Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling

WEI LI, ShanghaiTech University/SIMIT/UCAS
YIXIN CHEN, ShanghaiTech University/DGene
MATHIEU DESBRUN, ShanghaiTech/Caltech
CHANGXI ZHENG, Columbia University
XIAOPEI LIU, ShanghaiTech University

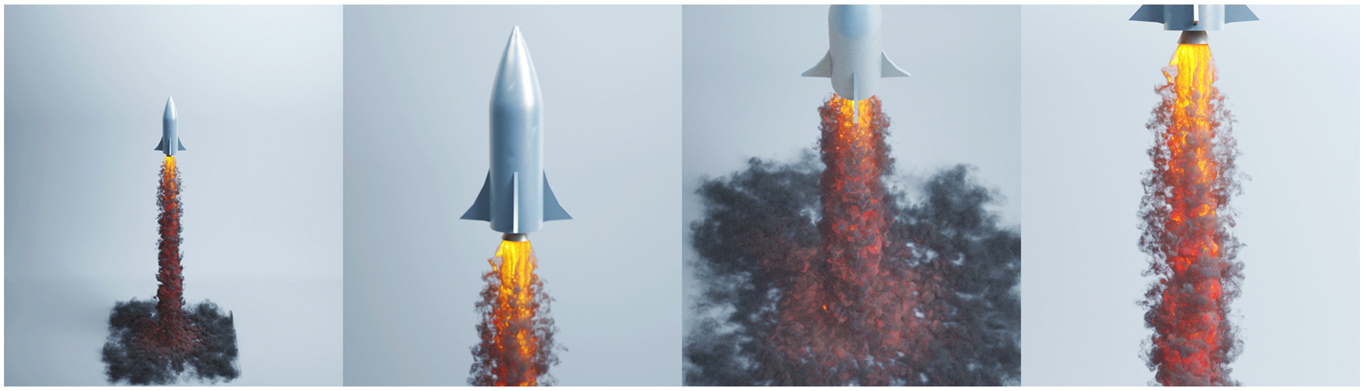

Fig. 1. **Rocket launching.** Our novel solver can accurately simulate fluid flows of either laminar or turbulent nature in an efficient and scalable manner. Here, a large-scale simulation of a rocket launch (computed with 4 GPUs on a grid of size 500×1200×500, 3.7 hours of computation for 1s of simulation); two-way coupling automatically propels the rocket in the air while generating a turbulent wake affecting the entire simulation domain (thermal effects are ignored).

Despite their cinematic appeal, turbulent flows involving fluid-solid coupling remain a computational challenge in animation. At the root of this current limitation is the numerical dispersion from which most accurate Navier-Stokes solvers suffer: proper coupling between fluid and solid often generates artificial dispersion in the form of local, parasitic trains of velocity oscillations, eventually leading to numerical instability. While successive improvements over the years have led to conservative and detail-preserving fluid integrators, the dispersive nature of these solvers is rarely discussed despite its dramatic impact on fluid-structure interaction. In this paper, we introduce a novel low-dissipation and low-dispersion fluid solver that can simulate two-way coupling in an efficient and scalable manner, even for turbulent flows. In sharp contrast with most current CG approaches, we construct our solver from a kinetic formulation of the flow derived from statistical mechanics. Unlike existing lattice Boltzmann solvers, our approach leverages *high-order moment relaxations* as a key to controlling both dissipation and dispersion of the resulting scheme. Moreover, we combine our new fluid solver with the immersed boundary method to easily handle fluid-solid coupling through time adaptive simulations. Our kinetic solver is highly parallelizable by nature, making it ideally suited for implementation on single- or multi-GPU computing platforms. Extensive comparisons with existing solvers on synthetic tests and real-life experiments are used to highlight the multiple advantages of our work over traditional and more recent approaches, in terms of accuracy, scalability, and efficiency.

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

Additional Key Words and Phrases: Fluid Simulation, Kinetic Theory, Lattice Boltzmann Method, Fluid-Solid Coupling, Immersed Boundary Method.

ACM Reference Format:
Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling. *ACM Trans. Graph.* 39, 4, Article 47 (July 2020), 20 pages. https://doi.org/10.1145/3386569.3392400

Authors' addresses: Wei Li, Yixin Chen, Xiaopei Liu, School of Information Science and Technology (Shanghai Engineering Research Center of Intelligent Vision and Imaging) of ShanghaiTech University, Shanghai, China; Wei Li is also affiliated with the Shanghai Institute of Microsystem and Information Technology (SIMIT) and the University of the Chinese Academy of Sciences (UCAS); Mathieu Desbrun, California Institute of Technology, Pasadena (CA), USA, on sabbatical at SIST in ShanghaiTech University, Shanghai, China; Changxi Zheng, Columbia University, New York (NY), USA.

1 INTRODUCTION

The intricate interplay between fluid and solids is a common occurrence all around us: from the fluttering of a dry leaf in the wind to the tumbling of a paper cup, the interaction, or *coupling*, between the motions of a fluid and of a solid is all the more complex than the flow is turbulent. Automatically capturing such complex phenomena being particularly desirable for movie production, numerous approaches to fluid-solid coupling have been proposed in the CG literature, most of them combining an incompressible Navier-Stokes solver with a rigid- or deformable-body simulator. Whether they rely on uniform grids [Carlson et al. 2004; Robinson-Mosher et al. 2008; Qiu et al. 2015], body-fitted meshes [Feldman et al. 2005; Klingner et al. 2006; Elcott et al. 2007; Clausen et al. 2013], cut-cells [Batty et al. 2007; Ng et al. 2009; Azevedo et al. 2016; Liu et al. 2015], or are based on particles [Akinci et al. 2012; Ihmsen et al. 2013; Band et al. 2018],

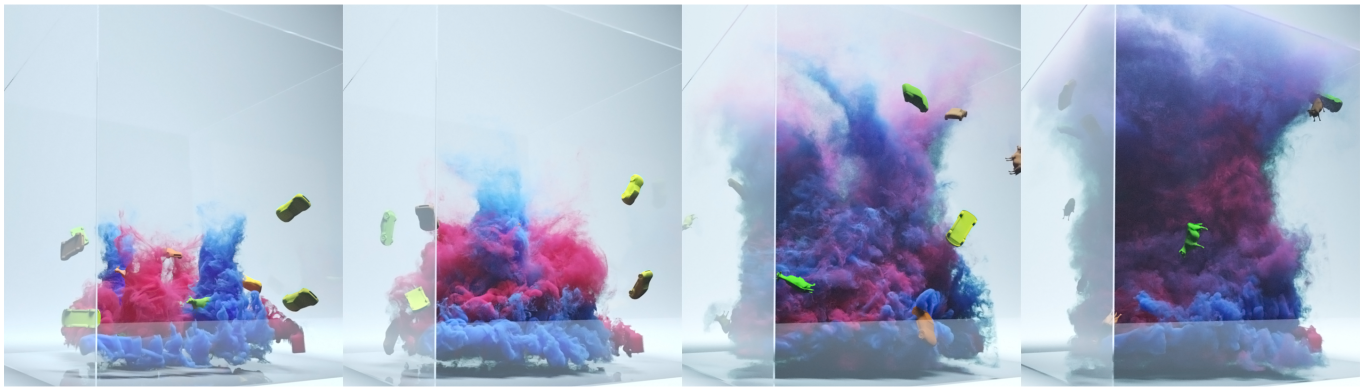

Fig. 2. **Tornado.** Four air inlets at the bottom of the simulation domain generate strong vortices, which quickly entangle to form a large tornado, entraining cars and cows. This two-way coupling simulation at Re = 80, 000 was performed on a 640×720×640 grid with a 4-GPU server; each frame required 139 sec.

existing simulation techniques for fluid-solid coupling all share the same limitation: they cannot handle *turbulent* flows robustly, as instabilities appear around fast moving solids. Moreover, the frequent need for a global pressure solve in these numerical methods is an additional hurdle towards obtaining efficient implementations allowing large-scale, high-resolution simulations.

In this paper, we start from a very different perspective. Instead of discretizing the usual incompressible Navier-Stokes equations, we adopt a more general *kinetic* description of fluid flows based on the continuous Boltzmann equation [Chen and Doolen 1998; Aidun and Clausen 2010]. However, we show that the existing lattice Boltzmann methods (LBM), used to discretize this statistical mechanics equation, systematically fail to properly treat high-order moments of the distribution functions. Instead, we introduce a variant of the central-moment relaxation model which provides a low-dissipation, low-dispersion fluid solver exceeding all current CG solvers on basic CFD obstacle courses such as 2D Taylor-Green vortex or vortex sheet tests. By combining this new kinetic solver with the Immersed Boundary (IB) method [Peskin 1972; Li et al. 2016] to achieve two-way fluid-solid coupling via time-adaptive LBM simulation, we offer a robust and efficient numerical approach that can not only offer realtime simulation for a 64×64×64 grid on a typical workstation (with an NVIDIA 2080Ti GPU), but also high-resolution simulations (e.g., 500×1200×500) on a compute cluster for large-scale phenomena through scalable parallel computing, see Fig. 1.

1.1 Related work

Flow simulation and fluid-solid coupling have a long history in both the Computer Graphics (CG) community and the Computational Fluid Dynamics (CFD) community as we now briefly review.

Turbulent flow simulation. Starting from the work of Foster and Metaxas [1996], fluid simulation quickly developed in graphics. Stam [1999] proposed an unconditionally stable semi-Lagrangian advection scheme, which unfortunately produced excessive numerical diffusion. In order to better capture vortices in flow simulations, higher order advection methods were then proposed, such as the BFECC scheme [Kim et al. 2005], the MacCormack scheme [Selle et al. 2008], and more recently, the advection-reflection scheme [Zehnder et al. 2018] and the "BiMocq^2" method [Qu et al. 2019], which were shown to better preserve vortical details in time. Although very efficient, these schemes rely on uniform grids, so extensions to octree [Losasso et al. 2004] and other forms of grids [Lentine et al. 2010; Zhu et al. 2013; Setaluri et al. 2014] were proposed, along with schemes designed for tetrahedral meshes [Klingner et al. 2006; Mullen et al. 2009; Ando et al. 2013]. Lagrangian particles have also been used to simulate fluids, based for instance on smoothed particle hydrodynamics [Desbrun and Gascuel 1996; Becker and Teschner 2007; Solenthaler and Pajarola 2009; Akinci et al. 2012; Ihmsen et al. 2014; Peer et al. 2015], but cannot capture turbulent flows for low particle counts due to their low order accuracy. Hybrid methods [Zhu and Bridson 2005; Raveendran et al. 2011; Jiang et al. 2015; Zhang et al. 2016] fare better in terms of accuracy, but are still most appropriate for slow, viscous fluids unless longer runtimes are allowed [Fu et al. 2017]. Vortex methods [Park and Kim 2005; Golas et al. 2012] excel at preserving vortical details, and solvers based on vortex particles [Selle et al. 2005], vortex filaments [Weißmann and Pinkall 2010] and vortex sheets [Brochu et al. 2012; Pfaff et al. 2012; Zhang and Bridson 2014; Zhang et al. 2015] have been shown effective and scalable. Two-way coupling using vortex methods remains, however, largely unexplored. Note that a number of approaches have also proposed the injection of artificial turbulence to flow simulation (see [Thürey et al. 2013], and recent data-driven approaches [Jeong et al. 2015; Chu and Thürey 2017; Xie et al. 2018]), but once again, fluid-solid coupling in this context is largely unexplored.

Fluid-solid coupling. Fluid-solid coupling has been actively studied in both CG and CFD. In graphics, various authors have discussed how to enforce proper solid boundary conditions (especially for pressure) to keep the fluid incompressible, which then provides the correct force to drive the motion of the solids. This has been achieved via voxelized boundary approximations [Takahashi et al. 2002; Génevaux et al. 2003; Robinson-Mosher et al. 2008; Azevedo et al. 2016], a fully-Eulerian method [Teng et al. 2016], or a rigid-fluid approach [Carlson et al. 2004], and can even be used for deformable shells [Guendelman et al. 2005]. However, all these methods are either inaccurate or unstable when fluids become turbulent: small timesteps inevitably increase *dispersion errors* for typical high-order accurate advection schemes (see Figs. 13 and 24), which eventually lead to inaccuracies. The use of boundary-conforming Eulerian simulation improves accuracy dramatically [Klingner et al. 2006], and has been adopted in most CFD approaches [Lv et al. 2010;

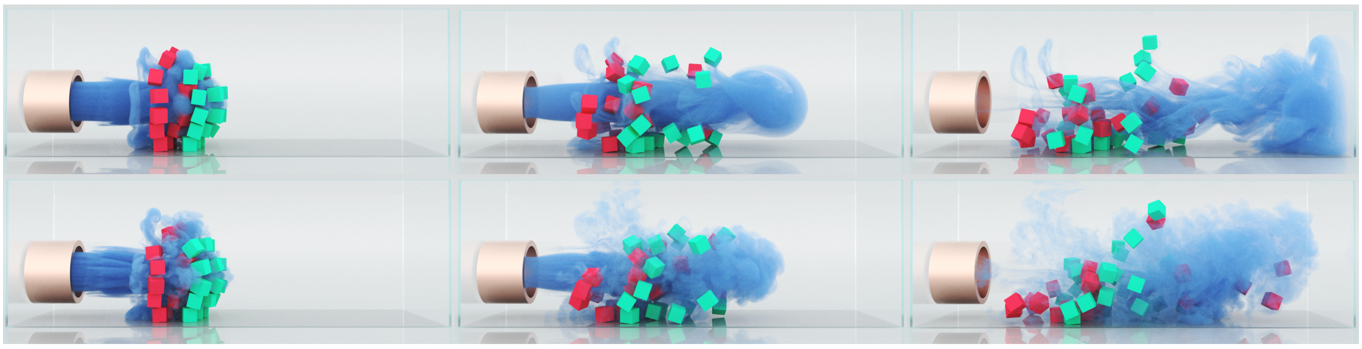

Fig. 3. **Blowing up stacked boxes.** In this two-way coupling example, an impulse jet through an air inlet is blowing away two stacks of boxes; passive smoke particles are injected to visualize the air flow interacting with the motion of the solid boxes. Two different Reynolds numbers are used: Re = 2, 666 (top), and Re = 8, 000 (bottom) where stronger turbulence (and thus, smoke plumes with higher frequencies) is observed.

Dai and Schmidt 2005]; but in the case of dynamic solids, frequent remeshing is required, which is computationally very expensive. Cut-cell-based approaches, where grids around solid boundaries are subdivided into boundary-conforming regions and finite-volume-based formulations are employed inside these regions, offering a good compromise [Roble et al. 2005; Batty et al. 2007; Ng et al. 2009; Gibou and Min 2012; Weber et al. 2015; Edwards and Bridson 2014], even for thin solids [Azevedo et al. 2016]. However, cut-cell-based approaches need to track fresh cells (cells that were inside the solid and are now in the fluid) and dead cells (in fluid before, now in solid), which can be costly. For particle-based fluid solvers, solids are often approximated by dense boundary particles to allow proper coupling [Colagrossi and Landrini 2003; Solenthaler and Pajarola 2008; Akinci et al. 2012; Schechter and Bridson 2012; de Goes et al. 2015; Bender and Koschier 2016; Band et al. 2017; Koschier and Bender 2017], but penalty forces have also been proposed to enforce boundary conditions [Becker et al. 2009; Ihmsen et al. 2013]. However, many of these methods do not ensure consistent pressure in the fluid domain and on solids [Band et al. 2018]. In hybrid grid-particle methods, boundary conditions are treated using grid-based techniques [Zhang et al. 2016; Fei et al. 2018; Hu et al. 2018]. The CFD community has also proposed a large number of numerical solutions to the problem of fluid-solid coupling [Hirt et al. 1974; Gibou and Min 2012; Lācis et al. 2016]. One of the most common and efficient approaches is the Immersed Boundary (IB) method [Peskin 2002], especially its diffused formulation [Li et al. 2016] as it does not need to track fresh cells and avoids pressure oscillation issues that sharp-interface variants suffer from [Seo and Mittal 2011]. Note that IB does not forgo the need for a global pressure solve.

Kinetic methods. While all the methods we discussed above are designed to approximate the traditional Navier-Stokes (NS) equations, a few alternatives have been proposed over the years, the most successful being arguably *kinetic methods* as used for fluid simulation in CG for over a decade [Thürey 2007; Thürey and Rüde 2009; Liu et al. 2014; Guo et al. 2017; Li et al. 2019; Li et al. 2020]. Instead of trying to numerically simulate the NS equations, kinetic approaches rely on statistical mechanics: they aim to simulate the continuous *Boltzmann transport equation*, which indicates the evolution in time of the probability density encoding the presence of fluid volumes at a given position with a given velocity. Key to the accuracy of kinetic methods is their collision (or relaxation) modeling. While early techniques relied on Bhatnagar-Gross-Krook (BGK) [Chen and Doolen 1998] and raw-moment multiple-relaxation-time (MRT) [d'Humières 2002] models, they suffered from low accuracy due to strong dispersion error and/or lack of Galilean invariance. The development of central-moment MRT models [Geier et al. 2006; De Rosis 2017; De Rosis and Luo 2019] dramatically increased both accuracy and stability. As kinetic methods do *not* need pressure projections, fluid-solid coupling can be quite easily included through a bounce-back treatment over curved solid surfaces [Mei et al. 1999], although tracking fresh and dead cells is now needed. Instead, the "diffused" IB method can be used [Feng and Michaelides 2004; Wu and Shu 2009, 2010; Li et al. 2016], for which no cell tracking is necessary. However, coupling for fast solid motions or turbulent flows is often unstable due to the penalty nature of IB: the resulting stiffness cannot be handled with the usual fixed normalized time steps of kinetic methods.

1.2 Overview of contributions

Given the bottleneck that pressure projection imposes on traditional CG methods offering accurate fluid-solid coupling, it is tempting to investigate other approaches. Kinetic approaches to fluid flows, such as the lattice Boltzmann method (LBM), offer an appealing alternative: due to its use of Boltzmann transport equations, the issue of properly discretizing advection is completely bypassed, and the absence of global pressure solves makes for extremely efficient parallel implementations; moreover, this approach can be easily combined with immersed boundary (IB) method [Peskin 1972; Li et al. 2016] to achieve one-way and two-way fluid-solid coupling, removing the difficulty of pressure boundary treatment. However, some really important practical issues remain. First, LBM does not currently compete with state-of-the-art CG solvers in terms of accuracy and preservation of turbulence details. In fact, the study of dissipative and dispersive errors in LBM is virtually inexistent (with the notable exception of [Xu and Sagaut 2011] in 2D). Second, the penalty-based IB method induces instability when interaction between fast-moving solids and/or turbulence occurs.

In this paper, we remedy these two issues to offer efficient, accurate and robust simulations of fluid-solid coupling in the presence of strong turbulence and large velocity variations. Our technical contributions are mostly two-fold:

- First, we show that the high-order relaxation rates of the non-orthogonal central-moment MRT model from [De Rosis and Luo 2019], typically dismissed as non-physical, are crucial to guarantee low-dissipation and low-dispersion of the fluid solver. We propose a simple linear regression to estimate relaxation rates on the fly, and show that the resulting modified LBM simulation outperforms existing incompressible fluid CG solvers in how it preserves turbulence details.
- Second, we introduce a universal mapping for LBM simulation, which formally links physical parameters with the typical normalized parameters used in LBM. This physical-to-simulated scaling is then used to provide time-adaptive simulation, ensuring stability and improved computational efficiency.

We will demonstrate the benefits (accuracy, low-dispersion, low-dissipation, and computational efficiency) of the resulting solver with tests and comparisons in 2D and 3D to current NS-based and kinetic methods, as well as comparisons with real-life flows.

2 PRIMER OF LATTICE BOLTZMANN SOLVERS

We start by briefly reviewing the main ideas of the lattice Boltzmann method (LBM) and its recent advances that impart several advantages over traditional Navier-Stokes solvers. Our goal in this section is not to provide a thorough introduction of LBM (we refer the reader to [Krüger et al. 2017] or our supplementary material instead), but to lay out the foundation for our contributions. Readers already familiar with LBM may directly jump to §2.4 instead.

2.1 Fluid model

From a macroscopic model... Behind most fluid simulation methods is the continuum model of fluid dynamics, namely the law of conservation of mass,

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho \boldsymbol{u}) = 0, \tag{1}$$

together with the Navier-Stokes (momentum) equation,

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla\cdot(\rho\, \boldsymbol{u}\otimes\boldsymbol{u}) = -\nabla p + \nabla\cdot\boldsymbol{\sigma} + \boldsymbol{F}. \tag{2}$$

This classical model views a fluid as a continuum body described by physical quantities such as density ρ, velocity $\boldsymbol{u}$, pressure p, shear stress tensor $\boldsymbol{\sigma}$ and external forces $\boldsymbol{F}$ per unit volume. These quantities are technically defined at each location $\boldsymbol{x}$ in the entire fluid domain; but they are *macroscopic*: they represent local averages that are observable in the fluid. In this sense, Eqs. (1) and (2) paint a *macroscopic* picture of fluid dynamics.

... To a microscopic description of fluid. In stark contrast to a macroscopic picture, one may also model fluid dynamics from the evolution of a large number of *microscopic* fluid particles, moving and colliding: after all, a fluid at rest is not merely a static continuum; it is teeming with molecules moving around all the time. Attempting a full description of those microscopic particles is intractable, so a practical approach, issued from statistical physics, considers instead *probability distributions* of fluid particles. Formally, it considers a distribution function, $f(\boldsymbol{x},\boldsymbol{v},t)$, that indicates the *probability* for a microscopic particle to be at position $\boldsymbol{x}$ at time t and moving with velocity $\boldsymbol{v}$. Note that here $\boldsymbol{v}$ is the possible velocity of a microscopic fluid particle, different from the macroscopic fluid velocity $\boldsymbol{u}$ used in the Navier-Stokes equation (2) — for example, even for a fluid at rest (i.e., $\boldsymbol{u}=0$), the velocity $\boldsymbol{v}$ of a fluid particle may be arbitrarily off from zero: local fluid velocities just average to zero locally.

... And finally, to a Boltzmann transport model. In the kinetic theory of statistical physics, the mesoscopic evolution of $f(\boldsymbol{x},\boldsymbol{v},t)$ is governed by the *continuous Boltzmann transport equation*,

$$\frac{\partial f}{\partial t} + \boldsymbol{v}\cdot\nabla f = \Omega(f) + \boldsymbol{F}\cdot\nabla_{\boldsymbol{v}} f, \tag{3}$$

where Ω is the so-called collision term, modeling the change of distribution function $f(\boldsymbol{x},\boldsymbol{v},t)$ due to particle collisions. This operator can be effectively modeled as a relaxation process of $f(\boldsymbol{x},\boldsymbol{v},t)$ towards its local equilibrium state: without external forces, a fluid over time tends to become distributed homogeneously. A well-known model of this relaxation process, used in many LBM works, is the Bhatnagar-Gross-Krook (BGK [Bhatnagar et al. 1954]) model,

$$\Omega(f) = -(f-\bar{f})/\tau, \tag{4}$$

where τ is the relaxation time, i.e., the averaged time between successive collisions determining how fast the equilibrium is being reached, which is related to the kinematic viscosity ν, and $\bar{f}$ is the (Maxwellian) equilibrium distribution, given by the Maxwell-Boltzmann distribution (see supplementary material for details).

Navier-Stokes vs. Boltzmann. From the distribution function f satisfying Eq. (3), one can in fact recover the macroscopic variables used in (1) and (2): macroscopic variables are simply integrals of microscopic values. For example, as shown in [Shan et al. 2006],

$$\rho = \int f\,\mathrm{d}\boldsymbol{v},\;\; \rho\boldsymbol{u} = \int \boldsymbol{v} f\,\mathrm{d}\boldsymbol{v},\;\text{and}\; p = \frac{1}{3}\int \|\boldsymbol{v}-\boldsymbol{u}\|_2^2 f\,\mathrm{d}\boldsymbol{v}. \tag{5}$$

That is, macroscopic quantities are simply "coarse-grained" from the microscopic probability distribution f. Consequently, it should come as no surprise that the Boltzmann model provides, in fact, a more detailed picture of fluid dynamics than the continuum NS model, so a proper numerical solver of Eq. (3) should be able to capture whatever fluid dynamics the continuum NS-based model can capture. Indeed, a multiscale perturbation analysis (known as the Chapman-Enskog expansion [Chen and Doolen 1998]) shows that the Navier-Stokes equation (2) is the first-order approximation of the continuous Boltzmann equation (3)[1].

Consequences on incompressibility. Not only is the continuous Boltzmann model more expressive, but its numerical solvers sidestep several challenges faced by traditional NS solver. The continuum model in Eqs. (1) and (2) considers the compressibility of fluids, which can impose severe limits on the timestep size of the numerical solver. To allow for much larger timestep sizes, most solvers further assume *incompressibility* of the fluids. This unphysical condition, however, gives rise to another challenge: the need for a pressure solve, involving a global equation. Since no fluid is absolutely incompressible, this pressure is merely a mathematical concept (a Lagrangian multiplier) that ensures incompressibility. As a result, while the no-penetration or no-slip pressure boundary conditions have been commonly used in graphics applications, it remains controversial what precise pressure boundary condition should be used

[1] There exist models that give higher-order approximations of the continuous Boltzmann equation. For example, the second-order approximation is known as the *Burnett* model. Physical measurements have shown that this latter model gives a better prediction than the Navier-Stokes model in certain cases that involve high-frequency wave phenomena.

to achieve high simulation accuracy [Gresho and Sani 1987]. In contrast, solving the continuous Boltzmann equation requires no assumption of incompressibility, thus, no pressure solve.

Consequences on non-linear advection. Another challenge stems from the fact that the advection term in Eq. (2) is nonlinear. To accurately capture the momentum advection, CFD methods rely on expensive nonlinear advection solves, while the advection step in CG is often approximated by the semi-Lagrangian scheme [Stam 1999] or any of its higher-order variants. Although computationally efficient, most semi-Lagrangian advection schemes suffer from numerical dissipation at large timestep sizes (the fluid motion quickly smears out), numerical dispersion at small timestep sizes (spurious wiggles appear), and are sometimes even unable to fully conserve mass, momentum, or energy. These limitations become prominent especially for high Reynolds number flows in which turbulence prevails — a scenario to which we pay particular attention in this work. On the contrary, the Boltzmann model (3) advects only f, not $\boldsymbol{u}$: this is just a *linear* advection, requiring a much simpler (and artifact-free) numerical recipe as briefly outlined next.

2.2 Lattice Boltzmann Methods

Generally known as lattice Boltzmann methods (LBM), numerical solvers of the continuous Boltzmann equation (3) first discretize the distribution function $f(\boldsymbol{x}, \boldsymbol{v}, t)$ in space, time, and velocity. The space and time discretization is similar to that in Navier-Stokes solvers: the spatial domain is filled with a grid and the simulation state is timestepped by Δt. For each position (i.e., grid node), we also choose a set of q discrete velocity directions $\{\boldsymbol{c}_i\}_{i=0}^{q-1}$. Therefore, the grid together with this set of vectors form a *lattice* structure. Two such lattice structures for 2D and 3D simulations are shown in Fig. 4, where $q = 3^D$ in dimension D, i.e., $q = 9$ in 2D and 27 in 3D.

Hermite basis. Key to the computational efficiency of LBM is its treatment of velocity discretization. Given a fixed $\boldsymbol{x}$ and t (e.g., for a grid node at time t), the idea is to view $f(\boldsymbol{x}, \boldsymbol{v}, t)$ as a function of the velocity $\boldsymbol{v}$ and to represent it using a *Hermite series* expansion:

$$f(\boldsymbol{x}, \boldsymbol{v}, t) \approx \frac{1}{(2\pi)^{D/2}} e^{-\frac{\|\boldsymbol{v}\|^2}{2}} \sum_{n=0}^{2D} \frac{1}{n!} \boldsymbol{a}^{(n)}(\boldsymbol{x}, t) : \boldsymbol{H}^{(n)}(\boldsymbol{v}), \qquad (6)$$

where D is the number of dimensions (e.g., $D = 3$ for 3D simulation), $\boldsymbol{H}^{(n)}(\boldsymbol{v})$ are the so-called Hermite polynomials of order n, and the operator ":" denotes full tensor contraction. Here the notation $\boldsymbol{H}$ is bold (and so are the coefficients $\boldsymbol{a}^{(n)}$) because there are multiple Hermite polynomials for each order n forming an orthogonal basis in the functional space of $\boldsymbol{v}$; see their specific forms in our supplementary material. The coefficients $\boldsymbol{a}^{(n)}(\boldsymbol{x}, t)$ are defined by projecting $f(\boldsymbol{x}, \boldsymbol{v}, t)$ onto the Hermite polynomials $\boldsymbol{H}^{(n)}(\boldsymbol{v})$:

$$\boldsymbol{a}^{(n)}(\boldsymbol{x}, t) = \int \frac{(2\pi)^{\frac{D}{2}} f(\boldsymbol{x}, \boldsymbol{v}, t)}{e^{-\|\boldsymbol{v}\|^2/2}} \boldsymbol{H}^{(n)}(\boldsymbol{v}) \mathrm{d}\boldsymbol{v} \approx \sum_{i=0}^{q-1} f_i(\boldsymbol{x}, t) \boldsymbol{H}^{(n)}(\boldsymbol{c}_i), \quad (7)$$

where the last approximation is realized through the *Gauss-Hermite* quadrature rule computed using the set of discrete velocity vectors $\boldsymbol{c}_i$, and $f_i(\boldsymbol{x}, t)$ denotes a properly scaled version of $f(\boldsymbol{x}, \boldsymbol{c}_i, t)$ (see supplemental material). Combining Eqs. (6) and (7) shows that $f(\boldsymbol{x}, \boldsymbol{v}, t)$ can be expressed as a linear combination of $\{f_i(\boldsymbol{x}, t)\}_{i=1}^{q}$: we just have to store these q values per node.

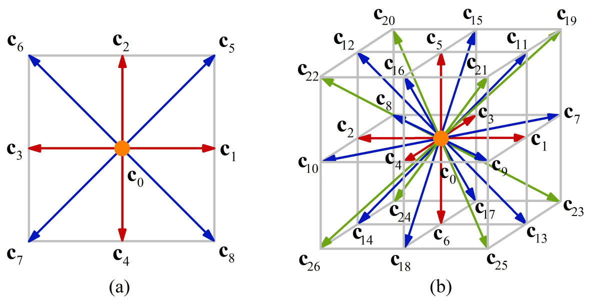

Fig. 4. **Lattice structures.** We use D2Q9 (a) in 2D and D3Q27 (b) in 3D, where $\mathbf{c}_i$ are discretized microscopic velocities. Each discretized distribution function f_i is associated with its corresponding velocity $\mathbf{c}_i$.

Discretization and update of distribution values. Next, we can apply spatial, temporal, and velocity discretizations to the continuous Boltzmann equation (3). (For now, we will avoid any mention of *scaling* between discrete and continuous variables for simplicity; this important topic will be discussed in detail in Sec. 4.) Assuming the BGK collision model from Eq. (4) as a particular simple, but illustrative example, we then obtain the *lattice Boltzmann* equations (LBE [Krüger et al. 2017]),

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t) - f_i(\boldsymbol{x}, t) = \Omega_i + F_i, \qquad (8)$$

where $\Omega_i = -\frac{1}{\tau}\left(f_i(\boldsymbol{x}, t) - \bar{f}_i(\boldsymbol{x}, t)\right)$ is the BGK operator along the direction $\boldsymbol{c}_i$, and F_i is the external force projected onto direction $\boldsymbol{c}_i$ (see the specific form of F_i in the supplementary material). The numerical recipe for timestepping the simulation simply follows the LBE (8): at each timestep, $\bar{f}_i$ and F_i are evaluated at each grid node independently, then the evaluation of $f_i(\boldsymbol{x}, t + \Delta t)$ involves only values on the nodes at $\boldsymbol{x} - \boldsymbol{c}_i \Delta t$ (recall the lattice structure of Fig. 4). Unlike the updates in Navier-Stokes solvers, no global solve is needed: the advection is linear, and data locality makes the update embarrassingly parallel. This gives LBM a *significant* advantage in terms of computational efficiency.

Moments. Another advantage of the lattice Boltzmann discretization is the direct connection of the values f_i on the grid to the macroscopic physical quantities. For example, the integrals in Eq. (5) can be evaluated also using Gauss-Hermite quadrature, namely,

$$\rho(\boldsymbol{x}, t) \equiv \sum_{i=0}^{q-1} f_i(\boldsymbol{x}, t) \quad \text{and} \quad \rho(\boldsymbol{x}, t)\boldsymbol{u}(\boldsymbol{x}, t) \equiv \sum_{i=0}^{q-1} \boldsymbol{c}_i f_i(\boldsymbol{x}, t). \qquad (9)$$

Note that ρ and $\rho\boldsymbol{u}$ in Eq. (5) are just the zeroth- and first-order *moments* of $f(\boldsymbol{x}, \boldsymbol{v}, t)$. These definitions can be extended to higher orders, where the n-th order moment is an n-th order tensor. For example, the second-order moment $\boldsymbol{\Pi}$ is a $D \times D$ matrix whose components (indexed by subscripts a, b) are

$$\Pi_{ab}(\boldsymbol{x}, t) = \int \boldsymbol{v}_a \boldsymbol{v}_b f \mathrm{d}\boldsymbol{v} \approx \sum_{i=0}^{q-1} \boldsymbol{c}_{i,a} \boldsymbol{c}_{i,b} f_i(\boldsymbol{x}, t). \qquad (10)$$

The second-order moment components also have physical interpretations: they represent the total momentum flux of the fluid at $(\boldsymbol{x}, t)$. Higher-order moments do not have physical interpretations, however; this does *not* mean, as we will see later, that they are irrelevant, far from it. Note that the components m_j of all the k-order moments (k up to 6 for the D3Q27 lattice discretization, and k up to 4 in 2D for D2Q9) are linear combination of terms of the form $\mathbf{c}_{i,a}^{\alpha}\mathbf{c}_{i,b}^{\beta}\mathbf{c}_{i,c}^{\gamma} f_i$ (for $a, b, c \in \{x,y,z\}$, and $\alpha, \beta, \gamma \in \{0,1,2\}$) as seen from Eqs. (9) and (10) for low orders; for instance, in 3D, m_0 is simply the zeroth order moment (equal to density), m_1, m_2, m_3 are the components of the vector representing the first-order moment, $m_4, ..., m_9$ are the components of the symmetric second-order moment $\mathbf{\Pi}$, etc. The vector of all moment components can thus be simply expressed through a linear transformation $\boldsymbol{m} = \mathbf{M}\boldsymbol{f}$, where $\boldsymbol{f}$ is a vector stacking all f_i for $i = 0..q-1$, $\boldsymbol{m}$ is a vector stacking the moment components m_j for $j = 0..q-1$ at the same location and time, and $\mathbf{M}$ is a fixed square and invertible $q\times q$ matrix whose simple expression can be found in [De Rosis and Luo 2019] or in our supplemental material.

Moment space relaxation. Historically, LBM has been considered less accurate than Navier-Stokes solvers. This is largely because the lattice BGK model (second-order Hermite expansion of Eq. (4), see supplementary material) used in early LBM simulations causes significant numerical dispersion. The recent introduction of the central-moment MRT model [Geier et al. 2006; De Rosis 2017; De Rosis and Luo 2019] significantly improves LBM's accuracy — one of our major contributions, explained in §3, is to further improve this model to offer low dissipation and low dispersion in order to allow for a stable and accurate fluid-solid two-way coupling in turbulent flows. The idea of moment space relaxation is simple: instead of relaxing f with a fixed rate $1/\tau$ as in the lattice BGK model derived from Eq. (4), we relax each moment with an *individual rate.* Using the notation defined above for the vector $\boldsymbol{f}$ of all values f_i and the vector $\boldsymbol{m}$ of all moment components at a grid node, the BGK operator in Eq. (8) amounts to write $\Omega = -\frac{1}{\tau}(\boldsymbol{f} - \bar{\boldsymbol{f}})$, where $\bar{\boldsymbol{f}}$ and Ω stack the local equilibrium distributions $\bar{f}_i$ and collision terms Ω_i respectively; moment space relaxation replaces this operator with

$$\Omega = -\mathbf{M}^{-1}\mathbf{R}\mathbf{M}(\boldsymbol{f} - \bar{\boldsymbol{f}}) = -\mathbf{M}^{-1}\mathbf{R}(\boldsymbol{m} - \overline{\boldsymbol{m}}), \tag{11}$$

instead, where $\overline{\boldsymbol{m}}$ stacks the moments of equilibrium distribution $\mathbf{M}\bar{\boldsymbol{f}}$, and $\mathbf{R}$ is a diagonal matrix whose diagonal elements $\{r_i\}_{i=1..q}$ specify the individual relaxation rates. The rationale behind using various relaxation rates is the observation that different moments approach their local equilibrium values with different rates: the zeroth-order moment is density (recall Eq. (9)), whose relaxation rate is zero since it is a conserved quantity; the first-order moments are momentum (which should in principle be conserved but is affected by body forces) whose relaxation rates are set to 1 [De Rosis et al. 2019]; the second-order moments, which represent the momentum flux, relax towards their equilibrium states depending on the *kinematic viscosity* ν of the fluid. Therefore, the relaxation rates of low-order moments (at least up to the second order) can be set from their physical interpretations, which was proven to increase the accuracy of LBM significantly [De Rosis and Luo 2019].

2.3 Incorporating immersed boundary

When a solid is fully or partially immersed in a fluid, boundary conditions $\mathbf{u}_b$ on the fluid velocity, such as no-slip ($\mathbf{u}_b = \mathbf{u}_s$) or slipping ($\mathbf{u}_b \cdot \mathbf{n} = \mathbf{u}_s \cdot \mathbf{n}$) conditions where $\mathbf{u}_s$ is the solid velocity and $\mathbf{n}$ is the boundary surface normal, need to be imposed. The immersed boundary (IB) method [Peskin 1972] enforces these conditions through penalty forces, and its diffuse-interface variant can easily be incorporated in an LBM solver to locally and efficiently deal with complex geometries in fluids [Li et al. 2016].

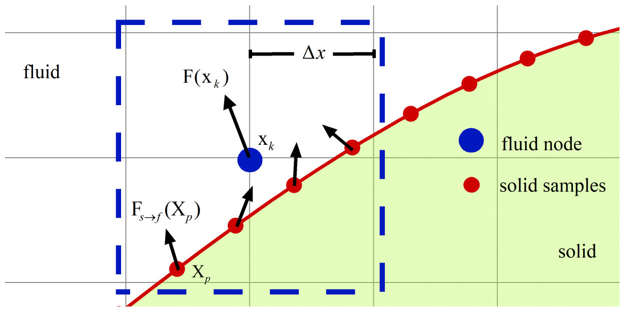

Fig. 5. **Immersed boundary method.** To enforce proper velocity conditions around solid boundaries, penalty forces $\mathbf{F}_{s\rightarrow f}(\mathbf{X}_p)$ are first computed at samples $\mathbf{X}_p$ on the solid. By spreading $\mathbf{F}_{s\rightarrow f}(\mathbf{X}_p)$ to fluid node $\mathbf{x}_k$ with a 2×2 kernel, the force $\mathbf{F}(\mathbf{x}_k)$ can be computed to affect the fluid motion around the solid boundary. Conversely, reactive forces $\mathbf{F}_{f\rightarrow s}(\mathbf{X}_p)$ then affects the motion of the solid.

Given a spatial location $\mathbf{X}_p$ located on the solid boundary (see red points in Fig. 5), we can first calculate the fluid velocity $\mathbf{u}(\mathbf{X}_p)$ by disregarding the presence of the solid and simply interpolating the local macroscopic fluid velocities $\mathbf{u}(\mathbf{x}_k)$ via a distance-based kernel at nearby grid nodes $\mathbf{x}_k$. As the resulting velocity is unlikely to match the desired boundary velocity $\mathbf{u}_b(\mathbf{X}_p)$ imposed by the solid, penalty forces per unit volume on both fluid and solid are used to enforce proper boundary condition, expressed as:

$$\mathbf{F}_{s\rightarrow f}(\mathbf{X}_p) = \rho\frac{\mathbf{u}_b(\mathbf{X}_p) - \mathbf{u}(\mathbf{X}_p)}{\Delta t}, \quad \mathbf{F}_{f\rightarrow s}(\mathbf{X}_p) = -\mathbf{F}_{s\rightarrow f}(\mathbf{X}_p), \tag{12}$$

where $\mathbf{F}_{s\rightarrow f}(\mathbf{X}_p)$ and $\mathbf{F}_{f\rightarrow s}(\mathbf{X}_p)$ denote forces from solid to fluid and from fluid to solid, respectively. The forces $\mathbf{F}_{s\rightarrow f}(\mathbf{X}_p)$ at various locations $\{\mathbf{X}_p\}_p$ sampling the solid boundary can be spread (using the same distance-based kernel as for the velocity interpolation) to the local fluid nodes $\mathbf{x}_k$ to obtain a resulting force $\mathbf{F}(\mathbf{x}_k)$ at each nearby fluid node, see Fig. 5. In order to capture boundary vortices as finely as possible, a fairly local kernel is desirable not to smooth the boundary flow too much (for the specific form of the kernel used in practice, see our supplementary material). For the solid, forces and their resulting torques are summed up (i.e., integrated) to give the total force $\mathbf{F}^s$ and total torque $\boldsymbol{\tau}^s$ which then drive the motion of the solid; they are expressed as:

$$\mathbf{F}^s = \sum\nolimits_p \mathbf{F}_{f\rightarrow s}(\mathbf{X}_p)\Delta s_p, \quad \boldsymbol{\tau}^s = \sum\nolimits_p (\mathbf{X}_p - \mathbf{B}) \times \mathbf{F}_{f\rightarrow s}(\mathbf{X}_p)\Delta s_p, \tag{13}$$

where $\mathbf{B}$ is the barycenter of the solid, and Δs_p is the associated surface area that sample $\mathbf{X}_p$ covers.

2.4 Discussion and overview

In recent LBM works based on the moment space relaxation model, the first $D(D+3)/2$ relaxation rates r_i (corresponding to components of the zeroth, first, and second order moments) are set based on the physical properties of the fluid; however, the other higher-order rates are usually set to a constant in $[0, 2]$, the lack of physical interpretations of their associated higher-order moments providing no clear way to fix them more meaningfully. However, various authors acknowledge that these high-order rates have *some* influence on the dissipation of the high-frequencies of the probability distributions in practice. Yet no concrete values offering a form of optimal choice have been proposed — even if a few authors suggested fixed, hand-tuned values that seem to offer good results in practice.

Despite having only a few meaningful rates set, this recent relaxation model has been shown far superior in accuracy to previous models. Still, the inability to optimally set most of the rates remains a frustrating limitation. In the next section, we show how these rates can be found on-the-fly based on the local distribution values to offer a low-dissipation and low-dispersion fluid solver.

3 LOW DISSIPATION AND LOW DISPERSION SOLVER

In this section, we introduce a fluid solver with low dissipation and low dispersion by formulating a high-order adaptive central-moment-based relaxation model, where "high-order" refers to high-order expansion of probability distributions by Hermite polynomials, not to high-order discretizations of spatial or temporal derivatives. More specifically, we show how locally adapting the relaxation rates according to the current fluid state makes the LBM solver outperform all recent NS schemes on common numerical tests.

3.1 Central-moment space relaxation

The intuition behind the moment-space relaxation model is to project the velocity distributions into a "macroscopic" moment space and use different rates to control their respective relaxation. However, the moment space defined by Eq. (11) may violate Galilean invariance, which is particularly problematic for high Reynolds number flows. To better respect Galilean invariance, Geier et al. [2006] proposed the central-moment relaxation model, where the new projection matrix onto the "central" moment space is now function of the local macroscopic velocity $\mathbf{u}$: the matrix $\mathbf{M}(\mathbf{u})$ is still square and invertible, but compared to the original form given in §2.2, it now contains terms of the form $(\boldsymbol{c}_{i,a} - \mathbf{u}_a)^{\alpha}(\boldsymbol{c}_{i,b} - \mathbf{u}_b)^{\beta}(\boldsymbol{c}_{i,c} - \mathbf{u}_c)^{\gamma} f_i$. As classically used in statistics, central moments are preferable to ordinary moments, computed in terms of deviations from their mean instead of from zero, because the high-order central moments depend *only* on the shape of the distribution, and not to its location, rendering the moments more meaningful. Note that this implies that the matrix $\mathbf{M}(\mathbf{u})$ must now be computed on-the-fly, as well as its inverse; however, since analytical forms of both matrices as a function of $\mathbf{u}$ can be computed analytically, this computational overhead (which increases the cost of evaluating the collision terms by less than 5% after optimization compared to the simpler BGK model) turns out to be well worth it, as we will see in detail shortly.

Consequences of central moments. The central-moment model of relaxation uses *all* six orders of moments, leading to important numerical consequences, the most obvious one being that the equilibrium central moments are Galilean invariant by construction: if we project the Hermite sixth-order polynomial expansion of the equilibrium distribution $\bar{f}$ into central-moment space via $\overline{\boldsymbol{m}} = \mathbf{M}(\mathbf{u})\bar{f}$, most components of $\overline{\boldsymbol{m}}$ become zero, except for

$$\overline{m}_0 = \overline{m}_9 = \rho, \quad \overline{m}_{17} = c_s^2\rho, \quad \overline{m}_{18} = c_s^4\rho, \quad \overline{m}_{26} = c_s^6\rho. \tag{14}$$

(Using fewer moments would not lead to such a simplification, justifying a posteriori the use of all 27 moment coefficients in 3D, covering all the way to the sixth-order moment.) Note that there is no macroscopic velocity $\mathbf{u}$ in this new equilibrium distribution expression in moment space, which therefore respects Galilean invariance exactly, dramatically improving the accuracy of LBM for high Reynolds number flows. This property is not shared by current LBM methods used in CG such as [Li et al. 2019]. Note that the same "cancellation" effect happens for external forces as well when all six orders are used [De Rosis et al. 2019], thus guaranteeing a more accurate treatment of external forces and coupling.

3.2 Empirical relaxation rates

When central moments are used, most authors end up using high-order relaxation rates $\{r_i\}_{i>i^*}$ (where $i^* = D(D+3)/2$ corresponds to the index of the final component of the second moment) equal to 1 by lack of physical guidance to select better values. However, recent work by Li et al. [2019] tried to dig further into setting these rates properly, albeit in an empirical fashion. First, they use an inverse re-parameterization to express the relaxation rates r_i as:

$$r_i = 1/(\tau_i + 1/2), \tag{15}$$

where $\tau_i \geq 0$ are rescaled high-order relaxation times. Second, they demonstrate that small values of τ_i introduce dissipation that helps *filter out dispersion errors*. That is, while $\tau_{i>i^*} = 0$ may seem best in terms of minimizing dissipative errors of the solver, small, flow-adapted high-order relaxation times can help combat numerical dispersion as well and thus preserve turbulence better. In other words, the higher-order relaxation rates $r_{i>i^*}$ (or, equivalently, the relaxation times $\tau_{i>i^*}$) help find a trade-off between low-dispersion and low-dissipation for the resulting scheme. They then proposed a series of fixed, hand-tuned values for τ_i that they scaled based on the local gradient of velocity $\nabla\mathbf{u}$ to add dissipation on the higher-order moments roughly proportional to the local amount of turbulence. Adapting the high-order relaxation rates according to the local flow characteristics was shown to result in turbulence details and flow instabilities being more faithfully captured. While their adaptive solution is only empirical, we leverage the main ideas behind it in order to construct a more systematic approach to compute proper high-order relaxation rates locally and on-the-fly, as explained next.

3.3 Optimal relaxation rates

We now formally introduce our way to compute higher-order relaxation rates to create a low-dissipation and low-dispersion solver.

Measurement functional. In order to evaluate the effects of high-order relaxation rates (or equivalently, the high-order relaxation times), we first form a local measurement functional $\epsilon(\mathbf{x}_k, t)$ by simply summing the normalized change in each of the first three (physically-meaningful) moments at grid location $\mathbf{x}_k$ and time t after one step of LBM integration of Eq. (8); that is,

$$\epsilon(\mathbf{x}_k, t) = \frac{\|\delta^t(\rho)_k\|}{\overline{\rho}} + \frac{\|\delta^t(\rho\mathbf{u})_k\|}{\overline{\|\rho\mathbf{u}\|}} + \frac{\|\delta^t(\boldsymbol{\Pi})_k\|}{\overline{\|\boldsymbol{\Pi}\|}}, \tag{16}$$

where the temporal difference operator is: $\delta^t(\cdot) = (\cdot)^{t+\Delta t} - (\cdot)^t$, and the overline indicates averaging over the whole simulation process, used for normalization purposes. This functional is such that when dissipation or dispersion errors increase, the differences of the three moments over consecutive time steps also increase. Note that this is not, per se, an *error* measurement in the sense that changes in density, momentum, or momentum flux at a given location happen based on the evolution of a fluid, so one should not expect this functional to be zero even in case of perfect accuracy; however, *tracking how this functional changes as we change the high-order relaxation times τ_i* is very instructive. As it happens, plotting this functional ϵ at various grid node $\mathbf{x}_k$ and different

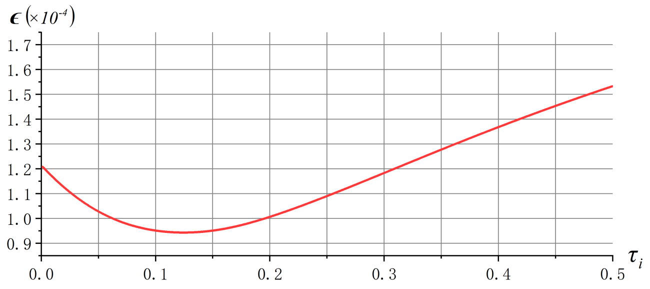

Fig. 6. **Typical measurement behavior.** The measurement functional as a function of the high-order relaxation rate τ_i typically exhibits a clear minimum, indicating a minimal amount of dissipation and dispersion.

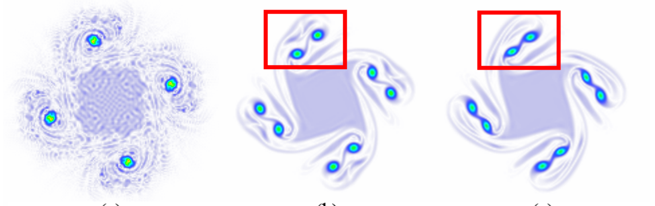

Fig. 7. **Effects of high-order relaxation time.** Using the vortex-sheet simulation as an example, if τ_i is set too small, strong dispersive effects appear (a); for τ_i minimizing our measurement functional, we obtain a low-dissipative and low-dispersive behavior (b); if a slightly higher rate is used instead, dissipation occurs as evidenced by the vortex pair merging (c).

times t for arbitrary simulations shows almost always the same *qualitative* behavior: when a high-order relaxation time τ_i is varied, the measurement functional behaves as indicated in Fig. 6, i.e., it forms a U-shape curve with a unique global minimum (this example was obtained from a grid node of the jetstream simulation in Fig. 10 for a particular $\tau_{i=10}$). This quasi-convex behavior does not hold on a few grid nodes, though: if a region is very uniform, then the functional becomes flat, as high-order relaxation time has basically no effect locally; inversely, if the grid node is on a sharp velocity change, the functional shows a constant decay as τ_i is increased, since increasing high-order relaxation time will proportionally blur the jump. Aside from these two special, rare cases, the U-shape confirms the empirical findings of [Li et al. 2019]: increasing a small amount of high-order relaxation times from zero reduces ϵ as it acts against numerical dispersion errors; however, further increasing the rate τ_i grows ϵ again due to both excessive dissipation and, at very large high-order relaxation times, excessive dispersion as well. Fig. 7 visualizes this phenomenon, where too low a time τ_i exhibits dispersion (a), while a larger time introduces dissipation errors (c) as evidenced by the paired vortices in the red box being partially merged. Somewhere in between, the ϵ curve reaches a bottom around which one can expect the best tradeoff between dispersion and dissipation errors, see Fig. 7(b). Moreover, the various curves for different high-order i vary quite a bit in their scales (with higher orders being overall flatter, especially in 3D), but the different values of τ_i at the curve's minimum remain always within a small range, see Fig. 8(a) for a 2D illustration. This observation indicates that there is very little to gain in searching for individual optimal $\tau_i^* = \operatorname{argmin}_{\tau_i} \epsilon(\mathbf{x}_k, t)$ for *each* of the high-order moments: we can simply assume a single high-order relaxation time τ^* such that $\tau_i^* = \tau^*$ for all high-order indices $i > i^*$, as it will not affect the simulation accuracy significantly; see Fig. 8 (b) which reinforces the fact that the choice of multiple high-order relaxation times v.s. a single one hardly matters.

Numerical optimization of τ^*. Based on these observations, one may try to adapt the high-order relaxation time τ^* optimally *at every grid node and at every time step* in order to reduce both dispersion and dissipation throughout the simulation. This idea can be implemented as follows: after using an initial τ^* (we found that $\tau^* = 0.005$ is a safe bet to reduce dispersion while not adding undue dissipation) for the very first time step of LBM integration, we can then numerically determine a good relaxation time τ^* at each grid node $\mathbf{x}_k$ and at each time step t_n by simply evaluating $\epsilon(\mathbf{x}_k, t_n)$ numerically from Eq. (16) to sample the curve through a fixed number of high-order relaxation times: if a clear local minimum is found, we use that time as our new τ^* at this node; if no local minimum is found (in the rare cases discussed above), we simply keep the previous local time used at the previous time step. While this approach favors accuracy in numerically computing the optimal times everywhere, it is far from practical and efficient as it requires too large an amount of functional evaluations and tentative time integrations just to evaluate the effect of the high-order relaxation time.

Regression-based evaluation of local τ^*. Instead, we still wish to adapt the high-order relaxation time at each node and at each time step, but using a much more efficient evaluation which, hopefully, only slightly degrades the resulting integrator. Ideally, one could predict an optimal τ^* at a node purely based on the local fluid state, which suggests the use of linear regression. As a simple local description of the fluid state, we select the *norms* of the first three moments (ρ, $\|\rho\mathbf{u}\|$, $\|\mathbf{\Pi}\|$) since the optimal high-order relaxation time should remain invariant to local rotations of the fluid state, so only local rotation-invariant measures should be used as explanatory variables of the regression. Then, in an offline precomputation, we collect a series of local states $\mathbf{s}_p = (\rho_p/\overline{\|\rho\|}, \|\rho_p\mathbf{u}_p\|/\overline{\|\rho\boldsymbol{u}\|}, \|\mathbf{\Pi}_p\|/\overline{\|\mathbf{\Pi}\|}, 1)$ (we add 1 as an extra variable to allow an affine relationship) and the corresponding optimal time τ_p^* computed using the brute-force numerical optimization described in the previous paragraph. In practice, we run 3 different simulations at different Reynold numbers and gather all grid nodes and their optimal relaxation times for

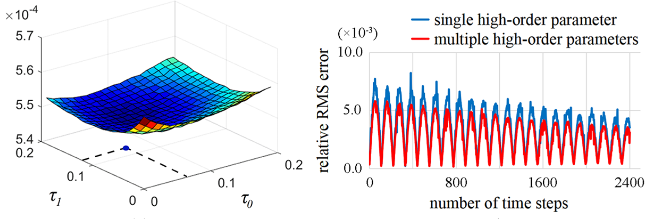

Fig. 8. **Single vs. multiple high-order relaxation times.** If half the high-order relaxation times are assigned a value τ_0 and the other half are assigned a different value τ_1, the measurement functional (a) indicates that the optimal times are in fact quite similar, with $\tau_0^* = 0.07$ and $\tau_1^* = 0.11$. Using these multiple times do indeed reduce the error measured here from 2D Taylor-Green vortex simulation (b), but only marginally — implying that a single high-order relaxation time is sufficient for practical purposes.

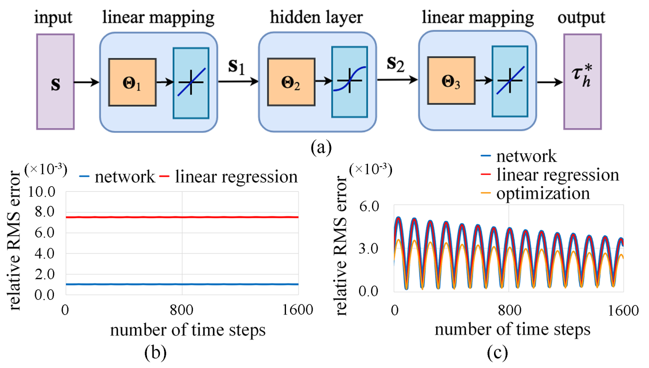

Fig. 9. **Comparison with non-linear regression.** Using a neural network structure (a) where each layer corresponds to a fully-connected network with a parameter matrix Θ_j reduces the relative root mean squared error of τ^* measured w.r.t. the fully optimized value (b), but the difference with linear regression is too small to affect the resulting velocity fields as indicated by the measured errors on the 2D Taylor-Green vortex simulation (c).

all time steps (for a total of around 10^8 samples) in order to make sure we cover a large variety of local behaviors. We then find the approximation of τ^* as $\hat{\tau}^*_p \approx \boldsymbol{\theta}^T \mathbf{s}_p$, and find the optimal four linear coefficients $\boldsymbol{\theta}^*$ by solving the usual least-squares problem:

$$\boldsymbol{\theta}^* = \operatorname{argmin}_{\boldsymbol{\theta}} \sum\nolimits_p \left(\hat{\tau}^*_p(\boldsymbol{\theta}) - \tau^*_p\right)^2. \tag{17}$$

Surprisingly, the solution we found, i.e., $\boldsymbol{\theta}^*$=(0.0003, -0.00775, 0.00016, 0.0087) leads to a very small residual, indicating that this simple linear regression essentially captures a nearly-optimal relaxation time. We verify this statement next by comparing this regression-based option with another more sophisticated learning approach.

Comparison with nonlinear regression. In order to test the validity of our linear regression, we also tried a nonlinear regression to obtain higher accuracy in the prediction. We implemented a neural-network-based nonlinear regression, where a fully-connected network with several layers and a sigmoid symmetric activation function is employed as shown in Fig. 9. The loss function is defined as in Eq. (17), with the difference that $\hat{\tau}^*_p$ is now represented by the network, where each layer contains a parameter matrix Θ_j (j indicating the index of the network layer) which is to be learned. We train the neural network using the neural net fitting toolbox in Matlab [Mathworks, Inc. 2017], and found that the prediction accuracy for τ^* is only slightly improved compared to linear regression, see Fig. 9(b). More importantly, the effects of this better prediction are barely noticeable on actual numerical tests of the

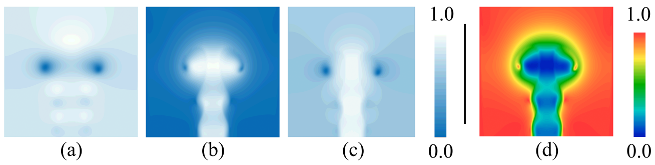

Fig. 10. **Spatial distribution of** τ^*: From the (normalized) visualizations of scalar fields (a) ρ, (b) $\|\rho\boldsymbol{u}\|$ and (c) $\|\boldsymbol{\Pi}\|$, the predicted τ^* field looks, to first order, like the inverse of $\|\rho\boldsymbol{u}\|$. This is confirmed by the optimal vector $\boldsymbol{\theta}^*$ of the linear regression: the coefficient for $\|\rho\boldsymbol{u}\|$ dominates all the other coefficients, explaining this approximate relationship.

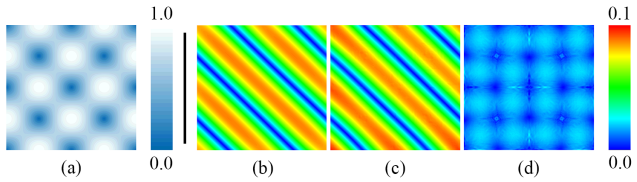

Fig. 11. **2D Taylor-Green vortex simulation.** The (normalized) converged reference velocity magnitude (a), with error distributions for MC+R solver (b), BiMocq^2 solver (c), and our kinetic solver with linear regression (d). Note that (b) to (d) are normalized to the same scale for visualization. The right colorbar shows the relative mean-squared error range.

resulting LBM integrator (see the ℓ_2 error difference of the velocity fields in Fig. 9(c)). Therefore, we adopt the linear regression to evaluate the best relaxation time τ^* due to higher efficiency in the remainder of this paper.

3.4 Evaluating the resulting LBM solver

Equipped with our regression-based high-order relaxation times which we efficiently compute at every grid node and for every time step, we can integrate in time the lattice Boltzmann equations. Fig. 10 illustrates that the optimal relaxation rates are rather intuitively distributed within the simulation: small momentum (and thus, smooth) regions of the flow are assigned larger relaxation times to dampen dispersion, whereas large momentum (i.e., often non-smooth) regions are typically assigned smaller relaxation times as accuracy is more important there. We now evaluate quantitatively and qualitatively the visual pertinence and accuracy of our resulting integrator compared to current state-of-the-art Navier-Stokes solvers as well as a recent LBM solver on two usual 2D simulation tests.

2D Taylor-Green vortex simulation. The 2D Taylor–Green vortex example, representing an unsteady flow of vortices (we use zero viscosity), is often used for testing and validation of the temporal accuracy of Navier–Stokes algorithms since an analytical solution is known, providing a ground-truth velocity field to evaluate numerical errors. In Fig. 11, we show the expected converged configuration through its vorticity plot (left), and display the mean-squared error distributions for three different methods: the reflection-advection MacCormack solver (MC+R) [Zehnder et al. 2018], the BiMocq^2 solver [Qu et al. 2019], and ours. Compared to these most recent NS solvers, our LBM approach produces significantly smaller error magnitudes (the root mean squared error is over 35% better, see Tab. 1), without anisotropic artifacts. Moreover, using either a network-based regression or a full numerical evaluation of the high-order relaxation time used in our LBM solver does not visually change the error plot significantly: Fig. 9(c) shows the time-varying root mean squared error over the entire domain of the predicted high-order relaxation time τ^* using both linear and network-based nonlinear regression compared to the slow, but more accurate full numerical optimization. Clearly, the slowest evaluation results in the

Table 1. **Taylor-Green statistics.** Relative mean-squared errors for different solvers; note that "optimization", "network" and "linear regression" refer to various predictions of high-order relaxation times for our kinetic solver.

MC+R	BiMocq^2	optimization	network	linear regression
0.0086962	0.0088046	0.0025124	0.0030662	0.0030658

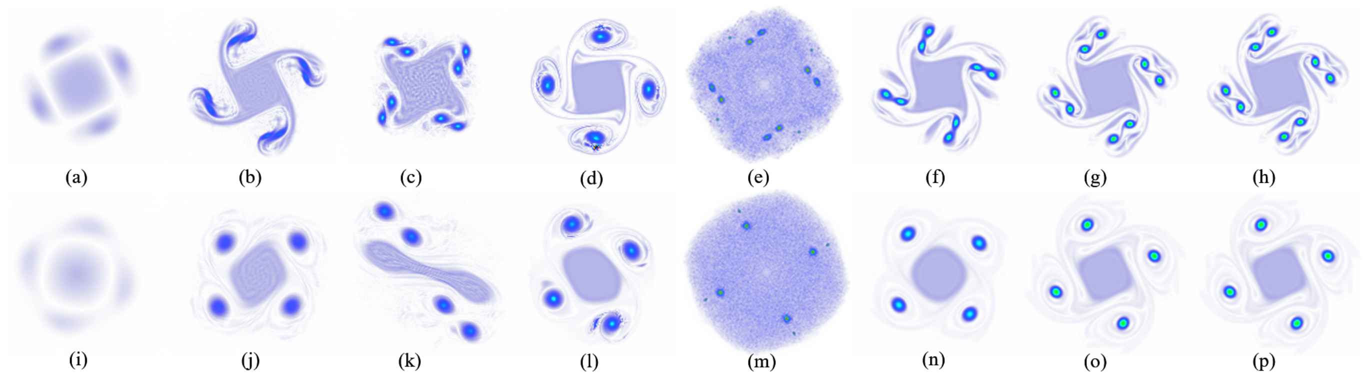

Fig. 12. **2D vortex sheet simulation.** Top row: simulation snapshots at an early stage ($t = 2.3$s); bottom row: simulation snapshots at $t = 7.5$s. (a) & (i) stable fluids solver (256×256); (b) & (j) unconditionally stable MacCormack solver (256×256); (c) & (k) reflection-advection MacCormack (MC+R) solver (256×256); (d) & (l) BiMocq^2 solver (256×256); (e) & (m) nonlinear APA* solver (1024×1024), which is taken as a reference; (f) & (n) kinetic solver from [Li et al. 2019] (256×256); (g) & (o) our new kinetic solver with direct optimization (256×256); (h) & (p) with linear regression instead.

lowest error, followed by the network-based regression approach, and finally our fast linear regression version; but the relative errors are all within 0.8% of each other. Moreover, it does not affect the simulation accuracy, confirming that the linear regression offers an excellent compromise between efficiency and accuracy.

2D vortex sheet simulation. We also perform a comparison of our results against various NS solvers on the 2D vortex sheet simulation. Fig. 12 shows the results on a 256×256 grid of the stable fluid solver of [Stam 1999], the MacCormack solver (MC) of [Selle et al. 2008], the reflection-advection solver (MC+R) [Zehnder et al. 2018], the recent BiMocq^2 solver [Qu et al. 2019], the LBM solver from [Li et al. 2019], our new LBM solver with linear regression and with full numerical optimization, as well as the structure-preserving and conservative solver APA* of [Mullen et al. 2009] computed at high resolution (1024×1024) as a reference. Our new kinetic solver (Fig. 12 (g)&(o) and (h)&(p)) produces similar vortex distributions to the reference (Fig. 12 (e)&(m)), without the high-frequency artifacts due to the strong dispersive nature of the reference conservative solver. In comparison, many existing solvers do not maintain the rotating vortices in time as they shear away (see the bottom row of Fig. 12 (i) to (l)). Note that using smaller time steps will also introduce a large amount of dispersive error for Navier-Stokes solvers, see Fig. 13; in contrast, our kinetic solver is visually noise-free due to low dispersion errors for arbitrary time step size. Moreover, Fig. 14 shows that our solver also exhibits excellent conservation of kinetic energy as compared to many other solvers, proving that our high-order relaxation times also promote low dissipation. Note that our kinetic solver with linear regression produces almost the same results as direct numerical optimization, confirming our previous findings.

3.5 Handling high Reynolds number flows

While our optimized high-order relaxation rates provide a compromise between dissipation and dispersion errors that helps resolve turbulence details more accurately, too coarse a grid resolution prevents the proper handling of *very* turbulent flows: vortices of length scales equal to or below grid spacing cannot be numerically well accounted for. Sub-grid modeling is typically used to improve this situation by trying to approximate the effects of unresolved small-scale fluid motions (small eddies, swirls, vortices). For instance, one can employ the wall-adaptive large-eddy (WALE [Weickert et al. 2010]) model to better resolve sub-grid turbulence: it predicts an eddy viscosity ν_{sub} at each grid node to approximate the unresolved effects. This model is trivially incorporated in our LBM solver by adding this eddy viscosity to the kinematic viscosity ν discussed in Sec. 2.2 to alter low-order relaxation rates in order to add sub-grid flow contributions to the existing grid scale. Note that this modification of our solver has zero influence on the numerical tests we presented earlier: these tests were for flows with very low Reynolds numbers, so the eddy viscosity ν_{sub} was negligible. However, it adds significant stability when flows are very turbulent, at almost negligible computational cost.

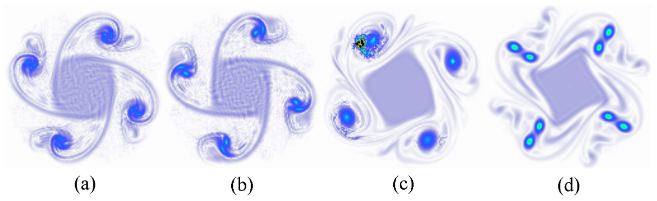

Fig. 13. **Dispersion errors.** Existing solvers with high-order advection exhibit different levels of dispersion at small time steps, while our solver produces low-dissipation and low-dispersion results: for a time step $\Delta t = 0.001$ (with $\Delta x = 1$), (a) MacCormack solver; (b) reflection-advection MacCormack solver; (c) BiMocq^2 solver; vs. (d) our new LBM solver.

4 ADAPTIVE SIMULATION FOR STABILITY

Now that we have described the construction of a flow simulator exhibiting very low dissipation and dispersion as well as capturing fine details and turbulence, directly incorporating the immersed boundary (IB) method as described in Sec. 2.3 is straightforward. It already improves upon existing IB-LBM techniques since, due to reduced dispersivity, our solver can better handle the larger velocity variations typically present in turbulence, and supports a smaller IB kernel size to limit the amount of velocity smoothing around solid boundaries. However, when solid velocity becomes too large (for instance, due to local spinning or gravity), large penalty forces

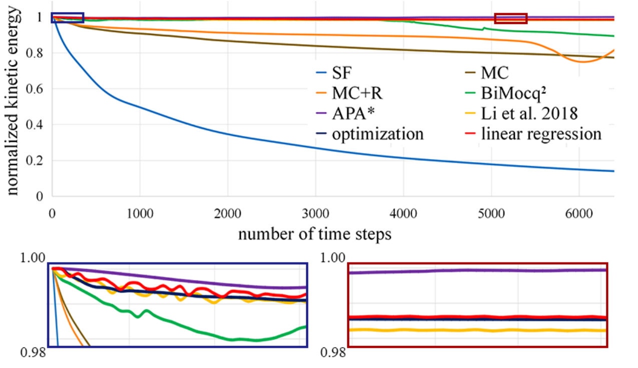

Fig. 14. **Energy conservation.** Variations of the total kinetic energy over time steps (top) show that many existing Navier-Stokes solvers in graphics are not quite conservative; however. our kinetic solver has a very similar energy behavior to the APA* solver, a good feature for turbulent flows.

imposed to the fluid can exceed the stability range of the LBM simulator, making the simulation fail. To address such cases and guarantee a more accurate treatment, one needs to look carefully at the spatial and temporal scales that typical LBM simulations proceed with. In fact, we show that one can adapt temporal (and optionally, spatial) discretizations of the LBM solver during simulation in order to safely and efficiently proceed based on the current solid velocity and macroscopic fluid velocity.

4.1 Dimension scaling in LBM

In Sec. 2.2 when we reviewed the basic LBM discretization and time integration, we purposely simplified the explanations by assuming that the discretization of physical fields did not introduce a scale change. Yet, the time integration of the lattice Boltzmann equation (8) actually assumes *unit* time step and grid spacing, which implies that all physical quantities are implicitly rescaled accordingly. This dimension scaling in LBM is rarely discussed at length, and we only found two unpublished technical reports [Junk 2006; Latt 2008] mentioning this point carefully, but none of the IB-LBM approaches even note this important fact. Therefore, we review next how the LBM quantities are related to their physical counterparts. To clearly distinguish physical units from LBM units, we will use two different fonts: while physical quantities will remain written in computer modern, outlined letters will be used to refer to LBM variables. So time t, for instance, becomes $\mathbb{t}$ when discretized at the lattice Boltzmann scale.

Physical vs. LBM units. Assume for now that one uses a regular spatial grid of spacing Δx (expressed in real physical units, meters in this case), and that a time step size Δt (in seconds) has been fixed. Lattice Boltzmann discretization first rescales all physical units to become dimensionless, then converts them into lattice units based on the grid size (see supplemental document for details). As a result, we have the following relations between physical and lattice units for the various physical (real) and lattice (macroscopic) variables:

$$t = \frac{\mathbb{u}_{\text{ref}}}{u_{\text{ref}}} \Delta x\, \mathbb{t}, \quad \Delta t = \frac{\mathbb{u}_{\text{ref}}}{u_{\text{ref}}} \Delta x\, \Delta\mathbb{t}, \quad \mathbf{x} = \Delta x\, \mathbb{x}, \quad \Delta\mathbb{x} = \Delta\mathbb{t} = 1,$$
$$\rho = \rho_0\, \varrho, \quad \mathbf{u} = \frac{u_{\text{ref}}}{\mathbb{u}_{\text{ref}}} \mathbb{u}, \quad \mathbf{F} = \frac{\rho_0}{\Delta x}\left(\frac{u_{\text{ref}}}{\mathbb{u}_{\text{ref}}}\right)^2 \mathbb{F}, \quad \nu = \frac{u_{\text{ref}}\Delta x}{\mathbb{u}_{\text{ref}}} \mathbb{v}, \tag{18}$$

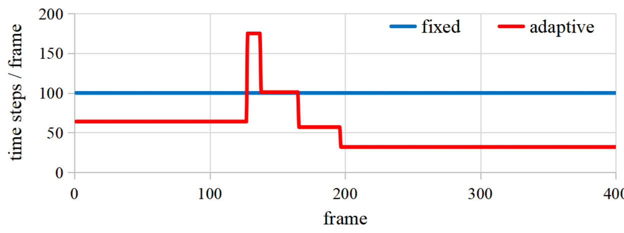

Fig. 15. **Effective time steps.** For the example in Fig. 18, an adaptive simulation (red cure) requires far fewer time steps than if a fixed u_{ref} is chosen (blue curve). The smallest effective time step happens when the box hits the ground, creating a sudden velocity change, which in turn reduces the time step to ensure stability and increased accuracy.

where ρ_0 is the fluid density, u_{ref} is an arbitrary reference magnitude for the fluid velocity $\mathbf{u}$ during the simulation, while $\mathbb{u}_{\text{ref}}$ is the corresponding velocity magnitude in LBM scale; the ratio $u_{\text{ref}}/\mathbb{u}_{\text{ref}}$ thus determines the speed scale between real and LBM velocities.

Physical time step sizes. Now, because LBM assumes unit time steps and grid sizes, the reference LBM velocity $\mathbb{u}_{\text{ref}}$ (which corresponds to the reference velocity u_{ref} in physical scale) is typically set in the range $[0.1, 0.2]$ to ensure stability of the time integration of Eq. (8) — i.e., to satisfy the CFL condition of stability at the LBM scale. This means that if one wants to enforce a stable LBM integration, the only parameter left to adjust scaling between LBM and physical units is u_{ref}. Picking $u_{\text{ref}} = u_{\max}$, i.e., the maximum physical velocity magnitude of a given simulation, amounts to setting $\mathbb{u}_{\text{ref}} = \mathbb{u}_{\max} \equiv 0.2$ in LBM scale; this choice means that the effective time step Δt that LBM performs when using $\Delta\mathbb{t}=1$ is $\Delta t = 0.2\, \Delta x/u_{\max}$, which corresponds to the Courant condition for typical fluid solvers working in physical space. However, picking $u_{\text{ref}} > u_{\max}$ is *also* feasible: we are then taking more conservative physical time steps with $\Delta t < 0.2/u_{\max}\Delta x$, thus improving the accuracy of the simulation — but spending more computing time to simulate a fixed physical duration of simulation as a consequence. In both cases, the LBM simulation is perfectly stable as we keep $\mathbb{u}_{\text{ref}}$ in a safe range.

Immersed boundary force exchange. We emphasize also that the scaling between physical forces $\mathbf{F}$ and LBM-derived forces given in Eq. (18) is non-trivial, and must be used in order to produce the right coupling between fluid and solid using the penalty approach described in Sec. 2.3. Surprisingly, we have not found a reference mentioning this factor in the existing IB-LBM literature. When a solid is present inside the fluid, its density ρ_s is used to convert forces to acceleration so that the solid motion can be integrated.

4.2 Adaptive simulation

Based on the differences between physical and LBM scales, a simple strategy emerges to ensure that our fluid solver can keep on running even when large velocities crop up in the course of fluid-solid interactions: we always keep $\mathbb{u}_{\text{ref}}=0.2$, but adapt u_{ref} to make sure we can safely handle the current maximum fluid velocity $u_{\max}$ while taking large time steps to efficiently advance the simulation in time.

Adaptive scaling. More precisely, since we always simulate in LBM scale, we keep the maximum velocity $\mathbb{u}_{\max} \in [0.18, 0.22]$ to ensure stability. During the simulation, we check $\mathbb{u}_{\max}$ at each time

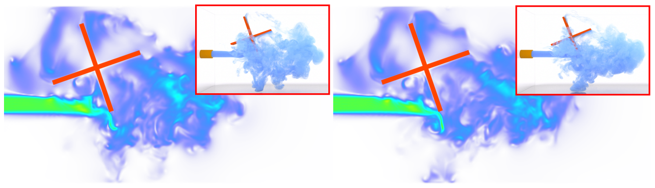

Fig. 16. **Fixed vs. adaptive time steps.** Fixed (a) vs. adaptive (b) time steps (where u_{ref} is changed adaptively) both lead to a velocity field which is quite similar even after a few seconds of simulation of a two-way coupling simulation (we show a cross section in pseudocolors), except for small-scale turbulence details. Insets show visualization of the smoke distribution and the interacting solid at corresponding times during the simulation.

step; for unstable flows, $\mathscr{u}_{\max}$ may either exceed 0.22 (thus potentially creating instability) or fall below 0.18, (thus unnecessarily decreasing the simulation efficiency). In either case, we determine a u_{ref} such that $\mathscr{u}_{\max}$ can be readjusted to 0.2. This leads to an update $u_{\text{ref}} \longleftarrow 5u_{\text{ref}}\mathscr{u}_{\max}$, which means that $\mathscr{u}$ is multiplied by a factor $0.2/\mathscr{u}_{\max}$ to rescale the velocity. Maintaining $\mathscr{u}_{\max}$ in a small range ensures efficiency (we are never far from the best effiency) while preventing time adaption from having to be performed at each time step, which could potentially hurt accuracy. Fig. 15 shows the variation of effective time steps due to time adaption happening during the simulation of Fig. 18 as a typical example.

Rescaling distribution functions. Note that each time u_{ref} is changed during a simulation, several variables in LBM including velocity and force must be properly rescaled according to Eqs. (18) before proceeding to the next time step. This means that the distribution function f_i and projected force F_i at each grid node need to be both reset to enforce the rescaling of both macroscopic velocity and force. While F_i can be easily evaluated via macroscopic variables, the difficulty of such rescaling is to reconstruct f_i so that the simulation can proceed with as few errors as possible from the reconstruction. For simplicity, we could reset f_i to the Maxwellian equilibrium $\bar{f}_i$, which can be directly computed based on $\wp$ and the rescaled velocity $\mathbb{u}$ (see supplementary material). A more accurate choice is to further add to the equilibrium value the deviation $(\bar{f}_i - f_i)\bar{f}_i'/\bar{f}_i$ to better reproduce the former distribution after rescaling, where $\bar{f}_i'$ is the the Maxwellian equilibrium after rescaling. This reconstruction exhibits very few visible artifacts, and quickly converges to the proper distribution within the following couple of time steps.

Discussion. With our rescaling strategy to adapt the effective time step in order to always safely capture the maximum physical velocities, the immersed boundary penalty forces are now safe to use: not only are we guaranteed to remain within the stability range of the solver, but the accuracy of the integral of the forces exerted by the fluid onto the solid and vice-versa are also improved due to our time adaption. Fig. 16 shows an example computed with a fine, constant effective time step (left) compared to our adaptive treatment (right), showing that the only visible differences between adaptive and non-adaptive simulations happen in highly turbulent regions, whose chaotic nature makes the presence of numerical errors quite unavoidable in the first place. Fig. 15 plots the variation of the effective time steps throughout the simulation from Fig. 18, saving 50% of the computational cost to generate the same duration of animation as the fixed time step.

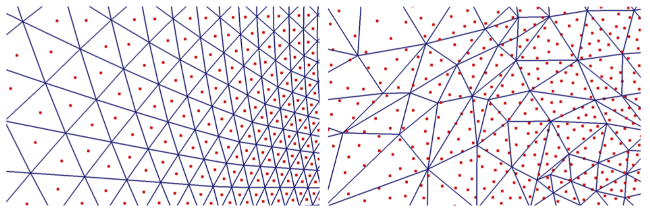

Fig. 17. **Sampling of solid surface.** If the solid is discretized with a mesh with fine, well-shaped triangles (a), one sample per triangle is sufficient to ensure accurate two-way coupling; if the solid mesh is irregular, uniform samples over the mesh (b) are necessary in order to guarantee accurate integration of fluid-solid interaction forces; remeshing is another option.

4.3 Optional spatial adaptivity

Up to now, we assumed that the grid node spacing Δx is constant throughout a simulation. However, improved *spatial* accuracy can also be achieved by using grids with varying spacings. A particularly efficient way to implement this idea is to keep a global grid at a fixed resolution, but to allow local grid refinements (for instance, near obstacles or solids) to increase spatial resolution. This approach leads to local node spacing equal to inverse powers of two of the global Δx, affecting the effective time step Δt as well as the local forces and viscosity in LBM space across different grid resolutions (see Eqs. (18)); note that the grid rescaling does not require rescaling of f_i, but forces need to be rescaled as in the effective time adaptive case. A full treatment of space-time adaption for LBM deserves a longer exposition and a more thorough examination of its numerical limitations, we thus leave it for future work.

5 IMPLEMENTATION DETAILS

We now provide implementation details to help with reproducibility.

Sampling of solid surfaces. Having a sufficient number of boundary samples over the solid is important for the immersed boundary method: each fluid grid cell around the solid should contain enough boundary samples to properly transmit forces to the fluid. If the solid is highly tessellated, then each face center can be used as a boundary sample and the area of the face is used in Eq. (13) to properly approximate the area integral. If, however, the mesh is not quite uniform or too coarse, Poisson disk sampling over the mesh [Yuksel 2015] can be used to distribute the boundary samples well, in which case, the area associated with each sample is defined as the total surface area divided by the total number of solid samples. Fig. 17 illustrates an example of these two sampling methods. Of course, mesh refinement or even remeshing can be used instead to improve the sampling quality, as it only involves pre-processing.

Parallel implementation. Boltzmann solvers are typically highly parallelizable, and our extensions crucially preserve this property as all computations are local, with no global solve needed, implying that the underlying scheme can be highly scalable. Therefore, we implemented our approach on GPU. We used two different implementations: a single-GPU one, using the NVIDIA TITAN XP GPU (except for real-time simulation and performance comparison, where NVIDIA 2080Ti GPU was used) to perform fast simulations for moderate grid sizes or even reach realtime at relatively small grid sizes; and a multi-GPU one on an NVIDIA P40 GPU cluster for high-resolution simulations. In both cases, we used parallel optimization with structure-of-array data layout and atomic summation on CUDA to improve performance. Tab. 3 reports performance statistics of the simulation results shown in this paper.

Rendering. In order to render the smoke, we employ particle tracers. To ensure high quality rendering, a large amount of particles are injected and traced. To render the results, we convert the particles into a density field stored on a high-resolution uniform grid for volume rendering, and employ commercial software Octane [OTOY Octane 2019] for the rendering — except for Figs. 3, 20, and 23, which are rendered by Mitsuba [Jakob 2010].

Immersed boundary kernel. Choosing a sharper kernel for IB-LBM has been shown difficult in previous works due to numerical instability. With our time-adaptive scaling, we can take the smallest, simplest kernel (thus, with the least amount of velocity blurring around the solid boundary), which is expressed as:

$$K(r) = \begin{cases} 1 - r, & r \le 1; \\ 0, & r > 1. \end{cases} \tag{19}$$

For a reasonably fine resolution grid, such a kernel function approximates a delta function on the grid well, ensuring proper handling of boundary conditions near solids.

Enforcing solid boundary conditions. Since the Immersed Boundary method is intrinsically a penalty force method, various types of solid boundary conditions can be easily dealt with. If Neumann boundary condition (i.e., free-slip) is desirable, we simply require $\mathbf{u}_b \cdot \mathbf{n} = \mathbf{u}_s \cdot \mathbf{n}$ to only enforce the normal components of solid and fluid velocities to match. Thus, a force purely along the normal direction is derived as:

$$\mathbf{F}_{s\to f}(\mathbf{X}_p) = \frac{\rho(\mathbf{u}_b - \mathbf{u}_s)\cdot\mathbf{n}}{\Delta t}\,\mathbf{n}\,. \tag{20}$$

Fig. 19 shows a line integral convolution (LIC) visualization of velocity fields near moving boundaries for no-slip v.s. free-slip conditions. As expected, free-slip exhibits a stronger turbulent wake than the no-slip case as evidenced by the additional vortices.

Bootstrapping a simulation. To start a simulation, we need first to determine the physical scales by defining a spatial resolution Δx for the grid, setting a velocity magnitude u_{ref} as described in Sec. 4, defining the fluid density ρ_0, the solid density ρ_s, and the kinematic viscosity ν of the fluid. Note that all these variables are set with real physical units. Then, our dimensional scaling of Eqs. (18) allows us to calculate all variables in LBM scale, and we initialize all f_i with the corresponding equilibrium distribution $\bar{f}_i$. Tab. 3 provides all physical parameters for the different simulations in this paper.

6 RESULTS AND DISCUSSIONS

We now review various tests and examples we computed to evaluate our turbulent fluid integrator and our immersed-boundary based numerical coupling between fluid and solid.

6.1 Simulation tests

In the figures of this paper, we provide a variety of simulation results that highlight the main characteristics of our contribution: we show examples for a large spectrum of Reynolds numbers (from nearly laminar to turbulent flows), for both one-way and two-way coupling, and for coarse and fine computational grids to demonstrate scalability. We discuss a few important points below.

One-way coupling. When solids are immersed in a fluid, the simplest case of fluid-solid coupling is when the motion of the solid (typically scripted or purely driven by gravity) ignores the forces exerted by the fluid, but the fluid reacts to the solid. This one-way coupling is easily achieved in our simulation by artificially setting the forces $\mathbf{F}_{f\to s}$ from Eq. (12) to zero. We show one-way coupling in Fig. 18 for a box moving in both a viscous and a turbulent fluid (only accounting for collisions against the container), and in Fig. 20 where a solid plate (with a certain thickness) is dropped from a low and a high altitude inside a viscous or a turbulent fluid, affecting the surrounding air to form complex smoke volutes.

Two-way coupling. The more complicated fluid-solid coupling is two-way coupling. Fig. 3 demonstrates a two-way simulation where an inlet generates a strong impulse jet flow from the left, pushing two stacks of boxes which end up being blown away — and affecting the flow in the process. Purely passive smoke particles are injected to help visualize the air motion. Fig. 23 demonstrates another example where one-way and two-way coupling are both present — the falling plank does not account for the air influence, but the boxes use two-way coupling. The falling plank makes the air topple the small boxes over, and smoke particles are used once again to help visualize the complexity of the resulting flow.

Realtime simulation on coarse grids. Since our method is very efficient due to its local, highly-parallelizable nature, we can achieve real-time fluid-solid coupling at a grid resolution of 100×50×50, with an average of 500 timesteps per second, leading to around 30 fps for animation depending on how fast the flow is (and dropping to roughly 20 fps if volumetric ray-casting of density field is activated) using an NVIDIA 2080Ti GPU card. Note that due to our accurate and low-dissipation solver, our method still manages to capture and maintain turbulence details on rather coarse resolutions: Fig. 21 depicts a simulation result obtained in realtime on a 50×100×50 grid as an example.

High-resolution simulations. With our multi-GPU implementation, we easily scale up in grid size for high-resolution simulations. We used a grid of size 1200×500×500, and a typical two-way coupling simulation at such a high resolution only takes from 1 to 2 seconds per time step; see Tab. 3 for more timing statistics about high-resolution examples. Fig. 31 shows a high-resolution (1800×400×400) virtual wind tunnel simulation of a high-speed train, while Fig. 2 shows a high-resolution simulation (640×720×640) of a tornado formed by four jets at the bottom of the domain, where solids (cars and cows) inside the tornado are entrained by the rotating turbulent air. In both cases, fine turbulence details can be observed.

6.2 Analysis

In order to further explain and justify our method, we comment on different aspects of our solver below.

Compressible vs. *incompressible fluid solver.* Unlike many fluid solvers in graphics, we do not strictly enforce incompressibility: our solver simulates a low Mach number compressible approximation

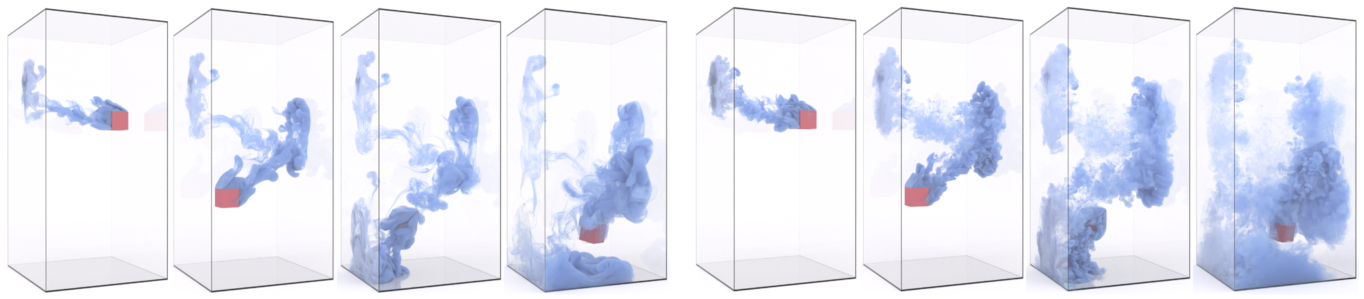

Fig. 18. **Falling box with one-way coupling.** A solid box falls inside a box, causing a flow disturbance; in order to visualize the wake finely, smoke particles are continuously scattered on one side of the box. We use two different Reynolds numbers for the same box motion: Re = 3, 333 in (a)-(d), and Re = 100, 000 in (e)-(h). The latter simulation, involving a less viscous fluid, exhibits higher turbulence as evidenced by the creation of very small vortices.

instead, thus avoiding the need for pressure solves that are hard to parallelize. This approximation is achieved by solving a Boltzmann equation instead, which is linear in advection. We can thus always use a constant time step during the simulation even for strong turbulence, and the advection is easily made conservative, which is usually difficult for many existing solvers in graphics.

Accuracy of coupling. Coupling accuracy deserves further discussion. We employ the immersed boundary method to handle dynamic solid boundaries, which is convenient and efficient but has only *first-order* accuracy. If higher accuracy is called for, we point out that our spatial adaptive scheme can locally increase the resolution of the grid near the solid if so desired. We believe that the solution we propose is a good compromise between efficiency and accuracy that can already handle most scenarios in graphics satisfactorily.

Influence of τ^* perturbation. We performed further tests to compare direct optimization of the high-order relaxation time v.s. the use of a simple linear regression. We noticed that solving for τ^* with direct optimization is quite sensitive to the number of equidistant samples of the function ϵ used to find the minimum: if one hopes to save computational time by selecting only few samples, it actually induces numerical errors that act like a non-physical dynamical instability: for example, in 2D vortex sheet simulation as shown in Fig. 12, too few samples makes vortices shear away at some point, see Fig. 22 (a). However, when we increase the number of samples to increase accuracy, non-physical vortex instability is reduced and longer preservation of the rotating vortex is observed, see Fig. 22 (b) — but the computational cost needed for optimization increases dramatically. The linear regression as our final predictor for τ^* is thus not only an efficient alternative to direct optimization, but in fact, a better solution since it naturally acts as a filter to remove numerical perturbations of τ^*, see Fig. 22 (c). This last statement

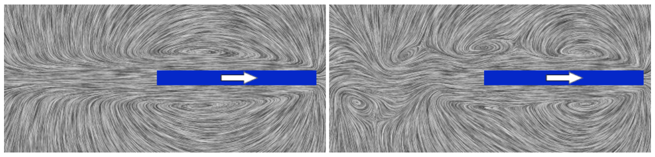

Fig. 19. **Boundary conditions.** Our two-way coupling can accommodate either no-slip (a) or free-slip (b) boundary condition; note the difference in turbulence patterns visualized by this LIC rendering of a slice of the 3D velocity field in a simulation where the box moves to the right.

was confirmed by performing a linear regression through a least squares solution based on (noisy) high-order relaxation times found through direct optimization with relatively few samples: this modified linear regression performed in fact better than the simulation using directly the poorly-optimized relaxation times.

Dispersion error. While rarely mentioned in practice, current Navier-Stokes solvers suffer from large dispersion errors when small time steps are used. This is not an issue when computations are performed with reasonable time steps, but two-way coupling often imposes much stronger restrictions on the time step size, especially for coupling in the case of turbulent flows where both fluids and solids can move very fast, making dispersion a real issue both numerically and visually. In order to illustrate the low dispersion property of our solver, we conducted a 2D double layer vortex simulation [Minion and Brown 1997], which is a simple vortical flow for which increasing grid resolution should not result in additional vortex structures. In Fig. 24, the top row (a–d) are simulations with the MC+R Navier-Stokes solver, for increasing resolutions (from 256×256 to 2048×2048, for the same unit domain), using a relatively small time step ($\Delta t = 0.001$): strong dispersion errors induce visible non-physical instability. *Even* if we enlarge the time step to $\Delta t = 0.01$ to try to suppress dispersion, non-physical secondary vortices still appear for relatively coarse simulations (see (e) and (f), at 256×256 and 1024× 1024 resp.). In contrast, our kinetic solver does not induce non-physical secondary vortices even at the lowest grid resolution (see (g) and (h), at 256×256 and 1024× 1024 resp.)) with the same small (effective) time step of $\Delta t = 0.001$. Increasing the resolution to 1024×1024 also results in similar vortex details compared to the MC+R Navier-Stokes solver at the same high resolution.

Convergence analysis. We also analyze the convergence of our solver based on several simulation experiments. In our first test, we ignore immersed solids and consider the case of 2D Taylor-Green vortex simulation again as shown in Fig. 11 (a) for which a ground truth solution is known. By varying spatial and time steps, we can plot the convergence graph as shown in Fig. 25, from which we can draw a few conclusions. First, compared to the MC+R solver, our error is much smaller (note our use of a log-log plot). MC+R solver has third-order convergence for a certain range of time steps (consistent with what is reported in their paper) while our solver is second-order in time. Note that we did not plot the curves for too small time steps to avoid having their dispersion errors obfuscating

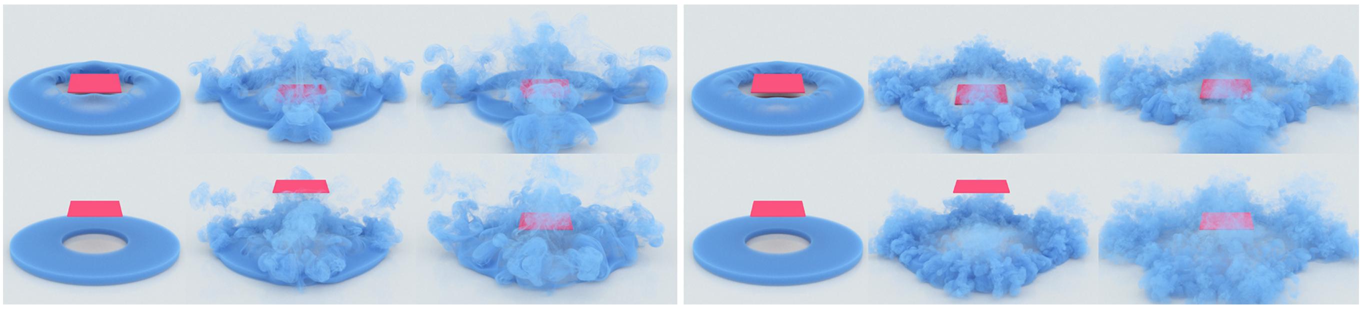

Fig. 20. **Plank drop with one-way-coupling.** A solid plank is dropped (and bounces back several times) from a low (top row) and high (bottom row) height, whose wake affects a static smoke ring. We show the resulting behavior for two different Reynolds numbers: Re=14,000 in (a)-(d), and Re = 140, 000 in (e)-(h).

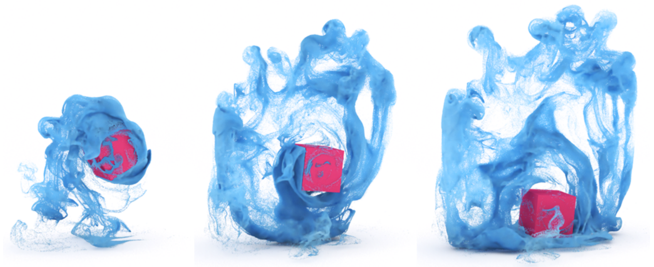

Fig. 21. **Coarse-grid simulation.** A falling box using one-way coupling with the surrounding fluid is computed in realtime on a coarse 50×100×50 grid. Due to our solver accuracy, turbulence details are still well captured.

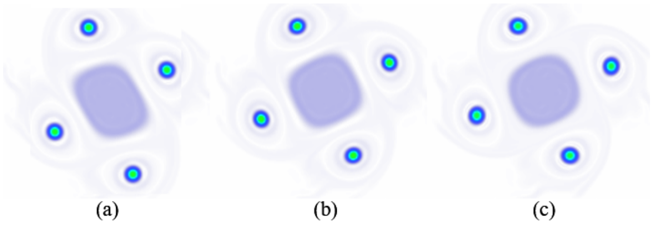

Fig. 22. **Influence of relaxation time perturbation.** If too few samples of the measurement functional are used during optimization, the resulting perturbations of τ^* create non-physical artifacts (a), shearing the otherwise symmetrically-rotating vortices; a finer sampling removes these pertuba-tions, resulting in a better symmetry (b); however, our linear regression applied to the noisy optimization values effectively reduces perturbations of τ^*, removing most (but not all) artifacts from the inaccurate optimizations.

the curve. However, for spatial convergence, our solver is not only leading to smaller errors, but it also converges at a rate larger than second order, a higher convergence compared the MC+R solver.

Time step restriction. A discussion about time step restrictions for traditional Navier-Stokes solvers and our kinetic solver is also worthwhile. In general, many Navier-Stokes solvers allow a rather large range of time steps given a fixed spatial grid resolution and physical scale (see Fig. 30), but small time steps usually introduce dispersion errors and should be avoided as we discussed earlier. In contrast, our kinetic solver allows for very small effective time steps by adjusting u_{ref}; there is, however, an upper limit on the time step size, above which the solver becomes neither accurate nor stable (a lower limit also exists due to machine accuracy to represent floating-point numbers). Since this upper limit is around 0.3 times the Courant number for flows of normal speed, we believe it is a rather reasonable constraint — even more so given the efficiency of our time steps. Besides, as argued earlier, two-way coupling is typically requiring very small time steps for interesting scenarios.

Table 2. **Profiling.** Timings of each step for MC+R and our algorithm.

	MC+R solver			**Our LBM solver**		
Grid Resolution	*Advection*	*Diffusion*	*Projection*	*Streaming*	*Immersed Bound.*	*Collision*
50×100×50	8.54ms	3.4ms	10.25ms	0.19ms	0.92ms	0.77ms
100×200×100	56.2ms	13.5ms	157.36ms	1.38ms	2.3ms	6.19ms
150×300×150	169.89ms	39.77ms	834.36ms	4.6ms	3.5ms	17.04ms
200×400×200	354.27ms	100.9ms	3166.16ms	11.23ms	8.08ms	32.6ms

Performance. Semi-Lagrangian solvers and game engines such as [NVIDIA FlameWorks 2015] have been shown to be extremely efficient for fluid animation: most of these fluid solvers can take large time steps very efficiently by, for instance, relaxing the accuracy of the pressure solve (via Jacobi iterations and/or fixed iteration counts) or adopting a multigrid approach. However, this efficiency often comes at the price of significant visual degradation: as an example, Fig. 30 shows how the semi-Lagrangian MC+R solver behaves at different timestep sizes, demonstrating that large timesteps lead to rather unrealistic behaviors. Moreover, the use of such large time steps prevents two-way coupling. Since our work emphasizes accuracy and predictivity instead, a direct and exhaustive comparison of efficiency with existing solvers is quite arduous. In order to offer a fair analysis of the efficiency of our solver compared to existing semi-Lagrangian solvers, we used our own GPU implementation of the stable fluids method [Stam 1999] with a CUDA-optimized preconditioned conjugate gradient (PCG) solver from NVIDIA for the pressure solve, run on an NVIDIA 2080Ti GPU card. PCG iterations were stopped when the error reached a low threshold or when the number of iterations reached 300, to allow for reasonable, yet not too stringent accuracy levels. We also used a timestep size ten times larger than our LBM physical timestep size, to mimic the typical setup of stable fluids. Fig. 26 compares computational times for this instance of stable fluids vs. our LBM solver on a one-second fluid animation of a turbulent jet flow through a ball (as in Fig. 30), showing that the embarrassing parallelism of LBM results in increasingly better efficiency as grid resolution increases. Note that a multigrid implementation on the GPU of the pressure solve may improve the timings of stable fluids by a factor 5 if the speedup witnessed in CPU implementations like [Qu 2019] holds; but the pressure solve is a non-trivial obstacle to parallelization in general, so our LBM approach will outperform a stable fluid solver (or any of its more involved variants) for a large enough number of cores and/or GPUs. We also measured the performance of the MC+R method using the

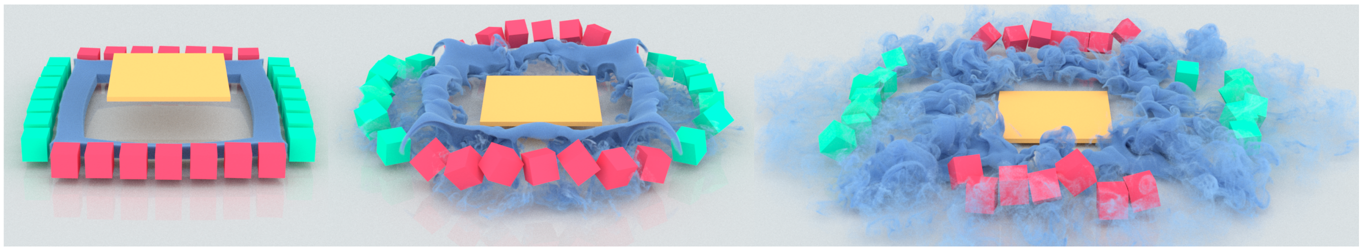

Fig. 23. **Falling plank toppling boxes.** In this example, we simulate a falling plank ignoring air interaction, but boxes with two-way coupling. The fall creates a strong draft which topples the boxes over, and passive smoke is used to visualize the air motion.

Table 3. **Statistics.** Parameters and performance statistics for the simulations shown in the paper.

Figure	*Grid Resolution*	*Solid Samples*	ν	Δx (m)	Δt (s)	*#GPUs*	*time per* Δt	*Total time steps*	*Fixed/Adaptive*	*Total time*
Fig. 1	500×1200×500	182,897	2×10^{-4}	1/150	1.6×10^{-5}	4	1.43 sec.	41,415	Fixed	987.1 mins.
Fig. 2	640×720×640	1,234,016	8×10^{-4}	1/40	*varying*	4	1.39 sec.	17,700	Adaptive	410.5 mins
Fig. 3	600×200×200	120,000	$6\times10^{-3}/2\times10^{-3}$	1/20	6.25×10^{-5}	1	0.42 sec.	32,450	Fixed	227.2 mins.
Fig. 18	200×400×200	21,600	$8\times10^{-3}/2\times10^{-4}$	1/40	*varying*	1	0.26 sec.	40,000	Adaptive	173.3 mins.
Fig. 20	400×200×400	41,440	$4\times10^{-3}/4\times10^{-4}$	1/80	*varying*	1	0.56 sec.	30,000	Adaptive	280 mins.
Fig. 21	50×100×50	9,600	4×10^{-3}	1/15	3.2×10^{-3}	1	1.9×10^{-3} sec.	3,000	Fixed	5.7 sec.
Fig. 23	540×260×540	97,920	1.6×10^{-3}	1/60	2.5×10^{-4}	1	1.22 sec.	12,800	Fixed	260.3 mins.
Fig. 31	1800×400×400	513,270	8×10^{-4}	1/100	5×10^{-5}	4	1.42 sec.	25,000	Fixed	591.6 mins.

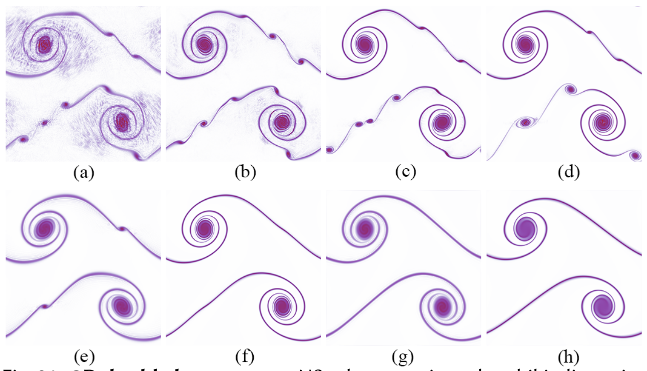

Fig. 24. **2D double layer vortex.** NS solvers consistently exhibit dispersion errors at small time steps: e.g., [Zehnder et al. 2018] for $\Delta t = 0.001$ and for power-of-two resolutions from 256^2 (a) to 2048^2 (d). Even for $\Delta t = 0.01$, dispersion errors appear at low (256^2) (e) & high (1024^2) (f) resolutions. Our kinetic solver does not exhibit this numerical issue at the same low (g) and high (h) grid resolutions for an effective time step size of $\Delta t = 0.001$.

same PCG solver for the same test using various timestep sizes: in Fig. 30, our LBM simulation shown in (d) using the same physical timestep size (3×10^{-3}s) and grid resolution as the MC+R result shown in (c) exhibits smaller-scale turbulence, yet runs 65 times faster; our LBM solver is still 11 times faster than MC+R if it now uses a 7× larger time step (b), but MC+R becomes slightly more efficient than our solver if a 42× larger timestep is used (a). Moreover, if we use a resolution reduced by a factor 8 (i.e., a 100×200×100 grid) in our LBM simulation, the visual result (e) is still superior to MC+R, yet it is now 16 times faster than (a), 110 times faster than (b), and over 670 times faster than (c). These results, obviously, depend heavily on the implementation used to compare (in particular, if a multigrid PCG solver is used), but they indicate a clear advantage in performance for our massively parallelizable algorithm compared to existing approaches when accuracy and visual complexity are taken into account. For completeness, we also provide timings of the various steps involved in one timestep of our MC+R implementation described above and of our LBM integrator in Tab. 2.

Comparison to existing coupling work. Fig. 27 shows an animation for which the whole fluid motion is created due to the no-slip condition of a cylinder rotating around its main axis: from a static velocity field, the cylinder motion begins affecting the nearby ring of smoke, exhibiting slight fluttering at first, and eventually dissipating the whole ring. This simulation was computed on a 300×150×300 grid. In

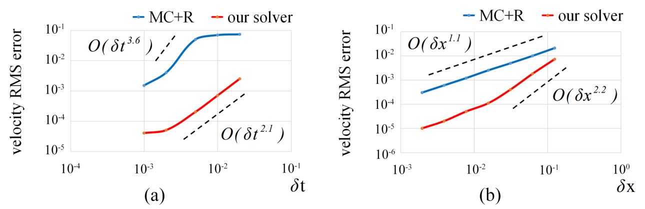

Fig. 25. **Convergence for 2D Taylor-Green vortex simulation.** We compare the reflection-advection solver (MC+R) [Zehnder et al. 2018] and ours in terms of (a) temporal convergence and (b) spatial convergence using the known analytical solution of the 2D Taylor-Green vortex simulation. Note that both plots are in log-log scale.

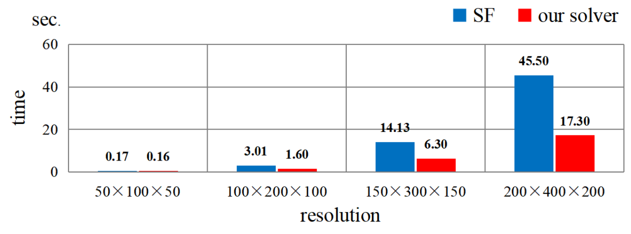

Fig. 26. **Performance.** We compare timings (for one second of animation) of our GPU-based kinetic solver with a CUDA-optimized implementation of stable fluids, where the timestep used in the stable fluids solver is 10 times larger than our solver, and the conjugate-gradient solver used in stable fluids stops when the error is below a small threshold or when it exceeds 300 iterations; as the grid resolution grows, the massive parallelism of our approach outperforms the semi-Lagrangian solvers quite significantly.

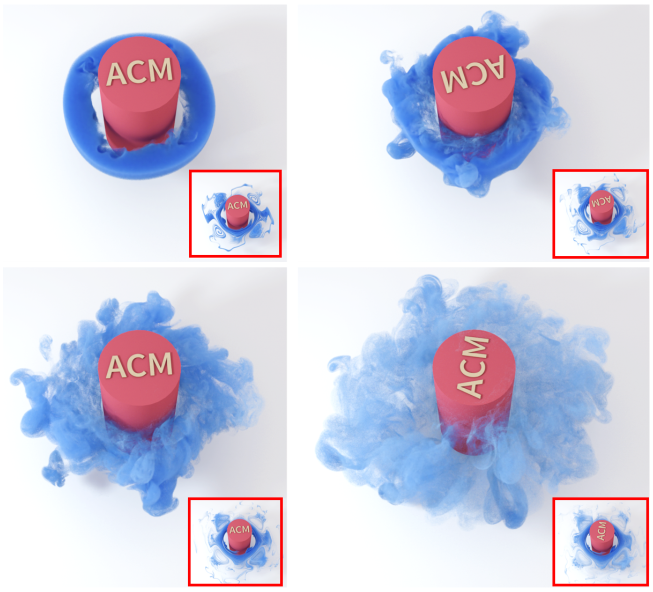

Fig. 27. **Drag-induced flow.** Starting from a static fluid, the rotation of cylinder with no-slip condition induces boundary layers which quickly affects the entire surrounding as evidenced by the early fluttering of an initially static smoke ring, and its entire dispersion as time evolves. The same simulation computed via a spectral solver [Cui et al. 2018] (insets, at corresponding times) cannot capture the same behavior due to the large amount of bases needed to capture fine details and their non-local support.

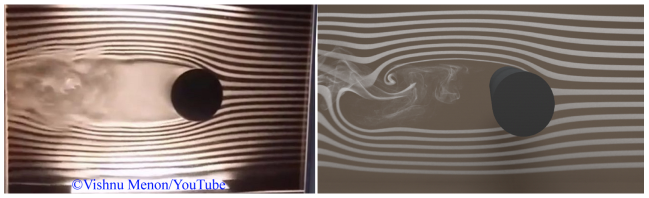

Fig. 28. **Small-scale virtual wind tunnel.** A simple setup where the right-side of a box acts as an inlet to simulate a wind tunnel reproduces the real experiment (top) of the usual flow over a cylinder (generating vortices) using several sources of passive white smoke particles (bottom).

comparison, the fluid-solid coupling approach from [Cui et al. 2018] cannot capture this fine interaction due to the global support of the basis functions and/or the number of bases used in this spectral simulation; we show their result as inset on the same figure, where we used the same grid resolution as in our simulation, with 5,000 basis functions to capture turbulence details.

Comparison with real experiments. Lastly, we provide two comparisons with real experiments. In the first example, we reproduce a real portable wind-tunnel where the air motion over a cylinder is highlighted by injecting several lines of smoke, see Fig. 28. We simply add white particles to the visualization to simulate similar smoke lines, exhibiting a visually similar wake. While our solid was held static in the first comparison, we also compare the flow created by a falling rectangular plane. The real experiment was produced through high-speed Schlieren photography [Liu et al. 2018]. We tried to simulate the same setup, and visualized the resulting 3D wake flow using a volume rendering of the norm of the velocity gradient to simulate Schlieren photography. The complexity of the chaotic flow is well captured by our simulation as Fig. 29 demonstrates.

6.3 Limitations

Finally, we conclude this section with limitations of our method. First, our solver cannot reliably support coupling fluids to thin rods or shells (i.e., with a thickness of the order of the grid spacing), where velocity gradients are very strong near the boundary, such as the turbulent jet hitting a thin shell, as force spreading may not enforce boundary conditions well in this case. Second, memory usage is relatively larger than traditional Navier-Stokes solvers for the same grid resolution — usually two to three times larger depending on which existing solver we compare to. Considering the higher effective accuracy and the significant performance improvement we provide, we believe it is a tolerable practical limitation, especially since our solver can produce very comparable visual results with a coarse grid as demonstrated in Fig. 30. Third, our solver does not allow very large time steps compared to existing semi-Lagrangian Navier-Stokes solvers. This is, however, not a particularly limiting issue: our solver is still more efficient than other solvers at equivalent accuracy levels, especially at high resolutions. Moreover, the guarantees we offer make our solver not only well suited for visual effects, but for CFD simulations as well where large time steps are avoided to maintain temporal accuracy. Finally, maybe the most important limitation is cultural: our solver cannot be implemented by just tweaking an existing Navier-Stokes solver. We realize that this fact is probably the most significant impediment to its quick adoption. However, all our simulation results and comparisons demonstrate that our kinetic solver performs better in both visual quality and efficiency, the two most important factors in graphics.

7 CONCLUSION

In this paper, a novel method for fluid simulation is proposed to enable an accurate, efficient, and robust simulation of complex fluid-solid coupling scenarios. In order to achieve this goal, we adopt a kinetic simulation framework based on the lattice Boltzmann method,

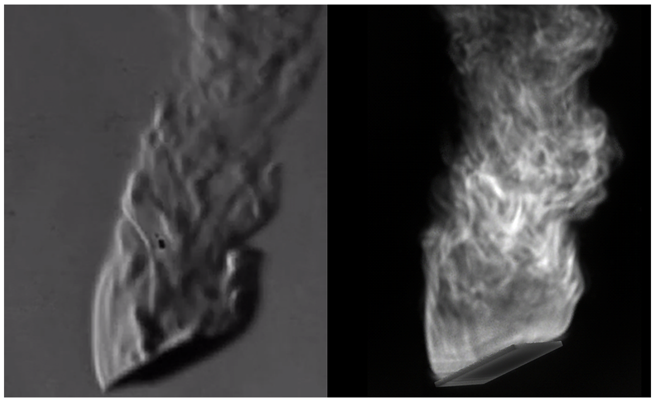

Fig. 29. **Falling plate.** Our two-way coupling simulation is used to compare the wake of a falling 3D rectangular plate with a real experiment obtained via high-speed Shlieren photography [Liu et al. 2018] (left); our simulation approximating the same setup (right) is visualized via 3D volume rendering of the velocity gradient, exhibiting similar wake flow structures.

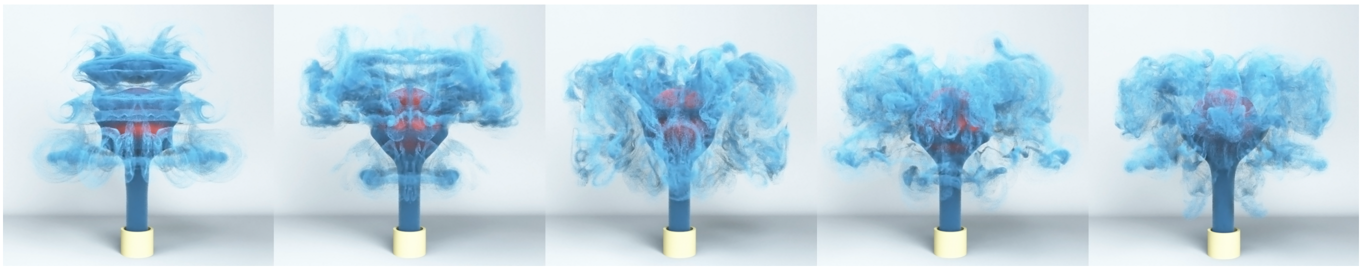
(a) MC+R with 42Δt
(b) MC+R with 7Δt
(c) MC+R with Δt
(d) Ours with Δt
(e) Ours, 8× coarser res.

Fig. 30. **Semi-Lagrangian solver *vs.* LBM.** For a simple, turbulent jet flow through a sphere obstacle computed over a domain of size 1m×2m×1m for Re=20, 000 and a jet speed of $0.4m/s$, we compare the results of our own GPU-optimized implementation of the MC+R solver [Zehnder et al. 2018] computed on a $200 \times 400 \times 200$ grid for a timestep size that is (a) 42 times (resp., (b) 7 times) larger than the one in (c). We also show the result of our LBM solver on a grid of size $100 \times 200 \times 100$ (i.e., 8 times coarser) in (e), still exhibiting small-scale turbulence that visually matches the finer LBM simulation. While using large timesteps in semi-Lagrangian methods may lead to higher performance, the resulting visual differences with our LBM solver are pronounced.

for which we introduce a new regression-based evaluation of the high-order relaxation rates of the central-moment relaxation model. The resulting fluid integrator exhibits significantly less dissipation and less dispersion errors than existing Navier-Stokes solvers. Two-way coupling is implemented via the immersed boundary method, of which we enforce stability by rescaling time steps (and potentially, the spatial grid size too if needed) based on the maximum velocity. Through a variety of simulation results and extensive comparisons, we prove the accuracy and efficiency of our method, and demonstrate that it outperforms existing Navier-Stokes solvers and fluid-solid coupling methods used in graphics.

Future work. As discussed in the paper, spatial adaptivity deserves more analysis and study, and how to construct space-time adaptive simulation with a hierarchy of grids is an interesting topic to explore. Additionally, since our method is highly scalable, it can be further extended to multi-node, multi-GPU systems. Since the amount of data copy is not large due to the locality of our scheme, we expect to achieve real-time, or at least interactive, high-resolution fluid simulation. Furthermore, finding theoretical grounds which could explain why our simple linear regression is so powerful in practice would be satisfying. Finally, extending our two-way coupling to handle thin and deformable bodies in a robust manner is an obvious direction for future work.

ACKNOWLEDGMENTS

We thank Qiaodong Cui from UC Santa Barbara, Jonas Zehnder from U. of Montreal, and Ziyin Qu from UPenn for sharing their codes for comparisons. We also thank Cangli Yao, Yihui Ma, Yiran Sun and Chenqi Luo from ShanghaiTech University for helping with rendering and video editing, as well as Christian Lessig for comments. Yixin Chen also received generous support from DGene Digital Technology. The train mesh is from `aigei.com`; the cow mesh is from `free3D.com`; and the rocket and car meshes are both from `cgtrader.com`. This work was supported by a startup funding from ShanghaiTech University, and in part by the US National Science Foundation (1717178, 1816041, 1910839). Finally, Mathieu Desbrun gratefully acknowledges the hospitality of ShanghaiTech University during his sabbatical.

REFERENCES

Cyrus K Aidun and Jonathan R Clausen. 2010. Lattice-Boltzmann method for complex flows. *Annual review of fluid mechanics* 42 (2010), 439–472.
Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph.* 31, 4 (2012), 62.
Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph.* 32, 4, Article 103 (2013).
Vinicius C Azevedo, Christopher Batty, and Manuel M Oliveira. 2016. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Trans. Graph.* 35, 4 (2016), 97.
Stefan Band, Christoph Gissler, Markus Ihmsen, Jens Cornelis, Andreas Peer, and Matthias Teschner. 2018. Pressure boundaries for implicit incompressible SPH. *ACM Trans. Graph.* 37, 2 (2018), 14.
Stefan Band, Christoph Gissler, and Matthias Teschner. 2017. Moving least squares boundaries for SPH fluids. In *VRIPHYS*. 21–28.
Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. In *ACM Trans. Graph.*, Vol. 26. 100.
Markus Becker and Matthias Teschner. 2007. Weakly compressible SPH for free surface flows. In *Symposium on Computer Animation*. 209–217.
Markus Becker, Hendrik Tessendorf, and Matthias Teschner. 2009. Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Trans. Vis. Comp. Graph.* 15, 3 (2009), 493–503.
Jan Bender and Dan Koschier. 2016. Divergence-free SPH for incompressible and viscous fluids. *IEEE Trans. Vis. Comp. Graph.* 23, 3 (2016), 1193–1206.
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. *Phys. Rev.* 94 (1954), 511–525. Issue 3.
Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-time Smoke Animation with Vortex Sheet Meshes. In *Symposium on Computer Animation*. 87–95.
Mark Carlson, Peter J Mucha, and Greg Turk. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM Trans. Graph.*, Vol. 23. 377–384.
Shiyi Chen and Gary D Doolen. 1998. Lattice Boltzmann method for fluid flows. *Annual review of fluid mechanics* 30, 1 (1998), 329–364.
Mengyu Chu and Nils Thürey. 2017. Data-driven synthesis of smoke flows with CNN-based feature descriptors. *ACM Trans. Graph.* 36, 4, Article 69 (2017).
Pascal Clausen, Martin Wicke, Jonathan R Shewchuk, and James F O'brien. 2013. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Trans. Graph.* 32, 2 (2013), 17.
Andrea Colagrossi and Maurizio Landrini. 2003. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *J. Comput. Phys.* 191, 2 (2003), 448–475.
Qiaodong Cui, Pradeep Sen, and Theodore Kim. 2018. Scalable Laplacian Eigenfluids. *ACM Trans. Graph.* 37, 4, Article 87 (2018).
Meizhong Dai and David P Schmidt. 2005. Adaptive tetrahedral meshing in free-surface flow. *J. Comput. Phys.* 208, 1 (2005), 228–252.
Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power particles: an incompressible fluid solver based on power diagrams. *ACM Trans. Graph.* 34, 4, Article 50 (2015).
Alessandro De Rosis. 2017. Nonorthogonal central-moments-based lattice Boltzmann scheme in three dimensions. *Physical Review E* 95, 1 (2017), 013310.
Alessandro De Rosis, Rongzong Huang, and Christophe Coreixas. 2019. Universal formulation of central-moments-based lattice Boltzmann method with external forcing for the simulation of multiphysics phenomena. *Physics of Fluids* 31, 11 (2019), 117102.
Alessandro De Rosis and Kai H. Luo. 2019. Role of higher-order Hermite polynomials in the central-moments-based lattice Boltzmann framework. *Physical Review E* 99, 1

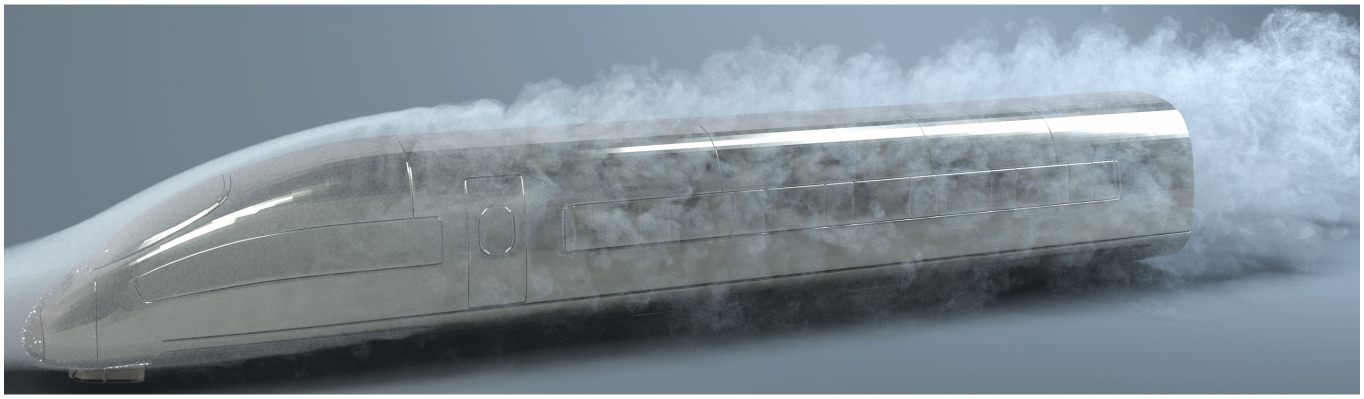

Fig. 31. **Large-scale virtual wind tunnel.** A static high-speed train is placed in a virtual wind tunnel (pushing air from the left); at a Reynold number of Re = 28, 000 and with no-slip condition inducing boundary layer turbulence, passive white particles visualize the flow separation and ensuing vortices. This simulation was computed at a resolution of 1800×400×400 on a 4-GPU server, each frame (i.e., 1/50 of a second) requiring 142 seconds to simulate.

(2019), 013301.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed Particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation.* 61–76.

Dominique d'Humières. 2002. Multiple–relaxation–time lattice Boltzmann models in three dimensions. *Philos. Trans. R. Soc. A* 360, 1792 (2002), 437–451.

Essex Edwards and Robert Bridson. 2014. Detailed water with coarse grids: combining surface meshes and adaptive discontinuous Galerkin. *ACM Trans. Graph.* 33, 4 (2014), 136.

Sharif Elcott, Yiying Tong, Eva Kanso, Peter Schröder, and Mathieu Desbrun. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph.* 26, 1 (2007), 4.

Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2018. A Multi-scale Model for Simulating Liquid-fabric Interactions. *ACM Trans. Graph.* 37, 4, Article 51 (2018).

Bryan E Feldman, James F O'brien, and Bryan M Klingner. 2005. Animating gases with hybrid meshes. In *ACM Trans. Graph.*, Vol. 24. 904–909.

Zhi-Gang Feng and Efstathios E Michaelides. 2004. The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems. *J. Comput. Phys.* 195, 2 (2004), 602–628.

Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. *Graph. Models Image Process.* 58, 5 (1996), 471–483.

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A Polynomial Particle-In-Cell Method. *ACM Trans. Graph. (SIGGRAPH)* 36, 6 (2017), 222:1–222:12.

Martin Geier, Andreas Greiner, and Jan G. Korvink. 2006. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Physical Review E* 73, 6 Pt 2 (2006), 066705–066705.

Olivier Génevaux, Arash Habibi, and Jean-Michel Dischler. 2003. Simulating Fluid-Solid Interaction.. In *Graphics Interface.* 31–38.

Frédéric Gibou and Chohong Min. 2012. Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions. *J. Comput. Phys.* 231, 8 (2012), 3246–3263.

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. (SIGGRAPH ASIA)* 31, 6, Article 148 (2012).

Philip M. Gresho and Robert L. Sani. 1987. On pressure boundary conditions for the incompressible Navier-Stokes equations. *Int. J. Numer. Methods Fluids* 7, 10 (1987), 1111–1145.

Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. 2005. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.* 24, 3 (2005), 973–981.

Yulong Guo, Xiaopei Liu, and Xuemao Xu. 2017. A unified detail-preserving liquid simulation by two-phase lattice Boltzmann modeling. *IEEE Trans. Vis. Comp. Graph.* 23, 5 (2017), 1479–1491.

Cyrill W Hirt, Anthony A Amsden, and JL Cook. 1974. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *J. Comput. Phys.* 14, 3 (1974), 227–253.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph.* 37, 4 (2018), 150.

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2013. Implicit incompressible SPH. *IEEE Trans. Vis. Comp. Graph.* 20, 3 (2013), 426–435.

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2014. Implicit incompressible SPH. *IEEE Trans. Vis. Comp. Graph.* 20, 3 (2014), 426–435.

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, Markus Gross, et al. 2015. Data-driven fluid simulations using regression forests. *ACM Trans. Graph.* 34, 6, Article 199 (2015).

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The Affine Particle-in-cell Method. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 51 (2015).

Dirk Junk, Michael; Kehrwald. 2006. *On the relation between lattice variables and physical quantities in lattice Boltzmann simulations.* Technical Report. Fraunhofer Institute for Industrial Mathematics. http://nbn-resolving.de/urn:nbn:de:bsz:352-254002

ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. 2005. FlowFixer: Using BFECC for Fluid Simulation. In *Eurographics Conference on Natural Phenomena.* 51–56.

Bryan M Klingner, Bryan E Feldman, Nuttapong Chentanez, and James F O'brien. 2006. Fluid animation with dynamic meshes. *ACM Trans. Graph.* 25, 3 (2006), 820–825.

Dan Koschier and Jan Bender. 2017. Density maps for improved SPH boundary handling. In *Symposium on Computer Animation.* Article 1.

Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. 2017. *The Lattice Boltzmann Method.* Springer International Publishing.

Uğis Lācis, Kunihiko Taira, and Shervin Bagheri. 2016. A stable fluid–structure-interaction solver for low-density rigid bodies using the immersed boundary projection method. *J. Comput. Phys.* 305 (2016), 300–318.

Jonas Latt. 2008. *Choice of units in lattice Boltzmann simulations.* Technical Report. http://lbmethod.org/_media/

Michael Lentine, Wen Zheng, and Ronald Fedkiw. 2010. A Novel Algorithm for Incompressible Flow Using Only a Coarse Grid Projection. *ACM Trans. Graph.* 29, 4 (2010), Art. 114.

Wei Li, Kai Bai, and Xiapei Liu. 2019. Continuous-Scale Kinetic Fluid Simulation. *IEEE Trans. Vis. Comp. Graph.* 25, 9 (2019), 2694–2709.

Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu. 2020. Kinetic-based Multiphase Flow Simulation. *IEEE Trans. Vis. Comp. Graph.* (2020).

Zhe Li, Julien Favier, Umberto D'Ortona, and Sébastien Poncet. 2016. An immersed boundary-lattice Boltzmann method for single-and multi-component fluid flows. *J. Comput. Phys.* 304 (2016), 424–440.

Beibei Liu, Gemma Mason, Julian Hodgson, Yiying Tong, and Mathieu Desbrun. 2015. Model-reduced Variational Fluid Simulation. *ACM Trans. Graph.* 34, 6, Article 244 (2015).

Xiaopei Liu, Wai-Man Pang, Jing Qin, and Chi-Wing Fu. 2014. Turbulence simulation by adaptive multi-relaxation lattice Boltzmann modeling. *IEEE Trans. Vis. Comp. Graph.* 20, 2 (2014), 289–302.

Yun Liu, Nuri Zeytinoglu, and Jiliang Li. 2018. Gallery of Fluid Motion - High-speed Schlieren photography of falling objects. *Physical Review Fluids* (2018).

Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.* 23, 3 (2004), 457–462.

Xin Lv, Qingping Zou, Yong Zhao, and Dominic Reeve. 2010. A novel coupled level set and volume of fluid method for sharp interface capturing on 3D tetrahedral grids. *J. Comput. Phys.* 229, 7 (2010), 2573–2604.

Mathworks, Inc. 2017. *MATLAB version 9.3 (R2017b).* Mathworks, Inc.

Renwei Mei, Li-Shi Luo, and Wei Shyy. 1999. An accurate curved boundary treatment in the lattice Boltzmann method. *J. Comput. Phys.* 155, 2 (1999), 307–330.

Michael L Minion and David L Brown. 1997. Performance of under-resolved two-dimensional incompressible flow simulations, II. *J. Comput. Phys.* 138, 2 (1997), 734–765.

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. 2009. Energy-preserving Integrators for Fluid Animation. *ACM Trans. Graph. (SIGGRAPH)* 28, 3 (2009), Art. 38.

Yen Ting Ng, Chohong Min, and Frédéric Gibou. 2009. An efficient fluid-solid coupling algorithm for single-phase flows. *J. Comput. Phys.* 228, 23 (2009), 8807–8829.

NVIDIA. 2015. FlameWorks: generating fire, smoke and explosion effects for games. https://developer.nvidia.com/flameworks.

OTOY, Inc. 2019. OctaneRender. https://home.otoy.com/render/octane-render/.

Sang Il Park and Myoung Jun Kim. 2005. Vortex Fluid for Gaseous Phenomena. In *Symposium on Computer Animation.* 261–270.

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An Implicit Viscosity Formulation for SPH Fluids. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 114 (2015).

Charles S Peskin. 1972. Flow patterns around heart valves: a numerical method. *J. Comput. Phys.* 10, 2 (1972), 252–271.

Charles S Peskin. 2002. The immersed boundary method. *Acta numerica* 11 (2002), 479–517.

Tobias Pfaff, Nils Thürey, and Markus Gross. 2012. Lagrangian Vortex Sheets for Animating Fluids. *ACM Trans. Graph. (SIGGRAPH)* 31, 4, Article 112 (2012).

Linhai Qiu, Yue Yu, and Ronald Fedkiw. 2015. On thin gaps between rigid bodies two-way coupled to incompressible flow. *J. Comput. Phys.* 292 (2015), 1–29.

Ziyin Qu. 2019. BiMocq: Algebraic multigrid implementation on CPU. Found on Github, at: https://github.com/ziyinq/Bimocq.

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Trans. Graph.* 38, 4, Article 128 (2019).

Karthik Raveendran, Chris Wojtan, and Greg Turk. 2011. Hybrid Smoothed Particle Hydrodynamics. In *Symposium on Computer Animation.* 33–42.

Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. In *ACM Trans. Graph.*, Vol. 27. 46.

Doug Roble, Nafees bin Zafar, and Henrik Falt. 2005. Cartesian grid fluid simulation with irregular boundary voxels. In *ACM SIGGRAPH Sketches.* 138.

Hagit Schechter and Robert Bridson. 2012. Ghost SPH for animating water. *ACM Trans. Graph.* 31, 4 (2012), 61.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An Unconditionally Stable MacCormack Method. *J. Sci. Comput.* 35, 2-3 (2008), 350–371.

Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A vortex particle method for smoke, water and explosions. In *ACM Trans. Graph.*, Vol. 24. 910–914.

Jung Hee Seo and Rajat Mittal. 2011. A Sharp-Interface Immersed Boundary Method with Improved Mass Conservation and Reduced Spurious Pressure Oscillations. *J. Comput. Phys.* 230, 19 (2011), 7347–7363.

Rajsekhar Setaluri, Mridul Aanjaneya, Sean Bauer, and Eftychios Sifakis. 2014. SPGrid: A Sparse Paged Grid Structure Applied to Adaptive Smoke Simulation. *ACM Trans. Graph. (SIGGRAPH ASIA)* 33, 6, Article 205 (2014).

Xiaowen Shan, Xue-Feng Yuan, and Hudong Chen. 2006. Kinetic theory representation of hydrodynamics: a way beyond the Navier–Stokes equation. *Journal of Fluid Mechanics* 550 (2006), 413–441.

Barbara Solenthaler and Renato Pajarola. 2008. Density contrast SPH interfaces. In *Symposium on Computer Animation.* 211–218.

B. Solenthaler and R. Pajarola. 2009. Predictive-corrective Incompressible SPH. *ACM Trans. Graph. (SIGGRAPH),* Article 40 (2009).

Jos Stam. 1999. Stable fluids. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques.* 121–128.

Tsunemi Takahashi, Heihachi Ueki, Atsushi Kunimatsu, and Hiroko Fujii. 2002. The simulation of fluid-rigid body interaction. In *ACM SIGGRAPH Sketches.* 266–266.

Yun Teng, David I. W. Levin, and Theodore Kim. 2016. Eulerian Solid-fluid Coupling. *ACM Trans. Graph.* 35, 6, Article 200 (2016).

Nils Thürey. 2007. Physically based animation of free surface flows with the lattice Boltzmann method. *Ph. D. Thesis, University of Erlangen* (2007).

Nils Thürey, Theodore Kim, and Tobias Pfaff. 2013. Turbulent Fluids. In *ACM SIGGRAPH Courses.* Art. 6.

Nils Thürey and Ulrich Rüde. 2009. Stable free surface flows with the lattice Boltzmann method on adaptively coarsened grids. *Computing and Visualization in Science* 12, 5 (2009), 247–263.

Daniel Weber, Johannes Mueller-Roemer, André Stork, and Dieter Fellner. 2015. A Cut-Cell Geometric Multigrid Poisson Solver for Fluid Simulation. In *Comp. Graph. Forum*, Vol. 34. 481–491.

M Weickert, G Teike, O Schmidt, and M Sommerfeld. 2010. Investigation of the LES WALE turbulence model within the lattice Boltzmann framework. *Computers & Mathematics with Applications* 59, 7 (2010), 2200–2214.

Steffen Weißmann and Ulrich Pinkall. 2010. Filament-based Smoke with Vortex Shedding and Variational Reconnection. *ACM Trans. Graph. (SIGGRAPH)* 29, 4, Article 115 (July 2010).

J Wu and Chang Shu. 2009. Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications. *J. Comput. Phys.* 228, 6 (2009), 1963–1979.

J Wu and Chang Shu. 2010. An improved immersed boundary-lattice Boltzmann method for simulating three-dimensional incompressible flows. *J. Comput. Phys.* 229, 13 (2010), 5022–5042.

You Xie, Erik Franz, Mengyu Chu, and Nils Thürey. 2018. tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Trans. Graph. (SIGGRAPH ASIA)* 37, 4 (2018), 95.

Hui Xu and Pierre Sagaut. 2011. Optimal low-dispersion low-dissipation LBM schemes for computational aeroacoustics. *J. Comput. Phys.* 230, 13 (2011), 5353–5382.

Cem Yuksel. 2015. Sample elimination for generating Poisson disk sample sets. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 25–32.

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An Advection-reflection Solver for Detail-preserving Fluid Simulation. *ACM Trans. Graph.* 37, 4, Article 85 (2018).

Xinxin Zhang and Robert Bridson. 2014. A PPPM Fast Summation Method for Fluids and Beyond. *ACM Trans. Graph. (SIGGRAPH ASIA)* 33, 6, Article 206 (2014).

Xinxin Zhang, Robert Bridson, and Chen Greif. 2015. Restoring the Missing Vorticity in Advection-projection Fluid Solvers. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 52 (2015).

Xinxin Zhang, Minchen Li, and Robert Bridson. 2016. Resolving Fluid Boundary Layers with Particle Strength Exchange and Weak Adaptivity. *ACM Trans. Graph. (SIGGRAPH)* 35, 4, Article 76 (July 2016).

Bo Zhu, Wenlong Lu, Matthew Cong, Byungmoon Kim, and Ronald Fedkiw. 2013. A New Grid Structure for Domain Extension. *ACM Trans. Graph.* 32, 4 (2013), Art. 63.

Yongning Zhu and Robert Bridson. 2005. Animating Sand As a Fluid. *ACM Trans. Graph.* (2005), 965–972.