Fast and versatile fluid-solid coupling for turbulent flow simulation

Chaoyang Lyu, Wei Li, Mathieu Desbrun, Xiaopei Liu

Fast and Versatile Fluid-Solid Coupling for Turbulent Flow Simulation

CHAOYANG LYU, ShanghaiTech University / SIMIT / UCAS
WEI LI, ShanghaiTech University / Inria
MATHIEU DESBRUN, Inria / Ecole Polytechnique / Caltech
XIAOPEI LIU, ShanghaiTech University

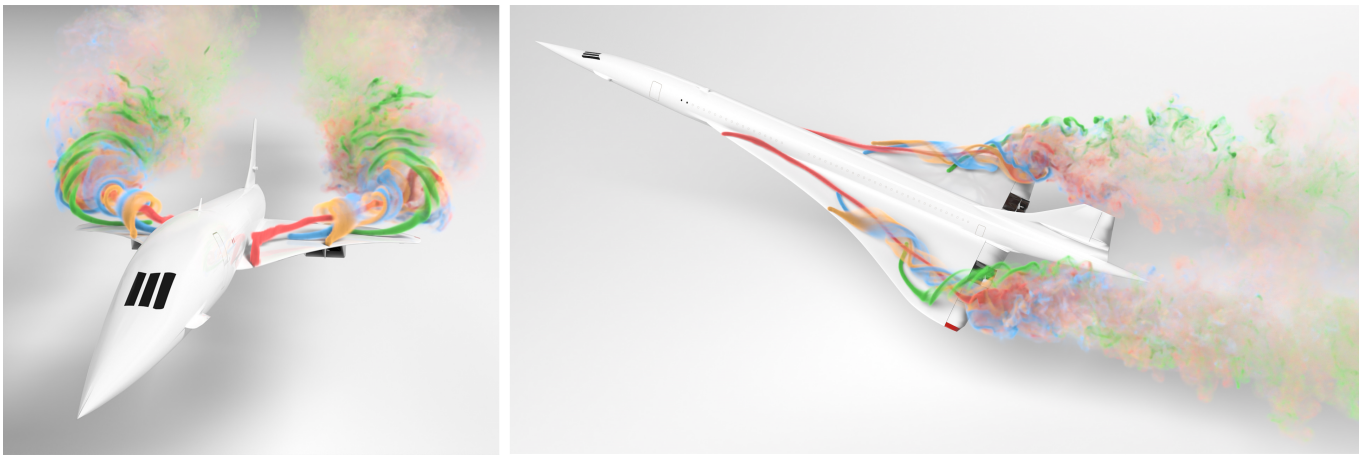


Fig. 1. **Fast, general, and robust fluid-solid coupling using an LBM-based solver.** We introduce an efficient and scalable approach to handling coupling in lattice Boltzmann-based fluid simulation. Our approach can handle arbitrary immersed objects (here, a Concorde airplane) with thick and/or thin structures, and generate the type of expected complex swirling vortex distributions seen in real life, while maintaining the massively parallel nature of LBM simulations. This 800×200×300 example took only 12 minutes for one second of animation using a single NVidia RTX 3090 GPU and 11 GB of GPU memory.

The intricate motions and complex vortical structures generated by the interaction between fluids and solids are visually fascinating. However, reproducing such a two-way coupling between thin objects and turbulent fluids numerically is notoriously challenging and computationally costly: existing approaches such as cut-cell or immersed-boundary methods have difficulty achieving physical accuracy, or even visual plausibility, of simulations involving fast-evolving flows with immersed objects of arbitrary shapes. In this paper, we propose an efficient and versatile approach for simulating two-way fluid-solid coupling within the kinetic (lattice-Boltzmann) fluid simulation framework, valid for both laminar and highly turbulent flows, and for both thick and thin objects. We introduce a novel hybrid approach to fluid-solid coupling which systematically involves a mesoscopic double-sided bounce-back scheme followed by a cut-cell velocity correction for a more robust and plausible treatment of turbulent flows near moving (thin) solids, preventing flow penetration and reducing boundary artifacts significantly. Coupled with an efficient approximation to simplify geometric computations, the whole boundary treatment method preserves the inherent massively parallel computational nature of the kinetic method. Moreover, we propose simple GPU optimizations of the core LBM algorithm which achieve an even higher computational efficiency than the state-of-the-art kinetic fluid solvers in graphics. We demonstrate the accuracy and efficacy of our two-way coupling through various challenging simulations involving a variety of rigid body solids and fluids at both high and low Reynolds numbers. Finally, comparisons to existing methods on benchmark data and real experiments further highlight the superiority of our method.

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

Additional Key Words and Phrases: Fluid-Solid Coupling, Turbulent Flow Simulation, Lattice Boltzmann Method

ACM Reference Format:
Chaoyang Lyu, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2021. Fast and Versatile Fluid-Solid Coupling for Turbulent Flow Simulation. *ACM Trans. Graph.* 40, 6, Article 201 (December 2021), 18 pages. https://doi.org/10.1145/3478513.3480493

1 INTRODUCTION

Fluid-solid coupling is responsible for a wide range of visually intricate phenomena: the presence of a surrounding flow has a strong impact on the motion of a solid structure, which in turn, often affects the behavior of the flow itself. This type of mutual interaction is complex to solve numerically but important in many applications. Whether it be for rapid product design, visual rendition in movie productions, or even robot training in a virtual environment, capturing the complex behavior and rich physical details of two-way coupling is of crucial importance. Yet, an efficient two-way coupling

Authors' addresses: C. Lyu, X. Liu: School of Information Science and Technology (Shanghai Engineering Research Center of Intelligent Vision and Imaging) of ShanghaiTech University, Shanghai, China; C. Lyu is also affiliated with the Shanghai Institute of Microsystem and Information Technology (SIMIT) and the University of the Chinese Academy of Sciences (UCAS); W. Li, M. Desbrun: Inria Saclay, LIX/DIX, Institut Polytechnique de Paris, Palaiseau, France; M. Desbrun is on leave from Caltech.

0730-0301/2021/12-ART201 $15.00
https://doi.org/10.1145/3478513.3480493

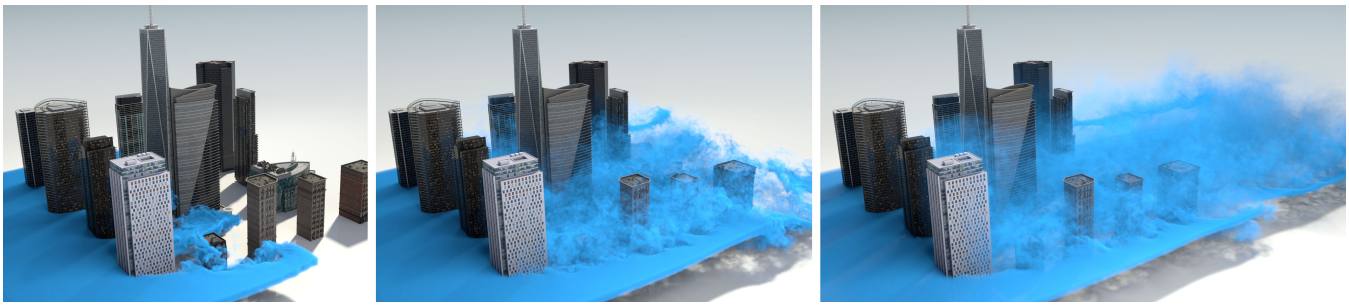


Fig. 2. **High-resolution simulation of airflow through a city neighborhood**. We simulate the airflow passing around and over buildings using a high resolution grid (889 × 333 × 556), where the buildings contain both thin and sharp structures. A layer of smoke particles with relatively small thickness is coming from the left, creating fine vortical structures behind and in between the buildings.

approach to simulating dynamic shells (i.e., solid objects with high width-to-thickness ratio) or rods (high length-to-cross-section ratio) in a turbulent flow that degrades gracefully in the presence of obstacles it cannot fully resolve remains challenging.

Recent approaches have demonstrated efficient and realistic simulations of shells [Grinspun et al. 2003; Bridson et al. 2003] or rods [Bergou et al. 2008] in *viscous fluids* [Fei et al. 2018a; Takahashi and Lin 2019; Fei et al. 2019], for which momentum transfer can be carried out both stably and efficiently using reduced models, as well as in *non-turbulent flows* through cut cells [Azevedo et al. 2016]. Progress in kinetic approaches have even led to stable and much more efficient treatment of coupling in *turbulent flows* via a diffuse-interface immersed boundary method [Li et al. 2019, 2020]; alas, non-volumetric structures such as thin obstacles are not supported, greatly limiting the variety of simulation scenarios in practice.

In this paper, we introduce an efficient and versatile approach for solving two-way fluid-solid coupling with complex solid shapes (involving both thick or thin structures) within the framework of kinetic fluid simulation. Our algorithm improves on the robustness and visual plausibility of the bounce-back scheme [Ladd 1994] through the addition of a new velocity correction, resolving the typical shortcomings of the sharp-interface immersed boundary method [Mittal et al. 2008] used in kinetic fluid simulation. The resulting hybrid coupling approach is local and conservative, preserves the highly parallelizable nature of kinetic fluid simulation containing all types of solids (arbitrary-shaped volumes, shells, and rods), and further improves boundary treatment of dynamic solids without having recourse to time rescaling [Li et al. 2020]. In addition, due to new geometric approximations and implementation-level GPU optimizations, our two-way coupling runs even faster than the state-of-the-art kinetic fluid simulation method of Chen *et al.* [2021] — typically 4 times faster at a normal grid resolution using recent GPU architectures. Fig. 1 shows a detailed example of airflow over a Concorde airplane obtained by our approach, for which one second of animation only took 12 minutes to compute.

1.1 Related Work

Numerically capturing the interplay between fluid and solid has been achieved in a variety of ways, depending mostly on the underlying fluid solver. Given the vast literature of fluid simulation in both computer graphics (CG) and computational fluid dynamics (CFD), we concentrate our review of previous works on closely-related approaches as a means to motivate our contributions.

Navier-Stokes-based Solvers. Fluid simulation in graphics is often based on Eulerian solvers using a regular grid to integrate the Navier-Stokes (NS) equations in time. Early work in coupling consisted in using the Lagrangian solid velocity as a Neumann boundary condition for the fluid and integrating the pressure force on the surface of the solid [Yngve et al. 2000; Foster and Fedkiw 2001; Carlson et al. 2004; Takahashi et al. 2002; Génevaux et al. 2003]. While these approaches were based on pressure solves treating fluid-solid coupling either explicitly or semi-implicitly, fully implicit formulation linking fluid and solid velocities were proposed [Klingner et al. 2006; Chentanez et al. 2006; Batty et al. 2007] as well. A fully Eulerian treatment of fluid-solid coupling was also formulated [Teng et al. 2016]. More recently, monolithic methods [Takahashi and Batty 2020; Fang et al. 2020] were proposed to support strong two-way coupling, but they are limited to relatively small-scale scenarios due to their computational complexity. Dealing with thin shell coupling proved to be more difficult: a number of modifications are required, including a one-sided treatment based on ray-casting to prevent leakage of information across the solids, as well as a second pressure solve to calculate resulting forces on solids [Guendelman et al. 2005]. Improvements involving more accurate momentum transfer and ghost cells were later added [Robinson-Mosher et al. 2008, 2009]. Eulerian solvers based on boundary-conforming meshes [Feldman et al. 2005b,a; Dai and Schmidt 2005; Klingner et al. 2006; Elcott et al. 2007] or even Lagrangian methods [Misztal et al. 2010; Clausen et al. 2013] can theoretically handle any solid, but this versatility comes at the cost of frequent and costly remeshing. In order to reduce computational complexity, cut-cell-based approaches employ finite-volume-like formulations inside sub-cell boundary-conforming regions to prevent fluid flowing into objects even if they are thin shells [Roble et al. 2005; Batty et al. 2007; Ng et al. 2009; Gibou and Min 2012; Weber et al. 2015; Edwards and Bridson 2014; Liu et al. 2015; Azevedo et al. 2016], often requiring clipping against the potentially complex geometry of immersed objects. For particle-based fluid solvers, penalty forces have often been used to enforce boundary conditions [Peer et al. 2015; Ihmsen et al. 2013]. Gao *et al.* [2017] integrated FLIP and shape matching constraints, but suffer from stability issues when applied to turbulent flows. Moreover, most particle methods do not ensure a consistent pressure in the fluid and on solids [Band et al. 2018]. Finally, hybrid particle-grid methods

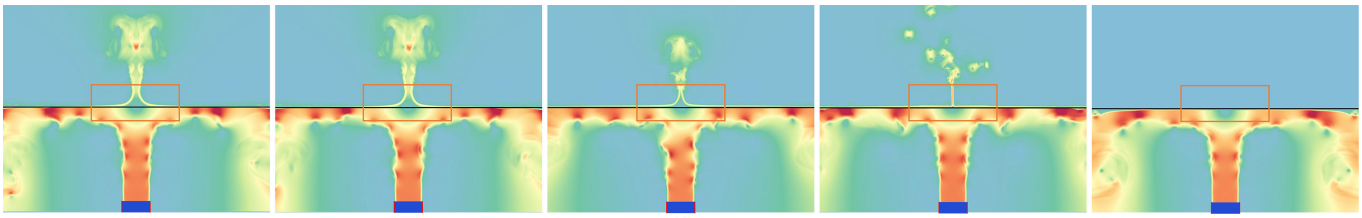


Fig. 3. **Fluid leakage.** If a thin shell separates a 2D domain into two chambers, a jet in the lower chamber is erroneously leaking into the upper chamber (see the boxed regions) for (a) the diffuse-interface immersed boundary method [Patel and Natarajan 2018] , (b) the iterative diffuse-interface immersed boundary method [Zhang et al. 2016a], (c) the sharp-interface immersed boundary method [Seo and Mittal 2011], and (d) the distribution function correction-based immersed boundary method [Tao et al. 2018]; in contrast, our approach (e) avoids fluid leakage perfectly.

often use grid-based techniques to handle coupling [Zhang et al. 2016b; Fei et al. 2018a; Hu et al. 2018; Fei et al. 2019].

Boltzmann-based Solvers. In computational science and to a lesser extent in graphics, the lattice Boltzmann method (LBM) has emerged as an efficient and accurate alternative to NS solvers. Many so-called "kinetic solvers" are now available, providing a numerical approximation of the Boltzmann transport equation satisfied by a statistical particle distribution in the fluid domain. A correct enforcement of boundary conditions is crucial for accurate and stable kinetic simulation of fluid flows; however, the *mesoscopic* nature of kinetic solvers implies that boundary conditions must be imposed on the particle distribution functions themselves rather than on fluid macroscopic quantities, thus requiring very different numerical treatments compared to NS solvers. As kinetic methods do not need pressure projections, fluid-solid coupling can be quite easily included through a simple bounce-back treatment [Ladd 1994; Mei et al. 1999], which reverses the distribution function advection against the boundary and applies a momentum exchange based on the velocity of the solid encountered. However, this streaming alteration assumes that the interface is always half-way between adjacent nodes, so a number of strategies were proposed over the years to reduce the numerical artifacts generated by this staircase approximation [Lallemand and Luo 2003; Kao and Yang 2008; Yin and Zhang 2012], often at the cost of larger stencils, reduced stability, or even mass loss. Moreover, while the original bounce-back treatment was applied to fluid and solid nodes alike (thus the solid contained a fictitious "interior fluid" for computational convenience), various authors proposed to apply bounce-back only to fluid nodes instead [Aidun et al. 1998; Geller et al. 2006]. However, new distribution functions need to be assigned for "fresh" grid nodes (i.e., fluid nodes that were in the solid at the previous time step) in the case of dynamic solids, and the choice of a "refilling" interpolation scheme used to derive fresh values from nearby nodes (similar in spirit to the Gauss-Jacobi process for *invalid* nodes advocated in [Guendelman et al. 2005] in the context of Eulerian Navier-Stokes fluid solvers) greatly influences the simulation accuracy around solid boundaries [Peng et al. 2016]. Interpolated bounce-back schemes [Bouzidi et al. 2001; Kao and Yang 2008] propose a higher-order treatment, but result in jumps in the hydrodynamic forces when a solid object moves from one grid node to another [Krüger et al. 2017]. Partially saturated bounce-back (PSBB) schemes, also known as continuous bounce-back schemes, are convenient for porous media [Dardis and McCloskey 1998; Walsh et al. 2009] and suspension flows [Noble and Torczynski 1998; Chen et al. 2013], treating a lattice node as a pure fluid node, a pure solid node, or a mixed node. Even if PSBB scheme is mass-conserving and does not require dealing with fresh fluid nodes, it needs the computation of solid/fluid fractions in cells intersecting the boundaries, which effectively limits its use to stationary geometries or spherical particles [Krüger et al. 2017]. To improve stability, artificially-increased viscosity can be added to remove spurious frequencies and reduce discretization artifacts [Olson 2015], at the cost of smearing turbulent vortices near solid boundaries. In sum, the bounce-back scheme is particularly attractive because it preserves the massively parallel nature of streaming and ensures conservative momentum exchange between fluid and solid, but its accuracy and robustness is limited: its staircase approximation induces velocity fluctuations around solid boundaries, producing numerical and visual artifacts especially on coarse grids, effectively preventing stable simulation of two-way coupling in a turbulent flow. More accurate simulations can only be achieved through high-order schemes or grid refinements near solid boundaries, which are both notably computationally expensive for moving and/or deformable solids immersed in the fluid.

Immersed Boundary Method. In addition to the usual boundary treatment methods in NS and kinetic solvers, the immersed boundary (IB) method [Peskin 1972; Griffith and Patankar 2020] is a general, yet conceptually simple approach based on penalty forces that is applicable to both types of solvers. Due to its local nature and easy geometric treatment, it is widely used with LBM in CFD [Lu et al. 2012; Kang and Hassan 2011; Mittal et al. 2008] as well as in graphics [Li et al. 2020] to simulate both one-way and two-way fluid-solid coupling in a variety of scenarios. There are usually two IB variants: the diffuse-interface (DI-IB, [Lu et al. 2012; Kang and Hassan 2011; Patel and Natarajan 2018]) vs. the sharp-interface (SI-IB [Seo and Mittal 2011; Mittal et al. 2008; Jiang et al. 2018]) IB methods. Traditional DI-IB method consists of velocity interpolation and penalty force spreading with the same kernel to satisfy the no-slip boundary condition [Peskin 1972; Li et al. 2016], while more recent variants include an iterative [Kang and Hassan 2011] and an implicit [Cai et al. 2018] version for improved boundary accuracy. SI-IB methods are different from DI-IB in that they extrapolate velocities of solid nodes along the surface normal directions, requiring accurate geometric computation [Teng et al. 2016] and thus extra computational overhead. They also suffer from the traditional error-prone interpolation of fresh cells, and thus cannot handle moving solids in a turbulent flow. A few IB-based methods specifically designed for kinetic fluid simulation framework exist: [Tao et al. 2019] proposed

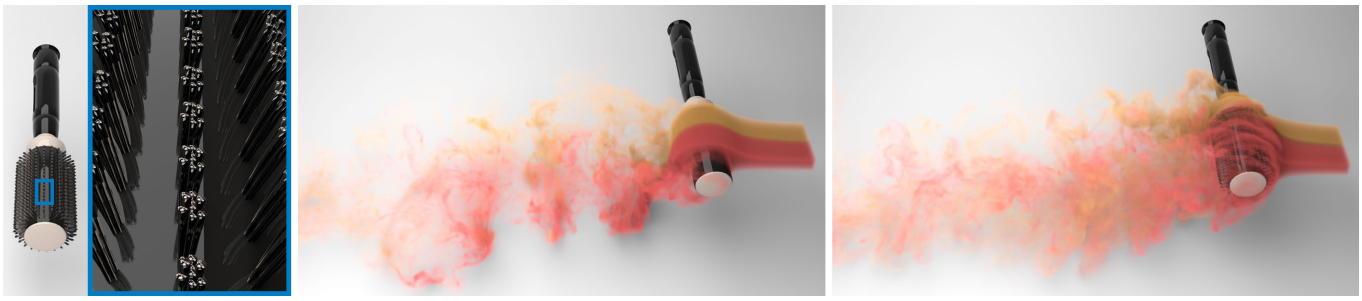


Fig. 4. **Hair brush**. The translating and rotating motion of a hair brush containing hundreds of bristles (left) creates fine vortices in its wake as well as around the bristles (right) as evidenced by the evolution of the smoke passing around it, properly capturing the intricate fluid-solid interaction engendered by this complex geometry. Compared to the coupling with bristles (right), the smoke near the hair brush without bristles (middle) is much smoother, and the wake flows contain relatively large vortices, indicating the efficacy of subgrid approximation in handling such complex solid shapes.

a non-iterative approach correcting the neighboring distribution functions directly; a combination of IB and bounce-back was also proposed [Wang et al. 2020], where distribution functions are first spread to samples, then streamed to Eulerian grid nodes. All these IB methods share the same basic limitation, however: *thin geometrical structures such as shells and rods are notoriously difficult to handle*: fluid velocities often propagate *through* solid boundaries, causing fluid leakage in thin structures, as demonstrated in Fig. 3.

1.2 Overview and Contributions

Based on our review of previous work, it is clear that a fast, robust, yet efficient coupling approach to handle both thin and thick obstacles inside fluid is needed. In this paper, we propose such an approach for LBM solvers. By combining bounce-back and one-sided velocity correction in cells intersecting solid boundaries through simple and efficient geometric approximations, we overcome the weaknesses of existing approaches while maintaining the highly parallel nature of LBM. Our novel boundary treatment avoids leakage, offers improved robustness and accuracy, and is versatile enough to handle two-way coupling with moving boundaries of thin or thick immersed objects.

2 BACKGROUND

Before introducing our approach to dynamic fluid-solid coupling, we first briefly review the kinetic fluid simulation framework that our work will build upon.

2.1 Kinetic Foundation

Fluid solvers aim to provide numerical solutions to physical equations that model the physics of real fluids. While NS equations describing the change in time of a fluid's velocity field have been at the root of most approaches in graphics, kinetic models rely instead on the time evolution of a distribution function $f(\boldsymbol{v}; \boldsymbol{x}, t)$ encoding the probability of a particle having a microscopic velocity $\boldsymbol{v}$ at position $\boldsymbol{x}$ and time t. In this representation, fluid flow is governed by the Boltzmann equation [Shan et al. 2006]:

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla f = \Omega(f) + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} f \,, \tag{1}$$

where ∇ and $\nabla_{\boldsymbol{v}}$ are the gradient operators w.r.t position $\boldsymbol{x}$ and particle velocity $\boldsymbol{v}$, respectively; Ω is the collision operator affecting the distribution function due to particle collisions; and $\boldsymbol{F}$ is the external force field per unit volume (e.g., gravity force). Macroscopic quantities at a position $\boldsymbol{x}$ and a time t can be recovered through the moments of f w.r.t particle velocity $\boldsymbol{v}$: the fluid density ρ, the momentum $\rho\boldsymbol{u}$, and the momentum flux $\boldsymbol{\Pi} = \rho\boldsymbol{u}\otimes\boldsymbol{u} + p\boldsymbol{I} - \boldsymbol{\sigma}$ (where $\boldsymbol{\sigma}$ denotes the shear stress tensor) correspond respectively to the zero-th, first, and second order moments, i.e.,

$$\rho = \int f \, \mathrm{d}\boldsymbol{v}, \quad \rho\boldsymbol{u} = \int \boldsymbol{v} \, f \, \mathrm{d}\boldsymbol{v}, \quad \text{and } \boldsymbol{\Pi} = \int \boldsymbol{v}\otimes\boldsymbol{v} \, f \, \mathrm{d}\boldsymbol{v} \,, \tag{2}$$

while the pressure p is the local variance w.r.t $\boldsymbol{v}$, i.e.,

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

Using the continuous Bhatnagar-Gross-Krook (BGK) collision model [Chen and Doolen 1998], the zeroth and first order moments of Eq. (1) lead to, respectively, the well-known continuity equation and Navier-Stokes equation used in conventional fluid simulation.

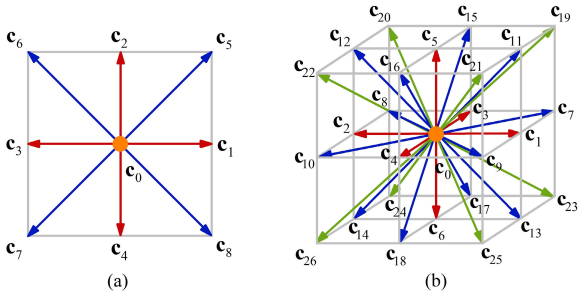


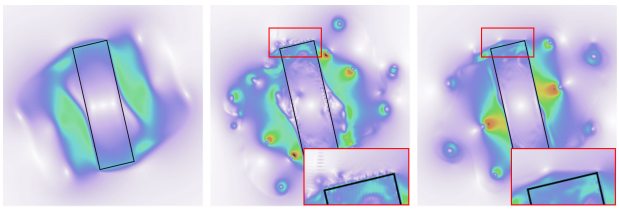
Fig. 5. **Lattice structures.** We employ D2Q9 lattice structures in 2D (a) and D3Q27 in 3D (b) as in [Li et al. 2020], where each $\mathbf{c}_i$ is the discretized particle velocities. A discrete distribution function f_i is stored for each $\mathbf{c}_i$.

2.2 Lattice Boltzmann Method

In LBM, the Boltzmann equation (1) is numerically integrated by first discretizing a) time with a constant step size, b) space through a uniform grid, and then c) the particle velocity field $\boldsymbol{v}$ through a set of q discrete velocities $\{\boldsymbol{c}_i\}_{i=0..q-1}$ per spatial grid node, thus forming a lattice (see Fig. 5). After discretization, a set of lattice Boltzmann equations (LBE) in normalized units [Li et al. 2020] are updated in time at each grid node $\boldsymbol{x}$ as:

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i, t+1) - f_i(\boldsymbol{x}, t) = \Omega_i + G_i \,, \tag{4}$$

where $f_i(\boldsymbol{x}, t) \propto f(\boldsymbol{c}_i; \boldsymbol{x}, t)$ are the discretized distribution functions; Ω_i are the *discretized* collision operators; and G_i are projections of



(a) (b) (c)

Fig. 6. **Different boundary treatments.** (a) a plain bounce-back scheme generates no visual artifacts for flows with low Reynolds number; (b) however, it exhibits strong aliasing artifacts near the solid boundary for high Reynolds number flows, which may seriously influence stability; (c) our new boundary treatment for the same high Reynolds number as in (b) is artifact-free, generating the type of vortices expected from such an example.

the external force $\boldsymbol{F}$ onto the discrete space of distribution functions. The discretization of particle velocity typically contains $q=9$ symmetric discrete velocities in 2D (forming a D2Q9 lattice structure) and $q = 27$ in 3D (D3Q27 lattice structure) as displayed in Fig. 5. Solving Eq. (4) is then achieved through standard splitting, where a *streaming process* first takes care of advection by assigning $f_i(\boldsymbol{x} - \boldsymbol{c}_i, t)$ to intermediate distribution function values $f_i^*(\boldsymbol{x}, t)$ as:

$$f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x} - \boldsymbol{c}_i, t) \,, \tag{5}$$

which amounts to an upwinding advection, followed by a *relaxation process* taking care of collision Ω_i and external force G_i via

$$f_i(\boldsymbol{x}, t + 1) = f_i^*(\boldsymbol{x}, t) + \Omega_i + G_i \,, \tag{6}$$

which is calculated independently at each grid node. Macroscopic quantities are calculated by discretizing Eq. 2 [Guo et al. 2002]:

$$\rho = \sum_{i=0}^{q-1} f_i, \qquad \rho\boldsymbol{u} = \sum_{i=0}^{q-1} \boldsymbol{c}_i f_i + \frac{1}{2}\boldsymbol{F}. \tag{7}$$

Note that boundary conditions are usually applied right after Eq. (5). The locality of these simple explicit updates makes the integration of LBE naturally embarrassingly-parallel, thus offering high computational efficiency.

Key to simulating both laminar and turbulent flows is the evaluation of the collision terms Ω_i, for which the non-orthogonal central-moment multiple-relaxation-time model [De Rosis and Luo 2019] has been proven a particularly accurate and stable approach. As its name indicates, it resorts to central moments to model the relaxation process and separate the relaxation rates of the conserved and non-conserved moments. We refer the reader to [Li et al. 2020] for more details on this collision model implementation, as we adopt their core LBM solver in our work.

2.3 Fluid-Solid Coupling

When a solid (be it held fixed or moving freely) is immersed in a fluid, one must enforce proper boundary conditions between fluid and solid — typically, no-slip or full-slip conditions. As briefly reviewed in Sec. 1.1, there are two types of methods in the LBM literature to enforce boundary conditions: bounce-back schemes and immersed boundary methods, both with their pros and cons. Here we dig deeper in the details of the most relevant methods in order to motivate our new hybrid coupling approach.

Bounce-back scheme. The basic streaming process from Eq. (5) is oblivious to the presence of obstacles in the fluid. Thus, it must be adapted when a solid is nearby: if a node $\boldsymbol{x}_b$ in the fluid is such that there exists at least an index $0 < j \leq q - 1$ for which a neighboring node $\boldsymbol{x}_b - \boldsymbol{c}_j$ is located inside a solid, then the distribution function(s) $f_j(\boldsymbol{x})$ cannot be updated through regular upwinding (see the distribution functions with orange color in Fig. 7) since the required node value is not even available. Instead, streaming must be altered to account for a "bounce-back" of the fluid against the solid boundary. As discussed in Sec. 1.1, the basic bounce-back scheme for no-slip conditions performs a direction reversal of streaming followed by a momentum exchange [Ladd 1994], so that the unknown distribution functions are obtained via:

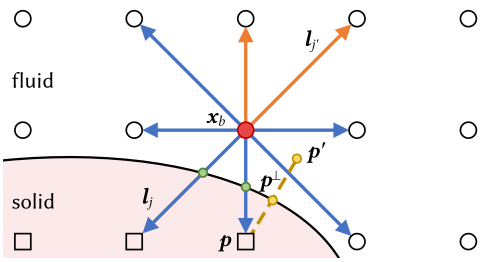


Fig. 7. **Bounce-back scheme near fluid-solid boundary.** During streaming, some of the distribution functions $f_{j'}$ (corresponding to the orange links) at the fluid node $\boldsymbol{x}_b$ (red circle) near a solid boundary are unknown, since the node at $\boldsymbol{x}_b - \boldsymbol{c}_{j'}$ is located inside the solid region (shown as boxes), requiring a bounce-back scheme using the velocity $\boldsymbol{u}_s$ of the boundary velocity at the intersection point (green circles) between the line segment from $\boldsymbol{x}_b$ to $\boldsymbol{x}_b - \boldsymbol{c}_{j'}$ and the solid boundary.

$$f_j(\boldsymbol{x}, t + 1) = f_{j'}(\boldsymbol{x}, t) - 6w_j\rho\boldsymbol{u}_s \cdot \boldsymbol{c}_j \,, \tag{8}$$

where j' indicates the index of the reversed velocity satisfying $\boldsymbol{c}_{j'} = -\boldsymbol{c}_j$ (see Fig. 7); w_j are the usual lattice weights used in the lattice Boltzmann equations (see [Li et al. 2020]), and $\boldsymbol{u}_s$ is the velocity of the solid at the intersection between the fluid-solid boundary and the line segment from $\boldsymbol{x}$ to $\boldsymbol{x}-\boldsymbol{c}_j$ (also called the "link", denoted as $\boldsymbol{l}_j$). The opposite node $\boldsymbol{x} - \boldsymbol{c}_j$ is treated symmetrically to ensure proper mass conservation, while the corresponding momentum transfer is communicated to the solid to enforce momentum conservation. The case of free-slip condition is treated similarly, with this time a mirrored direction for the streaming process with respect to the local boundary normal, see for instance [Thürey 2007]. However, the simplicity of the original bounce-back scheme inherently carries limitations: undesirable large velocity gradients can be formed near solid boundaries, creating dispersion errors that eventually damage the simulation for flows with high Reynolds numbers — thus preventing fluid-solid coupling in strong turbulent flows. Fig. 6 shows a 2D illustration of an erroneous velocity field near the solid boundary generated by bounce-back for a rotating rectangle inside a fluid with a small viscosity (b), compared to a large viscosity (a) which happens to dampen (and thus hide) the numerical issues.

Immersed boundary method. Immersed boundary (IB) methods are a common alternative to bounce-back schemes, mostly using penalty forces to enforce boundary conditions [Patel and Natarajan 2018; Mittal et al. 2008; Li et al. 2020]. Among various IB formulations, the sharp-interface IB (SI-IB) method [Mittal et al. 2008] gives the most accurate results. Given a fluid node with at least one link intersecting the solid boundary, the SI-IB method first reconstructs

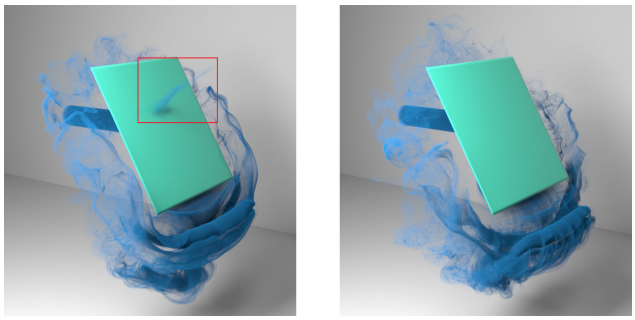


Fig. 8. **Leakage of Immersed Boundary.** For a kinetic fluid simulation coupled with a thin plate, the immersed boundary method used in [Li et al. 2020] (left) is compared with our method (right). Due to force spreading in both sides of the thin plate, this recent coupling approach employing a diffuse-interface immersed boundary method may generate leakage through the plate (see red box), while our method prevents this issue by design.

the distribution functions of the solid node (shown as boxes in Fig. 7) by first projecting the node onto the solid surface (node $\boldsymbol{p}^{\perp}$ in Fig. 7). Then, a mirrored node $\boldsymbol{p}'$ is found along the normal direction $\boldsymbol{p}^{\perp}-\boldsymbol{p}$ whose macroscopic velocity $\boldsymbol{u}(\boldsymbol{p}')$ is then interpolated from nearby nodes along one side of the solid boundary. Since the solid velocity $\boldsymbol{u}_b(\boldsymbol{p}^{\perp})$ at the intersected point is already known, we can easily compute the new macroscopic velocity at node $\boldsymbol{p}$ as $\boldsymbol{u}^*(\boldsymbol{p}) = 2\boldsymbol{u}_b(\boldsymbol{p}^{\perp}) - \boldsymbol{u}(\boldsymbol{p}')$. From the existing macroscopic velocity at the same node $\boldsymbol{u}(\boldsymbol{p})$, a velocity difference $\delta\boldsymbol{u}(\boldsymbol{p}) = \boldsymbol{u}^*(\boldsymbol{p}) - \boldsymbol{u}(\boldsymbol{p})$ is finally used to derive a penalty force $\boldsymbol{F}(\boldsymbol{p}) = \rho\delta\boldsymbol{u}(\boldsymbol{p})$ to correct the fluid velocity at node $\boldsymbol{p}$. The advantage of the SI-IB method is that undesirable large velocity gradients are suppressed along the solid boundary due to the use of interpolation. It also naturally supports curved solid boundaries at sub-grid scale. However, as argued in Sec. 1.1, since the method still requires identification of grid nodes inside and outside the solid, it is *inappropriate for solids with a thickness smaller than a grid cell size* since it incurs spurious velocities on the other side of the solid, an issue called fluid leakage and visible in Fig. 8 (note that the same issue also occurs for DI-IB methods, see Fig. 3). In addition, handling moving solid with the SI-IB method still requires node refilling, introducing additional errors due to the ill-defined notion of velocity extrapolation.

2.4 Discussion

Despite significant improvements over the past decade, the numerical treatment of fluid-solid coupling in kinetic solvers remains unsatisfactory. Typically, coupling in LBM necessitates a certain thickness of (and separation between) immersed solids to avoid fluid leakage, preventing its use on thin shells or rods or even objects smaller than the grid spacing. Moreover, coupling via bounce-back schemes generates artifacts in the form of spurious velocity fluctuations and vorticity generation near boundaries, whereas the immersed boundary method is computationally more intensive since the stiffness engendered by the coupling penalty forces calls for smaller time steps. While fluid-solid coupling for NS solvers has a number of viable solutions by now, kinetic methods cannot directly benefit from this success: first, the heavy use of pressure computations to drive coupling is unsuitable for kinetic solvers, which actually bypass pressure projection altogether; second, importing existing coupling strategies from NS solvers relying on intensive geometric intersection computations (such as the construction of cut-cells) would dramatically slow down the highly parallelizable nature of kinetic solvers. Yet, LBM is currently the only family of numerical methods in graphics which can very *efficiently handle dynamic two-way coupling in strong turbulence*. The development of a new, efficient, and LBM-compatible numerical approach that is capable of supporting fluid-solid coupling between turbulent flows and arbitrarily-shaped (thick or thin) structures is thus required.

3 HYBRID FLUID-SOLID COUPLING

We now delve into the details of our LBM-based hybrid coupling approach, describing first its motivations from the existing methods before reviewing each component.

3.1 Rationale

Since LBM uses a regular grid to discretize space, fluid-solid interactions are not trivial to enforce correctly: given a solid object immersed in a fluid, its boundary usually does *not* align with the fluid grid, resulting in the existence of "*cut-cells*", i.e., *grid cells containing a solid boundary surface*, thus only partially filled with fluid — see the green cell in Fig. 9 (left) for a 2D example. A kinetic solver must handle these cut-cells differently in order for coupling and boundary conditions to be accounted for. While no existing fluid-solid boundary treatment can boast about being efficient, accurate, and stable enough to properly handle coupling, each family of coupling methods has its own advantages — and limitations. The diffuse-interface immersed boundary (DI-IB) method, for instance, bypasses the need to track fresh/non-fresh cells that many bounce-back-based techniques necessitate, and thus maintains its high parallelism, but it typically induces stiff penalty forces that require time rescaling to ensure stability, and fails on thin structures as force spreading is done on both sides of the thin solid. On the other hand, bounce-back schemes do prevent flow penetration at the mesoscopic level, but their accuracy is limited due to their lattice-based nature, yielding non-smooth spurious velocity distributions around solid and inducing coupling instability in turbulent flow simulations.

Our hybrid approach stems from the fact that these two families of coupling strategies are, in essence, *complementary*. While they are inaccurate for handling thin solid structures, bounce-back schemes directly enforce non-penetration by bouncing distribution functions off solid boundaries, which, in effect, partially accounts for the penalty forces that the IB method would have generated instead. Thus, we can add ancillary penalty forces *after* the bounce-back scheme has been applied in order to further correct the velocity field near the solid boundary. Note that these ancillary forces will be *far weaker* than in usual IB methods as the bounce-back scheme has already applied an impulse to the fluid, thus not necessitating time rescaling (i.e., smaller time steps). *If we can modify the bounce-back scheme to support coupling with thin solids as well, a hybrid approach combining bounce-back and velocity correction would provide the desired robustness, efficiency, and accuracy needed to handle fluid-solid coupling in an LBM solver, without affecting its degree of parallelism.*

Our hybrid coupling approach implements this idea through three distinct computational stages added to a kinetic solver to handle dynamic fluid-solid interaction:

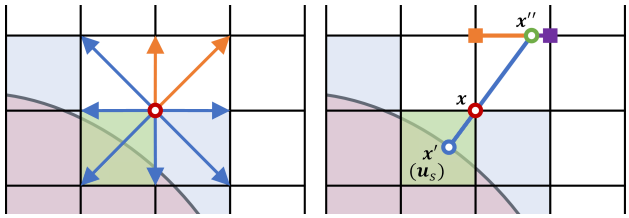


Fig. 9. **Interpolation of velocity on cut-cell nodes.** Left: cut-cell (marked in green) intersecting a solid boundary; Right: the velocity on a cut-cell node $\boldsymbol{x}$ inside the fluid region (red circle) needs to be interpolated using the velocities of its projected point $\boldsymbol{x}'$ onto the solid boundary (blue circle) and the intersected point between the ray from the projected point to the cut-cell node and the interpolated fluid point $\boldsymbol{x}''$ (green circle), where the velocity can be reliably evaluated through simple linear interpolation.

- *Double-sided bounce-back (DSBB)*: we introduce in Sec. 3.2 a variant of the original bounce-back scheme where the usual LBM streaming process is altered for nodes of cut-cells on *both* sides of the solid boundaries to prevent flow penetration and enforce a sided bounce against solid boundaries at the mesoscopic level.
- *Velocity correction in cut-cells*: in Sec. 3.3, we counteract the boundary artifacts that a bounce-back scheme can create through an ancillary velocity correction for cut-cell nodes derived from a one-sided interpolation using simpler geometric approximations to improve boundary subgrid treatment, thus drastically suppressing overshoots and increasing simulation stability.
- *Momentum transfer onto solids*: finally, we leverage a combination of momentum exchanges traditionally used at the mesoscopic level in bounce-back schemes and penalty forces during velocity correction to transfer to the solid all the forces exerted by the fluid present in cut-cells in order to drive solid motion.

Upon completion of our algorithm description, we will present tests and evaluations of our new hybrid boundary treatment method to validate its physical validity and improved accuracy to the typical coupling methods in the literature for LBM.

3.2 Double-Sided Bounce-Back Scheme

As reviewed in Sec. 2.3, the bounce-back scheme is often applied to nodes of cut-cells that are located in the fluid (see the red node in Fig. 9). However, fresh/non-fresh cells must be updated using interpolation/extrapolation when solid object moves, which usually causes strong dispersive velocities near solid boundaries for turbulent flows. Inspired by the DI-IB method, we apply bounce-back to *all* cut-cell nodes instead of restricting its use to fluid cut-cell nodes. Consequently, nearby solid nodes help the whole LBE solution to have the desired velocity transition when solid objects move. This simple change resembles the DI-IB method in spirit, where the penalty forces are replaced by the bounce-back scheme which naturally handles moving objects. To disambiguate with the original approach, we call this boundary treatment a *double-sided bounce-back* (DSBB) scheme as it is applied on both sides of the boundary. Used on its own, this scheme naturally prevents flow penetration through solid boundaries, *even for thin solids*, and no longer requires the handling of fresh/non-fresh cells. However, due to the staircase approximation of the boundary that the bounce-back approach implicitly assumes, sharp and spurious velocity gradients can arise, particularly in turbulent flows (see a 2D comparison in Fig. 6 between non-turbulent and turbulent flow simulations). Additionally, the momentum correction term of Ladd's scheme is only first-order accurate when the boundary is not at midpoints of the links [Aidun and Clausen 2010], sometimes leading to unexpected coupling behavior for complex boundaries. We thus introduce an additional velocity correction near solid boundaries next.

3.3 Velocity Correction in Cut-Cells

Since we seek a low-order (linear) accurate enforcement of boundary conditions, we propose to correct the velocity field near the boundaries after our double-sided bounce-back has been performed via ancillary penalty forces derived from linear interpolation of velocities in the direction normal to the boundary. More precisely, we evaluate expected velocities on cut-cell nodes based on nearby solid and fluid velocities, from which we simply apply a penalty force based on the difference between the current and expected values.

Expected velocities on cut-cell nodes. Expected velocities on cut-cell nodes are found through velocity interpolation between reliable velocities. For cut-cell nodes currently located inside a solid, their expected velocities have to match the exact solid velocity at the locations of these nodes, which can be directly computed from the solid's current linear and angular velocity. For cut-cell nodes located in the fluid, we derive a linear-accurate estimate of their expected velocities by interpolating along their normal directions as follows, as illustrated in 2D in Fig. 9 (right). Given a fluid node $\boldsymbol{x}$ of a cut-cell, we first compute its projected position $\boldsymbol{x}'$ onto the solid boundary surface. Then the ray starting at $\boldsymbol{x}'$ and going through $\boldsymbol{x}$ will continue on and hit the nearest axis-aligned cell face of the grid at a point $\boldsymbol{x}''$. If all the nodes on that nearest intersected cell face do not belong to any cut-cell, the fluid velocity at $\boldsymbol{x}''$ can be interpolated from the surrounding nodes of the cell face and their velocities through linear interpolation (linear in 2D and bilinear in 3D). Then, we can determine the "expected" (i.e., desirable) fluid velocity at $\boldsymbol{x}$ by another linear interpolation via:

$$\hat{\boldsymbol{u}}(\boldsymbol{x}) = (1-\alpha)\boldsymbol{u}_s + \alpha \boldsymbol{u}(\boldsymbol{x}'') \,, \tag{9}$$

where $\alpha = \|\boldsymbol{x}-\boldsymbol{x}'\|/\|\boldsymbol{x}''-\boldsymbol{x}'\|$, and $\boldsymbol{u}_s$ is the solid velocity at the projected point $\boldsymbol{x}'$ on the boundary.

Velocity correction through penalty forces. From an expected velocity $\hat{\boldsymbol{u}}(\boldsymbol{x})$ at a cut-cell node $\boldsymbol{x}$, which may differ from its current post-bounce velocity $\boldsymbol{u}(\boldsymbol{x})$, we derive a penalty force $\boldsymbol{F}_p(\boldsymbol{x})$ as

$$\boldsymbol{F}_p(\boldsymbol{x}) = \hat{\boldsymbol{u}}(\boldsymbol{x}) - \boldsymbol{u}(\boldsymbol{x}) \,, \tag{10}$$

which we apply as an external force to node $\boldsymbol{x}$. As done in the LBM solver of [Li et al. 2020], we first project the penalty force onto the space of discrete distribution functions using the highest-order Hermite polynomial expansion in order to retain accuracy and stability. Note that compared to the traditional IB methods, these ancillary penalty forces are far weaker: our double-sided bounce-back scheme applied earlier already accounted for a large part of the boundary condition enforcement, so the forces are just a further adjustment of the velocity. Thus, one does not need to reduce the physical time step via time rescaling, and a faster coupling simulation is achieved than with the traditional IB approach used in [Li et al. 2020].

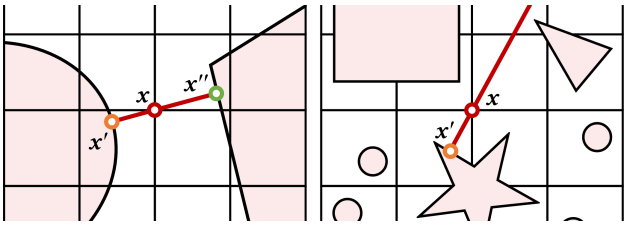


Fig. 10. **Two cases of solid proximity.** Left: the ray starting from a boundary point $\boldsymbol{x}'$ through a grid node $\boldsymbol{x}$ may hit another boundary surface point before it intersects with the non-cut-cell face; in this case, $\boldsymbol{x}''$ locates at another boundary point. Right: a cut-cell node may be surrounded by all cut-cell faces, so a ray may not hit any non-cut-cell face nearby.

Intersection cases. When solids are close to each other in the fluid compared to the size of a grid cell, there are a few cases where the velocity at $\boldsymbol{x}''$ *cannot* be interpolated from nearby non-cut-cell nodes. Indeed, the ray from $\boldsymbol{x}'$ may hit the boundary of the same or another solid first (Fig. 10, left). In this case, $\boldsymbol{x}''$ is defined as the first intersection along the ray with a boundary, and the solid velocity at this point is used in lieu of the bilinearly/trilinearly interpolated value described above; the final linear interpolation stays unchanged. Another case happens when the nearby axis-aligned faces intersected by the ray are all parts of cut-cells because of the proximity of different solids in the region (Fig. 10, right). In this case, it becomes impossible to find a nearby reliable velocity with which to perform a linear interpolation, so we just discard the penalty force correction, and use only the approximation offered by the double-sided bounce-back process.

Sub-grid approximation. Finally, in the case of thin shells or even thin rods being immersed in the fluid, it is possible that one fluid cell contains *multiple* thin structures, especially when grid resolution is coarse relative to the spacing between structures (see Fig. 11) and cannot resolve the boundary condition accurately. As an efficient, yet reasonable approximation, we treat multiple thin structures inside a cut-cell as an immersed solid formed by the approximate bounding volumes they occupy. To this end, we just project each fluid cut-cell node to the nearest boundary point to compute the penalty force as described in Fig. 11. Note that this bounding volume approximation seamlessly handles stacked objects: while each object may have sub-grid thickness, the agglomerates act as a monolithic solid without the need for a specific treatment, see Fig. 13.

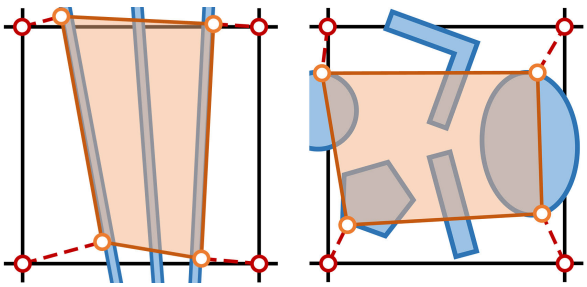


Fig. 11. **Sub-grid approximation.** To handle sub-grid scale solid structures, we project each cut-cell node onto its nearest boundary sample point. This basically amounts to using bounding volumes (orange regions) to approximate multiple thin structures within a cut-cell.

Discussion. Our penalty force correction is based on linear velocity interpolation along the normal direction, effectively acting as a filter around solid boundaries to suppress spurious velocity oscillations generated by the bounce-back scheme at high Reynolds numbers. This correction significantly enhances the stability of two-way coupling in turbulent flow simulations, leading to visually plausible results even for relatively coarse grids. However, since it is only a first-order linear correction, high accuracy is only retained for sufficiently fine grid resolutions. Given that we target graphical applications, we do not consider more advanced models for velocity correction near solid boundaries in this work.

3.4 Resulting Forces on Solids

Finally, we need to compute the momentum transfer from the surrounding fluid onto solids, which will be used to modify their motions for two-way coupling. Since we use a combination of bounce-back and velocity correction to derive the effects of the solids on the fluid, we must derive a new way to gather all the local fluid forces acting onto each solid in order to evaluate the resulting total force and torque exerted upon this solid. After our double-sided bounce-back treatment, we first accumulate all the momentum-exchange components from Eq. 8 used during the double-sided bounce-back treatment for each cut-cell node inside fluid, then deduce the resulting total force and torque on each solid. More precisely, the momentum of fluid after streaming at node $\boldsymbol{x}$ and discrete time t is expressed as: $\boldsymbol{p}(\boldsymbol{x}) = \sum_i \boldsymbol{c}_i f_i^*(\boldsymbol{x}, t)$. If L is the set of all the "cut-links" (i.e., links intersecting the solid boundary at $\boldsymbol{x}$), for each cut-link $\boldsymbol{l}_j \in L$ where the bounce-back treatment has been applied, the momentum transferred to a solid in direction $\boldsymbol{c}_j$ is $-\boldsymbol{c}_j\,(f_j^*(\boldsymbol{x}) + f_{j'}^*(\boldsymbol{x}))$. Therefore, the exchanged momentum $\Delta\boldsymbol{p}$ around $\boldsymbol{x}$ is:

$$\Delta\boldsymbol{p}(\boldsymbol{x}) = -\sum_{j\in L} \boldsymbol{c}_j\,(f_j^*(\boldsymbol{x}) + f_{j'}^*(\boldsymbol{x})), \tag{11}$$

where we did not explicitly write the multiplicative factor $(\Delta x)^3$ (where Δx is the grid spacing) in front of the sum for simplicity, since it is always assumed to be 1 in normalized LBE space. The force acting on a solid by our double-sided bounce-back treatment can thus be approximated as the sum of all the momenta exchanged by all cut links that the solid intersects, leading to the expression:

$$\boldsymbol{F}_B \equiv \sum_{\boldsymbol{x}} \Delta\boldsymbol{p}(\boldsymbol{x}), \tag{12}$$

where we removed a division by Δt (the time step), which is also 1 in normalized LBE space. Similarly, the total torque is derived via:

$$\boldsymbol{\tau}_B \equiv \sum_{\boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{x}_C) \times \Delta\boldsymbol{p}(\boldsymbol{x}). \tag{13}$$

where $\boldsymbol{x}_C$ is the center of mass of the solid object.

Note that the above forces and torques on solids we discussed thus far only account for our bounce-back treatment, thus only a portion of the entire coupling: our ancillary velocity correction (which also causes momentum change in cut-cells) exerts additional forces and torques on solids (albeit smaller) that should also be taken into account. We thus apply, at each time step, the opposite (reaction) penalty forces $\boldsymbol{F}_p$ from Eq. 10 onto the solids as typically done in DI-IB, resulting in a total correction force $\boldsymbol{F}_C$ and a total

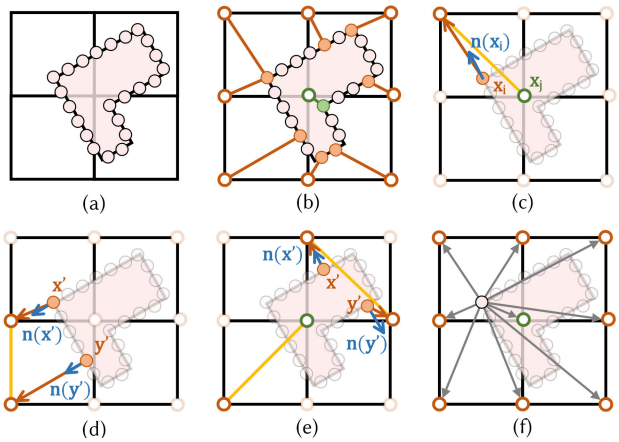


Fig. 12. **Efficient geometric approximation**. Given a solid geometry, we first sample its boundary (a); then the projected points of cut-cell nodes can be simply approximated by the nearest sample (b). To examine whether a link (yellow) crosses the solid boundary, we first check if solid nodes are involved. If the ends of the link are fluid and solid nodes, then it crosses the solid boundary (c) (green cut-cell nodes are inside the solid, whereas orange ones are in the fluid). For links connecting fluid nodes, we check the normals of the two projected cut-cell nodes forming the link; if the two normals have the same orientation (d), the link does not cross the solid boundary; otherwise, it intersects the solid boundary (e). For GPU implementation, we parallelize over solid samples instead of cut-cell nodes, where each solid sample will check its surrounding region of cut-cell nodes.

correction torque $\boldsymbol{\tau}_C$ induced by our velocity correction as:

$$\boldsymbol{F}_C = -\sum_{\boldsymbol{x}} \boldsymbol{F}_p(\boldsymbol{x}), \quad \boldsymbol{\tau}_C = -\sum_{\boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{x}_C) \times \boldsymbol{F}_p(\boldsymbol{x}). \tag{14}$$

Finally, the total force $\boldsymbol{F}_s$ and torque $\boldsymbol{\tau}_s$ acting on a solid are the sum of the forces and torques we just described:

$$\boldsymbol{F}_s = \boldsymbol{F}_B + \boldsymbol{F}_C \text{ and } \boldsymbol{\tau}_s = \boldsymbol{\tau}_B + \boldsymbol{\tau}_C.$$

3.5 Efficient Geometric Computations

Our hybrid coupling method requires geometric computations, e.g., point projection onto, and intersections of links with, possibly complex solid surfaces, as well as the identification of cut-cells and their node classifications so that our velocity corrections can be applied. Many existing methods could be used to offer precise or even exact geometric calculations [Azevedo et al. 2016; Robinson-Mosher et al. 2009]; however, they often significantly impact parallelism, drastically reducing the efficiency of the resulting LBM simulation. In order to compromise between accuracy and efficiency, we propose a sample-based approximation method which is of lower-order accuracy, but maintains the high degree of parallelism of traditional LBM. Moreover, visual simulation often does not necessitate very high accuracy (we will show however that our approach is still better at handling coupling in turbulent flows than most current CG techniques); but more accurate results can be obtained at the price of more computational time by increasing grid resolution.

Surface sampling and cut-cell identification. Instead of using the real geometry of boundaries, we first uniformly *sample* all solid boundary surface meshes — for instance via Poisson-disk sampling [Dunbar and Humphreys 2006]; note that the surface mesh should be watertight to facilitate this sample-based treatment. Each sample point $\boldsymbol{x}_s$ is then equipped with a *normal* $\boldsymbol{n}(\boldsymbol{x}_s)$, set to be the outward boundary normal at this point. We then declare a cell to be a *cut-cell* if it contains *at least one boundary sample*, which is simple and fast to evaluate, see Fig. 12 (a) for a 2D example.

Efficient cut-cell node projection. One of the most important (and potentially time-consuming) geometric computations is to project nodes $\boldsymbol{x}$ that lie in the fluid and belong to a cut-cell onto the solid boundary surface in order to compute their projection $\boldsymbol{x}'$, which will be used in both our double-sided bounce-back scheme and ancillary velocity correction. For boundaries described through a surface mesh, this is traditionally achieved by constructing a tree structure from the boundary mesh and searching the nearest triangle [Wang et al. 2012]; instead, our sampling of the boundary allows us to replace this projection procedure by an approximate, but very efficient algorithm: for each fluid node of a cut-cell, we pick the nearest sample $\boldsymbol{x}_s$ satisfying

$$(\boldsymbol{x} - \boldsymbol{x}_s) \cdot \boldsymbol{n}(\boldsymbol{x}_s) \geq 0 \tag{15}$$

to approximate the projected point $\boldsymbol{x}'$ — this constraint preventing the selection of samples that do not *face* the node. If no samples can be found, then the node is marked as a solid node; see the green node of Fig. 12 (b) for a 2D case. This sample-based technique is efficient and easily parallelizable, but the accuracy obviously depends on sample density; in practice, we ensure sufficient samples per cut-cell such that the approximation is still accurate by picking an appropriate minimum distance for the Poisson-disk sampling, typically set to half a grid cell size.

Approximate bounce-back scheme. In our double-sided bounce-back scheme, one must identify links that intersect a solid boundary (Fig. 7). For thick solids, since we have already identified whether cut-cell nodes are inside the solid, we only need to check the two end nodes of a link: if one of these two nodes is a fluid node and the other is a solid node, the link clearly intersects a solid boundary, and the DSBB scheme must be applied for that link instead of the regular streaming. However, for thin shells, all the nodes on a cut-cell are fluid nodes. Therefore, we proceed as follows. Given a link between node $\boldsymbol{x}$ and $\boldsymbol{y}$, we check *whether the normals of the two projected points* $\boldsymbol{x}'$ *and* $\boldsymbol{y}'$ *point in opposite directions*, i.e., $\boldsymbol{n}(\boldsymbol{x}') \cdot \boldsymbol{n}(\boldsymbol{y}') < 0$,

ALGORITHM 1: Our efficient hybrid boundary treatment in LBM

```
t ← 0;
while t < T do
    IdentifyCutCells();                                  ▷ Sec. 3.5
    CalculateCutCellNodeProjection();                    ▷ Sec. 3.5
    PerformStreaming();                                  ▷ Eq.5
    BounceBackTreatment();                               ▷ Secs. 3.2 and 3.5
    MomentUpdate();                                      ▷ Eq.7
    CalculateVelocityCorrection();                       ▷ Sec. 3.3
    PerformCollision();                                  ▷ Eq.6
    AddPenaltyForceToFluid();                            ▷ Eq.6
    CalculateForcesForRigidSimulation();                 ▷ Sec. 3.4
    t ← t + 1;
end
```

Fig. 13. **Smoke flow through stacked plates**. Smoke is blown towards a falling stack of thin plates. While smoke freely flows between separate plates, they become an airtight obstacle when they are stacked on top of each other. Our boundary treatment based on subgrid approximation deals with both cases seamlessly: plates getting closer accelerate the flow in between them, while tightly stacked plates are treated as thick solids.

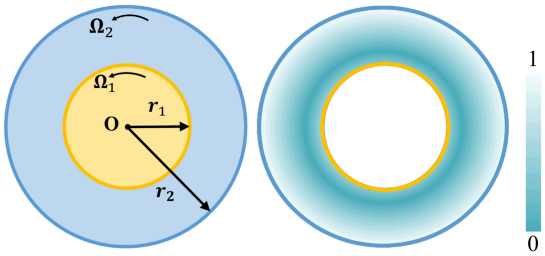


Fig. 14. **2D Taylor-Couette flow**. Left: the Taylor-Couette flow runs between two concentric circle boundaries locating at origin $\boldsymbol{O}$ with radii r_1 and r_2 whose rotating speeds are Ω_1 and Ω_2, respectively. Right: visualization of velocity magnitudes of the analytical solution between two rotating circles.

see Fig. 12 (d) and (e). If this last condition holds (Fig. 12 (e) shows two situations for a link crossing a solid boundary), $\boldsymbol{x}$ and $\boldsymbol{y}$ are on two different sides of a shell and the link is tagged as intersecting a boundary, and the DSBB scheme is then applied.

Algorithm 1 provides an overview of our whole boundary treatment method in the form of a pseudocode.

3.6 Analysis and Evaluations

In order to analyze the benefits of our boundary treatment and verify that it improves accuracy over existing methods, we conducted several 2D numerical simulations that we now discuss.

2D Taylor-Couette flow simulation. A classical 2D benchmark test for boundary conditions is the irrotational Taylor-Couette flow [Xu and Wang 2006] containing two concentric rotating circles acting as thin moving boundaries, see Fig. 14 (a) for an illustration. The two circles have radii $r_1 = 0.5$ and $r_2 = 1.0$, with rotating speeds $\Omega_1 = 1.0$ and $\Omega_2 = -1.0$, respectively. The Reynolds number is selected to be $\mathrm{Re} = 10$, corresponding to a kinematic viscosity $\nu = 0.1$. With this setup, the analytical expression of the velocity field $\boldsymbol{u}^{\mathrm{ref}} = (u_x^{\mathrm{ref}}, u_y^{\mathrm{ref}})$ between the two rotating boundaries for a Cartesian coordinate system (x, y) located at the center of the circles is, using $r = \sqrt{x^2 + y^2}$ and for $r \in [r_1, r_2]$,

$$u_x^{\mathrm{ref}} = -\left(A + B/r^2\right)y, \qquad u_y^{\mathrm{ref}} = \left(A + B/r^2\right)x\,, \tag{16}$$

where the constants A and B are defined as:

$$A = \frac{\Omega_2 r_2^2 - \Omega_1 r_1^2}{r_2^2 - r_1^2}, \qquad B = \frac{(\Omega_1 - \Omega_2) r_1^2 r_2^2}{r_2^2 - r_1^2}. \tag{17}$$

Fig. 14 (b) shows a visualization of the corresponding velocity distribution. We simulated this boundary-driven flow using different solvers: a typical kinetic solver with (a) bounce-back scheme [Ladd 1994], (b) DI-IB [Li et al. 2020], and (c) our hybrid boundary treatment method, as well as (d) an NS solver using commercial software ANSYS Fluent with adaptive meshing [Ansys Inc. 2014], where the average element area matches the area of the grid cell used in other uniform-grid simulations; since this last solver uses a boundary-fitted mesh, one expects less numerical error. Fig. 15 shows the visualization of relative error distributions for the different solvers (left to right) and different resolutions (top to bottom). Our method has markedly smaller errors around solid boundary compared to other kinetic solvers, and faster convergence as resolution increases. We also measured the error for different solvers using a weighted ℓ_2 metric to give more emphasis to the errors around the boundary:

$$\epsilon = \frac{\sum_i w(\boldsymbol{x}_i) \|\boldsymbol{u}(\boldsymbol{x}_i) - \boldsymbol{u}^{\mathrm{ref}}(\boldsymbol{x}_i)\|_2}{\sum_i w(\boldsymbol{x}_i) \|\boldsymbol{u}^{\mathrm{ref}}(\boldsymbol{x}_i)\|_2}\,, \tag{18}$$

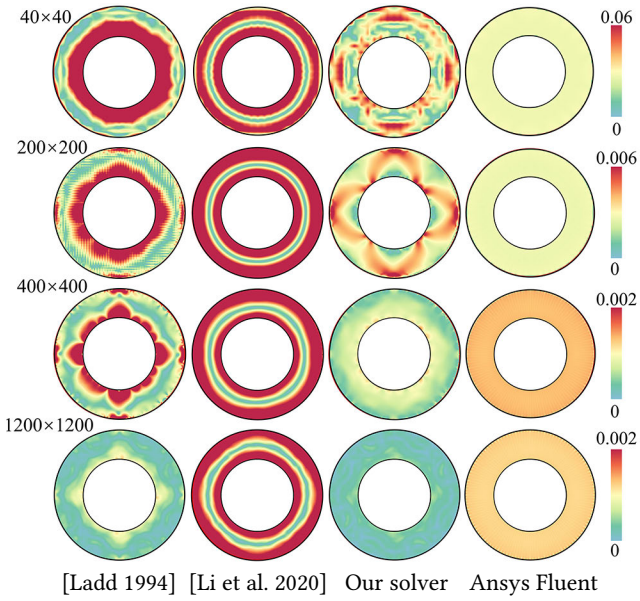


Fig. 15. **Comparison of 2D Taylor-Couette flow simulation**. We visualize the relative errors of the numerical solution to the 2D Taylor-Couette flow of Fig. 14 computed by different solvers (columns) and different resolutions (rows). From left to right: the original bounce-back scheme [Ladd 1994], the kinetic solver of [Li et al. 2020], our boundary treatment method, as well as the ANSYS Fluent solver using adaptive radial mesh for which the average element area matches the grid cell area used for the other solvers.

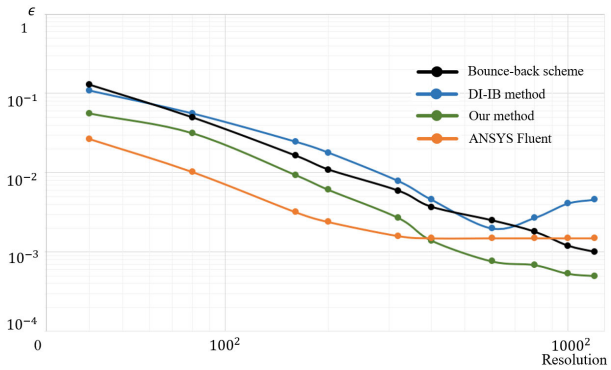


Fig. 16. **Convergence for different types of solvers**. We simulated the 2D Taylor-Couette flows with different types of solvers and measured the relative errors according to Eq. (18) to show the convergence for different solvers as resolution increases.

where $\boldsymbol{x}_i$ is a grid node inside the fluid domain, $\boldsymbol{u}^{\text{ref}}$ is the analytical solution, and the weight $w(\boldsymbol{x}_i) = 4|r - \bar{r}|$ where $\bar{r} = (r_1 + r_2)/2$. Fig. 16 shows the corresponding error plots for all solvers as a function of resolution. Our new boundary treatment is clearly and systematically providing better results compared to the two other existing kinetic methods. Moreover, it provides comparable or even more accurate results than Fluent as long as the regular grid is not too coarse: boundary fitted meshes offer an advantage that is obviously difficult to compete against — although the Fluent solver is surprisingly plateauing quickly in this test.

Comparison with 2D cut-cell based NS solver. There has been a variety of fluid-solid coupling methods for NS solvers proposed in computer graphics, among which cut-cell based approaches provide some of the most efficient, yet accurate results. We thus tried the approach from [Azevedo et al. 2016], where a finite-volume formulation is constructed within cut-cells for improved accuracy. We simulated a 2D rotating thin plate example with angular speed of $\omega_s = 3\text{rad}/s$ and compared our fluid velocity field with theirs, see Fig. 17. Since Azevedo *et al.* [2016] is supposed to simulate an inviscid Euler equation, we used no viscosity ($\nu = 0.0$) in our solver. Although their approach results in good boundary velocities around the thin plate, it only generates large vortices, even if the flow is supposed to be inviscid. While there also exist more accurate NS solvers [Zehnder et al. 2018; Qu et al. 2019], they were shown to be less computationally efficient than kinetic solvers in [Li et al. 2020]. Consequently, our novel hybrid boundary treatment is rather unique in that it offers more efficient and accurate simulations of turbulent flows, while being versatile in the type of solid structures that can be handled in coupling simulations.

4 IMPLEMENTATION AND RESULTS

We implemented our new fluid-solid coupling simulation and added a graphical user interface so that arbitrarily-shaped immersed obstacles can be loaded, interactively manipulated and simulated with a fluid flowing around. For rigid body simulation, we used the Chrono open source solver [Tasora et al. 2016]. In this section, we first discuss our specific implementation and the design choices we made to maximize efficiency, before discussing every result shown in the

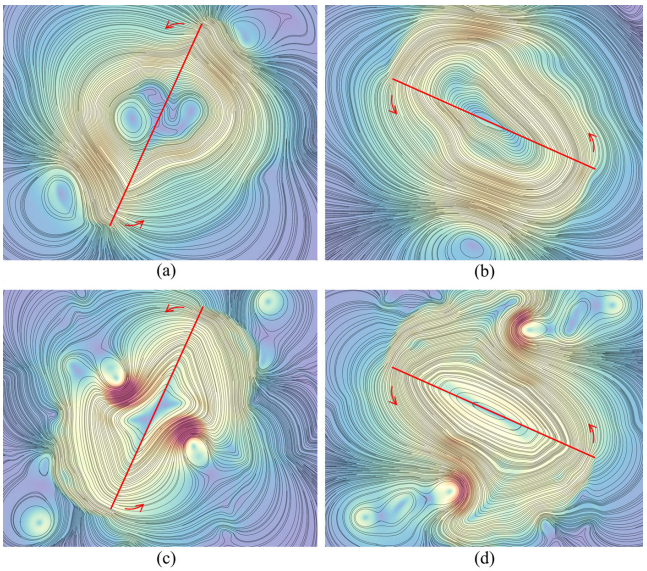


Fig. 17. **Comparison with 2D NS solver.** We simulated a 2D thin plate spinning in a fluid using a grid resolution of 256×256. Top: velocity magnitude plot from a cut-cell-based NS solver [Azevedo et al. 2016]; Bottom: velocity magnitude plot from our solver for the same frames as (a) and (b). Although both solvers retain good boundary velocities, ours produces much more detailed vortices, making it significantly more suitable for solving coupling in a turbulent flow scenario.

figures of this paper. Performance statistics and parameter values for all of our results are reported in Table 1 for reference; as traditionally done in LBM methods, all sizes and viscosity values shown in the table are *rescaled* from the actual ones based on *Reynolds number equivalence* (i.e., parameters are normalized without affecting the physics), and the timings reported in the table only include simulation time, excluding rendering. Visualization was achieved, depending on the examples, through cross sections showing the color-coded magnitude of the velocity field, or through smoke visualization using smoke particle tracers advected by the resulting vector field on the GPU and rendered by Redshift [Maxon 2021].

4.1 Efficient GPU-based Implementation

All of our algorithms so far, including geometric approximations, are designed to involve *purely local computations* with relatively simple algebra. It results in an overall coupling approach that is straightforward to implement on a parallel computing platform such as a GPU for efficient simulation: it respects the embarrassingly parallel nature of LBM. Our simulation method is very scalable since it can run on either a single or multiple GPU(s) by leveraging the GPU optimization techniques presented in [Chen et al. 2021]; our examples are computed with only one GPU, except for a high-resolution simulation shown in Fig. 2 where we used two GPUs instead. We discuss next a few specific accelerations and optimizations we incorporated in our GPU implementation in order to improve the overall performance for our two-way coupled simulations.

Memory layout. For LBM, a structure-of-arrays (SoA) memory layout is recommended to store discrete distribution functions f_i of all grid nodes to improve cache utilization and thus improve

Table 1. **Statistics.** Parameters and performance statistics for the simulations shown in the paper. Note that we use Reynolds number equivalence to rescale the physical parameters, as typically done in the LBM literature to offer normalized parameters; but by design, this does not affect the dynamic behavior.

Figures	*Grid Resolution*	*#Solid Samples*	*Domain Size* (m)	Δx (m)	*Viscosity* (ν)	*Average Flow Speed*	*#GPUs*	*Time for* 1 *s. of animation*
Fig. 1	800×200×300	820,897	0.8×0.2×0.3	1.0×10^{-4}	1.0×10^{-6}	1.0 m/s	1	727.45 s.
Fig. 2	889×333×556	1,701,440	0.8×0.3×0.5	9.0×10^{-4}	1.0×10^{-6}	1.0 m/s	2	1,095.93 s.
Fig. 4 (middle)	400×100×200	156,008	0.8×0.2×0.4	2.0×10^{-3}	1.0×10^{-6}	1.0 m/s	1	42.05 s.
Fig. 4 (right)	400×100×200	130,000	0.8×0.2×0.4	2.0×10^{-3}	1.0×10^{-6}	1.0 m/s	1	43.51 s.
Fig. 8 (left)	222×222×222	15,9200	0.4×0.4×0.4	1.8×10^{-3}	5.0×10^{-6}	0.5 m/s	1	172.35 s.
Fig. 8 (right)	222×222×222	15,9200	0.4×0.4×0.4	1.8×10^{-3}	5.0×10^{-6}	0.5 m/s	1	42.63 s.
Fig. 13	480×160×160	532,350	12.0×4.0×4.0	2.5×10^{-2}	1.56×10^{-5}	20.0 m/s	1	85.1 s.
Fig. 18 (top)	375×225×375	124,010	0.5×0.3×0.5	1.3×10^{-3}	4.4×10^{-5}	1.0 m/s	1	183.28 s.
Fig. 18 (bottom)	375×225×375	124,010	0.5×0.3×0.5	1.3×10^{-3}	4.4×10^{-7}	1.0 m/s	1	183.28 s.
Fig. 19	350×150×350	120,600	0.7×0.3×0.7	2.0×10^{-3}	1.0×10^{-6}	1.0 m/s	1	64.33 s.
Fig. 20	680×340×340	160,600	0.8×0.4×0.4	1.2×10^{-3}	1.0×10^{-6}	1.5 m/s	1	1,019.23 s.
Fig. 21	250×300×250	1,395,626	5.0×6.0×5.0	2.0×10^{-2}	1.56×10^{-5}	20.0 m/s	1	137.67 s.
Fig. 22	200×200×200	147,120	0.4×0.4×0.4	2.0×10^{-3}	1.0×10^{-6}	1.3 m/s	1	30.82 s.
Fig. 25 (top)	900×225×360	753,270	1.0×0.25×0.4	1.1×10^{-3}	8.0×10^{-6}	1.0 m/s	1	1,635.33 s.
Fig. 25 (bottom)	900×225×360	753,270	1.0×0.25×0.4	1.1×10^{-3}	8.0×10^{-6}	1.0 m/s	1	493.59 s.
Fig. 26	667×250×334	175,093	0.8×0.3×0.4	1.2×10^{-3}	5.0×10^{-6}	0.5 m/s	1	329.05 s.
Fig. 27 (top)	900×225×360	763,243	1.0×0.25×0.4	1.1×10^{-3}	8.0×10^{-6}	1.0 m/s	1	503.32 s.
Fig. 27 (bottom)	900×225×360	773,250	1.0×0.25×0.4	1.1×10^{-3}	8.0×10^{-6}	1.0 m/s	1	504.26 s.

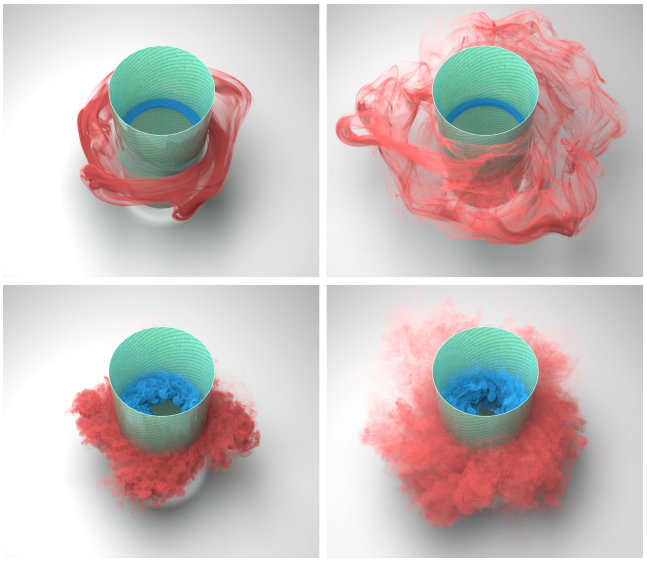


Fig. 18. **Rotating cylindrical thin shell**. A cylindrical thin shell rotating around its axis in a very high (top) and very low viscosity (bottom) fluid respectively, where smoke particles scattered inside (blue) and outside (red) the thin-shell cylinder are advected in the flow (left: after 4 seconds; right: after 7 seconds) to demonstrate that boundary layers are well resolved.

performance [Chen et al. 2021]. We follow this recommendation in our GPU implementation as our tests showed that the SoA layout is still the most efficient, even with our new coupling strategy.

Efficient collision evaluation. Also discussed and demonstrated in [Chen et al. 2021] was the computation of the inverse of the central-moment projection matrix $\boldsymbol{M}(\boldsymbol{u})$. They argued that this inverse should be computed analytically to preserve accuracy, but the algebraic expression is very lengthy and occupies many registers, resulting in low achieved GPU occupancy in practice. However, we observed that if one performs an analytical LU decomposition [Fei et al. 2018b] of $\boldsymbol{M}(\boldsymbol{u})^{-1}$, many elements of the resulting triangular matrices are very close to zero. Thus, we can discard those close-to-zero elements and employ the usual sparse matrix format to store the decomposed matrices, decreasing drastically the terms needed to compute the inverse. This does not influence accuracy (the reconstruction error is below machine accuracy), but significantly reduces the use of registers; the performance for parallel collision calculation is then three times faster on average compared to the implementation using [Chen et al. 2021] on the same GPU.

Parallel cut-cell point projection and identification. Throughout all the stages of our hybrid coupling algorithm, a key operation is to project a cut-cell node onto the nearest solid boundary, which is approximated by searching for the nearest sample point on the solid boundaries. While there are traditional GPU implementations to achieve this task in parallel over cut-cell nodes, they often generate load imbalance and low GPU occupancy issues. Thus, as suggested in [Chen et al. 2021], we parallelize over solid samples instead — but proceed differently to improve performance. For each sample, we compute the distance to each node of the cut cell in which the sample is located (see Fig. 12 (f)) and compare with the distance stored in that node: if the distance of the sample to a cut-cell node is smaller than the current distance stored in that node, we update this distance with the new distance, and change the index of the solid sample stored along with that same node. Since the same cut-cell node may be accessed from multiple solid sample threads, we employ an atomic comparison operation to avoid thread conflicts. Note that Eq. (15) still needs to be satisfied for every comparison, so the identification of solid nodes is included in the process.

4.2 Simulation Results

All simulations presented in this paper were conducted with our fluid simulator platform, on a workstation with a 14-core Intel Xeon E5-2690 CPU, 128 GB of memory and up to two NVIDIA RTX 3090 GPUs. We tested a number of different one-way and two-way examples and configurations with a wide range of geometric complexity to demonstrate that we can simulate fluid-solid coupling with both thick and thin solid structures, and for both laminar and turbulent flows. We also show a spectrum of results covering an interactive simulation as well as offline high-resolution large-scale simulations with complex solid structures to illustrate scalability. We discuss each example below (available in the *supplementary video*).

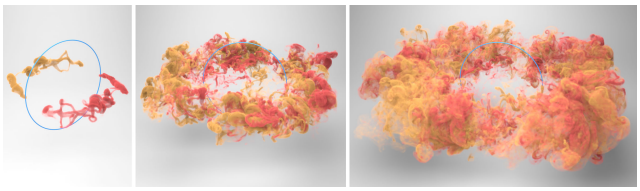


Fig. 19. **Rotating ring**. A very thin ring rotating along a vertical axis and emitting yellow and orange smoke particles is enough to create a turbulent wake as evidenced by the volutes of smoke resulting from the motion.

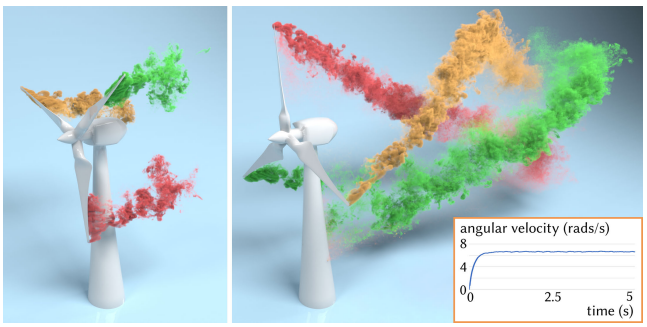


Fig. 20. **Turbine**. A large turbine emitting colored smoke particles at the three propellers of its large blade is simulated with a constant incoming air flow making it turn, creating a spiral vortex trail. This two-way coupling example also involves friction on the axis of the turbine to limit its maximum angular velocity, resulting in a time-varying but converging curve of angular velocity of the turbine shown in the bottom inset.

Coupling with thin shells. We first simulate a simple thin shell in the shape of a cylinder rotating at a constant speed, and thus generating a shear flow, see Fig. 18; this setup computed at low Reynolds number (high viscosity) exhibits a very different behavior from its high Reynolds number (low viscosity) counterpart, as expected. A case where multiple thin shells are piling up is also tested, see Fig. 13, where the plates are first well separated, before getting stacked up. Here, our sub-grid approximation produces plausible coupling results, with a seamless transition between separate plates early on and an airtight block of plates by the end of the animation: our method degrades gracefully in the presence of obstacles that it cannot be resolved by the grid. A two-way coupling example in Fig. 21 shows dried-up leaves (each far thinner than a grid cell size) being blown by an upward gust of wind. As expected, the dead leaves are lifted up by the air flow, and fall back slowly on the ground while being rotated by the surrounding air.

Coupling with thin rods. We also show that our solver can handle thin rods, first with a simple example where a thin ring-like object rotates with a constant speed in the air. Our boundary treatment method generates a reasonable turbulent wake flow, see Fig. 19. When several rods are grouped together, our sub-grid approximation still creates visually plausible coupling results as demonstrated in Fig. 4; note the increased amount of turbulent flows near the solid boundary as well as in the wake due to the addition of bristles.

Coupling with complex solid. A more complicated, yet very common case is when both thick and thin structures form a complex solid shape that interacts with a fluid. Our hybrid boundary treatment method is an efficient and unified approach to supporting such cases. Indeed, Fig. 20 shows the two-way coupling simulation of a three-blade wind turbine where the air flow blows through the turbine to drive its rotating motion, and in turn, the air flow gets affected by the motion. Additionally, Fig. 1 shows a simulation of the Concorde airplane flying in the air with an angle of attack of 20°; for such a complex shape and for a reasonable grid size, many of the plane's parts (in particular near the wings) are thin structures. In both cases, the results are consistent with expectations.

Fast simulation for interactive design. Our fluid-solid coupling simulator is particularly efficient given its high level of parallelism, which allows for interactive testing of product design even with a single GPU. As an example, an interactive simulation conducted with a 200×200×200 grid resolution for a fast rotating blade is performed in Fig. 22, where a cross-sectional magnitude plot of the velocity field of the surrounding air is displayed. One frame corresponding to an animation of 1/100 s. is computed in less than 0.4 second including the data read-back and visualization time (cross-section visualization is currently done on CPU), meaning that a preliminary design of the blade can be quickly evaluated, e.g., for turbulence reduction by optimizing the geometric shape. Our supplementary video contains a recording of the whole user interaction on our simulation platform for this example, captured in real time.

High-resolution simulations. Finally, we simulate a high-resolution large-scale scenario to demonstrate the scalability of our approach, using this time two GPUs. Fig. 2 shows a mass of smoke particles passing through a neighborhood of a high-rise, high-density city, where the buildings containing thin and sharp structures affect the air flow. For this very large-scale example with fine turbulence details, 5 seconds of animation took only around 1.5 hours to compute; as discussed in detail in [Li et al. 2020], such an efficiency for this level of accuracy is currently out of reach for NS-based solvers even with multi-GPU accelerations: their pressure projection and implicit viscosity handling require accurate matrix solves that are too time consuming if realistic vortical structures are desired.

4.3 Comparisons

In order to highlight the advantages of our new solver for fluid-solid coupling involving both thick and thin solid structures, we compare qualitatively and quantitatively our simulation platform with the most recent LBM-based solver used in CG [Li et al. 2020], as

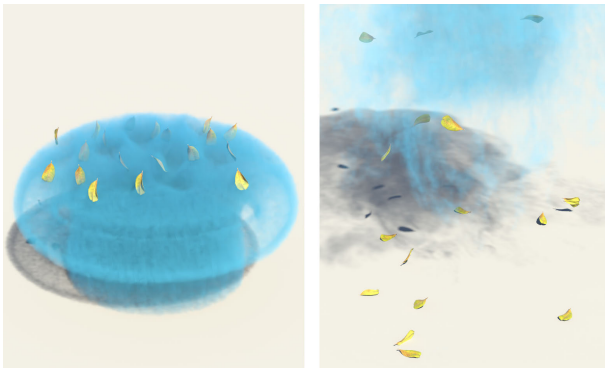


Fig. 21. **Wind blowing leaves**. In this two-way coupling example, a puff of wind from the ground drives a bunch of dried-up leaves up in the air.

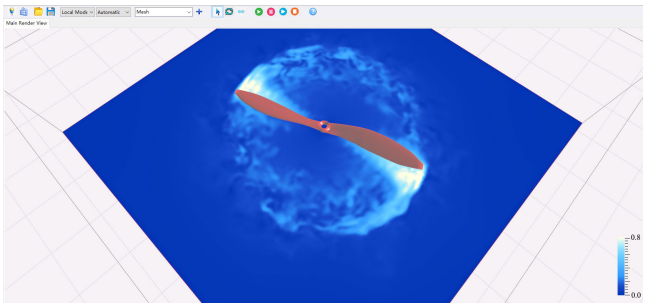


Fig. 22. **Fast simulation for interactive design**. We operated our GUI-based simulation system to produce a quick preliminary result of a fast rotating rotor-blade where turbulent wake flows can be observed interactively, which is very useful for efficient product design and verification.

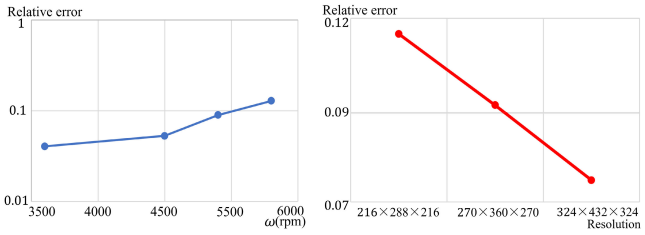


Fig. 23. **Thrust evaluation**. From the simulation of the coupling between a rotating drone blade and the surrounding air, we can compute the estimated thrust values at different rotation speeds with a grid resolution of $288 \times 324 \times 288$ (left). The relative errors of our thrust value at 3600 rpm w.r.t. experimental measurements as the grid resolution increases (right) indicates convergence of our solver. Note that the tested grid resolutions are not very high, and more accurate results are expected for higher grid resolutions.

well as with a set of real experiments. We also discuss our computational performance by comparing again to [Li et al. 2020] with improvements from [Chen et al. 2021], as they offer state-of-the-art efficiency for fluid-solid coupling in turbulent flows.

Comparison with recent coupling method in graphics. Recently, a kinetic solver demonstrated strong capabilities for fluid-solid two-way coupling with turbulent flows, employing a DI-IB method to treat solid boundaries [Li et al. 2020]. However, many limitations were acknowledged. Since DI-IB method does not act in a one-sided manner, it exhibits flow penetration for obstacles of size equal or smaller than a grid cell. Fig. 8 shows a comparison for a 3D jet flow shot onto a thin plate, where (a) [Li et al. 2020] creates an unexpected smoke flow on the other side of the plate as marked by the red box, while (b) our new method correctly enforces non-penetration through this thin structure. Additionally, the low accuracy of the DI-IB method can fail to produce the right simulation result, especially for boundary layers around a solid object when the flow is turbulent. A clear and obvious example of this issue is the airflow simulation around a moving car model shown in Fig. 25. As is well-known by car designers, a reasonable design for a car should induce a flow separation at the rear of the car to reduce aerodynamic drag, and the use of rear spoilers (or "ducktails") helps push the flow separation further away, as can be seen in Fig. 27 (d) in a real wind tunnel experiment. The DI-IB method of [Li et al. 2020] in this case predicts a boundary layer which separates at the top of the car, far too

Fig. 24. **Wind-tunnel test of a car**. A wind tunnel visualization of the airflow around a car shows that the design of the body delays boundary layer separation until the trunk, which reduces drag and vibration in practice. Image courtesy of *Auto Motor und Sport*, ©Frank Herzog/Sportauto.

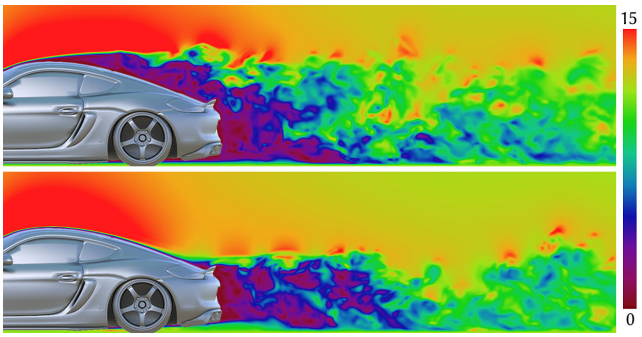


Fig. 25. **Comparison of aerodynamic simulations**. DI-IB method [Li et al. 2020] (top) vs. our result (bottom), using a visualization showing the color-coded velocity field magnitude (in km/h) of a cross section. While our simulation matches the wind-tunnel visualization of air flow around a car as Fig. 24 demonstrates, the DI-IB method, on the other hand, fails to have accurate boundary treatment, leading to unreasonable boundary layer separation at the top of the car.

forward compared to the wind tunnel experiment; instead, our new boundary treatment method (for exactly the same grid resolution) reproduces the correct behavior. To demonstrate further how a small, thin structure like a ducktail can affect the flow around a car, we also show in Fig. 27 that the absence of the rear spoiler changes dramatically the turbulence patterns witnessed behind the car: the spoiler thus acts as a wake generator, and sometimes lift reducer as well, to improve the maneuverability at high speed. These two examples both point to our method having an improved accuracy compared to the existing LBM methods in graphics.

Comparison with real experiments. To further estimate the accuracy of our boundary treatment especially for thin structures, we conducted comparisons of our simulations with two real experiments. In a first comparison, we simulated the airflow through a delta wing with a 75° swept angle and for an angle-of-attack of 20°, as shown in Fig. 26 (a). This well-known test case is expected to create stable spiral vortices above the wing to increase aerodynamic lift (the so-called "vortex lift" in aeronautics [Anderson 2010], which has frequently been used in modern aircraft design such as the Concorde airplane for example in Fig. 1), as shown by the experimental visualization in Fig. 26 (b) produced by [Délery 2001]. Our result matches the experimental results visually, producing similar spiral vortex structures. In addition to phenomenological comparisons, we provide an additional quantitative evaluation of our method by

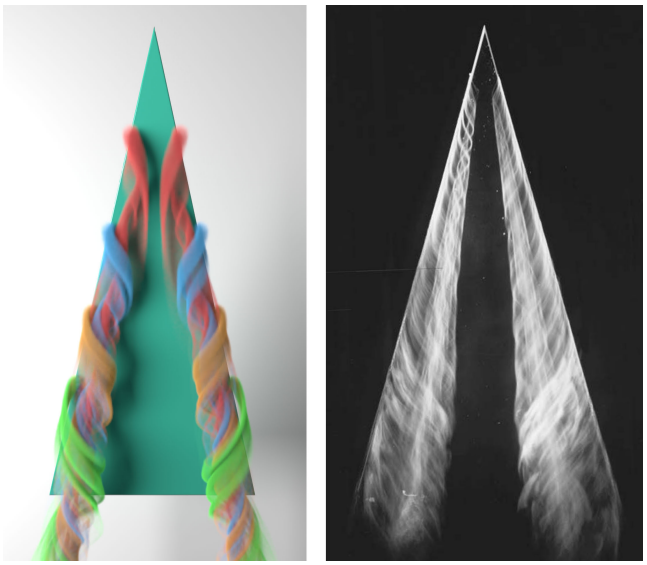


Fig. 26. **Delta-wing simulation**. The airflow over a thin-shell delta-wing simulated with our hybrid coupling approach (left) matches experimental visualizations from [Délery 2001] (right), exhibiting the same spiral vortex structure near the leading edge of the wing and demonstrating the accuracy of our solver in capturing boundary layer flows around thin structures.

simulating the aerodynamic flow of a rotating rotor blade, where we recreated the shape of a real drone blade based on the geometric description provided in a recent patent [Lin et al. 2020]. We simulated the coupling using different rotating speeds, computed the resulting thrust values according to [Leishman 2016], and compared our numerical approximations to the measurements given in the patent (which we assume to be accurate). Fig. 23 (a) shows a plot of our thrust approximation error as the rotating speed increases, indicating good agreement with the experimental measurements even with a relatively coarse grid. Note that a major reason for the increased error observed at higher rotating speed is that after rescaling, the effective viscosity in the LBM domain becomes so small that it reaches the limit of precision for 32-bit floating-point numbers, affecting the overall accuracy for collision; to retain accuracy for high rotating speed, double precision (64-bit) should be used instead. We also conducted convergence tests of our thrust approximation at a fixed rotating speed but for increasing grid resolutions in Fig. 23 (b), indicating consistency of our method.

Performance comparison. As demonstrated in [Li et al. 2020] and [Chen et al. 2021], a kinetic solver has much higher computational performance than NS solvers, especially for turbulent flows with coupling where relatively small time steps are required. Since our new kinetic solver is built upon the same framework, we retain this competitive advantage even when their DI-IB boundary treatment is replaced by our hybrid method. However, because of our hybrid boundary treatment and new GPU optimizations discussed in Sec 4.1, we actually exceed their performance at all grid resolutions (using the same NVIDIA GeForce RTX 3090 GPU for fairness of comparisons) as shown in Fig. 28. Two factors are responsible for this improved computational efficiency: profiling the two codes shows, as expected, that our simplification of the collision operator via analytical LU decomposition [Fei et al. 2018b] improves GPU occupancy, and that our hybrid coupling treatment with sample-based geometric approximations, despite being more accurate in practice, involve less computations with reduced penalty forces (less stiffness) than the original DI-IB method, thus allowing larger time steps for faster simulations.

4.4 Limitations

Despite its excellent numerical results and increased generality, our method still suffers from limitations. First, due to our sample-based geometric representation, the approximate bounce-back scheme may have undesired behavior at extremely sharp corners, for example, at the tip of a delta wing, since the projection achieved via the nearest available sample may not be accurate. Increasing sample size or using adaptive sampling can improve the accuracy, but at a cost of more memory usage. For visual applications, however, the resulting animations are not particularly different, so this issue mostly matters for the rapid testing of product design. Second, our total force and torque applied to the solid resulting from our two-way coupling approach inherit the typical limitation of a staircase approximation of the solid boundary on grid: for coarse grids, our proposed sub-grid approximation cannot faithfully reflect the true geometry, affecting the accuracy of the force and torque on solid. Finally, an efficient sampling method that can adjust to a deforming geometry is currently needed.

5 CONCLUSION

In this paper, we proposed a hybrid coupling treatment for kinetic solvers that enables robust, fast and plausible handling of a large variety of obstacle shapes including both thick and thin structures. Our approach leverages the complementary advantages of the bounce-back scheme and the sharp-interface immersed boundary method by introducing a novel unified approach: it consists of a double-sided bounce-back treatment followed by a one-sided velocity correction, together with a hybrid boundary force and torque calculation to enable two-way coupling. Efficient geometric approximations based on boundary point-samples (along with their parallel implementations on GPU) were proposed to improve computational performance, and further improvements in the GPU implementation of existing LBM solvers were also offered. The resulting simulator was shown to exceed the accuracy of existing state-of-the-art solvers, to further improve on their efficiency, and to capture the correct physical behavior of complex fluid-solid interactions even for very turbulent flows. A wide range of simulation results with different types of immersed boundary shapes were demonstrated, followed by comparisons to other coupling methods as well as physical experiments to show the potential applicability of our method in both efficient visual production and fast computational design. This capacity to rival both CG-based and CFD-based fluid methods in efficiency and accuracy is particularly promising, and we hope it will lead to a variety of follow-up works in both communities.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for helping us improve the exposition. This work was supported by the National Natural Science Foundation of China (No. 62072310 and No. 61976138)

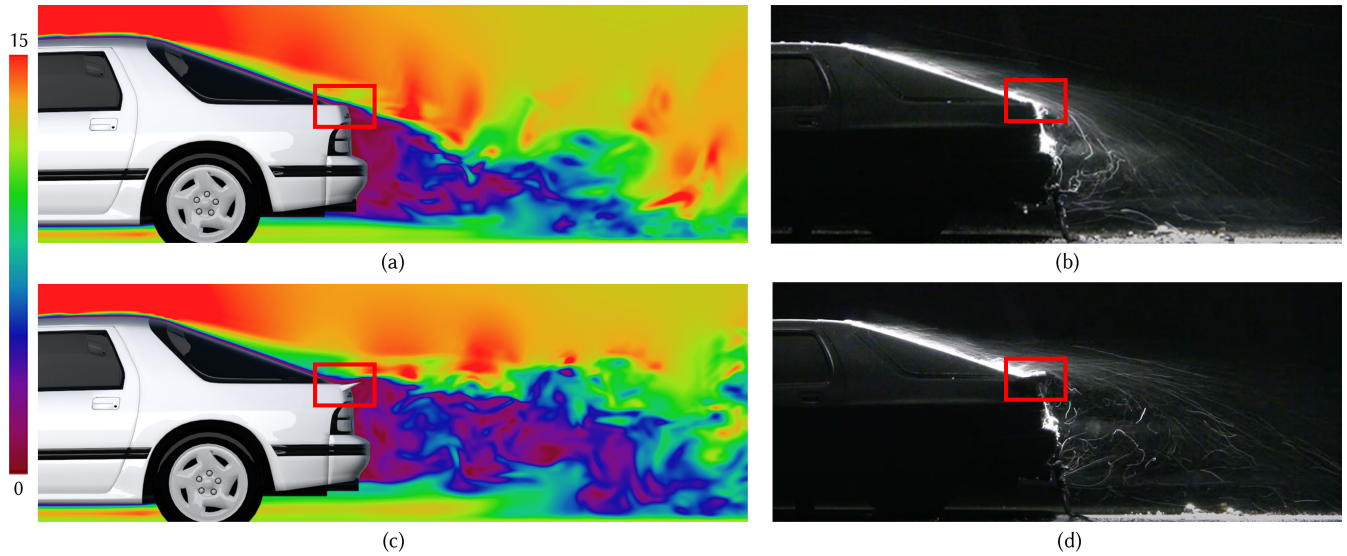


Fig. 27. **Aerodynamic design of a car model**. Simulating the airflow around a car model without a spoiler using our solver (a) matches the wind tunnel visualization for a similar car model (b); A small and thin spoiler (in red box) added on the back of the car model (c) changes the wake flow of the car model in our simulation quite significantly, in line with a wind-tunnel test of a car model with a spoiler (d) — indicating that our solver offers an efficient, yet predictive tool of air flows for computational shape design. The velocity magnitude is measured in km/h. Image (b) and (d) courtesy of Feltham [2014].

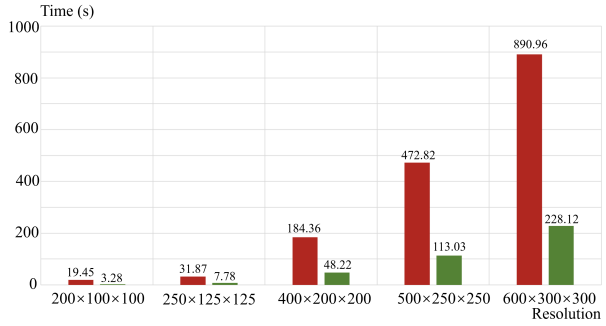


Fig. 28. **Performance comparison.** We compare the efficiency of the kinetic solver from [Li et al. 2020; Chen et al. 2021] (red) with our kinetic solver (green) based on one second of simulation of the car model shown in Fig. 25 and both executed on a same NVidia RTX 3090 GPU, showing significant performance improvement for all grid resolutions.

and ShanghaiTech University. M. Desbrun gratefully acknowledges generous support from Ansys Inc. 3D meshes were provided by GrabCAD users Aisak (Fig. 1), Mehmet Boztaş (turbine blade in Fig. 20), Vedad Saletovic (turbine tower in Fig. 20), Dhanasekar Vinayagamoorthy (Fig. 22), and CustomWorkx Belgium (Fig. 25), as well as Sketchfab users Cosche (Fig. 4), Opus Poly (Fig. 21), and Lexyc16 (Fig. 27).

REFERENCES

Cyrus K Aidun and Jonathan R Clausen. 2010. Lattice-Boltzmann method for complex flows. *Annual Review of Fluid Mechanics* 42 (2010), 439–472.

Cyrus K Aidun, Yannan Lu, and E-Jiang Ding. 1998. Direct analysis of particulate suspensions with inertia using the discrete Boltzmann equation. *Journal of Fluid Mechanics* 373 (1998), 287–311.

John D Anderson. 2010. *Aircraft Performance & Design*. McGraw-Hill Education (India) Pvt Limited.

Ansys Inc. 2014. Ansys Fluent. (2014). www.ansys.com/products/fluids/ansys-fluent

Vinicius C Azevedo, Christopher Batty, and Manuel M Oliveira. 2016. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Transactions on Graphics* 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 Transactions on Graphics* 37, 2, Article 14 (2018).

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. In *ACM Transactions on Graphics*, Vol. 26. 100.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete Elastic Rods. *ACM Transactions on Graphics*, Article 63 (2008).

M'hamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. 2001. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids* 13, 11 (2001), 3452–3459.

R. Bridson, S. Marino, and R. Fedkiw. 2003. Simulation of clothing with folds and wrinkles. In *Symposium on Computer Animation*. 28–36.

Yunan Cai, Sheng Li, and Jianhua Lu. 2018. An improved immersed boundary-lattice Boltzmann method based on force correction technique. *International Journal for Numerical Methods in Fluids* 87, 3 (2018), 109–133.

Mark Carlson, Peter J Mucha, and Greg Turk. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM Transactions on Graphics*, Vol. 23. 377–384.

Li Chen, Yang Yu, and Guoxiang Hou. 2013. Sharp-interface immersed boundary lattice Boltzmann method with reduced spurious-pressure oscillations for moving boundaries. *Physical Review E* 87, 5 (2013), 053306.

Shiyi Chen and Gary D Doolen. 1998. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics* 30, 1 (1998), 329–364.

Yixin Chen, Wei Li, Rui Fan, and Xiaopei Liu. 2021. GPU Optimization for High-Quality Kinetic Fluid Simulation. *IEEE Transactions on Visualization and Computer Graphics* (2021).

Nuttapong Chentanez, Tolga G. Goktekin, Bryan E. Feldman, and James F. O'Brien. 2006. Simultaneous Coupling of Fluids and Deformable Bodies. In *Symposium on Computer Animation*. 83–89.

Pascal Clausen, Martin Wicke, Jonathan R Shewchuk, and James F O'Brien. 2013. Simulating liquids and solid-liquid interactions with Lagrangian meshes. *ACM Transactions on Graphics* 32, 2, Article 17 (2013).

Meizhong Dai and David P Schmidt. 2005. Adaptive tetrahedral meshing in free-surface flow. *J. Comput. Phys.* 208, 1 (2005), 228–252.

Orla Dardis and John McCloskey. 1998. Lattice Boltzmann scheme with real numbered solid density for the simulation of flow in porous media. *Physical Review E* 57, 4 (1998), 4834.

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 (2019), 013301.

Daniel Dunbar and Greg Humphreys. 2006. A spatial data structure for fast Poisson-disk sample generation. *ACM Transactions on Graphics* 25, 3 (2006), 503–508.

Jean M Délery. 2001. Robert Legendre and Henri Werlé: Toward the Elucidation of Three-Dimensional Separation. *Annual Review of Fluid Mechanics* 33, 1 (2001), 129–154. https://doi.org/10.1146/annurev.fluid.33.1.129

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

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

Yu Fang, Ziyin Qu, Minchen Li, Xinxin Zhang, Yixin Zhu, Mridul Aanjaneya, and Chenfanfu Jiang. 2020. IQ-MPM: an interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids. *ACM Transactions on Graphics* 39, 4 (2020), 51–1.

Linlin Fei, Kai H Luo, and Qing Li. 2018b. Three-dimensional cascaded lattice Boltzmann method: Improved implementation and consistent forcing scheme. *Physical Review E* 97, 5 (2018), 053309.

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

Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2019. A Multi-Scale Model for Coupling Strands with Shear-Dependent Liquid. *ACM Transactions on Graphics* 38, 6 (2019).

Bryan E Feldman, James F O'Brien, and Bryan M Klingner. 2005a. Animating gases with hybrid meshes. In *ACM Transactions on Graphics*, Vol. 24. 904–909.

Bryan E. Feldman, James F. O'Brien, Bryan M. Klingner, and Tolga G. Goktekin. 2005b. Fluids in deforming meshes. In *Symposium on Computer Animation*. 255–259.

Grahaml Feltham. 2014. Automotive Aerodynamics Episode 1: Flow Visualizations of MR2, RX7, Supra, FRS. https://www.youtube.com/watch?v=quDLzxmJl5I

Nick Foster and Ronald Fedkiw. 2001. Practical Animation of Liquids. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*. 23–30.

Yang Gao, Shuai Li, Hong Qin, and Aimin Hao. 2017. A Novel Fluid-Solid Coupling Framework Integrating FLIP and Shape Matching Methods. In *Computer Graphics International*. Article 11.

Sebastian Geller, Jonas Tölke, and Manfred Krafczyk. 2006. Lattice-Boltzmann method on quadtree-type grids for fluid-structure interaction. In *Fluid-Structure Interaction*. 270–293.

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.

Boyce E Griffith and Neelesh A Patankar. 2020. Immersed methods for fluid–structure interaction. *Annual Review of Fluid Mechanics* 52 (2020), 421–448.

Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. 2003. Discrete Shells. In *Symposium on Computer Animation*. 62–67.

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

Zhaoli Guo, Chuguang Zheng, and Baochang Shi. 2002. Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Physical review E* 65, 4 (2002), 046308.

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 Transactions on Graphics* 37, 4, Article 150 (2018).

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.

Chen Jiang, Jian-Yao Yao, Zhi-Qian Zhang, Guang-Jun Gao, and GR Liu. 2018. A sharp-interface immersed smoothed finite element method for interactions between incompressible flows and large deformation solids. *Computer Methods in Applied Mechanics and Engineering* 340 (2018), 24–53.

Shin K Kang and Yassin A Hassan. 2011. A comparative study of direct-forcing immersed boundary-lattice Boltzmann methods for stationary complex boundaries. *International Journal for Numerical Methods in Fluids* 66, 9 (2011), 1132–1158.

Po-Hao Kao and Ruey-Jen Yang. 2008. An investigation into curved and moving boundary treatments in the lattice Boltzmann method. *J. Comput. Phys.* 227, 11 (2008), 5671–5690.

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

Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. 2017. The lattice Boltzmann method. *Springer International Publishing* 10, 978-3 (2017), 4–15.

Anthony J. C. Ladd. 1994. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of Fluid Mechanics* 271 (1994), 285–309.

Pierre Lallemand and Li-Shi Luo. 2003. Lattice Boltzmann method for moving boundaries. *J. Comput. Phys.* 184, 2 (2003), 406–421.

J Gordon Leishman. 2016. *Principles of helicopter aerodynamics*. Cambridge University Press.

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, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling. *ACM Transactions on Graphics* 39, 4 (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.

Jiajing Lin, Peng Chen, and Kuo Liang. 2020. Screw, power component and aircraft. CN Patent 110,896,626.

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

Jianhua Lu, Haifeng Han, Baochang Shi, and Zhaoli Guo. 2012. Immersed boundary lattice Boltzmann model based on multiple relaxation times. *Physical Review E* 85, 1 (2012), 016711.

Maxon. 2021. Redshift renderer. (2021). https://www.redshift3d.com/product

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.

Marek Krzysztof Misztal, Robert Bridson, Kenny Erleben, Jakob Andreas Bærentzen, and François Anton. 2010. Optimization-based fluid simulation on unstructured meshes. In *VRIPHYS*. 11–20.

Rajat Mittal, Haibo Dong, Meliha Bozkurttas, FM Najjar, Abel Vargas, and Alfred Von Loebbecke. 2008. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *J. Comput. Phys.* 227, 10 (2008), 4825–4852.

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.

DR Noble and JR Torczynski. 1998. A lattice-Boltzmann method for partially saturated computational cells. *International Journal of Modern Physics C* 9, 08 (1998), 1189–1201.

Nathan M. Olson. 2015. A Near-Boundary Modification for the Link Bounce-Back Boundary Condition in the Lattice Boltzmann Method. *J. Comput. Phys.* 301, C (2015), 102–110.

Jitendra Kumar Patel and Ganesh Natarajan. 2018. Diffuse interface immersed boundary method for multi-fluid flows with arbitrarily moving rigid bodies. *J. Comput. Phys.* 360 (2018), 202–228.

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An implicit viscosity formulation for SPH fluids. *ACM Transactions on Graphics* 34, 4 (2015), 1–10.

Cheng Peng, Yihua Teng, Brian Hwang, Zhaoli Guo, and Lian-Ping Wang. 2016. Implementation issues and benchmarking of lattice Boltzmann method for moving rigid particle simulations in a viscous flow. *Computers & Mathematics with Applications* 72, 2 (2016), 349–374.

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

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and conservative fluids using bidirectional mapping. *ACM Transactions on Graphics* 38, 4 (2019), 1–12.

Avi Robinson-Mosher, R. Elliot English, and Ronald Fedkiw. 2009. Accurate tangential velocities for solid fluid coupling. In *Symposium on Computer Animation*. 227–236.

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 Transactions on Graphics*, Vol. 27. 46.

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

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.

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.

Tetsuya Takahashi and Christopher Batty. 2020. Monolith: a monolithic pressure-viscosity-contact solver for strong two-way rigid-rigid rigid-fluid coupling. *ACM Transactions on Graphics* 39, 6 (2020), 1–16.

Tetsuya Takahashi and Ming C. Lin. 2019. A Geometrically Consistent Viscous Fluid Solver with Two-Way Fluid-Solid Coupling. *Comp. Graph. Forum* 38, 2 (2019), 49–58.

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

Shi Tao, Qing He, Baiman Chen, and Simin Huang. 2018. A distribution function correction-based immersed boundary-lattice Boltzmann method with truly second-order accuracy for fluid-solid flows. *arXiv preprint arXiv:1803.09380* (2018).

Shi Tao, Qing He, Jiechao Chen, Baiman Chen, Guang Yang, and Zhibin Wu. 2019. A non-iterative immersed boundary-lattice Boltzmann method with boundary condition enforced for fluid–solid flows. *Applied Mathematical Modelling* 76 (2019), 362–379.

Alessandro Tasora, Radu Serban, Hammad Mazhar, Arman Pazouki, Daniel Melanz, Jonathan Fleischmann, Michael Taylor, Hiroyuki Sugiyama, and Dan Negrut. 2016. Chrono: An open source multi-physics dynamics engine. In *International Conference on High Performance Computing in Science and Engineering*. 19–49.

Yun Teng, David I. W. Levin, and Theodore Kim. 2016. Eulerian Solid-fluid Coupling. *ACM Transactions on Graphics* 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).

Stuart DC Walsh, Holly Burwinkle, and Martin O Saar. 2009. A new partial-bounceback lattice-Boltzmann method for fluid flow through heterogeneous media. *Computers & Geosciences* 35, 6 (2009), 1186–1193.

Kevin Wang, J Grétarsson, A Main, and C Farhat. 2012. Computational algorithms for tracking dynamic fluid–structure interfaces in embedded boundary methods. *International Journal for Numerical Methods in Fluids* 70, 4 (2012), 515–535.

Zhengdao Wang, Yikun Wei, and Yuehong Qian. 2020. A bounce back-immersed boundary-lattice Boltzmann model for curved boundary. *Applied Mathematical Modelling* 81 (2020), 428–440.

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.

Sheng Xu and Z Jane Wang. 2006. An immersed interface method for simulating the interaction of a fluid with moving boundaries. *J. Comput. Phys.* 216, 2 (2006), 454–493.

Xuewen Yin and Junfeng Zhang. 2012. An improved bounce-back scheme for complex boundary conditions in lattice Boltzmann method. *J. Comput. Phys.* 231, 11 (2012), 4295–4303.

Gary D. Yngve, James F. O'Brien, and Jessica K. Hodgins. 2000. Animating Explosions. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*. 29–36.

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics* 37, 4 (2018), 1–8.

Chunze Zhang, Yongguang Cheng, Luoding Zhu, and Jiayang Wu. 2016a. Accuracy improvement of the immersed boundary–lattice Boltzmann coupling scheme by iterative force correction. *Computers & Fluids* 124 (2016), 246–260.

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