SIAM J. SCI. COMPUT.
Vol. 47, No. 2, pp. C451–C474

SEPARABLE PHYSICS-INFORMED NEURAL NETWORKS FOR SOLVING THE BGK MODEL OF THE BOLTZMANN EQUATION*

JAEMIN OH†, SEUNG YEON CHO‡, SEOK-BAE YUN§, EUNBYUNG PARK¶, AND YOUNGJOON HONG#

Abstract. In this study, we present a new approach based on separable physics-informed neural networks (SPINNs), specifically designed to efficiently solve the Bhatnagar–Gross–Krook (BGK) model. While the mesh-free nature of physics-informed neural networks (PINNs) offers significant advantages for handling high-dimensional partial differential equations, applying quadrature rules for accurate integral evaluation in the BGK operator can compromise these benefits and increase computational costs. To address this issue, we leverage the canonical polyadic decomposition structure of SPINNs and the linear nature of moment calculation to significantly reduce the computational expense associated with quadrature rules. However, the multiscale nature of the particle density function poses challenges for precisely approximating macroscopic moments using neural networks. To overcome this, we introduce SPINN-BGK, a specialized variant of SPINNs that fuses SPINNs with Gaussian functions and utilizes a relative loss approach. This modification enables SPINNs to decay as rapidly as Maxwellian distributions, enhancing the accuracy of macroscopic moment approximations. The relative loss design ensures that both large- and small-scale features are effectively captured by the SPINNs. The effectiveness of our approach is validated through six numerical experiments, including a complex three-dimensional Riemann problem. These experiments demonstrate the potential of our method to efficiently and accurately tackle intricate challenges in computational physics, offering significant improvements in computational efficiency and solution accuracy.

Key words. separable neural network, BGK model, Maxwellian splitting, canonical polyadic decomposition, micro-macro decomposition

MSC codes. 65M99, 65R20, 68T07

DOI. 10.1137/24M1668809

1. Introduction.

The Boltzmann equation fundamentally characterizes the temporal evolution of particle density functions, predicated on the binary collision model. This equation diverges from traditional fluid dynamics equations by its capacity to encapsulate dynamics that extend beyond the continuum regime, offering a broader applicative scope [53, 34, 40, 24]. However, the practical utility of the Boltzmann equation is significantly constrained by the computational intensity inherent in its high-

*Submitted to the journal's Machine Learning Methods for Scientific Computing section June 12, 2024; accepted for publication (in revised form) January 7, 2025; published electronically April 4, 2025.

https://doi.org/10.1137/24M1668809

Funding: The work of the fifth author was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)] and by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT) (RS-2023-00219980). The second author was supported by an NRF grant funded by the Korea government (MSIT; RS-2022-00166144). The work of the third author was supported by the National Research Foundation of Korea (NRF) grant funded bythe Korean goverment (MSIT) (RS-2023-NR076676).

†Department of Electrical and Computer Engineering Texas A&M University, USA (jaemin_oh@tamu.edu).

‡Department of Mathematics, Gyeongsang National University, Jinju, Republic of Korea (chosy89@gnu.ac.kr).

§Department of Mathematics, Sungkyunkwan University, Suwon, Republic of Korea (sbyun01@skku.edu).

¶Department of Artificial Intelligence, Yonsei University, Republic of Korea (epark@yonsei.ac.kr).

#Department of Mathematical Sciences and Research Institute of Mathematics, Seoul National University, Republic of Korea (hongyj@snu.ac.kr).

dimensional collision operator. This complexity imposes substantial computational demands, thereby limiting its application across a diverse spectrum of scientific fields [41, 19, 17, 50].

Much effort has been directed toward the development of numerical methods for simulating kinetic equations. Among these, the direct simulation Monte Carlo (DSMC) method [4], employing a stochastic framework for the direct resolution of the Boltzmann equation, is distinguished by its computational efficiency. Nevertheless, the accuracy of DSMC is somewhat undermined by the inherent presence of statistical noise and pronounced oscillations. In contrast, deterministic methodologies, exemplified by the Fourier spectral method [45, 46, 43, 36] and the discrete velocity model [41], demonstrate superior accuracy, albeit with a concomitant increase in computational load. A notable advancement in this domain is documented in [15, 16], where the Bhatnagar–Gross–Krook (BGK) model of the Boltzmann equation and the full Boltzmann equation were efficiently resolved within a three-dimensional (3D) spatial domain, albeit necessitating extensive parallelization of computational resources. This highlights the complexity inherent in the numerical simulation of kinetic equations, especially in devising feasible numerical schemes that simultaneously reduce computational costs. Consequently, the quest for an efficient and practical numerical method emerges as a formidable challenge, requiring rigorous theoretical development and strategic application approaches.

Recent developments have led to the application of neural networks in reducing computational complexities associated with kinetic equation simulations. These efforts are broadly divided into two categories. The first is a data-driven approach [48, 1, 54], akin to reduced-order models [52]. In this method, neural networks, trained with high-fidelity numerical solutions (data), are used to replace the most computationally demanding elements of the simulation. For example, Han et al. [23] utilized neural networks to tackle the moment equation by learning the moment closure relation. Similarly, Miller et al. [42] introduced a pretrained surrogate neural network for the entire Boltzmann collision operator, aiming to reduce computational expenses. However, these methods require a substantial number of input-output pairs for an accurate approximation of the true relationship, posing challenges in situations where data collection is difficult or expensive.

The second approach to reducing computational complexity is the data-free method, with physics-informed neural networks (PINNs) [49, 39, 33, 44, 57] being a prominent example. PINNs model the solution of partial differential equations (PDEs) using neural networks, integrating PDE residuals, initial conditions, and boundary conditions as penalty terms. The inherent advantage of PINNs, not requiring structured grids, positions them as a promising tool for efficiently solving high-dimensional PDEs. A significant advancement in this domain is demonstrated in [27, 26], where PDEs of extremely high dimensions (up to $10^5 D$) were solved by introducing stochastic elements in the dimensions for gradient descent, extending beyond collocation points. However, this method did not encompass integro-differential equations (IDEs), limiting its applicability to the BGK model. A notable challenge with IDEs is that the use of quadrature rules for integral evaluation partially compromises the grid-free nature of PINNs, involving a substantially higher number of network forward passes.

In this study, we introduce SPINN-BGK, a PINN-based method designed to effectively solve the BGK model. To address the previously outlined challenges, we have adopted the architecture of separable PINNs (SPINNs) [10], which are distinguished by their canonical polyadic decomposition structure. This methodology, combined with an integration strategy that reorders the conventional order of integrals and summation, significantly reduces both memory and computational overhead.

Consequently, applying SPINNs with this integration technique has enabled the effective resolution of the BGK model in a 3D spatial domain on a single GPU.

However, computational complexity is not the sole challenge. PINNs have shown limitations in accurately approximating macroscopic quantities such as density, velocity, and temperature. This issue arises partly due to the characteristics of neural networks, which, without specific modifications, fail to replicate the rapid decay in the velocity domain exhibited by the particle density function. To address this, we first adopted a micro-macro decomposition, representing the solution as a sum of equilibrium and nonequilibrium parts, as described in [2, 30, 35]. This decomposition is inspired by the BGK operator's role in driving the system toward local thermodynamic equilibrium. We then integrated Gaussian functions of microscopic velocities into the neural network for the nonequilibrium part to ensure rapid decay in the velocity domain of the approximated particle density function. The equilibrium part already exhibited fast decay for the microscopic velocities. Additionally, we employed a relative L^2 loss function to ensure a balanced approximation of the solution across varying magnitudes. Collectively, these strategies have enabled our proposed neural networks to achieve rapid decay in the velocity domain, effectively approximate features of diverse magnitudes, and expedite training, thereby yielding accurate macroscopic moments.

It is customary to mention several related works. Lou, Meng, and Karniadakis [38] investigated both forward and inverse problems within the framework of PINNs, specifically targeting flows across a range of Knudsen numbers. However, their approach was primarily limited to the lattice Boltzmann method, employing a comparatively coarse lattice for the microscopic velocity space. Another notable attempt by Li et al. [35] involved the application of canonical polyadic decomposition to the discretized microscopic velocity space, followed by solving the full Boltzmann equation using a singular value decomposition–based reduced-order formulation, underpinned by an ansatz derived from the solution to the BGK model. Their focus, however, was primarily on the 2D Boltzmann equation. Our research diverges from the aforementioned works by concentrating solely on the BGK model without any discretization in the microscopic velocity space. Moreover, SPINN-BGK achieves enhanced computational efficiency by applying tensor decomposition to the spatio-temporal domain, enabling us to test SPINN-BGK on computationally expensive $(3 + 3 + 1)$ dimensional problems. Beyond these computational advancements, we also address the potential inaccuracies inherent in PINNs for generating macroscopic moments and propose a strategy to mitigate these limitations.

The structure of the remaining sections of this paper is organized as follows. Section 2 provides a concise overview of the BGK model and the framework of SPINNs. In section 3, we delve into the detailed exposition of our proposed methodologies. Section 4 is dedicated to presenting the results derived from both smooth and Riemann problem scenarios. Finally, section 5 offers a comprehensive summary and conclusion of our research findings.

2. Preliminaries.

2.1. The BGK model of the Boltzmann equation. Consider a particle characterized by a spatial dimension $d \in \{1, 2, 3\}$. At any given time t, the position of this particle is denoted as $\mathbf{x} \in \Omega$, where Ω is a subset of the Euclidean space $\mathbb{R}^d$. The particle is capable of moving with a velocity $\mathbf{v}$, which is an element of $\mathbb{R}^3$. We define $f(t, \mathbf{x}, \mathbf{v})$ as the particle density function in the phase space $\Omega \times \mathbb{R}^3$, representing the distribution of particles at time t.

The Boltzmann equation characterizes the time evolution of the particle density function f, based on the binary collision assumption. A significant aspect of this equation is its collision integral, which involves a fivefold computation, often leading to substantial computational demands [17]. To simplify this, the binary collision integral is replaced with a relaxation process that approximates the movement toward local thermodynamic equilibrium. This adaptation results in the BGK model [3], which is formulated as follows:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f = \frac{1}{\mathrm{Kn}}(M[f] - f). \tag{2.1}$$

The Knudsen number, denoted as Kn, serves as a dimensionless parameter, representing the ratio of the mean free path of a particle to the physical length scale of interest. In this context, we assume a fixed collision frequency. The Maxwellian distribution, denoted as $M[f]$, is defined as the particle density at local thermodynamic equilibrium. This distribution is characterized by

$$M[f](t, \mathbf{x}, \mathbf{v}) = \frac{\rho(t, \mathbf{x})}{(2\pi T(t, \mathbf{x}))^{3/2}} e^{-\frac{|\mathbf{v} - \mathbf{u}(t, \mathbf{x})|^2}{2T(t, \mathbf{x})}},$$

where the macroscopic mass ρ, velocity $\mathbf{u}$, and temperature T are defined by

$$\begin{aligned}
\rho(t, \mathbf{x}) &= \int f(t, \mathbf{x}, \mathbf{v}) d\mathbf{v}, \\
\mathbf{u}(t, \mathbf{x}) &= \begin{pmatrix} u_x(t, \mathbf{x}) \\ u_y(t, \mathbf{x}) \\ u_z(t, \mathbf{x}) \end{pmatrix} = \frac{1}{\rho(t, \mathbf{x})} \int \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} f(t, \mathbf{x}, \mathbf{v}) d\mathbf{v}, \\
T(t, \mathbf{x}) &= \frac{1}{3} \left(\frac{1}{\rho(t, \mathbf{x})} \int |\mathbf{v}|^2 f(t, \mathbf{x}, \mathbf{v}) d\mathbf{v} - |\mathbf{u}(t, \mathbf{x})|^2 \right).
\end{aligned}$$

Note that all the integrals mentioned above involve triple integration over $\mathbb{R}^3$. As a result, although there are various numerical methods available for solving the BGK model [41, 2, 47, 11, 18, 55, 6, 20], they typically face high computational costs. This challenge arises mainly from the model's high dimensionality and the necessary integrals for calculating the macroscopic moments.

2.2. Physics-informed neural networks. Consider $\phi : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ as a feed-forward neural network (FNN) comprising L layers, with d_l denoting the number of neurons in the lth layer. We define $d_0 = d_{\text{in}}$ and $d_L = d_{\text{out}}$. The affine transformation A_l for the lth layer is characterized by a weight matrix $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$ and a bias vector $b_l \in \mathbb{R}^{d_l}$. Let $\theta := \{W_l, b_l : 1 \le l \le L\}$ represent the set of network parameters. With an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ applied elementwise, the FNN, parameterized by θ, is defined as

$$\phi(\cdot) = A_L \circ \sigma \circ A_{L-1} \circ \sigma \circ \cdots \circ A_1(\cdot).$$

In this context, ϕ is often denoted as ϕ_θ to emphasize its dependency on the network parameters θ.

A primary objective of PINNs is to approximate the solution of a PDE using a neural network. This task essentially translates to identifying an optimal set of network parameters, denoted as θ, such that the neural network ϕ_θ accurately satisfies the stipulations of the given PDE. To elucidate the conventional methodology for determining these network parameters within the PINN framework, we consider the BGK model in the following specific form:

$$\mathcal{L}_{\mathrm{BGK}}[f](t,\mathbf{x},\mathbf{v}) = 0 \quad \forall (t,\mathbf{x},\mathbf{v}) \in (0,\infty)\times\Omega\times\mathbb{R}^3, \tag{2.2}$$

$$\mathcal{B}[f](t,\mathbf{x},\mathbf{v}) = 0 \quad \forall (t,\mathbf{x},\mathbf{v}) \in (0,\infty)\times\partial\Omega\times\mathbb{R}^3, \tag{2.3}$$

where $\mathcal{L}_{\mathrm{BGK}}[\cdot]$ is the integro-differential operator of the BGK model and $\mathcal{B}[\cdot]$ may encompass Dirichlet, Neumann, or periodic boundary conditions depending on the specific requirements of the problem at hand. Note that this setup is not limited to the BGK model. Given an activation function σ, a specified boundary condition $\mathcal{B}$, and an initial condition f_0, the objective is to determine a set of network parameters θ. For any positive value of p, we define three components of the L^p loss functions

$$\begin{aligned}
\mathcal{L}_r(\theta) &:= \iiint |\mathcal{L}_{\mathrm{BGK}}[f_\theta](t,\mathbf{x},\mathbf{v})|^p dt d\mathbf{x} d\mathbf{v},\\
\mathcal{L}_{\mathrm{bc}}(\theta) &:= \iiint |\mathcal{B}[f_\theta](t,\mathbf{x},\mathbf{v})|^p dt d\mathbf{x} d\mathbf{v},\\
\mathcal{L}_{\mathrm{ic}}(\theta) &:= \iint |f_\theta(0,\mathbf{x},\mathbf{v}) - f_0(\mathbf{x},\mathbf{v})|^p d\mathbf{x} d\mathbf{v},
\end{aligned} \tag{2.4}$$

corresponding to the PDE, boundary condition, and initial condition, respectively. Subsequently, we construct a comprehensive PINN loss function

$$\mathcal{L}(\theta) = \lambda_r \mathcal{L}_r(\theta) + \lambda_{bc}\mathcal{L}_{bc}(\theta) + \lambda_{ic}\mathcal{L}_{ic}(\theta). \tag{2.5}$$

In this formulation, λ_r, λ_{bc}, and λ_{ic} are positive real numbers, selected based on the empirical evaluation. The integrals present in (2.5) are computed using appropriate numerical integration methods. The optimal network parameters, denoted as θ^*, are determined by solving

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta),$$

employing a suitable optimization algorithm, such as Adam [32].

3. Methodology. In this section, we address various challenges encountered in the application of PINNs to solve the BGK model. We also propose a series of methodologies designed to effectively mitigate these difficulties.

3.1. Methodology for reduction of computational cost. Traditional mesh-based numerical solvers for the BGK model often face the challenge of dimensionality, particularly due to the incorporation of the microscopic velocity space. In contrast, PINNs are not constrained by the need for structured meshes, which initially suggests an advantage in this context. However, the requirement to compute integrals for macroscopic moments somewhat offsets this perceived benefit. Furthermore, when dealing with solutions that have complex profiles, PINNs typically demand a large number of collocation points. Therefore, efficiently solving the BGK model using PINNs demands a reduction in the number of network forward passes and a more rapid evaluation of these integrals.

3.1.1. Separable physics-informed neural networks. The computation of macroscopic moments $\rho, \mathbf{u}, T$ in the context of PINNs significantly increases the number of network forward passes, especially when solving PDEs of similar dimensionality that do not involve such integrals. Consider m_{ijk} as the ijkth moment, defined by the following expression:

$$m_{ijk}(t,\mathbf{x}) := \int_{\mathbb{R}^3} f(t,\mathbf{x},\mathbf{v}) v_x^i v_y^j v_z^k d\mathbf{v}. \tag{3.1}$$

Employing a trapezoidal rule with N points allocated along each axis results in a total of N^3 points. Consequently, for each pair of $(t, \mathbf{x})$, it becomes necessary to perform $O(N^3)$ evaluations of the function f to accurately compute $m_{ijk}(t, \mathbf{x})$.

In this study, we utilize the SPINN framework [10] to effectively reduce the number of network forward passes required. This approach is grounded in the principles of canonical polyadic decomposition, a technique known for its efficacy in managing high-dimensional problems [25]; for other low-rank methods, see, e.g., [21, 22]. For instance, canonical polyadic decomposition has been applied in classical numerical methods for solving the 3D BGK model [5]. Consider ϕ as a scalar-valued function on $\mathbb{R}^d$. Let G denote a rectilinear grid in $\mathbb{R}^d$, characterized by N_i points along the ith axis. It can be demonstrated that, given a sufficiently large value of R, there exist functions $\phi_1, \phi_2, \ldots, \phi_d : \mathbb{R} \to \mathbb{R}^R$ such that for every point $(x_1, x_2, \ldots, x_d)$ in the grid G, the following relationship holds:

$$\phi(x_1, x_2, \ldots, x_d) \approx \sum_{r=1}^{R} \prod_{i=1}^{d} \phi_{i,r}(x_i).$$

In this expression, $\phi_{i,r}(x_i)$ denotes the rth element of the vector $\phi_i(x_i)$. Motivated by this, SPINNs assign d FNNs $\{\phi(\cdot; \theta_i) : \mathbb{R} \to \mathbb{R}^R\}_{i=1}^{d}$ to the d axes of the domain, and then combine them in the same way to approximate ϕ at $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ by

$$\phi(\mathbf{x}) \approx \phi_\theta^{\mathrm{SPINN}}(\mathbf{x}) = \sum_{r=1}^{R} \prod_{i=1}^{d} \phi_r(x_i; \theta_i).$$

It is important to highlight that for a rectilinear grid configured as $(N_1, N_2, \ldots, N_d)$, SPINNs significantly reduce the number of network forward passes from $O(\prod_{i=1}^{d} N_i)$ to $O(\sum_{i=1}^{d} N_i)$. This reduction offers a substantial computational advantage when compared to traditional FNNs. Furthermore, the computation of partial derivatives can be efficiently executed using Jacobian-vector products, also known as forward-mode automatic differentiation.

Neural networks, when equipped with a sufficient number of hidden units, are recognized for their ability to approximate any continuous function. However, the extension of this universal approximation capability to SPINNs is not immediately obvious. To address this, and to enhance the rigor of our paper, we provide the universal approximation theorem for SPINNs (Theorem 3.3).

THEOREM 3.1. *Let X and Y be compact subsets of $\mathbb{R}^d$, and $\phi \in L^2(X \times Y)$. For $\epsilon > 0$, there exists a SPINN ϕ_θ such that*

$$\|\phi - \phi_\theta\|_{L^2(X \times Y)} < \epsilon.$$

Proof. See Supplementary Materials A of [10]. ∎

Remark 3.2. We note that the universal approximation theorem [13] works for one hidden layer network. Its extension to an arbitrary, bounded depth network is easy, as outlined in [31]: express the lth layer as $B_l \circ \sigma \circ A_l$, where A_l and B_l are affine transformations and σ is an activation function; let $l > 1$th layers approximate the identity function.

Theorem 3.1 requires modifications to be suitably applied to the particle density function f since it is defined over the microscopic velocity domain $\mathbb{R}^3$, rather than

on a compact subset of $\mathbb{R}^3$. In light of these considerations, we present a tailored approximation theorem for f as a function defined on the domain $\Omega \times \mathbb{R}^3$, ensuring its compatibility with the unique characteristics of the BGK model.

THEOREM 3.3. *For $d \in \{1,2,3\}$, let $\Omega \subset \mathbb{R}^d$ be a compact set and $f : \Omega \times \mathbb{R}^3 \to \mathbb{R}$ be a particle density function at a fixed time, and $\phi(\mathbf{v}) = 1 + |\mathbf{v}|^2$. We assume that $f \in L^2(\Omega \times \mathbb{R}^3)$, and $\int_{\mathbb{R}^3} f\phi d\mathbf{v} \in L^2(\Omega)$. Then, for $\epsilon > 0$, there exists a SPINN f_θ such that*

$$\|f - f_\theta\|_{L^2(\Omega\times\mathbb{R}^3)} < \epsilon \qquad \textit{and} \qquad \left\| \int_{\mathbb{R}^3} (f - f_\theta)\phi d\mathbf{v} \right\|_{L^2(\Omega)} < \epsilon.$$

Proof. See Appendix A. ∎

Remark 3.4. Note that our convergence proof is carried out in L^2 space whereas the kinetic distribution functions lie in L^1 space. The main reason is that the underlying Theorem 3.1 is established in L^2 space using the orthogonality in L^2. We, however, mention that such L^2 based convergence analysis provides a relevant perspective for the convergence of our simulation since we are working on a truncated domain so that we have by Hölder's inequality

$$\|f - f_\theta\|_{L^1} \le C_\gamma \|f - f_\theta\|_{L^2},$$

where γ is the truncation parameter.

3.1.2. Fast evaluation of the macroscopic moments. Consider N_v as the number of points on each axis of the microscopic velocity space, utilized for evaluating the macroscopic moments as outlined in (3.1). As previously discussed, SPINNs necessitate $O(3N_v)$ network forward passes for the computation of N_v^3 function values. However, directly evaluating the integrals from the output of the SPINN is not computationally efficient, as it requires $O(N_v^3)$ operations and memory cost. To optimize this process, we leverage the linear nature of moment calculation in (3.1) and the inherent separable structure of the SPINNs. Afterward, the macroscopic moments can be evaluated more efficiently, within $O(3RN_v)$ operations.

The integrals for the macroscopic moments m_{ijk} can be computed as follows:

$$\begin{aligned} \int f_\theta(t,\mathbf{x},\mathbf{v}) v_x^{i_x} v_y^{i_y} v_z^{i_z} d\mathbf{v} &= \int \sum_{r=1}^{R} f_r(t;\theta_t) \prod_{p\in\{x,y,z\}} f_r(p;\theta_p) f_r(v_p;\theta_{v_p}) v_p^{i_p} d\mathbf{v} \\ &= \sum_{r=1}^{R} f_r(t;\theta_t) \prod_{p\in\{x,y,z\}} f_r(p;\theta_p) \int f_r(v_p;\theta_{v_p}) v_p^{i_p} dv_p. \end{aligned} \tag{3.2}$$

Consequently, (3.2) significantly reduces both the computational cost and memory requirements to $O(3RN_v)$, a substantial improvement over the $O(N_v^3)$ required for standard computation. Table 1 shows the elapsed times (measured in milliseconds (m), microseconds (μ), and nanoseconds (n)) required to compute $(\rho, \mathbf{u}, T)$ using SPINN with two different integration strategies. The timings are presented as functions of the rank (R) and the number of quadrature points (N) per velocity axis. We chose N in the form $2^n + 1$ to ensure the number of intervals is 2^n, although this choice was not tied to a specific reason. For hardware specifications, please refer to section 4. We considered only one spatial dimension ($d = 1$) and discretized the space-time domain by $(N_t, N_x) = (12, 12)$, respectively. Here, SPINN has five FNNs

TABLE 1

Computation times (mean ± std) for calculating $(\rho, \mathbf{u}, T)$ for SPINN. "Separate" corresponds to the integration strategy outlined in (3.2). *"Standard" corresponds to the strategy without* (3.2). *Here, $(n, \mu, m) = (10^{-9}, 10^{-6}, 10^{-3})$ seconds, respectively. R is the rank of the SPINN. N is the number of quadrature points per velocity axis. We considered the $d = 1$ case with $(N_t, N_x) = (12, 12)$. OOM means "out of memory." We only considered $R = 1$ for $N = 257$, as $R = 2$ produces OOM for standard integration. All measurements were conducted on single precision, via the* `%timeit` *command of iPython.*

(R, N)	(32,65)	(32,129)	(64,65)	(64,129)
Standard	$892\mu \pm 3.57\mu$	$5.57m \pm 3.53\mu$	$899\mu \pm 882n$	$5.78m \pm 7.27\mu$
Separate	$167\mu \pm 1.78\mu$	$167\mu \pm 336n$	$169\mu \pm 1.66\mu$	$169\mu \pm 107n$
(R, N)	(128,65)	(128,129)	(1,257)	(1024,1025)
Standard	$988\mu \pm 935n$	$6.02m \pm 4.42\mu$	$32.2m \pm 33.7\mu$	OOM
Separate	$166\mu \pm 1.39\mu$	$168\mu \pm 394n$	$150\mu \pm 6.69\mu$	$213\mu \pm 3.4\mu$

which have a width of 128, and a depth of 3 for each axis. As a baseline, single FNN-based methods such as vanilla PINN resulted in out-of-memory when $N = 65$ even though we employed only one hidden layer. For $N = 33$, it took $6.4m \pm 621n$ seconds, which is less efficient than SPINN. For both the "standard" and "separate" integration strategies for SPINN, there appears to be no clear relationship between R and computing times. This lack of correlation can be attributed to the following factors. In the standard strategy, a significant portion of the computing time is consumed by the integration phase, with its duration depending solely on N. Conversely, in the separate integration strategy, the computational cost of the network's forward pass dominates over integration, once again dependent solely on N. However, the computing times for standard integration increase approximately 6 to 8 times as N doubles. For $N = 257$, we observed the out-of-memory issue for standard integration immediately after increasing $R = 1$ to $R = 2$. In contrast, separate integration shows excellent scalability, as the computing times remain similar across all configurations. Although SPINN achieves notable efficiency gains through its network architectural design compared to the basic form like FNNs, the memory constraints and limited scalability of standard integration become significantly more pronounced as the spatial dimension increases. Therefore, the efficiency of the separate integration strategy enables fast evaluation of integrals on fine quadrature points, provided that the rank R is not too large. In this study, we selected $R \in \{128, 256\}$ and $N_v = 257$ for our computations.

3.2. Strategies for enhancing accuracy. Neural networks are designed and trained to precisely approximate the particle density f. However, there are scenarios where, despite the particle density f being accurately approximated to a certain level of mean squared error, the macroscopic moments $(\rho, \mathbf{u}, T)$ may still be inaccurately represented. This issue often stems from the accumulation of numerical errors during the integration process, as the macroscopic moments are derived by integrating the particle density. To mitigate this challenge, we introduce a neural network architecture tailored for kinetic equations, complemented by a relative loss function, aimed at enhancing the accuracy of these macroscopic moments.

3.2.1. Maxwellian splitting. The BGK operator characterizes the evolution of an initial particle density function toward a state of thermodynamic equilibrium. Consequently, it is natural to conceptualize the particle density function as a composite of two components: the local equilibrium part and the residual part. This

decomposition approach aligns with the methodologies proposed in the works of Lou et al. [38] and Li et al. [35], where the motivation might be brought from [2, 30]. Inspired by the decomposition approach, we propose the employment of two distinct SPINNs: f_θ^{eq} and f_θ^{neq}. The first network, f_θ^{eq}, is designed to accept inputs $t, \mathbf{x}$ and outputs a tuple $(\rho_\theta, \mathbf{u}_\theta, T_\theta) \in \mathbb{R} \times \mathbb{R}^3 \times \mathbb{R}$. The second network, f_θ^{neq}, processes the full set of variables $t, \mathbf{x}, \mathbf{v}$, yielding a real-valued output. These networks are combined as follows to approximate the solution to the BGK model:

$$f_\theta(t, \mathbf{x}, \mathbf{v}) = M[\rho_\theta(t, \mathbf{x}), \mathbf{u}_\theta(t, \mathbf{x}), T_\theta(t, \mathbf{x})](\mathbf{v}) + \alpha f_\theta^{\text{neq}}(t, \mathbf{x}, \mathbf{v}). \tag{3.3}$$

Here, $M[\rho, \mathbf{u}, T]$ denotes the Maxwellian distribution parameterized by the macroscopic moments $\rho, \mathbf{u}, T$, albeit used somewhat loosely in this notation. The coefficient α is a parameter dependent on the specific problem. While one might consider setting α equal to the Knudsen number (Kn) following the Chapman–Enskog expansion [8], in this study, we have fixed α at 1 since it performed well throughout all the experiments.

To compute the macroscopic moments as specified in (3.1), we perform separate integrations for the equilibrium term $M[\rho_\theta, \mathbf{u}_\theta, T_\theta]$ and the nonequilibrium term f_θ^{neq}, taking advantage of the linear nature of the moment calculation. Importantly, for the equilibrium term, the zeroth, first, and second moments are directly given by ρ_θ, $\rho_\theta \mathbf{u}_\theta$, and $\rho_\theta(3T_\theta + |\mathbf{u}_\theta|^2)$, respectively, eliminating the need for numerical integration. The nonequilibrium term, in contrast, is efficiently integrated using the method described in (3.2). Although this decomposition approach does not have formal theoretical justification, empirical evidence indicates that splitting the density function into these two terms significantly improves both training speed and accuracy.

3.2.2. Integration of decay property into neural networks. Accurate computation of macroscopic moments necessitates a more careful consideration of high relative velocities. Given that the density function typically exhibits rapid decay as the magnitude of $\mathbf{v}$ increases, contributions from high relative velocities to the macroscopic moments are generally negligible. However, if the neural network employed does not mirror this rapid decay, its output at high relative velocities might contribute excessively, leading to inaccuracies in the calculated macroscopic moments. Therefore, ensuring that the neural network appropriately reflects the decay characteristics of the particle density function is crucial for the precision of the macroscopic moment calculations.

For an illustration, we consider the standard Gaussian function g on the truncated domain $\mathcal{D} := (-10, 10) \subset \mathbb{R}$:

$$g : v \in \mathcal{D} \mapsto \frac{1}{\sqrt{2\pi}} e^{-\frac{v^2}{2}}.$$

When performing numerical integration of its moments, the function values near the boundary of the domain $\partial\mathcal{D}$ (specifically around ± 10) are so small that the contribution of v near $\partial\mathcal{D}$ to the moments should be negligible. However, during the training phase, a neural network g_θ that approximates g may not exhibit the same rapid decay as g. To investigate this issue, we employ an FNN, denoted as g_θ, to approximate the Gaussian function g. This approximation is achieved by minimizing a loss function $\mathcal{L}_{\text{MSE}}$, which is defined as follows:

$$\mathcal{L}_{\text{MSE}} : \theta \mapsto \frac{1}{20} \int_{-10}^{10} |g_\theta(v) - g(v)|^2 dv. \tag{3.4}$$

TABLE 2
Additional details for toy experiments in subsections 3.2.2 *and* 3.2.3.

Layers	(1, 256, 1)
activation function	tanh
training points	16, uniform distribution
optimizer	Adam
number of iterations	1M
initial learning rate	10^{-4}
learning rate schedule	cosine decay to 0 [37]
initial (μ, τ)	$(2, 1/\sqrt{2})$

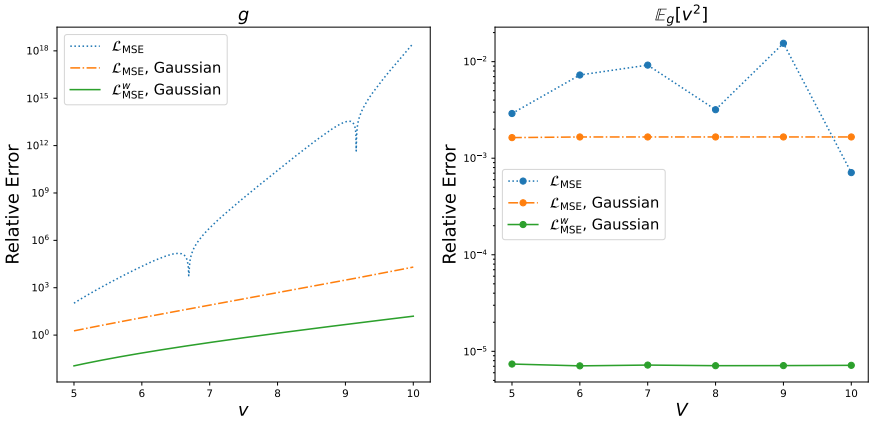


FIG. 1. *Left:* $v \in (5, 10) \mapsto |g_\theta(v) - g(v)|/|g(v)|$. *Right:* $V \mapsto \int_{-V}^{V} v^2 |g_\theta(v) - g(v)| dv / \int_{-V}^{V} v^2 g(v) dv$. *Uniform meshes of* 1024 *points are used to evaluate pointwise errors and integrals. Dotted line: results for the* $\mathcal{L}_{\text{MSE}}$ *loss* (3.4) *approximation with a neural network (baseline). Dot-dashed and solid lines: results for* $\mathcal{L}_{\text{MSE}}$ *loss and relative loss* $\mathcal{L}^w_{\text{MSE}}$ (3.6)*, respectively, with a neural network multiplied by* g_p (3.5).

The loss function was evaluated with the Monte Carlo integration. Additional details can be found in Table 2.

Figure 1 shows the outcomes of three distinct numerical experiments designed to assess various aspects of our methodology. These experiments include an investigation into the effects of the approximated tail values of g_θ on the second moment, an examination of the impact of incorporating a Gaussian function into the neural network (referred to as Gaussian in the figure legends), which is aimed at facilitating the decay speed, and an effect of employing a relative loss function $\mathcal{L}^w_{\text{MSE}}$ (to be defined in (3.6) below), that imposes large weights on small-scale features. The left panel of Figure 1 presents relative pointwise error $|g_\theta(v) - g(v)|/|g(v)|$ on a part of the domain $(5, 10)$. The right panel presents errors for the second moment evaluated on the truncated interval $(-V, V)$. The dotted lines correspond to the results of minimizing $\mathcal{L}_{\text{MSE}}$ function, without any modifications in network architecture. As observed, the relative pointwise error exhibits an exponential increase with the growth of v. This pattern suggests that the optimized neural network, g_θ, does not decay as rapidly as g. A contributing factor to this behavior is the nature of the $\mathcal{L}_{\text{MSE}}$ loss function, which tends to relatively overlook the smaller-scale values. Consequently, the errors observed in the second moment are not uniform, implying that g_θ possesses thicker tails in comparison to g. Furthermore, despite being a regression setting, the

magnitude of the errors is notably significant, particularly above 10^{-2} when $V = 9$. This level of error could lead to substantial inaccuracies in training PINNs, given that direct access to the exact solution of the PDE is typically unavailable.

The equilibrium term $M[\rho_\theta, \mathbf{u}_\theta, T_\theta]$ in (3.3) decays as quickly as g, yet the nonequilibrium term f_θ^{neq} does not. To make f_θ^{neq} decay exponentially fast to $|\mathbf{v}|$, we modified its form slightly into

$$f_\theta^{\text{neq}}(t, \mathbf{x}, \mathbf{v}) = \sum_{r=1}^{R} f_r(t; \theta_t) \prod_{p \in \{x,y,z\}} f_r(p; \theta_p) \left[f_r(v_p; \theta_p) g_p(v_p; \tau, \mu_p) \right],$$

where

$$g_p : v_p \mapsto e^{-\tau^2 (v_p - \mu_p)^2 / 2}. \tag{3.5}$$

We set $\tau, \mu := (\mu_x, \mu_y, \mu_z)$ as network parameters to avoid hand-tuning procedures. We approximated the function g by a neural network multiplied by g_p. Initial values for parameters τ and μ were set to $1/\sqrt{2}$ and 2, respectively. Note that the standard Gaussian function corresponds to $\tau = 1$ and $\mu = 0$. The dot-dashed lines of Figure 1 present the result. From the left panel, we can observe that values for large $|v|$ are well captured, compared to the dotted line (baseline). As we can see from the right panel, even though the approximated second moment was better than the baseline overall, they were in the same order of magnitude.

3.2.3. Relative loss. As previously discussed, the standard L^2 loss function $\mathcal{L}_{\text{MSE}}$ tends to prioritize learning large-scale features. However, in our context, small-scale features are equally crucial, particularly since our focus is on accurately approximating the macroscopic moments $\rho, \mathbf{u}, T$. Therefore, to appropriately emphasize different scales, we propose the adoption of a weighted L^2 loss function, defined as

$$\mathcal{L}_{\text{MSE}}^w : \theta \mapsto \int (w \mathcal{N}[f_\theta])^2 dt d\mathbf{x} d\mathbf{v}, \tag{3.6}$$

where $\mathcal{N}$ represents either the PDE residual operator, the boundary condition, or the initial condition. The weighting function w is defined as $1/(|f_\theta| + \epsilon)$. During backpropagation, w is treated as a constant with respect to θ by using `jax.lax.stop_gradient`. The term ϵ is included to ensure numerical stability, with a practical value of 10^{-3} typically employed. For the initial condition loss, $w[f_\theta]$ can be substituted with $w[f_0]$. It is important to note that $\mathcal{L}_{\text{MSE}}^w$ assigns larger weights when the magnitude of f_θ is small, and smaller weights when it is large, thus enabling more balanced learning of different scale features.

We conducted a demonstrative example, as illustrated in Figure 1, where we approximated the function g using a neural network in conjunction with (3.5), employing the loss function defined in (3.6). The outcomes of this approximation are represented by the solid lines in Figure 1. The results indicate a significant improvement over the dot-dashed line, which represents the approximation using the standard L^2 loss function $\mathcal{L}_{\text{MSE}}$ with uniform weights, as per (3.4). Notably, the relative error for the second moment was reduced by two orders of magnitude when using our proposed method.

4. Numerical results. In this section, we present a series of numerical experiments to demonstrate the efficacy of SPINN-BGK. The simulations cover a range of problems: one problem with smooth initial data and one Riemann problem for

TABLE 3
Additional details for section 4. We conducted all experiments in section 4 with this setting for consistency.

Width	128
depth	3
tensor rank R	128 (smooth), 256 (Riemann)
numerical integration	trapezoidal rule, 257 points
optimizer	Lion [9]
number of iterations	100K
initial learning rate	10^{-5}
learning rate schedule	cosine decay to 0 [37]
$(\lambda_r, \lambda_{\text{ic}}, \lambda_{\text{bc}})$ of (2.5)	$(1, 10^3, 1)$
initial (μ, τ) of (3.5)	$(\mathbf{0}, 1)$

each spatial dimension $d \in \{1, 2, 3\}$. Throughout all the experiments, the velocity dimension is fixed at 3. For 1D and 2D problems, we generated reference solutions employing a high-order conservative semi-Lagrangian scheme [11] on a personal computer equipped with an AMD Ryzen 9 5900X 12-core processor operating at 3.70 GHz and 64 GB of RAM, using MATLAB [28]. For 3D problems, we generated reference solutions on an HPC cluster equipped with an Intel Xeon Gold 6348 56-core processor operating at 2.60 GHz and 128 GB of RAM, using MATLAB. For further details on the reference solutions, please consult Table 11 (Appendix C).

Throughout these experiments, we consider three different Knudsen numbers, $\text{Kn} \in \{10^0, 10^{-1}, 10^{-2}\}$. The activation function for our MLP is set as $x \mapsto \sin(w_0 x)$, with network parameters initialized following the guidelines in [51]. Empirically, we chose $w_0 = 10$ for our experiments. For the training of SPINN-BGK, we utilized the Lion optimizer [9] implemented in the Optax library [14], which bases its updates on the sign of the gradient momentum rather than the gradient momentum itself. We observed that the Lion optimizer tends to achieve a lower training loss compared to the Adam optimizer [32].

To evaluate the accuracy of SPINN-BGK solutions, we employed the relative L^2 error metric, defined as $\|\phi_{\text{true}} - \phi_{\text{pred}}\|_2 / \|\phi_{\text{true}}\|_2$ for ρ and T, and $\|\phi_{\text{true}} - \phi_{\text{pred}}\|_2 / (\|\phi_{\text{true}}\|_2 + 1)$ for $\mathbf{u}$, to avoid division by zero. All experiments were conducted on a single NVIDIA RTX 4090 GPU, implemented using JAX [7]. Further experimental details are provided in Table 3. The computing times for the SPINN-BGK are provided in Appendix C.

4.1. 1D smooth problem. In this section, we conduct a test on the 1D smooth problem as outlined in section 6.1 of [35]. The temporal scope of our test is confined to the interval $(0, 0.1]$. We consider a spatial domain of $(-0.5, 0.5)$, imposing periodic boundary conditions. For the microscopic velocity space, the computational domain is set to $(-10, 10)^3$. The initial condition for this test is defined by a Maxwellian distribution characterized by specific macroscopic moments

$$\rho_0(x) = 1 + 0.5\sin(2\pi x), \quad \mathbf{u}_0(x) = \mathbf{0}, \quad T_0(x) = 1 + 0.5\sin(2\pi x + 0.2).$$

In this experiment, we employ the trapezoidal rule with 257 points for the numerical evaluation of macroscopic moments. The optimization process involves 100K gradient descent steps. For each descent step, we randomly sample collocation points from uniform distributions in the size of $(N_t, N_x, N_v^3) = (12, 16, 12^3)$.

The numerical results are depicted in Figure 2. The entire computation was completed in approximately 4 minutes. For comparison, the reference solution was

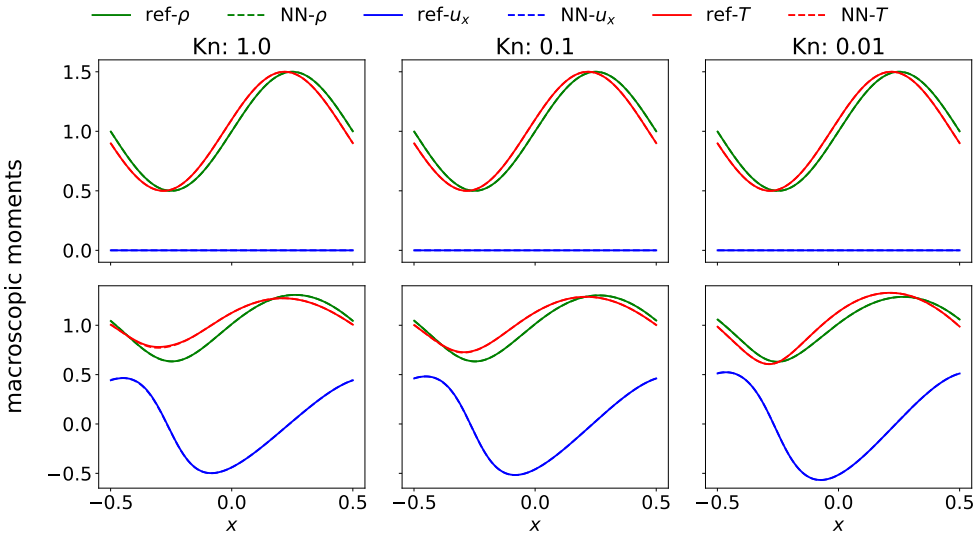


FIG. 2. *Numerical solutions to subsection* 4.1. *Macroscopic moments* ρ, u_x, T *are presented. The top row corresponds to* $t = 0$. *The bottom row corresponds to* $t = 0.1$. *Left column:* $\mathrm{Kn} = 1.0$. *Middle column:* $\mathrm{Kn} = 0.1$. *Right column:* $\mathrm{Kn} = 0.01$. *The solid lines are the reference solutions, and the dotted lines are the neural network predictions.*

TABLE 4

Relative L^2 *errors for the macroscopic moments* ρ, u_x, T *of the* 1*D smooth problem. For each parenthetical, the first component corresponds to* $t = 0$ *and the second to* $t = 0.1$.

$(t = 0, t = 0.1)$	$\mathrm{Kn} = 1$	$\mathrm{Kn} = 0.1$	$\mathrm{Kn} = 0.01$
ρ	(2.74e-4, 4.65e-4)	(1.81e-4, 3.99e-4)	(1.92e-4, 3.47e-4)
u_x	(4.19e-5, 1.36e-3)	(2.63e-5, 8.44e-4)	(3.66e-5, 3.53e-4)
T	(2.16e-4, 2.89e-3)	(1.49e-4, 1.58e-3)	(1.95e-4, 2.07e-4)

generated using a finer grid with $N_x = 1280$ and $N_v^3 = 25^3$. It is important to note that u_y and u_z are omitted in our analysis, as they do not exhibit significant dynamics along the y and z directions. The plot demonstrates that the predicted macroscopic moments align closely with the reference solutions for a range of Knudsen numbers, specifically $\mathrm{Kn} \in \{10^0, 10^{-1}, 10^{-2}\}$. In Table 4, we present the relative L^2 errors between the neural network predictions and the reference solutions. Notably, all errors are equal to or less than the order of $O(10^{-3})$, which signifies a strong agreement between SPINN-BGK predictions and the reference solutions. This result is indicative not only of a qualitative match but also of a quantitative alignment, underscoring the effectiveness of SPINN-BGK in accurately capturing the dynamics of the system.

4.2. 1D Riemann problem. In this section, we conduct a test on the 1D Sod tube problem, as detailed in section 6.2 of [35]. The computational settings for this test largely mirror those used in subsection 4.1, with the exceptions being the boundary condition, the number of collocation points, and the initial condition. For the spatial domain, we implement the free-flow boundary condition. The initial condition is represented by a local Maxwellian distribution with macroscopic moments $(\rho_0, \mathbf{u}_0, T_0)$. Specifically, the values on the interval $(-0.5, 0)$ are set to $(1, \mathbf{0}, 1)$, and on $[0, 0.5)$, they are $(0.125, \mathbf{0}, 0.8)$. Given the challenge of approximating jump functions with neural networks, we opt for a smoothed version of the macroscopic moments:

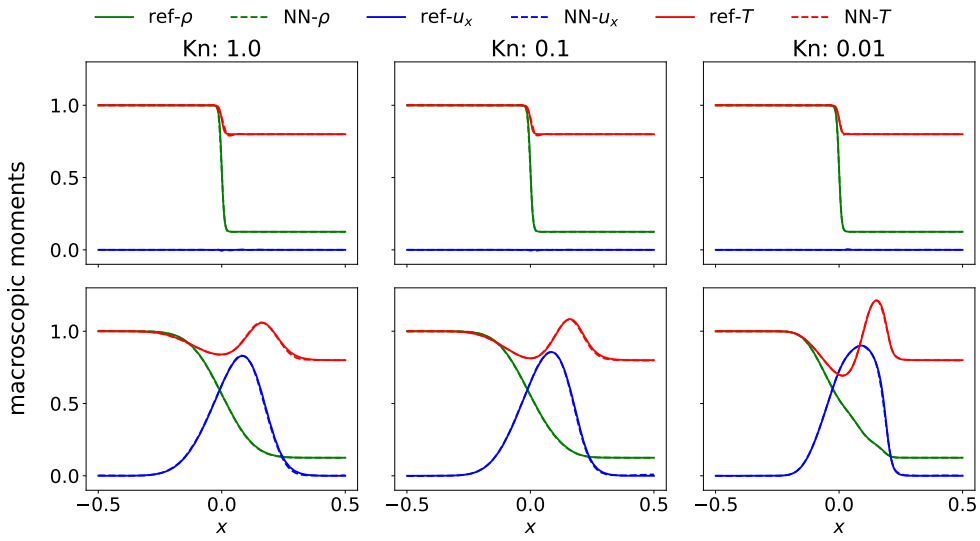


FIG. 3. *Predicted macroscopic moments ρ, u_x, T of the 1D Riemann problem. Unlike Figure 2, slight deviations are observable. This could be attributed to the increased computational complexity explained in Appendix B.*

TABLE 5
Relative L^2 errors for the macroscopic moments ρ, u_x, T of the 1D Riemann problem.

$(t=0, t=0.1)$	Kn = 1	Kn = 0.1	Kn = 0.01
ρ	(6.72e-3, 2.96e-3)	(5.45e-3, 3.08e-3)	(2.76e-3, 1.77e-3)
u_x	(1.00e-3, 4.65e-3)	(9.50e-4, 4.10e-3)	(8.47e-4, 3.67e-3)
T	(4.58e-3, 4.92e-3)	(3.83e-3, 4.58e-3)	(1.96e-3, 3.02e-3)

$$\rho_0(x) = 1 - 0.875H(x), \quad \mathbf{u}_0(x) = \mathbf{0}, \quad T_0(x) = 1 - 0.2H(x).$$

Here, $H : x \mapsto (1 + \tanh(100x))/2$ serves as a smoothed approximation of the Heaviside function. In each iteration of the process, we randomly sample points in the configuration $(N_t, N_x, N_{v_x}, N_{v_y}, N_{v_z}) = (12, 32, 32, 12, 12)$.

In Figure 3, we present the macroscopic moments obtained through SPINN-BGK alongside the reference solution. The entire computation was completed in approximately 4 minutes and 10 s. The reference solution was generated using a grid with $N_x = 1280$ and $(N_{v_x}, N_{v_y}, N_{v_z}) = (97, 25, 25)$. The figure demonstrates that the neural network solutions closely align with the reference solutions. Relative L^2 errors for this experiment are detailed in Table 5. Notably, all errors are equal to or less than the order of $O(10^{-3})$, which quantitatively demonstrates a strong alignment between SPINN-BGK solutions and reference solutions. This level of accuracy underscores the effectiveness of our method in capturing the essential dynamics of the problem.

4.3. 2D smooth problem. In this section, we conduct a test on the 2D smooth problem as outlined in section 6.3 of [35]. The experimental setup largely follows that of subsection 4.2, with a few notable adjustments. The spatial domain is set as $(-0.5, 0.5)^2$, over which we impose periodic boundary conditions. The initial condition is defined by a Maxwellian distribution with specific macroscopic moments

TABLE 6
Relative L^2 errors for the macroscopic moments ρ, u_x, u_y, T of the 2D smooth problem.

$(t=0, t=0.1)$	$\mathrm{Kn}=1$	$\mathrm{Kn}=0.1$	$\mathrm{Kn}=0.01$
ρ	(1.82e-4, 1.29e-4)	(1.71e-4, 1.16e-4)	(1.87e-4, 1.53e-4)
u_x	(1.21e-5, 1.17e-4)	(1.62e-5, 1.08e-4)	(1.40e-5, 1.12e-4)
u_y	(1.37e-5, 1.07e-4)	(9.46e-6, 8.75e-5)	(1.06e-5, 1.23e-4)
T	(1.50e-5, 4.14e-5)	(1.85e-5, 3.50e-5)	(1.61e-5, 8.51e-5)

TABLE 7
Relative L^2 errors for the macroscopic moments ρ, u_x, u_y, T of the 2D Riemann problem.

$(t=0, t=0.1)$	$\mathrm{Kn}=1$	$\mathrm{Kn}=0.1$	$\mathrm{Kn}=0.01$
ρ	(4.51e-2, 2.56e-2)	(4.41e-2, 2.17e-2)	(3.38e-2, 2.27e-2)
u_x	(4.31e-3, 3.48e-2)	(3.92e-3, 2.70e-2)	(4.57e-3, 2.64e-2)
u_y	(4.38e-3, 3.41e-2)	(3.86e-3, 2.79e-2)	(3.54e-3, 2.76e-2)
T	(1.02e-2, 1.82e-2)	(1.03e-2, 1.47e-2)	(8.39e-3, 2.29e-2)

$$\rho_0(x,y) = 1 + 0.5\sin(2\pi x)\sin(2\pi y), \quad \mathbf{u}_0(x,y) = \mathbf{0}, \quad T_0(x,y) = 1.$$

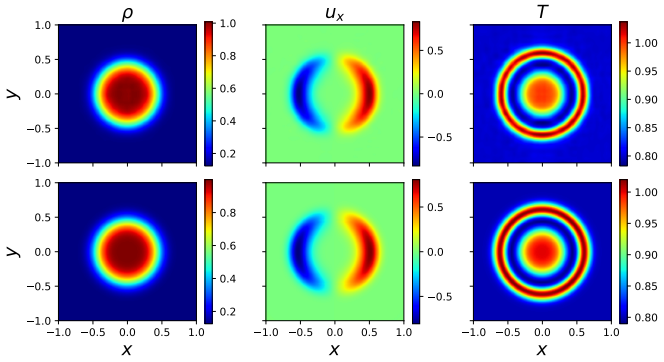
For this test, we utilize 12 collocation points per axis. The relative L^2 errors obtained from our experiment are detailed in Table 6. Remarkably, the entire computation, including both training and inference phases, is completed in approximately 5 minutes and 10 s. The reference solution for comparison was generated using a grid with $N_x^2 = 160^2$ and $N_v^3 = 25^3$. Our results demonstrate that we can achieve solutions that are quantitatively well-aligned with the reference solutions in the 2D case.

4.4. 2D Riemann problem. In this section, we conduct a test on the 2D Riemann problem as presented in [11]. The spatial domain for this test is defined as $(-1,1)^2$, where we apply a free-flow boundary condition. Within this domain, we consider a circle S defined by $x^2+y^2 = 0.2$. The initial condition is a local Maxwellian distribution with macroscopic moments $(\rho_0, \mathbf{u}_0, T_0)$. Inside the sphere, the values are set to $(1, \mathbf{0}, 1)$, while outside the sphere, they are $(0.125, \mathbf{0}, 0.8)$. To address the discontinuity in the initial data, similar to the approach in the 1D Riemann problem, we employ a relaxed version using the Heaviside function $H(r^2) = \left(1 + \tanh(100r^2)\right)/2$, where $r^2 = 0.2 - x^2 - y^2$. The computational domain for the microscopic velocity space is established as $(-6,6)^3$. For each iteration of the process, $(N_t, N_x^2, N_v^3) = (12, 12^2, 12^3)$ collocation points are randomly sampled from uniform distributions.

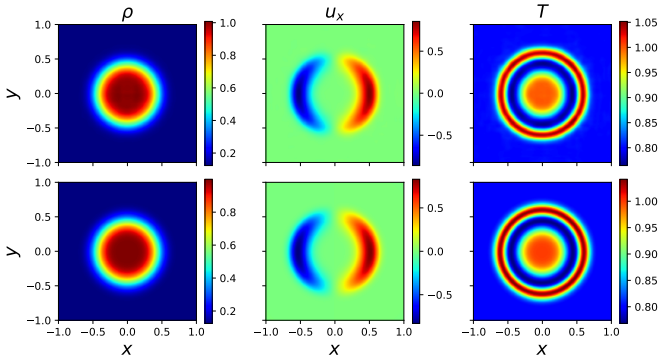
Figure 4 displays the numerical results obtained from our test. The entire computation was completed in approximately 5 minutes and 30 s. For comparison, the reference solution was generated using a grid with $N_x^2 = 160^2$ and $N_v^3 = 33^3$. Our SPINN-BGK successfully generated results that closely align, in qualitative terms, with the reference solution within this timeframe.

Table 7 presents the relative L^2 errors for the predicted macroscopic moments in comparison to the reference solutions. Despite the complexity of the 2D Riemann problem, the magnitude of errors remains within the order of $O(10^{-2})$, demonstrating the effectiveness of our approach in accurately capturing the dynamics of this challenging problem.

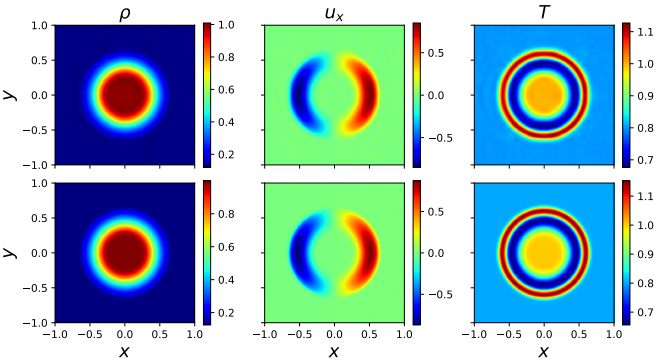
4.5. 3D smooth problem. To test the scalability and effectiveness of SPINN-BGK, we consider a problem of three spatial dimensions with smooth initial data. The spatial domain is set as $(-0.5, 0.5)^3$, over which we impose periodic boundary conditions. The initial condition is defined by a Maxwellian distribution with specific macroscopic moments



(a) Kn = 1.0.



(b) Kn = 0.1.



(c) Kn = 0.01.

FIG. 4. *Macroscopic moments* ρ, u_x, T *at* $t = 0.1$ *for the 2D Riemann problem. For each subfigure, the top row presents neural network predictions, and the bottom row presents reference solutions. Due to the symmetry, we omitted plots for macroscopic velocities* u_y *and* u_z.

$$\rho_0(x,y) = 1 + 0.5\sin(2\pi x)\sin(2\pi y)\sin(2\pi z), \quad \mathbf{u}_0(x,y) = \mathbf{0}, \quad T_0(x,y) = 1.$$

At each iteration of the training procedure, we randomly sample 12 collocation points from a uniform distribution for each axis. The entire computation took approx-

TABLE 8
Relative L^2 errors for the macroscopic moments ρ, u_x, u_y, T of the 3D smooth problem.

$(t=0, t=0.1)$	$\mathrm{Kn}=1$	$\mathrm{Kn}=0.1$	$\mathrm{Kn}=0.01$
ρ	(1.98e-5, 2.47e-5)	(2.40e-5, 7.22e-5)	(2.65e-5, 7.67e-5)
u_x	(3.802-6, 2.07e-5)	(7.59e-6, 3.49e-5)	(1.03e-5, 8.13e-5)
u_y	(5.70e-6, 2.06e-5)	(1.05e-5, 4.99e-5)	(1.14e-5, 1.17e-4)
u_z	(4.98e-6, 2.06e-5)	(8.36e-6, 4.36e-5)	(8.36e-6, 7.23e-5)
T	(6.20e-6, 3.51e-5)	(1.23e-5, 3.96e-5)	(1.18e-5, 1.24e-4)

TABLE 9
Relative L^2 errors for the macroscopic moments ρ, u_x, u_y, T of the 3D Riemann problem.

$(t=0, t=0.1)$	$\mathrm{Kn}=1$	$\mathrm{Kn}=0.1$	$\mathrm{Kn}=0.01$
ρ	(2.58e-2, 1.43e-2)	(2.32e-2, 1.35e-2)	(2.45e-2, 1.16e-2)
u_x	(2.86e-3, 2.03e-2)	(2.70e-3, 2.12e-2)	(2.16e-3, 1.78e-2)
u_y	(2.76e-3, 2.06e-2)	(2.56e-3, 1.99e-2)	(2.56e-3, 1.77e-2)
u_z	(2.84e-3, 2.08e-2)	(2.77e-3, 2.08e-2)	(2.13e-3, 1.75e-2)
T	(6.47e-3, 7.31e-3)	(5.94e-3, 6.58e-3)	(5.23e-3, 7.95e-3)

imately 50 minutes. Table 8 presents relative L^2 errors for the computed macroscopic moments of the 3D smooth problem.

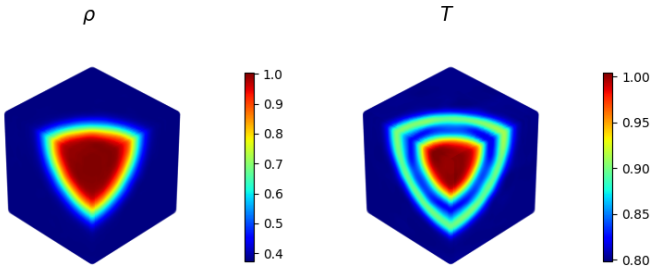
The table shows about four to five digits of accuracy compared to the reference solution generated using a grid with $N_x^3 = 80^3$ and $N_v^3 = 15^3$ on the velocity domain $(-5, 5)^3$, empirically validating the effectiveness of SPINN-BGK.

4.6. 3D Riemann problem. We finally explore the 3D Riemann problem. The spatial domain for this test is defined as $(-1, 1)^3$, where we apply a free-flow boundary condition. Within this domain, a sphere S characterized by $x^2 + y^2 + z^2 = 0.5^2$ is considered. The initial condition is modeled as a local Maxwellian distribution with macroscopic moments $(\rho_0, \mathbf{u}_0, T_0)$. Inside the sphere, the values are set to $(1, \mathbf{0}, 1)$, while outside, they are $(0.375, \mathbf{0}, 0.8)$. As sharper initial data and higher Knudsen numbers require more spatial grids and velocity grids, respectively, we chose this value of initial data to generate a reliable reference solution within a limited computational budget. As with the previous Riemann problems, we employ a smoothed version of the Heaviside function, $H(r^2) = (1 + \tanh(100r^2))/2$, where $r^2 = 0.5^2 - x^2 - y^2 - z^2$, to relax the discontinuity in the initial data. The computational domain for the microscopic velocity space is $(-6, 6)^3$. At each iteration in the optimization procedure, we randomly sample 12 collocation points from a uniform distribution per axis.

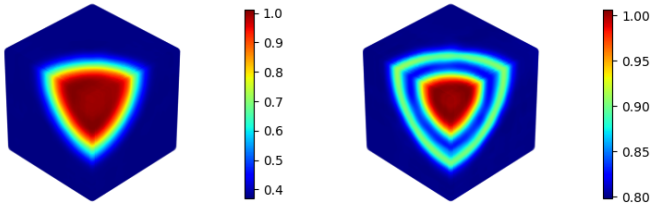
Figure 5 displays the predicted values of ρ and T at $t = 0.1$ within the octant $(0, 1)^2 \times (-1, 0)$. The other octants are not displayed due to the symmetry of the problem. The entire computation was completed in approximately 50 minutes.

Table 9 presents the relative L^2 errors for the predicted macroscopic moments in comparison to the reference solutions generated with $N_x^3 = 40^3$ spatial grids and $N_v^3 = 33^3$ velocity grids. The magnitude of errors remains within the order of $O(10^{-2})$, demonstrating SPINN-BGK's capability of capturing the dynamics of a challenging 3D Riemann problem.

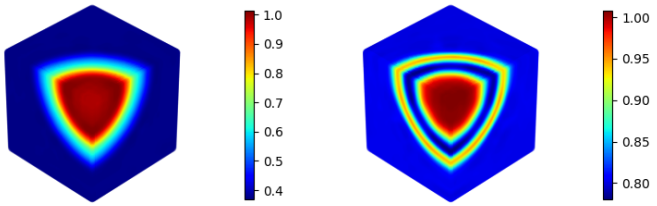
5. Conclusion. In this study, we developed a separable physics-informed neural network–based (SPINN-based) methodology for solving the BGK model of the Boltzmann equation, addressing several key computational challenges. These challenges include the high dimensionality of the equation, the computational burden of integral evaluations for macroscopic moments, and the need for appropriate decay be-



(a) Kn = 1.0



(b) Kn = 0.1



(c) Kn = 0.01

FIG. 5. *Predicted macroscopic moments at* $t = 0.1$ *for the* 3*D Riemann problem. Left: the macroscopic density* ρ*. Right: the macroscopic temperature* T*.*

havior of the solution for large velocity values. To tackle these issues, we adopted the SPINN structure, which significantly reduces the number of network forward passes and leverages a separable structure to minimize the computational burden of integrals. Furthermore, we introduced SPINN-BGK, a tailored variant incorporating Gaussian

functions into the neural network for each velocity component, and implemented a relative loss function to dynamically adjust weights at each collocation point, enhancing model accuracy. Our numerical tests demonstrated that SPINN-BGK produces solutions closely aligned with reference solutions for 1D, 2D, and 3D BGK problems. In addition to our methodological contributions, we are committed to releasing our source code[1] to benefit the computational science and numerical analysis communities.

Our developments suggest several promising extensions. One avenue is the partial separation strategy to better handle complex geometries within the velocity domain of the BGK model. This involves segmenting the domain into time-space-velocity components and further subdividing the velocity domain. Another area for improvement is addressing the memory-intensive nature of backward mode automatic differentiation used in our network parameter updates. Exploring rematerialization strategies or developing methods that do not require full tensors could significantly reduce memory usage while maintaining computational efficiency. Additionally, investigating more effective training methods and alternative activation functions, such as ReLU or their smooth variants, may enhance the performance of SPINN-BGK, especially for problems covering a wide range of Knudsen numbers or discontinuous initial data; see, e.g., [29, 56].

Appendix A. Proof of Theorem 3.3. We begin by highlighting the necessity of the assumption $\int f\phi\, d\mathbf{v} \in L^2(\Omega)$, where $\phi(\mathbf{v}) = 1 + |\mathbf{v}|^2$. This condition is stronger than the unweighted counterpart and is equivalent to requiring that $\int v f\, d\mathbf{v}$ and $\int |\mathbf{v}|^2 f\, d\mathbf{v}$ belong to $L^2(\Omega)$. These assumptions are crucial for guaranteeing the finiteness of the macroscopic velocity $\mathbf{u}$ and temperature T in the L^2 sense.

Next, we outline the proof of Theorem 3.3. The space $\mathbb{R}^3$ is partitioned into a compact subset C and its complement $C^c := \mathbb{R}^3 \setminus C$, where the contribution of f is negligible outside C. Within C, we apply Theorem 3.1, while for C^c, a zero extension of the neural network is utilized.

Proof. We first note that $f\phi$ exhibits rapid decay along the velocity domain since

$$M := \int_{\mathbb{R}^3} f\phi d\mathbf{v} \in L^2(\Omega).$$

Let $B(r)$ be a ball of radius r centered at the origin. We define M_r as $\int_{B(r)} f\phi d\mathbf{v}$. Since $f\phi \geq 0$, we have $0 \leq (M - M_r)^2 \leq M^2$, and $(M - M_r) \downarrow 0$ as $r \to \infty$. With $M^2 \in L^1(\Omega)$, the dominated convergence theorem implies

$$\lim_{r\to\infty} \int_\Omega (M - M_r)^2 dx = \lim_{r\to\infty} \left\| \int_{B(r)^c} f\phi d\mathbf{v} \right\|^2_{L^2(\Omega)} = 0.$$

Hence, we can choose a compact subset $A \subset \mathbb{R}^3$ such that

$$\left\| \int_{A^c} f\phi d\mathbf{v} \right\|_{L^2(\Omega)} < \frac{\epsilon}{2}.$$

On the other hand, from $f \in L^2(\Omega \times \mathbb{R}^3)$, we have

$$\int_\Omega \int_{\mathbb{R}^3} f^2 d\mathbf{v} dx < \infty.$$

[1] https://github.com/jaeminoh/SPINN-BGK.

We choose a compact subset $B \subset \mathbb{R}^3$ such that

$$\int_\Omega \int_{B^c} f^2 d\mathbf{v} dx < \frac{\epsilon^2}{4}.$$

Let $C = A \cup B$. We divide the microscopic velocity space $\mathbb{R}^3$ into two parts, C and C^c. For C, by Theorem 3.1, there exists a separable neural network f_θ such that

$$\|f - f_\theta\|_{L^2(\Omega\times C)} < \frac{\epsilon}{2(1+|C|^2)\left(1+m(C)^{1/2}\right)} \le \frac{\epsilon}{2},$$

where $|C| := \sup\{|v| : v \in C\}$ and $m(C) = \int_C dx$. For C^c, we extend f_θ by zero to $\Omega \times \mathbb{R}^3$. Then we have

$$\begin{aligned}\|f - f_\theta\|_{L^2(\Omega\times\mathbb{R}^3)} &\le \|f - f_\theta\|_{L^2(\Omega\times C)} + \|f - f_\theta\|_{L^2(\Omega\times C^c)} \\ &= \|f - f_\theta\|_{L^2(\Omega\times C)} + \|f\|_{L^2(\Omega\times C^c)} \\ &\le \frac{\epsilon}{2(1+|C|^2)} + \frac{\epsilon}{2} \le \epsilon.\end{aligned}$$

Moreover,

$$\begin{aligned}\left\|\int_{\mathbb{R}^3}(f-f_\theta)\phi d\mathbf{v}\right\|_{L^2(\Omega)} &\le \left\|\int_C (f-f_\theta)\phi d\mathbf{v}\right\|_{L^2(\Omega)} + \left\|\int_{C^c}(f-f_\theta)\phi d\mathbf{v}\right\|_{L^2(\Omega)} \\ &= \left\|\int_C (f-f_\theta)\phi d\mathbf{v}\right\|_{L^2(\Omega)} + \left\|\int_{C^c} f\phi d\mathbf{v}\right\|_{L^2(\Omega)} \\ &\le \left\|\int_C (f-f_\theta)\phi d\mathbf{v}\right\|_{L^2(\Omega)} + \frac{\epsilon}{2} \\ &\le \left[\int_\Omega \left(\int_C (f-f_\theta)^2 d\mathbf{v} \int_C \phi^2 d\mathbf{v}\right) dx\right]^{1/2} + \frac{\epsilon}{2} \\ &\le \|f-f_\theta\|_{L^2(\Omega\times C)}(1+|C|^2)^{1/2} m(C)^{1/2} + \frac{\epsilon}{2} \le \epsilon. \qquad \square\end{aligned}$$

Appendix B. Computational complexity and Knudsen numbers. In Figure 3, slight deviations are observable at both $t = 0$ and $t = 0.1$, unlike in Figure 2. These deviations can be attributed to the abrupt changes in the particle density function f across $x = 0$, where both the magnitude and dispersion of the particle density function undergo more pronounced shifts compared to those observed in Figure 2. Furthermore, for $\mathrm{Kn} = 1$, the deviations observed in the temperature may be linked to the distinct "shape" of the particle density function.

Figure 6 displays slices of the particle density function $v_x \mapsto f$ at $t = 0.1$, $x = 0.1496$, $v_y = v_z = 0$, for $\mathrm{Kn} \in \{1, 0.1, 0.01\}$. For $\mathrm{Kn} = 0.01$, the density function exhibits a bell-shaped curve. In contrast, for $\mathrm{Kn} \in \{1, 0.1\}$, the function displays a bimodal distribution with two peaks, suggesting the need for a higher density of collocation points in the v_x direction, thereby increasing the computational complexity.

Appendix C. Computation time and additional details for the reference method. Table 10 presents the computation times for solutions across various problems addressed in section 4. Even though the number of forward passes decreased drastically thanks to the canonical polyadic decomposition structure of the SPINN, the computational cost for the 3D problems is huge. One of the reasons is that SPINN-BGK requires a full tensor to evaluate the loss function, and it significantly increases the memory cost as discussed in section 5.

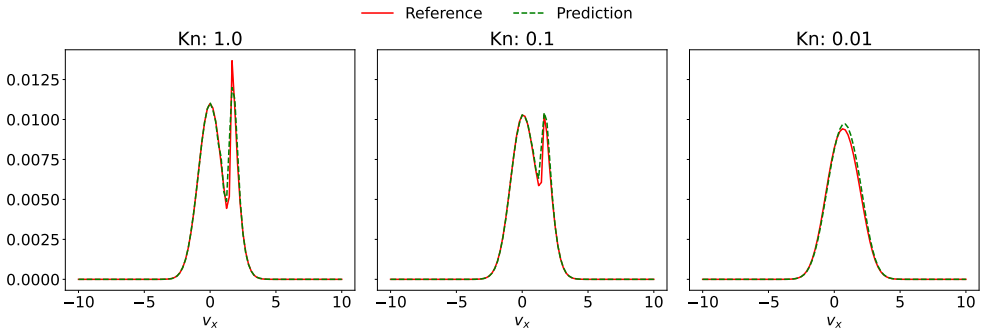


FIG. 6. *Slices of the particle density* $v_x \mapsto f(t, x, v_x, v_y, v_z)$ *at* $t = 0.1$, $x = 0.1496$, *and* $v_y = v_z = 0$ *for three Knudsen numbers:* 1.0, 0.1, *and* 0.01. *Bimodal shapes significantly increase computational complexity when solving the BGK model for* $\mathrm{Kn} \in \{1.0, 0.1\}$. *The red solid lines represent the reference solutions, while the green dotted lines are the SPINN-BGK outputs.*

TABLE 10
Required computation times for section 4.

SPINN-BGK	$d = 1$	$d = 2$	$d = 3$
Smooth	240s (4.1)	308s (4.3)	2998s (4.5)
Riemann	250s (4.2)	331s (4.4)	3012s (4.6)

TABLE 11
Additional details for generating reference solutions used in section 4.

Problem	Grid (N_x^d, N_v^3)	Computational domain	Elapsed time
4.1	$(1280, 25^3)$	$(x, \mathbf{v}) \in (-0.5, 0.5) \times (-10, 10)^3$	2598s
4.2	$(1280, 97 \times 25^2)$	$(x, \mathbf{v}) \in (-0.5, 0.5) \times (-10, 10)^3$	5312s
4.3	$(160^2, 25^3)$	$(\mathbf{x}, \mathbf{v}) \in (-0.5, 0.5)^2 \times (-10, 10)^3$	13764s
4.4	$(160^2, 33^3)$	$(\mathbf{x}, \mathbf{v}) \in (-1, 1)^2 \times (-6, 6)^3$	36020s
4.5	$(80^3, 15^3)$	$(\mathbf{x}, \mathbf{v}) \in (-0.5, 0.5)^3 \times (-5, 5)^3$	47040s
4.6	$(40^3, 33^3)$	$(\mathbf{x}, \mathbf{v}) \in (-1, 1)^3 \times (-6, 6)^3$	30600s

The computational details for generating reference solutions throughout section 4 are presented in Table 11. Despite the 3D Riemann problem (4.6) having more degrees of freedom than the 3D smooth problem (4.5), the last two rows indicate that computation times are reversed. This phenomenon is attributed to the CFL condition [12] associated with the usage of second-order explicit Runge–Kutta time stepping, where a finer spatial discretization necessitates a finer temporal discretization.

REFERENCES

[1] A. ALEKSEENKO, R. MARTIN, AND A. WOOD, *Fast evaluation of the Boltzmann collision operator using data driven reduced order models*, J. Comput. Phys., 470 (2022), 111526, https://doi.org/10.1016/j.jcp.2022.111526.

[2] M. BENNOUNE, M. LEMOU, AND L. MIEUSSENS, *Uniformly stable numerical schemes for the Boltzmann equation preserving the compressible Navier–Stokes asymptotics*, J. Comput. Phys., 227 (2008), pp. 3781–3803, https://doi.org/10.1016/j.jcp.2007.11.032.

[3] P. L. BHATNAGAR, E. P. GROSS, AND M. KROOK, *A model for collision processes in gases.* I. *Small amplitude processes in charged and neutral one-component systems*, Phys. Rev., 94 (1954), pp. 511–525, https://doi.org/10.1103/PhysRev.94.511.

[4] G. A. BIRD, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Oxford University Press, New York, 1994.

[5] A. M. Boelens, D. Venturi, and D. M. Tartakovsky, *Tensor methods for the Boltzmann-BGK equation*, J. Comput. Phys., 421 (2020), 109744, https://doi.org/10.1016/j.jcp.2020.109744.

[6] W. Boscheri and G. Dimarco, *High order central weno-implicit-explicit Runge Kutta schemes for the BGK model on general polygonal meshes*, J. Comput. Phys., 422 (2020), 109766, https://doi.org/10.1016/j.jcp.2020.109766.

[7] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, *JAX: Composable Transformations of Python+NumPy Programs*, 2018, http://github.com/google/jax.

[8] S. Chapman and T. G. Cowling, *The Mathematical Theory of Non-uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases*, Cambridge University Press, Cambridge, UK, 1990.

[9] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, et al., *Symbolic discovery of optimization algorithms*, Adv. Neural Inform. Process., 36 (2023), pp. 49205–49233.

[10] J. Cho, S. Nam, H. Yang, S.-B. Yun, Y. Hong, and E. Park, *Separable physics-informed neural networks*, Adv. Neural Inf. Process. Syst., 36 (2024).

[11] S. Y. Cho, S. Boscarino, G. Russo, and S.-B. Yun, *Conservative semi-Lagrangian schemes for kinetic equations Part* II*: Applications*, J. Comput. Phys., 436 (2021), 110281, https://doi.org/10.1016/j.jcp.2021.110281.

[12] R. Courant, K. Friedrichs, and H. Lewy, *Über die partiellen differenzengleichungen der mathematischen physik*, Math. Ann., 100 (1928), pp. 32–74, https://doi.org/10.1007/BF01448839.

[13] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Systems, 2 (1989), pp. 303–314, https://doi.org/10.1007/BF02551274.

[14] DeepMind, I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios, J. Quan, R. Ring, F. Ruiz, A. Sanchez, L. Sartran, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, M. Stanojević, W. Stokowiec, L. Wang, G. Zhou, and F. Viola, *The DeepMind JAX Ecosystem*, 2020, http://github.com/google-deepmind.

[15] G. Dimarco and R. Loubere, *Towards an ultra efficient kinetic scheme. Part* I*: Basics on the BGK equation*, J. Comput. Phys., 255 (2013), pp. 680–698, https://doi.org/10.1016/j.jcp.2012.10.058.

[16] G. Dimarco, R. Loubère, J. Narski, and T. Rey, *An efficient numerical method for solving the Boltzmann equation in multidimensions*, J. Comput. Phys., 353 (2018), pp. 46–81, https://doi.org/10.1016/j.jcp.2017.10.010.

[17] G. Dimarco and L. Pareschi, *Numerical methods for kinetic equations*, Acta Numer., 23 (2014), pp. 369–520, https://doi.org/10.1017/S0962492914000063.

[18] M. Ding, J.-M. Qiu, and R. Shu, *Semi-Lagrangian nodal discontinuous Galerkin method for the BGK model*, Adv. Comput. Sci. Eng., 2 (2024), pp. 222–245.

[19] F. Filbet and S. Jin, *A class of asymptotic-preserving schemes for kinetic equations and related problems with stiff sources*, J. Comput. Phys., 229 (2010), pp. 7625–7648, https://doi.org/10.1016/j.jcp.2010.06.017.

[20] I. M. Gamba, S. Jin, and L. Liu, *Micro-macro decomposition based asymptotic-preserving numerical schemes and numerical moments conservation for collisional nonlinear kinetic equations*, J. Comput. Phys., 382 (2019), pp. 264–290, https://doi.org/10.1016/j.jcp.2019.01.018.

[21] W. Guo, J. F. Ema, and J.-M. Qiu, *A Local Macroscopic Conservative (LoMaC) low rank tensor method with the discontinuous Galerkin method for the Vlasov dynamics*, Commun. Appl. Math. Comput., (2023), pp. 1–26.

[22] W. Guo and J.-M. Qiu, *A Conservative Low Rank Tensor Method for the Vlasov Dynamics*, SIAM J. Sci. Comput., 46 (2024), pp. A232–A263, https://doi.org/10.1137/22M1473960.

[23] J. Han, C. Ma, Z. Ma, and W. E, *Uniformly accurate machine learning-based hydrodynamic models for kinetic equations*, Proc. Natl. Acad. Sci. USA, 116 (2019), pp. 21983–21991, https://doi.org/10.1073/pnas.1909854116.

[24] R. D. Hazeltine, *The Framework of Plasma Physics*, CRC Press, Boca Raton, FL, 2018.

[25] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, J. Math. Phys., 6 (1927), pp. 164–189, https://doi.org/10.1002/sapm192761164.

[26] Z. Hu, Z. Shi, G. E. Karniadakis, and K. Kawaguchi, *Hutchinson trace estimation for high-dimensional and high-order physics-informed neural networks*, Comput. Methods Appl. Mech. Engrg., 424 (2024), 116883.
[27] Z. Hu, K. Shukla, G. E. Karniadakis, and K. Kawaguchi, *Tackling the curse of dimensionality with physics-informed neural networks*, Neural Netw., 176 (2024), 106369.
[28] *MATLAB Version:* 9.13.0 *(r*2022*b)*, The MathWorks, 2022, https://www.mathworks.com.
[29] S. Jin, Z. Ma, and T.-a. Zhang, *Asymptotic-preserving neural networks for multiscale Vlasov–Poisson–Fokker–Planck system in the high-field regime*, J. Sci. Comput., 99 (2024), 61, https://doi.org/10.1007/s10915-024-02527-z.
[30] S. Jin and Y. Shi, *A micro-macro decomposition-based asymptotic-preserving scheme for the multispecies Boltzmann equation*, SIAM J. Sci. Comput., 31 (2010), pp. 4580–4606, https://doi.org/10.1137/090756077.
[31] P. Kidger and T. Lyons, *Universal approximation with deep narrow networks*, in Proceedings of the Conference on Learning Theory, PMLR, 2020, pp. 2306–2327.
[32] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, preprint, arXiv: 1412.6980, 2014.
[33] J. Y. Lee, J. Jang, and H. J. Hwang, *opPINN: Physics-informed neural network with operator learning to approximate solutions to the Fokker-Planck-Landau equation*, J. Comput. Phys., 480 (2023), 112031, https://doi.org/10.1016/j.jcp.2023.112031.
[34] E. E. Lewis and W. F. Miller, *Computational Methods of Neutron Transport*, John Wiley and Sons, New York, 1984.
[35] Z. Li, Y. Wang, H. Liu, Z. Wang, and B. Dong, *Solving the Boltzmann equation with a neural sparse representation*, SIAM J. Sci. Comput., 46 (2024), pp. C186–C215, https://doi.org/10.1137/23M1558227.
[36] L. Liu and K. Qi, *Convergence of the Fourier-Galerkin Spectral Method for the Boltzmann Equation with Uncertainties*, https://arxiv.org/abs/2212.04083, 2024.
[37] I. Loshchilov and F. Hutter, *SGDR: Stochastic Gradient Descent with Warm Restarts*, preprint, arXiv:1608.03983, 2016.
[38] Q. Lou, X. Meng, and 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, https://doi.org/10.1016/j.jcp.2021.110676.
[39] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, *Physics-informed neural networks with hard constraints for inverse design*, SIAM J. Sci. Comput., 43 (2021), pp. B1105–B1132, https://doi.org/10.1137/21M1397908.
[40] P. A. Markowich, C. A. Ringhofer, and C. Schmeiser, *Semiconductor Equations*, Springer Science & Business Media, New York, 2012.
[41] L. Mieussens, *Discrete velocity model and implicit scheme for the BGK equation of rarefied gas dynamics*, Math. Models Methods Appl. Sci., 10 (2000), pp. 1121–1149, https://doi.org/10.1142/S0218202500000562.
[42] S. T. Miller, N. V. Roberts, S. D. Bond, and E. C. Cyr, *Neural-network based collision operators for the Boltzmann equation*, J. Comput. Phys., 470 (2022), 111541, https://doi.org/10.1016/j.jcp.2022.111541.
[43] C. Mouhot and L. Pareschi, *Fast algorithms for computing the Boltzmann collision operator*, Math. Comp., 75 (2006), pp. 1833–1852, https://doi.org/10.1090/S0025-5718-06-01874-6.
[44] J. Müller and M. Zeinhofer, *Achieving high accuracy with pinns via energy natural gradient descent*, in Proceedings of the International Conference on Machine Learning, PMLR, 2023, pp. 25471–25485.
[45] L. Pareschi and B. Perthame, *A Fourier spectral method for homogeneous Boltzmann equations*, Transport Theory Stat. Phys., 25 (1996), pp. 369–382, https://doi.org/10.1080/00411459608220707.
[46] L. Pareschi and G. Russo, *Numerical solution of the Boltzmann equation* I*: Spectrally accurate approximation of the collision operator*, SIAM J. Numer. Anal., 37 (2000), pp. 1217–1245, https://doi.org/10.1137/S0036142998343300.
[47] S. Pieraccini and G. Puppo, *Implicit–explicit schemes for BGK kinetic equations*, J. Sci. Comput., 32 (2007), pp. 1–28, https://doi.org/10.1007/s10915-006-9116-6.
[48] W. A. Porteous, M. P. Laiu, and C. D. Hauck, *Data-driven, Structure-Preserving Approximations to Entropy-Based Moment Closures for Kinetic Equations*, preprint, arXiv: 2106.08973, 2021.
[49] M. Raissi, P. Perdikaris, and 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), pp. 686–707, https://doi.org/10.1016/j.jcp.2018.10.045.

[50] G. Romano, *OpenBTE: A Solver for Ab-Initio Phonon Transport in Multidimensional Structures*, preprint, arXiv:2106.02764, 2021.

[51] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, Adv. Neural Inf. Process. Syst., 33 (2020), pp. 7462–7473.

[52] P.-H. Tsai, S. W. Chung, D. Ghosh, J. Loffeld, Y. Choi, and J. L. Belof, *Accelerating Kinetic Simulations of Electrostatic Plasmas with Reduced-Order Modeling*, preprint, arXiv:2310.18493, 2023.

[53] C. Villani, *A review of mathematical topics in collisional kinetic theory*, Handb. Math. Fluid Dyn., 1 (2002), pp. 3–8, https://doi.org/10.1016/S1874-5792(02)80004-0.

[54] T. Xiao and M. Frank, *RelaxNet: A structure-preserving neural network to approximate the Boltzmann collision operator*, J. Comput. Phys., 490 (2023), 112317, https://doi.org/10.1016/j.jcp.2023.112317.

[55] T. Xiong, J. Jang, F. Li, and J.-M. Qiu, *High order asymptotic preserving nodal discontinuous Galerkin IMEX schemes for the BGK equation*, J. Comput. Phys., 284 (2015), pp. 70–94, https://doi.org/10.1016/j.jcp.2014.12.021.

[56] T.-a. Zhang and S. Jin, *AP-MIONet: Asymptotic-Preserving Multiple-Input Neural Operators for Capturing the High-Field Limits of Collisional Kinetic Equations*, preprint, arXiv:2407.14921, 2024.

[57] J. Zhou, R. Li, and T. Luo, *Physics-informed neural networks for solving time-dependent mode-resolved phonon Boltzmann transport equation*, npj Comput. Mater., 9 (2023), 212, https://doi.org/10.1038/s41524-023-01165-7.