

Solving continuum and rarefied flows using differentiable programming

Tianbai Xiao  ^{a,b,*}

^a Institute of Mechanics, State Key Laboratory of High Temperature Gas Dynamics and Centre for Interdisciplinary Research in Fluids, Chinese Academy of Sciences, Beijing, China

^b School of Engineering Science, University of Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Keywords:

Computational fluid dynamics
Boltzmann equation
Kinetic theory
Scientific machine learning
Differentiable programming

ABSTRACT

Accurate and efficient prediction of multi-scale flows remains a formidable challenge. Constructing theoretical models and numerical methods often involves the design and optimization of parameters. While gradient descent methods have been mainly manifested to shine in the wave of deep learning, composable automatic differentiation can advance scientific computing where the application of classical adjoint methods alone is infeasible or cumbersome. Differentiable programming provides a novel paradigm that unifies data structures and control flows and facilitates gradient-based optimization of parameters in a computer program. This paper addresses the notion and implementation of the first solution algorithm for multi-scale flow physics across continuum and rarefied regimes based on differentiable programming. The fully differentiable simulator provides a unified framework for the convergence of computational fluid dynamics and machine learning, i.e., scientific machine learning. Specifically, parameterized flow models and numerical methods can be constructed for forward physical processes, while the parameters can be trained on the fly with the help of the gradients that are taken through the backward passes of the whole simulation program, a.k.a., end-to-end optimization. As a result, versatile data-augmented modeling and simulation can be achieved for physics discovery, surrogate modeling, and simulation acceleration. The fundamentals and implementation of the solution algorithm are demonstrated in detail. Numerical experiments, including forward and inverse problems for hydrodynamic and kinetic equations, are presented to demonstrate the performance of the numerical method. The open-source codes to reproduce the numerical results are available under the MIT license.

1. Introduction

Gaseous flows are endowed with a multi-scale structure. Sufficient separation of scales facilitates the development of theories of fluid dynamics at different scales. At the molecular mean free path, the Boltzmann equation can be employed to describe the flight and collision effects of individual particles. The Navier-Stokes equations depict the collective behavior of the many-particle system upon the fluid element models [1]. Intrigued by the well-known Hilbert's 6th problem [2], continuous efforts have been made to bridge the gaps between models from different scales, e.g., the Hilbert expansions from a theoretical point of view [3] and asymptotic-preserving numerical methods [4]. These approaches build a cross-scale path to represent the upscaling effects with reasonable

* Corresponding author.

E-mail address: txiao@imech.ac.cn

<https://doi.org/10.1016/j.jcp.2025.114224>

Received 27 February 2025; Received in revised form 25 June 2025; Accepted 3 July 2025

Available online 6 July 2025

0021-9991/© 2025 Elsevier Inc. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

Table 1
Nomenclature.

∂P	differentiable programming
AD	automatic differentiation
t	time variable
\mathbf{x}	space variables (x, y, z)
\mathbf{v}	particle velocity variables (u, v, w)
f	particle distribution function
Q, R	collision and relaxation operators in the kinetic equation
\mathcal{M}	Maxwellian distribution function
ν, τ	relaxation frequency and time ($\tau = 1/\nu$)
m	molecular mass
k	Boltzmann constant
ρ, \mathbf{V}, T, E	macroscopic primitive variables
ψ	collision invariants
\mathbf{W}	macroscopic conservative variables
$\mathbf{P}, \mathbf{T}, \mathbf{q}$	stress tensor, deviator tensor and heat flux
$\tilde{f}, \tilde{\mathbf{W}}$	cell-averaged distribution function and conservative variables
$\mathbf{F}^f, \mathbf{F}^w$	numerical fluxes for distribution function and conservative variables
\mathbf{u}	generic representation of flow variables
\mathcal{F}	operator of fluxes and sources in semi-discrete equations
\mathcal{C}	operator of initial and boundary conditions in semi-discrete equations
\mathbf{p}	control parameters
C	cost function
g	function integrated in the cost function
λ	adjoint variable
\mathbf{d}	data points in the training set
\mathcal{F}	generic representation of numerical operation
$\partial \mathcal{F}$	Jacobian and associated VJP and JVP of \mathcal{F}
\mathbf{s}	states in a sequence of operations
\mathbf{t}, \mathbf{r}	intermediate variables for derivation in forward- and reverse-mode ADs
NN_{θ}	neural network with trainable parameters θ
\mathcal{L}	function layer in a neural network
ω, \mathbf{b}, ϕ	weights, biases and activation function in a neural network
α	trainable parameters in the mechanical model
\mathbf{u}^{ref}	referenced flow solution
ϵ	regularization parameter
μ	dynamic viscosity coefficient
χ	proportion of upwind contribution to numerical fluxes
\hat{f}	reconstructed distribution function at cell face
σ	sigmoid function
h	reduced distribution function

asymptotics. However, it remains a formidable challenge to recover a continuous spectrum of flow physics and, in particular, to provide a succinct and accurate description in the transition regime.¹

Modeling and simulation of flows is a task of intertwined forward and inverse problems. Building reliable theoretical models and numerical methods requires a proper determination of design variables. At the molecular mean free path, i.e., the mesoscopic scale, phenomenological parameters in the collision kernel of the Boltzmann equation need to be routinely calibrated by experiments to preserve correct transport coefficients [5]. At the macroscopic level, constitutive functions are required as the closure of the Navier-Stokes equations and extended hydrodynamic models [6]. From a numerical perspective, a numerical scheme's success depends on the optimization of the parametric solution algorithm, e.g., the nonlinear weights in the reconstruction stencil, the ratio of central and upwind contributions in the numerical flux function, and the coefficients in the Butcher tableau of the Runge-Kutta integrator. Such optimization is more challenging for multi-scale flows since models and algorithms that are optimal at one scale are not necessarily suitable for another.

The burgeoning discipline of machine learning, particularly deep learning, has widened the possibilities for studying complex flows under extreme conditions that once seemed beset with difficulties. Neural networks as black-box models can be directly optimized using high-fidelity data without requiring interpretable expressions, thus enabling flexible modeling and simulation of flow physics, including efficient solution of high-dimensional differential equations [7,8], operator learning for mappings of functions and distributions [9,10], and data-driven discovery of non-equilibrium physics [11,12]. The architecture of neural networks allows for effortless scaling of parameters to deliver sufficient approximation capability. To improve the prediction of flow physics for such models, a typical workflow is to construct an objective function to be optimized with respect to the trainable parameters (known as the loss function in deep learning), following the supervised or unsupervised learning approach. The gradient information is usually obtained by backpropagating the loss through a chain of matrix operations, and the optimization problem is subsequently solved using stochastic gradient descent and its variants incorporating momentum and adaptive learning rates [13].

¹ <https://github.com/vavrines/KitAD.jl>

The optimization of parametric flow models and numerical methods can be performed based on two paradigms. The first idea is to build prior flow datasets through high-fidelity experiments or fine-scale simulations, followed by supervised learning. This approach is called offline training, and it can be applied to any parametric functions and operators on discrete spatio-temporal sensors. However, this type of training usually has no direct perception of the time-evolving processes and spatio-temporal coupling embedded in the fluid dynamic equations. As a result, the training data may not be utilized efficiently, which requires greater access to costly, high-confidence data [14]. On the other hand, the solution process of computational fluid dynamic (CFD) systems at the corresponding characteristic scales can be included in the loss function, forming the PDE-constrained optimization problem [15]. Such a methodology, called end-to-end training, is in favor as it covers the dynamics of a continuous-time model with necessary prior knowledge and physical structure. Each moment of the training data and predictions can be aligned and both supervised and unsupervised learning can be adapted.

Due to the high computational cost of CFD solvers, gradient-based optimization is arguably a natural pair with end-to-end training. However, the existence of solution trajectories of governing equations has made it trickier to obtain accurate gradient information. As analyzed in [16], direct evaluation of gradients in initial-value problems for partial differential equations can lead to an explosion of computational complexity, making it impractical to perform for a large number of flow variables or control parameters. The adjoint method addresses this challenge by introducing auxiliary variables and constructing the dual form of the optimization problem [17,18]. It is a well-suited tool for optimization and sensitivity analysis in high-dimensional space, e.g., aerodynamic shape design and optimization [19]. While the adjoint method encounters limitations in assimilating black-box modules, one might note that the counterpart of it within deep learning manifests as reverse-mode automatic differentiation (AD), commonly referred to as backpropagation, wherein the chain rule unfolds through a sequence of vector-Jacobian product operations from the loss function.

Differentiable programming (denoted as ∂P) has become a prominent notion for conducting scientific machine learning research. Unlike the convention where AD is limited to accumulating the gradients of matrix operations, neural networks in ∂P are regarded as generic nonlinear functions specified through a computer program. Other differentiable physical models or agents, including differential equations, can be incorporated as nodes in a computation graph equivalent to a neural network. This has allowed us to integrate principled differentiable operations and have them act as building blocks for each other to better approximate the structure of the problem in the task at hand, e.g., neural ODE models [20] and parametric high-dimensional differential equations [21]. The differentiability of the individual nodes in the computation graph facilitates taking gradients of the computer program under the chain rule, which is the essential difference between ∂P and classical computer programming [22]. In summary, ∂P provides a novel paradigm that unifies data structures and control flows to enable end-to-end AD and gradient-based optimization in machine learning and scientific computing tasks.

Although AD undoubtedly revolutionizes the paradigm of computing the gradient of complicated functions, which can be extremely tedious (or even impossible) to implement manually, it is important to note that it is not a panacea. The practice of AD faces at least two challenges. First, existing AD engines often restrict types and styles of code. In the case of JAX [23], for example, all functions need to be mathematically valid (a.k.a. pure functions), and thus control flows must be organized through functional programming. Such requirements are not trivial to meet, especially when external libraries from other compilers or languages are called, or heterogeneous computing is used. Second, AD may generate less efficient codes. An example goes to iterative schemes, e.g., the Krylov subspace methods [24]. Here, the direct application of AD to the solution of a linear system leads to a high computational overhead since the AD engine will treat the iteration as recurrence and store all intermediate steps. A feasible workaround for the above challenges is to implement the gradient manually, e.g., by applying the adjoint method to the final-state solution, and to compose the self-defined vector-Jacobian product into the chain rule for the surrounding operations. In essence, the full potential of ∂P can only be realized through the interplay of composable AD and adjoints (designing differentiable operations and combining them with efficient adjoints). In this context, ∂P may be properly conceptualized as a generalization of adjoint methods that plays nicely with machine-learning components.

There is an emerging consensus in the academic community on the importance of ∂P in CFD practices. Among others, Belbute-Peres et al. combined a differentiable CFD simulator and graph neural networks to accelerate the CFD prediction [25]. Zhuang et al. built differentiable codes to learn the optimal discretization for passive scalar advection in turbulent flows [26]. Bezgin et al. constructed a differentiable CFD program for multi-phase flows based on the JAX engine [27]. Fan and Wang employed ∂P to model fluid-structure interaction efficiently [28]. Ho and Farhat developed a differentiable embedded boundary method for aerodynamic optimization [29]. Kochkov et al. employed end-to-end differentiable learning for subgrid model discovery in turbulence [30]. Um et al. placed differentiable physics into the training process to reduce the error of iterative PDE solvers [31]. To the best of the author's knowledge, existing ∂P -related work has focused on solving single governing equations, while work on the physics of multi-scale flows with multiple governing equations and multiple degrees of freedom is limited.

Several neural network-based methods have also been developed for numerical simulation of rarefied and non-equilibrium flows. Lou et al. developed a physics-informed neural network (PINN) framework for the Boltzmann-BGK equation for forward and inverse flow problems [32]. Florio et al. applied PINNs to the solution of Poiseuille and thermal creep flows [33,34]. Tucny et al. investigated the microflows in cylinder arrays using PINNs [35], and identified the viscosity function in rarefied gases [36]. Zhang et al. developed PINNs based on the discrete velocity method of the linearized BGK equation [37]. Note that PINNs significantly differ from ∂P as a single neural network is treated as the PDE solver, whereas ∂P treats an existing solver as a differentiable computation graph, in which neural modules are inserted. Besides, neural networks have been used to reduce the complexity of solving the Boltzmann and related equations. Related work has been conducted in developing surrogate models for the Boltzmann equation [38–40] and constructing closure models for extended hydrodynamic equations [41–44]. Most of these approaches rely on offline training with prior datasets without tightly integrating numerical solvers and neural networks through ∂P .

This paper serves as the first comprehensive application of ∂P to solve continuum and rarefied flows. The unique distinction of this work compared to existing approaches lies in its seamless integration of automatic differentiation and adjoint sensitivity analysis. The parametric kinetic model and solution algorithm with differentiable operations are constructed upon the Boltzmann equation and its hydrodynamic asymptotics. The continuous adjoint equations are developed based on the semi-discrete governing equations derived from the finite volume method and then bundled into the AD engine. The design parameters in flow models and numerical methods, especially in neural networks, can then be end-to-end optimized on the fly with the gradients through the backward passes of the whole simulation program. Thus, a unified differentiable simulator is constructed that can tightly integrate the solution and optimization processes and is suitable for forward and inverse problems arising in rarefied and multi-scale gaseous flows. The program implementation is based on the Julia language, which enables language-wide AD via source-to-source transformation [45,46]. Numerical experiments for both continuum and rarefied flow problems will be presented to elucidate the ∂P -based solution paradigm and validate the computer program. For reproducible science, the relevant codes (augmented by Kinetic.jl, an indigenously developed differentiable framework designed for scientific and neural computing tasks [47]) for this paper are available under the MIT license.²

The rest of this paper is organized as follows. Section 2 presents a brief introduction to the kinetic theory of gases and numerical discretizations. Section 3 derives the adjoint equations and illustrates the joint use with AD in flow optimization problems. Section 4 describes the complete solution and optimization algorithms. Section 5 includes numerical experiments to demonstrate the validity and performance of the current method. The last section is the conclusion. The nomenclature of this paper is presented in Table 1.

2. Basic theory

The kinetic theory can inscribe the flow physics of rarefied and continuum gases. Lying at its core, the fluid is modeled as a many-particle system and its time-space evolution is statistically tracked using the single-particle distribution function. In the absence of internal degrees of freedom and external force, the Boltzmann equation for the distribution function $f(t, \mathbf{x}, \mathbf{v})$ writes

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = \mathcal{Q}(f, f) = \int_{\mathbb{R}^3} \int_{\mathbb{S}^2} [f(\mathbf{v}') f(\mathbf{v}_*) - f(\mathbf{v}) f(\mathbf{v}_*)] \mathcal{K}(\cos \theta, g) d\beta d\mathbf{v}_*, \quad (1)$$

where $\{\mathbf{v}, \mathbf{v}_*\}$ and $\{\mathbf{v}', \mathbf{v}'_*\}$ denote the pre- and post-collision velocities of two classes of colliding particles. The collision kernel $\mathcal{K}(\cos \theta, g)$ is a measure of the probability of collisions in different directions, where θ is the deflection angle and $g = |\mathbf{g}| = |\mathbf{v} - \mathbf{v}_*|$ is the magnitude of relative pre-collision velocity. The deflection angle satisfies the relation $\theta = \beta \cdot \mathbf{g}/g$, where the solid angle β is the unit vector along the relative post-collision velocity $\mathbf{v}' - \mathbf{v}'_*$.

The Boltzmann equation is an integro-differential equation with extremely high dimensionality and nonlinearity. To reduce the computational overhead of the fivefold integral, simplified relaxation models, e.g. the Bhatnagar-Gross-Krook (BGK) model, are commonly adopted in the simulation of complex flows. The relaxation model writes

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = \mathcal{R}(f) = \nu(\mathcal{E} - f), \quad (2)$$

where \mathcal{E} is the equilibrium distribution of relaxation directions and ν denotes the relaxation frequency. In the BGK model, \mathcal{E} takes the form of the Maxwellian, i.e.,

$$\mathcal{E} = \mathcal{M} := \rho \left(\frac{m}{2\pi kT} \right)^{3/2} \exp\left(-\frac{m}{2kT} (\mathbf{v} - \mathbf{V})^2\right), \quad (3)$$

where $\{\rho, \mathbf{V}, T\}$ are the macroscopic density, velocity and temperature, m is the molecular mass, k is the Boltzmann constant. The kinetic equations provide a mesoscopic view to describe particle transports and are consistent with first physical laws, including boundedness, conservation, invariance, and entropy principle [48,49]. In the following, we denote the Boltzmann collision operator $\mathcal{Q}(f, f)$ and relaxation term $\mathcal{R}(f)$ uniformly as $\mathcal{Q}(f)$.

A particle distribution function is related to a unique macroscopic state. The conservative variables in fluid mechanics can be obtained by taking moments of particle distribution function over velocity space, i.e.,

$$\mathbf{W}(t, \mathbf{x}) = \begin{pmatrix} \rho \\ \rho \mathbf{V} \\ \rho E \end{pmatrix} := \int_{\mathbb{R}^3} f \psi d\mathbf{v}, \quad (4)$$

where $\psi = (1, \mathbf{v}, \mathbf{v}^2/2)^T$ is a vector of collision invariants satisfying $\int_{\mathbb{R}^3} \mathcal{Q}(f) \psi d\mathbf{v} = 0$, and temperature is defined as

$$\frac{3}{2} kT = \frac{1}{2n} \int_{\mathbb{R}^3} (\mathbf{v} - \mathbf{V})^2 f d\mathbf{v}, \quad (5)$$

where n is the number density of gas. Taking conservative moments of the kinetic Eqs. (1) or (2) yields the conservation laws which write

$$\partial_t \mathbf{W} + \int_{\mathbb{R}^3} \psi \mathbf{v} \cdot \nabla_{\mathbf{x}} f d\mathbf{v} = 0, \quad (6)$$

² <https://github.com/vavrines/KitAD.jl>

i.e.,

$$\begin{aligned}\frac{\partial \rho}{\partial t} + \nabla_{\mathbf{x}} \cdot (\rho \mathbf{V}) &= 0, \\ \frac{\partial \rho \mathbf{V}}{\partial t} + \nabla_{\mathbf{x}} \cdot (\rho \mathbf{V} \otimes \mathbf{V}) + \nabla_{\mathbf{x}} \cdot \mathbf{P} &= 0, \\ \frac{\partial \rho E}{\partial t} + \nabla_{\mathbf{x}} \cdot (\rho E \mathbf{V}) + \nabla_{\mathbf{x}} \cdot (\mathbf{P} \cdot \mathbf{V}) + \nabla_{\mathbf{x}} \cdot \mathbf{q} &= 0,\end{aligned}\quad (7)$$

where \otimes denotes dyadic product, and the stress tensor \mathbf{P} and heat flux \mathbf{q} are defined as

$$\mathbf{P} = \int_{\mathbb{R}^3} (\mathbf{v} - \mathbf{V}) \otimes (\mathbf{v} - \mathbf{V}) f d\mathbf{v}, \quad \mathbf{q} = \int_{\mathbb{R}^3} \frac{1}{2} (\mathbf{v} - \mathbf{V})(\mathbf{v} - \mathbf{V})^2 f d\mathbf{v}. \quad (8)$$

Choosing a suitable closure strategy for \mathbf{P} and \mathbf{q} yields solvable Euler, Navier-Stokes, and extended hydrodynamic equations [50].

CFD is dedicated to approximating the solution of governing equations at a discrete level. For Eq. (2), we consider the domain $\Omega = \Omega_{\mathbf{x}} \times \Omega_{\mathbf{v}}$ with $N_{\mathbf{x}} \times N_{\mathbf{v}}$ non-overlapping cells,

$$\begin{aligned}\Omega_{\mathbf{x}} &= \bigcup_{i=1}^{N_{\mathbf{x}}} \Omega_i, \quad \bigcap_{i=1}^{N_{\mathbf{x}}} \Omega_i = \emptyset, \\ \Omega_{\mathbf{v}} &= \bigcup_{j=1}^{N_{\mathbf{v}}} \Omega_j, \quad \bigcap_{j=1}^{N_{\mathbf{v}}} \Omega_j = \emptyset,\end{aligned}\quad (9)$$

and the particle distribution function is approximated as

$$f \simeq \bigoplus_{i=1, j=1}^{N_{\mathbf{x}}, N_{\mathbf{v}}} f_{i,j}, \quad (10)$$

where $f_{i,j}$ denotes the piecewise-defined distribution function inside each cell. Different discretization methods can be used to approximate the solution $f_{i,j}$. Here we take the finite volume method as an example to illustrate.

We define the cell-averaged distribution function as

$$\bar{f}_{i,j} = \frac{1}{V_i V_j} \int_{\Omega_i} \int_{\Omega_j} f(t, \mathbf{x}, \mathbf{v}) d\mathbf{v} d\mathbf{x}, \quad (11)$$

where V_i and V_j are the volumes of Ω_i and Ω_j , respectively. Integrating Eq. (2) with respect to \mathbf{x} and applying Gauss's law yields

$$\begin{aligned}\frac{\partial \bar{f}_{i,j}}{\partial t} &= -\frac{1}{V_i} \oint_{\partial \Omega_i} \mathbf{F}_j^f(t, \mathbf{x}) \cdot d\mathbf{S} + \mathcal{Q}(\bar{f}_{i,j}) \\ &= -\frac{1}{V_i} \sum_{k=1}^{N_f} \mathbf{F}_{k,j}^f \cdot \Delta \mathbf{S}_k + \bar{v}_i (\bar{\mathcal{E}}_{i,j} - \bar{f}_{i,j}),\end{aligned}\quad (12)$$

where \mathbf{F}^f denotes the numerical flux of distribution function, $\mathbf{S} = \mathbf{n} \Delta \mathbf{S}$ is the area vector pointing out of the cell, and N_f is the number of faces. Different approaches can be employed to construct the numerical flux \mathbf{F}^f . Since the kinetic equation is consistent with particle transport processes, a neat choice is to build the numerical flux in an upwind manner. We take the k th face of cell i as an example and assume that the cell index on the other side of the face is $i+1$, then the numerical flux can be constructed as

$$\begin{aligned}\mathbf{F}_{k,j}^f &= \mathbf{v}_j f_{k,j}^f, \\ f_{k,j}^f &= \hat{f}_{i,j}^k H(\mathbf{n} \cdot \mathbf{v}_j) + \hat{f}_{i+1,j}^k (1 - H(\mathbf{n} \cdot \mathbf{v}_j)),\end{aligned}\quad (13)$$

where $\hat{f}_{i,j}^k$ denotes the reconstructed distribution function at the face based on in-cell slopes, and H is the Heaviside step function.

Following the derivation of Eq. (7), taking moments of Eq. (12) over velocity space $\Omega_{\mathbf{v}}$ yields the semi-discrete formulation of conservation laws, i.e.,

$$\frac{\partial \bar{\mathbf{W}}_i}{\partial t} = -\frac{1}{V_i} \oint_{\partial \Omega_i} \mathbf{F}^W \cdot d\mathbf{S} = -\frac{1}{V_i} \sum_{k=1}^{N_f} \mathbf{F}_k^W \cdot \Delta \mathbf{S}_k. \quad (14)$$

Here, the cell-averaged conservative variables in Ω_i can be approximated by numerical quadrature at the discrete level, i.e.,

$$\bar{\mathbf{W}}_i := \int_{\mathbb{R}^3} f_i \psi d\mathbf{v} \simeq \sum_{j=1}^{N_{\mathbf{v}}} w_j f_{i,j} \psi_j, \quad (15)$$

where w_j denotes the quadrature weights.

Given the number of elements $N_{\mathbf{x}}$ and $N_{\mathbf{v}}$, Eqs. (12) and (14) form a system of ordinary differential equations (ODEs) or differential-algebraic equations (DAEs), respectively. We uniformly denote the variables as $\mathbf{u} \in \mathbb{R}^{N_u}$, and the solution system can then be written as

$$\begin{aligned}\frac{\partial \mathbf{u}}{\partial t} &= \mathcal{F}(t, \mathbf{u}, \mathbf{p}), \\ \mathcal{C}(t, \mathbf{u}, \mathbf{p}) &= 0,\end{aligned}\quad (16)$$

where $\mathbf{p} \in \mathbb{R}^{N_p}$ is the collection of control parameters of the solution algorithm. The contributions of numerical fluxes and source terms are represented by the operator \mathcal{F} , and the initial and boundary conditions are bounded by the operator \mathcal{C} .

The solution of Eq. (16) can be obtained by integrating it along the time direction. Note that the relaxation frequency in Eq. (2) is proportional to the gas density, and thus the choice of the integrator is related to the regime of the flow problem. In the continuum limit, Eq. (2) can become stiff. Therefore, an appropriate integrator is chosen in the hope that it is efficient and A- or L-stable for stiff and oscillatory problems. The choices available include the backward differentiation formula (BDF) [51], multi-stage implicit Runge–Kutta (IRK) methods [52], and implicit-explicit (IMEX) methods [53]. The performance of different integrators for solving kinetic equations is briefly summarized in [54].

3. Differentiation strategy

3.1. Adjoint system

For the differential-equation-constrained optimization problem, a cost function denoted $C(\mathbf{u}, \mathbf{p})$ will be computed throughout the solution trajectory of the governing equation. This problem can often be handled efficiently by the adjoint sensitivity method [55], which is well-suited for situations requiring the sensitivity analysis of a scalar (or low-dimensional) function of the solution with respect to a potentially large number of parameters. We follow the derivation presented in [56], but modify it to specialize on the adjoint system of Eq. (16). Eq. (16) is index-0 and index-1 differential-algebraic equations (DAEs) for hydrodynamic and kinetic equations. Since it is linear with respect to the derivative term, we introduce the linear mass matrix and reformulate it as

$$M\mathbf{u}' = \mathcal{G}(t, \mathbf{u}, \mathbf{p}), \quad (17)$$

where \mathbf{u}' denotes the time derivative for brevity.

For a time-varying problem, a viable cost function can be constructed as

$$C(\mathbf{u}, \mathbf{p}) = \int_{t_0}^{t_1} g(t, \mathbf{u}, \mathbf{p}) dt, \quad (18)$$

where t_0 and t_1 denote two moments in time. We expect to obtain the derivative $\partial C / \partial \mathbf{p}$, and the problem translates into computing the intermediate quantity λ (called the adjoint variable) as the solution of the adjoint system. The derivatives $\partial_{\mathbf{u}} C$ and $\partial_{\mathbf{p}} C$ should exist and be bounded. We introduce the adjoint variable λ as a Lagrange multiplier that conforms

$$I(\mathbf{u}, \mathbf{p}) = C(\mathbf{u}, \mathbf{p}) - \int_{t_0}^{t_1} \lambda^* \mathcal{H}(t, \mathbf{u}, \mathbf{u}', \mathbf{p}) dt, \quad (19)$$

where λ^* denotes the conjugate transpose of λ , and $\mathcal{H} = M\mathbf{u}' - \mathcal{G} = 0$. The partial derivatives of C with respect to \mathbf{p} can thus be written as

$$\frac{\partial C}{\partial \mathbf{p}} = \frac{\partial I}{\partial \mathbf{p}} = \int_{t_0}^{t_1} (g_{\mathbf{p}} + g_{\mathbf{u}} \mathbf{u}_{\mathbf{p}}) dt - \int_{t_0}^{t_1} \lambda^* (\mathcal{H}_{\mathbf{p}} + \mathcal{H}_{\mathbf{u}} \mathbf{u}_{\mathbf{p}} + \mathcal{H}_{\mathbf{u}'} \mathbf{u}'_{\mathbf{p}}) dt. \quad (20)$$

Applying integration by parts leads to

$$\frac{\partial C}{\partial \mathbf{p}} = \int_{t_0}^{t_1} (g_{\mathbf{p}} - \lambda^* \mathcal{H}_{\mathbf{p}}) dt + \int_{t_0}^{t_1} (g_{\mathbf{u}} - \lambda^* \mathcal{H}_{\mathbf{u}} + (\lambda^* \mathcal{H}_{\mathbf{u}'})) \mathbf{u}_{\mathbf{p}} dt - [\lambda^* \mathcal{H}_{\mathbf{u}'} \mathbf{u}_{\mathbf{p}}]_{t_0}^{t_1}. \quad (21)$$

We require that

$$g_{\mathbf{u}} - \lambda^* \mathcal{H}_{\mathbf{u}} + (\lambda^* \mathcal{H}_{\mathbf{u}'})' = 0, \quad (22)$$

and

$$\lambda^* \mathcal{H}_{\mathbf{u}'}|_{t=t_1} = 0, \quad (23)$$

and thus the sensitivity equation for $\partial C / \partial \mathbf{p}$ becomes

$$\begin{aligned} \frac{\partial C}{\partial \mathbf{p}} &= \int_{t_0}^{t_1} (g_{\mathbf{p}} - \lambda^* \mathcal{H}_{\mathbf{p}}) dt + (\lambda^* \mathcal{H}_{\mathbf{u}'} \mathbf{u}_{\mathbf{p}})|_{t=t_0} \\ &= \int_{t_0}^{t_1} (g_{\mathbf{p}} + \lambda^* \mathcal{G}_{\mathbf{p}}) dt + \lambda^*(t_0) M \mathbf{u}_{\mathbf{p}}. \end{aligned} \quad (24)$$

Thus we have derived the sensitivity equation along with the adjoint DAE system for λ and its boundary condition. The derivative of the solution with respect to a cost function can be obtained by solving the adjoint and sensitivity equations in turn. Note that even if C is discrete, it can be similarly expressed as

$$C(\mathbf{u}, \mathbf{p}) = \int_{t_0}^{t_1} \sum_i^{N_d} \|\mathbf{d}_i - \mathbf{u}(t_i, \cdot)\|^2 \delta(t_i - t) dt, \quad (25)$$

in which case

$$g_{\mathbf{u}}(t_i) = 2(\mathbf{d}_i - \mathbf{u}(t_i, \cdot)), \quad (26)$$

where \mathbf{d}_i denotes the data point at t_i [21].

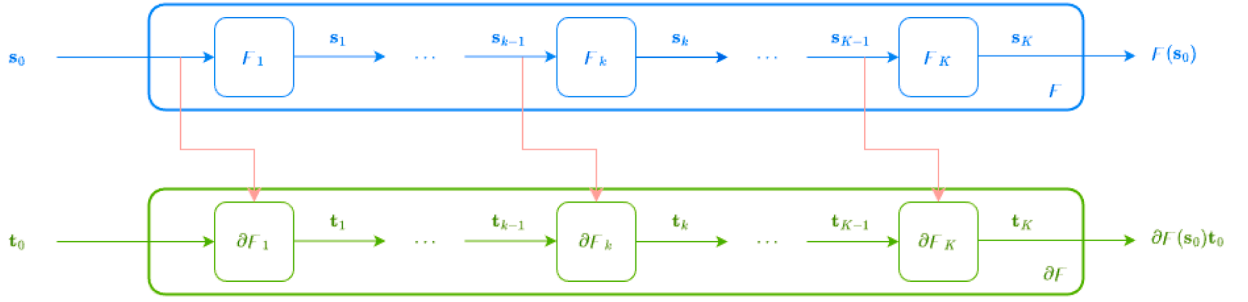


Fig. 1. Schematic of forward-mode automatic differentiation for a sequence of functions.

3.2. Automatic differentiation

In ∂P , the solution of the adjoint system is nested within the AD workflow. We consider a computer program in which a numerical operation $F : S_0 \rightarrow S_K$ can generally be written as a sequence of compositions, i.e.,

$$F = F_K \circ F_{K-1} \circ \cdots \circ F_1, \quad (27)$$

where $F_k : S_{k-1} \rightarrow S_k$. The inputs and outputs of functions are $s_{k-1} \in S_{k-1}$ and $s_k \in S_k$, respectively. Note that multiple dependencies of intermediate functions can be efficiently represented using directed acyclic graphs (DAGs), thus keeping the consistency of the above equation. Based on the chain defined in Eq. (27), the full Jacobian matrix can be obtained as

$$\partial F(s_0) = \partial F_K(s_{K-1}) \partial F_{K-1}(s_{K-2}) \cdots \partial F_1(s_0). \quad (28)$$

The computational overhead of the above equation is high due to the matrix multiplications of intermediate Jacobians. However, in most cases, we need the derivatives of the composition of F and a scalar-valued cost function $C \circ F$. This translates into solving the right or left multiplication of the Jacobian, rather than itself. Forward-mode and reverse-mode ADs are developed on this basis, respectively.

Forward-mode AD

The computation of Jacobian can be understood as a composition of primitively known linear maps. The evaluation of $\partial F(s_0)$ on an input vector \mathbf{w} can be performed by computing Jacobian-vector products (JVPs) along the same direction as the computation of intermediate states s_k , hence the name forward-mode AD. This corresponds to the right multiplication of Eq. (28). Such a scheme can often be succinctly implemented using dual numbers [57]. Since the computational complexity and memory load of computing ∂F_k is comparable to the cost of computing F_k , the computational cost of a JVP is roughly twice that of F_k . The schematic of forward-mode AD is presented in Fig. 1, and the detailed solution steps can be found in Algorithm 1.

Algorithm 1 Forward-mode automatic differentiation for a sequence of functions.

Function: $F = F_K \circ F_{K-1} \circ \cdots \circ F_1$

Input variable: $s_0 \in S_0$

Input direction: $\mathbf{w} \in S_0$

Initialize $\mathbf{t}_0 = \mathbf{w}$

for $k = 1, \dots, K$ **do**

 Compute $s_k = F_k(s_{k-1})$

 Compute $\mathbf{t}_k = \partial F_k(s_{k-1})\mathbf{t}_{k-1}$

end for

Output function value: $F(s_0) = s_K$

Output JVP: $\partial F(s_0)\mathbf{w} = \mathbf{t}_K$

Reverse-mode AD The gradient of $C \circ F$ takes the form

$$\nabla(C \circ F)(s_0) = \partial F(s_0)^* \nabla C(F(s_0)), \quad (29)$$

where the adjoint map is defined as $\partial F(s_0)^* : S_K \rightarrow S_0$, and it yields

$$\partial F(s_0)^* = \partial F_1(s_0)^* \circ \partial F_2(s_1)^* \circ \cdots \circ \partial F_K(s_{K-1})^*. \quad (30)$$

Each intermediate adjoint $\partial F_k(s_{k-1})^*$ is equivalent to a vector-Jacobian product (VJP). Here, VJPs are computed recursively along the opposite direction of s_k , hence the name reverse-mode AD. The computational complexity of VJPs, is roughly twice that of the

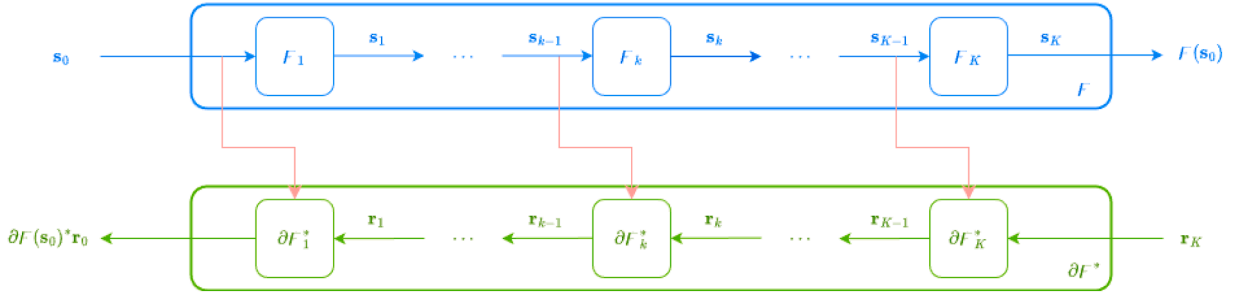


Fig. 2. Schematic of reverse-mode automatic differentiation for a sequence of functions.

Algorithm 2 Reverse-mode automatic differentiation for a sequence of functions.

Function: $F = F_K \circ F_{K-1} \circ \dots \circ F_1$

Input variable: $s_0 \in S_0$

Output direction: $w \in S_K$

for $k = 1, \dots, K$ **do**

 Compute $s_k = F_k(s_{k-1})$

end for

Initialize $r_K = w$

for $k = 1, \dots, K$ **do**

 Compute $r_{k-1} = \partial F_k(s_{k-1})^* r_k$

end for

Output function value: $F(s_0) = s_K$

Output VJP: $\partial F(s_0)^* w = r_0$

▷ Forward pass

▷ Backward pass

original function. The memory usage grows linearly with respect to the sequence length K . The schematic of reverse-mode AD is presented in Fig. 2, and the detailed solution steps can be found in Algorithm 2.

As discussed in [22], the computational efficiency of forward- and reverse-mode ADs depends on the dimension of S_k . Given $S_k \subseteq \mathbb{R}^{D_k}$, the forward-mode AD is more advantageous in the case of $D_K \geq D_0$, while the reverse-mode is more favorable when $D_K < D_0$. Note that the latter is the more common case when a considerable number of parameters are involved, as is the case in neural networks.

Based on the predefined chain rules [58], the adjoint system in Section 3.1 can be automatically invoked when the solution of differential equations is encountered during the differentiation process. Therefore, the adjoint and AD systems can be bundled and work together as a whole. The current study employs Enzyme, an AD engine that performs code generation at the intermediate representation (IR) level of the LLVM. As introduced above, Enzyme generally incurs a memory overhead of two to several times that of the original code, owing to the need to store adjoints and cached intermediate values. Enzyme demonstrates runtime superiority since it differentiates after LLVM's optimization passes and thus can produce gradient code that often approaches or exceeds the speed of manually written adjoints. It maintains robust compatibility and can interoperate with any code that gets lowered to the LLVM IR. For a more detailed description, we refer the interested reader to the original paper [46].

4. Solution algorithm

4.1. Machine learning

Machine learning models are parametric representations that map inputs (features) and outputs (targets) without being explicitly programmed. Among these, neural networks have emerged as the preeminent architecture within the contemporary landscape of deep learning advancements. Taking the feedforward neural network NN_θ , as an example, it can be viewed as a sequence of parameterized functions, i.e.,

$$\begin{aligned}
 s_0 &:= \mathbf{u}, \\
 s_1 &:= \mathcal{L}_1(s_0, \theta_1), \\
 s_2 &:= \mathcal{L}_2(s_1, \theta_2), \\
 &\vdots \\
 s_K &:= \mathcal{L}_K(s_{K-1}, \theta_K),
 \end{aligned} \tag{31}$$

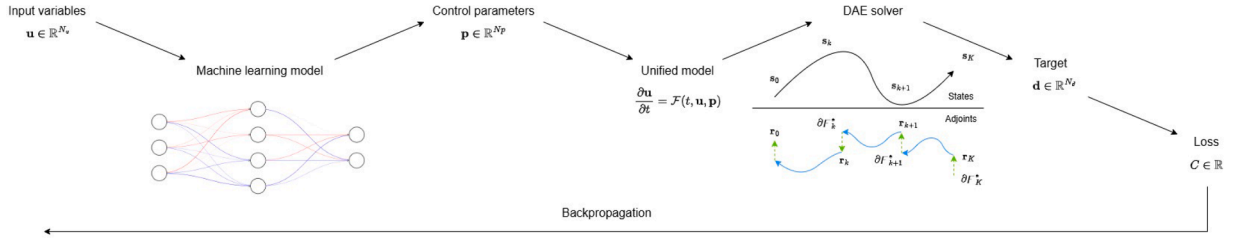


Fig. 3. Flow of the solution and training algorithms for the unified neural-network-augmented model based on differentiable programming.

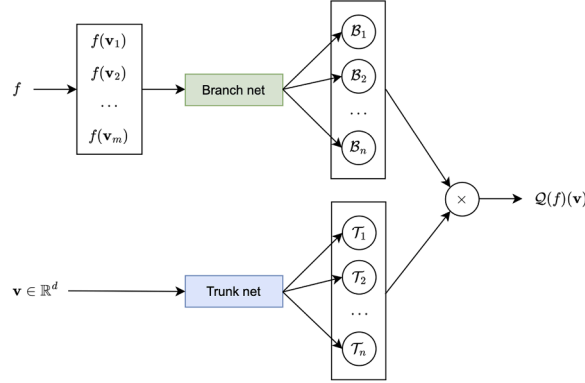


Fig. 4. Architecture of DeepONet surrogate model of the Boltzmann equation.

where \mathcal{L}_k indicates a function layer, and $\theta := (\theta_1, \dots, \theta_K)$ denotes the parameters to be optimized. As a typical parameterization, the multi-layer perceptron (MLP) adopts the fully-connected layers of the form

$$\mathcal{L}_k := \phi_k(\omega_k s_{k-1} + \mathbf{b}_k) \quad (32)$$

where the affine layer with the weight matrix ω_k and bias vector \mathbf{b}_k and the activation function ϕ_k are combined to describe a nonlinear transformation. The affine layer in Eq. (32) can be replaced by other linear functions, e.g., convolution and filtering, while Eq. (31) can be replaced with more general models with more complex structures, e.g., that used in residual and recurrent learning.

Machine learning provides versatile means for describing non-equilibrium flows. Trained on high-fidelity data, parameterized models can effectively promote the applicability of constitutive relations, numerical fluxes, source terms, and related components. Taking neural networks as an example, based on Eq. (16), a unified neural-network-augmented model can be formulated as

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= F(t, \mathbf{u}, \boldsymbol{\alpha}, \text{NN}_{\theta}(t, \mathbf{u})), \\ C(t, \mathbf{u}, \boldsymbol{\alpha}, \theta) &= 0, \end{aligned} \quad (33)$$

where NN_{θ} denotes the forward pass of a neural network, $\mathbf{p} = (\boldsymbol{\alpha}, \theta)$ signifies the parameters in the mechanical and neural network models, respectively. The architecture of Eq. (33) is similar to that of neural ordinary differential equations, and thus offers advantages, e.g., memory efficiency and adaptive computation [20]. As a trainable system, it has the same solution methodology as Eq. (16). With the introduction of neural networks, the dimensionality of the model's parameter space increases dramatically, and the ability to depict non-equilibrium flows can be subsequently improved. Note that neural networks typically perform inference significantly faster than theoretical models of equivalent accuracy, while the training processes can be computationally intensive. A neural network-based model can be less generalizable compared to interpretable theoretical models, while high-quality data, careful hyperparameter tuning, and incorporation of physical structure can be effective in improving its universality.

4.2. Solution algorithm

The unified model developed in Eq. (33) requires the solution of both forward and optimization problems. The solution of the forward problem follows a similar principle as Eq. (16) in Section 2, where an appropriate integrator implemented with differentiable operations is employed to iterate numerical solutions. For the constrained optimization problem, a cost function is needed to align the numerical solution towards the referenced data points. A commonly adopted definition of the cost function for supervised learning tasks writes

$$C = \sum_i^{N_c} \sum_j^{N_t} \|\mathbf{u}_i^{\text{ref}}(t_j) - \mathbf{u}_i(t_j)\|^2 + \epsilon \|\theta\|^2, \quad (34)$$

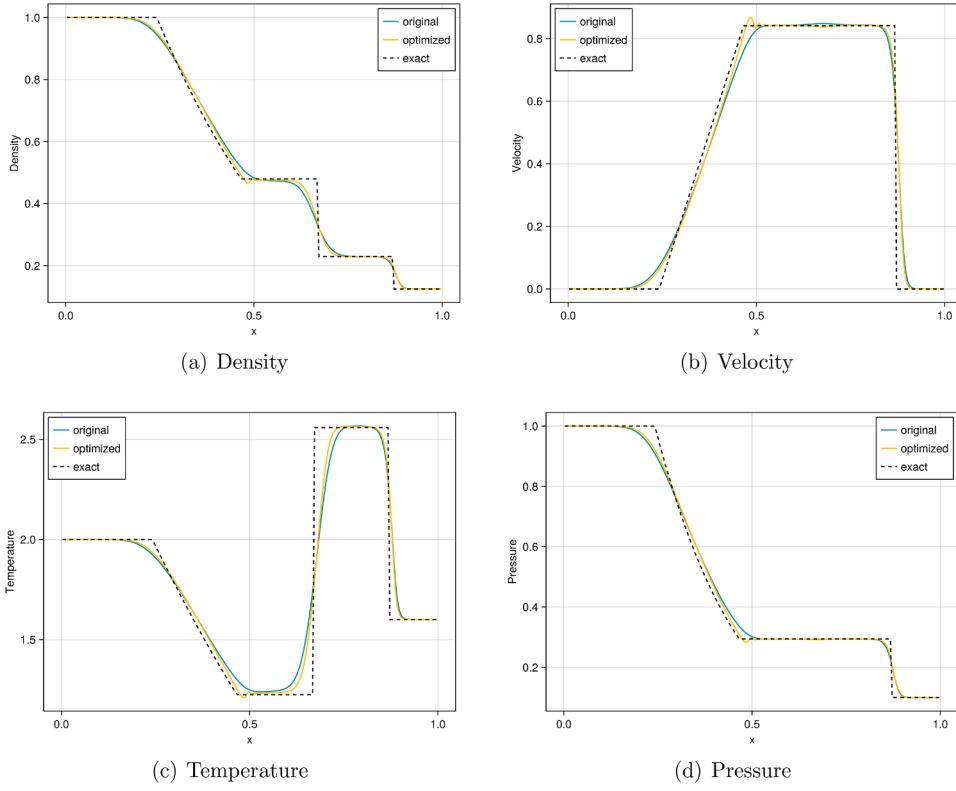


Fig. 5. Profiles of density, U -velocity, temperature, and pressure at $t = 0.2$ simulated by the single-parameter model in the Sod shock tube problem.

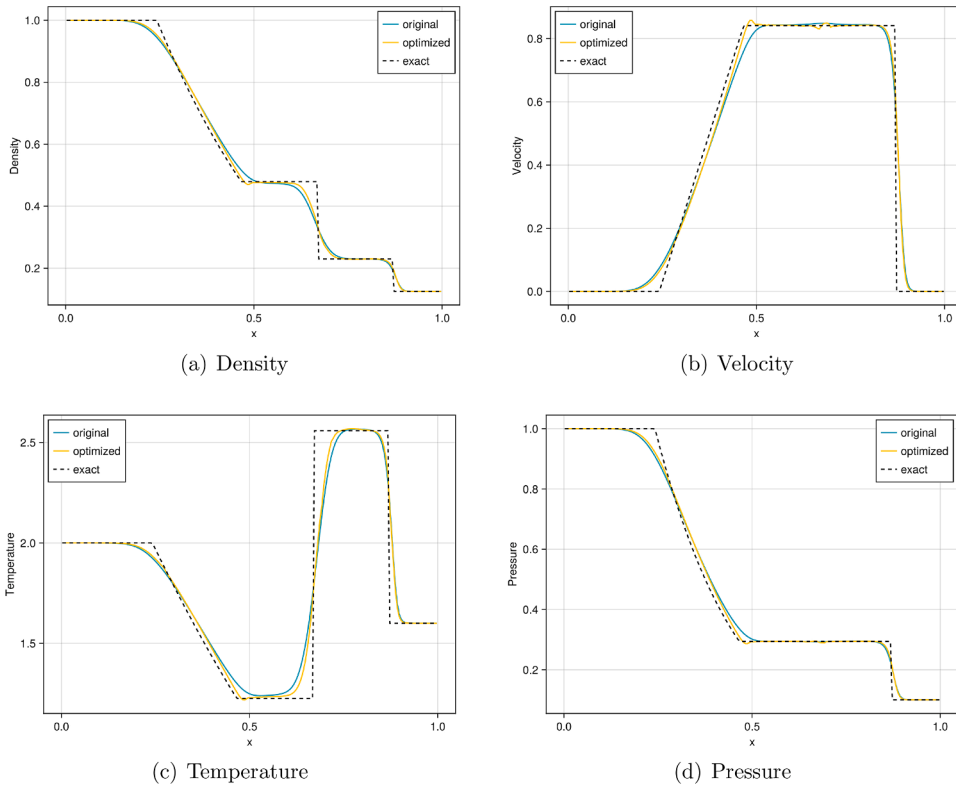


Fig. 6. Profiles of density, U -velocity, temperature, and pressure at $t = 0.2$ simulated by the multi-parameter model in the Sod shock tube problem.

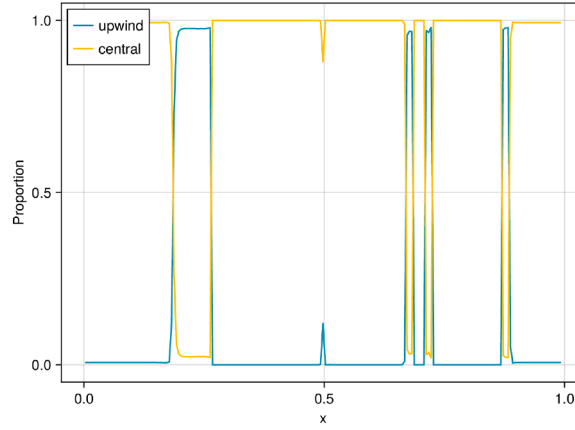


Fig. 7. Proportions of contributions of the multi-parameter model in the Sod shock tube problem.

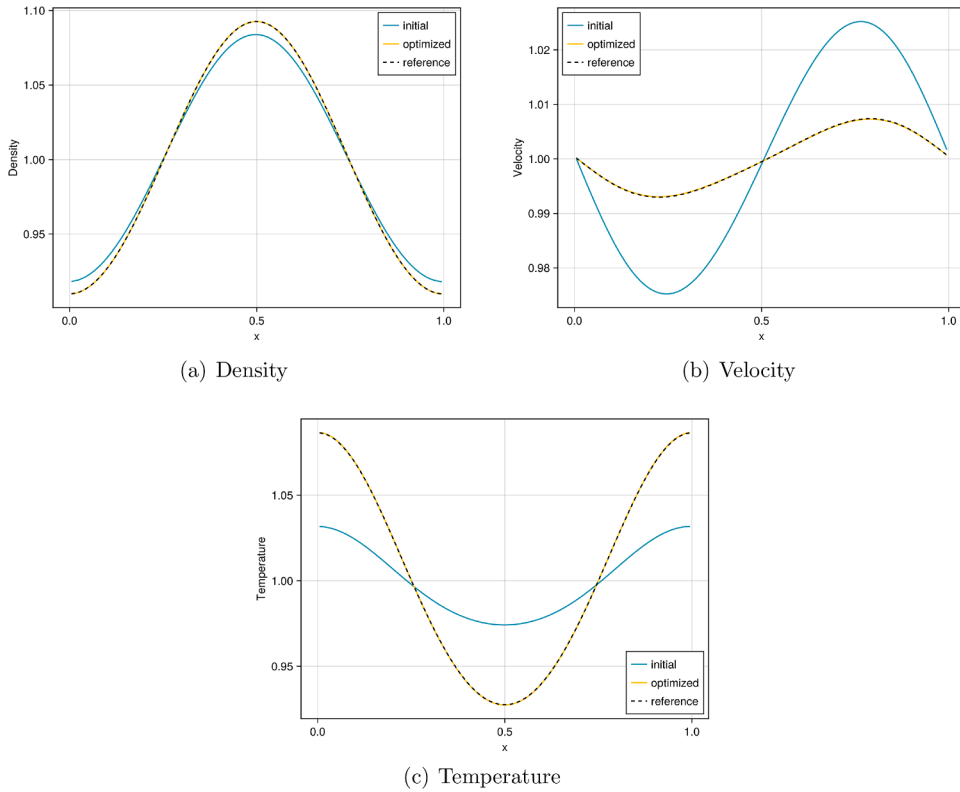


Fig. 8. Profiles of density, U -velocity, and temperature at $t = 0.25$ simulated by the initial and optimized models in the wave propagation problem.

which corresponds to the discrete cost function defined in Eq. (25). Here, N_c represents the number of different flow conditions to be simulated and N_t is the number of time steps. Thus, the trajectories of the numerical solution are tracked upon the total number of data samples $N_d = N_c N_t$. The referenced solution \mathbf{u}^{ref} can be obtained from fine-grained models with high confidence and fidelity, e.g., molecular simulation results. The L_2 regularization term mitigates overfitting and improves model generalization by penalizing large weights. The regularization parameter ϵ is an empirical parameter that needs to be chosen in a trade-off between bias and variance.

The gradient information of the cost function C is required to leverage gradient-based optimization methods. Since a considerable number of parameters is introduced with neural networks, usually the reverse-mode AD is favored. As discussed in Section 3.2, the solution algorithm can be expressed as a sequence of operations, and its derivatives can be computed with the help of sequenced

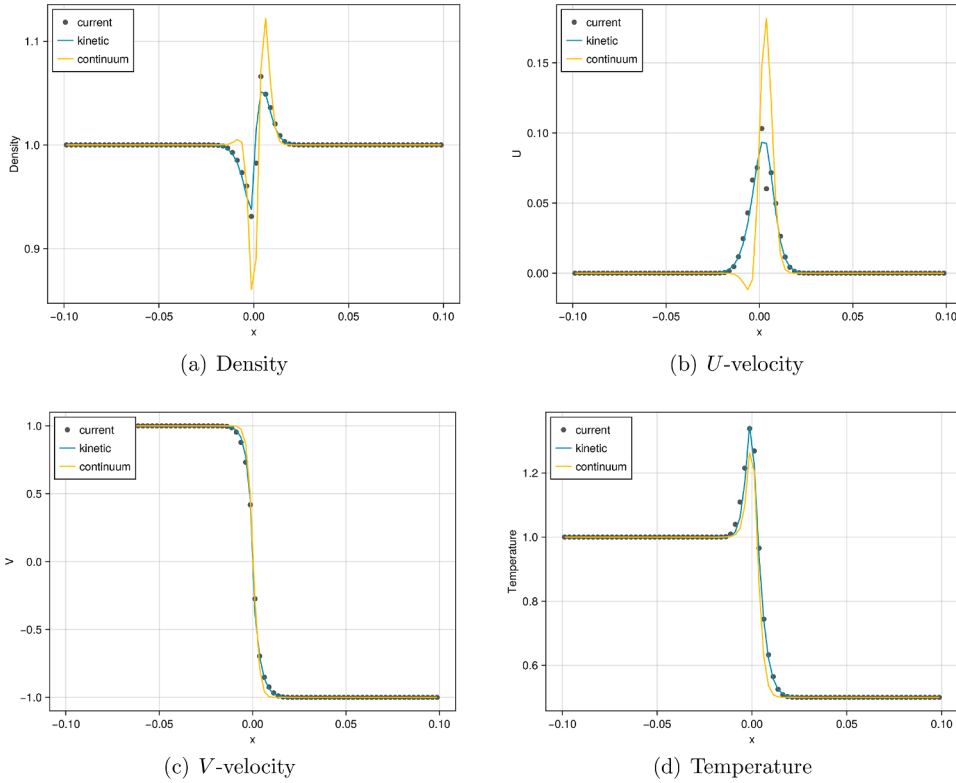


Fig. 9. Profiles of density, U -velocity, V -velocity, and temperature at $t = \tau_0$ simulated by different models in the shear layer problem (in the training set).

VJPs, which allows for a recursive decomposition of a primitively known set of pullbacks. The predefined adjoints can be incorporated into the chain rule to accelerate the gradient computation. After the derivatives of the cost function have been obtained, gradient-descent-type methods can be employed to optimize the cost function efficiently, e.g. the first-order stochastic gradient descent (SGD) method [59], and the second-order Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [60]. The optimized mechanical and neural parameters are then used to perform the subsequent forward computation for the unified model, and so on iteratively. The flow of the ∂P -based solution algorithm is briefly summarized in Fig. 3. Note that ∂P incurs additional runtime and memory overhead from backpropagation of neural networks, but can benefit from hardware acceleration and just-in-time compilation. As the adjoint system is bundled into the AD engine, in the absence of neural networks, the computational efficiency of current ∂P is comparable to that of classical adjoint methods.

5. Numerical experiments

In this section, we will conduct numerical experiments to validate the ∂P -based solution algorithm. To illustrate the applicability of the methodology to cross-scale flows, cases with different degrees of gas rarefaction are considered. Dimensionless variables are uniformly adopted in the numerical simulations, which are defined as

$$\begin{aligned} \tilde{t} &= \frac{t}{L_0/V_0}, \quad \tilde{\mathbf{x}} = \frac{\mathbf{x}}{L_0}, \quad \tilde{\rho} = \frac{\rho}{\rho_0}, \quad \tilde{\mathbf{V}} = \frac{\mathbf{V}}{V_0}, \quad \tilde{T} = \frac{T}{T_0}, \quad \tilde{E} = \frac{E}{V_0^2}, \\ \tilde{\mathbf{P}} &= \frac{\mathbf{P}}{\rho_0 V_0^2}, \quad \tilde{\mathbf{q}} = \frac{\mathbf{q}}{\rho_0 V_0^3}, \quad \tilde{\mu} = \frac{\mu}{\rho_0 L_0 V_0}, \quad \tilde{\mathbf{v}} = \frac{\mathbf{v}}{V_0}, \quad \tilde{f} = \frac{f}{\rho_0/V_0^3}, \end{aligned} \quad (35)$$

where μ denotes the dynamic viscosity coefficient. Physical quantities with a subscript 0 indicate their value in the reference state, where $V_0 = \sqrt{2kT_0/m}$ is the most probable molecular speed. The global Knudsen number is defined as

$$\text{Kn} = \frac{\ell_0}{L_0} = \frac{V_0}{L_0 \nu_0}, \quad (36)$$

where $\ell_0 = V_0/\nu_0$ is the referenced molecular mean free path and ν_0 is the mean collision frequency. For brevity, we drop the tilde notation to denote dimensionless variables henceforth.

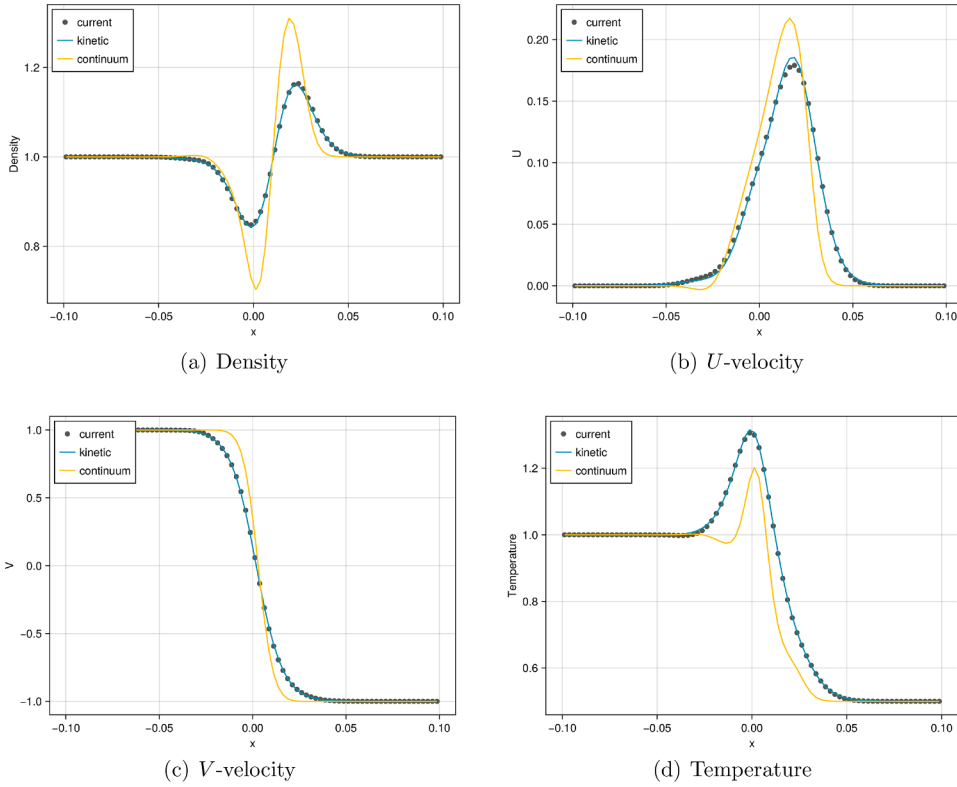


Fig. 10. Profiles of density, U -velocity, V -velocity, and temperature at $t = 5\tau_0$ simulated by different models in the shear layer problem (in the training set).

5.1. Optimization of numerical flux

Developing low-dissipation, strongly robust numerical fluxes is an important element of modern CFD. The success of many numerical methods can be attributed to the combination of central and upwind discretizations, e.g., hybrid central-upwind schemes [61,62] and gas-kinetic schemes [63,64]. The proportion of upwind and central contributions is usually adjustable to accommodate the mechanisms of convection and diffusion in different flows. In addition to relying on a priori assumptions, the proportion can be determined by solving the optimization problem using ∂P .

Based on the definition in Eq. (13), here we explicitly write the distribution function at the k th face as

$$f_k^f = \chi f_k^u + (1 - \chi) f_k^c, \quad (37)$$

where f_k^u and f_k^c denote the particle distribution functions constructed by the upwind and central approaches, respectively. The two distribution functions are unified by the coefficient $\chi \in [0, 1]$. We assume that the cell index to which the normal vector of face points is $i + 1$ and the other cell corresponding to it is i , and the upwind distribution function can be constructed according to Eq. (13), i.e.,

$$f_k^u = \hat{f}_i^k H(\mathbf{n} \cdot \mathbf{v}) + \hat{f}_{i+1}^k (1 - H(\mathbf{n} \cdot \mathbf{v})), \quad (38)$$

where $\{\hat{f}_i^k, \hat{f}_{i+1}^k\}$ are the reconstructed distribution functions on both sides of the face, and H denotes the Heaviside step function. The central contribution can be modeled as a Maxwellian distribution,

$$f_k^c = \mathcal{M}_k^c, \quad (39)$$

which can be determined with the help of the compatibility condition, i.e.,

$$\int_{\mathbb{R}^3} \mathcal{M}_k^c \psi d\mathbf{v} = \int_{\mathbf{n} \cdot \mathbf{v} \geq 0} \hat{f}_i^k \psi d\mathbf{v} + \int_{\mathbf{n} \cdot \mathbf{v} < 0} \hat{f}_{i+1}^k \psi d\mathbf{v}. \quad (40)$$

Note that incorporating a specific form of the distribution function in Eq. (38) leads to different gas dynamics. Different truncation orders of the Chapman-Enskog expansion can yield the Euler, Navier-Stokes, and extended hydrodynamic solutions.

Here, we consider the construction of numerical fluxes for the Euler equations. The particle distribution functions on both sides of the face adopt the Maxwellian determined by the reconstructed conservative variables,

$$\hat{f}_i^k = \hat{\mathcal{M}}_i^k, \quad \hat{f}_{i+1}^k = \hat{\mathcal{M}}_{i+1}^k. \quad (41)$$

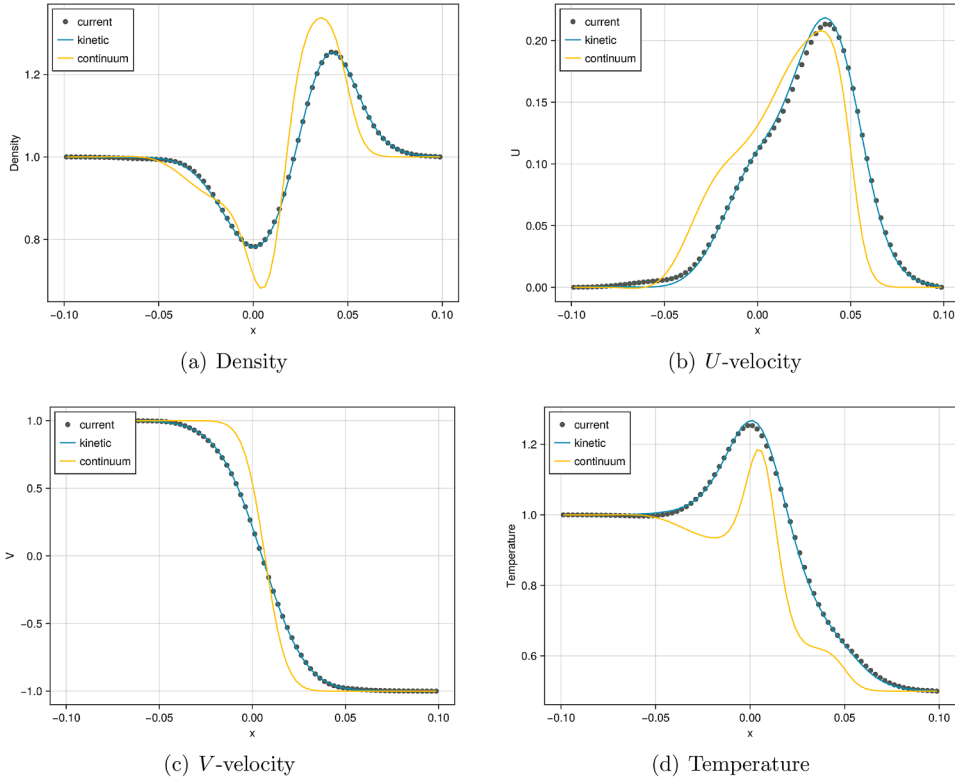


Fig. 11. Profiles of density, U -velocity, V -velocity, and temperature at $t = 10\tau_0$ simulated by different models in the shear layer problem (in the training set).

Table 2
Computational setup of the Sod shock tube problem.

Equation	Gas	t	x	N_x	Order	Flux
Euler	Argon	$(0, 0.2]$	$[0, 1]$	200	1	Central-Upwind
Integrator	Boundary	CFL	N_p	Optimizer		
Euler	Dirichlet	0.5	$\{1, 199\}$	AdamW		

The macroscopic fluxes for conservative variables are thus given by

$$\mathbf{F}_k^W = \int_{\mathbb{R}^3} \mathbf{v} f_k^f \psi d\mathbf{v}. \quad (42)$$

Since the distribution function f_k^f consists of three Maxwellian distributions, the above integral can be analytically solved. The Sod shock tube problem is employed as the numerical experiment. The initial particle distribution function is set as Maxwellian in correspondence with the following macroscopic variables,

$$\begin{pmatrix} \rho \\ U \\ p \end{pmatrix}_{t=0, x<0.5} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} \rho \\ U \\ p \end{pmatrix}_{t=0, x \geq 0.5} = \begin{pmatrix} 0.125 \\ 0 \\ 0.1 \end{pmatrix}. \quad (43)$$

The system is non-dimensionalized by the tube length together with the initial physical quantities on the left side. The detailed computational setup is presented in Table 2, where N_p indicates the number of trainable parameters.

The flux function is optimized using two approaches. The first approach creates and optimizes a single parameter that controls the global behavior of the flux function in Eq. (42). The second strategy constructs a parameter at each face (199 independent parameters in total) that provides fine-grained control of local evolution. To bound the predicted proportions of central and upwind contributions, the sigmoid function is employed to normalize the trainable parameters, i.e.,

$$\chi = \sigma(\mathbf{p}). \quad (44)$$

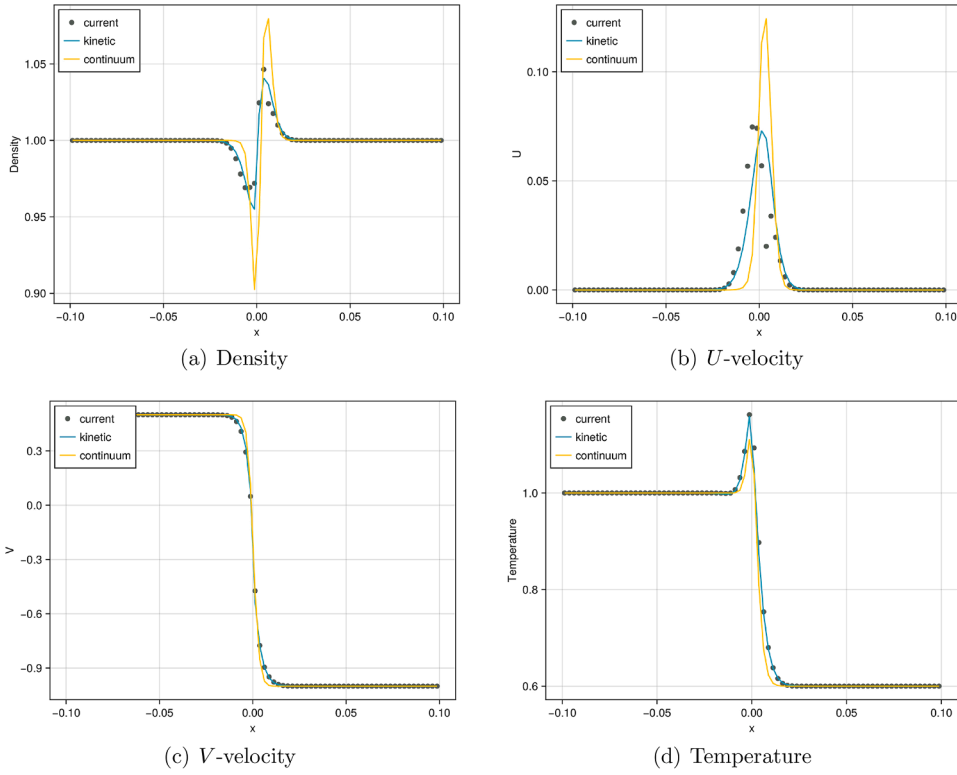


Fig. 12. Profiles of density, U -velocity, V -velocity, and temperature at $t = \tau_0$ simulated by different models in the shear layer problem (beyond the training set using different initial condition).

Table 3

Proportions of central and upwind fluxes of the single-parameter model in the Sod shock tube problem.

	Central	Upwind
before optimization	0.67%	99.33%
after optimization	97.54%	2.46%

The initial value of \mathbf{p} is set as 5.0. The cost function is defined as

$$C = \sum_i^{N_x} \|\mathbf{W}_i^{\text{ref}}(t = 0.2) - \mathbf{W}_i(t = 0.2)\|^2, \quad (45)$$

where the reference solution \mathbf{W}^{ref} is obtained from the theoretical solution of the one-dimensional Riemann problem.

Fig. 5 presents the profiles of density, velocity, temperature, and pressure in the shock tube at $t = 0.2$ simulated by the single-parameter model. Table 3 provide the contribution proportions of central and upwind fluxes before and after optimization. It can be seen that as the dominant mechanism shifts from the upwind to the central scheme, the numerical dissipation in the numerical method is significantly reduced and the numerical solution is thus closer to the reference. The optimized flux function indicates that less than 2.5% of the upwind fluxes is sufficient to obtain and maintain robust discontinuous solutions. Fig. 6 presents the numerical results of the multi-parameter model. With the increased degrees of freedom due to multiple parameters, localized numerical dissipation can be better controlled. As a result, the undershoot and oscillation near the tail of the rarefaction wave are mitigated, while the rest of the domain remains highly accurate. Fig. 7 shows the contribution proportions of central and upwind fluxes after optimization in the domain. It is clear that the flux function increases the share of upwind contributions near the shock wave, contact discontinuity, and front region of the rarefaction wave, thus enhancing the robustness of the numerical scheme in these highly dissipative regions. In other regions, the central-dominant flux effectively reduces the numerical dissipation of the scheme, ensuring that accurate physical solutions can be obtained.

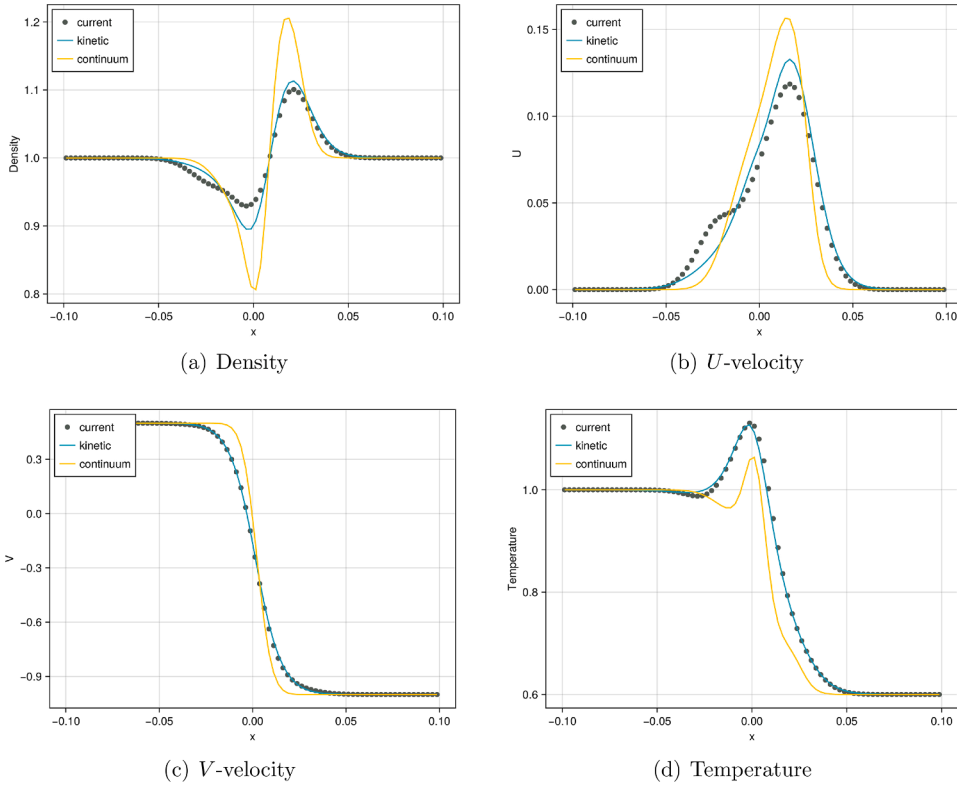


Fig. 13. Profiles of density, U -velocity, V -velocity, and temperature at $t = 5\tau_0$ simulated by different models in the shear layer problem (beyond the training set using different initial condition).

5.2. Identification of fluid property

Obtaining accurate fluid properties is a prerequisite for analysis and prediction. Due to the limited measurement accuracy and the sparseness of sensors, physical parameters of gases often need to be obtained indirectly through inversion [65]. Among others, viscosity is an important property, which determines the strength of pressure and viscous effects in different flow regimes. In this numerical experiment, a calibration problem of determining the viscosity coefficient from flow field data is considered.

Here we employ the hard-sphere model for monatomic gas. The dynamic viscosity coefficient for hard-sphere gas can be determined as

$$\mu = \mu_{\text{ref}} \left(\frac{T}{T_{\text{ref}}} \right)^{\eta}, \quad (46)$$

where μ_{ref} and T_{ref} denote the viscosity and temperature in the reference state, and $\eta = 0.5$ is the viscosity index. Once the viscosity is determined, the mean relaxation time can be obtained from the kinetic theory [66], i.e.,

$$\tau = \frac{1}{\nu} = \frac{\mu}{p}, \quad (47)$$

which can then be used to solve the BGK model equation.

The wave propagation problem is employed as the numerical experiment. The particle distribution function is initialized as Maxwellian, which corresponds to the following macroscopic variables,

$$\begin{pmatrix} \rho \\ U \\ p \end{pmatrix}_{t=0} = \begin{pmatrix} 1 + 0.1 \sin(2\pi x) \\ 1 \\ 0.5 \end{pmatrix}. \quad (48)$$

The system is non-dimensionalized by the domain length and initial unperturbed quantities. The computational setup is listed in Table 4, where Tsit5 refers to Tsitouras' 5/4 Runge-Kutta method [67]. To bound the prediction, the reference viscosity is set as the absolute value of the trainable parameter $\mathbf{p} \in \mathbb{R}^{N_p=1}$, i.e.,

$$\mu_{\text{ref}} = |\mathbf{p}|. \quad (49)$$

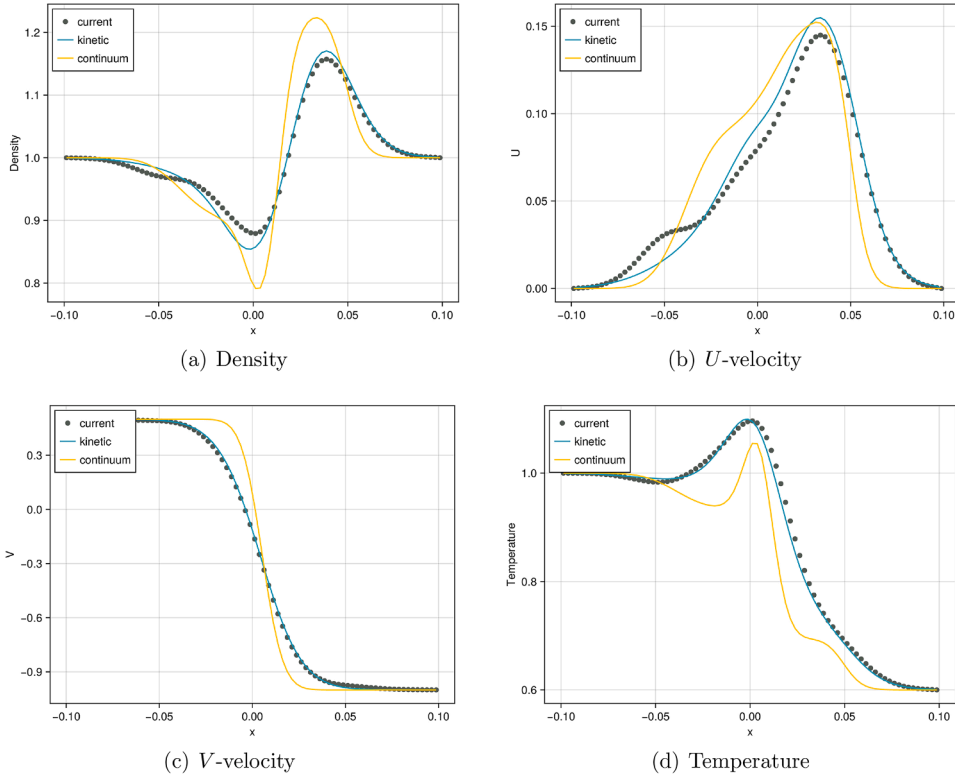


Fig. 14. Profiles of density, U -velocity, V -velocity, and temperature at $t = 10\tau_0$ simulated by different models in the shear layer problem (beyond the training set using different initial condition).

Table 4

Computational setup of the wave propagation problem.

Equation	Gas	t	x	N_x	Order	v	N_p	η
BGK	Argon	(0,0.25]	[0,1]	100	1	$[-5, 5]$	48	0.5
Flux Upwind	Quadrature Rectangular	Integrator Tsit5	Boundary Periodic	CFL 0.5	N_p 1	Optimizer LBFGS		

Table 5

Initial, optimized, and target values of dynamic viscosity coefficient in the wave propagation problem.

Target	Optimized	Initial
0.01	0.00986	10

The initial value of \mathbf{p} is 10.0. The cost function is defined as

$$C = \sum_i^{N_x} \sum_j^{N_t} \sum_k^{N_t} |f_{i,j}^{\text{ref}}(t_k) - f_{i,j}(t_k)|^2, \quad (50)$$

where the reference solution f^{ref} is the numerical solution at $\mu_{\text{ref}} = 0.01$.

Fig. 8 presents the density, velocity, and temperature profiles at $t = 0.25$ simulated with the initial and optimized parameters. The correct viscosity is recovered by aligning the trajectories of the numerical solution and reference target. Table 5 shows the solution of the parameter in the optimization problem. It can be seen that the accuracy of the solution is more than 98%. To illustrate the superiority of ∂P -based solution algorithm, we compare its performance with standard genetic algorithm (GA, with population size 150, mutation probability 10^{-5} , crossover probability 0.5) and ensemble Kalman inversion (EKI) [68,69]. The former is a classical population-based heuristic optimization technique that evolves encoded candidate solutions via selection, crossover, and mutation. The latter leverages the principles of the ensemble Kalman filter within the framework of the Bayesian inverse problem and is one

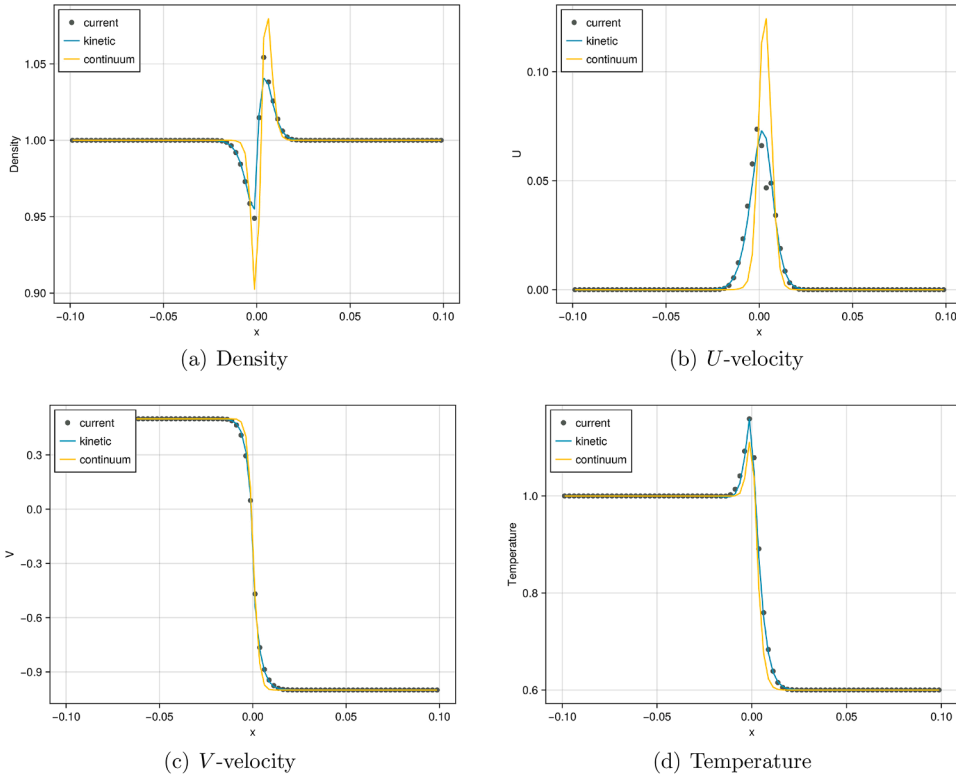


Fig. 15. Profiles of density, U -velocity, V -velocity, and temperature at $t = \tau_0$ simulated by different models in the shear layer problem (with fine-tuned model).

Table 6

Computational costs of the genetic algorithm (GA), ensemble Kalman inversion (EKI), and ∂P -based solution algorithm in the wave propagation problem.

	Time (s)	Allocation (GB)
GA	1096.30	1087.16
EKI	455.61	434.54
∂P	26.26	10.08

of the state-of-the-art gradient-free methods for solving optimization problems. To accelerate the convergence of the optimization problem, we incorporate the prior normal distribution $\mathbf{p} \sim \mathcal{N}(0, 0.1^2)$ to sample the initial ensemble of parameters. Table 6 provides the computational costs until the cost value reduces to $C = 0.00001$ based on ∂P , EKI, and GA, respectively. It can be seen that even in the case of strong intervention (by manually presetting parameter distributions closer to the true value), the computational time and allocations of EKI are still more than an order of magnitude higher than that of ∂P . This indicates the effectiveness and necessity of developing ∂P -based solution algorithms, compared to ensemble-based methods.

5.3. Construction of hydrodynamic closure

Due to the high dimensionality and strong nonlinearity of the Boltzmann equation, numerous efforts have been devoted to extending the applicability of hydrodynamic models in non-equilibrium flow regimes. The core task here is to construct reliable algebraic or evolutionary models for higher-order moment variables to approximate the particle distribution function, through which Eq. (7) becomes solvable. It is challenging since the particle distribution information has been partially filtered out during the coarse-grained upscaling modeling processes. Established theoretical work includes the Burnett and Super-Burnett equations based on the asymptotic Chapman-Enskog expansion, as well as moment equations based on monomials and polynomials hierarchies. Due to the aforementioned challenge, these efforts have achieved limited success within specific flow regimes.

Neural networks, as a multi-parameter model, provide an alternative for constructing hydrodynamic closures through a data-driven approach. Here, we construct a neural network-based constitutive model based on the Navier-Stokes equations that can depict

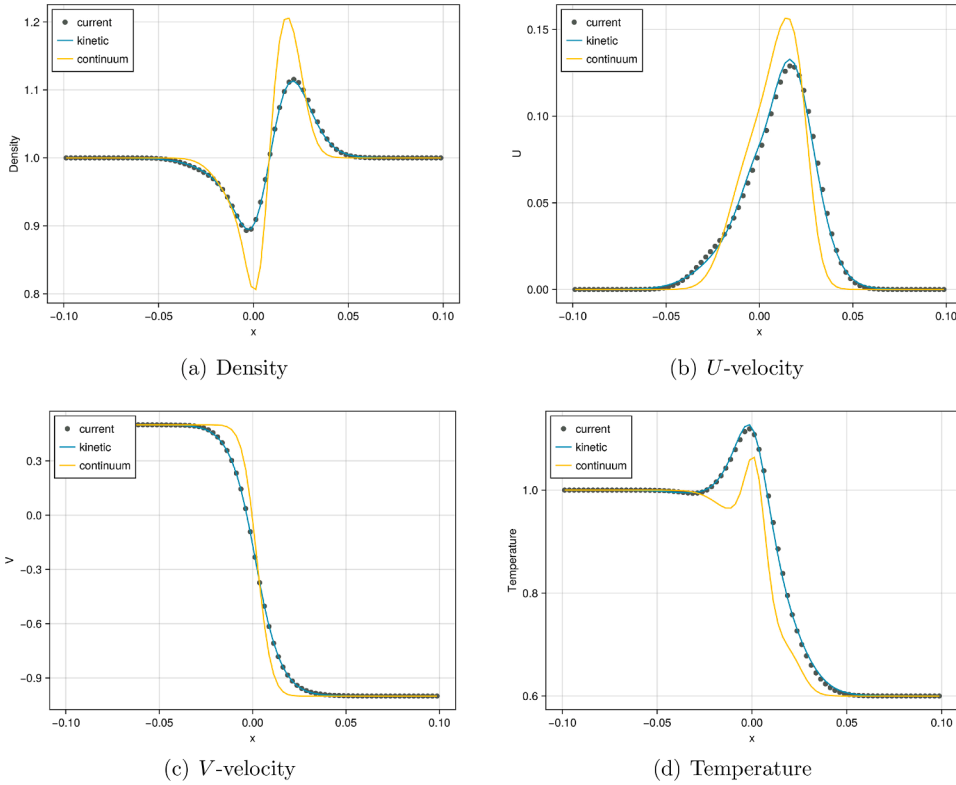


Fig. 16. Profiles of density, U -velocity, V -velocity, and temperature at $t = 5\tau_0$ simulated by different models in the shear layer problem (with fine-tuned model).

non-equilibrium flows more accurately. In the case of monatomic gas, for example, the constitutive relations in the Navier-Stokes equations include the generalized Newton's law and Fourier's law, i.e.,

$$\begin{aligned} \mathbf{P} &= -p\mathbf{I} + \mathbf{T}, \\ \mathbf{T} &= 2\mu(\nabla_{\mathbf{x}}\mathbf{V} + (\nabla_{\mathbf{x}}\mathbf{V})^T) - \frac{2}{3}\mu(\nabla_{\mathbf{x}} \cdot \mathbf{V})\mathbf{I}, \\ \mathbf{q} &= -\kappa\nabla_{\mathbf{x}}T, \end{aligned} \quad (51)$$

where \mathbf{T} is the stress tensor and \mathbf{I} refers to the unit tensor. In the computational framework of the finite volume method, the constitutive relations are usually incorporated in the flux function for ease of computation, e.g., the flux splitting scheme [70] and gas-kinetic flux solver [71]. Thus, we organize the unified model through the flux function, i.e.,

$$\mathbf{F}^W = \mathbf{F}^{\text{NS}} + \mathbf{F}^{\text{NN}}, \quad (52)$$

where \mathbf{F}^{NS} denotes the Navier-Stokes fluxes simulated by the gas-kinetic scheme, and \mathbf{F}^{NN} refers to their deviation from ground-truth non-equilibrium flow physics. The output values of the neural network are equal to \mathbf{F}^{NN} , while its inputs include macroscopic variables \mathbf{W} , their gradients $\nabla_{\mathbf{x}}\mathbf{W}$, and the Knudsen number Kn in the reference state.

The shear layer problem is employed as the numerical experiment. The flow field is initialized as

$$\begin{pmatrix} \rho \\ U \\ V \\ T \end{pmatrix}_{t=0, x<0} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} \rho \\ U \\ V \\ T \end{pmatrix}_{t=0, x\geq 0} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0.5 \end{pmatrix}. \quad (53)$$

The initial particle distribution function is Maxwellian everywhere in correspondence to the macroscopic variables. The system is non-dimensionalized by the domain length and physical quantities on the left side. The computational setup is provided in Table 7, where τ_0 denotes the mean relaxation time in the reference state. A fully connected neural network with 4 layers, 420 parameters, and tanh activation functions is employed to build the modified flux function. The cost function is defined as

$$C = \sum_i^{N_x} \sum_j^{N_t} \|\mathbf{W}_i^{\text{ref}}(t_j) - \mathbf{W}_i(t_j)\|^2 + \epsilon \|\boldsymbol{\theta}\|^2, \quad (54)$$

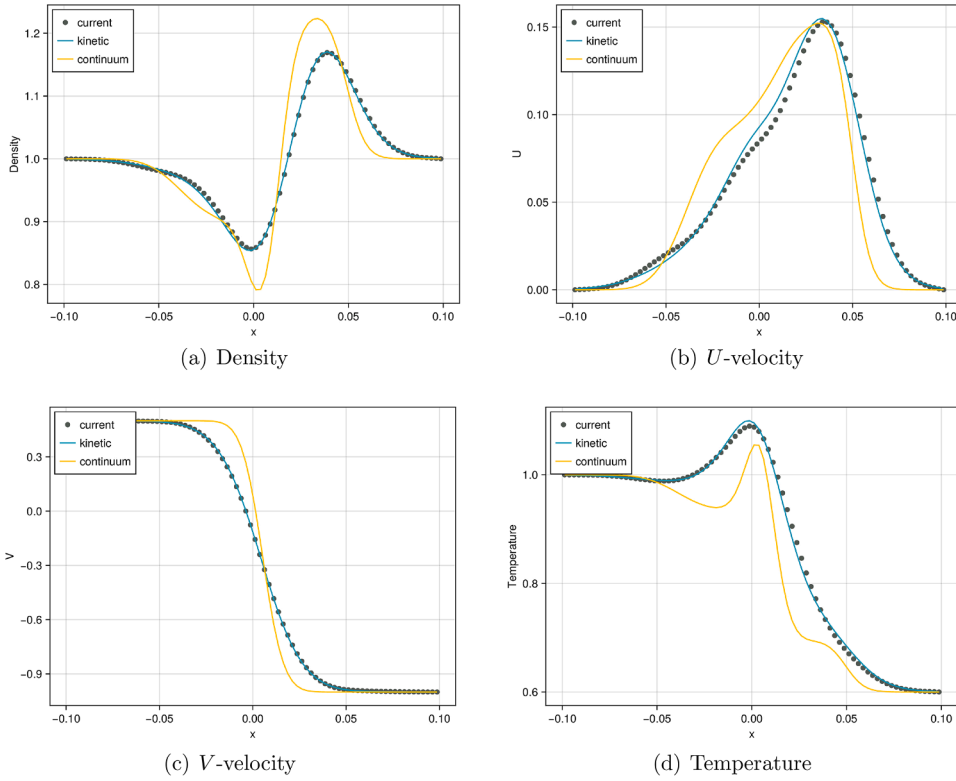


Fig. 17. Profiles of density, U -velocity, V -velocity, and temperature at $t = 10\tau_0$ simulated by different models in the shear layer problem (with fine-tuned model).

Table 7

Computational setup of the shear layer problem.

Equation	Gas	t	x	N_x	Order	Flux
Extended NS	Argon	$(0, 10\tau_0]$	$[-0.1, 0.1]$	100	1	GKS + NN
Integrator	Boundary	Kn	CFL	N_p	ϵ	Optimizer
Euler	Dirichlet	0.005	0.5	420	10^{-5}	AdamW

where the reference solution \mathbf{W}^{ref} is obtained by solving the BGK kinetic equation and applying moments to the distribution function.

Figs. 9–11 present the profiles of macroscopic flow variables at $t = \tau_0$, $5\tau_0$, and $10\tau_0$. The pressure-driven transport of momentum and energy forms a transition layer that thickens as time evolves. As the results show, due to the lack of effective non-equilibrium constitutive relations, the Navier-Stokes equations predict a narrower transition layer along with greater density fluctuations. With the supplement of the neural network-based closure model, non-equilibrium effects are well described within the framework of hydrodynamic equations, and the rate and pattern of viscous transport, which are identical to those of the BGK equation, are accurately recovered.

Due to the black-box nature of neural networks, the generalization capability of the closure model requires further verification. We apply the model to a new initial flow field, i.e.,

$$\begin{pmatrix} \rho \\ U \\ V \\ T \end{pmatrix}_{t=0, x<0} = \begin{pmatrix} 1 \\ 0 \\ 0.5 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} \rho \\ U \\ V \\ T \end{pmatrix}_{t=0, x\geq 0} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0.6 \end{pmatrix}, \quad (55)$$

while keeping all the rest computational parameters unchanged. Figs. 12–14 present the profiles of macroscopic flow variables at $t = \tau_0$, $5\tau_0$, and $10\tau_0$. It can be seen that the current model demonstrates satisfactory extrapolation capabilities and yields physically reasonable flow fields. While the generalization error induces a solution deviation of up to 6%, its accuracy still significantly surpasses that of the Navier-Stokes solutions. An effective approach to enhance the prediction accuracy of the model on out-of-distribution data is through fine-tuning. Figs. 15–17 illustrate the numerical experimental results after 100 iterations of fine-tuning. As evidenced, the limited fine-tunings have further improved the model's prediction accuracy. Note that the ∂P -based

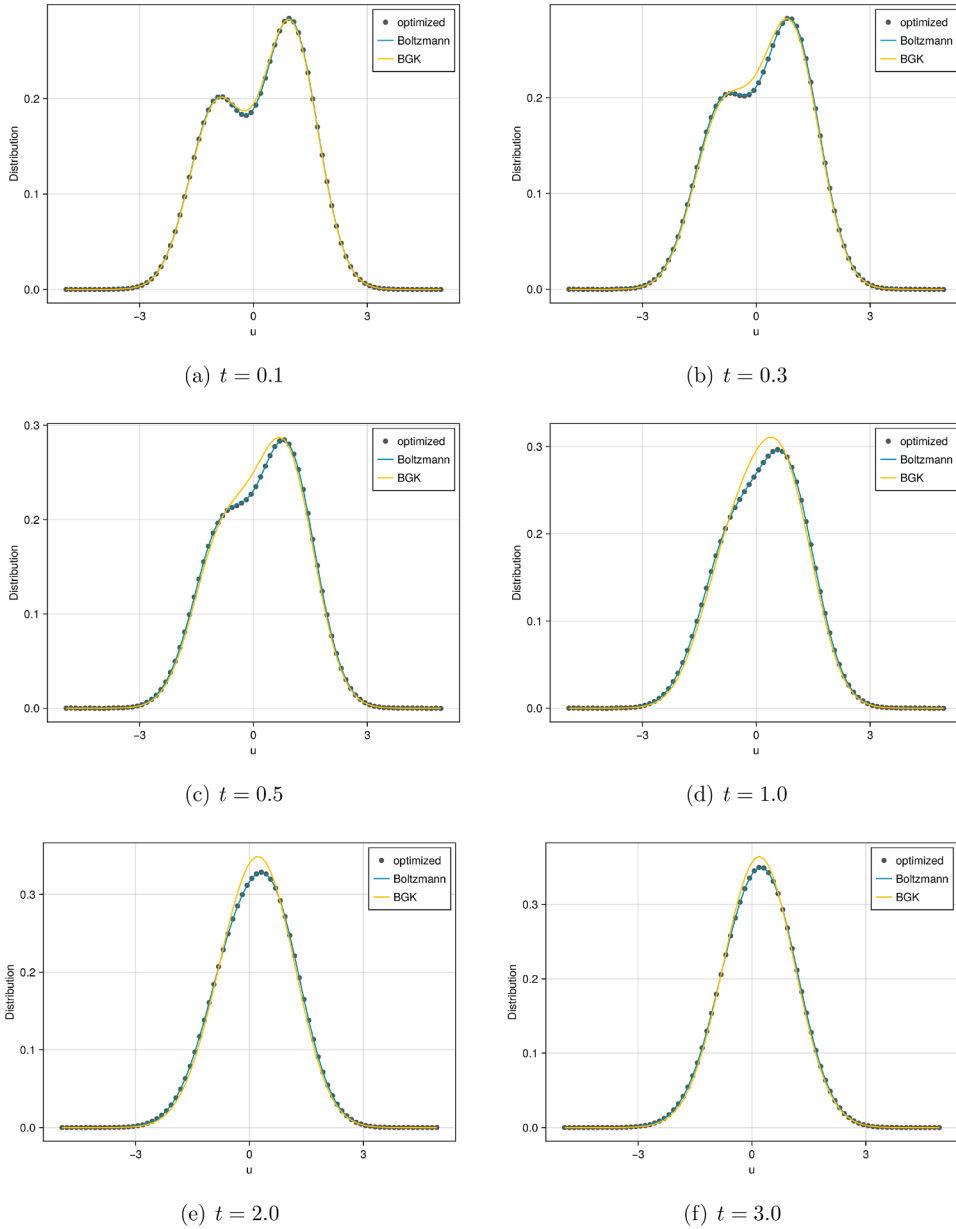


Fig. 18. Particle distribution functions at different time instants (in the training set) simulated by different models in the homogeneous relaxation problem (full Boltzmann equation as reference solution).

solution algorithm is inherently suitable for fine-tuning, as it facilitates real-time data supplementation throughout the solution process.

Table 8 presents the computational costs of a single computation of Navier-Stokes fluxes and neural network inference in the ∂P -based solution algorithm. It can be seen that the unified model dramatically improves the accuracy of hydrodynamic equations while adding only around 25% additional computational overhead.

5.4. Operator learning for the kinetic equation

Due to the ability of neural networks in feature identification and dimension reduction, an alternative to solving non-equilibrium flows is to directly solve the Boltzmann equation with the help of neural networks. Since the complexity of the algorithm for solving the Boltzmann equation lies mainly in the fivefold collision integral (larger than $O(N_v^6)$ for naive point-to-point computation), we construct the surrogate model for this operator based on the neural network. The model employed here is the deep operator network

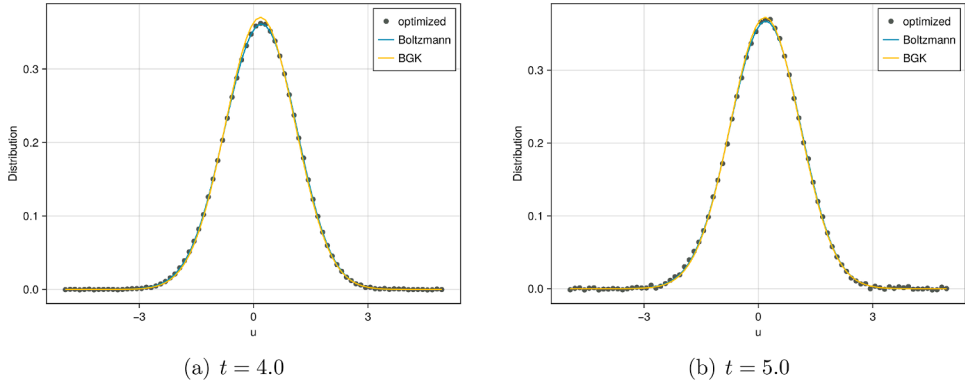


Fig. 19. Particle distribution functions at different time instants (beyond the training set through extrapolation) simulated by different models in the homogeneous relaxation problem (full Boltzmann equation as reference solution).

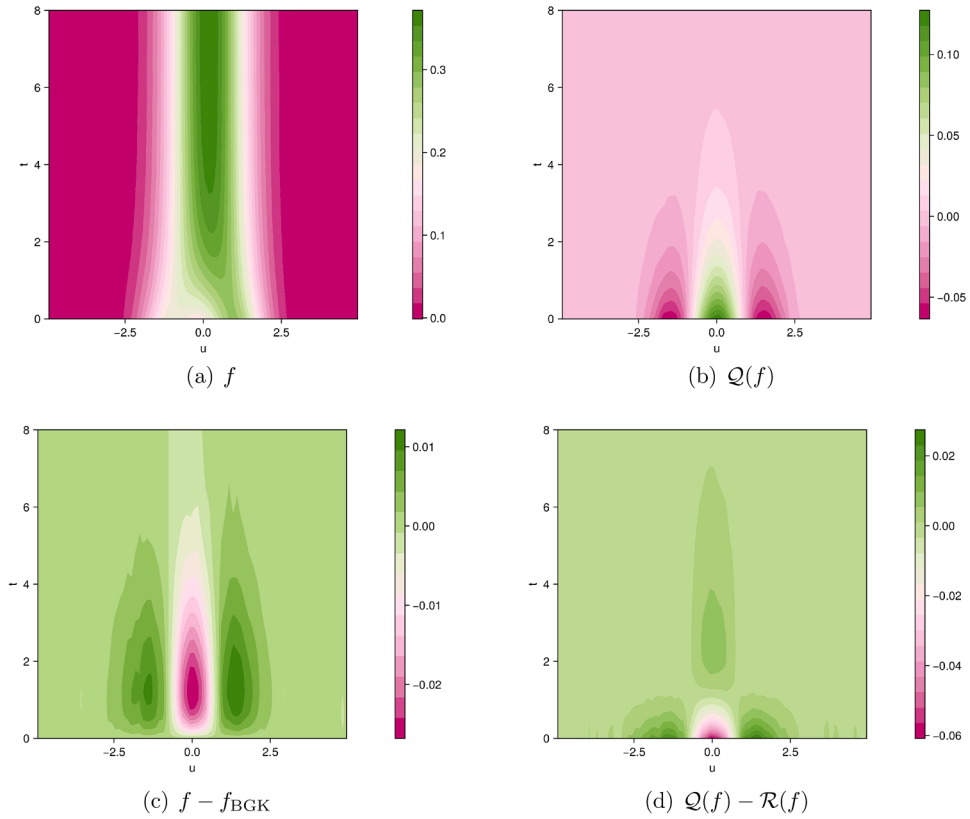


Fig. 20. Particle distribution functions, collision terms, and their differences over the time-velocity domain simulated by the DeepONet and BGK models in the homogeneous relaxation problem.

Table 8

Computational costs of Navier-Stokes fluxes and neural network model in the ∂P -based solution algorithm in the shear layer problem.

	Time (10^{-4} s)	Allocation (KB)
Navier-Stokes	1.43	13.36
Neural Network	0.37	5.20

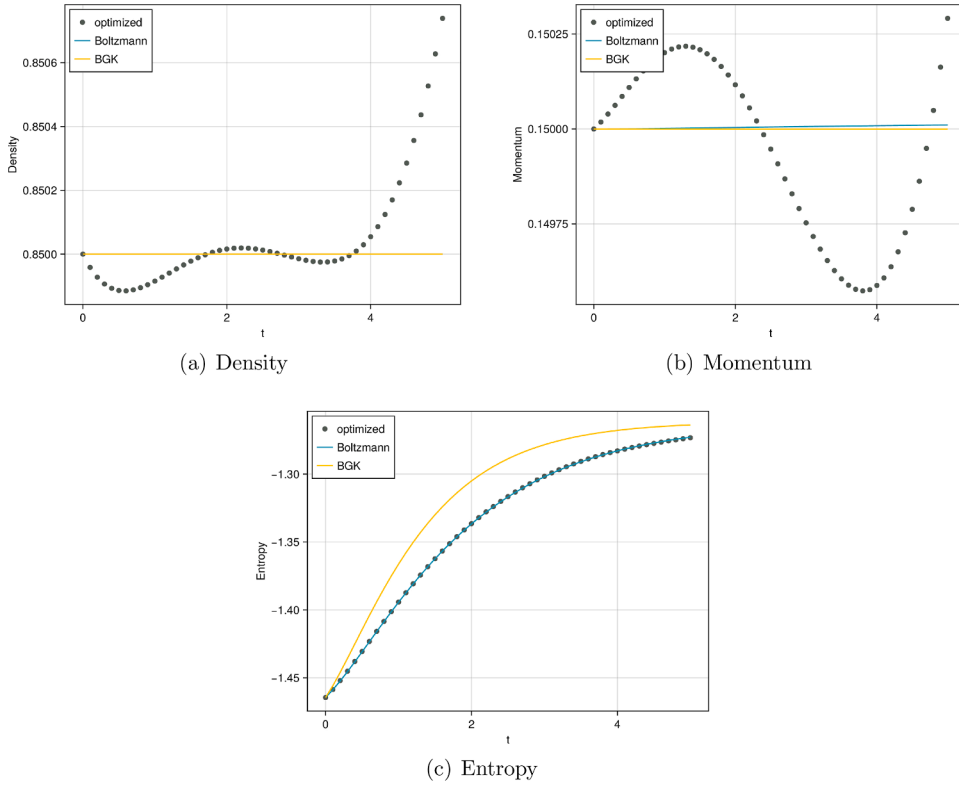


Fig. 21. Temporal evolution of macroscopic variables simulated by different models in the homogeneous relaxation problem (full Boltzmann equation as reference solution).

(DeepONet) [9], which is a neural architecture designed to learn nonlinear operator mappings between function spaces of infinite dimension. The model consists of two subnetworks, i.e., a branch net \mathcal{B} , which encodes the input functions evaluated at fixed sensor points into a finite-dimensional representation in the latent space, and a trunk net \mathcal{T} , which encodes the locations at which the output function is evaluated. The outputs of these networks are combined, typically via a dot product, to yield the output function's value at the specified location. In this case, the input function to \mathcal{B} is set as the particle distribution function evaluated at N_v collocation points in the velocity space, i.e., $\{f_j\}_{j=1:N_v}$, and the desired output is the collision term of the Boltzmann equation evaluated at the input velocity point $\mathbf{v} \in \mathbb{R}^d$, where d is the flow dimension of interest. The architecture of the DeepONet model is shown in Fig. 4. The DeepONet-enhanced Boltzmann equation can be written as

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = \frac{1}{Kn} \text{NN}_{\theta}(f). \quad (56)$$

5.4.1. Relaxation of non-equilibrium distribution

We first consider the relaxation of a non-equilibrium distributed many-particle system. The initial particle distribution function is set as

$$f = \frac{1}{2} \left(\frac{1}{\pi} \right)^{3/2} (\exp(-(u-1)^2) + 0.7 \exp(-(u+1)^2)) \exp(-v^2) \exp(-w^2). \quad (57)$$

We are mainly concerned with non-equilibrium effects in the x -direction, which can be characterized as

$$h = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f dv dw. \quad (58)$$

Based on the DeepONet, we expect to construct a one-dimensional model for the reduced distribution function h that can recover correct three-dimensional effects. The computational setup is detailed in Table 9. The DeepONet model employed consists of two fully connected neural networks with 4 layers, a total of 51600 parameters, and tanh activation functions. The cost function is defined as

$$C = \sum_j^{N_v} \sum_k^{N_t} |f_j^{\text{ref}}(t_k) - f_j(t_k)|^2, \quad (59)$$

where the reference solution f^{ref} is obtained by solving the Boltzmann equation with the fast spectral method [72,73] and then projecting it to one-dimensional space.

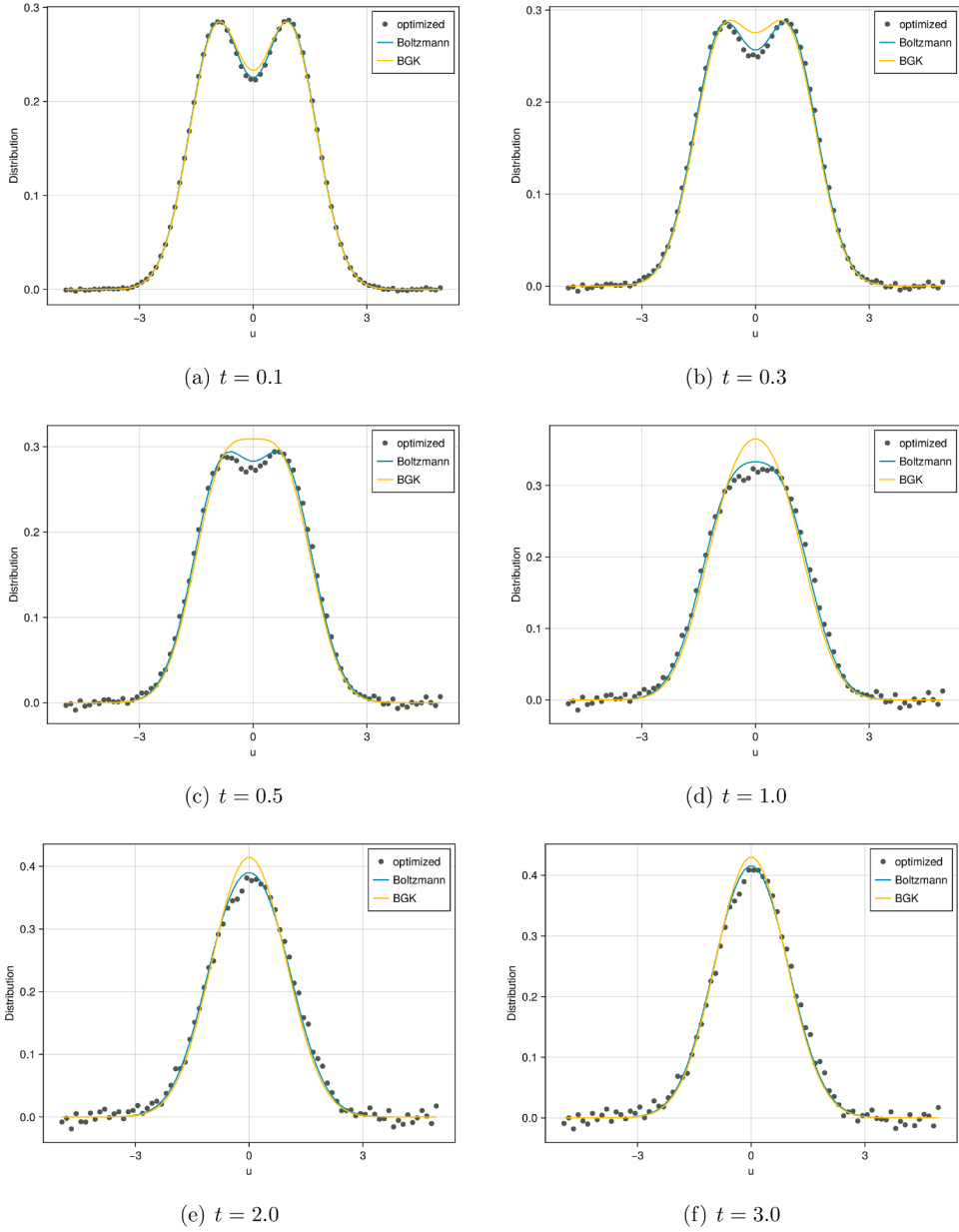


Fig. 22. Particle distribution functions at different time instants (beyond the training set using different initial distribution) simulated by different models in the homogeneous relaxation problem (full Boltzmann equation as reference solution).

Table 9

Computational setup of the relaxation problem.

Equation	Gas	t	\mathbf{v}	N_{v_x}	N_{v_y}	N_{v_z}
Boltzmann	Argon	(0,3]	$[-5, 5]^3$	80	28	28
Quadrature	Integrator	Kn	CFL	N_p	Optimizer	
Rectangular	Tsit5	1	0.5	51600	LBFGS	

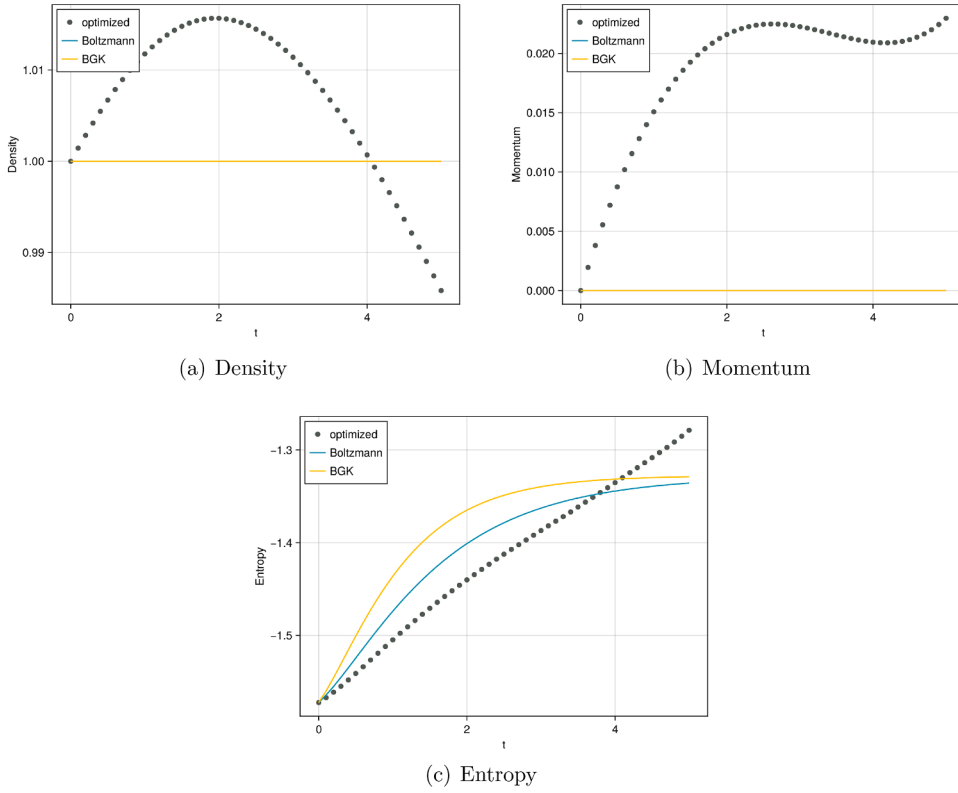


Fig. 23. Temporal evolution of macroscopic variables (beyond the training set using different initial distribution) simulated by different models in the homogeneous relaxation problem (full Boltzmann equation as reference solution).

Fig. 18 presents the solutions of particle distribution function at different time instants in the training set simulated by the Boltzmann, BGK, and the current ∂P -based model. It can be seen that the current model outperforms the BGK model in terms of accuracy, and it provides a non-equilibrium evolutionary solution equivalent to the Boltzmann equation. Fig. 19 presents the extrapolation results from the same initial distribution to extended time durations. It can be seen that the current model demonstrates satisfactory extrapolation capabilities and effectively converges toward the equilibrium state. Fig. 20 provides the difference between the distribution function and the collision term provided by these two models over the entire time-velocity domain. Clearly, it is the difference in the collision terms provided by the BGK equation (i.e., $\mathcal{R}(f)$) and the DeepONet (i.e., $\mathcal{Q}(f)$) for the non-equilibrium distribution function that leads to the different solutions of the two models. As the intermolecular interactions occur, the distribution function gradually approaches the equilibrium state, at which point the results of the two models converge. Fig. 21 illustrates the temporal evolution of macroscopic density, momentum, and entropy (defined as $\int f \log f d\mathbf{v}$). The computational results demonstrate compliance with the physical structure of conservation and entropy dissipation (H-theorem) as prescribed by the Boltzmann equation.

Due to the black-box nature of the DeepONet in Eq. (56), its generalization capability is generally limited. We apply the model based on Eq. (57) to the relaxation of a new initial distribution function, i.e.,

$$f = \frac{1}{2} \left(\frac{1}{\pi} \right)^{3/2} (\exp(-(u-1)^2) + \exp(-(u+1)^2)) \exp(-v^2) \exp(-w^2), \quad (60)$$

while keeping all other computational parameters unchanged. Fig. 22 presents the simulation results at different time instants. It can be seen that the generalization error introduces approximately a 5% deviation in the solutions. Fig. 23 provides the temporal evolution of macroscopic density, momentum, and entropy. The numerical errors also manifest in the conservation properties of the collision term, while concurrently leading to incorrect entropy dissipation rates. Different methods can be employed to enhance the generalization performance of the model. A viable solution involves incorporating additional physical constraints. As shown in the following equation, the BGK model can be embedded into the data-augmented operator.

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = \frac{1}{\text{Kn}} \text{NN}_{\theta}(f) + \nu(\mathcal{M} - f). \quad (61)$$

In this case, the DeepONet is sorely required to approximate the discrepancy between the Boltzmann and BGK collision terms, thereby better preserving the physical structure in the out-of-distribution scenarios. The numerical experimental results presented

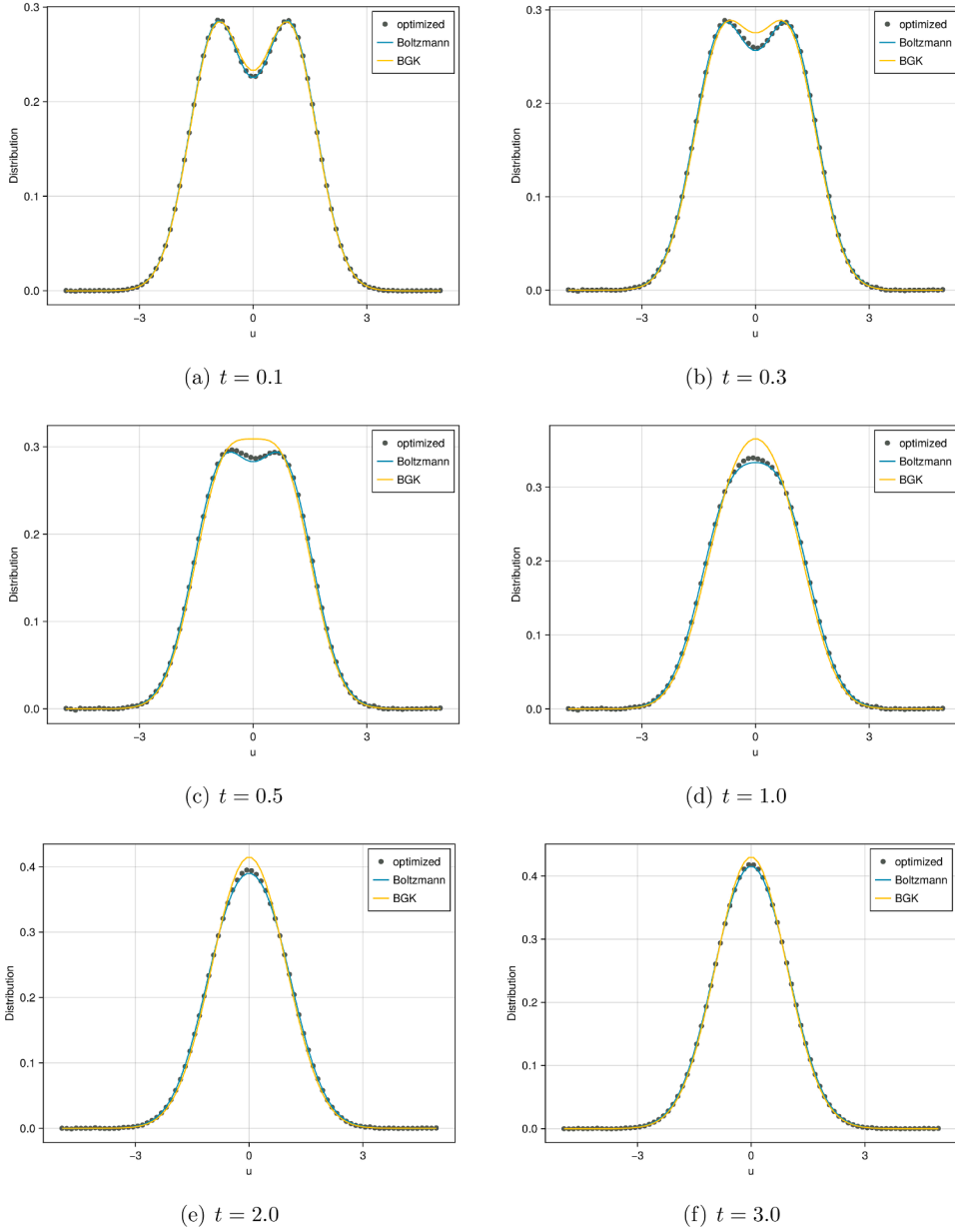


Fig. 24. Particle distribution functions at different time instants (beyond the training set using different initial distribution) simulated by different models (with physics-augmented DeepONet) in the homogeneous relaxation problem (full Boltzmann equation as reference solution).

in Fig. 24 provide compelling evidence supporting this argument. The excellent performance of the model is also demonstrated in terms of conservation properties and entropy dissipation, as illustrated in Fig. 25. The ∂P -based solution algorithm provides a convenient platform for seamless model switching. It also accommodates additional methodologies for improving generalization capabilities, including transfer learning. For a comprehensive exploration of this topic, we refer the interested reader to Reference [74].

Table 10 presents the computational costs of a single simulation using the Boltzmann equation, the BGK model, and the ∂P -based model. Since the high-dimensional convolution operations in the fast spectral method are replaced by the tensor summations and products in the DeepONet, the computational cost is significantly reduced. With the current parameter settings, the computational efficiency has improved by more than three orders of magnitude. This numerical experiment verifies the ability of the ∂P -based solution algorithm to solve the Boltzmann equation accurately and efficiently.

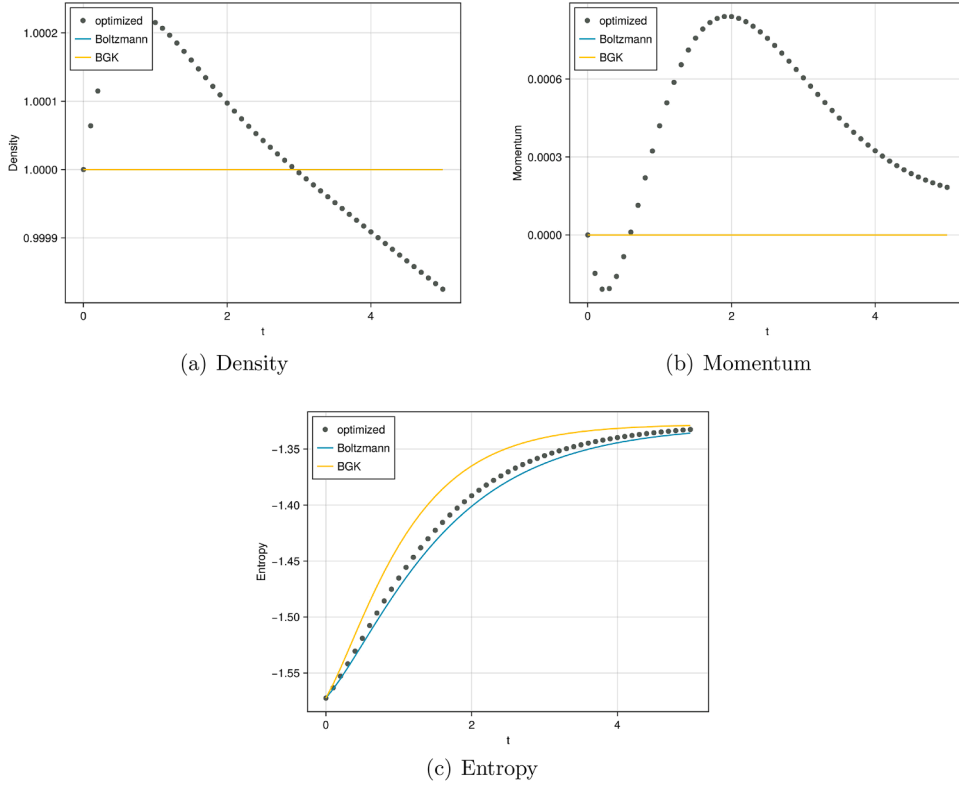


Fig. 25. Temporal evolution of macroscopic variables (beyond the training set using different initial distribution) simulated by different models (with physics-augmented DeepONet) in the homogeneous relaxation problem (full Boltzmann equation as reference solution).

Table 10

Computational costs of a single simulation using the Boltzmann equation, the BGK equation, and the data-augmented unified model in the relaxation problem.

	Time (10^{-3} s)	Allocation (MB)
Boltzmann	2.32×10^3	6.35×10^3
BGK	0.90	1.37
∂P	3.91	47.94

5.4.2. Normal shock wave structure

We then turn to the normal shock wave structure problem. Based on the reference frame of the shock wave, the initial flow field is set as

$$\begin{pmatrix} \rho \\ U \\ T \end{pmatrix}_{t=0, x<0} = \begin{pmatrix} \rho_- \\ U_- \\ T_- \end{pmatrix}, \quad \begin{pmatrix} \rho \\ U \\ T \end{pmatrix}_{t=0, x \geq 0} = \begin{pmatrix} \rho_+ \\ U_+ \\ T_+ \end{pmatrix}, \quad (62)$$

where the subscripts $-$ and $+$ denote the upstream and downstream states of the shock wave, respectively. The upstream and downstream conditions are related through the Rankine-Hugoniot relation, i.e.,

$$\begin{aligned} \frac{\rho_+}{\rho_-} &= \frac{(\gamma + 1)\text{Ma}^2}{(\gamma - 1)\text{Ma}^2 + 2}, \\ \frac{U_+}{U_-} &= \frac{(\gamma - 1)\text{Ma}^2 + 2}{(\gamma + 1)\text{Ma}^2}, \\ \frac{T_+}{T_-} &= \frac{((\gamma - 1)\text{Ma}^2 + 2)(2\gamma\text{Ma}^2 - \gamma + 1)}{(\gamma + 1)^2\text{Ma}^2}, \end{aligned} \quad (63)$$

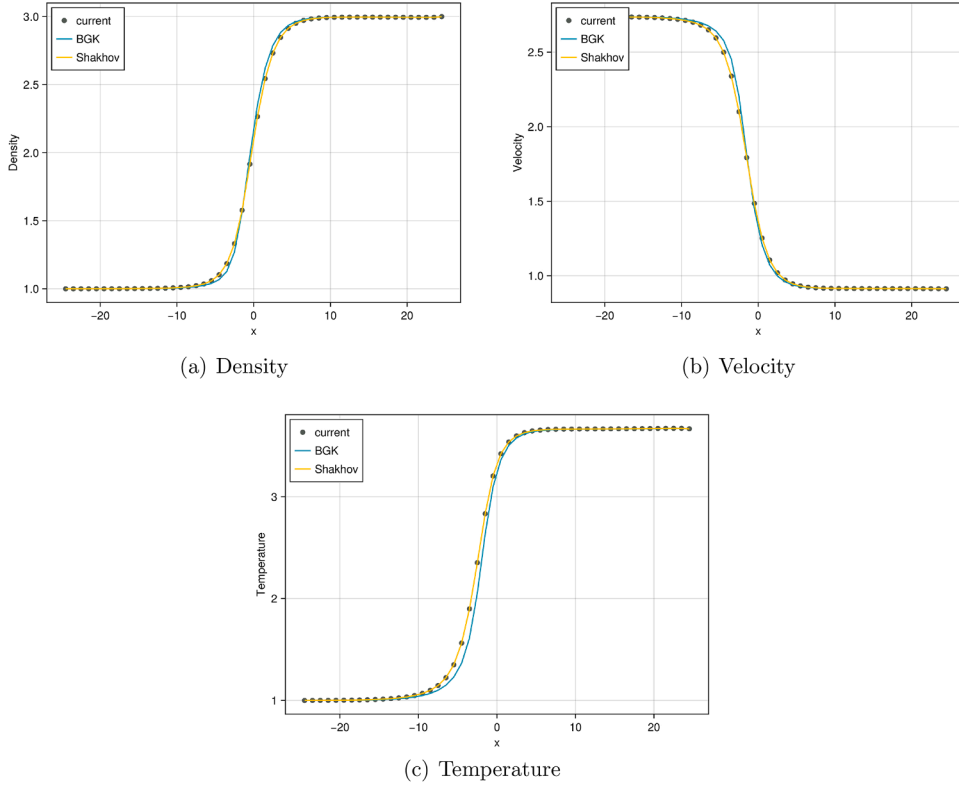


Fig. 26. Profiles of density, U -velocity, and temperature simulated by different models in the normal shock wave structure problem (Shakhov model as reference solution).

Table 11

Computational setup of the normal shock wave structure problem.

Equation	Gas	t	x	N_x	Order	v	N_v	Ma
Boltzmann	Argon	(0,50]	[-25, 25]	50	1	[-5, 5]	36	3
Flux Upwind	Quadrature Rectangular	Integrator Euler	Boundary Dirichlet	CFL 0.5	N_p 20736	Optimizer AdamW		

where Ma is the upstream Mach number, and the specific heat ratio takes $\gamma = 5/3$ for a monatomic molecule. The initial particle distribution function is set as Maxwellian in correspondence with the above macroscopic variables. The reference state corresponds to the upstream conditions, and the system is non-dimensionalized by the upstream molecular mean free path. The computational setup is detailed in Table 11. The DeepONet model consists of two fully connected neural networks with 4 layers, a total of 20736 parameters, and tanh activation functions. The cost function is defined as

$$C = \sum_i^{N_x} \sum_j^{N_v} |f_{i,j}^{\text{ref}}(t = 50) - f_{i,j}(t = 50)|^2, \quad (64)$$

where the reference solution f^{ref} is obtained by solving the Shakhov model equation with the same computational setup, which provides more accurate solutions for the evolution of high-temperature gases thanks to the heat flux-based correction.

Fig. 26 presents the distributions of density, velocity, and temperature simulated by the Boltzmann equation, the BGK equation, and the current unified model. The current model provides predictions that are equivalent to the reference solution. Fig. 27 details the contours of distribution functions and collision terms for both models over the entire space-velocity domain. As is shown, within the range of the shock wave (the width is of $O(10\ell)$), the flow variables change dramatically, leading to intensive intermolecular collisions. Accurate prediction of collision effects is a prerequisite for capturing the correct shock profile. The difference in the collision terms leads to different distribution functions and the corresponding macroscopic variables of the BGK and the current unified models. This numerical experiment verifies the ability of the current ∂P -based solution algorithm to solve highly dissipative non-equilibrium flows with spatial inhomogeneity.

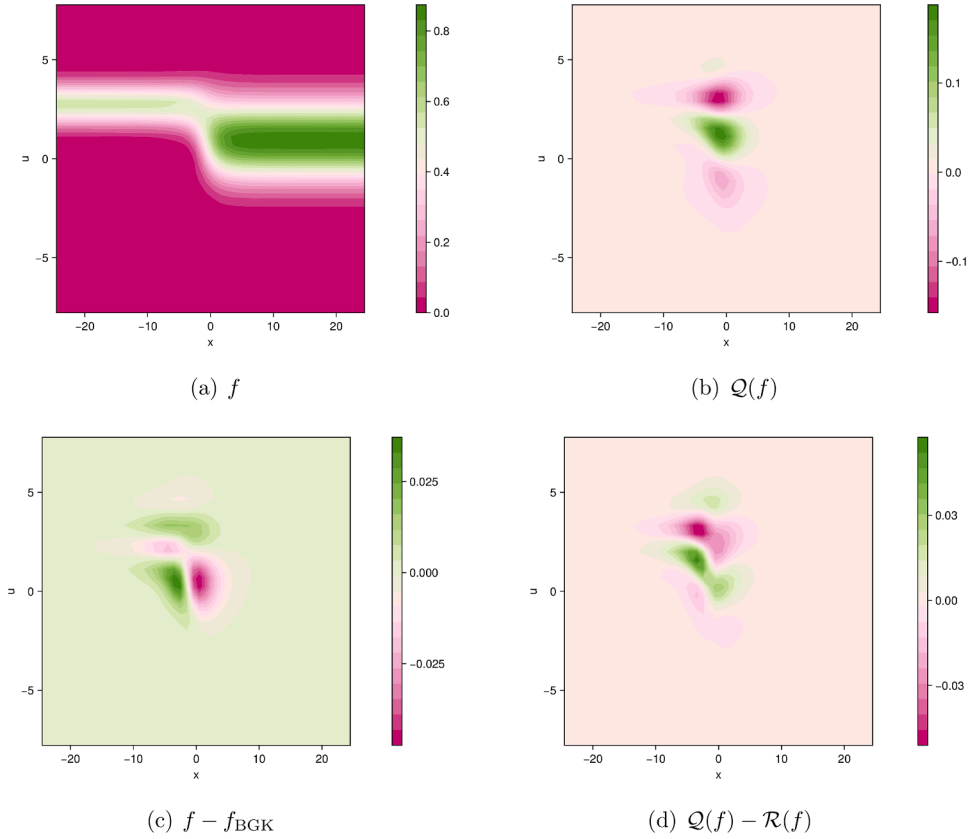


Fig. 27. Particle distribution functions, collision terms, and their differences over the space-velocity domain simulated by the unified model and BGK equation in the normal shock wave structure problem.

6. Conclusion

Research on multi-scale and non-equilibrium flows faces challenges arising from high dimensionality and strong nonlinearity in theoretical models and solution algorithms. For the first time, this paper systematically addresses the application of differentiable programming to the construction of solution algorithms for multi-scale flow physics across continuum and rarefied regimes. The distinctiveness of this work resides in its seamless integration of automatic differentiation with adjoint sensitivity analysis. The fully differentiable simulator provides a unified framework that seamlessly integrates end-to-end optimization of physical models and solution algorithms with forward numerical simulation by computing gradients throughout the backward passes of the simulation program. As a result, classical scientific computing and machine learning workflows are organically fused. The paradigm of differentiable simulation lays a solid foundation for building unified flow models, enabling versatile data-driven approaches for physics discovery, surrogate modeling, and simulation acceleration. It has great potential to be extended to the study of other complex systems, e.g., complex-geometry flows [75,76], radiative transfer [77,78], plasma physics [79,80], stochastic simulation [81,82], and so on. We aim to further enhance its applicability in massively parallel computing in future work, e.g., constructing MPI-compatible automatic differentiation and differentiable programming systems [83].

CRediT authorship contribution statement

Tianbai Xiao: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The author declares that there are no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The current research is funded by the [National Science Foundation](#) of China (No. 12302381) and the [Chinese Academy of Sciences](#) Project for Young Scientists in Basic Research (YSBR-107). The computing resources provided by Hefei Advanced Computing Center and ORISE Supercomputer are acknowledged.

References

- [1] H.-S. Tsien, Superaerodynamics, mechanics of rarefied gases, *J. Aeronaut. Sci.* 13 (12) (1946) 653–664.
- [2] D. Hilbert, Mathematical problems, *Bull. Amer. Math. Soc* 8 (1902) 437–479.
- [3] Y. Sone, *Kinetic Theory and Fluid Dynamics*, Springer, 2002.
- [4] S. Jin, Asymptotic-preserving schemes for multiscale physical problems, *Acta Numer.* 31 (2022) 415–489.
- [5] C. Cercignani, *The Boltzmann Equation and Its Applications*, Springer, 1988.
- [6] P. Rosenau, Extending hydrodynamics via the regularization of the Chapman-Enskog expansion, *Phys. Rev. A* 40 (12) (1989) 7193.
- [7] 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.
- [8] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, *Proc. Natl. Acad. Sci.* 115 (34) (2018) 8505–8510.
- [9] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nat. Mach. Intell.* 3 (3) (2021) 218–229.
- [10] Z. Li, N.B. Kovachki, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, et al., Fourier neural operator for parametric partial differential equations, in: *International Conference on Learning Representations*, 2020.
- [11] D. Floryan, M.D. Graham, Data-driven discovery of intrinsic dynamics, *Nat. Mach. Intell.* 4 (12) (2022) 1113–1120.
- [12] S. Pan, K. Duraisamy, Data-driven discovery of closure models, *SIAM J. Appl. Dyn. Syst.* 17 (4) (2018) 2381–2413.
- [13] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.
- [14] G. Menghani, Efficient deep learning: a survey on making deep learning models smaller, faster, and better, *ACM Comput. Surv.* 55 (12) (2023) 1–37.
- [15] J.C. De los Reyes, *Numerical PDE-Constrained Optimization*, Springer, 2015.
- [16] S.G. Johnson, Notes on adjoint methods for 18.335, *Introduction to Numerical Methods* (2012).
- [17] G. Strang, *Computational Science and Engineering*, SIAM, 2007.
- [18] D. Givoli, A tutorial on the adjoint method for inverse problems, *Comput. Methods Appl. Mech. Eng.* 380 (2021) 113810.
- [19] A. Jameson, Aerodynamic shape optimization using the adjoint method, *Lectures at the Von Karman Institute, Brussels* (2003).
- [20] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, *Advances in neural information processing systems vol. 31* (2018).
- [21] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, A. Edelman, Universal differential equations for scientific machine learning, (2020). [arXiv preprint arXiv:2001.04385](#)
- [22] M. Blondel, V. Roulet, The elements of differentiable programming, (2024). [arXiv preprint arXiv:2403.14606](#)
- [23] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. Vander Plas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python + NumPy programs, <http://github.com/google/jax> (2018).
- [24] J. Liesen, Z. Strakos, Krylov subspace methods: principles and analysis, *Numerical Mathematics and Science*, 2013.
- [25] B.-P.F. De Avila, T. Economou, Z. Kolter, Combining differentiable PDE solvers and graph neural networks for fluid flow prediction, in: *International Conference on Machine Learning*, PMLR, 2020, pp. 2402–2411.
- [26] J. Zhuang, D. Kochkov, Y. Bar-Sinai, M.P. Brenner, S. Hoyer, Learned discretizations for passive scalar advection in a two-dimensional turbulent flow, *Phys. Rev. Fluids* 6 (6) (2021) 064605.
- [27] D.A. Bezgin, A.B. Buhendwa, N.A. Adams, JAX-Fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows, *Comput. Phys. Commun.* 282 (2023) 108527.
- [28] X. Fan, J.-X. Wang, Differentiable hybrid neural modeling for fluid-structure interaction, *J. Comput. Phys.* 496 (2024) 112584.
- [29] J. Ho, C. Farhat, Aerodynamic optimization with large shape and topology changes using a differentiable embedded boundary method, *J. Comput. Phys.* 488 (2023) 112191.
- [30] D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, S. Hoyer, Machine learning-accelerated computational fluid dynamics, *Proc. Natl. Acad. Sci.* 118 (21) (2021) e2101784118.
- [31] K. Um, R. Brand, Y.R. Fei, P. Holl, N. Thuerey, Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers, *Adv. Neural Inf. Process. Syst.* 33 (2020) 6111–6122.
- [32] Q. Lou, X. Meng, G.E. Karniadakis, Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation, *J. Comput. Phys.* 447 (2021) 110676.
- [33] M. De Florio, E. Schiassi, B.D. Ganapol, R. Furfaro, Physics-informed neural networks for rarefied-gas dynamics: thermal creep flow in the Bhatnagar–Gross–Krook approximation, *Phys. Fluids* 33 (4) (2021).
- [34] M. De Florio, E. Schiassi, B.D. Ganapol, R. Furfaro, Physics-informed neural networks for rarefied-gas dynamics: poiseuille flow in the BGK approximation, *Z. Angew. Math. Phys.* 73 (3) (2022) 126.
- [35] J.-M. Tucny, M. Lauricella, M. Durve, G. Guglielmo, A. Montessori, S. Succi, Physics-informed neural networks for microflows: rarefied gas dynamics in cylinder arrays, *J. Comput. Sci.* 87 (2025) 102575.
- [36] J.-M. Tucny, M. Durve, A. Montessori, S. Succi, Learning of viscosity functions in rarefied gas flows with physics-informed neural networks, *Comput. Fluids* 269 (2024) 106114.
- [37] L. Zhang, W. Ma, Q. Lou, J. Zhang, Simulation of rarefied gas flows using physics-informed neural network combined with discrete velocity method, *Phys. Fluids* 35 (7) (2023).
- [38] T. Xiao, M. Frank, Using neural networks to accelerate the solution of the Boltzmann equation, *J. Comput. Phys.* 443 (2021) 110521.
- [39] S.T. Miller, N.V. Roberts, S.D. Bond, E.C. Cyr, Neural-network based collision operators for the Boltzmann equation, *J. Comput. Phys.* 470 (2022) 111541.
- [40] Z. Li, Y. Wang, H. Liu, Z. Wang, B. Dong, Solving the Boltzmann equation with a neural sparse representation, *SIAM J. Sci. Comput.* 46 (2) (2024) C186–C215.
- [41] J. Huang, Y. Cheng, A.J. Christlieb, L.F. Roberts, Machine learning moment closure models for the radiative transfer equation I: directly learning a gradient based closure, *J. Comput. Phys.* 453 (2022) 110941.
- [42] S. Schotthöfer, T. Xiao, M. Frank, C.D. Hauck, Structure preserving neural networks: a case study in the entropy closure of the Boltzmann equation, in: *ICML*, 2022, pp. 19406–19433.

- [43] J. Han, C. Ma, Z. Ma, W. E, Uniformly accurate machine learning-based hydrodynamic models for kinetic equations, *Proc. Natl. Acad. Sci.* 116 (44) (2019) 21983–21991.
- [44] A.J. Christlieb, M. Ding, J. Huang, N.A. Krupansky, Hyperbolic machine learning moment closures for the BGK equations, *Multiscale Model. Simul.* 23 (1) (2025) 187–217.
- [45] M. Innes, A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V.B. Shah, W. Tebbutt, A differentiable programming system to bridge machine learning and scientific computing, (2019). *arXiv preprint arXiv:1907.07587*
- [46] W. Moses, V. Churavy, Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients, *Adv. Neural Inf. Process. Syst.* 33 (2020) 12472–12485.
- [47] T. Xiao, Kinetic.jl: a portable finite volume toolbox for scientific and neural computing, *J. Open Source Softw.* 6 (62) (2021) 3060.
- [48] G.W. Alldredge, M. Frank, C.D. Hauck, A regularized entropy-based moment method for kinetic equations, *SIAM J. Appl. Math.* 79 (5) (2019) 1627–1653.
- [49] T. Xiao, M. Frank, RelaxNet: a structure-preserving neural network to approximate the Boltzmann collision operator, *J. Comput. Phys.* 490 (2023) 112317.
- [50] M. Torrilhon, Modeling nonequilibrium gas flow based on moment equations, *Annu. Rev. Fluid Mech.* 48 (1) (2016) 429–458.
- [51] R.K. Brayton, F.G. Gustavson, G.D. Hachtel, A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas, *Proc. IEEE* 60 (1) (1972) 98–108.
- [52] A. Jameson, Evaluation of fully implicit Runge Kutta schemes for unsteady flow calculations, *J. Sci. Comput.* 73 (2) (2017) 819–852.
- [53] U.M. Ascher, S.J. Ruuth, B.T.R. Wetton, Implicit-explicit methods for time-dependent partial differential equations, *SIAM J. Numer. Anal.* 32 (3) (1995) 797–823.
- [54] T. Xiao, A flux reconstruction kinetic scheme for the Boltzmann equation, *J. Comput. Phys.* 447 (2021) 110689.
- [55] R.M. Errico, What is an adjoint model?, *Bull. Am. Meteorol. Soc.* 78 (11) (1997) 2577–2592.
- [56] Y. Cao, S. Li, L. Petzold, R. Serban, Adjoint sensitivity analysis for differential-algebraic equations: the adjoint DAE system and its numerical solution, *SIAM J. Sci. Comput.* 24 (3) (2003) 1076–1089.
- [57] J. Revels, M. Lubin, T. Papamarkou, Forward-mode automatic differentiation in Julia, (2016). *arXiv preprint arXiv:1607.07892*
- [58] F. White et al., JuliaDiff/ChainRules.jl: v1.72.0, <https://doi.org/10.5281/zenodo.14064067> (2024).
- [59] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics* Paris France, August 22–27, 2010 Keynote, Invited and Contributed Papers, Springer, 2010, pp. 177–186.
- [60] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Programm.* 45 (1) (1989) 503–528.
- [61] S. Evje, T. Flåtten, Hybrid central-upwind schemes for numerical resolution of two-phase flows, *ESAIM Math. Modell. Numer. Anal.* 39 (2) (2005) 253–273.
- [62] A. Kurganov, C.-T. Lin, On the reduction of numerical dissipation in central-upwind schemes, *Commun. Comput. Phys.* 2 (1) (2007) 141–163.
- [63] K. Xu, Direct Modeling for Computational Fluid Dynamics: Construction and Application of Unified Gas-Kinetic Schemes, vol. 4, World Scientific, 2014.
- [64] Y. Sun, C. Shu, C.J. Teo, Y. Wang, L.M. Yang, Explicit formulations of gas-kinetic flux solver for simulation of incompressible and compressible viscous flows, *J. Comput. Phys.* 300 (2015) 492–519.
- [65] T. Liu, J. Montfort, S. Stanfield, S. Palluconi, J. Crafton, Z. Cai, Inverse heat transfer methods for global heat flux measurements in aerothermodynamics testing, *Prog. Aerosp. Sci.* 107 (2019) 1–18.
- [66] W.G. Vincenti, C.H. Kruger Jr, T. Teichmann, Introduction to physical gas dynamics, Krieger Publishing Company, 1965.
- [67] C. Tsitouras, I.T. Famelis, T.E. Simos, On modified Runge–Kutta trees and methods, *Comput. Math. Appl.* 62 (4) (2011) 2101–2111.
- [68] M.A. Iglesias, K.J.H. Law, A.M. Stuart, Ensemble Kalman methods for inverse problems, *Inverse Probl.* 29 (4) (2013) 045001.
- [69] N.B. Kovachki, A.M. Stuart, Ensemble Kalman inversion: a derivative-free technique for machine learning tasks, *Inverse Probl.* 35 (9) (2019) 095005.
- [70] Y. Wada, M.-S. Liou, An accurate and robust flux splitting scheme for shock and contact discontinuities, *SIAM J. Sci. Comput.* 18 (3) (1997) 633–657.
- [71] K. Xu, A gas-kinetic BGK scheme for the Navier–Stokes equations and its connection with artificial dissipation and Godunov method, *J. Comput. Phys.* 171 (1) (2001) 289–335.
- [72] C. Mouhot, L. Pareschi, Fast algorithms for computing the Boltzmann collision operator, *Math. Comput.* 75 (256) (2006) 1833–1852.
- [73] L. Wu, C. White, T.J. Scanlon, J.M. Reese, Y. Zhang, Deterministic numerical solutions of the Boltzmann equation using the fast spectral method, *J. Comput. Phys.* 250 (2013) 27–52.
- [74] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, *Proc. IEEE* 109 (1) (2020) 43–76.
- [75] A. Montessori, P. Prestininzi, M. La Rocca, S. Succi, Lattice Boltzmann approach for complex nonequilibrium flows, *Phys. Rev. E* 92 (4) (2015) 043308.
- [76] A. Montessori, P. Prestininzi, M. La Rocca, G. Falcucci, S. Succi, E. Kaxiras, Effects of Knudsen diffusivity on the effective reactivity of nanoporous catalyst media, *J. Comput. Sci.* 17 (2016) 377–383.
- [77] E.W. Larsen, A. Kumar, J.E. Morel, Properties of the implicitly time-differenced equations of thermal radiation transport, *J. Comput. Phys.* 238 (2013) 82–96.
- [78] R.G. McClarren, C.D. Hauck, Robust and accurate filtered spherical harmonics expansions for radiative transfer, *J. Comput. Phys.* 229 (16) (2010) 5597–5614.
- [79] A. Blaustein, F. Filbet, A structure and asymptotic preserving scheme for the Vlasov-Poisson-Fokker-Planck model, *J. Comput. Phys.* 498 (2024) 112693.
- [80] S. Von Alfthan, D. Pokhotelov, Y. Kempf, S. Hoilijoki, I. Honkonen, A. Sandroos, M. Palmroth, Vlasiator: first global hybrid-Vlasov simulations of Earth's foreshock and magnetosheath, *J. Atmos. Solar-Terrestrial Phys.* 120 (2014) 24–35.
- [81] J. Hu, S. Jin, Uncertainty quantification for kinetic equations, *Uncertainty Quantification for Hyperbolic and Kinetic Equations* (2017) 193–229.
- [82] T. Xiao, M. Frank, A stochastic kinetic scheme for multi-scale flow transport with uncertainty quantification, *J. Comput. Phys.* 437 (2021) 110337.
- [83] W.S. Moses, S.H.K. Narayanan, L. Paehler, V. Churavy, M. Schanen, J. Hückelheim, J. Doerfert, P. Hovland, Scalable automatic differentiation of multiple parallel paradigms through compiler augmentation, in: *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 2022, pp. 1–18.