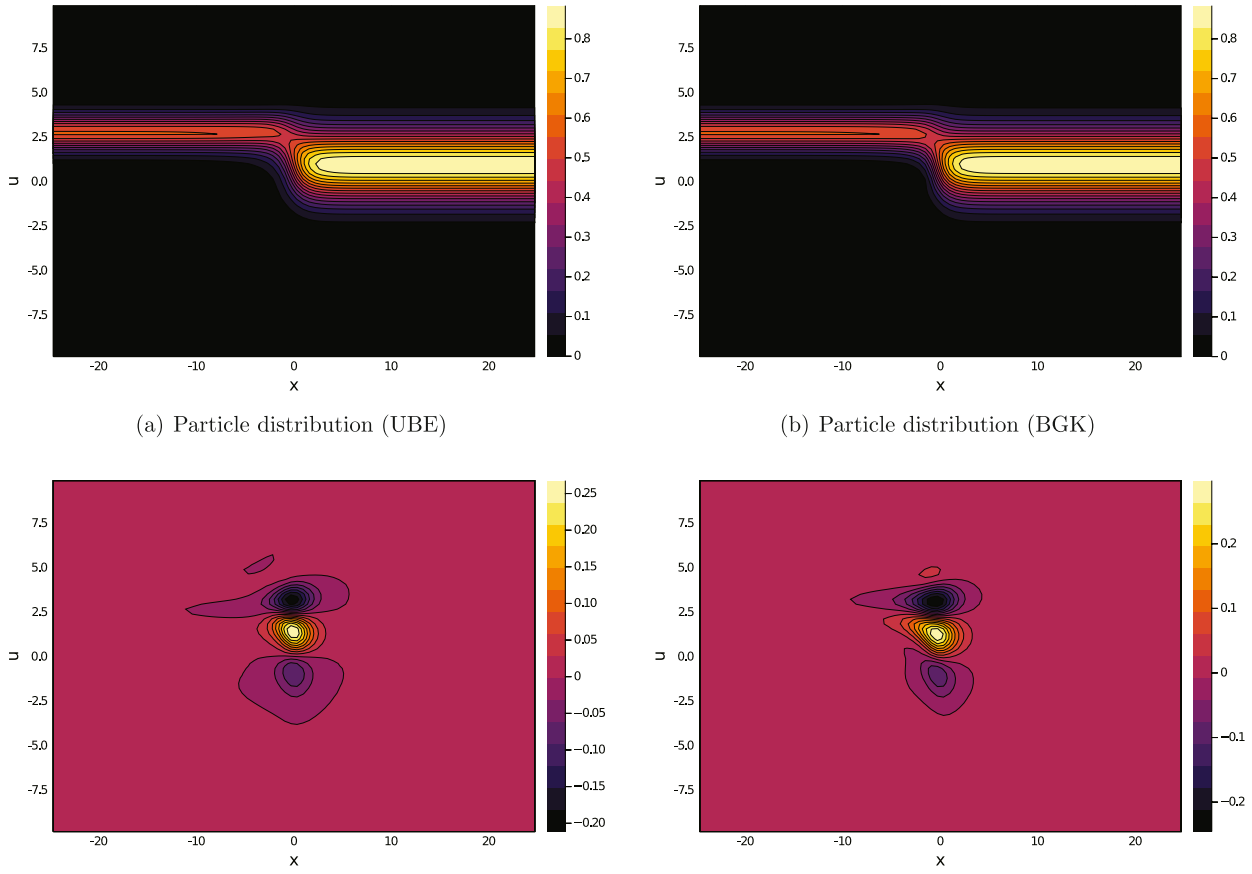
In the numerical simulation, the training and solving processes are handled in a coupled way. First, the training set consists of 100 equally distributed data set (extracted every 20 time stepping and covers the entire physical domain). After the training process, we utilize the UBE solver to continue the simulation. However, if the residuals of flow variables keep increasing within a successive 20 time steps, we downgrade the generalization performance of the current neural network. To overcome the overfitting of existing training data, the particle distribution functions at current time step will be extracted and added into training set to conduct parameter retraining. The detailed training approach and solution algorithm are illustrated in Fig. 9.

Fig. 10 presents the profiles of gas density, velocity and temperature along x direction, and Fig. 11 provides the distributions of stress P_{xx} and heat flux q. As is shown, the current UBE provides equivalent solutions as reference results. Fig. 12 presents the contours of reduced particle distribution functions h and collision term $\nu(\mathcal{M}_h - h)$ in the convergent state.



(a) Ma = 2 (b) Ma = 3

Fig. 11. Stress and heat flux at different Mach numbers in the normal shock wave problem.



(a) Particle distribution (UBE) (b) Particle distribution (BGK)

(c) Collision term (UBE) (d) Collision term (BGK)

Fig. 12. Contours of reduced distribution functions and collision terms at Ma = 3 in the normal shock structure problem.

In spite of the similar patterns of particle distributions, obvious difference between UBE and BGK solution can be observed from the distribution of collision terms over the phase space $\{x, u\}$. Table 4 lists the detailed computational cost for evaluating collision term inside each cell, and Table 5 presents the corresponding mean computational time with different ODE solvers. Due to the data transmission through neurons, the UBE is far more efficient than the fast spectral method (FSM) for Boltzmann integral from higher dimensions in phase space.

Table 4
Computational cost for evaluating the right-hand side of the Boltzmann equation in the normal shock structure problem.

	MEM	n_{allocs}	t_{min}	t_{median}	t_{mean}	t_{max}	n_{samples}
UBE	4.78 MB	25	412.69 µs	502.26 µs	508.88 µs	547.00 µs	8008
FSM	214.42 MB	1839	66.14 ms	71.82 ms	71.55 ms	73.00 ms	170

Table 5
Mean computational time for solving the right-hand side of the Boltzmann equation with different solvers in the normal shock structure problem.

	Euler	Midpoint	BS3	RK4	Tsit5
UBE	3.39 ms	15.73 ms	17.66 ms	37.29 ms	37.08 ms
FSM	211.04 ms	369.22 ms	443.07 ms	669.31 ms	669.71 ms

Table 6
Computational setup of lid-driven cavity.

x	y	N_x	N_y	u	v	N_u
[0, 1]	[0, 1]	45	45	[−5, 5]	[−5, 5]	24
N_v	Quadrature	Kn	Pr	μ_{ref}	α	ω
24	Rectangular	0.075	0.67	0.042	1.0	0.5
CFL	Integrator	Boundary	Layer	Activation	Optimizer	
0.8	Midpoint	Maxwell	Dense	tanh	ADAM	

5.3. Lid-driven cavity

We then test the performance of current method with multi-dimensional geometry. The lid-driven cavity is employed for the numerical experiment. The rectangular domain is enclosed by four solid walls with equal temperature $T_w = 1$. The upper wall moves in the tangent direction with $\mathbf{V}_w = [0.15, 0]^T$, and the rest three walls stay still. Maxwell's diffusive boundary is adopted to all the walls. The initial particle distribution function is set as Maxwellian corresponding to the uniform fluid, i.e.,

$$\begin{bmatrix} \rho \\ U \\ V \\ T \end{bmatrix}_{t=0} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Here we are concerned about two-dimensional distributions of flow variables, and thus the reduced distribution function is introduced as

$$h(t, x, u, v) = \iint f(t, x, u, v, w)dw,$$

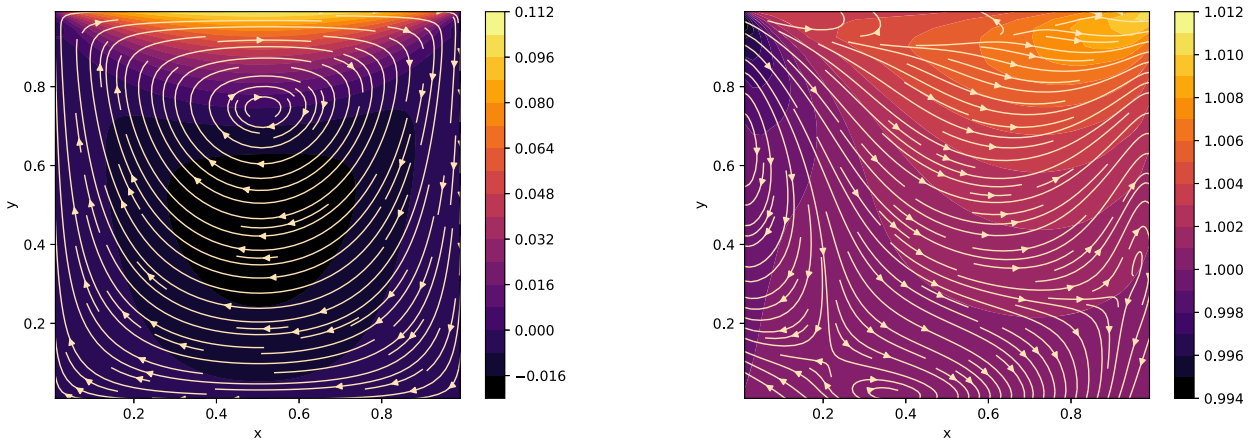
and the corresponding universal Boltzmann equations become,

$$\frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} + v\frac{\partial h}{\partial y} = \nu(\mathcal{M}_h - h) + \text{NN}_\theta(\mathcal{M}_h - h).$$

The neural network chain consists of an input layer which accepts h with $(N_u \times N_v)$ neurons, two hidden dense layers with $(N_u \times N_v \times 8)$ neurons each, and an output layer that is of the same shape with input. The detailed computational setup is provided in Table 6.

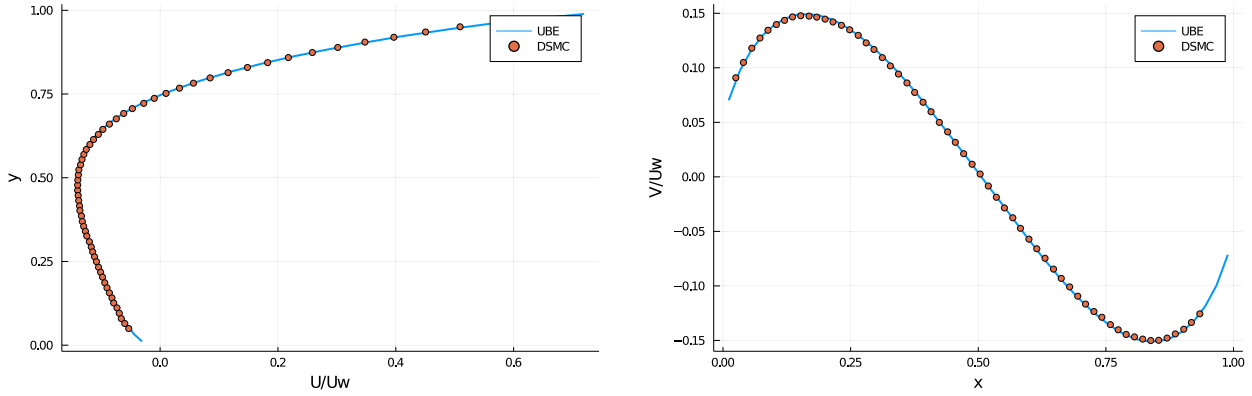
Similar as the shock wave problem, the BGK equation is employed first to evolve the flow field from initial homogeneity. The particle distribution functions at different time instants are extracted as the initial status of the Boltzmann collision integral, which produce time-series dataset with the midpoint rule. The loss function is the same as that described in the section 5.2. The unified solution algorithm to solve the UBE and to append the training set follows the diagram in Fig. 9.

Fig. 13 presents the contours of U-velocity with streamlines and of temperature with heat flux vectors. As illustrated in [57], the anti-Fourier's heat flux driven by stress is clearly identified. Fig. 14 shows the velocity profiles along the vertical and horizontal central lines of the cavity. The DSMC solutions with 60×60 physical mesh are plotted as benchmark. The quantitative comparison demonstrates that the current approach is able to provide equivalent DSMC solutions. Fig. 15–17 presents the collision operators from the UBE and the pure BGK term. Although the difference is not as great as inner shock wave, the corrective effect of the neural network is clearly identified, which provide a Prandtl number fix at molecular level for the BGK system. Table 7 lists the detailed computational cost for evaluating collision term inside each cell. Due to the massive data transmission through neurons, the UBE calculation costs few more resources than soving the Shakhov



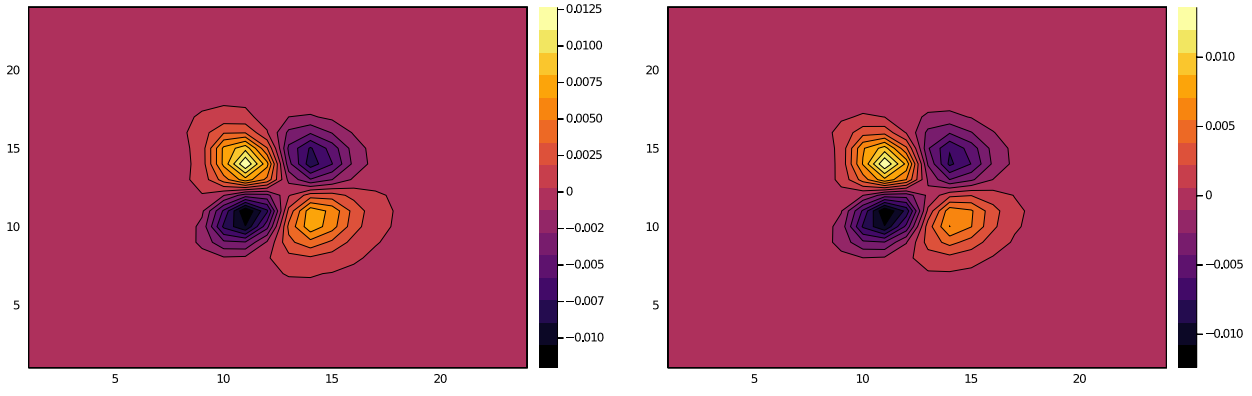
(a) U-velocity (b) Temperature

Fig. 13. Contours of U-velocity with streamlines and temperature with heat flux vectors inside the cavity.



(a) U-velocity (b) Temperature

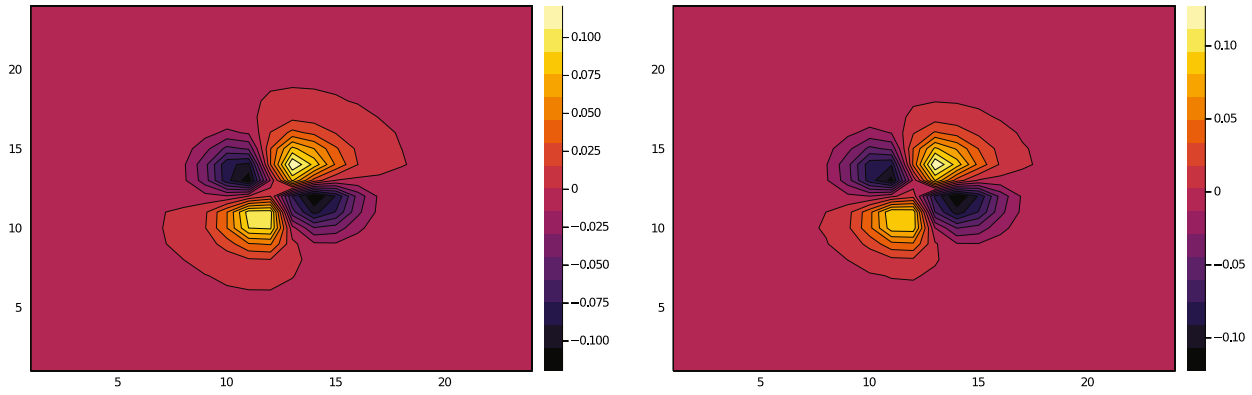
Fig. 14. Velocity profiles along vertical and horizontal central lines inside the cavity.



(a) Collision term (UBE) (b) Collision term (BGK)

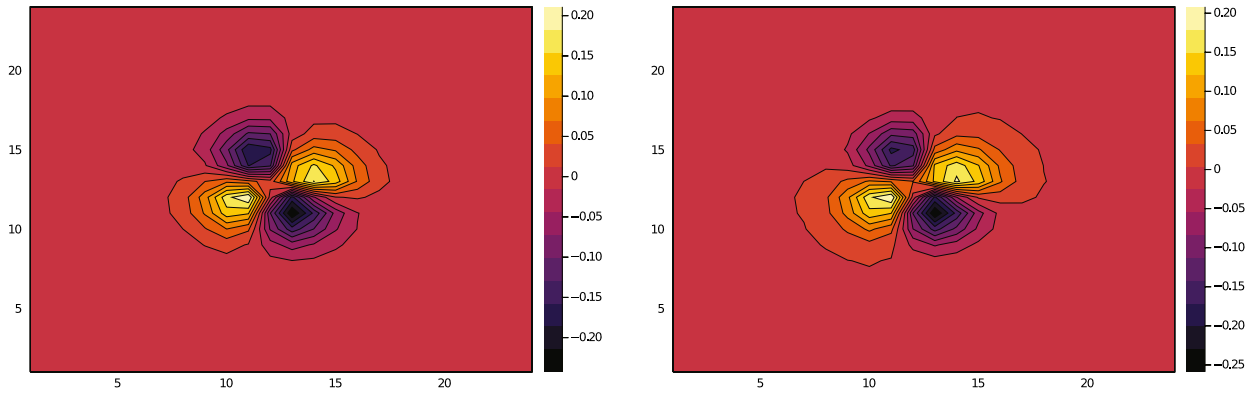
Fig. 15. Contours of collision terms at the center point of cavity.

term directly, but is still much more efficient than the fast spectral method (FSM) for the Boltzmann integral from higher dimensions in phase space. Also, with the direct matrix manipulation possessed in neural network, fewer allocations, e.g. the collision frequency, are needed to evaluate collision terms.



(a) Collision term (UBE) (b) Collision term (BGK)

Fig. 16. Contours of collision terms at the topleft point of cavity.



(a) Collision term (UBE) (b) Collision term (BGK)

Fig. 17. Contours of collision terms at the topright point of cavity.

Table 7
Computational cost for evaluating the right-hand side of the Boltzmann equation in the lid-driven cavity problem.

	MEM	n_{allocs}	t_{min}	t_{median}	t_{mean}	t_{max}	n_{samples}
UBE	52.72 KB	24	459.22 µs	524.73 µs	710.88 µs	4.97 ms	6977
Shakhov	40.97 KB	62	13.17 µs	15.00 µs	16.59 µs	176.58 µs	10000
FSM	68.8 MB	212572	27.83 ms	28.21 ms	30.32 ms	35.08 ms	166

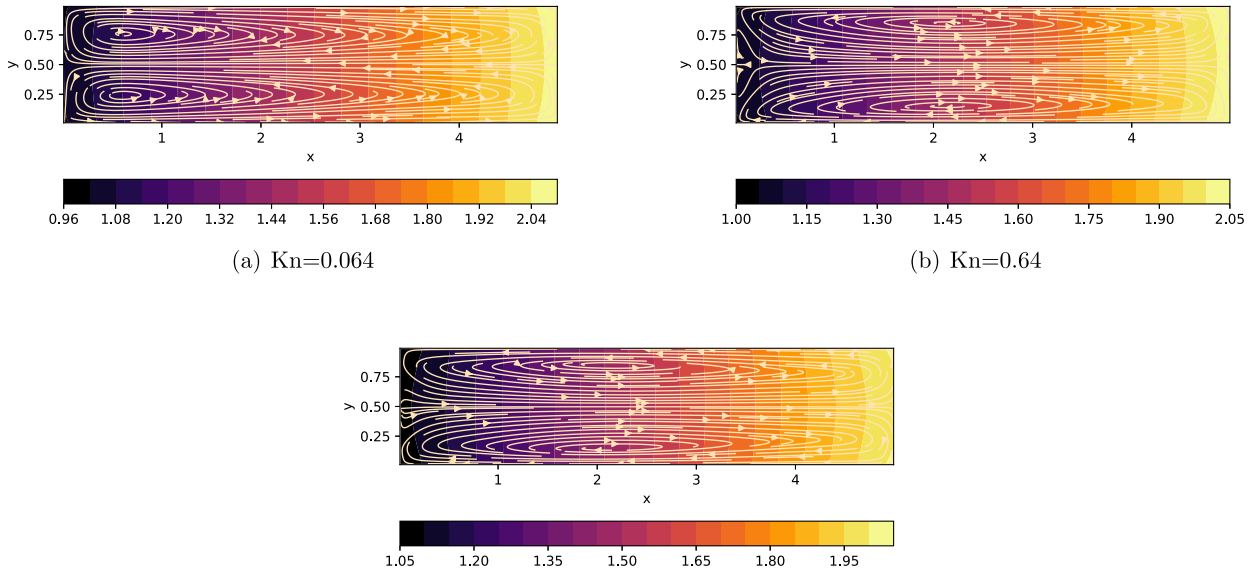
5.4. Thermal creep

The last test case is the thermal creep problem. It is a typical non-equilibrium flow of rarefied gas induced by temperature gradient. In this case, we follow the setup given in [58]. A sealed two-dimensional channel is enclosed by four solid walls. The left and right ends hold different temperatures as $T_L = 273K$ and $T_R = 573K$, and the upper and lower walls show a linear temperature distribution in between. The initial pressure is equal to a unit of atmospheric pressure, and thus the molecular mean free path is $\ell = 64nm$. The initial temperature of gas is set to be equal to the wall temperature at the same horizontal location. The particle distribution functions are the Maxwellian with respect to local macroscopic variables. We consider different channel widths and the reference Knudsen numbers change correspondingly. The schematic of the problem is shown in Fig. 2, and the detailed computational setup can be found in Table 8.

Similar as the previous problem, the BGK equation is employed first to evolve the flow field from initial status. The particle distribution functions at different time instants are extracted and play as the initial value of homogeneous Boltzmann equation. The explicit Euler method is employed to generate the time-series dataset from the homogeneous Boltzmann equation. The loss function is the same as that described in the section 5.2. The unified solution algorithm to generate the training set and to solve the UBE follows the diagram in Fig. 9

Table 8
Computational setup of thermal creep problem.

x	y	N_x	N_y	u	v
$[0, 5]$	$[0, 1]$	200	40	$[-5, 5]$	$[-5, 5]$
Quadrature	Kn	Pr	α	ω	CFL
Nonuniform	[0.064, 10]	0.67	1.0	0.5	0.8
Integrator	Boundary	Layer	Activation	Optimizer	
Euler	Maxwell	Dense	tanh	ADAM	



(a) Kn=0.064 (b) Kn=0.64

(c) Kn=3.2

Fig. 18. Contours of temperature with streamlines in the thermal creep problem.

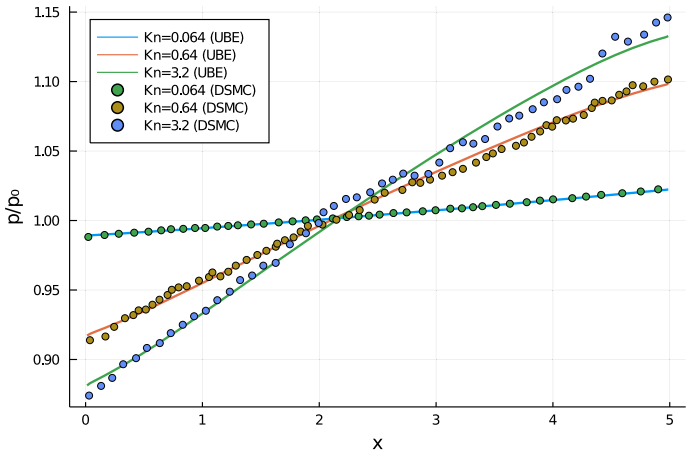
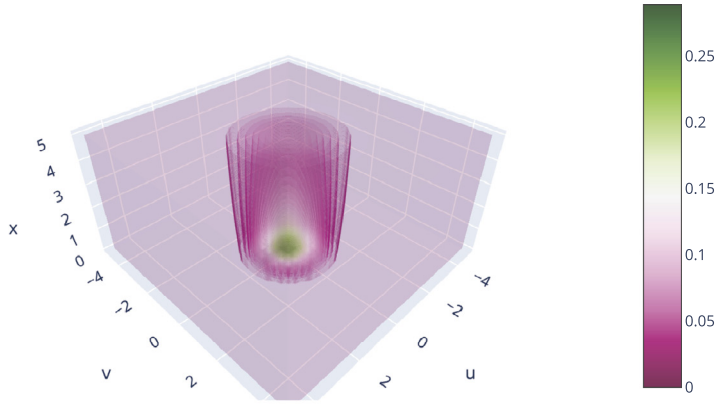
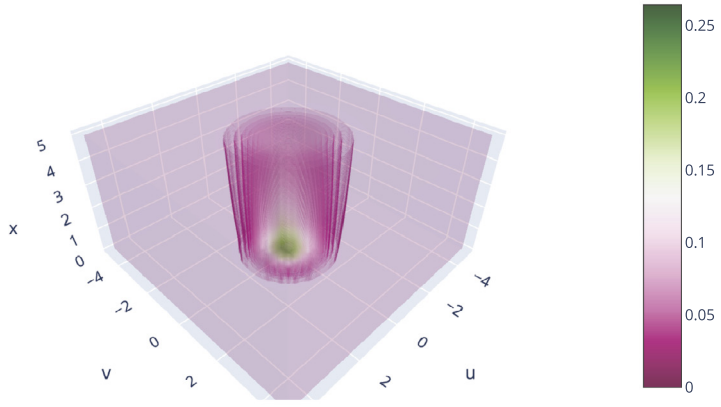


Fig. 19. Pressure distributions along the horizontal center line in the thermal creep problem.

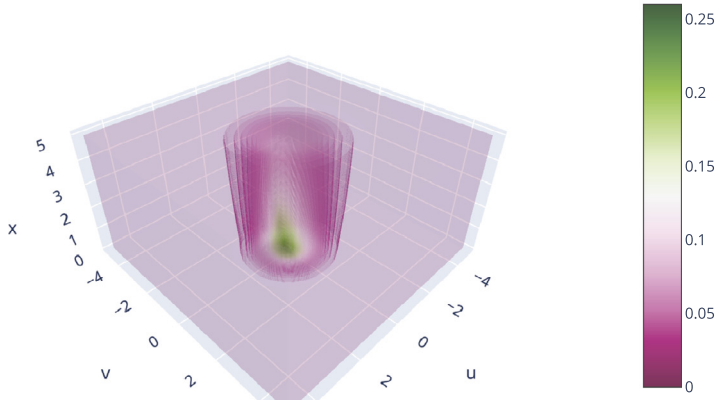
Fig. 18 presents the contours of temperature with streamlines inside the channel at different reference Knudsen numbers with Kn = 0.064, 0.64, 3.2. The algebraic non-uniform numerical quadrature [59] with 48, 64, and 96 points is used in u and v directions respectively. As can be seen, with different rarefaction, the flow patterns change dramatically. Fig. 19 provides the pressure distribution along the horizontal center line of the channel. The DSMC solutions with the same geometry resolution are plotted as reference. As is shown, the current approach provides the equivalent and noise-free solutions as DSMC. Fig. 20 draws the corresponding particle distribution functions along the horizontal center line, and Fig. 21 shows the collision operators from the UBE and the pure BGK term. The corrective effect from the neural network can be clearly seen, especially in the highly rarefied regime.



(a) Kn=0.064



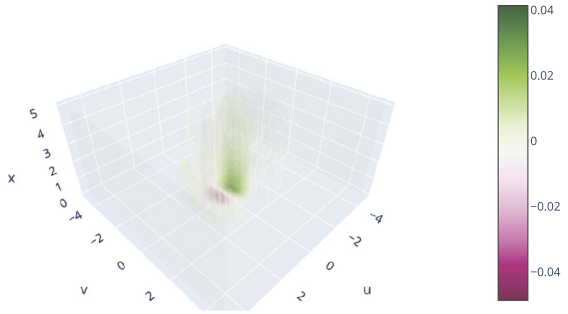
(b) Kn=0.64



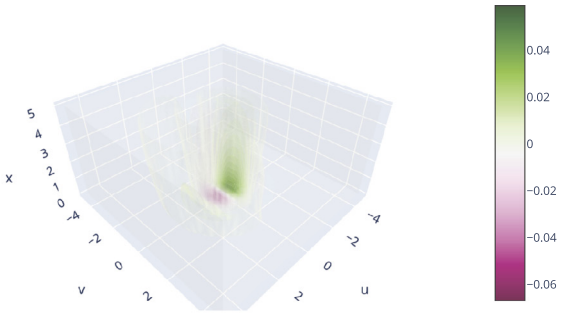
(c) Kn=3.2

Fig. 20. Particle distribution functions along the horizontal center line of channel at different reference Knudsen numbers.

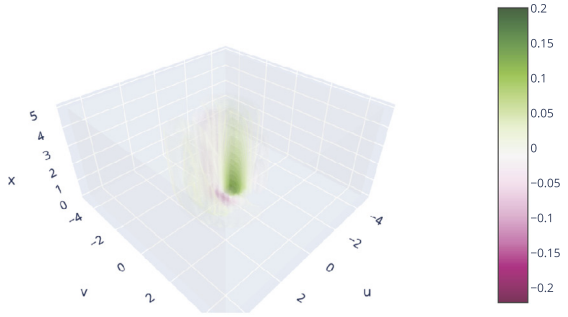
To further verify the current setup of physical mesh and numerical quadrature, we simulate the thermal creep case in [59] as well. The computational setup remains consistent, except that we now simulate the Argon gas with the right-hand temperature being twice as high as the left end with $T_R = 2T_L$. The initial uniform gas holds equal temperature as the left wall. Fig. 22 and 23 compare the U-velocity profiles along horizontal and vertical central lines at $\mathrm{Kn} = 0.08$ and $\mathrm{Kn} = 10$. The grid-convergence test in highly rarefied gaseous flow illustrates that the current quadrature setups are reasonable.



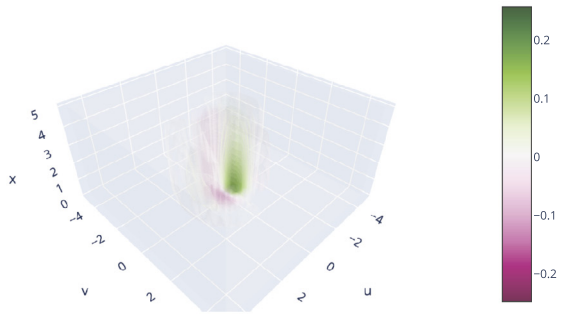
(a) Kn=0.064 (UBE)



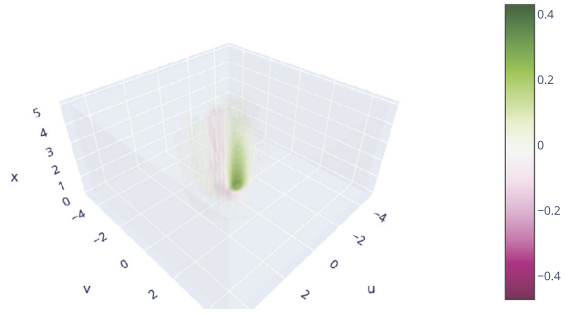
(b) Kn=0.064 (BGK)



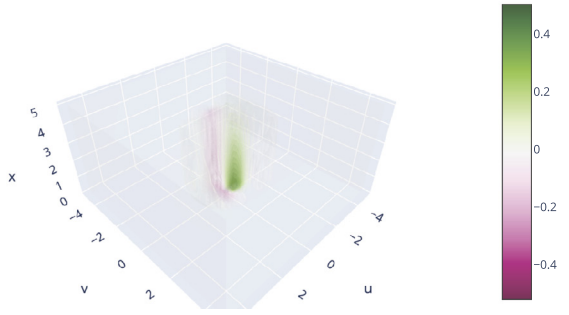
(c) Kn=0.64 (UBE)



(d) Kn=0.64 (BGK)



(e) Kn=3.2 (UBE)

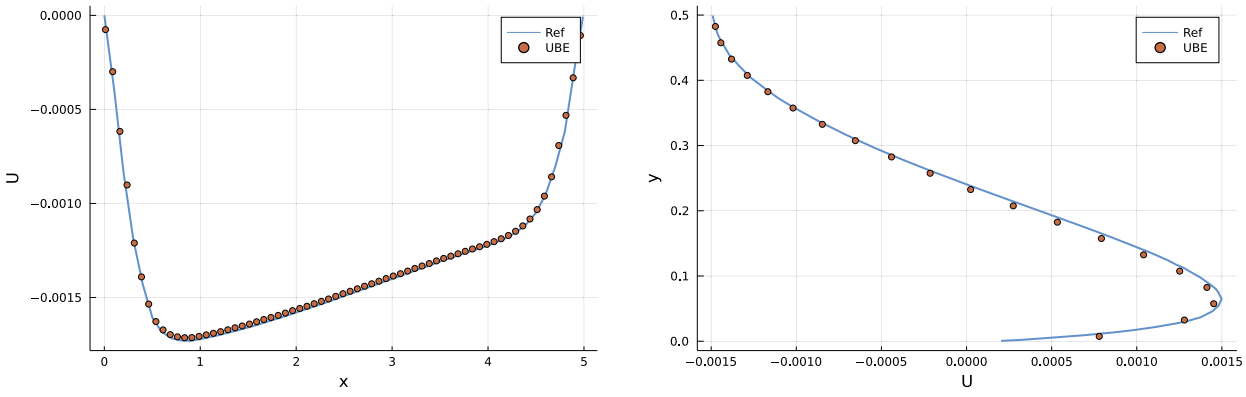


(f) Kn=3.2 (BGK)

Fig. 21. Collision operators along the horizontal center line of channel at different reference Knudsen numbers.

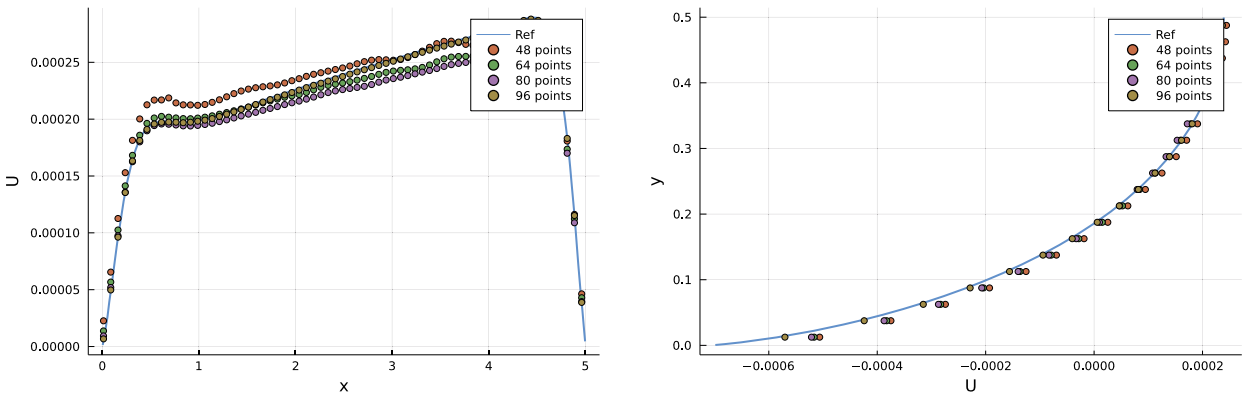
6. Conclusion

Deep learning offers another possibility for the future development of scientific modeling and simulation. In this paper, we hybridize mechanical and neural modelings in the context of gas dynamics and present a neural network-enhanced universal Boltzmann equation (UBE). The complicated fivefold Boltzmann integral is replaced by the neural network surrogation, which forms a differentiable framework that can be trained and solved via source-to-source automatic differentiation and various differential equation solvers. The proposed neural differential equation maintains a well balance of interpretability from deep physical insight and flexibility from deep learning techniques. The asymptotic limit of the UBE in the hydrodynamic limit is preserved independent of the neural network parameters. The solution algorithm for the UBE is provided, and numerical experiments of both spatially uniform and non-uniform cases are presented to validate the current modeling and simulation approach. The universal Boltzmann equation method enjoys considerable potential to be further extended to complex systems with nonelastic collisions [60], real-gas effects [61], chemical reactions [62], uncertainty quantification [63,64], etc.



(a) Horizontal (b) Vertical

Fig. 22. Velocity profiles along vertical and horizontal central lines inside the channel at Kn = 0.08 with 48 quadrature points.



(a) Horizontal (b) Vertical

Fig. 23. Velocity profiles along vertical and horizontal central lines inside the channel at Kn = 10 with grid-convergent quadrature settings.

CRediT authorship contribution statement

Tianbai Xiao: Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Software, Visualization, Writing – original draft, Writing – review & editing. **Martin Frank:** Conceptualization, Formal analysis, Methodology, Project administration, Resources, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We acknowledge the help from Dr. Christopher Rackauckas in the open-source community of scientific machine learning, and the fruitful discussion with Steffen Schotthöfer. The current research is funded by the Alexander von Humboldt Foundation (Ref. 3.5-CHN-1210132-HFST-P).

References

[1] Mark Nixon, Alberto Aguado, Feature Extraction and Image Processing for Computer Vision, Academic press, 2019.
[2] K.R. Chowdhary, Natural language processing, in: Fundamentals of Artificial Intelligence, Springer, 2020, pp. 603–649.
[3] Eric Mjolsness, Dennis DeCoste, Machine learning for science: state of the art and future prospects, Science 293 (5537) (2001) 2051–2055.
[4] E. Isaac Lagaris, Aristidis Likas, Dimitrios I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[5] Justin Sirignano, Konstantinos Spiliopoulos Dgm, A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.

[6] E. Weinan, Bing Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[7] Jens Berg, Kaj Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.
[8] Jiequn Han, Arnulf Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. USA 115 (34) (2018) 8505–8510.
[9] Zeyu Liu, Yantao Yang, Qingdong Cai, Neural network as a function approximator and its application in solving differential equations, Appl. Math. Mech. 40 (2) (2019) 237–248.
[10] Maziar Raissi, Paris Perdikaris, George 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.
[11] Luning Sun, Han Gao, Shaowu Pan, Jian-Xun Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Eng. 361 (2020) 112732.
[12] Xiaowei Jin, Shengze Cai, Hui Li, George Em Karniadakis, NSFnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations, preprint, arXiv:2003.06496, 2020.
[13] Zhiping Mao, Ameya D. Jagtap, George Em Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Eng. 360 (2020) 112789.
[14] Hui Xu, Wei Zhang, Yong Wang, Explore missing flow dynamics by physics-informed deep learning: the parameterised governing systems, preprint, arXiv:2008.12266, 2020.
[15] Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E. Hurtado, Ellen Kuhl, Physics-informed neural networks for cardiac activation mapping, Front. Phys. 8 (2020) 42.
[16] Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R. Witschey, John A. Detre, Paris Perdikaris, Machine learning in cardiovascular flows modeling: predicting arterial blood pressure from non-invasive 4d flow MRI data using physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 358 (2020) 112623.
[17] Zhiwei Fang, Justin Zhan, Deep physical informed neural networks for metamaterial design, IEEE Access 8 (2019) 24506–24513.
[18] Dehao Liu, Yan Wang, Multi-fidelity physics-constrained neural network and its application in materials modeling, J. Mech. Des. 141 (12) (2019).
[19] Yuyao Chen, Lu Lu, George Em Karniadakis, Luca Dal Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, Opt. Express 28 (8) (2020) 11618–11633.
[20] Yibo Yang, Paris Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, J. Comput. Phys. 394 (2019) 136–152.
[21] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, Paris Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, J. Comput. Phys. 394 (2019) 56–81.
[22] Yuehaw Khoo, Lexing Ying, Switchnet: a neural network model for forward and inverse scattering problems, SIAM J. Sci. Comput. 41 (5) (2019) A3182–A3201.
[23] Yuwei Fan, Lin Lin, Lexing Ying, Leonardo Zepeda-Núnez, A multiscale neural network based on hierarchical matrices, Multiscale Model. Simul. 17 (4) (2019) 1189–1213.
[24] Yingzhou Li, Jianfeng Lu, Anqi Mao, Variational training of neural network approximations of solution maps for physical models, J. Comput. Phys. 409 (2020) 109338.
[25] Jiequn Han, Linfeng Zhang, Roberto Car, et al., Deep potential: a general representation of a many-body potential energy surface, preprint, arXiv: 1707.01478, 2017.
[26] Maziar Raissi, Alireza Yazdani, George Em Karniadakis, Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[27] Pat Langley, Data-driven discovery of physical laws, Cogn. Sci. 5 (1) (1981) 31–54.
[28] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (4) (2017) e1602614.
[29] Maziar Raissi, Paris Perdikaris, George Em Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, preprint, arXiv:1801.01236, 2018.
[30] Jun Zhang, Wenjun Ma, Data-driven discovery of governing equations for fluid dynamics based on molecular simulation, J. Fluid Mech. 892 (2020).
[31] Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA 113 (15) (2016) 3932–3937.
[32] Struchtrup Henning, Macroscopic transport equations for rarefied gas flows, in: Macroscopic Transport Equations for Rarefied Gas Flows, Springer, 2005, pp. 145–160.
[33] Clément Mouhot, Lorenzo Pareschi, Fast algorithms for computing the Boltzmann collision operator, Math. Comput. 75 (256) (2006) 1833–1852.
[34] Tianbai Xiao, Kun Xu, Qingdong Cai, A unified gas-kinetic scheme for multiscale and multicomponent flow transport, Appl. Math. Mech. 40 (3) (2019) 355–372.
[35] Balázs Csanád Csáji, et al., Approximation with artificial neural networks, Faculty of Sciences, Etvs Lornd University, Hungary 24(48), 7, 2001.
[36] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, Universal differential equations for scientific machine learning, preprint, arXiv:2001.04385, 2020.
[37] Carlo Cercignani, The Boltzmann Equation and Its Applications, Springer, 1988.
[38] Prabhu Lal Bhatnagar, Eugene P. Gross, Max 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.
[39] E.M. Shakhov, Generalization of the Krook kinetic relaxation equation, Fluid Dyn. 3 (5) (1968) 95–96.
[40] Ricky T.Q. Chen, Yulia Rubanova, Jesse Bettencourt, David K. Duvenaud, Neural ordinary differential equations, in: Advances in Neural Information Processing Systems, 2018, pp. 6571–6583.
[41] Lars Ruthotto, Eldad Haber, Deep neural networks motivated by partial differential equations, J. Math. Imaging Vis. (2019) 1–13.
[42] Sydney Chapman, Thomas George Cowling, The Mathematical Theory of Non-Uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases, Cambridge University Press, 1970.
[43] Tianbai Xiao, Chang Liu, Kun Xu, Qingdong Cai, A velocity-space adaptive unified gas kinetic scheme for continuum and rarefied flows, J. Comput. Phys. (2020) 109535.
[44] Tianbai Xiao, Kun Xu, Qingdong Cai, Tiezheng Qian, An investigation of non-equilibrium heat transport in a gas system under external force field, Int. J. Heat Mass Transf. 126 (2018) 362–379.
[45] Tianbai Xiao, Qingdong Cai, Kun Xu, A well-balanced unified gas-kinetic scheme for multiscale flow transport under gravitational field, J. Comput. Phys. 332 (2017) 475–491.
[46] Mike Innes, Alan Edelman, Keno Fischer, Christopher Rackauckas, Elliot Saba, Viral B. Shah, Will Tebbutt, A differentiable programming system to bridge machine learning and scientific computing, CoRR, arXiv:1907.07587 [abs], 2019.

[47] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., Tensorflow: a system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 16, 2016, pp. 265–283.
[48] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, Adam Lerer, Automatic Differentiation in Pytorch, 2017.
[49] Diederik P. Kingma, Jimmy Ba Adam, A method for stochastic optimization, preprint, arXiv:1412.6980, 2014.
[50] Yu Nesterov, Efficiency of coordinate descent methods on huge-scale optimization problems, SIAM J. Optim. 22 (2) (2012) 341–362.
[51] Jorge Nocedal, Stephen Wright, Numerical Optimization, Springer Science & Business Media, 2006.
[52] Lei Wu, Craig White, Thomas J. Scanlon, Jason M. Reese, Yonghao Zhang, Deterministic numerical solutions of the Boltzmann equation using the fast spectral method, J. Comput. Phys. 250 (2013) 27–52.
[53] Tianbai Xiao, Kinetic.jl: A portable finite volume toolbox for scientific and neural computing, J. Open Source Softw. 6 (1) (2021) 3060.
[54] Christopher Rackauckas, Qing Nie, DifferentialEquations. jl–a performant and feature-rich ecosystem for solving differential equations in Julia, J. Open Res. Softw. 5 (1) (2017).
[55] Lawrence F. Shampine, Implementation of Rosenbrock methods, ACM Trans. Math. Softw. 8 (2) (1982) 93–113.
[56] Ch Tsitouras, I. Th Famelis, T.E. Simos, On modified Runge–Kutta trees and methods, Comput. Math. Appl. 62 (4) (2011) 2101–2111.
[57] John Benzi, Xiao-Jun Gu, David R. Emerson, Effects of incomplete surface accommodation on non-equilibrium heat transfer in cavity flow: a parallel DSMC study, Comput. Fluids 45 (1) (2011) 197–201.
[58] Nathan D. Masters, Wenjing Ye, Octant flux splitting information preservation DSMC method for thermally driven flows, J. Comput. Phys. 226 (2) (2007) 2044–2062.
[59] Lei Wu, Jason M. Reese, Yonghao Zhang, Solving the Boltzmann equation deterministically by the fast spectral method: application to gas microflows, J. Fluid Mech. 746 (2014) 53–84.
[60] V. Garzó, J.W. Dufty, Dense fluid transport for inelastic hard spheres, Phys. Rev. E 59 (5) (1999) 5895.
[61] Céline Baranger, Yann Dauvois, Gentien Marois, Jordane Mathé, Julien Mathiaud, Luc Mieussens, A BGK model for high temperature rarefied gas flows, Eur. J. Mech. B, Fluids 80 (2020) 1–12.
[62] M. Groppi, G. Spiga, Kinetic approach to chemical reactions and inelastic transitions in a rarefied gas, J. Math. Chem. 26 (1–3) (1999) 197–219.
[63] Tianbai Xiao, Martin Frank, A stochastic kinetic scheme for multi-scale flow transport with uncertainty quantification, J. Comput. Phys. 437 (2021) 110337.
[64] Tianbai Xiao, Martin Frank, A stochastic kinetic scheme for multi-scale plasma transport with uncertainty quantification, J. Comput. Phys. 432 (2021) 110139.