Kinetic Predicted-Moment Flux Reconstruction for High-Order High-Performance Fluid Simulation

ZIKE XU, ShanghaiTech University, China
XUAN ZHANG, ShanghaiTech University, China
XIAOPEI LIU, ShanghaiTech University, China

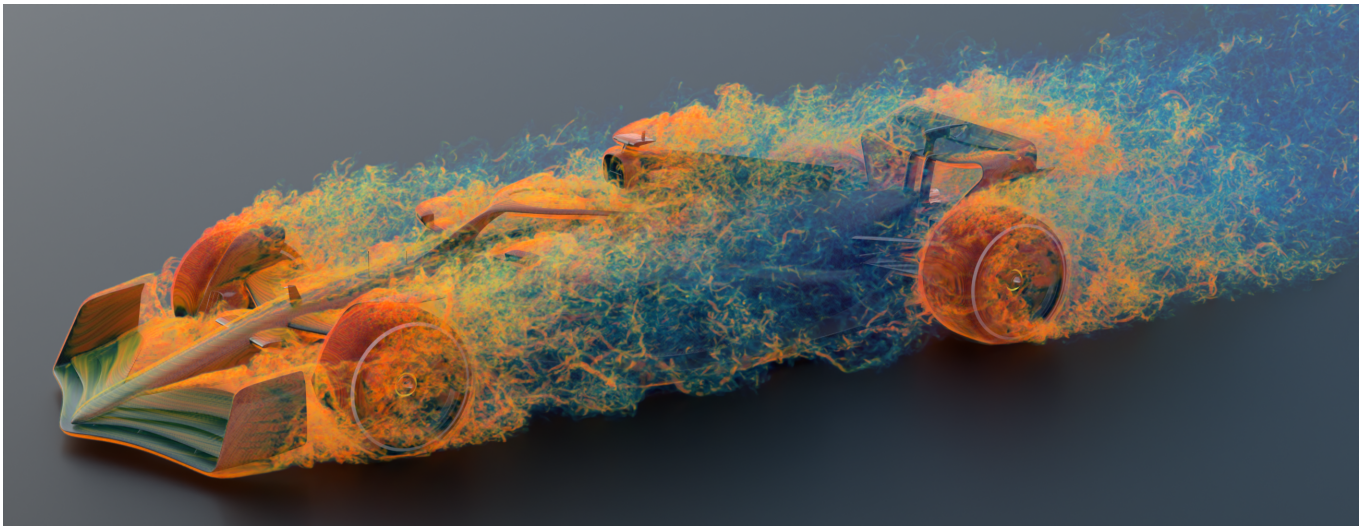


Fig. 1. **Turbulent airflow surrounding a running single-seater race car.** We develop Kinetic Predicted-Moment Flux Reconstruction (KPM-FR), a spatially high-order fluid simulation method that combines high simulation fidelity, large computational throughput, and compact state storage in a hardware-aware GPU solver. Here, we present a single-resolution aerodynamic simulation of a 5.5 m-long single-seater race car traveling at 220 km/h, with a uniform grid of $2000 \times 332 \times 800$ solution points. The simulation demonstrates the scale reached by the single-GPU implementation: it captures this high-resolution scenario using only 21 GiB of memory. This brings high-resolution aerodynamics into a desktop-GPU memory budget, with per-frame computation time comparable to that of highly optimized LBM, all while preserving the rich multiscale vortical structures observed in the volume rendering.

The simultaneous pursuit of high fidelity, large computational throughput, and a minimal memory footprint has long constituted the central challenge in fluid simulation research. Yet state-of-the-art methods struggle to reconcile all these objectives, and often entail navigating trade-offs among them. We present Kinetic Predicted-Moment Flux Reconstruction (KPM-FR), a high-order kinetic-based scheme for low-Mach-number weakly compressible flows that advances all three fronts within a single framework. KPM-FR is a flux-form fluid flow solver rooted in the principles of the gas-kinetic scheme (GKS), deriving numerical fluxes from the locally evolved Boltzmann-BGK equation to recover Navier-Stokes (NS) solutions. Departing from the GKS and its variants, it carries out kinetic evolution entirely in moment space within the high-order flux reconstruction (FR) framework through a concise predictor-corrector scheme. This translates to two fused GPU kernels per time step, streamlining computation and confining intermediate data to on-chip memory. This design confers several practical advantages. First, compared to conventional high-fidelity lattice Boltzmann methods (LBM), the moment-based formulation reduces the per-point memory footprint by more than fivefold. Second, at matched resolutions, its high-order spatial formulation exhibits markedly lower numerical dissipation, preserving fine-scale vortical structures with greater fidelity. Combining these advantages with a near-saturated throughput exceeding 8 billion solution-point updates per second on a single consumer GPU, KPM-FR delivers large-scale fluid simulation on commodity hardware. Quantitative benchmarks confirm high-order spatial convergence and spectral-like dissipation characteristics, while validation against reference data for flow past solid bodies verifies its practical accuracy. Ultimately, we demonstrate the versatility of KPM-FR across complex geometries and large-scale turbulent flows, capturing multiscale structures with 1.8 billion solution points on a single desktop workstation.

Authors' Contact Information: Zike Xu, ShanghaiTech University, Shanghai, China, kriaeth@outlook.com; Xuan Zhang, ShanghaiTech University, Shanghai, China, xuanzhang2002@gmail.com; Xiaopei Liu, ShanghaiTech University, Shanghai, China, aurorean.xp@gmail.com.

ACM 1557-7368/2026/7-ART67
https://doi.org/10.1145/3811292

CCS Concepts: • **Computing methodologies → Physical simulation**.

ACM Reference Format:
Zike Xu, Xuan Zhang, and Xiaopei Liu. 2026. Kinetic Predicted-Moment Flux Reconstruction for High-Order High-Performance Fluid Simulation. *ACM Trans. Graph.* 45, 4, Article 67 (July 2026), 25 pages. https://doi.org/10.1145/3811292

1 Introduction

Simulating the transient dynamic behavior of complex fluid flows, an endeavor central to computational fluid dynamics (CFD) and a key research area in computer graphics (CG), has been shaped for decades by a persistent tension among three critical objectives. *High fidelity* mandates stable solutions that are both physically accurate and visually plausible. *Large computational throughput* requires enabling the execution of high-resolution simulations within feasible timeframes. *Minimal memory footprint* ensures these computationally demanding simulations are tractable on commodity hardware. For much of the existing literature, this tension has necessitated compromises in prioritizing these three objectives, driving the field to converge on a pragmatic trade-off: the widespread adoption of specialized solvers tailored to distinct application requirements.

This trade-off is clearly reflected in the dominant methodologies across both CFD and CG. Traditional incompressible Navier-Stokes (NS) solvers for low-speed flows, for instance, excel at enforcing incompressibility [Chorin 1968; Harlow and Welch 1965], but suffer from the computational cost of a global pressure solve and heightened numerical dissipation in advection-dominated flows [Ferziger et al. 2020; Karniadakis and Sherwin 2005]. To mitigate these limitations, kinetic methods have emerged as a compelling alternative. The lattice Boltzmann method (LBM), a weakly compressible solver for approximating incompressible fluid flow behavior, owes its low numerical dissipation to its local, conservative streaming and collision processes, rendering it highly effective for resolving fine-scale structures in vortical and even turbulent flows [Krüger et al. 2017]. This advantage, however, is offset by the substantial memory footprint incurred by storing full distribution functions. A promising remedy, explored in moment-representation LBM (MR-LBM) [Ferrari et al. 2023; Li et al. 2023; Valero-Lara et al. 2023], is to store only macroscopic moments. This approach, however, introduces a critical dilemma: the computationally expensive reconstruction of distribution functions from moments, a step necessary for streaming, reintroduces enhanced numerical dissipation. Furthermore, the large shared memory footprint of these reconstructed distribution functions introduces a new computational bottleneck. Developed in parallel with LBM, the gas-kinetic scheme (GKS) [Xu 2001] acts as a flux-based alternative that obviates the need to store distribution functions by constructing macroscopic fluxes through local kinetic evolution. This conceptual elegance, however, is offset by its high arithmetic intensity and the numerical dissipation inherent to its conventional finite-volume spatial discretization [Guo et al. 2008].

While the LBM has become a widely adopted kinetic solver, particularly for vorticity-dominated regimes where low numerical dissipation is paramount, its reliance on lattice-based streaming imposes an intrinsic structural limitation: the trade-off between memory overhead and solution fidelity. Rather than continuing within this paradigm, we propose a fundamental departure. We pivot to the flux-centric philosophy of the GKS, which fundamentally decouples kinetic evolution from velocity-space storage to ensure a minimal memory footprint. To address the computational complexity and numerical dissipation typically associated with traditional GKS, we reformulate the kinetic flux within the high-order flux reconstruction (FR) framework [Huynh 2007]. By retaining the core GKS principles for interface flux reconstruction and exploiting its coupled advection-diffusion physics, we enable FR to serve as a highly efficient high-order spatial discretization engine, which reduces numerical dissipation significantly. Collectively, this novel formulation, which we name Kinetic Predicted-Moment Flux Reconstruction (KPM-FR), mitigates the individual limitations of both GKS and FR, yielding a new solver that fulfills our design objectives.

We realize KPM-FR through an efficient *predictor-corrector* scheme designed for low-Mach-number flows, combining a compact mathematical formulation with hardware-oriented optimization to achieve high computational performance. The key to its efficiency lies in the use of kinetic moments as intermediaries, which enables a compact *encode-transmit-decode* pipeline that substantially reduces memory traffic and register consumption, while condensing the kinetic evolution into a streamlined algebraic operation sequence. Built on this mathematical compactness, we design a hardware-aware architecture that significantly improves the efficiency of high-order FR methods for tensor-product elements on GPUs. By explicitly fusing the entire workflow into a minimal two-kernel cycle, this design achieves near-roofline memory-bandwidth utilization, sustaining a throughput of over 8 billion solution-point updates per second for a complete time step on an NVIDIA RTX 4090 GPU. Consequently, this raw computational throughput translates to time-to-solution gains over state-of-the-art LBMs at comparable dissipation error, while simultaneously reducing memory requirements to only 18% of standard LBM implementation in graphics [Lyu et al. 2023], 34% of a high-fidelity in-place streaming implementation in CFD [Geier et al. 2025], and 53% of MR-LBM [Li et al. 2023]. Notably, this efficiency and compact memory footprint are attained without compromising numerical fidelity: our method exhibits lower numerical dissipation than the state-of-the-art cumulant LBM [Geier et al. 2017].

Our scheme is systematically validated through a series of tests, ranging from fundamental vortex benchmarks that verify spectral-like fidelity to challenging high-Reynolds-number aerodynamic flows. We extensively assess the method's performance in terms of computational throughput and memory efficiency by comparing it against state-of-the-art methods, revealing a compounding advantage: our solver not only sustains high computational throughput at equivalent resolutions, but more importantly, achieves comparable fidelity with significantly fewer solution points. Finally, we demonstrate the solver's versatility in resolving intricate turbulent flow features across practical scenarios, as illustrated by the airflow around a moving race car in Fig. 1. It faithfully captures complex wake dynamics under tight computational budgets, verifying that high-order fidelity, high efficiency, and large-scale simulation can be concurrently achieved on a single commodity GPU.

2 Background and Related Work

Prior to introducing our kinetic-based scheme, we begin by providing background on fluid models governed by both the NS equations and the Boltzmann-BGK equation. We then briefly review numerical methods for these models, spanning both the computer graphics and computational fluid dynamics fields. Given that our scheme is grounded in the flux reconstruction framework, we also summarize

relevant prior work in this area. We close this section by outlining the distinctions between our approach and existing methods, followed by a breakdown of our contributions in itemized form.

2.1 Navier-Stokes formulation

Fluid flow under typical low-speed isothermal conditions, a common scenario in practice, is usually governed by the following incompressible Navier-Stokes (NS) equations [Ferziger et al. 2020]:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\frac{1}{\rho_0} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{g}, \\ \nabla \cdot \mathbf{u} &= 0, \end{aligned} \tag{1}$$

where $\mathbf{u}$ and p denote the velocity and pressure fields, respectively; ρ_0 is the reference density; ν is the kinematic viscosity; and $\mathbf{g}$ is the volumetric acceleration field. The primary challenge in solving the above model equations lies in efficiently enforcing the divergence-free condition $\nabla \cdot \mathbf{u} = 0$, as well as achieving conservative advection with minimal numerical dissipation and dispersion. In the CFD field, fundamental numerical solutions for the incompressible NS equations were first established via finite-difference methods on staggered grids, most notably the marker-and-cell (MAC) scheme [Harlow and Welch 1965]. To enhance conservation enforcement on body-fitted meshes, finite-volume (FV) discretizations emerged as the de facto standard, enabling geometric conformity for practical applications through pressure-correction algorithms such as SIMPLE [Patankar 1980]. Concurrently, finite-element methods (FEM) were developed to offer a rigorous variational foundation, with inherent support for unstructured meshes [Zienkiewicz and Taylor 2013]. However, the divergence-free constraint inevitably requires solving a computationally expensive Poisson equation globally. To circumvent this bottleneck and leverage modern parallel hardware architectures, artificial compressibility methods (ACM) were proposed to relax this constraint [Chorin 1967; Clausen 2013], thereby converting the system to a hyperbolic form suitable for explicit time-stepping. This explicit computational efficiency aligns with the growing demands for large-eddy simulation (LES) of turbulent flows, a trend that has further spurred the adoption of high-order spectral element discretizations aimed at minimizing numerical dissipation and dispersion [Wang et al. 2013].

While CFD methods typically target predictive accuracy, numerical methods in computer graphics have historically prioritized cost-effectiveness and stability, reflecting their respective application requirements. The dominant paradigm discretizes governing equations on regular grids using low-order finite-difference or finite-volume schemes, paired with a pressure-projection step analogous to CFD approaches. The landmark Stable Fluids method [Stam 1999] introduced an efficient semi-Lagrangian framework that remains ubiquitous; its variants enhance accuracy or visual fidelity through higher-order or error-compensated advection schemes [Dupont and Liu 2003; Fedkiw et al. 2001; Qu et al. 2019; Selle et al. 2008; Zehnder et al. 2018]. Complementary work has developed turbulence synthesis techniques [Kim et al. 2008; Pfaff et al. 2010], while additional efforts boost small-scale detail via vorticity-based formulations, including vortex filaments, vortex sheets [Brochu et al. 2012; Pfaff et al. 2012; Weißmann and Pinkall 2010; Zhang et al. 2015], and covector methods [Nabizadeh et al. 2022]. Beyond structured grids, researchers have explored energy-preserving formulations and finite-volume schemes for tetrahedral meshes [Mullen et al. 2009]. More recently, a body of work has proposed using flow maps to represent the flow state, enabling low-dissipation long-time advection [Deng et al. 2023; Zhou et al. 2024]. Hybrid Eulerian-Lagrangian methods couple grid-based pressure and diffusion solvers with particle-based advection, yielding PIC/FLIP-style schemes (e.g., APIC) that enforce incompressibility and reduce dissipation of original PIC [Jiang et al. 2015; Markidis et al. 2018; Nabizadeh et al. 2024; Qu et al. 2022; Raveendran et al. 2011]. Similar to our work targeting large-scale fluid simulation under limited computational resources, Golas et al. [2012] and Liu et al. [2016] have shown that pressure-projection solvers can achieve very large grid resolutions. All the above-mentioned methods have matured considerably, delivering impressive results across a diverse range of applications. As the graphics community increasingly pursues quantitative predictive fidelity alongside visual realism, however, two additional properties have become critical: conservative, low-dissipation nonlinear advection, and fully explicit time integration free of global linear solves. This motivates the development of a novel solver that combines predictive fidelity with fully explicit GPU-optimized execution.

2.2 Kinetic formulation

To address the longstanding challenges of incompressible NS solvers, kinetic approaches have grown increasingly prominent in both CFD and CG communities in recent years. Intuitively, their appeal arises from the mathematical structure of the underlying continuous Boltzmann equation [Krüger et al. 2017]: it transforms nonlinear macroscopic advection into a linear transport process in the particle velocity space, coupled with a local nonlinear collision operator. This formulation is inherently hyperbolic, obviating the challenge of global pressure solvers and facilitating strictly local data propagation. The continuous Boltzmann equation employed in kinetic methods is expressed as follows:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f = \Omega(f), \tag{2}$$

where $\boldsymbol{\xi}$ is the particle velocity, and the collision operator Ω relaxes the particle distribution function $f(\mathbf{x}, \boldsymbol{\xi}, t)$ toward the local Maxwellian equilibrium distribution function g, all in continuous form. For isothermal flows, $g = g(\rho, \mathbf{u})$ depends on the macroscopic fluid density ρ and velocity $\mathbf{u}$ only, both derived from the velocity moments of f as: $\rho = \int f \, \mathrm{d}\boldsymbol{\xi}$ and $\rho \mathbf{u} = \int \boldsymbol{\xi} f \, \mathrm{d}\boldsymbol{\xi}$. The collision operator is conservative, satisfying $\int \Omega \, \mathrm{d}\boldsymbol{\xi} = 0$ and $\int \boldsymbol{\xi} \Omega \, \mathrm{d}\boldsymbol{\xi} = \mathbf{0}$. In the continuum limit, the isothermal Boltzmann equation reduces to the weakly compressible NS equations to approximate the incompressible flow equations, providing a mesoscopic foundation for fluid flow simulation. This framework has spawned two primary families of kinetic solvers: the lattice Boltzmann method (LBM), which uses a stream-and-collide algorithm, and gas-kinetic schemes (GKS), which reconstruct fluxes at element interfaces.

2.2.1 Lattice Boltzmann method. In both CFD and CG, the lattice Boltzmann method (LBM) has emerged as a well-established alternative to classical NS-based schemes, primarily owing to its high-performance, GPU-friendly computational structure [Li et al. 2003], and excellent spectral-like characteristics that enable extremely low numerical dissipation [Geier et al. 2021]. Standard LBM comprises two distinct, sequentially executed steps: a streaming step and a collision step, written as:

$$f_i(\mathbf{x} + \boldsymbol{\xi}_i, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i, \tag{3}$$

where $\boldsymbol{\xi}_i$ denotes the discretized lattice velocity, and Ω_i is the corresponding discretized collision operator that relaxes the discretized distribution functions f_i toward their local equilibrium counterparts g_i [Krüger et al. 2017]. Note that Ω_i needs to be modeled to recover NS equations, and the simplest formulation is the lattice BGK model [Bhatnagar et al. 1954; Chen and Doolen 1998], given by $\Omega_i = -(f_i - g_i)/\tau$, with relaxation time $\tau = 3\nu + 1/2$, where ν is the kinematic viscosity. Macroscopic variables are retrieved via discrete moments: $\rho = \sum_i f_i$ and $\rho\mathbf{u} = \sum_i \boldsymbol{\xi}_i f_i$. While this stream-and-collide structure is highly efficient in implementation, the widely adopted D3Q27 LBM requires storing 27 distribution functions in 3D at each lattice node, resulting in a considerable memory footprint.

Owing to the well-documented stability issues of the lattice-BGK model in simulating fluid flows at high Reynolds numbers, a range of enhanced formulations have been proposed to mitigate this challenge, from the regularized lattice-BGK model [Latt and Chopard 2006] to multiple-relaxation-time [d'Humières 2002; Lallemand and Luo 2000], central-moment [Geier et al. 2006, 2009], and more advanced cumulant [Geier et al. 2017, 2015] models. These models are designed to balance stability with minimal numerical dissipation. While the streaming process is conceptually straightforward, it imposes substantial memory demands: two full sets of distribution functions must be stored per node to avoid data write conflicts. To mitigate this overhead, various streaming optimizations have been developed [Bailey et al. 2009; Geier and Schönherr 2017]. In computer graphics, LBM has been extensively explored for high-efficiency, high-quality fluid flow simulation, spanning from early gaseous phenomena solvers [Wei et al. 2004] and GPU-accelerated implementations [Li et al. 2003], to recent turbulence-aware and detail-preserving models [Li et al. 2020a, 2023; Liu et al. 2014; Lyu et al. 2023, 2021], and two-phase extensions [Li et al. 2022; Xiao et al. 2025]. More recent work has also addressed turbulent boundary layer treatments to enhance the physical consistency of the corresponding simulations [Liu et al. 2025; Liu and Liu 2023].

The streaming step in Eq. (3) embodies a defining trade-off in LBM: per-direction non-locality yields ultra-low numerical dissipation but incurs significant memory overhead. The state-of-the-art MR-LBM [Li et al. 2023] reduces memory usage by storing moments rather than distribution functions f_i, yet streaming still requires on-the-fly reconstruction of f_i from moments at each link, increasing per-link data volume and necessitating shared memory and half-precision storage to fit within on-chip memory. Furthermore, the reconstruction step introduces additional numerical dissipation compared to the native streaming formulation.

2.2.2 Gas-kinetic scheme. As an alternative to the streaming-based formulation, we turn to *streaming-free* kinetic formulations, specifically the gas-kinetic scheme (GKS). GKS is a hybrid formulation that couples the traditional FV framework for the conservative-form NS equations with the locally-evolved Boltzmann equation [Xu 2001]. The core of the GKS resides in computing the *common flux* $\mathbf{F}^*$ at the element interface as a key component of the update rule for macroscopic conservative variables, e.g., in 1D as:

$$\mathbf{U}_i(\Delta t) = \mathbf{U}_i(0) - \frac{\Delta t}{\Delta x}\left(\mathbf{F}^*_{i+1/2} - \mathbf{F}^*_{i-1/2}\right),$$

where $\mathbf{F}^*_{i+1/2}$ denotes the flux at the element interface, accounting for the inter-element coupling. A key challenge arises when neighboring elements exhibit distinct states, introducing ambiguity regarding which state to adopt at the interface. However, naive approaches such as averaging fluxes from both sides result in unconditionally unstable schemes [Toro 2009]. By directly taking moments of Eq. (2) with respect to $\boldsymbol{\varphi}$, where $\boldsymbol{\varphi} = \left[1, \boldsymbol{\xi}^\top\right]^\top$ denotes the moment vector, it follows that the flux is related to f as $\mathbf{F} = \langle \boldsymbol{\xi}\boldsymbol{\varphi}^\top f\rangle$, in which the moment operator $\langle Af\rangle = \int Af\,\mathrm{d}\boldsymbol{\xi}$ is adopted. Using the *half-range* integration defined as $\langle Af\rangle_\pm = \int_{\xi_n \gtrless 0} Af\,\mathrm{d}\boldsymbol{\xi}$ along each side of the element interface, the flux is constructed by decomposing the contribution into terms from the distribution function to the left of the interface f_L and that to the right f_R as $\mathbf{F}^* = \langle \xi_n\boldsymbol{\varphi} f_L\rangle_+ + \langle \xi_n\boldsymbol{\varphi} f_R\rangle_-$, where ξ_n denotes the component of $\boldsymbol{\xi}$ normal to the element interface [Chou and Baganoff 1997]. A core insight of GKS is the construction of a time-evolving analytical distribution function at the element interface, which is derived from the initial states f_L and f_R (obtained via Eq. (2)) and a kinetic evolution scheme that we elaborate upon subsequently. This simulation fidelity is accompanied by increased analytical complexity and computational overhead.

Like the LBM, the original GKS [Xu 2001] is a single-stage, second-order scheme that obviates the need for Runge-Kutta time integration. However, the analytical formulation of the GKS results in flux functions that are highly complex and computationally expensive [Sun et al. 2015; Toro 2009]. To mitigate this complexity, the lattice Boltzmann flux solver (LBFS) replaces analytical integration with numerical quadrature by discretizing particle velocity [Shu et al. 2014] and incorporates Runge-Kutta time-stepping to broaden its applicability. Similarly, the gas-kinetic flux solver (GKFS) [Sun et al. 2015] and other simplified variants [Capdeville 2023; Yang et al. 2014, 2017] streamline flux calculation through a variety of approximations. For multiscale flow simulations, the unified gas-kinetic scheme (UGKS) [Xu and Huang 2010] and discrete unified gas-kinetic scheme (DUGKS) [Guo et al. 2013], analogous to the LBM, explicitly store distribution functions to capture non-equilibrium effects. Conceptually, Grad-based methods [Liu et al. 2023] replace DUGKS's discrete distribution functions with Grad's moments for rarefied flows to reduce memory overhead; though their regimes and objectives differ, as we shall demonstrate later, the concept of non-equilibrium encoding finds independent application in FR.

Despite these notable advancements, GKS and its variants within the second-order FV framework have not yet matched the efficiency and spectral characteristics of cumulant LBM for low-speed flow regimes, which form the core focus of this work, most computer graphics and many industrial design-oriented applications. The

substantial analytical complexity of the GKS, coupled with the numerical dissipation inherent to second-order FVM [Wang et al. 2013], limits the competitiveness of GKS-based frameworks for simulating low-speed, nearly incompressible flows, especially those with turbulence. Nevertheless, high-order discretization frameworks offer a natural pathway to harnessing the GKS's unified inviscid-viscous flux formulation [Ren et al. 2015], which yields key insights that motivate our focused exploration of a high-order kinetic framework.

2.3 High-order flux reconstruction

Capturing intricate vortical structures requires extremely low numerical dissipation. High-order approaches provide the foundation for low-dissipation schemes that GKS demands, by fundamentally replacing simple linear reconstruction with richer polynomial approximations, which is analogous to how bicubic interpolation preserves sharp details that would otherwise be smoothed by numerical dissipation. The update procedure of LBM in Eq. (3) achieves this efficiency effectively by conservative on-lattice streaming and collision with adaptive high-order relaxation rates in advanced collision models such as cumulant LBM, yet second-order schemes such as GKS fail to provide this level of fidelity [Wang et al. 2013], a limitation that motivates the adoption of high-order spatial discretizations.

While high-order interpolations are prevalent for Eulerian fluid dynamics, an alternative perspective emerges: high-order representation of the fluid field. In computer graphics, this modeling capability of high-order schemes has been harnessed to handle topological changes in solids [Kaufmann et al. 2009] and extended to capture sub-grid scale fluid details [Edwards and Bridson 2014]. Among the high-order approaches, the flux reconstruction (FR), proposed by Huynh [2007], stands out for its simplicity and computational efficiency [Liang et al. 2013]. FR offers a generalized framework that unifies both DG and SD schemes via parameterization of the *correction function*, enabling navigation of trade-offs between accuracy and stability, with adjustments to timestep [Vincent et al. 2011] and the achievement of spectral characteristics that exceed those of traditional schemes [Asthana and Jameson 2015].

To accommodate viscous flows involving second-order spatial derivatives, the original FR was subsequently extended by Huynh [2009], which encompasses conventional schemes such as the Bassi-Rebay (BR1 and BR2) methods [Bassi and Rebay 1997, 2000] and the local discontinuous Galerkin (LDG) method [Cockburn and Shu 1998]. All these approaches share a structural commonality: viscous flux computation relies on an ad hoc procedure to ensure inter-element consistency of second-order derivatives [Huynh 2009]. While mathematically rigorous, these formulations, however, augment algorithmic complexity and communication overhead [Zhang et al. 2018]. Alternative approaches include the interior penalty (IP) method [Arnold et al. 2002], which is relatively straightforward to implement but requires a stabilization penalty dependent on the polynomial degree and mesh parameters, as well as the recovery-based DG (RDG) method [Van Leer and Nomura 2005], which employs patch recovery, resulting in a wider stencil and additional computational overhead.

The kinetic formulation provides a powerful alternative for resolving the aforementioned challenges. The unified handling of inviscid and viscous fluxes in GKS [Xu 2004] and its inherent kinetic penalty obviates the need for auxiliary gradient treatments, enabling a unified kinetic formulation [Xu 2004]. Several GKS-FR frameworks have been proposed to leverage this advantage. Early implementations [Ren et al. 2015; Xu 2004; Zhang et al. 2018] inherit the single-stage structure and unified inviscid-viscous treatment of classical GKS, yet retain its analytical complexity, resulting in costly flux evaluation. To alleviate this computational burden, simplified variants introduce deliberate trade-offs: the kinetic inviscid flux method [Li et al. 2020b] enables lightweight flux computation by replacing unified viscous treatment with a BR1-like auxiliary procedure. Meanwhile, approaches based on LBFS [Ma et al. 2022] achieve conceptual and implementation simplicity at the expense of high arithmetic intensity and multi-stage Runge-Kutta time integration.

In summary, these approaches reveal a recurring pattern: each method adopts deliberate trade-offs within the design space. Retaining the original GKS incurs computationally expensive flux evaluation in the low-speed regime; employing multi-stage time-stepping increases wall-clock time; and decoupling the unified viscous treatment reintroduces communication overhead from auxiliary gradient computations. For a practical kinetic flow solver aiming to simultaneously achieve high accuracy, extreme throughput, and low memory footprint, all three aspects must be co-optimized, a goal not yet attained by existing kinetic FR schemes.

2.4 Our contributions

In this paper, we propose Kinetic Predicted-Moment Flux Reconstruction (KPM-FR), a novel high-order kinetic scheme tailored to co-optimize accuracy, throughput, and memory footprint for high-performance fluid simulations. Our contributions are threefold:

- *A moment-based spatially high-order kinetic-based framework.* This framework unifies the spectral accuracy of high-order flux reconstruction with the unified inviscid-viscous treatment of kinetic schemes, specifically tailored for low-Mach-number flows. Our key innovation is an *encode-transmit-decode* pipeline that operates entirely in moment space to minimize GPU memory traffic, while streamlining flux reconstruction into a sequence of highly efficient algebraic operations.
- *A hardware-aware predictor-corrector architecture.* We realize this framework through a lightweight predictor-corrector pair tailored for tensor-product FR on modern GPUs, fusing core operations into two kernels. This design leverages the hierarchy of on-chip memory to execute the single-stage explicit update, along with mixed-precision storage to saturate GPU memory bandwidth for high solution-point throughput.
- *State-of-the-art performance and scalability.* We demonstrate favorable time-to-solution compared with state-of-the-art LBM solvers at equivalent error levels, while achieving over 8 billion solution-point updates per second on a single consumer GPU and reducing the memory footprint to 42 bytes per point. We further demonstrate the method's versatility through comprehensive applications encompassing low-dissipation to high-throughput scenarios, scaling from million- to billion-point simulations on a single GPU.

3 Kinetic Predicted-Moment Flux Reconstruction

We formulate KPM-FR for isothermal Navier-Stokes (NS) flows: gas-kinetic evolution derives macroscopic numerical fluxes directly from kinetic theory, and flux reconstruction (FR) discretizes the resulting conservation law with high-order spatial accuracy.

3.1 Method overview

KPM-FR solves a kinetically derived conservation law (Section 3.2). The conventional approach combines finite-volume discretization with a multi-stage Runge-Kutta integrator. KPM-FR improves upon both components. In space, it employs FR (Section 3.3), a high-order element-based method that, similar to finite-volume method (FVM), couples elements via macroscopic fluxes at shared interfaces. Within each element, it uses K solution points per coordinate direction, achieving spectral-like resolution without mesh refinement. In time, it replaces multi-stage integration with a single-stage update (Section 3.4), halving the inter-element data exchanges per time step, and the associated off-chip memory traffic.

Building on these foundations, a central challenge persists for all flux-based kinetic solvers: *how to efficiently compute the kinetically derived macroscopic flux?* (detailed in Section 3.5). A direct implementation incurs two compounding overheads: (i) gradient tensors must be retrieved from neighboring elements via global memory accesses, and (ii) coupling temporal evolution with flux evaluation leads to analytically intricate expressions requiring special functions or numerical quadrature. As illustrated in Fig. 2, KPM-FR addresses both issues by decomposing flux evaluation into a modular predictor-corrector cycle centered on *predicted moments*, which are compact scalars generated by the predictor and utilized by the corrector. This simultaneously eliminates gradient data transfers through compact moment exchanges (Section 3.6) and analytical complexity via lightweight arithmetic operations on pre-integrated moments (Section 3.7), with a concise implementation recipe provided in Section 3.8. The resulting two-stage structure is well adapted to GPU architectures: each stage maps to a single fused kernel where all computations including FR differentiation are executed on-chip, preventing gradient data from spilling to DRAM (see Section 4), thereby significantly boosting computational throughput. Each component is described sequentially in the subsections that follow.

3.2 From Boltzmann to Navier-Stokes equations

The governing equation discretized by the FR scheme is derived from a single kinetic equation. KPM-FR models fluid flow at the mesoscopic level through the continuous Boltzmann equation incorporating the Bhatnagar-Gross-Krook (BGK) collision operator [Bhatnagar et al. 1954], which is valid for dense fluids, and is written as:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f = -\frac{1}{\tau}(f - g), \tag{4}$$

where $f = f(\mathbf{x}, \boldsymbol{\xi}, t)$ denotes the continuous particle distribution function, representing the density of microscopic particles at position $\mathbf{x}$ traveling with velocity $\boldsymbol{\xi}$ at time t. The relaxation time τ defines the timescale over which particle collisions drive f toward the local Maxwell-Boltzmann equilibrium distribution g (see Section B); it is related to the kinematic viscosity by $\tau = \nu/(RT)$, where R is the specific gas constant and T the temperature, held constant at $T = T_0$ in the isothermal model. For isothermal flows, $g = g(\rho, \mathbf{u})$ therefore depends solely on the density ρ and velocity $\mathbf{u}$.

The connection between the aforementioned kinetic model and macroscopic fluid dynamics is established via velocity-space moments. Defining the moment vector $\boldsymbol{\varphi} = \left[1, \boldsymbol{\xi}^\top\right]^\top$, the conserved macroscopic variables and their fluxes are recovered as:

$$\mathbf{U} = \langle \boldsymbol{\varphi} f \rangle, \qquad \mathbf{F}_d = \langle \xi_d \boldsymbol{\varphi} f \rangle. \tag{5}$$

The angle brackets $\langle \cdot \rangle$ denote velocity-space integration, and ξ_d represents the d-th component of $\boldsymbol{\xi}$ for $d \in \{x, y, z\}$. Accordingly, $\mathbf{U} = \left[\rho, \rho\mathbf{u}^\top\right]^\top$ combines density and momentum, while $\mathbf{F}_d$ denotes the flux along direction d. Since the BGK collision model locally conserves mass and momentum, multiplying Eq. (4) by $\boldsymbol{\varphi}$ and integrating over $\boldsymbol{\xi}$ yields the following conservative governing equation:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = \mathbf{0}. \tag{6}$$

Here, the flux tensor $\mathbf{F}$, assembled from directional fluxes $\mathbf{F}_d$, incorporates both inviscid and viscous contributions within a single kinetic expression. Eq. (6) recovers the isothermal, weakly compressible NS equations [Krüger et al. 2017], which approximate incompressible flow at low Mach numbers. The following subsection details the spatial discretization of Eq. (6) via the high-order FR scheme.

3.3 Spatial discretization by flux reconstruction

Flux reconstruction [Huynh 2007] discretizes the conservation law (Eq. (6)) by first representing the solution within each element as a degree-P polynomial defined by its values at $K = P + 1$ solution points. Similar to high-order interpolation that extracts finer variations from a fixed set of control points, this high-resolution per-element representation captures finer-scale flow features without extensive grid/mesh refinement, which low-order schemes tend to smooth out. Elements communicate only through macroscopic numerical fluxes at shared interfaces; all other computations act locally on the element's K solution point values via small, dense matrix operations. We detail these components below.

Polynomial representation. Consider a single 1D element with local coordinate $r \in [-1, 1]$. The solution within the element is represented continuously by a polynomial of degree $P = K - 1$, interpolated from its values at K *solution points* (SPs):

$$\mathbf{U}(r) = \textstyle\sum_{j=1}^{K} \mathbf{U}_j \, \ell_j(r),$$

where ℓ_j denote Lagrange basis polynomials satisfying $\ell_j(r_i) = \delta_{ij}$ (Fig. 3). In this work, the SPs $\{r_i\}_{i=1}^K$ are chosen as Gauss-Lobatto points, which include the endpoints $r = \pm 1$ [Huynh 2007]. The flux is interpolated analogously as $\mathbf{F}(r) = \sum_j \mathbf{F}_j \, \ell_j(r)$ with $\mathbf{F}_j = \mathbf{F}(\mathbf{U}_j)$. Differentiating $\mathbf{F}(r)$ at the SPs and substituting into Eq. (6) yields:

$$\frac{\partial \mathbf{U}}{\partial t} = -\frac{2}{L}\mathbf{D}\mathbf{F}, \qquad D_{ij} = \frac{\partial \ell_j(r_i)}{\partial r}, \tag{7}$$

where $\mathbf{U}$ and $\mathbf{F}$ denote block vectors assembling the solution and flux at all K SPs, with each block containing C conserved-variable components; L is the element length, and $\mathbf{D}$ is a $K \times K$ differentiation matrix. Note that this equation governs intra-element evolution only but does not couple neighboring elements.

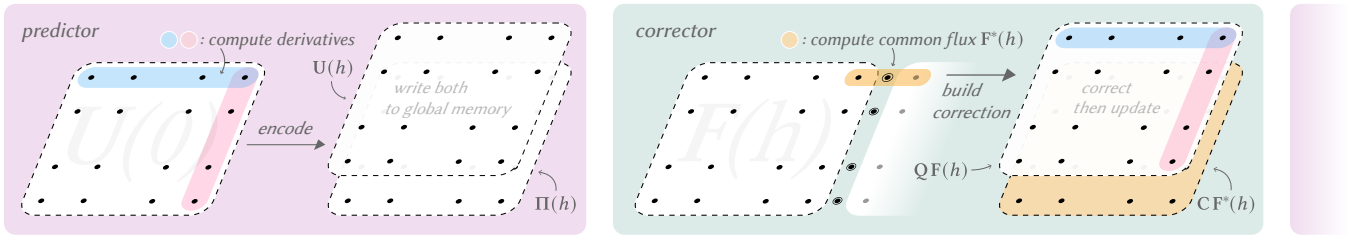


Fig. 2. **Schematic of the KPM-FR scheme.** The overall algorithmic framework follows a predictor-corrector cycle. The predictor (left) computes local spatial derivatives and encodes the resulting physical state into compact moments, which are stored in global memory. The corrector (right) retrieves these moments to evaluate the element-local flux $\mathbf{F}(h)$ and common flux $\mathbf{F}^*(h)$ in closed form, then applies the solution correction to advance the macroscopic variables. This moment-only data path reduces inter-element bandwidth and memory footprint while simplifying the kinetic flux expressions.

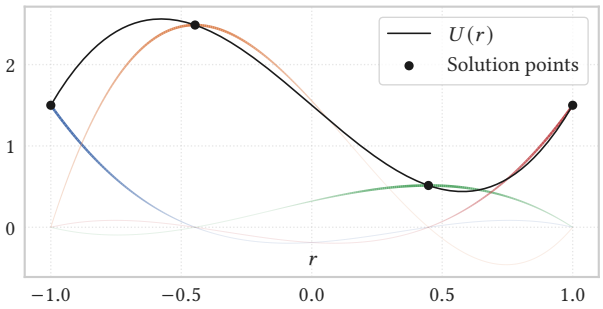


Fig. 3. **Lagrange polynomial basis in 1D.** We show an example in 1D of the weighted basis functions $U_j \ell_j(r)$ and the reconstructed degree-$(K-1)$ polynomial at $K = 4$ Gauss-Lobatto solution points.

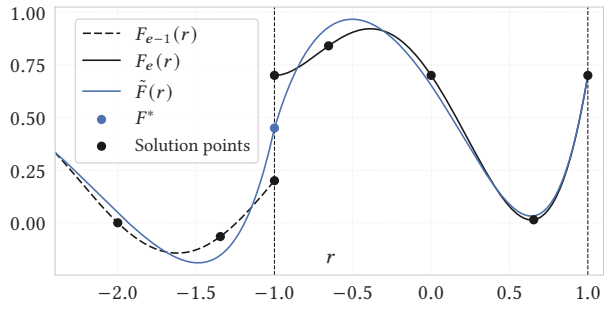


Fig. 4. **Continuous flux reconstruction.** The discontinuous element-local flux $F(r)$ is corrected at each element interface to match the common flux F^*, yielding a globally continuous corrected flux $\tilde{F}(r)$.

Inter-element coupling. As each element forms its polynomial approximation *independently*, the flux is typically discontinuous across element interfaces. FR addresses this by computing a *common flux* $\mathbf{F}^*$ at each interface from both neighboring elements that enforces global flux continuity. Analogous to the numerical flux evaluation in FVM, this is usually derived using a Riemann solver [Toro 2009]. Specifically in FR, two *correction functions* $g_{LB}(r)$ and $g_{RB}(r)$ of degree K (one degree higher) are introduced for the left ($r = -1$) and right ($r = +1$) element boundaries, respectively. They satisfy $g_{LB}(-1) = g_{RB}(1) = 1$, $g_{LB}(1) = g_{RB}(-1) = 0$, and the symmetry condition $g_{LB}(r) = g_{RB}(-r)$. These functions blend the common flux $\mathbf{F}^*$ into the interior of elements, generating a globally continuous flux $\tilde{\mathbf{F}}(r) = \mathbf{F}(r) + g_{LB}(r)\left[\mathbf{F}^*_{LB} - \mathbf{F}(-1)\right] + g_{RB}(r)\left[\mathbf{F}^*_{RB} - \mathbf{F}(1)\right]$, as in Fig. 4. Differentiating $\tilde{\mathbf{F}}$ at the SPs yields the semi-discrete system:

$$\frac{\partial \mathbf{U}}{\partial t} = -\frac{2}{L}(\mathbf{QF} + \mathbf{CF}^*), \quad \mathbf{Q} = \mathbf{D} - \mathbf{CR}, \quad \mathbf{F}^* = \begin{bmatrix} \mathbf{F}^*_{LB} \\ \mathbf{F}^*_{RB} \end{bmatrix}, \tag{8}$$

where $\mathbf{F}^*_{LB}$ and $\mathbf{F}^*_{RB}$ denote the C-component common fluxes at the left and right element boundaries, $\mathbf{R}$ is a $2 \times K$ extrapolation matrix evaluating the polynomial at the element boundaries, and $\mathbf{C}$ is a $K \times 2$ correction matrix with entries corresponding to the derivatives of the correction functions at the SPs:

$$\mathbf{R}^\top = \begin{bmatrix} \ell_1(-1) & \ell_1(1) \\ \ell_2(-1) & \ell_2(1) \\ \vdots & \vdots \\ \ell_K(-1) & \ell_K(1) \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} g'_{LB}(r_1) & g'_{RB}(r_1) \\ g'_{LB}(r_2) & g'_{RB}(r_2) \\ \vdots & \vdots \\ g'_{LB}(r_K) & g'_{RB}(r_K) \end{bmatrix}. \tag{9}$$

The choice of correction function governs the properties of the scheme [Huynh 2007]. In this work, we set $g_{LB} = g_2$, with the correction function g_2 from Huynh [2007] (distinct from the equilibrium distribution g in Section 3.2), and $g_{RB}(r) = g_2(-r)$ by symmetry; other choices recover nodal DG and related spectral-element methods [Vincent et al. 2011]. For Gauss-Lobatto SPs, g_2 yields a highly sparse correction matrix: $g'_{LB}(r_1) = -K(K-1)/2$ and $g'_{LB}(r_i) = 0$ for $i \geq 2$, with right-boundary derivatives following symmetry. Note that matrices $\mathbf{D}$, $\mathbf{R}$, $\mathbf{C}$, and $\mathbf{Q}$ are globally constant.

Extension to multiple dimensions. On hexahedral (3D) or quadrilateral (2D) elements, SPs are arranged on a tensor-product grid, with K^2 points in 2D and K^3 in 3D. As each coordinate direction acts independently, the multi-dimensional update decomposes into 1D FR subproblems: in 2D, the K^2 points form $2K$ independent lines of K points along each axis, and directional contributions are accumulated at each solution point. On axis-aligned grids, the common flux $\mathbf{F}^*$ approximates the physical normal flux $\mathbf{F}(\mathbf{U}) \cdot \mathbf{n}$, where the normal $\mathbf{n}$ points along the positive coordinate axis direction. Unstructured meshes require only a per-element Jacobian mapping, after which

all operations proceed identically. The details are presented in the supplementary material (Sec. 3).

Inviscid limitation. Conventionally, $\mathbf{F}$ and $\mathbf{F}^*$ are computed solely from inviscid formulations, and extending this scheme to viscous flows requires non-trivial, computationally expensive auxiliary treatments [Huynh 2009]. The kinetic approach developed in the remainder of this work provides both quantities through a single element-local evolution that naturally incorporates both inviscid and viscous effects within a unified computational framework, remedying the aforementioned drawbacks of conventional FR schemes.

3.4 Single-stage temporal integration

Advancing $\mathbf{U}$ in time requires integrating the semi-discrete system in Eq. (8) over each time step. As demonstrated below, both $\mathbf{F}$ and $\mathbf{F}^*$ can be evaluated in closed form at the half-time step $h = \Delta t/2$, yielding a single-stage fully discrete update:

$$\mathbf{U}^{(n+1)} = \mathbf{U}^{(n)} - \frac{2\Delta t}{L}\big(\mathbf{Q}\mathbf{F} + \mathbf{C}\mathbf{F}^*\big), \tag{10}$$

where $\mathbf{U}^{(n)}$ and $\mathbf{U}^{(n+1)}$ denote the block vectors of K nodal conservative states at consecutive time steps, and $\mathbf{F}$, $\mathbf{F}^*$ represent the corresponding element-local and common fluxes evaluated at h. We write the update for one coordinate direction and, in 2D or 3D, sum the corresponding directional contributions at each solution point (Section 3.3). The common flux $\mathbf{F}^*$, which governs all inter-element coupling, is computed only once, whereas a second-order Runge-Kutta integrator for Eq. (8) would demand at least two evaluations. This midpoint choice achieves second-order temporal accuracy; higher-order schemes are available [Gassner et al. 2011], but at increased computational cost. With the spatial discretization and single-stage temporal integration defined, only the evaluation of $\mathbf{F}$ and $\mathbf{F}^*$ at the half-time step remains. In the subsequent sections, we develop a kinetic framework operating at each solution point; we thus drop the directional subscript d and use $\mathbf{F}(t)$ and $\mathbf{F}^*(t)$ to denote the per-point element-local and common fluxes along a single coordinate direction d at time t, with superscripts reserved for block vectors as in Eq. (10).

3.5 Kinetic flux evaluation

The single-stage update (Eq. (10)) requires evaluating the element-local flux $\mathbf{F}(h)$ and the common flux $\mathbf{F}^*(h)$ at the half-time level $h = \Delta t/2$. Both are velocity-space moments of their corresponding distribution function (Eq. (5)), reducing the task to solving Eq. (4) over the half-step and computing moments of the solution. Recall that $\mathbf{U}(t)$ and $\mathbf{F}(t)$ now denote per-point quantities (Section 3.4). We show below that $\mathbf{F}(h)$ admits a compact expression locally, while $\mathbf{F}^*(h)$ introduces an inter-element data dependency that motivates the moment-based encoding developed in Section 3.6.

Half-time distribution function. Our initial objective is to express $f(h)$ entirely in terms of $\mathbf{U}(0)$ and its spatial gradient. Integrating Eq. (4) over $[0, h]$ using a finite-difference [Shu et al. 2014] yields an explicit approximation for the half-time distribution function:

$$f(h) \approx A\, f^s(h) + B\, g(h), \quad A = \frac{\tau}{h}, \quad B = 1 - \frac{\tau}{h}, \tag{11}$$

where $f^s(h) = g(0) - h\,\boldsymbol{\xi}\cdot\nabla g(0)$, the *propagated distribution*, is the first-order Taylor expansion of g along the direction of $\boldsymbol{\xi}$. Here, $g(h) = g\,(\mathbf{U}(h))$ denotes the equilibrium distribution corresponding to the half-time macroscopic state. Dependencies on $\mathbf{x}$ and $\boldsymbol{\xi}$ are omitted for brevity; the relation holds locally at every spatial point $\mathbf{x}$, with τ defined as in Section 3.2. The only remaining unknown is $\mathbf{U}(h)$, which governs $g(h)$. This term is determined by imposing the compatibility condition, yielding $\mathbf{U}(h) = \langle \boldsymbol{\varphi}\,[g(0) - h\,\boldsymbol{\xi}\cdot\nabla g(0)]\rangle$, where $g(0)$ and $\nabla g(0)$ are directly provided by the FR discretization.

Per-point flux definitions. With $f(h)$ determined, the element-local and common fluxes in Eq. (10) follow directly from the moment relation in Eq. (5). At each solution point,

$$\mathbf{F}(h) = \langle \xi_d\, \boldsymbol{\varphi}\, f(h)\rangle\,, \tag{12}$$

where $f(h)$ is obtained from Eq. (11) evaluated at the corresponding point. At each axis-aligned element interface with unit normal $\mathbf{n}$ oriented along the positive coordinate direction ($\boldsymbol{\xi}\cdot\mathbf{n} = \xi_d$),

$$\mathbf{F}^*(h) = \langle \xi_d\, \boldsymbol{\varphi}\, f^*(h)\rangle\,, \tag{13}$$

where f^* denotes a *common distribution function* constructed from both adjacent elements in the subsequent development.

Construction of f^*. The common distribution function adopts the same form as Eq. (11) [Xu 2001]. At the interface, each adjacent element first reconstructs its own half-time distribution, denoted by $f_L(h)$ and $f_R(h)$, from Eq. (11). The common state is approximated from the half-range moments of these full distributions,

$$\mathbf{U}^*(h) = \langle \boldsymbol{\varphi}\, f_L(h)\rangle_+ + \langle \boldsymbol{\varphi}\, f_R(h)\rangle_-\,, \tag{14}$$

after which the propagated part in f^* is selected upwind from the originating element:

$$f^*(h) \approx B\, g\,(\mathbf{U}^*(h)) + A \begin{cases} f_L^s(h) & \text{if } \boldsymbol{\xi}\cdot\mathbf{n} > 0, \\ f_R^s(h) & \text{if } \boldsymbol{\xi}\cdot\mathbf{n} < 0, \end{cases} \tag{15}$$

where $\langle\cdot\rangle_\pm$ denotes integration over the half-spaces $\boldsymbol{\xi}\cdot\mathbf{n} \gtrless 0$. Eqs. (11) and (15) fully specify the half-time fluxes required for the single-stage temporal update, yet two challenges hinder efficient implementation. First, velocity-space integrals, especially half-range ones, lead to lengthy analytical expressions whose direct evaluation is computationally expensive [Sun et al. 2015]. Second, each sidewise half-time distribution relies on the spatial gradient ∇g (and thus $\nabla\mathbf{U}$) from the element that computed it, requiring the data entering $\mathbf{U}^*(h)$ and $\mathbf{F}^*(h)$ to be communicated across element boundaries, increasing inter-element data traffic. Subsequent sections resolve both issues via a moment-based encoding.

3.6 Moment-based prediction and correction

Consider the data each element requires to perform the single-stage temporal update in Eq. (10). As elements serve as the basic unit of parallel execution in our implementation, the key concern is which data must be exchanged between neighboring elements. The element-local flux $\mathbf{F}(h)$ is self-contained: Eq. (11) relies solely on $g(0)$ and $\nabla g(0)$, both available at each solution point. The common flux $\mathbf{F}^*(h)$, however, requires ∇g from the adjacent element, amounting to sixteen scalars per solution point in 3D—far in excess of the compact moment representation developed below. Since

inter-element data must be written by the predictor and read by the corrector across element boundaries, these values must pass through global DRAM for communication between kernels of adjacent elements; each scalar in the payload increases both global memory footprint and bandwidth consumption. The element interface thus acts as a bandwidth-limited channel, and efficient implementation requires minimizing its transmission payload. As shown below, the kinetic framework admits a natural compact encoding that reduces inter-element data transfer to its physically necessary minimum, eliminating gradient traffic through global memory when combined with the fused-kernel implementation design (Section 4).

Evolution-state identity. To evaluate $\mathbf{F}^*(h)$, each element requires the propagated distribution f^s from its neighbor, but not the raw gradient ∇g from which f^s was built; this distinction opens the door to a more compact encoding. In kinetic theory, any distribution function can be decomposed into an equilibrium component g, fully determined by $\mathbf{U}$, and a non-equilibrium deviation f^{neq} that carries viscous information. This decomposition reveals a duality: the half-time distribution function admits both an *evolution* form, dependent on element-local gradients of macroscopic variables, and a *state* form, relying solely on macroscopic quantities and f^{neq}:

$$\underbrace{Af^s(h) + Bg(h)}_{\text{evolution (requires local gradients)}} = f(h) = \underbrace{g(h) + f^{\text{neq}}(h)}_{\text{state (compact representation)}}, \quad (16)$$

Since $A + B = 1$ (Eq. (11)), each element can evaluate the evolution form from its own data and extract $f^{\text{neq}}(h) = A\left[f^s(h) - g(h)\right]$. Elements then recover $Af^s(h) = Ag(h) + f^{\text{neq}}(h)$ from their neighbors, bypassing $\nabla\mathbf{U}$ entirely, provided f^{neq} can be compactly encoded.

Encoding of f^{neq}. Recall that f^{neq}, in its full mathematical form, is a function of the particle velocity $\boldsymbol{\xi}$, and thus cannot be directly transmitted. For the isothermal model, where the flux depends only on second-order velocity moments, all flux-relevant information in f^{neq} is encapsulated by a single symmetric tensor $\boldsymbol{\Pi}$, physically related to the viscous stress. Grad's closure [Grad 1949] yields an explicit velocity-space expression of f^{neq} as:

$$f^{\text{neq}} \approx \frac{1}{(2\pi RT)^{D/2}} \exp\left(-\frac{\|\boldsymbol{\xi}\|^2}{2RT}\right) \frac{\Pi_{\alpha\beta}\left(\xi_\alpha \xi_\beta - RT\,\delta_{\alpha\beta}\right)}{2(RT)^2}, \quad (17)$$

where $\alpha, \beta \in \{x, y, z\}$ are spatial component indices with repeated indices denoting summation, $\delta_{\alpha\beta}$ is the Kronecker delta, and $\boldsymbol{\Pi}$ is in turn defined as the second-order moment of f^{neq}:

$$\Pi_{\alpha\beta} = \left\langle \xi_\alpha\, \xi_\beta\, f^{\text{neq}} \right\rangle. \quad (18)$$

Substituting Eq. (17) into Eq. (18) recovers $\Pi_{\alpha\beta}$ exactly, making the encode-decode cycle lossless for the second-order moments that govern the continuum flux. As $\boldsymbol{\Pi}$ is symmetric, its nine components reduce to six independent values; for low-Mach-number flows, $\text{tr}(\boldsymbol{\Pi}) \approx 0$, eliminating one further component and leaving five independent values to be produced by the predictor.

Moment-based predictor. With this encoding, the predictor produces the following half-time quantities at each solution point from element-local data (see Fig. 2):

$$\mathbf{U}(h) = \langle \boldsymbol{\varphi}\, f^s(h) \rangle, \quad \Pi_{\alpha\beta}(h) = A\left\langle \xi_\alpha\, \xi_\beta \left[f^s(h) - g(h)\right] \right\rangle, \quad (19)$$

where $f^s(h)$ denotes the propagated distribution function from Eq. (11). Together, $\mathbf{U}(h)$ and $\boldsymbol{\Pi}(h)$ form the predicted moments in the predictor stage: nine scalar values per point (four conserved variables plus five independent stress components), reduced from the sixteen required by the gradient-based approach. Only these quantities are exchanged across element interfaces.

Moment-based corrector. Given the predicted moments from both the local element and the neighboring elements, the corrector can compute the half-time fluxes without requiring any spatial gradients. First, the element-local flux at each solution point is expressed as:

$$\mathbf{F}(h) = \left\langle \xi_d\, \boldsymbol{\varphi} \left[g\left(\mathbf{U}(h)\right) + f^{\text{neq}}\left(\boldsymbol{\Pi}(h)\right) \right] \right\rangle, \quad (20)$$

where f^{neq} is reconstructed from $\boldsymbol{\Pi}(h)$ using Grad's closure (Eq. (17)). Second, the common flux $\mathbf{F}^*(h) = \langle \xi_d\, \boldsymbol{\varphi}\, f^*(h) \rangle$ simplifies similarly at the interface: applying the evolution-state identity (Eq. (16)) rewrites each Af^s term in Eq. (15) as $Ag + f^{\text{neq}}$ and each sidewise half-time distribution f to compute Eq. (26) as $g + f^{\text{neq}}$, so that f^* depends only on the predicted moments carried by the neighboring elements. With $\mathbf{F}(h)$ and $\mathbf{F}^*(h)$ obtained, the single-stage temporal update (Eq. (10)) is abstractly finalized; only the closed-form evaluation of these velocity-space moment integrals remains.

3.7 Efficient quadrature-free flux evaluation

Note that the moment integrals in Eqs. (13), (19) and (20) are intended to be computed using simple arithmetic operations for high efficiency. A standard approach in LBM is Gauss-Hermite quadrature, replacing each moment integral $\langle \cdots \rangle$ with a weighted sum $\sum_{i=1}^{N_q} w_i\, (\cdots)$ over N_q discrete abscissae. This discretization introduces a computational bottleneck, however: extensive loop unrolling for the N_q-point summation creates significant register pressure and arithmetic overhead, and the summation itself proves suboptimal. Consider the equilibrium moments $m_{abc} = \left\langle \xi_x^a\, \xi_y^b\, \xi_z^c\, g \right\rangle$. The distribution function g is a Gaussian weight multiplied by a polynomial (Section B), so each m_{abc} admits a simple closed-form expression; replacing this with a discrete N_q-point summation is clearly inefficient. The question is whether this simplicity carries over to the actual predictor and corrector expressions, which also involve spatial gradients and half-range integrals of the non-equilibrium distribution function f^{neq}. This is indeed the case, and the remainder of this section derives the explicit 3D formulations. Throughout, $d \in \{x, y, z\}$ denotes the tensor-product pass (cf. algorithms 1 and 2), while $\alpha, \beta \in \{x, y, z\}$ are free tensor component indices labeling physical quantities (e.g., u_α, $\Pi_{\alpha\beta}$); $\mathbf{e}_0$, $\mathbf{e}_x$, $\mathbf{e}_y$, $\mathbf{e}_z$ denote multi-indices for moment subscripts (e.g., $m_{\mathbf{e}_x + \mathbf{e}_y} \equiv m_{110}$).

Predicted moments. We first address Eq. (19). The conserved-variable moments $\langle \boldsymbol{\varphi} g \rangle$ are equilibrium moments with closed-form evaluations; the non-trivial contribution comes from the gradient term $\boldsymbol{\xi} \cdot \nabla g$ in $f^s(h) = g(0) - h\, \boldsymbol{\xi} \cdot \nabla g(0)$ (Eq. (11)), which couples spatial derivatives to velocity-space integration. A forward-difference linearization decouples these terms. For each $d \in \{x, y, z\}$, we introduce a shifted equilibrium state:

$$\mathbf{U}_d^{\triangle} = \mathbf{U} - u_c\, h\, \nabla_d \mathbf{U}, \qquad u_c = \epsilon\sqrt{RT}, \quad (21)$$

such that $\nabla_d g \approx \left[g(\mathbf{U}) - g(\mathbf{U}_d^{\triangle})\right]/(u_c\, h)$, where $u_c = \epsilon\sqrt{RT}$ denotes the characteristic velocity used to ensure dimensional consistency. We hereafter fix $\epsilon = 0.1$, aligning u_c with the standard low-Mach limit employed in weakly compressible formulations. Because $g(\mathbf{U}_d^{\triangle})$ is again an equilibrium at a known macroscopic state, its moments $m_{abc}(\mathbf{U}_d^{\triangle})$ are available in closed form, so the moment-only path remains open. Define the spatial moment difference $\Delta_d m_{abc} := [m_{abc}(\mathbf{U}_d^{\triangle}) - m_{abc}]/u_c$ for each pass direction d. The temporal counterpart is $\Delta_t m_{abc} := m_{abc}(\mathbf{U}(h)) - m_{abc}$. The predicted conserved variables then become

$$U_\alpha(h) = m_{\mathbf{e}_\alpha} + \sum_d \Delta_d m_{\mathbf{e}_\alpha + \mathbf{e}_d}, \tag{22}$$

where $\alpha \in \{0, x, y, z\}$, and the predicted stress follow analogously:

$$\Pi_{\alpha\beta}(h) = A\left(-\Delta_t m_{\mathbf{e}_\alpha+\mathbf{e}_\beta} + \sum_d \Delta_d m_{\mathbf{e}_\alpha+\mathbf{e}_\beta+\mathbf{e}_d}\right), \tag{23}$$

where $\alpha, \beta \in \{x, y, z\}$ denote tensor component indices, d sums over all spatial directions, and the traceless constraint is enforced by subtracting $\mathrm{tr}(\mathbf{\Pi})/3$ from each diagonal component; off-diagonal symmetry yields $\Pi_{\alpha\beta} = \Pi_{\beta\alpha}$. As established in Section 3.3, the multi-dimensional temporal update decomposes into 1D subproblems along each coordinate direction. Note that by linearity of the moment integrals, each pass d contributes an increment to the predicted moments, and their accumulation across passes is exact (illustrated later in algorithm 1). Every term is a linear combination of equilibrium moments evaluated at known states, with no velocity-space summation involved.

Element-local flux. The integrand in Eq. (20) splits into equilibrium (Euler) and non-equilibrium (viscous) contributions. The Euler component reduces to the standard kinetic flux expressed via $m_{abc}(\mathbf{U}(h))$. For the viscous term, Grad's closure (Eq. (17)) directly maps each component of $\langle \xi_d\, \boldsymbol{\varphi}\, f^{\mathrm{neq}}\rangle$ to the corresponding entry of $\mathbf{\Pi}$. Evaluating Eq. (20) for each coordinate direction d, $\mathbf{F}(h)$ yields each column of the flux tensor at a solution point as:

$$\begin{bmatrix} \rho u & \rho v & \rho w \\ \Pi_{xx} + \rho u^2 + p & \Pi_{xy} + \rho uv & \Pi_{xz} + \rho uw \\ \Pi_{xy} + \rho uv & \Pi_{yy} + \rho v^2 + p & \Pi_{yz} + \rho vw \\ \Pi_{xz} + \rho uw & \Pi_{yz} + \rho vw & \Pi_{zz} + \rho w^2 + p \end{bmatrix}, \tag{24}$$

where $p = \rho RT$ with all quantities evaluated at the half-step state. This recovers the familiar NS-form flux with the viscous stress $\mathbf{\Pi}$ appearing explicitly, with no remaining velocity-space integration. The clean separation between Euler and viscous terms also enables the split-form anti-aliasing strategy used in Section 5.1.

Common flux. Finally, the common flux (Eq. (13)) involves half-range rather than full-range integrals. Half-range integration is usually the most intricate step in kinetic flux solvers [Guo et al. 2008], yet Grad's closure renders it tractable: the half-range moments of f^{neq} are linear in $\mathbf{\Pi}$, which itself is already pre-integrated in the predictor. Without loss of generality, let $\xi_d = \xi_x$; other directions follow by rotation invariance [Toro 2009]. Upon substituting Eq. (15) into Eq. (13), the common flux decomposes as:

$$\mathbf{F}^*(h) = \bar{\mathbf{F}}(h) + B\left[m^*_{100}, m^*_{200}, m^*_{110}, m^*_{101}\right]^\top, \tag{25}$$

where $m^*_{abc} = m_{abc}(\mathbf{U}^*(h))$ and $\bar{\mathbf{F}}(h)$ is the half-range contribution from the propagated part f^s. Recall the half-range integration notation $\langle\chi\rangle_\pm = \int_{\xi_x \gtrless 0} \chi\, \mathrm{d}\boldsymbol{\xi}$, integrating over the half-space of positive or negative normal velocity ξ_x. Analogously to the predictor, define $m^\pm_{abc,S} = \left\langle \xi_x^a \xi_y^b \xi_z^c\, g_S(h)\right\rangle_\pm$ and $M^\pm_{abc,S} = \left\langle \xi_x^a \xi_y^b \xi_z^c\, f_S^{\mathrm{neq}}(h)\right\rangle_\pm$ as the half-range equilibrium and non-equilibrium moments of $g_S(h)$ and $f_S^{\mathrm{neq}}(h)$, which are constructed from the predicted moments (transformed to local coordinates) on side $S \in \{L, R\}$, with closed-form values listed in Section B. The common state (Eq. (14)) evaluates as:

$$U^*_\alpha(h) = m^+_{\mathbf{e}_\alpha,L} + M^+_{\mathbf{e}_\alpha,L} + m^-_{\mathbf{e}_\alpha,R} + M^-_{\mathbf{e}_\alpha,R}, \tag{26}$$

where $\alpha \in \{0, x, y, z\}$; these quantities constitute $\mathbf{U}^*(h)$ at the interface. The half-range flux $\bar{\mathbf{F}}(h)$ follows the same decomposition into equilibrium and non-equilibrium half-range moments:

$$\bar{F}_\alpha(h) = A\left(m^+_{\mathbf{e}_x+\mathbf{e}_\alpha,L} + m^-_{\mathbf{e}_x+\mathbf{e}_\alpha,R}\right) + M^+_{\mathbf{e}_x+\mathbf{e}_\alpha,L} + M^-_{\mathbf{e}_x+\mathbf{e}_\alpha,R}, \tag{27}$$

where $\alpha \in \{0, x, y, z\}$. The common flux is then obtained by substituting Eqs. (26) and (27) into Eq. (25). This moment-only formulation eliminates loop unrolling and special-function evaluations that limit quadrature-based methods. All flux components, including half-range terms, are computed using only basic arithmetic operations. Further enhancements are described in Section 5.1. In the pure-diffusion limit, this framework recovers a structure analogous to the diffusive generalized Riemann problem employed in interior penalty schemes [Gassner et al. 2007].

3.8 Short recipe for implementation

Although the derivation draws heavily on kinetic theory, the resulting solver relies purely on basic arithmetic operations. The overall computational workflow, illustrated in Fig. 2, proceeds as follows:

- Using the solution points and correction function formulations from Huynh [2007], construct the globally constant matrices $\mathbf{D}$, $\mathbf{R}$, $\mathbf{C}$, and $\mathbf{Q}$ via Eqs. (7) to (9).
- Compute the kinematic viscosity ν and collision time $\tau = \nu/RT$. The isothermal RT is treated as a unified constant (the square of the speed of sound, typically set to 1/3), and the reference velocity is typically set to approximately 0.1.
- Prior to each time step, compute the time step size Δt via Eq. (28).
- In the *predictor step*, compute the element-local gradients $\nabla\mathbf{U}$ at all solution points via the matrix $\mathbf{D}$.
- Compute the predicted moments $\mathbf{U}(h)$ and $\mathbf{\Pi}(h)$ at all solution points via Eqs. (22) and (23), using the moment expressions detailed in Appendix B.
- In the *corrector step*, compute the element-local flux $\mathbf{F}(h)$ at all solution points via Eq. (24).
- Compute the common state $\mathbf{U}^*(h)$ and common flux vector $\mathbf{F}^*(h)$ via Eqs. (25) to (27) at the interfaces.
- Assemble the local block vectors defined in Eq. (8), and update $\mathbf{U}^{(n)} \to \mathbf{U}^{(n+1)}$ at all solution points via Eq. (10).
- Repeat this procedure for the next spatial dimensions, accumulating all corresponding contributions.

As a practical guideline, we recommend that the *cell/element Reynolds number* satisfy $uL/\nu \le 4000$ to avoid visible under-resolution oscillations. In our tests, such oscillations did not cause blow-up, and can

be mitigated with the technique described in Section 5.1 if desired. For the determination of the stable time step Δt, we adopt:

$$\Delta t = \frac{\mathrm{C}}{2K-1}\frac{L}{u_{\max}+\sqrt{RT}}, \tag{28}$$

where L is the element length and $u_{\max}$ is the maximum velocity magnitude. The CFL constant C should be $C \leq 0.6$ for $K \leq 5$ and $C \leq 0.5$ for $K \geq 6$. This pipeline maps naturally to GPU hardware; gradient data never leaves the chip, as described in Section 4.

4 IO-Aware GPU-Optimized Implementation

To achieve full throughput for KPM-FR, we present an IO-aware design tailored for commodity GPUs. We focus on tensor-product elements over structured grids, exploiting dimensional separability for regular data access patterns and high computing efficiency. The design of our implementation is guided by a core objective: *IO awareness*. For modern GPUs, memory bandwidth has failed to keep pace with the exponential growth in computational throughput. This mismatch is particularly pronounced for tensor-product FR schemes [Trojak et al. 2022]: the formulation entails extensive matrix operations, and executing these via general matrix-matrix multiplication (GEMM) kernels requires writing large intermediate arrays (e.g., gradient tensors) to global memory, which effectively bottlenecks performance. This pattern is typical in general FR implementations that use GEMM kernels for derivative calculations. We optimize the scheme to enforce data locality, ensuring FR computations reside almost entirely in the low-latency on-chip memory hierarchy, including registers, shared memory, and L2 cache.

This design is realized through two coupled architectural choices: *fully fused per-stage kernels* and *L2-aware halo data access*.

(1) Each stage (predictor or corrector) runs as a single kernel launch, performing differentiation, flux evaluation, split-form, and correction entirely on-chip, with shared memory for dimensional transposition.
(2) Gauss-Lobatto solution points expose boundary values directly, removing the need for explicit flux storage or dedicated flux kernels, and enabling the fused corrector to fetch neighbor data without extrapolation or extra buffers.

Together, these designs ensure gradient data never leaves on-chip memory. The only quantities transferred through global memory between the predictor and corrector are the predicted moments $\mathbf{U}(h)$ and $\mathbf{\Pi}(h)$ from Section 3.6, whose compactness is inherent to the formulation. Hardware constraints and the mathematical framework therefore converge on the same minimal payload.

Thread mapping. Thread mapping serves as the critical enabler of high computational throughput. Each 3D element is allocated K^2 threads, where each thread processes a 1D slice of length K, analogous to that proposed in Świrydowicz et al. [2019]. This enables adjacent K^2 threads to collaboratively load element data through K coalesced memory reads, minimizing global memory access latency. Intra-element operations (e.g., the matrix-vector product $\mathbf{DU}$) are executed as in-register computations, eliminating the overhead of intermediate memory accesses. Coefficients of constant matrices (e.g., $\mathbf{D}$, $\mathbf{CR}$) are *precomputed* and fully unrolled at compile time via template metaprogramming. A 128-thread block is configured to accommodate the maximum number of K^2-thread element groups, with any residual threads remaining idle to maintain alignment with hardware warp sizes. In practical deployments, we restrict the value of K to the range $2 \leq K \leq 6$. This 128-thread block size is determined via systematic parameter sweeps to achieve peak throughput.

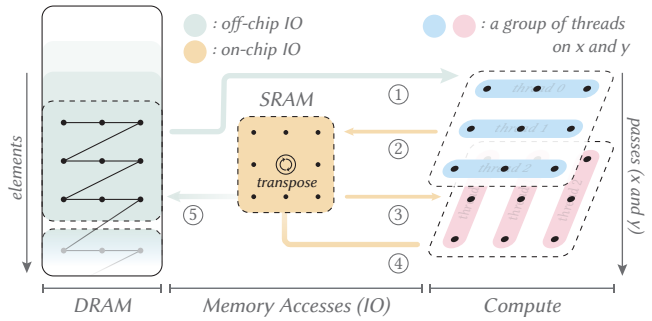


Fig. 5. **Schematic of the IO-aware GPU-optimized implementation.** Within a single fused kernel, tensor-product FR operations are executed via directional splitting without intermediate global-memory IO. Threads are mapped to 1D SP lanes (right; blue and pink groups correspond to the x- and y-direction passes), while shared memory (SRAM, middle) serves as a low-latency buffer with in-place transposition for switching between spatial passes. Green and orange arrows distinguish off-chip and on-chip data movement. The numbered workflow proceeds as: (1) load flow data from DRAM; (2) execute 1D updates and write intermediate results to SRAM; (3) read transposed data for the next spatial pass; (4) buffer processed results in SRAM; and (5) stream final results back to DRAM. Only steps (1) and (5) traverse global memory; all intermediate operations remain on-chip.

Memory layout. We adopt a *structure of arrays of structures* (SoAoS) format for the storage of per-element data. All solution points belonging to a single element are stored in a contiguous memory block, whereas distinct physical components (e.g., density, momentum) are assigned the maximum memory stride. When combined with the thread mapping strategy detailed above, this memory layout ensures coalesced memory loads and stores for element data.

Dimensional splitting. Multi-dimensional operators are computed within a single kernel via directional splitting. Each element group allocates exactly K^3 `FP32` entries of shared memory for tensor transpositions across different spatial directions. Following the completion of 1D operations along the x-direction, threads write intermediate results to the K^3 scratch buffer, synchronize across the thread block, and subsequently read back data as y-aligned slices; this process is then repeated for the z-direction. Memory bank conflicts at $K = 4$ are eliminated via padding one dimension of the shared memory array. Prior to global memory writes, per-element results are staged in shared memory and subsequently written back in a coalesced, component-wise fashion. This memory and computation structure can further be leveraged for transformations between nodal and modal representations [Hesthaven and Warburton 2008].

Halo access. For the evaluation of common flux vectors, it is critical to constrain non-local memory accesses to the same data arrays. We therefore avoid allocating a dedicated array for interface-resident

quantities. The adoption of Gauss-Lobatto solution points inherently exposes boundary values at $r = \pm 1$, obviating the need for a distinct boundary array. Within the corrector, each element group first retrieves boundary points from neighboring elements before loading interior solution points. This access pattern is engineered to maximize cache reuse and minimize halo data traffic to global memory. We implement the common flux vector formulation for the face normal $\mathbf{n} = [1, 0, 0]^\top$. For other face orientations, the identical kernel is reused by applying a fixed cyclic permutation to state and flux components both prior to and following kernel evaluation.

Mixed-precision storage. In FP32 precision, the predictor performs $K^3 \times 4$ memory reads and $K^3 \times (4 + 5)$ memory writes per element, whereas the corrector executes $K^3 \times (4 + 4 + 5)$ effective reads and $K^3 \times 4$ writes per element. To mitigate memory traffic, we propose a mixed-precision optimization tailored to the viscous stress tensor $\mathbf{\Pi}$. Each component of $\mathbf{\Pi}$ remains on the order of $O(\tau)$ and is nearly zero, exhibits insensitivity to reference velocity shifts, and is not accumulated across iterative stages. As such, these components tolerate numerical quantization without introducing discernible bias. We therefore store $\mathbf{\Pi}$ in FP16 precision while retaining all arithmetic operations in FP32 to preserve accuracy. With this strategy, each solution point incurs only 25 effective accesses per time step (down from 30) and occupies approximately 11 FP32 word equivalents of storage (compared to 13 originally), inclusive of auxiliary data.

Taken together, these choices yield an implementation whose on-chip SRAM footprint depends only on the element size K, not on the complexity of the underlying physical model. The resulting throughput is quantified in Section 6.3 and Fig. 10, alongside a comparison with mainstream FR implementations.

5 Practical Considerations

Applying KPM-FR to practical flow problems raises two additional requirements: ensuring robustness at coarse grid resolutions or near the inviscid limit, and enforcing boundary conditions on both immersed and domain-aligned object surfaces.

5.1 Numerical robustness

When applied to under-resolved high-Reynolds-number flows where the grid fails to adequately resolve the smallest fluid scales, KPM-FR encounters three critical issues: nonlinear aliasing, general numerical instabilities, and spurious oscillations in shear-dominated regions. While turbulence modeling [Dzanic et al. 2022; Pope 2015] would be more appropriate here, it requires further investigation and lies outside the scope of this work. To enable stable high-Reynolds-number flow simulations, each issue is addressed via a lightweight modification to the core update loop (Section 3.8) in the following.

Anti-aliasing via split-form. High-order discretizations are prone to nonlinear aliasing errors: when nonlinear flux terms are evaluated within finite polynomial spaces, high-wavenumber content is aliased back into the resolved scales as spurious low-frequency energy [Jameson et al. 2012; Vincent et al. 2011], thereby destabilizing numerical simulations. We employ the split-form flux formulation [Abe et al. 2018] as a lightweight anti-aliasing remedy, by modifying the evaluation of the internal flux divergence term in Eq. (10). The split form recasts the flux divergence as:

$$\frac{\partial}{\partial r}\left(\rho u_n \boldsymbol{\phi}\right) = \frac{1}{2}\frac{\partial}{\partial r}\left(\rho u_n \boldsymbol{\phi}\right) + \frac{1}{2}\boldsymbol{\phi}\frac{\partial}{\partial r}\left(\rho u_n\right) + \frac{1}{2}\rho u_n \frac{\partial}{\partial r}\boldsymbol{\phi},$$

where u_n denotes the advective velocity and $\boldsymbol{\phi} = [1, u, v, w]^\top$ comprises the state variables. In our numerical implementation, the term $\mathbf{QF}$ in Eq. (10) is replaced with the following:

$$\begin{aligned}\mathbf{QF} \leftarrow\ & \frac{1}{2}\mathbf{D}(\rho u_n \boldsymbol{\phi}) + \frac{1}{2}\boldsymbol{\phi}\mathbf{D}(\rho u_n) + \frac{1}{2}\rho u_n \mathbf{D}(\boldsymbol{\phi}) \\ & + \mathbf{DF}^p + \mathbf{DF}^\nu - \mathbf{CRF},\end{aligned}$$

where $\mathbf{F}^p$ and $\mathbf{F}^\nu$ represent the pressure and viscous flux components, both evaluated at the half-time step h. This should *always* be employed in practice: since the differentiation matrix $\mathbf{D}$ is already applied in-kernel (Section 4), the additional split-form passes introduce negligible computational overhead.

Tunable dissipation via flux blending. We stabilize the scheme by blending with the canonical KFVS flux [Chou and Baganoff 1997], which is naturally dissipative. We introduce a blending parameter λ to interpolate between our scheme and the KFVS flux, enabling tunable numerical dissipation, with both limiting cases (i.e., $\lambda = 0$ and $\lambda = 1$) ensuring numerical robustness [Sun et al. 2016]. This requires no additional overhead, and is realized simply by modifying the coefficients A and B in Eqs. (25) and (27) to $\tilde{A} = (1 - \lambda)A + \lambda$ and $\tilde{B} = (1 - \lambda)B$, respectively. We recommend $\lambda = 0.5$.

Tangential transport correction. Near the inviscid limit, isotropic flux formulations (e.g., the Rusanov scheme [Toro 2009], the kinetic flux defined in Eqs. (25) and (27), or conventional GKS variants) tend to induce spurious numerical oscillations. To mitigate this issue, we adopt a hybrid formulation that incorporates a tangential-component treatment inspired by Liou and Steffen [1993] into our kinetic framework. The key idea is to decouple the transport processes: the normal flux retains its fully kinetic form to ensure stability, whereas the tangential components are reinterpreted as passive scalars that are advected exclusively by the kinetic mass flux m^*_{100}. The revised tangential flux expressions then *replace* the original kinetic moment terms in Eq. (25) as follows:

$$\begin{aligned}F^*_2(h) &= \bar{F}_2(h) + \frac{B}{2}\left[m^*_{100}\left(v_L + v_R\right) + \left|m^*_{100}\right|\left(v_L - v_R\right)\right], \\ F^*_3(h) &= \bar{F}_3(h) + \frac{B}{2}\left[m^*_{100}\left(w_L + w_R\right) + \left|m^*_{100}\right|\left(w_L - w_R\right)\right],\end{aligned} \tag{29}$$

where $F^*_2(h)$ and $F^*_3(h)$ represent the modified tangential flux components, i.e., the $\alpha = y$ and $\alpha = z$ components, respectively. Note that the stabilization strategy in Section 5.1 should be applied anisotropically: we restrict the blending operation to the normal component of the kinetic flux, whereas the tangential components in Eq. (29) retain a blending parameter of $\lambda = 0$. This adds tangential dissipation that acts as an implicit large-eddy-simulation (iLES) model while suppressing unwanted spurious oscillations (see supplementary material for comparisons).

5.2 Solid boundary treatments

We handle solid boundaries in two ways: volumetric forcing within elements for immersed obstacles and ghost states at element interfaces for domain boundary faces.

5.2.1 Volumetric forcing for immersed boundaries. The volumetric approach is employed for immersed obstacles [Kou et al. 2022] to enforce the no-slip boundary condition. To this end, we implement a split forcing substep with a discrete source term:

$$\mathbf{S} = \frac{\chi}{\Delta t}[0, \rho(u_w - u), \rho(v_w - v), \rho(w_w - w)]^\top,$$

where $\chi \in [0, 1]$ denotes the solid volume fraction; Δt is the time-step size specified in Eq. (28), and $\mathbf{u}_w = [u_w, v_w, w_w]^\top$ represents the wall velocity vector of the obstacle. In practice, to integrate this substep into the computational kernels, we substitute $\mathbf{U}^{(n)}$ with $(1-\chi)\mathbf{U}^{(n)} + \chi\mathbf{U}_w$ during the macroscopic variable reconstruction of the corrector. Here, $\mathbf{U}_w$ is constructed using the fluid density ρ and the solid velocity $\mathbf{u}_w$; that is, only the momentum components are enforced to $\rho\mathbf{u}_w$, while the density ρ remains unchanged. This yields a simple forcing update over the time interval Δt with source term $\mathbf{S}$. In our implementation, the volume fraction χ is computed by rasterizing the obstacle geometry into a volumetric mask and sampling this mask at the solution points. For moving obstacles, the wall velocity $\mathbf{u}_w$ is evaluated pointwise based on rigid-body motion, encompassing both translational and rotational degrees of freedom. To suppress numerical artifacts, a smoothed profile can be adopted in the construction of the volumetric mask $\chi = 0.5\,[1 + \tanh(\psi/\delta)]$, where ψ denotes the signed distance (with $\psi > 0$ indicating the interior region of the obstacle), δ is a user-defined smoothing width that controls the interface thickness (typically set to $\delta \approx L/K$ for visualizations, i.e., the average solution point spacing where L is the element length), and the sign of ψ is determined via winding number [Barill et al. 2018].

5.2.2 Interface boundaries via ghost variables. For boundaries that align with element interfaces, we tailor the computation of the common flux vector. Given that both the Riemann solver and our kinetic formulation require flow variables on both sides of an interface, we construct a *ghost state* to substitute for the missing neighboring state. For a no-slip wall boundary, this ghost state is simply defined by reversing the momentum components of the interior state: $\mathbf{U}_B = [\rho, -\rho\mathbf{u}]^\top_{\text{in}}$, where the subscript "in" denotes the interior flow state adjacent to the boundary. For gradient-dependent terms, we adopt the interior values for no-slip walls, whereas these terms are set to zero for free-slip and far-field boundaries. Wall forces are computed via the integration of surface stresses. Far-field boundaries, which are employed to model open computational domains and mitigate reflections of outgoing waves, are handled using the isothermal characteristic approach [Mengaldo et al. 2014]. Considering a right-hand boundary as an example, the interior flow is represented by the left state $\mathbf{U}_L$, and the ghost variable $\mathbf{U}_B$ is constructed on the right-hand side of the boundary. Let $\mathbf{U}_\infty$ denote the prescribed far-field reference variable. We define the normal velocity $u_\text{n} = \mathbf{u} \cdot \mathbf{n}$ and the speed of sound $a = \sqrt{RT}$, then introduce the Riemann invariants $J_+ = u_\text{n} + a \ln \rho$ and $J_- = u_\text{n} - a \ln \rho$. This yields the following expression:

$$u_{B,\text{n}} = \frac{1}{2}\left[J_+(\mathbf{U}_L) + J_-(\mathbf{U}_\infty)\right], \quad \rho_B = \exp\left(\frac{J_+(\mathbf{U}_L) - J_-(\mathbf{U}_\infty)}{2a}\right).$$

Tangential velocity components are assigned values from the interior variable if the normal velocity component of the interior variable satisfies $u_\text{n}(\mathbf{U}_L) \geq 0$; otherwise, they are taken from the far-field reference variable $\mathbf{U}_\infty$. The ghost variable thus constructed, $\mathbf{U}_B = [\rho_B, \rho_B\mathbf{u}_B]^\top$, is subsequently employed to evaluate the common flux vector at the boundary interface.

6 Results and Discussion

Our GPU-accelerated KPM-FR solver is implemented using NVIDIA CUDA [NVIDIA 2025], with optimized support for tensor-product elements on structured grids. Unless otherwise specified, all simulations were performed on a workstation equipped with an Intel i9-14900K CPU, 62 GiB DDR5 RAM, and a single NVIDIA RTX 4090 GPU. The only exceptions include a specialized test case that utilized a single NVIDIA RTX 5090 GPU, as well as a large-scale simulation case that used a single NVIDIA RTX PRO 6000 GPU. Photorealistic flow visualizations are produced with Blender [Blender Online Community 2025], while in-situ post-processing and quantitative flow analysis are implemented via ParaView Catalyst [Ayachit et al. 2021] and the Visualization Toolkit (VTK) [Schroeder et al. 2006].

We adopt a hybrid four-component visualization strategy to characterize the flow field comprehensively. First, direct volume rendering of vorticity magnitude is used to resolve intricate 3D flow structures. Second, coherent flow features are identified via Q-criterion isosurfaces [Hunt et al. 1988], which are exported to Blender for high-fidelity geometric representation. Third, planar cross-sectional slices are extracted to interrogate the spatial distribution of key physical quantities. Lastly, passive Lagrangian smoke particles are advected through the flow field to enable photorealistic rendering.

To demonstrate the intrinsic robustness of the proposed method, we employ a consistent baseline configuration across most test cases: Gauss-Lobatto solution points with the g_2 correction (Section 3.3), mixed-precision storage (Section 4), and a fixed stabilization factor of $\lambda = 0.5$ (Section 5.1), with no additional tangential transport corrections applied (Section 5.1), since the considered cases are neither significantly under-resolved nor inviscid. For the PyFR-based experiments (Sections 6.1.1 and 6.1.4), the FR correction operator follows PyFR's implementation, which corresponds to the DG correction. For each individual test case, the CFL number is selected to maximize computational throughput while preserving strict numerical stability. Across all investigated flow regimes, the scheme maintains robust stability with no further empirical parameter tuning required. Table 2 presents the configurations and timing statistics corresponding to the key simulation results reported in this work.

6.1 Validations and comparisons

We validate and benchmark KPM-FR against established methods to assess accuracy, numerical dissipation, memory footprint, and throughput. The validation framework of our solver is structured around three core objectives: fundamental numerical consistency, boundary treatment accuracy on structured and unstructured grids, and intrinsic solution fidelity in complex transitional flow regimes.

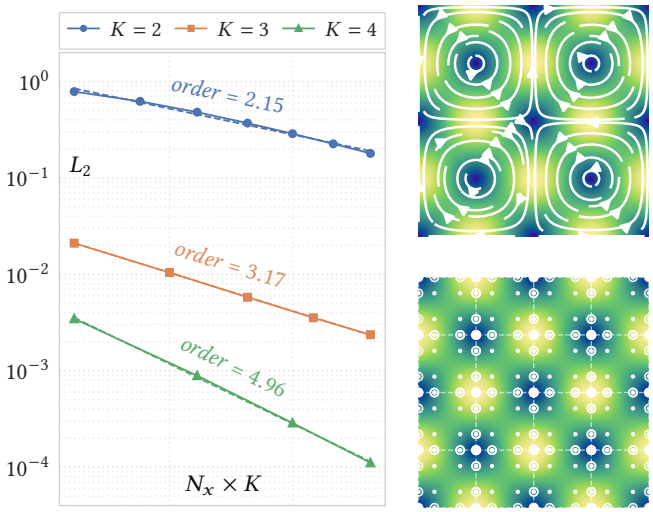


Fig. 6. **High-order convergence in 2D Taylor-Green vortex simulation.** (a) Relative L_2 velocity error at $t = 60$ plotted against the total degrees of freedom ($N_x \times K$). The slope of each curve closely aligns with the theoretically predicted order of accuracy, with the $K = 4$ case exhibiting superconvergence. (b) Visualization of flow streamlines (top panel) and the distribution of internal solution points for a grid with $N_x = 4$ and $K = 4$ (bottom panel).

First, using the Taylor-Green vortex (TGV) benchmark, we verify its high-order convergence characteristics in 2D and quantify its numerical dissipation behavior and computational efficiency against reference methods in 3D. Next, we validate the immersed boundary method via the canonical flow-past-sphere benchmark case. Finally, to decouple the scheme's solution accuracy from potential grid-induced artifacts inherent to structured meshes, we have integrated KPM-FR into the PyFR framework [Witherden et al. 2025] to validate the challenging SD7003 airfoil test case with a body-fitted unstructured mesh. Together, these benchmarks assess the solver's memory footprint, computational throughput, and numerical dissipation characteristics.

In subsequent sections, we denote our scheme as KPM-FR-KX, where X denotes the number of solution points along each coordinate direction in a single element. For reference, C17-LBM (cumulant LBM) refers to the scheme proposed by Geier et al. [2017], which features optimized adaptive high-order relaxation to reach spatially fourth-order accuracy in diffusion; this variant is used without its native limiter to minimize numerical dissipation and leverages the highly optimized GPU code from Geier et al. [2025]. HOME-LBM denotes the state-of-the-art MR-LBM by Li et al. [2023]; FluidX3D (v3.5) [Lehmann 2022], a highly memory-efficient SRT-LBM solver using a D3Q19 lattice structure with `FP32/FP16C` precision (the `FP32/FP16S` mode showed suboptimal throughput on our hardware), is included as a representative SRT-LBM baseline.

6.1.1 2D Taylor-Green vortex simulation. To validate the spatial accuracy of our scheme and evaluate its superconvergence behavior, we first conduct simulations of the 2D Taylor-Green vortex (TGV) in a periodic domain of $[0, 2\pi]^2$. This benchmark admits the following analytical solution to the NS equations:

$$\rho^*(x, y, t) = 1 + \frac{v_0^2}{4RT}\left(\cos(2x) + \cos(2y)\right)\exp(-4\nu t),$$
$$u^*(x, y, t) = -v_0 \sin(x)\cos(y)\exp(-2\nu t),$$
$$v^*(x, y, t) = v_0 \cos(x)\sin(y)\exp(-2\nu t),$$

where v_0 denotes the reference velocity. We specify $\mathrm{Ma} = 0.01$ and $\mathrm{Re} = 100$ for the simulations. Double-precision arithmetic is adopted to minimize machine-level numerical noise. The discrepancy is quantified using the relative L_2 norm of the velocity field at $t = 60$. The integrals are computed via high-order quadrature over the elements. As shown in Fig. 6, the velocity error exhibits algebraic decay with increasing spatial resolution. The fitted slopes are approximately 2.15, 3.17, and 4.96 for $K = 2, 3, 4$, respectively. Notably, the $K = 4$ case demonstrates distinct superconvergence behavior, with a fitted slope of approximately 5.

6.1.2 3D Taylor-Green vortex simulation. While striking visualizations of fully developed turbulence can illustrate a scheme's capacity to preserve coherent flow structures, such qualitative results are often inadequate for a rigorous validation of a fluid solver's intrinsic resistance to numerical dissipation. Our second benchmark, the 3D Taylor-Green vortex simulation, acts as a well-established canonical test case for quantifying such numerical dissipation effects. Propagating in a fully periodic domain of $[0, 2\pi]^3$, the 3D TGV is devoid of boundary condition-related complexities, enabling a clean, comprehensive evaluation of the scheme's inherent numerical dissipation behavior and spectral performance. We focus primarily on a Reynolds number of $\mathrm{Re} = 1600$, a canonical transitional flow regime where the flow field is highly sensitive to the numerically dissipative characteristics of the discretization schemes. A visualization of the resulting flow field is presented in Fig. 7b.

We monitor the temporal evolution of the volume-averaged kinetic energy and enstrophy. Unless otherwise noted, all quantities presented in this subsection are non-dimensional. Time is normalized by the characteristic time scale $1/v_0$, with $E_k(t)$ and $\mathcal{E}(t)$ denoting the non-dimensional kinetic energy and enstrophy, respectively. The volume-averaged kinetic energy is defined as:

$$E_k(t) = \frac{1}{2\rho_0|\Omega|}\int_\Omega \rho(\mathbf{x}, t)|\mathbf{u}(\mathbf{x}, t)|^2 \,\mathrm{d}\mathbf{x}, \tag{30}$$

where Ω denotes the domain volume and ρ_0 represents the reference density. The enstrophy is defined as:

$$\mathcal{E}(t) = \frac{1}{2\rho_0|\Omega|}\int_\Omega \rho(\mathbf{x}, t)|\nabla \times \mathbf{u}(\mathbf{x}, t)|^2 \,\mathrm{d}\mathbf{x}, \tag{31}$$

where $\nabla \times \mathbf{u}$ denotes the vorticity field. It is worth noting that on structured grids, enstrophy evaluation necessitates high-order finite-difference schemes to preserve computational accuracy [Geier et al. 2021]. While the global kinetic energy $E_k(t)$ characterizes the overall state of the flow system, the enstrophy $\mathcal{E}(t)$ is highly sensitive to local small-scale flow gradients and thus directly quantifies dissipative effects. For ideal incompressible flows, these two integral quantities satisfy the following balance equation:

$$\mathcal{E}(t) = -\frac{1}{2\nu}\epsilon(t) = -\frac{1}{2\nu}\frac{\mathrm{d}E_k(t)}{\mathrm{d}t}, \tag{32}$$

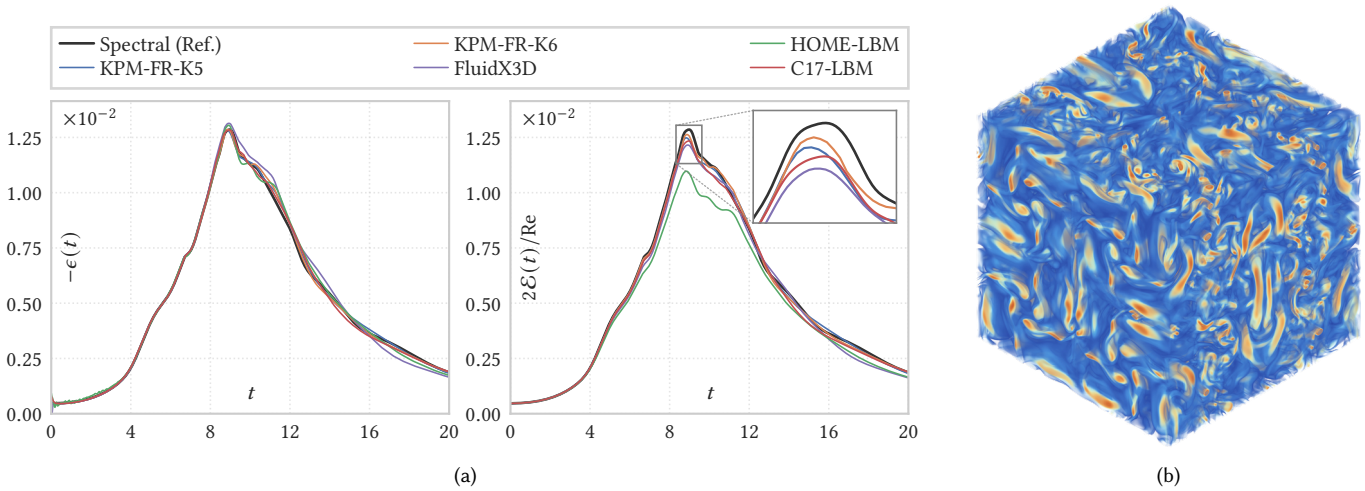


Fig. 7. **Spectral accuracy and dissipation characteristics.** (a) Temporal evolution of dissipation rates at 256^3 solution points, referenced against the spectral direct numerical simulation (DNS) of Wang et al. [2013]. KPM-FR schemes and C17-LBM faithfully capture both the peak timing and magnitude; FluidX3D falls between C17-LBM and HOME-LBM, while HOME-LBM exhibits a notable deficit in the enstrophy-based metric, consistent with stronger numerical dissipation. (b) Volume rendering of vorticity magnitude at $t = 20$ for KPM-FR-K6, illustrating well-resolved fine-scale vortical structures.

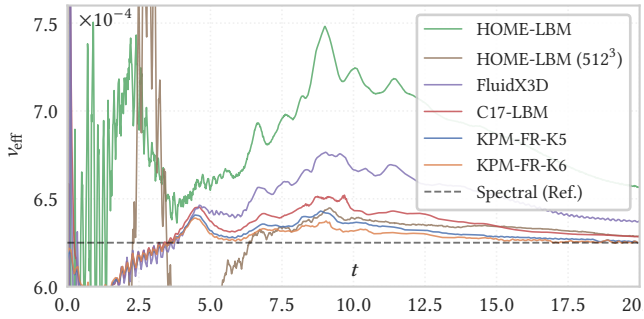
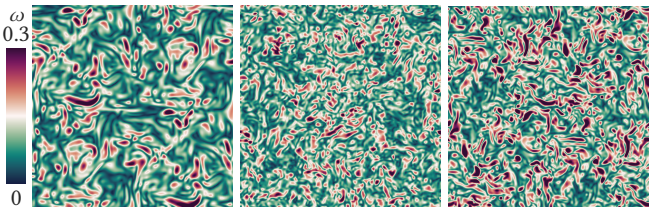


Fig. 8. **Quantitative assessment of numerical dissipation.** Proximity to the spectral reference ($\nu = 6.25\times10^{-4}$) indicates lower numerical dissipation. KPM-FR-K5 and KPM-FR-K6 closely recover the physical viscosity (Ref.) with negligible numerical bias. C17-LBM and FluidX3D retain a moderate positive offset, while HOME-LBM exhibits a markedly elevated effective viscosity even at 512^3 ($8\times$ the solution points of the 256^3 baseline).



(a) HOME-LBM (128^3) (b) KPM-FR-K6 (128^3) (c) HOME-LBM (192^3)

Fig. 9. **Visual comparison of vorticity magnitude**. For the 3D Taylor-Green vortex simulation at $t = 20$ and a different Reynolds number of Re = 16000 (following the high-Reynolds setup of Geier et al. [2021] with tangential correction enabled) to magnify dissipation differences between schemes, we compare slices of the vorticity magnitude field derived from different numerical schemes and spatial resolutions. Notably, KPM-FR-K6 resolves fine-scale vortical structures, exhibiting qualitatively finer structures than the higher-resolution HOME-LBM (192^3) simulation.

where $\epsilon(t)$ is the kinetic energy dissipation rate [Pope 2015]. This relationship enables the quantification of numerical dissipation. Following Geier et al. [2021]; Zhou et al. [2014], we define the effective viscosity as $\nu_{\text{eff}}(t) = -\epsilon(t)/(2\mathcal{E}(t))$. In a strictly incompressible solver free of numerical errors, $\nu_{\text{eff}}(t)$ recovers the physical viscosity ν. Accordingly, for our weakly compressible flow solver, deviations from ν provide a direct quantitative metric for the numerical dissipation of the scheme. The initial condition of the flow is:

$$\begin{aligned}
\rho(x,y,z,0) &= 1 + \frac{v_0^2}{16RT}\left(\cos(2x)+\cos(2y)\right)\left(\cos(2z)+2\right),\\
u(x,y,z,0) &= v_0 \sin(x)\cos(y)\cos(z),\\
v(x,y,z,0) &= -v_0 \cos(x)\sin(y)\cos(z),\\
w(x,y,z,0) &= 0,
\end{aligned}$$

which yields Ma = 0.1. The simulation is executed until $t = 20$. To ensure rigorous cross-method comparison, we normalize the spatial resolution to approximately 256 solution points per dimension. For LBM methods, this value corresponds to the lattice resolution; for our scheme, it is defined as $N \times K$, where $N = \lfloor 256/K \rfloor$. The CFL numbers are set to 0.6 for KPM-FR-K5 and 0.5 for KPM-FR-K6.

Fig. 7 depicts the temporal evolution of two key dissipation metrics: the kinetic energy dissipation rate $-\epsilon(t)$ (left panel in (a)) and the enstrophy-based dissipation rate $2\mathcal{E}(t)/\text{Re}$ (right panel in (a)). Our KPM-FR-K5 and KPM-FR-K6 schemes exhibit excellent agreement with the high-resolution spectral reference [Wang et al. 2013], faithfully reproducing both the timing of peak dissipation ($t \approx 9$) and its corresponding magnitude. While C17-LBM captures the general trend with a slightly attenuated peak, HOME-LBM shows a marked discrepancy in the enstrophy-based metric. This substantial

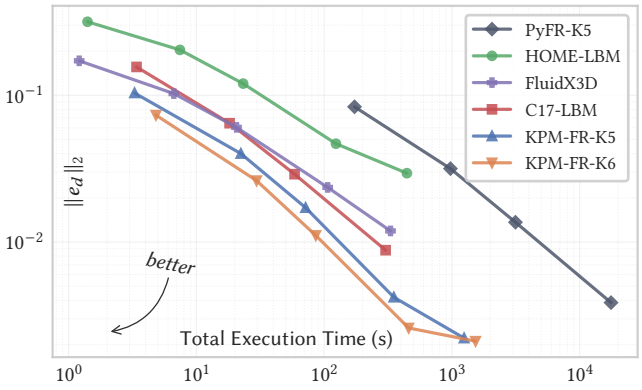


Fig. 10. **Pareto frontier between fidelity and efficiency.** Comparison of different solvers in L_2 dissipation error against computational cost (total execution time) for the 3D Taylor-Green vortex simulation at Re = 1600. The KPM-FR schemes (blue and orange curves) push the Pareto frontier toward the optimal bottom-left region, achieving lower dissipation error at comparable or lower cost than the PyFR and LBM baselines. Each curve corresponds to simulations conducted at spatial resolutions of 128, 192, 256, 384, and 512 solution points per dimension. Simulations that resulted in out-of-memory errors are excluded from the plot.

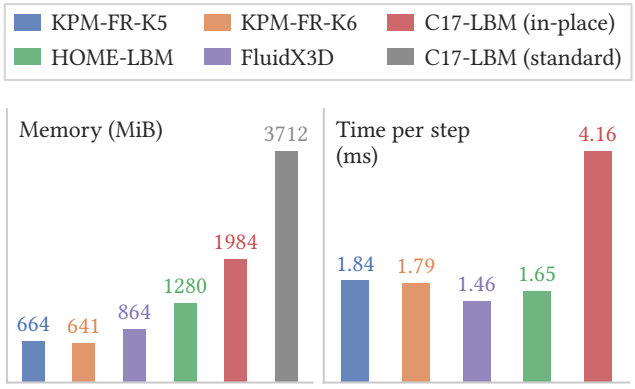


Fig. 11. **Memory efficiency and runtime performance.** We present the estimated state memory (left) and per-step wall-clock time (right) at a fixed 256^3 resolution on an NVIDIA RTX 4090. Memory bars estimate resident state sizes rather than measured peak allocation to normalize between implementations; for C17-LBM, both the in-place (single-buffer) layout and a standard (double-buffer) reference layout are shown. It is clear that KPM-FR uses the least estimated state memory among the compared methods while maintaining comparable throughput.

underprediction points to stronger numerical dissipation, which attenuates small-scale turbulent structures prior to the full development of the energy cascade. FluidX3D falls between C17-LBM and HOME-LBM in both metrics, consistent with its SRT collision operator [Geier et al. 2021]. This trend is further reflected by the temporal evolution of the effective viscosity $\nu_{\text{eff}}(t)$ (Fig. 8), where the KPM-FR-based schemes recover the theoretical viscosity value $\nu = 6.25 \times 10^{-4}$ with negligible bias. In contrast, C17-LBM retains

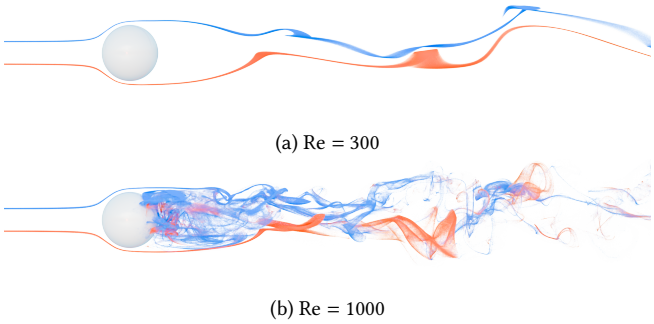


Fig. 12. **Flow transition behind a sphere.** Flow field visualizations at (a) Re = 300, depicting periodic vortex shedding, and (b) Re = 1000, displaying a chaotic, turbulent wake structure. The immersed boundary method effectively captures the laminar-to-turbulent flow transition.

Table 1. **Drag prediction for flow past a sphere.** Time-averaged drag coefficient C_d compared with an experimental correlation [Cheng 2009] and direct numerical simulation (DNS) reference data [Johnson and Patel 1999; Ploumhans et al. 2002]. Over the Reynolds number range of Re = 100 to 1000, the maximum absolute relative error (with respect to either reference dataset) is below 3 %. Here, N_d denotes the number of solution points spanning the diameter of the sphere.

Re	N_d	Ours	Exp.	DNS	Max. err. (%)
100	27	1.0802	1.1024	1.09	2.01
300	40	0.6550	0.6708	0.66	2.36
1000	54	0.4787	0.4663	0.48	2.66

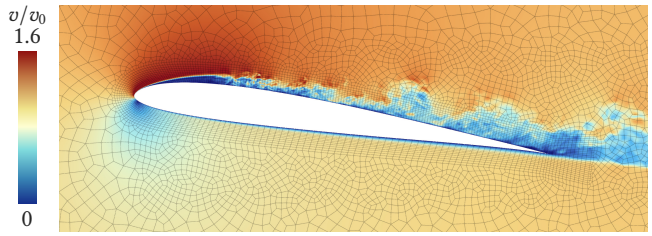
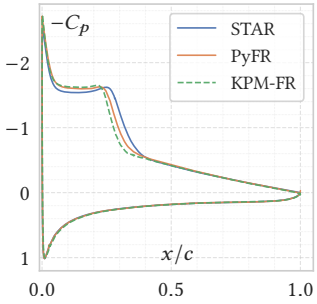


Fig. 13. **Transitional flow features over the SD7003 airfoil.** We present a 2D slice of a 3D instantaneous velocity magnitude field at Re = 60000 and $\alpha = 8°$. The solver clearly resolves the laminar separation bubble on the suction surface and its subsequent transition to turbulent wake.

a slight bias in $\nu_{\text{eff}}(t)$, whereas HOME-LBM yields a markedly elevated $\nu_{\text{eff}}(t)$ together with pronounced oscillations that persist at 512^3, resulting from the conditioned FP16 shared-memory storage required by the configuration published in their paper, which our implementation faithfully reproduces. Visual comparisons of the vorticity field at $t = 20$ further corroborate these differences (Fig. 9).

Beyond spectral accuracy, practical utility is dictated by the Pareto frontier between fidelity and efficiency. We benchmark total execution time against dissipation error under the hardware specifications detailed previously. A PyFR (v2.1) solver [Witherden et al. 2025] is included for reference, configured with Gauss-Legendre quadrature



Method	C_l	C_d	x_{sep}	x_{rea}
DGSEM	0.923	0.045	0.027	0.310
PyFR	0.941	0.049	0.045	0.315
STAR	0.945	0.055	0.036	0.381
Ours	**0.940**	**0.047**	**0.036**	**0.298**
Exp.	0.92	0.029	—	—

Fig. 14. **Quantitative validation of the SD7003 airfoil.** Time-averaged pressure coefficient distribution over the airfoil surface (left panel) and comparison of integral coefficients (right panel) at Re = 60000 and $\alpha = 8°$. Reference datasets are sourced from Beck et al. [2014] for the discontinuous Galerkin spectral element method (DGSEM), Vermeire et al. [2017] for PyFR and industrial software STAR-CCM+, and Selig [1995] for experimental measurements. Here, x_{sep} and x_{rea} represent the separation and reattachment points, respectively, in normalized chord coordinates.

points and second-order Runge-Kutta time integration to mitigate aliasing errors. In graphics applications, numerical dissipation constitutes the key factor degrading visual fidelity, whereas in CFD applications, it represents the main source of error that undermines the accurate resolution of vortical structures and turbulent characteristics; we therefore prioritize a dissipation-centric metric $\|e_d\|_2$, defined as the residual derived from the enstrophy-energy balance:

$$\|e_d\|_2 = \sqrt{\frac{\int_0^{20} [2\mathcal{E}(t)/\text{Re} + \epsilon(t)]^2 \, \mathrm{d}t}{\int_0^{20} \epsilon(t)^2 \, \mathrm{d}t}}.$$

Fig. 10 depicts the relationship between dissipation error and total execution time. Our KPM-FR-K6 scheme defines the Pareto frontier, delivering the minimal error for a given computational budget. At matching resolutions, the dissipation error from FluidX3D remains above C17-LBM and KPM-FR. At comparable dissipation error, KPM-FR shifts the efficiency frontier beyond the community-standard PyFR framework in this setting. This advantage is expected when contrasting a specialized implementation with a general-purpose framework, and highlights the efficiency gains afforded by hardware-oriented solver design. Fig. 11 presents a comprehensive performance summary at a resolution of 256^3 solution points. Memory is estimated rather than measured to exclude implementation-specific overhead (for LBM, the estimate covers distribution populations and macroscopic variables); for C17-LBM, both the widely used double-buffer layout and the memory-optimized in-place variant of VirtualFluids [Geier et al. 2025] are shown to span the practical range of LBM configurations. From a roofline analysis perspective (refer to Section 6.3), all computational kernels are memory-bound. Specifically, HOME-LBM, C17-LBM, FluidX3D, and our KPM-FR-KX attain effective bandwidth utilization rates of above 80% relative to a `memcpy` operation. As a result, the comparative analysis is insensitive to implementation details, and further low-level code optimization would not significantly alter the core conclusions drawn herein.

6.1.3 Flow past a sphere. To validate our structured-grid immersed boundary implementation, we simulate flow past a sphere across a range of flow regimes. As visualized in Fig. 12, our solver faithfully captures the physical flow transition, from periodic vortex shedding at Re = 300 to a chaotic, three-dimensional wake at Re = 1000. For quantitative validation at moderate Reynolds numbers, simulations are configured as follows: the Mach number is set to Ma $\approx$ 0.18, and the computational domain is defined as $[25D, 7.5D, 7.5D]$ with D denoting the sphere diameter, which is set to 0.4 m in our experiments. The simulations use $K = 5$ and omit the smoothing profile to maximize sharpness. We report the time-averaged drag coefficient $C_d = F_D/(0.5\rho_0 v_0^2 A)$ over an extended temporal window, where $A = \pi D^2/4$ is the sphere's projected area and F_D is the time-averaged drag force. As summarized in Table 1, our computed C_d are in good agreement with literature data, with relative errors generally below 3 % for Re $\leq$ 1000. This confirms the accuracy of our method for predicting integral quantities in external aerodynamic flows at moderate Reynolds numbers. Note that for higher Reynolds number flows, the boundary layer demands significantly finer grid resolution, or alternatively, near-wall modeling is requisite to ensure accuracy—a factor that motivates the exclusion of high-Reynolds-number simulations from the present tests.

6.1.4 Flow over an SD7003 airfoil. Lastly, we validate the scheme's capability to capture sensitive transitional flow physics via simulations of flow over an SD7003 airfoil at Re = 60000 (Ma = 0.1, $\alpha = 8°$). This flow regime is distinguished by a laminar separation bubble (LSB) on the suction surface—a feature highly susceptible to numerical dissipation. To eliminate grid-related discrepancies, we employ the identical high-order unstructured mesh utilized in Vermeire et al. [2017]. While the current architectural constraints of PyFR preclude the hardware optimizations detailed in Section 4, the setup enables a direct, topology-independent verification of the formulation's spectral accuracy. The simulation uses $K = 5$ with over-integration and Gauss-Legendre points to mitigate aliasing errors. Statistics are collected over a duration of 30 characteristic time units (t_c), subsequent to an initial spin-up period of $30t_c$. Fig. 13 depicts the instantaneous velocity field, clearly resolving the flow separation, vortex roll-up, and reattachment processes. Quantitatively, Fig. 14 presents a comparison of the time-averaged pressure coefficient ($-C_p$) and integral force coefficients against well-established numerical and experimental data. The results exhibit consistent agreement with other solvers, confirming that our scheme yields reliable solutions for sensitive external aerodynamic flows.

6.2 Simulation results

With the numerical accuracy and performance of our scheme validated through the rigorous quantitative benchmarks outlined above, we can deploy the solver for applications demanding both compelling visual effects and physical fidelity. The following results show that our method satisfies tight resource constraints while preserving the multiscale details needed for high-fidelity simulations.

6.2.1 Leapfrogging vortex. We begin with the canonical interaction of two coaxial vortex rings, a simulation scenario for visualizing the impact of numerical dissipation. Excessive artificial viscosity would

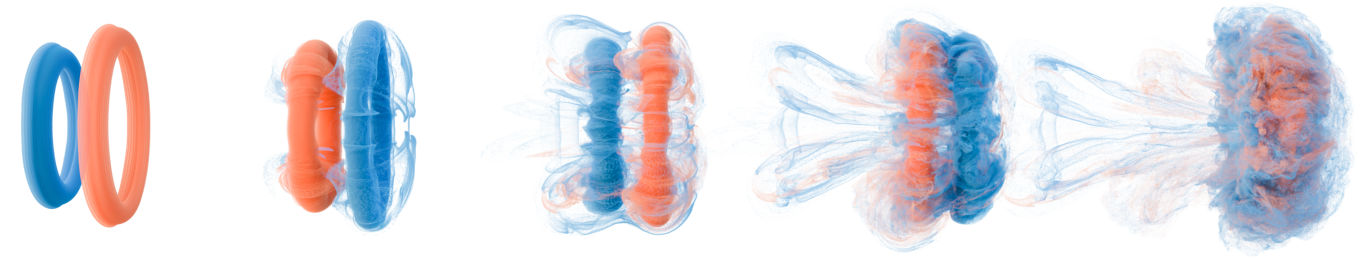


Fig. 15. **Visual simulation of leapfrogging vortex rings.** Left to right, the time sequence captures the rear vortex ring (blue) accelerating through the expanding frontal ring (orange). The minimal numerical dissipation of our scheme enables this complex topological transition to evolve spontaneously into chaotic, turbulent breakdown (far right), driven purely by intrinsic physical instabilities of nonlinear vortex dynamics.

Fig. 16. **Visual simulation of a maneuvering fighter jet.** To demonstrate the scheme's reliability in handling complex geometries, we conducted simulations of a fighter aircraft across a range of angles of attack (AoA = $8^\circ, 16^\circ, 22^\circ$). Flow field visualization was achieved via particle tracers released from the leading edges, as well as the trailing edges of both the canards and main wings. The top row presents the particle rendering results, whereas the bottom row depicts the corresponding time-averaged streamline distributions. These results clearly demonstrate that our scheme captures the intricate multiscale vortex dynamics and the complex mutual interactions between the canard and main wing vortices. Notably, the high-fidelity resolution of small-scale flow structures induces intense turbulent mixing, as clearly evidenced by the highly mixed particle distributions trailing the main wing.

dampen the vortex cores and suppress the characteristic leapfrogging behavior of the system. As illustrated in Fig. 15, our scheme faithfully captures the contraction-expansion cycles inherent to the flow, driven purely by the governing fluid dynamics equations. Moreover, it faithfully resolves the natural transition from coherent flow structures to fine-scale turbulence without supplementary models to compensate for numerical dissipation, confirming strong energy-preserving characteristics.

6.2.2 Vortex dynamics of a maneuvering jet. We further test the scheme's capability to resolve intricate flow physics via simulations of a fighter jet featuring a close-coupled canard-wing configuration, a layout defined by intricate geometric features and coupled with highly complex tip vortex dynamics. In such a geometric setup, vortex dynamics is dominated by the intricate mutual interactions between the canard and main wing. As illustrated in Fig. 16, our scheme explicitly resolves these coherent vortex structures and their mutual inductive interactions. Through the deployment of flow tracers at the leading edges of both the canards and main wing, we visualize how the upstream canard vortex energizes the boundary layer atop the main wing. Specifically, our method qualitatively captures the characteristic spiral lift vortices emanating from the sharp leading edges, as well as the squeezing effect exerted by the canard vortex on the fuselage. Notably, at high angles of attack (AoA $\geq 16^\circ$), the solver stably reproduces the onset of vortex breakdown and subsequent transition to turbulence. Furthermore, its inherent capability to resolve small-scale vortices enables the clear visualization of intense turbulent mixing over the main wing.

6.2.3 Simulating RTX 5090 airflow on an RTX 5090. To demonstrate the solver's capability for high-resolution simulations of complex geometries with moving components, we present a self-referential benchmark: simulating the aerodynamic performance of an NVIDIA

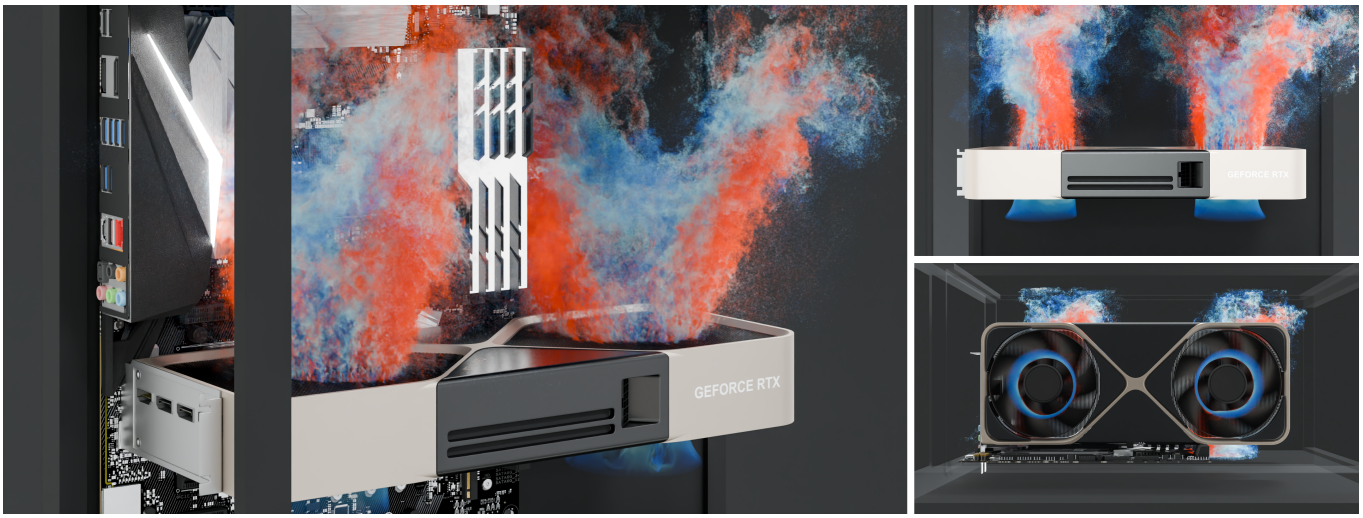


Fig. 17. **Simulating RTX 5090 airflow on an RTX 5090.** We present a self-aspirated aerodynamic simulation of an NVIDIA RTX 5090, with the entire computation executed on the GPU hardware under test. Unlike conventional wind-tunnel simulations reliant on artificial inflow conditions, the airflow herein is driven exclusively by the rotating fans, a setup that mandates precise resolution of the fan blades to avoid non-physical flow stall phenomena. The airflow is subsequently forced to traverse the dense array of heat sink fins; in such scenarios, lower-order solvers typically exhibit elevated numerical dissipation, leading to artificial flow blockage artifacts. Our solver faithfully resolves the two turbulent jets emanating from the heat sink fins, demonstrating low-dissipation characteristics. This simulation consumes only 10 GiB of memory, which fits comfortably within the memory capacity of the GPU under investigation.

RTX 5090 GPU equipped with rotating fans, with the entire computation executed directly on the GPU hardware in question, see Fig. 17. This result validates the solver's handling of dynamic components and multiscale flow interactions. The shape exhibits extreme geometric scale disparities, spanning from the ≈ 45 cm-scale chassis down to sub-millimeter gaps between heat sink fins. Airflow is driven entirely by two fans rotating at 1500 RPM; ring-shaped flow tracers are deployed beneath the fans to visualize the airflow as it is forced through the explicitly resolved heat sink fins toward the GPU backplate. The resulting high-speed airflow jets then impinge on and interact with intricate motherboard components before being confined by the top chassis panel. A total physical time duration of 1 s is simulated. This configuration requires resolving the intense interactions between fan-driven airflow and the dense array of heat sink fins, thereby imposing stringent demands on solver stability, numerical dissipation control, and memory efficiency. Note that the fin pitch is adjusted to 2.6 mm to match the resolvable spatial resolution within a computationally feasible time budget; even so, the simulation successfully captures the key qualitative aerodynamic characteristics of this complex dynamic system.

6.2.4 Extreme-scale urban airflow dynamics. Traditionally, city-scale airflow simulations that retain meter-scale spatial detail have typically required large-scale computing resources. To test the operational limits of our solver, we conduct a large-scale wind environment simulation of a 3 km-long segment of the Manhattan skyline (see Fig. 18), using 1.87 billion solution points with an average SP spacing of approximately 1 m. Thanks to the high-order properties and compact memory footprint of our solver, simulations of this scale fit within and run on a single NVIDIA RTX PRO 6000 Blackwell workstation GPU. For this highly under-resolved case, we use a 6th-order hybrid configuration ($K = 6$, $\lambda = 0$) to preserve low numerical dissipation while maintaining robustness. The computational domain is discretized into a structured grid with a resolution of $2982 \times 480 \times 1302$ solution points. For a reference inflow velocity of 10 m/s, simulating 200 s of physical time takes merely 5.5 h of computation time. The sharp geometric features of the skyscrapers induce extensive vortex shedding that blankets the entire ground-level domain, forming multiscale vortex structures that are clearly visualized in the multiscale illustrations of Fig. 18. This scenario imposes rigorous demands on the solver's numerical dissipation control, under-resolved flow stability, computational efficiency, and low GPU memory consumption. To the best of our knowledge, no existing LBM variant attains this level of memory efficiency to accommodate simulations of this scale on a *single* workstation GPU, while preserving comparably low numerical dissipation.

6.2.5 Real-time simulation of organic geometries. Benefiting from its inherent high-order properties, our solver is capable of real-time simulation of turbulent flows replete with intricate vortical structures. Here, we demonstrate the solver's capacity to yield high-fidelity results while maintaining real-time performance in Fig. 19, enabled by its high raw computational throughput. We simulate flow over a rotating sphere with a complex porous organic surface structure, while also showcasing the robustness of the immersed boundary method in handling intricate geometric topologies. The entire simulation, featuring $450 \times 150 \times 150$ solution points and a $K = 5$ configuration, was completed in 17 seconds of wall-clock time, with rendering implemented offline. This level of performance supports real-time generation of physically consistent turbulent flows around complex objects, shortening the iteration cycle for

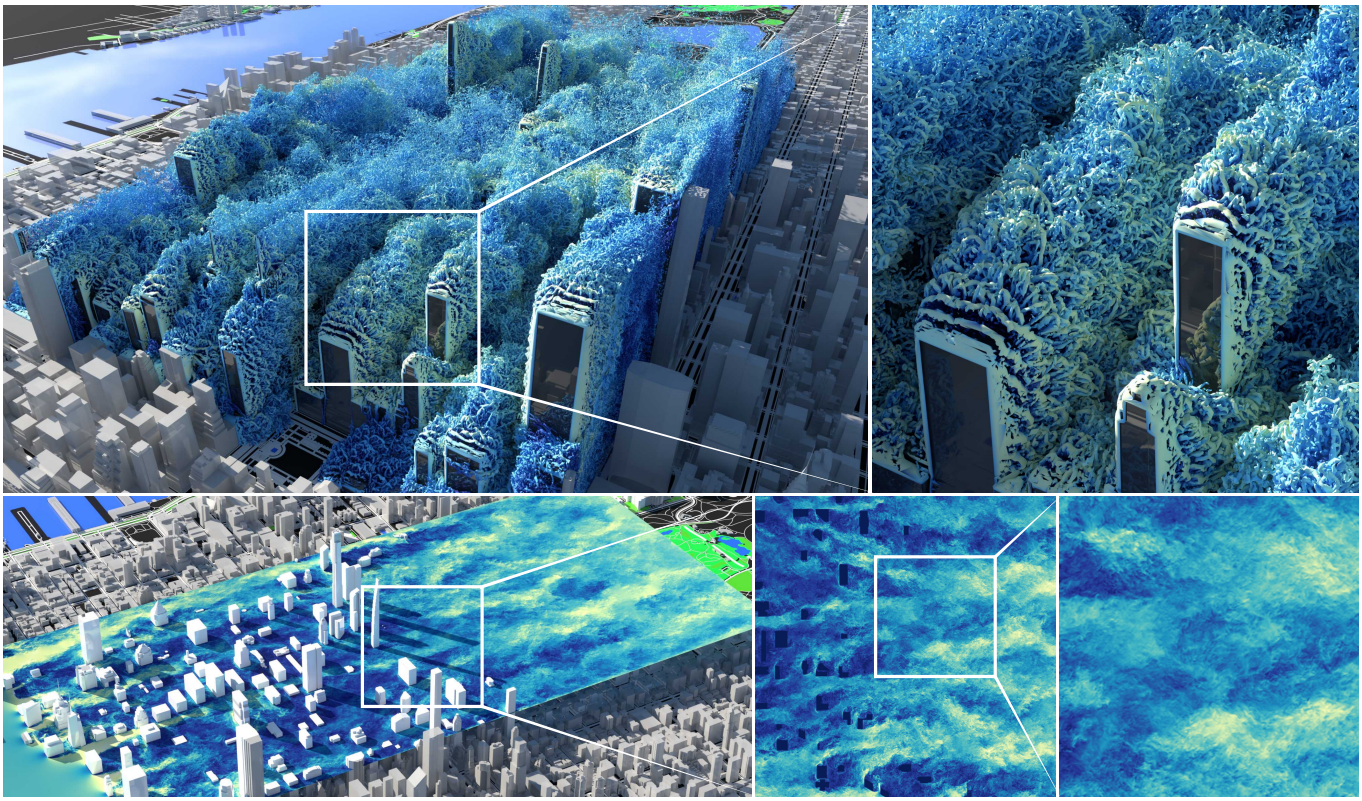


Fig. 18. **Extreme-scale urban airflow dynamics.** This large-scale urban wind scenario pushes the throughput and memory limits of single-GPU simulation. Utilizing our high-performance 6th-order configuration ($K = 6$), the computational domain is discretized into *1.87 billion solution points*, an exceptionally high resolution density for single-device simulations. Executed on an NVIDIA RTX PRO 6000 Blackwell GPU, the solver sustains a computational throughput of over 10 billion solution-point updates per second, with the entire simulation completed in 5.5 hours. The top panels illustrate the rich Q-value isosurface structures surrounding the buildings, while the bottom panels depict instantaneous velocity magnitude fields at heights of 140 m via a multiscale visualization approach, revealing intricate, oil-painting-like turbulent flow structures that maintain sharp definition and rich structural details.

visual flow design and facilitating time-critical applications such as reinforcement learning-based policy training.

Readers are encouraged to refer to the supplementary video for animations of all key demonstration results presented in this work.

6.3 Discussion

There are several technical aspects that merit further discussion.

High-order advantage. Under the constraints of computational resources, GPU memory and wall-clock time both limit achievable fidelity. Compact storage mitigates memory limits: fewer bytes per point allow more grid points to fit on the device. High-order methods address both limitations: higher spectral efficiency means fewer points suffice for a given fidelity [Wang et al. 2013], and since the computational cost of 3D unsteady flow on uniform grids scales as $O(N^4)$, reducing required resolution yields a quartic cost saving. This resolving efficiency, distinct from formal convergence rates, shifts the Pareto frontier in Fig. 10: KPM-FR resolves finer flow structures with fewer computational resources.

Beyond kernel timings. While speedup ratios are widely adopted as performance metrics, they often conflate algorithmic innovations with implementation-level optimizations; Roofline analysis [Williams et al. 2009] offers a more transparent framework by relating floating-point throughput to arithmetic intensity (FLOP/byte). As shown in Fig. 20 (left), all benchmarked kernels saturate the memory bandwidth ceiling, confirming that each implementation is memory-bound. In this regime, lower arithmetic intensity is advantageous: it provides greater margin relative to the compute ceiling and, as the right panel demonstrates across three GPU generations with progressively tighter performance envelopes, maintains kernels in a purely memory-bound state even on more resource-constrained hardware. KPM-FR lies between C17-LBM and the more computationally intensive HOME-LBM; its bandwidth saturation is enabled by a highly coalesced predictor and a corrector whose halo traffic is almost entirely absorbed by the L2 cache. FluidX3D is also highly optimized, with its higher arithmetic intensity originating from its `FP32/FP16C` storage scheme. All schemes remain well below the compute-bound ceiling, preserving headroom for future arithmetic-intensive extensions.

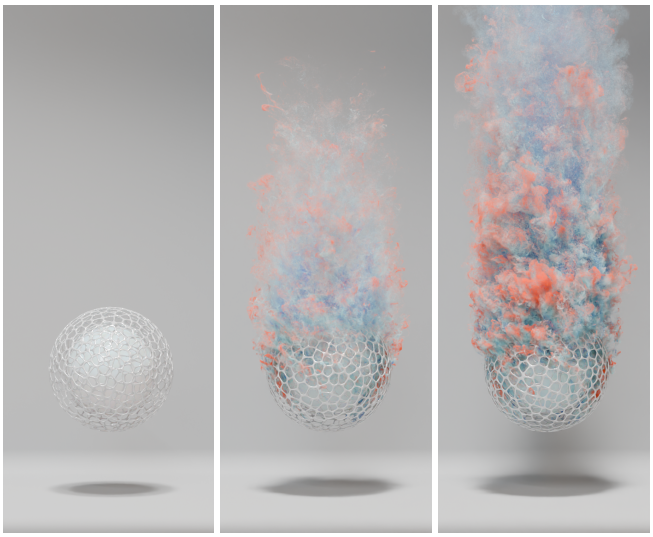


Fig. 19. **Real-time simulation of a rotating organic sphere.** We simulate aerodynamic flow past a porous organic structure in real time to demonstrate the high raw computational throughput of our scheme. The simulation sequence captures the flow evolution from a quiescent initial state (left) to a fully developed turbulent wake (right). Notably, the entire simulation, with a grid resolution of $450 \times 150 \times 150$ solution points, was completed in 17 seconds. It should be noted that only the simulation itself is executed in real time, whereas rendering is implemented offline.

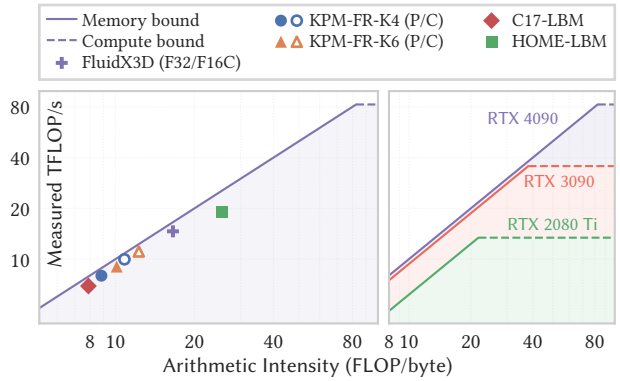


Fig. 20. **Roofline analysis on RTX 4090.** Left: measured throughput versus arithmetic intensity for the GPU kernels studied in this work; the solid diagonal and dashed horizontal lines denote the memory-bandwidth and peak-compute ceilings, respectively. Filled and open markers distinguish predictor (P) and corrector (C) stages. Right: examples of hardware Roofline envelopes for three GPU generations.

Notably, FR exhibits an intrinsic computational advantage over classical FVM schemes such as MUSCL-TVD [Toro 2009; van Leer 1979]: the element abstraction provides inherent data and compute locality. (1) Data are naturally grouped by element, avoiding artificial tiling and hiding load latency through intra-element parallelism. (2) Arithmetic intensity is largely decoupled from common-flux evaluation, while intra-element operations reduce to structured linear algebra on a fixed reference element. (3) Reconstruction is strictly element-local, substantially narrowing the performance gap between structured and unstructured meshes: similar cache-friendly kernels can be applied to diverse mesh topologies.

6.4 Limitations

Our method possesses certain limitations.

Governing equations and temporal properties. As with standard GKS formulations [Xu 2004], the kinetic evolution is truncated to the NS level, thus restricting our scheme to continuum flow regimes where $\tau \ll \Delta t$. Our particular formulation further restricts formal temporal accuracy to second order, which is regained in the continuum limit ($\tau \to 0$). Nevertheless, across all tested configurations, the solver maintains consistent stability and fidelity under this second-order temporal integration. The present framework supports extension to higher-order temporal accuracy, e.g., via CERK-type schemes [Gassner et al. 2011], which we leave for future work. Consequently, the proposed method is best suited for low-speed, *convection-dominated flows* at moderate to high Reynolds numbers, which we believe cover a wide range of practical applications and many computer graphics scenarios.

Time-step constraints. The explicit FR formulation is subject to CFL-based time-step constraints (Eq. (28)), requiring more time steps per unit physical time than LBM. In our benchmarks, however, its higher per-step throughput and resolving efficiency more than compensate for this, as shown in Fig. 11 and discussed in Section 6.3.

Geometric handling and tensor-product grids. The present optimized implementation targets uniform tensor-product grids, and the immersed-boundary treatment method (Section 5.2) provides geometric flexibility. However, resolving critical flow features still requires sufficient local grid refinement or near-wall modeling. Two complementary approaches are feasible: body-fitted unstructured meshes with additional element types, for which we have obtained preliminary results, and multi-resolution refinement within the tensor-product setting. Both are supported by FR's element-local architecture yet pose unresolved challenges.

7 Conclusions and Outlook

In this work, we present KPM-FR, a kinetic and spatially high-order numerical scheme for low-Mach-number fluid simulation, built upon a novel moment-based predictor-corrector paradigm within the flux reconstruction framework. This inherently compact formulation minimizes memory overhead by operating directly on kinetic moments and reduces local kinetic evolution to efficient arithmetic operations, yielding a favorable frontier of simulation fidelity, computational throughput, and memory footprint. It further extends the performance envelope of high-order fluid simulation methods, whose exceptional resolving power can now be paired with a near-minimal memory footprint and bandwidth-saturating throughput approaching the hardware limits of single-GPU execution.

KPM-FR advances the shared ambitions of the CFD and computer graphics communities by demonstrating that these objectives

can be simultaneously achieved within a unified, hardware-aware framework. It also opens new research avenues for deploying high-order methods on commodity GPU: from mixed-precision strategies that trade fidelity for throughput in latency-sensitive workflows, to compressible extensions, thermal modeling, and adaptive spatial resolution that relax current governing-equation and resolution limitations for predictive industrial simulations. Furthermore, the core design philosophy of KPM-FR—tailoring algorithmic architectures to modern hardware platforms—offers a pathway to other element types, governing equations, and numerical scheme families. This approach ultimately embodies the ethos encapsulated by the adage "There's plenty of room at the Top" [Leiserson et al. 2020].

Acknowledgments

The authors thank the anonymous reviewers for their insightful comments and constructive suggestions. This work was supported by New Generation Artificial Intelligence-National Science and Technology Major Project (No. 2025ZD0124002) and ShanghaiTech University. 3D assets were provided by peterkissdesign for Fig. 1, 3DWAR for Fig. 16, mistjs, and amefizthepubgpro for Fig. 17. City data for Fig. 18 were derived from OpenStreetMap contributors (ODbL 1.0).

References

Yoshiaki Abe, Issei Morinaka, Takanori Haga, Taku Nonomura, Hisaichi Shibata, and Koji Miyaji. 2018. Stable, Non-Dissipative, and Conservative Flux-Reconstruction Schemes in Split Forms. *J. Comput. Phys.* 353 (Jan. 2018), 193–227. doi:10.1016/j.jcp.2017.10.007

Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, and L. Donatella Marini. 2002. Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. *SIAM J. Numer. Anal.* 39, 5 (Jan. 2002), 1749–1779. doi:10.1137/S0036142901384162

Kartikey Asthana and Antony Jameson. 2015. High-Order Flux Reconstruction Schemes with Minimal Dispersion and Dissipation. *Journal of Scientific Computing* 62, 3 (March 2015), 913–944. doi:10.1007/s10915-014-9882-5

Utkarsh Ayachit, Andrew C. Bauer, Ben Boeckel, Berk Geveci, Kenneth Moreland, Patrick O'Leary, and Tom Osika. 2021. Catalyst Revised: Rethinking the ParaView in Situ Analysis and Visualization API. In *High Performance Computing*. 484–494. doi:10.1007/978-3-030-90539-2_33

P. Bailey, J. Myre, S.D.C. Walsh, D.J. Lilja, and M.O. Saar. 2009. Accelerating Lattice Boltzmann Fluid Flow Simulations Using Graphics Processors. In *2009 International Conference on Parallel Processing*. IEEE, Vienna, 550–557. doi:10.1109/ICPP.2009.38

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. 2018. Fast Winding Numbers for Soups and Clouds. *ACM Transactions on Graphics* 37, 4 (July 2018), 43:1–43:12. doi:10.1145/3197517.3201337

F. Bassi and S. Rebay. 1997. A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations. *J. Comput. Phys.* 131, 2 (March 1997), 267–279. doi:10.1006/jcph.1996.5572

F. Bassi and S. Rebay. 2000. A High Order Discontinuous Galerkin Method for Compressible Turbulent Flows. In *Discontinuous Galerkin Methods*, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick, Bernardo Cockburn, George E. Karniadakis, and Chi-Wang Shu (Eds.). Vol. 11. Springer Berlin Heidelberg, Berlin, Heidelberg, 77–88. doi:10.1007/978-3-642-59721-3_4

Andrea D. Beck, Thomas Bolemann, David Flad, Hannes Frank, Gregor J. Gassner, Florian Hindenlang, and Claus-Dieter Munz. 2014. High-Order Discontinuous Galerkin Spectral Element Methods for Transitional and Turbulent Flow Simulations. *International Journal for Numerical Methods in Fluids* 76, 8 (Nov. 2014), 522–548. doi:10.1002/fld.3943

P. L. Bhatnagar, E. P. Gross, and M. Krook. 1954. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Physical Review* 94, 3 (May 1954), 511–525. doi:10.1103/PhysRev.94.511

Blender Online Community. 2025. *Blender - a 3D Modelling and Rendering Package*. Stichting Blender Foundation, Amsterdam.

Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-Time Smoke Animation with Vortex Sheet Meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. Eurographics Association, Goslar, DEU, 87–95.

G. Capdeville. 2023. Gas Kinetic Principles in Navier-Stokes Finite-Volume Solvers. *J. Comput. Phys.* 488 (Sept. 2023), 112162. doi:10.1016/j.jcp.2023.112162

Shiyi Chen and Gary D. Doolen. 1998. LATTICE BOLTZMANN METHOD FOR FLUID FLOWS. *Annual Review of Fluid Mechanics* 30, 1 (Jan. 1998), 329–364. doi:10.1146/annurev.fluid.30.1.329

Nian-Sheng Cheng. 2009. Comparison of Formulas for Drag Coefficient and Settling Velocity of Spherical Particles. *Powder Technology* 189, 3 (Feb. 2009), 395–398. doi:10.1016/j.powtec.2008.07.006

Alexandre Joel Chorin. 1967. A Numerical Method for Solving Incompressible Viscous Flow Problems. *J. Comput. Phys.* 2, 1 (Aug. 1967), 12–26. doi:10.1016/0021-9991(67)90037-X

Alexandre Joel Chorin. 1968. Numerical Solution of the Navier-Stokes Equations. *Math. Comp.* 22, 104 (1968), 745–762. jstor:2004575 doi:10.2307/2004575

S.Y. Chou and D. Baganoff. 1997. Kinetic Flux–Vector Splitting for the Navier–Stokes Equations. *J. Comput. Phys.* 130, 2 (Jan. 1997), 217–230. doi:10.1006/jcph.1996.5579

Jonathan R. Clausen. 2013. Entropically Damped Form of Artificial Compressibility for Explicit Simulation of Incompressible Flow. *Physical Review E* 87, 1 (Jan. 2013), 013309. doi:10.1103/PhysRevE.87.013309

Bernardo Cockburn and Chi-Wang Shu. 1998. The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems. *SIAM J. Numer. Anal.* 35, 6 (Dec. 1998), 2440–2463. doi:10.1137/S0036142997316712

Yitong Deng, Hong-Xing Yu, Diyang Zhang, Jiajun Wu, and Bo Zhu. 2023. Fluid Simulation on Neural Flow Maps. *ACM Transactions on Graphics* 42, 6 (Dec. 2023), 1–21. doi:10.1145/3618392

Dominique d'Humières. 2002. Multiple–Relaxation–Time Lattice Boltzmann Models in Three Dimensions. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 360, 1792 (March 2002), 437–451. doi:10.1098/rsta.2001.0955

Todd F. Dupont and Yingjie Liu. 2003. Back and Forth Error Compensation and Correction Methods for Removing Errors Induced by Uneven Gradients of the Level Set Function. *J. Comput. Phys.* 190, 1 (Sept. 2003), 311–324. doi:10.1016/S0021-9991(03)00276-6

T. Dzanic, S. S. Girimaji, and F. D. Witherden. 2022. Partially-Averaged Navier–Stokes Simulations of Turbulence within a High-Order Flux Reconstruction Framework. *J. Comput. Phys.* 456 (May 2022), 110992. doi:10.1016/j.jcp.2022.110992

Essex Edwards and Robert Bridson. 2014. Detailed Water with Coarse Grids: Combining Surface Meshes and Adaptive Discontinuous Galerkin. *ACM Transactions on Graphics* 33, 4 (July 2014), 136:1–136:9. doi:10.1145/2601097.2601167

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual Simulation of Smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 15–22. doi:10.1145/383259.383260

Marco A. Ferrari, Waine B. de Oliveira Jr., Alan Lugarini, Admilson T. Franco, and Luiz A. Hegele Jr. 2023. A Graphic Processing Unit Implementation for the Moment Representation of the Lattice Boltzmann Method. *International Journal for Numerical Methods in Fluids* 95, 7 (2023), 1076–1089. doi:10.1002/fld.5185

Joel H. Ferziger, Milovan Perić, and Robert L. Street. 2020. *Computational Methods for Fluid Dynamics*. Springer International Publishing, Cham. doi:10.1007/978-3-319-99693-6

Gregor Gassner, Frieder Lörcher, and Claus-Dieter Munz. 2007. A Contribution to the Construction of Diffusion Fluxes for Finite Volume and Discontinuous Galerkin Schemes. *J. Comput. Phys.* 224, 2 (June 2007), 1049–1063. doi:10.1016/j.jcp.2006.11.004

Gregor J. Gassner, Michael Dumbser, Florian Hindenlang, and Claus-Dieter Munz. 2011. Explicit One-Step Time Discretizations for Discontinuous Galerkin and Finite Volume Schemes Based on Local Predictors. *J. Comput. Phys.* 230, 11 (May 2011), 4232–4247. doi:10.1016/j.jcp.2010.10.024

Martin Geier, Andreas Greiner, and Jan G. Korvink. 2006. Cascaded Digital Lattice Boltzmann Automata for High Reynolds Number Flow. *Physical Review E* 73, 6 (June 2006), 066705. doi:10.1103/PhysRevE.73.066705

M. Geier, A. Greiner, and J. G. Korvink. 2009. A Factorized Central Moment Lattice Boltzmann Method. *European Physical Journal Special Topics* 171, 1 (April 2009), 55–61. doi:10.1140/epjst/e2009-01011-1

Martin Geier, Konstantin Kutscher, Martin Schönherr, Anna Wellmann, Sören Peters, Hussein Alihussein, Jan Linxweiler, and Manfred Krafczyk. 2025. VirtualFluids – Open Source Parallel LBM Solver. *Computer Physics Communications* 317 (Dec. 2025), 109810. doi:10.1016/j.cpc.2025.109810

Martin Geier, Stephan Lenz, Martin Schönherr, and Manfred Krafczyk. 2021. Under-Resolved and Large Eddy Simulations of a Decaying Taylor–Green Vortex with the Cumulant Lattice Boltzmann Method. *Theoretical and Computational Fluid Dynamics* 35, 2 (April 2021), 169–208. doi:10.1007/s00162-020-00555-7

Martin Geier, Andrea Pasquali, and Martin Schönherr. 2017. Parametrization of the Cumulant Lattice Boltzmann Method for Fourth Order Accurate Diffusion Part I: Derivation and Validation. *J. Comput. Phys.* 348 (Nov. 2017), 862–888. doi:10.1016/j.jcp.2017.05.040

Martin Geier and Martin Schönherr. 2017. Esoteric Twist: An Efficient in-Place Streaming Algorithmus for the Lattice Boltzmann Method on Massively Parallel Hardware. *Computation* 5, 2 (March 2017), 19. doi:10.3390/computation5020019

Table 2. **Simulation configurations and timing statistics.** Solution points correspond to the total number of degrees of freedom within the simulation domain. Memory consumption is quantified based on the memory footprint of the solver's core data structures. Throughput denotes the effective update rate of solution points, expressed in millions of solution-point updates per second (MUPS). The hardware platform designated as RTX PRO 6000 corresponds to the NVIDIA RTX PRO 6000 Blackwell GPU.

Case	*Device*	K	*Solution points*	ν (m^2/s)	*Memory* (GiB)	*Wall-clock per step* (ms)	*# Steps per frame*	*Throughput* (MUPS)
Fig. 1	RTX 4090	4	$2000 \times 332 \times 800$	8×10^{-7}	20.81	66.19	380	8024.90
Fig. 15	RTX 4090	5	$600 \times 300 \times 300$	6.26×10^{-6}	2.11	5.96	73	9060.40
Fig. 16 (AoA = 8°)	RTX 4090	4	$1360 \times 340 \times 680$	4×10^{-6}	12.30	36.95	129	8509.66
Fig. 16 (AoA = 16°)	RTX 4090	4	$1360 \times 340 \times 680$	4×10^{-6}	12.30	36.95	134	8509.66
Fig. 16 (AoA = 22°)	RTX 4090	4	$1360 \times 340 \times 680$	4×10^{-6}	12.30	36.95	138	8509.66
Fig. 17	RTX 5090	4	$932 \times 700 \times 384$	2×10^{-7}	9.80	18.63	833	13447.21
Fig. 18	RTX PRO 6000	6	$2982 \times 480 \times 1302$	1×10^{-7}	72.90	162.94	182	11437.53
Fig. 19	RTX 4090	5	$450 \times 150 \times 150$	8×10^{-6}	0.39	1.18	28	8580.51

Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. 2015. The Cumulant Lattice Boltzmann Equation in Three Dimensions: Theory and Validation. *Computers & Mathematics with Applications* 70, 4 (Aug. 2015), 507–547. doi:10.1016/j.camwa.2015.05.001

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-Scale Fluid Simulation Using Velocity-Vorticity Domain Decomposition. *ACM Trans. Graph.* 31, 6 (Nov. 2012), 148:1–148:9. doi:10.1145/2366145.2366167

Harold Grad. 1949. On the Kinetic Theory of Rarefied Gases. *Communications on Pure and Applied Mathematics* 2, 4 (Dec. 1949), 331–407. doi:10.1002/cpa.3160020403

Zhaoli Guo, Hongwei Liu, Li-Shi Luo, and Kun Xu. 2008. A Comparative Study of the LBE and GKS Methods for 2D near Incompressible Laminar Flows. *J. Comput. Phys.* 227, 10 (May 2008), 4955–4976. doi:10.1016/j.jcp.2008.01.024

Zhaoli Guo, Kun Xu, and Ruijie Wang. 2013. Discrete Unified Gas Kinetic Scheme for All Knudsen Number Flows: Low-speed Isothermal Case. *Physical Review E* 88, 3 (Sept. 2013), 033305. doi:10.1103/PhysRevE.88.033305

Francis H. Harlow and J. Eddie Welch. 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids* 8, 12 (Dec. 1965), 2182–2189. doi:10.1063/1.1761178

Jan S. Hesthaven and Tim Warburton. 2008. *Nodal Discontinuous Galerkin Methods*. Texts in Applied Mathematics, Vol. 54. Springer New York, New York, NY. doi:10.1007/978-0-387-72067-8

J. C. R. Hunt, A. A. Wray, and P. Moin. 1988. Eddies, Streams, and Convergence Zones in Turbulent Flows. In *Proceedings of the 1988 Summer Program*. Center for Turbulence Research, Stanford University, 193–208.

H. T. Huynh. 2007. A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods. In *18th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, Miami, Florida. doi:10.2514/6.2007-4079

H. T. Huynh. 2009. A Reconstruction Approach to High-Order Schemnes Including Discontinuous Galerkin for Diffusion. In *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Orlando, Florida. doi:10.2514/6.2009-403

A. Jameson, P. E. Vincent, and P. Castonguay. 2012. On the Non-Linear Stability of Flux Reconstruction Schemes. *Journal of Scientific Computing* 50, 2 (Feb. 2012), 434–445. doi:10.1007/s10915-011-9490-6

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The Affine Particle-in-Cell Method. *ACM Transactions on Graphics* 34, 4 (July 2015), 51:1–51:10. doi:10.1145/2766996

T. A. Johnson and V. C. Patel. 1999. Flow Past a Sphere up to a Reynolds Number of 300. *Journal of Fluid Mechanics* 378 (Jan. 1999), 19–70. doi:10.1017/S0022112098003206

George Karniadakis and Spencer Sherwin. 2005. *Spectral/Hp Element Methods for Computational Fluid Dynamics*. Oxford University Press. doi:10.1093/acprof:oso/9780198528692.001.0001

Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. 2009. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM. *Graphical Models* 71, 4 (July 2009), 153–167. doi:10.1016/j.gmod.2009.02.002

Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008. Wavelet Turbulence for Fluid Simulation. *ACM Transactions on Graphics* 27, 3 (Aug. 2008), 1–6. doi:10.1145/1360612.1360649

Jiaqing Kou, Saumitra Joshi, Aurelio Hurtado-de-Mendoza, Kunal Puri, Charles Hirsch, and Esteban Ferrer. 2022. Immersed Boundary Method for High-Order Flux Reconstruction Based on Volume Penalization. *J. Comput. Phys.* 448 (Jan. 2022), 110721. doi:10.1016/j.jcp.2021.110721

Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. 2017. *The Lattice Boltzmann Method: Principles and Practice*. Springer International Publishing, Cham. doi:10.1007/978-3-319-44649-3

Pierre Lallemand and Li-Shi Luo. 2000. Theory of the Lattice Boltzmann Method: Dispersion, Dissipation, Isotropy, Galilean Invariance, and Stability. *Physical Review E* 61, 6 (June 2000), 6546–6562. doi:10.1103/PhysRevE.61.6546

Jonas Latt and Bastien Chopard. 2006. Lattice Boltzmann Method with Regularized Pre-Collision Distribution Functions. *Mathematics and Computers in Simulation* 72, 2-6 (Sept. 2006), 165–168. doi:10.1016/j.matcom.2006.05.017

Moritz Lehmann. 2022. FluidX3D.

Charles E. Leiserson, Neil C. Thompson, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, and Tao B. Schardl. 2020. There's Plenty of Room at the Top: What Will Drive Computer Performance after Moore's Law? *Science* 368, 6495 (June 2020), eaam9744. doi:10.1126/science.aam9744

Ji Li, Chengwen Zhong, and Sha Liu. 2020b. High-Order Kinetic Flow Solver Based on the Flux Reconstruction Framework. *Physical Review E* 102, 4 (Oct. 2020), 043306. doi:10.1103/PhysRevE.102.043306

Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020a. Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling. *ACM Transactions on Graphics* 39, 4 (Aug. 2020). doi:10.1145/3386569.3392400

Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient Kinetic Simulation of Two-Phase Flows. *ACM Transactions on Graphics* 41, 4 (July 2022), 1–17. doi:10.1145/3528223.3530132

Wei Li, Tongtong Wang, Zherong Pan, Xifeng Gao, Kui Wu, and Mathieu Desbrun. 2023. High-Order Moment-Encoded Kinetic Simulation of Turbulent Flows. *ACM Transactions on Graphics* 42, 6 (Dec. 2023), 1–13. doi:10.1145/3618341

Wei Li, Xiaoming Wei, and Arie Kaufman. 2003. Implementing Lattice Boltzmann Computation on Graphics Hardware. *The Visual Computer* 19, 7 (Dec. 2003), 444–456. doi:10.1007/s00371-003-0210-6

Chunlei Liang, Christopher Cox, and Michael Plesniak. 2013. A Comparison of Computational Efficiencies of Spectral Difference Method and Correction Procedure via Reconstruction. *J. Comput. Phys.* 239 (April 2013), 138–146. doi:10.1016/j.jcp.2013.01.001

Meng-Sing Liou and Christopher J. Steffen. 1993. A New Flux Splitting Scheme. *J. Comput. Phys.* 107, 1 (July 1993), 23–39. doi:10.1006/jcph.1993.1122

Haixiang Liu, Nathan Mitchell, Mridul Aanjaneya, and Eftychios Sifakis. 2016. A Scalable Schur-Complement Fluids Solver for Heterogeneous Compute Platforms. *ACM Trans. Graph.* 35, 6 (Dec. 2016), 201:1–201:12. doi:10.1145/2980179.2982430

Mengyun Liu, Kai Bai, and Xiaopei Liu. 2025. A Hybrid Near-Wall Model for Kinetic Simulation of Turbulent Boundary Layer Flows. *ACM Transactions on Graphics* 44, 4 (Aug. 2025), 1–24. doi:10.1145/3730829

Mengyun Liu and Xiaopei Liu. 2023. A Parametric Kinetic Solver for Simulating Boundary-Dominated Turbulent Flow Phenomena. *ACM Transactions on Graphics* 42, 6 (Dec. 2023), 1–20. doi:10.1145/3618313

W. Liu, Z.J. Liu, Z.L. Zhang, C.J. Teo, and C. Shu. 2023. Grad's Distribution Function for 13 Moments-Based Moment Gas Kinetic Solver for Steady and Unsteady Rarefied Flows: Discrete and Explicit Forms. *Computers & Mathematics with Applications* 137 (May 2023), 112–125. doi:10.1016/j.camwa.2023.02.006

Xiaopei Liu, Wai-Man Pang, Jing Qin, and Chi-Wing Fu. 2014. Turbulence Simulation by Adaptive Multi-Relaxation Lattice Boltzmann Modeling. *IEEE Transactions on Visualization and Computer Graphics* 20, 2 (Feb. 2014), 289–302. doi:10.1109/TVCG.2012.303

Chaoyang Lyu, Kai Bai, Yiheng Wu, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2023. Building a Virtual Weakly-Compressible Wind Tunnel Testing Facility. *ACM Transactions on Graphics* 42, 4 (Aug. 2023), 1–20. doi:10.1145/3592394

Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2021. Fast and Versatile Fluid-Solid Coupling for Turbulent Flow Simulation. *ACM Transactions on Graphics* 40, 6 (Dec. 2021), 1–18. doi:10.1145/3478513.3480493

Chao Ma, Jie Wu, and Liming Yang. 2022. A Novel High-Order Solver for Simulation of Incompressible Flows Using the Flux Reconstruction Method and Lattice Boltzmann Flux Solver. *Computers & Fluids* 248 (Nov. 2022), 105673. doi:10.1016/j.compfluid.2022.105673

Stefano Markidis, Vyacheslav Olshevsky, Chaitanya Prasad Sishtla, Steven W. D. Chien, Erwin Laure, and Giovanni Lapenta. 2018. PolyPIC: The Polymorphic-Particle-in-Cell Method for Fluid-Kinetic Coupling. *Frontiers in Physics* 6 (Oct. 2018). doi:10.3389/fphy.2018.00100

Gianmarco Mengaldo, Daniele De Grazia, Freddie Witherden, Antony Farrington, Peter Vincent, Spencer Sherwin, and Joaquim Peiro. 2014. A Guide to the Implementation of Boundary Conditions in Compact High-Order Methods for Compressible Aerodynamics. In *7th AIAA Theoretical Fluid Mechanics Conference*. American Institute of Aeronautics and Astronautics, Atlanta, GA. doi:10.2514/6.2014-2923

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. 2009. Energy-Preserving Integrators for Fluid Animation. *ACM Transactions on Graphics* 28, 3 (July 2009), 38:1–38:8. doi:10.1145/1531326.1531344

Mohammad Sina Nabizadeh, Ritoban Roy-Chowdhury, Hang Yin, Ravi Ramamoorthi, and Albert Chern. 2024. Fluid Implicit Particles on Coadjoint Orbits. *ACM Transactions on Graphics* 43, 6 (Nov. 2024), 270:1–270:38. doi:10.1145/3687970

Mohammad Sina Nabizadeh, Stephanie Wang, Ravi Ramamoorthi, and Albert Chern. 2022. Covector Fluids. *ACM Transactions on Graphics* 41, 4 (July 2022), 113:1–113:16. doi:10.1145/3528223.3530120

NVIDIA. 2025. CUDA Toolkit Documentation 13.1. https://docs.nvidia.com/cuda/.

Suhas Patankar. 1980. *Numerical Heat Transfer and Fluid Flow*. CRC Press.

Tobias Pfaff, Nils Thuerey, Jonathan Cohen, Sarah Tariq, and Markus Gross. 2010. Scalable Fluid Simulation Using Anisotropic Turbulence Particles. In *ACM SIGGRAPH Asia 2010 Papers (SIGGRAPH ASIA '10)*. Association for Computing Machinery, New York, NY, USA, 1–8. doi:10.1145/1866158.1866196

Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian Vortex Sheets for Animating Fluids. *ACM Transactions on Graphics* 31, 4 (July 2012), 112:1–112:8. doi:10.1145/2185520.2185608

P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. 2002. Vortex Methods for Direct Numerical Simulation of Three-Dimensional Bluff Body Flows: Application to the Sphere at *Re*=300, 500, and 1000. *J. Comput. Phys.* 178, 2 (May 2002), 427–463. doi:10.1006/jcph.2002.7035

Stephen B. Pope. 2015. *Turbulent Flows* (1. publ., 12. print ed.). Cambridge Univ. Press, Cambridge.

Ziyin Qu, Minchen Li, Fernando De Goes, and Chenfanfu Jiang. 2022. The Power Particle-in-Cell Method. *ACM Transactions on Graphics* 41, 4 (July 2022), 118:1–118:13. doi:10.1145/3528223.3530066

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and Conservative Fluids Using Bidirectional Mapping. *ACM Transactions on Graphics* 38, 4 (July 2019), 128:1–128:12. doi:10.1145/3306346.3322945

Karthik Raveendran, Chris Wojtan, and Greg Turk. 2011. Hybrid Smoothed Particle Hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '11)*. Association for Computing Machinery, New York, NY, USA, 33–42. doi:10.1145/2019406.2019411

Xiaodong Ren, Kun Xu, Wei Shyy, and Chunwei Gu. 2015. A Multi-Dimensional High-Order Discontinuous Galerkin Method Based on Gas Kinetic Theory for Viscous Flow Computations. *J. Comput. Phys.* 292 (July 2015), 176–193. doi:10.1016/j.jcp.2015.03.031

Will Schroeder, Ken Martin, and Bill Lorensen. 2006. *The Visualization Toolkit (4th Ed.)*. Kitware.

Michael S. Selig. 1995. *Summary of Low Speed Airfoil Data*. SoarTech Publications.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An Unconditionally Stable MacCormack Method. *J. Sci. Comput.* 35, 2-3 (June 2008), 350–371. doi:10.1007/s10915-007-9166-4

C. Shu, Y. Wang, C. J. Teo, and J. Wu. 2014. Development of Lattice Boltzmann Flux Solver for Simulation of Incompressible Flows. *Advances in Applied Mathematics and Mechanics* 6, 4 (Aug. 2014), 436–460. doi:10.4208/aamm.2014.4.s2

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 121–128. doi:10.1145/311535.311548

Y. Sun, C. Shu, C.J. Teo, Y. Wang, and L.M. Yang. 2015. Explicit Formulations of Gas-Kinetic Flux Solver for Simulation of Incompressible and Compressible Viscous Flows. *J. Comput. Phys.* 300 (Nov. 2015), 492–519. doi:10.1016/j.jcp.2015.07.060

Yu Sun, Chang Shu, Liming Yang, and C. J. Teo. 2016. A Switch Function-Based Gas-Kinetic Scheme for Simulation of Inviscid and Viscous Compressible Flows. *Advances in Applied Mathematics and Mechanics* 8, 5 (Oct. 2016), 703–721. doi:10.4208/aamm.2015.m1071

Kasia Świrydowicz, Noel Chalmers, Ali Karakus, and Tim Warburton. 2019. Acceleration of Tensor-Product Operations for High-Order Finite Element Methods. *Int. J. High Perform. Comput. Appl.* 33, 4 (July 2019), 735–757. doi:10.1177/1094342018816368

E. F. Toro. 2009. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction* (3rd ed ed.). Springer, Dordrecht New York.

W. Trojak, R. Watson, and F.D. Witherden. 2022. Hyperbolic Diffusion in Flux Reconstruction: Optimisation through Kernel Fusion within Tensor-Product Elements. *Computer Physics Communications* 273 (April 2022), 108235. arXiv:2107.14027 [cs] doi:10.1016/j.cpc.2021.108235

Pedro Valero-Lara, Jeffrey Vetter, John Gounley, and Amanda Randles. 2023. Moment Representation of Regularized Lattice Boltzmann Methods on NVIDIA and AMD GPUs. In *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W '23)*. Association for Computing Machinery, New York, NY, USA, 1697–1704. doi:10.1145/3624062.3624250

Bram van Leer. 1979. Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method. *J. Comput. Phys.* 32, 1 (July 1979), 101–136. doi:10.1016/0021-9991(79)90145-1

Bram Van Leer and Shohei Nomura. 2005. Discontinuous Galerkin for Diffusion. In *17th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, Toronto, Ontario, Canada. doi:10.2514/6.2005-5108

B.C. Vermeire, F.D. Witherden, and P.E. Vincent. 2017. On the Utility of GPU Accelerated High-Order Methods for Unsteady Flow Simulations: A Comparison with Industry-Standard Tools. *J. Comput. Phys.* 334 (April 2017), 497–521. doi:10.1016/j.jcp.2016.12.049

P. E. Vincent, P. Castonguay, and A. Jameson. 2011. A New Class of High-Order Energy Stable Flux Reconstruction Schemes. *J. Sci. Comput.* 47, 1 (April 2011), 50–72. doi:10.1007/s10915-010-9420-z

Z.J. Wang, Krzysztof Fidkowski, Rémi Abgrall, Francesco Bassi, Doru Caraeni, Andrew Cary, Herman Deconinck, Ralf Hartmann, Koen Hillewaert, H.T. Huynh, Norbert Kroll, Georg May, Per-Olof Persson, Bram Van Leer, and Miguel Visbal. 2013. High-Order CFD Methods: Current Status and Perspective. *International Journal for Numerical Methods in Fluids* 72, 8 (July 2013), 811–845. doi:10.1002/fld.3767

Xiaoming Wei, Wei Li, Klaus Mueller, and Arie E. Kaufman. 2004. The Lattice-Boltzmann Method for Simulating Gaseous Phenomena. *IEEE Transactions on Visualization and Computer Graphics* 10, 2 (March 2004), 164–176. doi:10.1109/TVCG.2004.1260768

Steffen Weißmann and Ulrich Pinkall. 2010. Filament-Based Smoke with Vortex Shedding and Variational Reconnection. *ACM Transactions on Graphics* 29, 4 (July 2010), 115:1–115:12. doi:10.1145/1778765.1778852

Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM* 52, 4 (April 2009), 65–76. doi:10.1145/1498765.1498785

Freddie D. Witherden, Peter E. Vincent, Will Trojak, Yoshiaki Abe, Amir Akbarzadeh, Semih Akkurt, Mohammad Alhawwary, Lidia Caros, Tarik Dzanic, Giorgio Giangaspero, Arvind S. Iyer, Antony Jameson, Marius Koch, Niki Loppi, Sambit Mishra, Rishit Modi, Gonzalo Sáez-Mischlich, Jin Seok Park, Brian C. Vermeire, and Lai Wang. 2025. PyFR v2.0.3: Towards Industrial Adoption of Scale-Resolving Simulations. *Computer Physics Communications* 311 (June 2025), 109567. doi:10.1016/j.cpc.2025.109567

Xiaoyu Xiao, Ding Lin, Yiheng Wu, Kai Bai, and Xiaopei Liu. 2025. Simulating Two-Phase Fluid-Rigid Interactions With an Overset-Grid Kinetic Solver. *IEEE Transactions on Visualization and Computer Graphics* 31, 10 (Oct. 2025), 8397–8412. doi:10.1109/TVCG.2025.3570570

Kun Xu. 2001. A Gas-Kinetic BGK Scheme for the Navier–Stokes Equations and Its Connection with Artificial Dissipation and Godunov Method. *J. Comput. Phys.* 171, 1 (July 2001), 289–335. doi:10.1006/jcph.2001.6790

Kun Xu. 2004. Discontinuous Galerkin BGK Method for Viscous Flow Equations: One-Dimensional Systems. *SIAM Journal on Scientific Computing* 25, 6 (Jan. 2004), 1941–1963. doi:10.1137/S1064827502416113

Kun Xu and Juan-Chen Huang. 2010. A Unified Gas-Kinetic Scheme for Continuum and Rarefied Flows. *J. Comput. Phys.* 229, 20 (Oct. 2010), 7747–7764. doi:10.1016/j.jcp.2010.06.032

L.M. Yang, C. Shu, and J. Wu. 2014. A Simple Distribution Function-Based Gas-Kinetic Scheme for Simulation of Viscous Incompressible and Compressible Flows. *J. Comput. Phys.* 274 (Oct. 2014), 611–632. doi:10.1016/j.jcp.2014.06.033

L.M. Yang, C. Shu, W.M. Yang, and Y. Wang. 2017. A Simplified Circular Function–Based Gas Kinetic Scheme for Simulation of Incompressible Flows. *International Journal for Numerical Methods in Fluids* 85, 10 (Dec. 2017), 583–598. doi:10.1002/fld.4398

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An Advection-Reflection Solver for Detail-Preserving Fluid Simulation. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–8. doi:10.1145/3197517.3201324

Chao Zhang, Qibing Li, Song Fu, and Z.J. Wang. 2018. A Third-Order Gas-Kinetic CPR Method for the Euler and Navier–Stokes Equations on Triangular Meshes. *J. Comput. Phys.* 363 (June 2018), 329–353. doi:10.1016/j.jcp.2018.02.040

Xinxin Zhang, Robert Bridson, and Chen Greif. 2015. Restoring the Missing Vorticity in Advection-Projection Fluid Solvers. *ACM Transactions on Graphics* 34, 4 (July 2015), 52:1–52:8. doi:10.1145/2766982

Algorithm 1 Fused predictor: half-step moment accumulation.

Input: $\mathbf{U}^{(n)}$ (macroscopic state at t^n; from DRAM)
Output: $\mathbf{U}^{(p)}$, $\mathbf{\Pi}^{(p)}$ (stored to DRAM)
procedure PREDICTOR
LOAD $\mathbf{U}^{(n)}$ from DRAM ▷ *x-aligned 1-D slices*
$\mathbf{U}^{(p)} \leftarrow \mathbf{U}^{(n)}$; $\mathbf{\Pi}^{\mathrm{acc}} \leftarrow \mathbf{0}$
DIRECTIONAL PASS
for $d \in \{x, y, z\}$ **do**
  $\nabla_d \mathbf{U}^{(n)} \leftarrow \mathbf{D}\,\mathbf{U}^{(n)}$ ▷ *unrolled at compile time*
  for each SP $s = 0, \ldots, K-1$ **do**
    $\mathbf{U}_d^{\triangle}[s] \leftarrow \mathbf{U}^{(n)}[s] - u_c\, h\, \nabla_d \mathbf{U}^{(n)}[s]$
    $\Delta_d m_{abc} \leftarrow \left[m_{abc}(\mathbf{U}_d^{\triangle}[s]) - m_{abc}(\mathbf{U}^{(n)}[s])\right]/u_c$
    Accumulate $\Delta_d m_{abc}$ into $\mathbf{U}^{(p)}[s]$ and $\mathbf{\Pi}^{\mathrm{acc}}[s]$
  STORE to SRAM; SYNC; LOAD next-d slices
POINTWISE FINALIZATION
for each SP $s = 0, \ldots, K-1$ **do**
  $\mathbf{\Pi}^{(p)}[s] \leftarrow A\big(\mathbf{\Pi}^{\mathrm{acc}}[s] - m_{abc}(\mathbf{U}^{(p)}[s]) + m_{abc}(\mathbf{U}^{(n)}[s])\big)$
  $\Pi_{\alpha\alpha}^{(p)}[s] \leftarrow \Pi_{\alpha\alpha}^{(p)}[s] - \mathrm{tr}\,\mathbf{\Pi}^{(p)}[s]/3$
STORE $\mathbf{U}^{(p)}$, $\mathbf{\Pi}^{(p)}$ to DRAM

Algorithm 2 Fused corrector: flux evaluation and update.

Input: $\mathbf{U}^{(n)}$, $\mathbf{U}^{(p)}$, $\mathbf{\Pi}^{(p)}$ (from DRAM)
Output: $\mathbf{U}^{(n+1)}$ (stored to DRAM)
procedure CORRECTOR
LOAD interior $\mathbf{U}^{(p)}$, $\mathbf{\Pi}^{(p)}$ from DRAM ▷ *x-aligned 1-D slices*
LOAD halo $\mathbf{U}^{(p)}$, $\mathbf{\Pi}^{(p)}$ from DRAM ▷ *Gauss-Lobatto, served by L2*
$\delta\mathbf{U} \leftarrow \mathbf{0}$
DIRECTIONAL PASS
for $d \in \{x, y, z\}$ **do**
  for each SP $s = 0, \ldots, K-1$ **do**
    $\mathbf{F}_d^{\mathrm{eq}}[s] \leftarrow \mathbf{F}_d^{\mathrm{eq}}(\mathbf{U}^{(p)}[s])$
    $\mathbf{F}_d^{\mathrm{neq}}[s] \leftarrow \mathbf{F}_d^{\mathrm{neq}}(\mathbf{\Pi}^{(p)}[s])$
  $\mathbf{F}_{d,L}^{*}$, $\mathbf{F}_{d,R}^{*} \leftarrow$ common flux at d-faces
  $\delta\mathbf{U} \leftarrow \delta\mathbf{U} + \mathcal{S}_d[\mathbf{U}^{(p)}]$ ▷ *split-form equilibrium*
    $+\ \mathbf{Q}\,\mathbf{F}_d^{\mathrm{neq}} + \mathbf{C}\big(\mathbf{F}_d^{*} - \mathbf{R}\,\mathbf{F}_d^{\mathrm{eq}}\big)$
  STORE to SRAM; SYNC; LOAD next-d slices
UPDATE
LOAD $\mathbf{U}^{(n)}$ from DRAM
$\mathbf{U}^{(n+1)} \leftarrow \mathbf{U}^{(n)} - 2\,\Delta t\,\delta\mathbf{U}/L$ ▷ *midpoint rule*
STORE $\mathbf{U}^{(n+1)}$ to DRAM

Junwei Zhou, Duowen Chen, Molin Deng, Yitong Deng, Yuchen Sun, Sinan Wang, Shiying Xiong, and Bo Zhu. 2024. Eulerian-Lagrangian Fluid Simulation on Particle Flow Maps. *ACM Transactions on Graphics* 43, 4 (July 2024), 1–20. arXiv:2405.09672 [cs] doi:10.1145/3658180

Ye Zhou, Fernando F. Grinstein, Adam J. Wachtor, and Brian M. Haines. 2014. Estimating the Effective Reynolds Number in Implicit Large-Eddy Simulation. *Physical Review E* 89, 1 (Jan. 2014), 013303. doi:10.1103/PhysRevE.89.013303

O. C. Zienkiewicz and R. L. Taylor. 2013. *The Finite Element Method: Its Basis and Fundamentals.* Butterworth-Heinemann.

A Fused Kernel Pseudocode

Algorithms 1 and 2 summarize the two per-stage GPU kernels used to realize the workflow in Section 3.8 under the IO-aware design of Section 4. The superscript (p) denotes the predicted half-step block vector produced by the predictor, corresponding to $\mathbf{U}(h)$ and $\mathbf{\Pi}(h)$ in Section 3.6. The operator $\mathcal{S}_d[\cdot]$ denotes the split-form derivative of the equilibrium flux in direction d described in Section 5.1.

B Low-Mach Maxwellian Equilibrium and Its Moments

For isothermal flows in the low-Mach regime, the Maxwellian equilibrium distribution function g introduced in Section 3.2 is expanded to the second order in $\mathbf{u}$ [Guo et al. 2013; Krüger et al. 2017] as:

$$g(\boldsymbol{\xi}) = \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\frac{\|\boldsymbol{\xi}\|^2}{2RT}\right) \times \left[1 + \frac{\boldsymbol{\xi}\cdot\mathbf{u}}{RT} + \frac{1}{2(RT)^2}(\boldsymbol{\xi}\cdot\mathbf{u})^2 - \frac{\|\mathbf{u}\|^2}{2RT}\right], \tag{33}$$

where all symbols follow the definitions in Section 3.2 and D denotes the number of spatial dimensions. We define $m_{abc} = \left\langle \xi_x^a \xi_y^b \xi_z^c\, g \right\rangle$ for non-negative integers a, b, and c. The first-order moments are:

$$m_{100} = \rho u, \qquad m_{010} = \rho v, \qquad m_{001} = \rho w.$$

The second-order moments are related to the momentum flux:

$$m_{200} = \rho(u^2 + RT), \quad m_{020} = \rho(v^2 + RT), \quad m_{002} = \rho(w^2 + RT),$$
$$m_{110} = \rho uv, \quad m_{011} = \rho vw, \quad m_{101} = \rho uw.$$

Finally, the third-order moments are:

$$m_{300} = 3\rho RTu, \quad m_{030} = 3\rho RTv, \quad m_{003} = 3\rho RTw,$$
$$m_{210} = \rho RTv, \quad m_{120} = \rho RTu, \quad m_{201} = \rho RTw,$$
$$m_{021} = \rho RTw, \quad m_{102} = \rho RTu, \quad m_{012} = \rho RTv,$$

with $m_{111} = 0$. These moments supply all the equilibrium data required by the predictor formulation (Eqs. (22) and (23)). For the common-flux evaluations (Eqs. (26) and (27)), the half-range moments are additionally required. We define $m_{abc}^{\pm} = \left\langle \xi_x^a \xi_y^b \xi_z^c\, g \right\rangle_{\pm}$ and $M_{abc}^{\pm} = \left\langle \xi_x^a \xi_y^b \xi_z^c\, f^{\mathrm{neq}} \right\rangle_{\pm}$ as the half-range equilibrium and non-equilibrium moments. Their closed-form values are:

$$m_{000}^{\pm} = \rho\,(1/2 \pm u/C_1), \qquad M_{000}^{\pm} = 0,$$
$$m_{100}^{\pm} = \rho\left[u/2 \pm (u^2 + 2RT)/(2C_1)\right], \qquad M_{100}^{\pm} = \pm\Pi_{xx}/(2C_1),$$
$$m_{010}^{\pm} = v\, m_{000}^{\pm}, \qquad M_{010}^{\pm} = \pm\Pi_{xy}/C_1,$$
$$m_{001}^{\pm} = w\, m_{000}^{\pm}, \qquad M_{001}^{\pm} = \pm\Pi_{xz}/C_1,$$
$$m_{200}^{\pm} = \rho\left(u^2 + RT \pm C_2 u\right)/2, \qquad M_{200}^{\pm} = \Pi_{xx}/2,$$
$$m_{110}^{\pm} = \rho v\,(u/2 \pm C_3), \qquad M_{110}^{\pm} = \Pi_{xy}/2,$$
$$m_{101}^{\pm} = \rho w\,(u/2 \pm C_3), \qquad M_{101}^{\pm} = \Pi_{xz}/2,$$

where $C_1 = \sqrt{2\pi RT}$, $C_2 = 2\sqrt{2RT/\pi}$, $C_3 = \sqrt{RT/(2\pi)}$. In our implementation, these are treated as compile-time expressions that directly map $\mathbf{U}$ or $\mathbf{\Pi}$ to the higher-order moments.