Building a Virtual Weakly-Compressible Wind Tunnel Testing Facility

Chaoyang Lyu, Kai Bai, Yiheng Wu, Mathieu Desbrun, Changxi Zheng, Xiaopei Liu

Building a Virtual Weakly-Compressible Wind Tunnel Testing Facility

CHAOYANG LYU, ShanghaiTech University / SIMIT / UCAS, China
KAI BAI, ShanghaiTech University / AEROCAE Digital Ltd., China
YIHENG WU, ShanghaiTech University, China
MATHIEU DESBRUN, Inria / Ecole Polytechnique, France
CHANGXI ZHENG, Tencent Pixel Lab / Columbia University, USA
XIAOPEI LIU, ShanghaiTech University, China

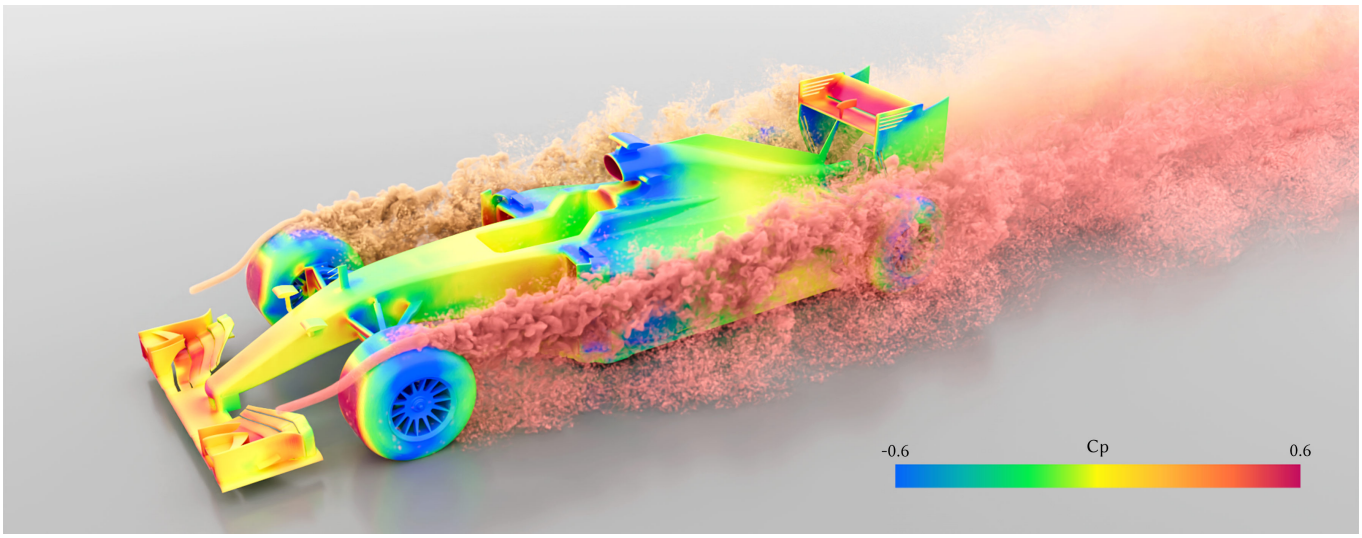


Fig. 1. **Aerodynamics of an F1 racing car.** In this paper, we construct a virtual subsonic weakly-compressible wind tunnel testing facility for the simulation of fluid flows around 3D models with which accurate physical quantity prediction (such as surface forces and drag coefficient) as well as precise flow field and surface pressure visualizations can be readily obtained. Here, the mean pressure field (C_p) is shown on the car body surface, while passively-advected dyed particles from two front sources show the wake flow behind the rotating wheels and the car body. Five-level multiresolution grids are used to capture an effective resolution of 4mm on the body of an F1 racing car measuring 4.15 meters. With our optimized GPU implementation, a 2-second simulation of such a complex model takes less than one hour to compute, with an accuracy meeting current industrial standards for automotive aerodynamics.

Virtual wind tunnel testing is a key ingredient in the engineering design process for the automotive and aeronautical industries as well as for urban planning: through visualization and analysis of the simulation data, it helps optimize lift and drag coefficients, increase peak speed, detect high pressure zones, and reduce wind noise at low cost prior to manufacturing. In this paper, we develop an efficient and accurate virtual wind tunnel system based on recent contributions from both computer graphics and computational fluid dynamics in high-performance kinetic solvers. Running on one or multiple GPUs, our massively-parallel lattice Boltzmann model meets industry standards for accuracy and consistency while exceeding current mainstream industrial solutions in terms of efficiency — especially for unsteady turbulent flow simulation at very high Reynolds number (on the order of 10^7) — due to key contributions in improved collision modeling and boundary treatment, automatic construction of multiresolution grids for complex models, as well as performance optimization. We demonstrate the efficacy and reliability of our virtual wind tunnel testing facility through comparisons of our results to multiple benchmark tests, showing an increase in both accuracy and efficiency compared to state-of-the-art industrial solutions. We also illustrate the fine turbulence structures that our system can capture, indicating the relevance of our solver for both VFX and industrial product design.

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

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

ACM Reference Format:
Chaoyang Lyu, Kai Bai, Yiheng Wu, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2023. Building a Virtual Weakly-Compressible Wind Tunnel Testing Facility. *ACM Trans. Graph.* 42, 4 (August 2023), 20 pages. https://doi.org/10.1145/3592394

Authors' addresses: C. Lyu, K. Bai, Y. Wu, 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); K. Bai is now at the Aerocae Digital Ltd., Beijing, China; M. Desbrun: Inria Saclay / LIX, Institut Polytechnique de Paris, Palaiseau, France; C. Zheng: Tencent Pixel Lab / Columbia University, New York City, USA .

1 INTRODUCTION

Nowadays, we expect cars to be sleek, streamlined and aerodynamic. Yet, early automotive models like the Ford Model T were quite boxy. It was only after aeronautical engineers started to use wind tunnels to test aircrafts (often with scale-models) that the idea of aerodynamics began to gain popularity among automotive designers in the early twentieth century. Since then, wind tunnel testing has streamlined car and airplane design for decades, helping to lower energy consumption as speed has become an increasingly important factor — leading at times to sale slogans like "shaped by the wind" for the 1955 Chrysler Ghia Gilda. Even prototypes of high-rise buildings, electricity transmission towers, or long-span bridges are often tested in wind tunnels early in their design to evaluate the aerodynamic loadings to be borne by their structural frames.

Advent of virtual wind tunnels. Due to their high costs and limited ability to provide a fine analysis of the airflow, real-life wind tunnel experiments on physical prototypes have quickly given way in the digital era to full-scale virtual aerodynamic testing — an easy, fast, and cost-effective solution to the design of products without the need for physical prototyping. When designing a new car, plane, or tall building, manufacturers now increasingly rely on numerical simulation before the initial model is constructed in order to predict the airflow around the model: identifying the regions of high pressure or strong shear offers detailed insights that save time and money in the design cycle. As a consequence, design iterations have been notably shortened by the introduction of virtual wind tunnels. Today, the wall-clock efficacy of a full-scale simulation can still deeply impact the time to market of a new automotive, architectural, or aeronautical model by accelerating the inevitable "test, refine, iterate" process [Irwin et al. 2013; Solari 2019].

Current virtual wind-tunnel testing. For the most common subsonic weakly compressible case (for Mach numbers smaller than 0.3 where fluid incompressibility remains a good approximation), current virtual wind tunnels require a time-consuming preprocessing stage: a body-fitted mesh of the model to test must be constructed, often involving a lot of manual intervention to guarantee high-quality mesh elements. Then a finite-volume or finite-element Computational Fluid Dynamics (CFD) solver for the incompressible Navier-Stokes equations is used to evaluate aerodynamic quantities such as lift/drag coefficients and pressure distributions over the model surface. Simulation time is often in the order of hours per time step to ensure accuracy on CPU even if large compute clusters are used (especially on unstructured meshes where global solves are particularly inefficient) for the Reynolds numbers needed for realistic conditions ($Re \sim 10^7$ for cars, and near 10^8 for a landing airplane). Even with recent GPU accelerations, existing software packages like Siemens' StarCCM+ [Siemens PLM Software 2023] are still mostly used for steady flow simulation, as the case of time-dependent unsteady flow with turbulence is out of practical reach.

Upending aerodynamics with LBM. In recent years, improvements in lattice Boltzmann methods (LBM) for efficient turbulent flow simulation have been considerable. Their recent progress on collision models along with their ability to explicitly and locally solving the Boltzmann transport equation on massively parallel architectures [Li et al. 2020; Lallemand et al. 2021] have now started to upend traditional CFD solvers (of note, industrial software suites like PowerFLOW and XFlow [Simulia Corp. 2023] are based on LBM, but little is known in terms of the exact methods they implement) as well as typical Computer Graphics (CG) approaches which cannot handle Reynolds numbers above a few thousands, thus limiting their realism. In terms of efficiency, GPU optimizations for LBM and extensibility to multiple GPUs [Chen et al. 2022] have outclassed the current state-of-the-art solvers for virtual wind tunnels; but recent CG works [Li et al. 2020; Lyu et al. 2021] are still not close to meet the demands — or rival the accuracy — of industrial applications.

Overview. In this paper, we argue that by addressing a few crucial shortcomings of previous LBM works in CG, we can bridge the gap between simulating fluids for visual effects and for real-world industrial applications. We show that improving the collision model in LBM and the boundary treatment near solid objects to enable Reynolds numbers of up to tens of millions, incorporating multiresolution grids with proper grid transition to resolve boundary layers more accurately, and further enhancing performance with GPU optimizations result in a *virtual subsonic weakly-compressible wind tunnel testing facility* that not only matches in accuracy and outperforms in efficiency many existing commercial CFD packages, but provides efficient and scalable high-resolution turbulent flow simulations for cinematic productions. In order to validate our novel LBM fluid solver, we present a series of validation tests, varying from the drag coefficients of the flow around a sphere for a large range of Reynolds numbers to the aerodynamic evaluation of standard car models of different shapes. We also assess our contribution in terms of efficiency and accuracy by comparing to the state-of-the-art methods and existing industrial solutions for the simulation of unsteady, turbulent flows over complex models. Finally, a series of visualizations are provided to illustrate the ability of our solver to capture very fine turbulence structures — as seen for a flow around a racing car in Fig. 1 visualized through dyed particles while coloring of the car body indicates pressure distribution.

2 BACKGROUND AND RELATED WORK

We first review related work on general fluid simulation in both CFD and CG, before discussing LBM and its recent advances that will guide our design choices to build an accurate, yet efficient virtual wind tunnel facility. Readers may check out recent review papers such as [Lallemand et al. 2021] for a more in-depth look at LBM-based simulation of nearly incompressible flows.

2.1 Navier-Stokes solvers

Classically, subsonic fluid flows at low Mach numbers are simulated by solving the incompressible Navier-Stokes (NS) equations:

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

where $\boldsymbol{u}$, ρ_0, p and $\boldsymbol{g}$ denote velocity, (constant) density, pressure and external forces respectively, while ν is the kinematic viscosity.

NS solvers in CFD. Over the years, a variety of numerical approaches have been developed in Computational Fluid Dynamics to solve the NS equations, covering finite difference [Strikwerda 2004; Shukla et al. 2007], finite volume [Eymard et al. 2000; Jasak and

Uroić 2020] and finite element [Zienkiewicz et al. 2013] methods. The *finite difference method* is relatively simple due to a Cartesian grid formulation, but cannot accurately deal with complex geometries or preserve important physical quantities [Ferziger et al. 2002]. *Finite volume methods*, on the other hand, discretize space into an unstructured mesh and leverage a conservative formulation, handling complex geometries more accurately [Tu et al. 2018]. However, this simulation accuracy requires stringent conditions on the quality of the mesh, whose construction is therefore both time consuming and labor intensive [Chawner and Taylor 2019], preventing more challenging scenarios such as dynamic coupling [Lo 2014; Zhang et al. 2020]. *Finite element methods* adopt a similar spatial discretization (with similar requirements on the mesh) but with a more versatile numerical formulation that can achieve higher-order convergence; however, their implementation is reputedly not straightforward [Zienkiewicz et al. 2013], and conservation is not guaranteed compared to a finite volume approach.

NS solvers in graphics. While CFD methods are not focusing on performance as they typically aim for accuracy, the Navier-Stokes solvers proposed in computer graphics community usually sacrifice accuracy for much higher performance and better flexibility. For instance, early work by Stam [1999] proposed a simple and unconditionally-stable semi-Lagrangian advection; but the resulting numerical dissipation did not allow strong turbulence. A series of numerical schemes were introduced to counteract this flaw, such as BFECC [Kim et al. 2005], unconditionally stable MacCormack [Selle et al. 2008], advection-reflection solver [Zehnder et al. 2018] and BiMocq^2 [Qu et al. 2019], often at the price of adding numerical dispersion for small timesteps instead. Vortex methods reformulate the NS equations with vorticity [Park and Kim 2005; Selle et al. 2005] and use vortex filaments [Angelidis and Neyret 2005; Weißmann and Pinkall 2010] or vortex sheets [Pfaff et al. 2012; Zhang and Bridson 2014; Zhang et al. 2015], which naturally preserve more vortices in the flow. While grid-based methods are more suitable for gaseous phenomena, particle-based methods such as smoothed particle hydrodynamics (SPH) [Desbrun and Gascuel 1996; Müller et al. 2003; Adams et al. 2007; Becker and Teschner 2007; Ihmsen et al. 2014b] have also been proposed for liquid simulations. To better enforce incompressibility, density-based [Solenthaler and Pajarola 2009; Bender and Koschier 2015; Ihmsen et al. 2014a] or position-based [Macklin and Müller 2013] corrections were derived to improve visual complexity. Finally, hybrid methods leveraging the benefits of both grid and particle-based approaches were developed [Harlow 1962; Brackbill and Ruppel 1986; Foster and Metaxas 1996; Zhu and Bridson 2005], with later improvements in particle-grid transition [Jiang et al. 2015; Fu et al. 2017] to enhance momentum preservation. More recently, Fei et al. [2021] reduced numerical dissipation with a separable integration scheme, while Qu et al. [2022] proposed a new particle-grid transfer kernel for enhanced particle distribution and volume preservation.

2.2 Kinetic Boltzmann solvers

Whether it is in CFD or CG, Navier-Stokes equations are difficult to simulate for unsteady turbulent flows because nonlinear advection is involved, making it hard to have accurate conservative solutions. Moreover, global linear solves are often required for incompressibility and/or stability, which slows down the whole simulation process, even if GPUs are used [Chentanez and Müller 2014; Wu et al. 2018] since fully exploiting parallelization for these solves is extremely difficult. Kinetic solvers have recently upended this long-standing issue by relying on the Boltzmann transport equation [Krüger et al. 2016] instead of the usual NS equations, typically written as

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla f + \boldsymbol{F} \cdot \nabla_{\boldsymbol{v}} f = \Omega, \tag{2}$$

where f, shorthand for $f(\boldsymbol{x}, \boldsymbol{v}, t)$, is the distribution function characterizing the probability of fluid particles moving with microscopic velocity $\boldsymbol{v}$ at position $\boldsymbol{x}$ and time t; $\boldsymbol{F}$ is the external force density such as gravity; and Ω is the collision operator that relaxes the distribution function towards its local equilibrium, expressed to first order as $\Omega = -(f - f^{\mathrm{eq}})/\tau$ (BGK form) where $\tau = 3\nu + 1/2$ is the relaxation time determined by the kinematic viscosity ν [Dellar 2001]. By discretizing Eq. (2) via second-order space-time finite differences and using Hermite polynomial expansion for the distribution function, we obtain the lattice Boltzmann equations (LBE):

$$f_i(\boldsymbol{x}_k + \boldsymbol{c}_i, t+1) - f_i(\boldsymbol{x}_k, t) = \Omega_i + F_i, \tag{3}$$

where $\boldsymbol{x}_k$ is a grid node, $\boldsymbol{c}_i$ is the discretized microscopic "lattice" velocities, and F_i is the external forcing term projected into lattice space; Ω_i, the discretized collision, can be as simple as $-(f_i - f_i^{\mathrm{eq}})/\tau$ for the lattice BGK collision model [Chen and Doolen 1998]. Through operator splitting, Eq. (3) can be rewritten as a streaming process $f_i^*(\boldsymbol{x}, t) = f_i(\boldsymbol{x} - \boldsymbol{c}_i, t)$ followed by a collision process and force injection $f_i(\boldsymbol{x}, t+1) = f_i^* + \Omega_i + F_i$, leading to an update that is explicit and localized, making it very suitable for high-efficiency GPU-based parallel implementation. Typical macroscopic fluid quantities such as density, velocity, or pressure can be recovered simply through moments of the distribution functions at any time step:

$$\rho = \sum_i f_i, \quad \rho \boldsymbol{u} = \sum_i \boldsymbol{c}_i f_i, \quad p = c_s^2 \rho, \tag{4}$$

where $c_s = 1/\sqrt{3}$ is the normalized speed of sound in lattice unit. Note that in LBM, spatial and temporal steps are all normalized to one, and the other quantities are scaled accordingly. The lattice Boltzmann equations generate physical quantities that satisfy the weakly-compressible Navier-Stokes equations [Nie et al. 2008].

Collision modeling. The simplistic lattice BGK collision model is numerically very unstable due to significant truncation errors for high Reynolds number flows (i.e., very small viscosity), which led to a quest for improved collision treatments to enhance both accuracy and stability of turbulent flow simulation with LBM using a typical floating-point format (e.g., 32 bits). The multiple relaxation time (MRT) collision model [D'Humières 1992; Lallemand and Luo 2000; D'Humières et al. 2002], on the other hand, performs relaxation in moment space, which improves stability and accuracy dramatically. While initially constructed with only raw moments (and thus, violating Galilean invariance), the introduction of central moments [Geier et al. 2006, 2009], Hermite moments [Shan and Chen 2007; Chen et al. 2014; Adhikari and Succi 2008], central Hermite moments [Mattila et al. 2017; Shan 2019] and cumulants [Geier et al. 2015, 2017a] have gradually removed this limitation. Other collision approaches based on truncation of the Hermite polynomial expansion, referred to as regularized collision models, were

also proposed [Zhang et al. 2006; Lätt and Chopard 2006] to filter out non-hydrodynamic spurious modes. This model was later improved via a recursive formulation [Malaspinas 2015; Coreixas et al. 2017], as well as a hybrid recursive formulation [Jacob et al. 2018]. Another approach to improve collision modeling is to incorporate dynamic relaxation rates, primarily by utilizing subgrid models of large-eddy simulations (LES) [Eggels 1996; Sagaut 2010] or entropic collision models [Karlin et al. 1999; Ansumali et al. 2003]. For LES subgrid models, the Smagorinsky model was first used in LBM [Hou et al. 1994; Krafczyk et al. 2003], and later extended to include dynamic parameter estimation [Premnath et al. 2009], a van Driest damping function [Malaspinas and Sagaut 2014], or an improved shear term [Lévêque et al. 2007], for more precise descriptions of wall boundary flows. Besides models using eddy viscosity, there are also models based on approximate deconvolution [Malaspinas and Sagaut 2011; Nathen et al. 2018] or on the wall-adaptive large eddy (WALE) model for LBM [Weickert et al. 2010]. Concurrently, entropic collision models provide a useful constraint on the relaxation times of high-order moments (inherently lacking physical interpretation) by maximizing the local entropy as dictated by the second law of thermodynamics. However, solving a constrained entropy maximization numerically is expensive. Consequently, more efficient methods have been developed based on the KBC model [Karlin et al. 2014], pseudoentropy [Krämer et al. 2019] or using asymptotic solutions from perturbation theory [Tang et al. 2022]. A last alternative to subgrid and entropic models is through optimization [Li et al. 2020] with an offline regression of relaxation rates.

Streaming algorithms. A potential bottleneck in LBM simulation is data dependency of the distribution functions, which hurts parallelism. Typically, two sets of distribution functions are used to avoid conflicts during streaming updates, but this simple fix is memory-demanding. A series of streaming have been devised to only read and write a single set of distribution functions, from the AA-Pattern of Bailey et al. [2009], to Shift-and-Swap [Mohrhard et al. 2019], Periodic-shift [Kummerländer et al. 2023], Esoteric Twist [Geier and Schönherr 2017], and even Esoteric Pull and Push [Lehmann 2022] to further reduce peak memory bandwidth. While we do not employ the most advanced LBM streaming algorithms in this paper, the aforementioned methods could be combined with our method to further improve efficiency.

Boundary treatment. Besides collision modeling, another important topic in LBM is the treatment of boundaries [Marson 2022]. The most widely-used approaches to enforce no-slip conditions are through either the bounce-back scheme [Ladd 1994; Bouzidi et al. 2001; Ginzburg and D'Humières 2003; Chun and Ladd 2007] or the immersed boundary method (IBM) [Peskin 1972; Lu et al. 2012; Kang and Hassan 2011; Patel and Natarajan 2018; Seo and Mittal 2011; Chen et al. 2013]. The simple bounce-back (SBB) scheme is often preferred due to its simplicity of implementation, but its voxelized treatment of boundaries makes it only first-order accurate in general. To remedy this issue, a common approach (called interpolated bounce-back or IBB [Bouzidi et al. 2001; Yu et al. 2003]) is to interpolate the distribution functions according to the actual intersection between a lattice link and the boundary. Ginzburg and d'Humières [2003] achieved higher-order accuracy for curved boundaries through a multi-reflection boundary treatment. A recent hybrid method combining bounce-back and immersed boundary treatment also reduces staircase artifacts using a boundary force correction [Lyu et al. 2021]. While these methods are effective for complex geometries, they often introduce non-local operations involving distribution functions from more than one-hop neighboring nodes; this can create issues for narrow gaps with no neighboring nodes, or for thin boundary layers, requiring higher grid resolution and thus more memory. This drawback can be avoided by defining an interpolated distribution function from the boundary velocity [Chun and Ladd 2007], or through single-node schemes [Zhao and Yong 2017; Geier et al. 2015; Tao et al. 2018] and their generalizations [Zhao et al. 2019; Chen et al. 2021; Marson et al. 2021].

Grid refinement. When accuracy and efficiency are of paramount importance, grid refinement near object boundaries and wake flow regions is inevitable to offer focused precision with minimal overhead [Lagrava 2012]. Grid refinement methods in LBM can be categorized in two main families: node-based vs. cell-based methods. Node-based methods, pioneered by [Filippova and Hänel 1998] and later improved by [Dupuis and Chopard 2003], store all the computed quantities (including distribution functions and macroscopic values) on the corner nodes of the lattice cells of various resolutions and allow fine-to-coarse transfers through rescaling. Spatial filtering of distribution functions on fine grid nodes can also be applied to obtain better distribution functions on coarser grids with reduced discontinuity of macroscopic quantities [Lagrava et al. 2012], increasing stability for flows with high Reynolds numbers. Cell-based methods [Rohde et al. 2006; Chen et al. 2006] store all quantities at the center of grid cells instead. This family of methods can achieve conservation of mass and momentum without any rescaling of distribution functions [Schornbaum and Rüde 2016; Hasert et al. 2014; Lätt et al. 2021]. To allow smoother transition across multiresolution grids, compact interpolation [Goraki Fard 2015], directional splitting [Gendre et al. 2017] and direct coupling [Astoul et al. 2021] have been proposed for applications requiring high accuracy such as aeroacoustics. Nevertheless, all these grid refinement methods typically enforce that each fine grid is a subdivision of a coarser grid, reducing the flexibility with which grids of different resolutions are positioned — except for [Li et al. 2019] whose continuous-scale multi-block formulation relaxes the grid alignment constraints.

2.3 Requirements for industrial applications

Subsonic nearly-incompressible fluid flow solvers that are used to reduce time to market for new automotive or aviation designs are not significantly different from CG solvers producing big, realistic fluid flow simulations in the VFX industry: they also strive for efficiency first and foremost, and would rather not spend too much preprocessing time on mesh cleaning, construction and tuning. They mainly differ, however, on their accuracy requirements.

Efficiency. Wall-clock simulation time of fluid flows is crucial to accelerate design cycles. Many commercial solvers are still executed on CPU clusters, requiring a huge (and expensive) processing power to accomplish a simulation in an acceptable time: it can take hours to simulate one time step for an accurate NS solver, and several days for a short time sequence, rendering unsteady flow simulation

with turbulence impractical. Even if steady-state solutions are often used instead, their accuracy largely depends on the choice of turbulence models (such as the Reynolds-Averaged Navier-Stokes (RANS) models [Alfonsi 2009] for simulations at high Reynolds numbers), whose results deviate from the actual physical measurements for what is intrinsically an unsteady flow — making the prediction of physical quantities sometimes unreliable. Therefore, efficiency in the simulation of unsteady flows running on small computational devices to reliably guide design is an industrial must.

Preprocessing simplicity. Traditional solvers are mainly finite-volume based, and thus, they require the construction of unstructured body-fitted meshes over the simulation domain before proceeding. This preprocessing requirement is typically very time consuming as it involves tedious manual mesh adjustment, often by trial and error. This is another source of significant delays in the iterative design of a new model: each time the shape is altered based on the feedback obtained from its simulation test, another round of preprocessing is required before the shape is computationally evaluated again. It is thus important that meshes be automatically constructed with as few, short manual interventions as possible, so that preparation time for a simulation can be kept to a minimum.

Accuracy. Finally, a virtual wind tunnel should be predictive enough so as to supersede an actual wind tunnel test when needed. Besides identifying the regions of high pressure, strong shear, or other fine details of the flow, a virtual simulation must be able to evaluate common measures in industrial design to quantify key performance characteristics — a famous example is the *drag coefficient*, a dimensionless value (commonly denoted as C_d and computed as the ratio of the drag force to the force produced by the dynamic pressure times the projected frontal area of the model) that quantifies the air resistance of a shape, with a small coefficient indicating that it will move more efficiently through air. An accepted range of error for this drag coefficient prediction is within 5% (sometimes even 3%) of an actual wind tunnel evaluation on a series of known benchmarks including cars of various sizes and shapes. Given the importance of boundary layers on the evaluation of forces on the model for very high Reynolds numbers (typically in the order of 10^6 to 10^7), such an accuracy is out of reach for NS solvers in graphics, even for recent CG works on LBM, e.g., [Li et al. 2020] and [Lyu et al. 2021] that use single-resolution grid, lower-order collision models, and boundary treatments for efficiency: it calls for higher-order LBM models with adaptive or multiresolution grids to capture down to a few millimeters of geometric resolution to reach this level of accuracy for prediction of relevant physical quantities.

2.4 Our contributions

In this paper, we construct a virtual subsonic weakly-compressible wind tunnel testing facility which can be used for aerodynamic design in automotive and aeronautical industries and for high-resolution, realistic fluid flows for VFX. Given the requirements we just discussed, we base our approach on the lattice Boltzmann method as it allows for massively-parallel computational runs. Compared to state-of-the-art LBM techniques proposed in both CFD and CG, we contribute a number of improvements to enhance accuracy and accelerate computations:

- we employ a cumulant-based collision model that we improve through local entropy maximization to gain higher accuracy and less force fluctuations; this new collision model can simulate turbulent flows with very high Reynolds numbers of up to 10^8;
- we introduce a novel single-node interpolated bounce-back scheme for static and dynamic boundary treatment which improves boundary force evaluation;
- we propose an automatic and flexible multiresolution grid construction with node propagation to enable both external and internal flows over very complex prototypes, allowing high-resolution, detailed flow simulations at low computational cost while avoiding tedious mesh preprocessing;
- we also contribute various optimizations to speed up execution on both single and multiple GPUs, offering high-accuracy predictions within hours, instead of days.

3 VIRTUAL WIND TUNNEL

We now delve into our contributions by providing a brief overview of the literature we build upon, before detailing our changes to ensure improvements in efficiency and accuracy. In order to keep our exposition short and focused on our specific improvements, we assume that the reader is aware of the traditional LBM pipeline, i.e., solving the Boltzmann transport equations through operator splitting by alternating streaming, boundary handling, and collision; for a short overview, one may refer to [Li et al. 2020].

3.1 Entropy-based cumulant collision model

Existing LBM methods for turbulent flow simulation in graphics rely on central moment collision models with adaptive high-order relaxation rates [Li et al. 2020; Lyu et al. 2021]. These models are appropriate for visual effects as they generate physically consistent turbulence for many natural phenomena. However, it still lacks sufficient accuracy for flows with very high Reynolds numbers required in industrial applications. To remedy this limitation, we employ a *cumulant-based collision model* that we further improve in order to enable higher numerical precision and reliability. Cumulant collision models fall into the category of multiple relaxation time (MRT) collision models as reviewed in Sec. 2.2: they proceed by transforming the distribution functions f_i first into cumulant (also called cumulative-moment) space, then perform collision by relaxing each cumulant towards its equilibrium with an *individual* rate, before finally transforming the post-collision results back into new distribution functions.

Cumulants vs. central moments. Compared to recent kinetic solvers in CG which are mostly based on the non-orthogonal central moment collision model, cumulants can be understood as intensive and statistically-independent quantities while central moments are extensive and interdependent; e.g., the first-order moments correspond to the macroscopic momentum $\rho\boldsymbol{u}$, while the first-order cumulants correspond to the velocity $\boldsymbol{u}$. Their independence (and Galilean invariance) implies that the use of *individual relaxation rates* for cumulants can further improve accuracy numerically [Geier et al. 2015]. More formally, cumulants are defined as partial derivatives in frequency space of the logarithm of the Laplace transform of the

distribution function, i.e.,

$$k_{\alpha\beta\gamma} = \frac{\partial^\alpha \partial^\beta \partial^\gamma}{\partial K_x^\alpha \, \partial K_y^\beta \, \partial K_z^\gamma} \log \mathcal{L}\{f(\boldsymbol{v})\}(\boldsymbol{\zeta})\Big|_{\zeta_x=\zeta_y=\zeta_z=0},$$

where $\boldsymbol{v}$ is the microscopic particle velocity while $\boldsymbol{\zeta} = (\zeta_x, \zeta_y, \zeta_z)$ encodes frequency (wavenumber). A cumulant $k_{\alpha\beta\gamma}$ is said to be of order n if $\alpha+\beta+\gamma = n$. Note that cumulants are computed from central moments in practice [Geier et al. 2015]. Once cumulants are evaluated, cumulant-based collision modeling simply relaxes each cumulant $k_{\alpha\beta\gamma}$ towards its Maxwellian equilibrium $k^{\text{eq}}_{\alpha\beta\gamma}$ with its own relaxation rate $s_{\alpha\beta\gamma}$ through:

$$k^*_{\alpha\beta\gamma} = s_{\alpha\beta\gamma}\, k^{\text{eq}}_{\alpha\beta\gamma} + (1 - s_{\alpha\beta\gamma})\, k_{\alpha\beta\gamma}. \tag{5}$$

As in all MRT models, the relaxation rates of low-order cumulants are determined by their physical interpretations so as to enforce proper convergence to the Navier-Stokes equations: the zeroth-order cumulants must use a zero rate to conserve density; the first-order ones must be unit to preserve momentum, and the second-order cumulants, corresponding to momentum fluxes, must be determined by the kinematic viscosity of the fluid. While the third-order cumulants can be assigned optimal rates based on Taylor series analysis as proposed in [Geier et al. 2017a], rates for higher-order cumulants are often set to an arbitrary constant (typically, 1) by lack of clear guidance on how to optimize their values. Once all the relaxed rates $k^*_{\alpha\beta\gamma}$ are computed as described, they are converted back to the updated distribution functions to complete the whole collision. Due to Galilean invariance and better statistical independence, cumulant-based collision models often exhibit higher accuracy and stability for flow simulation with strong turbulence [Geier et al. 2015].

Optimizing higher-order relaxation rates. While the cumulant model described above can equal or outperform existing collision models, optimizing higher-order rates is known to further improve the accuracy of results, particularly for turbulent flow simulations: Li et al. [2020] showed in the case of central-moment collision modeling that adapting the rates based on local macroscopic quantities of the fluid via regression greatly reduced numerical diffusion and dispersion, enhancing simulation accuracy. Thus, one can also hope to further enhance the cumulant-based collision accuracy through an online optimization of the higher-order rates — although the ad-hoc approach of [Li et al. 2020] did not help in our tests. However, since the local equilibrium distribution functions f_i^{eq} are the maximizer (under the constraint of conserved density and momentum density) of the entropy $H(f_i)$ defined as

$$H(f_i) = -\sum_i f_i \log\left(\frac{f_i}{\omega_i}\right), \tag{6}$$

where ω_i is the lattice weight associated with each discretized distribution function, it seems reasonable to expect that optimizing directly the higher-order rates for cumulants such that they maximize entropy is bound to further improve accuracy compared to picking arbitrary rates: this is in fact precisely what Krämer et al. [2019] successfully achieved in the case of central-moment collision modeling. We must, however, adapt their approach to the case of cumulants: while they exploit the linear transform $\boldsymbol{T}$ between central moments $\boldsymbol{m} = \{m_i\}_i$ and distribution functions $\boldsymbol{f} = \{f_i\}_i = \boldsymbol{T}\boldsymbol{m}$, this is no longer true for (non-linear) cumulants. To make the maximization easier, we use a concave, quadratic approximation $\tilde{H}$ of H, called pseudo-entropy in [Krämer et al. 2019] and expressed as:

$$\tilde{H}(\boldsymbol{f}) = -\sum_i \left(\frac{f_i^2}{\omega_i} - f_i\right) = \rho - \sum_i \frac{f_i^2}{\omega_i}\,. \tag{7}$$

Given that cumulants are non-linear transformation of moments of $\boldsymbol{f}$, finding in closed form the higher-order relaxation rates that maximize this simpler entropy still seems impossible. However, a closer inspection of the expressions linking cumulants $k_{\alpha\beta\gamma}$ and central moments $m_{\alpha\beta\gamma}$ given in [Geier et al. 2017a] reveals a key property that we can exploit: for all orders $p \leq 3$, cumulants and central moments exactly match (Eqs. (16)-(19) in [Geier et al. 2017a]), while each higher-order cumulant is the sum of its corresponding central moment and a linear combination of other high-order cumulants, plus a non-linear function of low-order cumulants (Eqs. (20)-(23) in [Geier et al. 2017a]). Since the relaxed cumulants up to order 2 are updated with physically-imposed relaxation rates to ensure fluid density and momentum conservation and that the cumulants of order 3 have known optimized relaxation rates as well [Geier et al. 2017a], we can leverage the properties mentioned above to find an *analytical* solution to the entropy maximization. Indeed, if we denote by $r_{\alpha\beta\gamma} = m_{\alpha\beta\gamma} - k_{\alpha\beta\gamma}$ the difference between a central moment and its associated cumulant, and store all these differences for orders higher than 3 into a vector $\boldsymbol{r}$, then there exists a constant matrix $\underline{\boldsymbol{L}}$ known in closed form and a vector $\underline{\boldsymbol{n}}$ depending only on the known low-order cumulants such that:

$$\boldsymbol{r} = \underline{\boldsymbol{L}}\,\boldsymbol{k} + \underline{\boldsymbol{n}}\,, \tag{8}$$

where $\boldsymbol{k}$ denotes the vector representing the cumulants of orders higher than 3. Moreover, the condition of optimality

$$\frac{\partial \tilde{H}(\boldsymbol{f})}{\partial k_{\alpha\beta\gamma}} = -\sum_i \frac{\partial f_i}{\partial k_{\alpha\beta\gamma}} \frac{2 f_i}{\omega_i} = 0 \quad \forall \alpha+\beta+\gamma \geq 4 \tag{9}$$

for the pseudo-entropy of Eq. (7) can then be written explicitly using the matrices $\boldsymbol{T}$ and $\underline{\boldsymbol{L}}$: if one expresses $\boldsymbol{T}$ as $\boldsymbol{T} = [\boldsymbol{T}_{\text{l}}; \boldsymbol{T}_{\text{h}}]$ where $\boldsymbol{T}_{\text{l}}$ represents the first ten columns of $\boldsymbol{T}$ and $\boldsymbol{T}_{\text{h}}$ represents the rest of the columns, one then finds the optimal higher-order cumulants $\boldsymbol{k}$ that maximize the pseudo-entropy given the vector $\boldsymbol{k}_{\text{l}}$ of the first ten (known) cumulants through:

$$(\boldsymbol{I} + \underline{\boldsymbol{L}})\boldsymbol{k} = -(\boldsymbol{D}^T \boldsymbol{W}^{-1} \boldsymbol{T}_{\text{h}})^{-1} \boldsymbol{D}^T \boldsymbol{W}^{-1} \boldsymbol{T}_{\text{l}} \boldsymbol{k}_{\text{l}} - \underline{\boldsymbol{n}}\,, \tag{10}$$

where $\boldsymbol{I}$ is the identity matrix; $\boldsymbol{D} = \boldsymbol{T}_{\text{h}}(\boldsymbol{I} + \underline{\boldsymbol{L}})$ is the matrix of partial derivatives of $\boldsymbol{f}$ w.r.t higher-order cumulants $\boldsymbol{k}$, and $\boldsymbol{W}$ is the diagonal matrix containing the lattice weights ω_i. Therefore, for the typical D3Q27 lattice structure, finding the optimal high-order cumulants $\boldsymbol{k}$ consists in a simple linear solve which we compute analytically for efficiency. To avoid numerical drift, cumulants are clamped between equilibrium and pre-collision values. These final cumulants are then turned back into distribution functions [Geier et al. 2017a] to complete the collision process.

Comparison and analysis. In order to validate the resulting accuracy of our collision model for turbulent flow simulation, we conducted an experiment consisting in evaluating drag coefficients C_{d} at high Reynolds numbers to show how the cumulant model with our entropy-optimized higher-order relaxation rates behaves compared to the existing cumulant model from [Geier et al. 2017a]

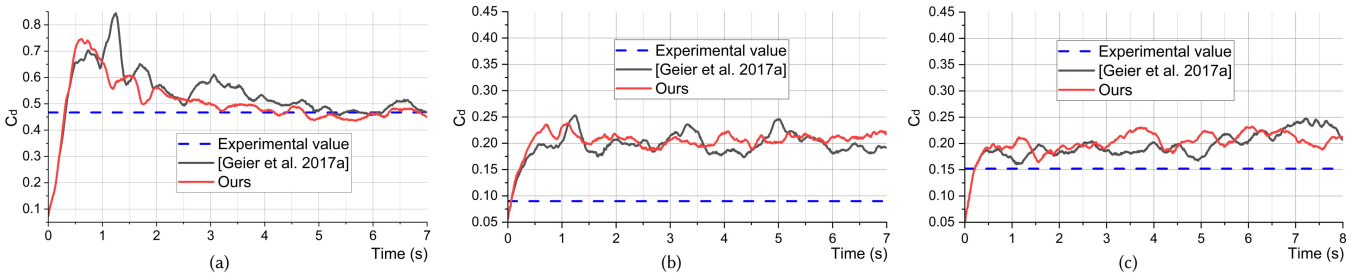


Fig. 2. **Comparing drag for various cumulant models.** We plot drag coefficient C_d over time for a sphere in a flow using the cumulant model of [Geier et al. 2017a] (black) vs. our entropy-based cumulant model (red) for Re=100,000 (a), Re=400,000 (b) at which drag crisis happens, and Re=800,000 (c) indicating better performance of our model as evidenced by faster convergence and smaller fluctuations — which both imply a more reliable mean drag estimation.

using optimized third-order relaxation rates only. We simulate a flow passing around a sphere at a relatively low Reynolds number (Re = 100, 000) and two relatively high Reynolds numbers (Re = 400, 000 and Re = 800, 000). The case at Re = 400, 000 is where *drag crisis* (also known as "Eiffel paradox") happens. It is well-documented that the drag coefficient of a sphere will change rapidly from about 0.47 at Re = 100, 000 down to around 0.1 at Re = 400, 000 when the Reynolds number increases within the corresponding short range [Tiwari et al. 2020]. This atypical regime is difficult for many numerical solvers, and Geier et al. [2017b] recently offered the first LBM-based numerical validation. We thus compared our result to theirs at the different Reynolds numbers mentioned above and plotted the curve of C_d over time in Fig. 2, while we estimate the effective drag coefficient C_d as the mean value of the curve once it stabilizes exactly as was done in [Geier et al. 2017b] for fairness of comparison. All experiment values are taken from [Barati et al. 2014]. Fig. 2 (a) demonstrates that our solver enters a converged state faster and produces a mean value closer to the reference than the one from [Geier et al. 2017a], indicating higher accuracy in physical quantity estimation for relatively low Reynolds numbers. For higher Reynolds values where drag crisis happens, both solvers cannot predict C_d values very accurately as shown in Fig. 2 (b), but this regime is often considered beyond the realm of direct solvers [Tiwari et al. 2020] anyway. However, at such a regime, many small scale vortices are generated near the boundary, so drag fluctuations are expected to also decrease [Deshpande et al. 2017]. Given that our model exhibits reduced fluctuations in time compared to [Geier et al. 2017a], we argue that even at this regime, our entropy-driven collision model brings better predictability of physical quantities. Fig. 2 (c), indicates convergence of our new collision at a early stage followed by a periodic fluctuation, while the large, irregular fluctuations of the previous method does not provide a meaningful confidence region for estimating the mean drag. Note that our use of the adaptive relaxation rates for third-order moments given in [Geier et al. 2017a] along with our entropy-based optimization for higher-order relaxation rates could be conceived as a form of sub-grid modeling, explaining why our lattice Boltzmann simulation without turbulence modeling can capture drag crisis better than other NS-based direct numerical solvers at similar grid resolutions. Nevertheless, a *real* turbulence model, such as k-ω model [Menter 1994], can be added to our solver to better resolve sub-grid flows with higher accuracy — but at the cost of significantly increased

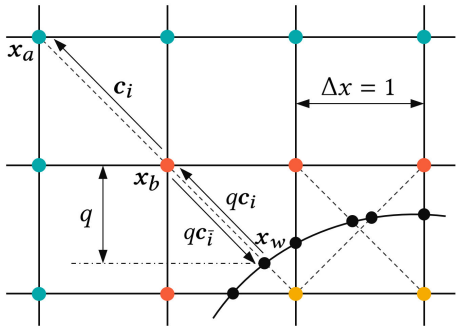


Fig. 3. **Boundary treatment near solid object.** For "cut-cell" boundary nodes marked in orange, their unknown distribution functions must be determined through boundary schemes instead of the regular streaming. Yellow nodes mark nodes inside the solid while cyan nodes are inner fluids nodes. Our boundary treatment for a node $\boldsymbol{x}_b$ uses the normalized distance q to the boundary surface along a link direction of $\boldsymbol{c}_i$ with its inverse direction denoted as $\boldsymbol{c}_{\bar{i}}$ to improve the approach of [Tao et al. 2018].

computing time. Instead, using our online entropy-based optimization does *not* imply a significant reduction of efficiency: compared to [Geier et al. 2017a], our adaptive relaxation is only around 10% slower, thus slowing down the whole simulation by less than 5%.

3.2 Boundary treatment and surface force evaluation

In the context of designing a virtual wind tunnel, we need to address three types of boundary conditions: static boundaries (e.g., a fixed car model), fluid-solid coupling where the obstacle undergoes limited rigid transformations (e.g., wheels rotating), and computational domain boundaries (which should not dramatically impact the numerical evaluation of physical quantities over the model surface). The accurate computations of forces induced by these three cases are needed to obtain a detailed flow over the model and to evaluate the required physical quantities such as the drag coefficient C_d.

Static boundary treatment. A typical virtual wind tunnel embeds a model of a solid object within the computational domain in order to perform aerodynamic testing. In the case of our LBM flow solver, the normal streaming process assumes a non-obstructed neighborhood; when an object is present within the domain, we need to alter the distribution functions on nodes adjacent to the object boundary to account for the interaction of the fluid with the solid — see Fig. 3 where the orange nodes are fluid nodes that require boundary treatment while the yellow nodes are inside the solid body. Among

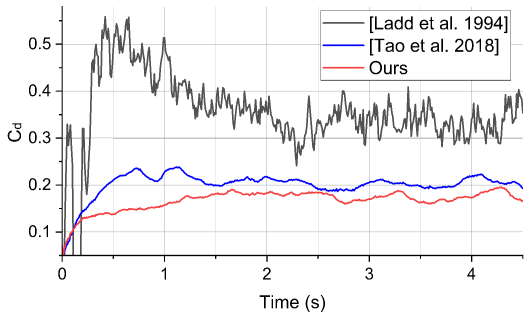


Fig. 4. **Comparing boundary treatments.** We plot the variation over time of drag coefficient of a sphere at $Re = 400,000$, for different boundary treatments but with the same entropy-optimized cumulant model. The experimental value is near $C_d = 0.1$ for this drag crisis case; a simple bounce-back, still used in many LBM implementations, leads to unacceptable results, producing large prediction errors and wide force fluctuations.

several boundary treatments in LBM, bounce-back [Ladd 1994] is a well-established scheme which simply reverts any streaming of a distribution function along a link that intersects the surface of the model boundary. However, the simplicity of this method implies a limited first-order accuracy when applied to curved boundary, leading to a much larger error with stronger-than-expected force fluctuations in time as shown in the black curve in Fig. 4 for the C_d temporal variation of a flow passing over a sphere at a high Reynolds number. Interpolated bounce-back (IBB [Bouzidi et al. 2001]) and its later variants [Yu et al. 2003; Ginzburg and D'Humières 2003; Chun and Ladd 2007], on the other hand, offer a second- or third-order treatment of curved boundary by considering a larger neighborhood and taking into account the distance to the boundary surface along a link. While this family of techniques improves accuracy, boundary handling becomes less local, both impairing parallelism and preventing the treatment of narrow gaps between obstacles which regularly happen in practical design engineering simulations; moreover, it usually fails to better resolve thin boundary layers occurring in high Reynolds number flows unless a fine grid resolution is used. More recently, Tao et al. [2018] proposed a single-node approach by employing approximations of distribution functions at boundary points on links. More specifically, for a node $\boldsymbol{x}_b$ near the solid boundary whose link $\boldsymbol{c}_i$ points to node $\boldsymbol{x}_a$ and whose opposite link $\boldsymbol{c}_{\bar{i}}$ intersects the boundary at $\boldsymbol{x}_w$ as depicted in Fig. 3, the authors noted that by linear interpolation, one should have:

$$f_i(\boldsymbol{x}_b, t+1) = \frac{1}{1+q} f_i(\boldsymbol{x}_w, t+1) + \frac{q}{1+q} f_i(\boldsymbol{x}_a, t+1)\,, \qquad (11)$$

where $q = \|\boldsymbol{x}_b - \boldsymbol{x}_w\| / \|\boldsymbol{c}_i\|$ is the normalized distance of the boundary node to the object (i.e., the distance divided by the length of the link). However, the value of the distribution function $f_i(\boldsymbol{x}_a, t+1)$ was directly streamed from node $\boldsymbol{x}_b$ at the previous time step, thus the above equation is actually a single-node interpolation. Moreover, the unknown value $f_i(\boldsymbol{x}_w, t+1)$ was argued to be well approximated by summing the equilibrium $f_i^{\mathrm{eq}}(\boldsymbol{u}_w(t), \rho_b(t))$ (with $\boldsymbol{u}_w(t)$ being the current boundary velocity) to which the non-equilibrium part

$$f_i^{\mathrm{neq}}(\boldsymbol{x}_b, t) = f_i(\boldsymbol{x}_b, t) - f_i^{\mathrm{eq}}(\boldsymbol{u}_b, t) \qquad (12)$$

of the distribution function at $\boldsymbol{x}_b$ is simply added back — resulting in a simple single-node computation for boundary handling. However, their suggested evaluations of equilibrium and non-equilibrium distributions are not accurate enough for high Reynolds numbers as they are derived from a series of low-order approximations. To further improve accuracy, we change the way equilibrium and non-equilibrium parts are approximated. First, we leverage the fact that non-equilibrium parts are typically an order higher than equilibrium parts [Chun and Ladd 2007], which means that we can simply employ the bounce-back of the non-equilibrium part of the distribution function at boundary node $\boldsymbol{x}_b$ to obtain a second-order approximation of the non-equilibrium part of the unknown distribution functions via

$$f_i^{\mathrm{neq}}(\boldsymbol{x}_b, t+1) = f_{\bar{i}}^{\mathrm{neq}}(\boldsymbol{x}_b, t). \qquad (13)$$

Note that this approximation is more reliable, in particular for large gradients near boundary layers, than other possible interpolation-based methods which typically introduce smoothing. For the equilibrium part, we can also employ the same second-order accurate single-node interpolation as done in [Tao et al. 2018] by writing:

$$f_i^{\mathrm{eq}}(\boldsymbol{x}_b, t+1) = \frac{1}{1+q} f_i^{\mathrm{eq}}(\boldsymbol{x}_w, t+1) + \frac{q}{1+q} f_i^{\mathrm{eq}}(\boldsymbol{x}_a, t+1). \qquad (14)$$

While $f_i^{\mathrm{eq}}(\boldsymbol{x}_w, t+1)$ is well approximated as $f_i^{\mathrm{eq}}(\boldsymbol{u}_w(t), \rho_b(t))$ because of density continuity, we still need to find how to approximate $f_i^{\mathrm{eq}}(\boldsymbol{x}_a, t+1)$. We initially tried to reconstruct the equilibrium distribution based on the two macroscopic quantities $\boldsymbol{u}(\boldsymbol{x}_a, t)$ and $\rho(\boldsymbol{x}_a, t)$; however, the use of t instead of $t+1$ and the Hermite series truncation implied by the equilibrium distribution reconstruction negatively influence the accuracy of the boundary layer for high Reynolds number flow simulations. Instead, we found more judicious to leverage the fact that one has, very close to a solid boundary in a nearly incompressible flows, $f_i^{\mathrm{eq}}(\boldsymbol{x}_a, t+1) \approx f_i(\boldsymbol{x}_a, t+1)$ as no series truncation is involved, and this latter expression can be directly streamed from the post-collision of distribution functions at $\boldsymbol{x}_b$. Therefore, we obtain:

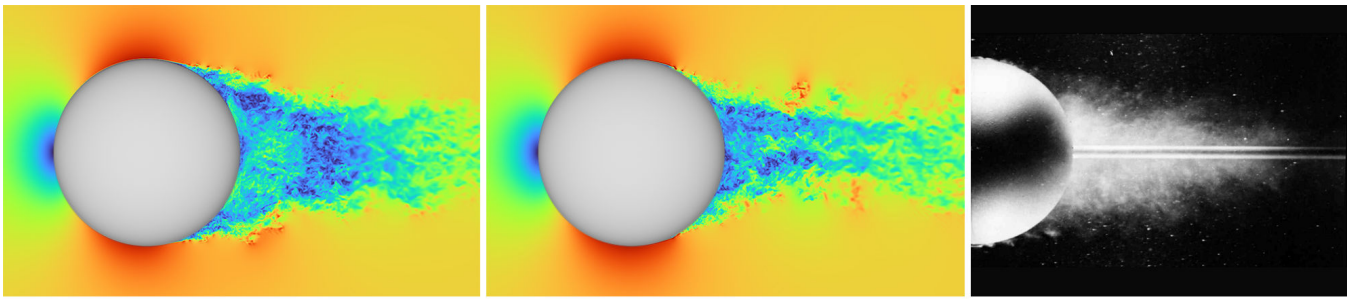
$$f_i^{\mathrm{eq}}(\boldsymbol{x}_a, t+1) \approx f_i^{*}(\boldsymbol{x}_b, t). \qquad (15)$$

Combining Eqs. (13) and (15), the unknown distribution function $f_i(\boldsymbol{x}_b, t+1)$ is now evaluated as:

$$\begin{aligned} f_i(\boldsymbol{x}_b, t+1) &= f_i^{\mathrm{eq}}(\boldsymbol{x}_b, t+1) + f_i^{\mathrm{neq}}(\boldsymbol{x}_b, t+1) \qquad (16)\\ &= \frac{1}{1+q} f_i^{\mathrm{eq}}(\boldsymbol{u}_w(t), \rho_b(t)) + \frac{q}{1+q} f_i^{*}(\boldsymbol{x}_b, t) + f_{\bar{i}}^{\mathrm{neq}}(\boldsymbol{x}_b, t). \end{aligned}$$

To validate the improved accuracy of our new boundary treatment, we again turn to the challenging case of the drag crisis of the flow passing around a sphere at a Reynolds number of 400,000. As Fig. 5 demonstrates, the flow velocity near the solid boundary and the boundary layer separation point are better captured by our method in Fig. 5 (b) when compared to an effectively equivalent experimental visualization shown in Fig. 5 (c). Moreover, the C_d curve over time in Fig. 4 better matches the real wind tunnel C_d estimation, and with smaller fluctuations. This confirms that, for the same grid, our improvement over [Tao et al. 2018] allows for even smaller vortices to be generated near the solid boundary, leading to a better converging shape of the wake flow and thus, more accurate simulations of turbulent flows at high Reynolds numbers.

Dynamic coupling. A virtual wind tunnel also needs to accommodate limited dynamic one-way coupling, such as the rotating



(a) (b) (c)

Fig. 5. **Wake of sphere in drag crisis.** The flow passing through a sphere at $Re = 400,000$ is known to exhibit in real-life experiment a narrower turbulent wake than at lower Reynolds numbers [Van Dyke 1982] (c) due to smaller scale vortices generated near the solid boundary, thus reducing drag significantly — a paradox explained by the boundary-layer theory. Compared to [Tao et al. 2018] (a), our improved boundary treatment scheme at the same Reynolds number (b) captures the separation point farther around the back of the sphere with a narrower wake, better matching the experimental visualization.

wheels of a car. Previous LBM methods adopted either a bounce-back based approach with node refilling [Tao et al. 2016], or the immersed boundary method [Li et al. 2016, 2020]. While a bounce-back scheme with node-refilling may create spurious velocities around object boundaries for high Reynolds number flows, the immersed boundary method is usually much easier to implement but is only first-order accurate, requiring higher resolutions and thus more memory storage. A recent hybrid approach [Lyu et al. 2021] has also been formulated to offer efficiency, but is not advisable for accurate simulation of very high Reynolds number flow simulations due to the thicker boundary layer it produces. In order to preserve higher accuracy for thin boundary layer flows with dynamic fluid-solid coupling, we decided to rely on the approach proposed above for static boundary treatment, where now the boundary velocity $\boldsymbol{u}_w$ is involved at each intersected link. However, we apply such a new boundary treatment in an *immersed* manner to avoid node refilling; that is, fluid grids are built assuming no solid objects, and all the nodes of a cut cell use this same boundary treatment. Note that computing the intersection of a link with the solid boundary mesh needs to be performed at each time step, requiring a search in a bounding volume hierarchy [Karras 2012] on GPU which can be slow if the grid resolution is very high near the boundary to guarantee accuracy. We thus propose a novel implementation which greatly reduces the computational cost of ray intersection and dynamic coupling on GPU, which we will cover in detail in Sec. 3.4.

Domain wall treatment. The manner with which a domain boundary is treated can also significantly influence the accuracy of the physical quantities calculated inside. In order to minimize impact, slipping boundary conditions are usually applied along the domain wall away from the inlet and outlet. While there are multiple ways to implement slipping boundary [Krüger et al. 2016], their accuracy is typically not sufficient for strong turbulence, and the domain has to be further expanded which is very inefficient and memory-consuming. To enable more reliable simulation, we first enforce the boundary condition as $\boldsymbol{u}_w = \boldsymbol{u}_n$ where $\boldsymbol{u}_n$ is the nearest grid node along the wall normal direction; then, using this wall velocity, we treat the wall as a moving boundary and apply the treatment described in the previous paragraph to calculate the wall's distribution functions. A similar treatment can be applied for inlets and outlets if needed: knowing their velocity and pressure, missing distribution functions can be reconstructed using the same boundary treatment scheme we proposed. This simple treatment prevents the domain boundaries from overly influencing the accuracy of our simulator.

Ground treatment. As virtual models are often placed on or near the ground, the ground floor also requires special treatment: if the flow near the ground is not well resolved, it may heavily affect the overall simulation accuracy. Instead of the treatment of domain walls discussed above, we adopt a special treatment for the ground based on the reconstruction of missing distributions using a wall model [Malaspinas and Sagaut 2014] which algebraically determines the velocity $\boldsymbol{u}_g$ at a node right above the ground, based on wall friction velocity deduced from nearby nodes, while its density ρ_g is assumed to be the same as the node right above it to enforce the Neumann condition $\nabla\rho \cdot \boldsymbol{n} = 0$. We then recontruct the missing distributions $\{f_i\}$ as $f_i = f_i^{eq} + f_i^{neq}$ with second-order accuracy using:

$$f_i^{eq} \approx w_i \rho_g + \left(1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}_g}{c_s^2} + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{u}_g)^2}{2c_s^4} - \frac{\boldsymbol{u}_g \cdot \boldsymbol{u}_g}{2c_s^2}\right), \tag{17}$$

$$f_i^{neq} \approx -\frac{\tau \rho_g \omega_i}{2c_s^2} \langle \boldsymbol{Q}, \nabla \boldsymbol{u}_g + \nabla \boldsymbol{u}_g^T \rangle, \tag{18}$$

where $\boldsymbol{c}_i$ is the lattice velocity from Eq. (3), and $Q_{\alpha\beta} = c_\alpha c_\beta - c_s^2 \delta_{\alpha\beta}$ ($\delta_{\alpha\beta}$ being the Kronecker delta). Such a wall function approach could still be insufficient at very high Reynolds number, but developing an accurate ground treatment for treating turbulent flow interaction within the narrow channel between the virtual model and the ground is still an active research topic; so any progress in this direction could be adopted instead.

Force evaluation over the model surface. With the boundary treatment described above, the force acting on a solid is computed using the momentum exchange method [Ladd 1994; Mei et al. 2002]: the momentum transferred to the solid at a given cut-cell node is expressed as the summation of fluxes across the boundary via:

$$\Delta \boldsymbol{p}(\boldsymbol{x}) = \sum_{j \in L(\boldsymbol{x})} \boldsymbol{c}_{j'} \left(f_j(\boldsymbol{x}) + f_{j'}^*(\boldsymbol{x})\right), \tag{19}$$

where $L(\boldsymbol{x})$ denotes the set of outgoing intersecting links (going from the solid to the fluid) at $\boldsymbol{x}$, and $f_j(\boldsymbol{x})$ is the distribution after boundary treatment. The total force on a solid is thus the sum of this term per cut-cell node inside the fluid, i.e.,

$$\boldsymbol{F} = \sum_{\boldsymbol{x}} \Delta \boldsymbol{p}(\boldsymbol{x}). \tag{20}$$

Note that if an immersed treatment is adopted for fluid-solid coupling instead, the summation is then over *all* cut-cell nodes.

3.3 Multiresolution simulation

Running the simulation framework we described up to now on a single LBM grid would be enough for most CG applications looking to efficiently capture a fluid flow over relatively simple geometric models for visual purposes. But in an engineering context, a virtual wind tunnel must accommodate large domains (typically of size 40m×20m×20m for a 16-foot car model) with a constraint on resolving geometric details as small as 3mm typically to capture small-scale flow features well enough to ensure an accurate evaluation of key physical quantities. A single grid would thus have to be impractically large to fit these requirements, creating an unacceptable wall-clock time and memory storage for simulation. Multiresolution simulation is thus an indispensable ingredient of a physically-accurate fluid simulator for engineering design, and grid refinement has thus been systematically adopted in most industrial simulators [Hou et al. 2019; Aultman et al. 2022; Romani et al. 2022]. However, as argued before, many traditional FVM-based NS solvers usually require body-fitted unstructured meshes, for which a versatile and fully automatic mesh construction algorithm is extremely challenging to generate due to the complex geometry (and sometimes, topology) of the prototypes. In the LBM literature, multiresolution methods often employ octree-based grid data structure [Eitel-Amor et al. 2013; Hasert et al. 2014], but their GPU implementations are quite involved [Schornbaum and Rüde 2016, 2018]. Instead, we propose a fully automatic block-based multiresolution grid construction method for our LBM fluid simulator that is better adapted to efficient GPU implementation and can handle flows through extremely complex structures as described next.

Multiresolution grid construction. Grid refinement is often guided by different criteria, among which incoming flow direction and distance to a model [Lagrava 2012; Li et al. 2019] are arguably the most common choices. We thus designed our automatic multiresolution grid construction algorithm around these two factors. Given an axis-aligned bounding box (AABB) of the model, we first extend it by a certain distance d^* representing the boundary layer thickness determined by the Reynolds number, in order to form a larger box-shaped region labeled Ω_n — see the red box in Fig. 6. From the whole simulation domain Ω, we then define the pure flow region $\Omega_f = \Omega \setminus \Omega_n$. Inside Ω_f, we construct our multiresolution data structure based solely on the direction of the incoming flow by using axis-aligned grids of power-of-two resolutions which partially overlap (to ease grid transition); i.e., we go from coarse at the domain boundary to finest at Ω_n by ensuring that each uniform grid level is twice as fine as its parent while being offset from the center Ω_n along the incoming flow direction so as to cover more of the wake flow (where the turbulence must be well resolved) than in the front — see how the regions from blue to yellow unevenly straddle the building in Fig. 6. As for inside Ω_n, we refine the sampling once further compared to the finest grid level in Ω_f, but this time we transition to a refinement based on the distance to the model: using an unsigned distance field of the model computed within Ω_n, we use a once-refined uniform grid compared to the finest grid level

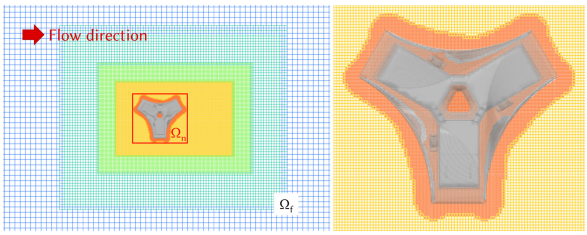


Fig. 6. **Multiresolution grid construction.** To keep memory size low when computing accurate predictions of physical quantities, multiresolution grids are automatically constructed, with a refinement guided by the incoming flow direction for all but the last grid level (left) — explaining the offset of the different levels of grid compared to this architectural building so as to capture its wake accurately — then by the distance to the object. All active fluid nodes are finally found by flooding to ensure that the flow goes through all the mesh openings larger than the finest grid resolution (right).

in Ω_f, and consider its nodes to be *valid* only if they are within a distance d^* of the model — see the orange nodes in Fig. 6. Note that we use a mask per grid node to indicate whether or not it is a valid fluid node, and we adopt the efficient GPU-based approach of [Imre et al. 2017] to compute the distance field. The resulting multiresolution sampling is thus block-based and independent, as it does not form a tree-like hierarchy — but it allows for a good transition between multiple levels of resolutions without the prohibitive memory storage required by the continuous-scale approach of [Li et al. 2019], for instance.

Dynamic objects. Because of the specific nature of a wind tunnel facility, we can restrict our handling of dynamic objects in a simulation with one-way coupling to mostly rotations, such as the wheels of a car spinning. From an axis-aligned bounding box of the rotating object, we construct a grid with the finest resolution and set all its nodes as valid fluid nodes. More complex motions, such as the opening of a plane's wheel well with the landing gear coming out, can in fact be handled the same way, but one must then pick an axis-aligned bounding box that encloses the complete motion of the dynamic objects, which can be potentially wasteful in the number of finest nodes depending on the scenario at hand. We leave such very specific improvements to future work.

Dealing with complex models. We assumed until now that the model surface is closed, for which using a signed distance is sufficient to identify valid fluid regions. However, in real applications, many models are not truly closed: e.g., a car may have a grill wire as an external air inlet, or an architectural model can have small openings, still larger in size than the finest grid cell, to let the wind flow inside parts of the building. In this case, the above grid construction is not sufficient: we must run a flooding algorithm on the *finest* grid level to properly tag as valid the nodes that are connected to the external flow. Starting from a known fluid node — e.g., the node at the corner of the region Ω_n — we check its 6 neighbor nodes and perform link-surface intersection to see whether a neighboring node is connected to the current fluid node, see Fig. 7 in 2D as an illustration. Only if a neighboring node is connected to the current fluid node do we mark it as a valid fluid node, and we propagate through the valid fluid nodes using a breath-first search order such that all fluid nodes inside Ω_n are visited.

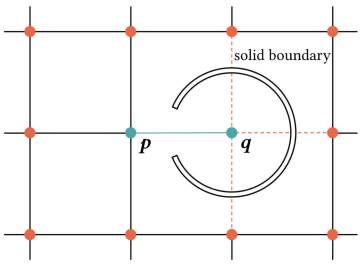


Fig. 7. **Propagation of valid fluid nodes.** To find and tag all valid fluid nodes, we start from an already known fluid node, e.g., node $\boldsymbol{p}$ in 2D, then we check its immediate neighbors to see whether a link intersects a boundary. If no intersection is detected (e.g., link $\boldsymbol{pq}$), the untagged node $\boldsymbol{q}$ is tagged as a new valid fluid node. This process repeats in a bread-first search order, enabling internal regions to be connected to the external fluid region.

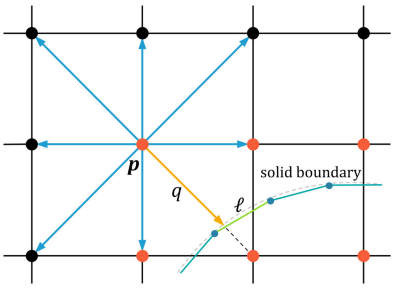


Fig. 8. **Efficient link-boundary intersection.** Instead of building a tree-based data structure, we parallelize our link-mesh intersection over all boundary elements — here boundary edges ℓ in 2D. For a given boundary edge, we efficiently locate the cells that the edge straddles, and perform intersection for every link of the nodes that belong to the covered cells.

Grid interpolation. Our multiresolution grid construction enforces that only *two* grids that are one level apart can overlap. We must therefore define how to evaluate and update in time the distribution functions at a given node based on the values stored on these two levels. For this purpose, we strictly follow the approach described in [Lagrava et al. 2012]. The internal nodes of a coarser parent grid provide boundary values which allow the finer grid to be updated twice (since a twice-finer grid requires twice-smaller integration time steps in LBM), where the distribution functions at the boundary nodes of the finer grid are readily interpolated from the coarser grid at the current time step. Then, the boundary nodes of the coarser grid can be updated by interpolating from the already updated finer grid at the next time step, and so on. The interpolation is done by separating the distribution functions into equilibrium and non-equilibrium parts, where the equilibrium part is computed by first interpolating macroscopic quantities before evaluating its equilibrium distribution values, while the non-equilibrium part is interpolated directly; both interpolations are done using high-order schemes [Lagrava et al. 2012]. This process is applied recursively from the coarsest grid to the finest grid in a cascaded serialized manner. Fig. 16 for instance shows an instantaneous macroscopic velocity field cross-section from the simulation of the same architecture model from Fig. 6 using this interpolation approach.

3.4 Implementation details

We now go over a number of implementation details that we found have significant impacts on the overall efficiency of our technique once implemented on GPU. An overview of our simulation process is summarized in Alg. 1.

Efficient link-mesh intersections. Computing the intersections between links and one or more 3D models is required as preprocessing for static objects or at each time step if dynamic solids like a rotating wheel are involved in the simulation. Usually, link-mesh intersection relies on the construction of a tree structure (e.g., a bounding volume hierarchy (BVH)) of the mesh elements on GPU to improve efficiency [Karras 2012]. However, in the context of our LBM simulator, this acceleration still leads to intersection costs superior to the actual flow simulation at each time step. Instead, we thus opted for a spreading-based algorithm inspired by the immersed boundary treatment from [Chen et al. 2022]. While parallelization is over all cut-cell fluid nodes in tree-based approaches, we parallelize instead over *mesh elements*. For each element on the boundary surface, we check which grid cells they straddle; then all the grid nodes belonging to the straddled cells are taken as candidate nodes for intersection test, see the orange nodes in Fig. 8 for a 2D illustration where the elements are 2D edges (ℓ) of the boundary. For each of the candidate nodes, all links are checked through an intersection test against the current mesh element, and if a link is intersected, we compute and store the lattice-normalized distance value q for this node which will be used for the boundary treatment described in Sec. 3.2. For each cut cell node, we thus keep an array to store these potentially different q values (8 in 2D, 26 in 3D). Note that for a given mesh element, the number of candidate nodes that need to be tested always remains relatively small, thus the new algorithm runs much faster on GPU due to improved data coalescence and load balancing: for a model having a size of around 4.5 meters whose mesh contains about 10 million elements with a spatial resolution of 4mm (dictating the size of the finest grid resolution), the acceleration over tree-based approach is around a factor ten, significantly reducing the total computational cost of a simulation — especially when dynamic objects are involved.

Data layout. In LBM, streaming of distribution functions requires accessing neighboring nodes; thus specific data structures are needed for efficient access in our multiresolution setup. First, all distribution functions are stored in an array-of-structure layout as proposed in [Chen et al. 2022]. However, since our valid fluid nodes are not always contiguous, a compact and GPU-friendly data structure is preferable; but perfect spatial hashing [Lefebvre and Hoppe 2006] or compact storage as used for sparse matrices [Greathouse and Daga 2014] either increase the number of memory fetches or violate continuity of data storage, resulting in latency for memory access. Moreover, block-based structures with z-ordering for irregular geometric domains as proposed in [Chen et al. 2022] increase implementation complexity and memory fetching as well. As a compromise, we simply create an *index map* for each node of a grid which directly indexes the local distribution functions by pointing to a linear array in global memory containing all valid fluid nodes packed in an x-y-z order. We found this layout to have similar overall efficiency as the block-based structure in [Chen et al. 2022] for a much simpler implementation.

4 TESTING OF OUR VIRTUAL WIND TUNNEL

We now discuss the various tests we performed to validate the accuracy and efficiency of our virtual wind tunnel testing facility, along with a series of additional rendered results to show how the improvements we made to state-of-the-art LBM techniques do not only allow for faster wind tunnel testing in automotive, architectural, or aeronautical industries, but also for high-resolution realistic fluid flows for VFX production.

4.1 Simulation setup and visualization

We implemented our virtual wind tunnel to exploit the massive parallelism of graphics processors, and simulations were executed with one or more NVIDIA A100 GPU(s). We ran our code on a workstation equipped with an Intel 20-core CPU and 128 GB of RAM.

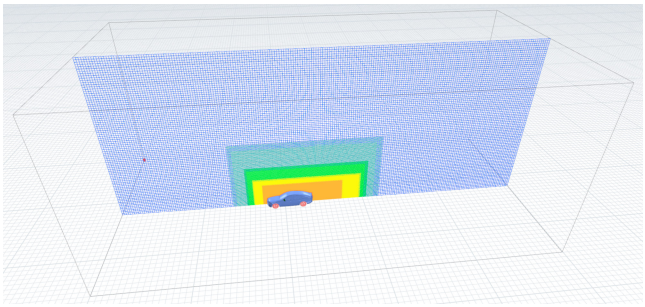


Fig. 9. **Typical virtual wind tunnel test.** For accurate prediction of physical quantities, the computational domain (in wireframe) should have a size typically 10 times the size of the model's bounding box, in each direction.

Algorithm 1: Our virtual wind tunnel testing facility

```
Initialization
    Multiresolution grid construction;          ▷ Sec. 3.3
    Precompute link-mesh intersections;         ▷ Sec. 3.4
    Initialize ρ, u and f_i;
for t = 1 to N do
    Collision
        Transform f to k;                       ▷ [Geier et al. 2017a]
        Calculate relaxed k*;                   ▷ Sec. 3.1
        Transform k* to f;                      ▷ [Geier et al. 2017a]
    Streaming
        switch link flag do
            case unknown cut-cell link do
                Boundary treatment;             ▷ Sec. 3.2
            case unknown ground link do
                Ground treatment;               ▷ Sec. 3.2
            otherwise do
                Normal streaming;
            end
        end
    Post-processing
        Evaluate forces on objects;             ▷ Sec. 3.2
    end
end
```

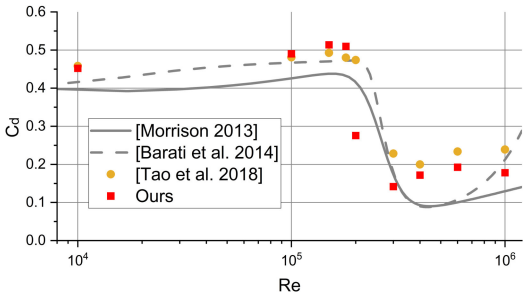


Fig. 10. **Drag of a sphere vs. Reynolds number.** Just like real-life experiments [Morrison 2013; Barati et al. 2014] exhibit a sudden drop in drag for a sphere at a Reynolds number around Re=400,000 (drag crisis), both [Tao et al. 2018] and our kinetic solver demonstrate a similar drop at roughly the same Re, followed by a partial drag recovery.

Each simulation is typically set up by placing the test model at one third along the direction of the flow (streamwise), and in the middle of the orthogonal plane (spanwise) to capture the turbulent wake flow well. For test models flying in the air, we place the model in the middle along the vertical direction; otherwise, they are placed on the ground. The simulation domain is selected to be typically 8 to 10 times in size compared to the bounding box of the model — see Fig. 9 for an example illustrating the setup of our virtual wind tunnel domain for the aerodynamic simulation of a car model. Cross-section visualizations of the magnitudes of velocity fields as in Figs. 12, 13, 15 and 16 were generated on the finest grid resolution of the simulation, where the color of each pixel is mapped from the linearly interpolated velocity magnitude — other fields such as vorticity or pressure are done similarly. To display the pressure (or pressure coefficient C_p) *on* the surface of a model, we did not use a simple projection or extrapolation from nearby cut-cell nodes as the nodes used could have different distances to the surface, causing sudden changes of pressure values and generating obvious staircase phenomena; instead, for each mesh vertex, we perform pressure interpolation at an offset position one-cell-width away along the outer normal direction to ensure consistent interpolation, and use this resulting value as the mapped color, which leads to much improved surface visualization — see Fig. 1 for instance. We also use passively-advected particles injected in the virtual flow, which were converted to VDB files and rendered with [Maxon 2023] to offer a Lagrangian visualization of the flow in some of our figures, such as Fig. 17. Simulations were computed in a few hours depending on the scale of the model, the size of the computational domain, as well as the finest spatial resolution required, with our multiresolution grid construction usually taking a few minutes: Tab. 2 details timings and configurations for all the simulations we conducted in this paper.

4.2 Validation of our virtual wind tunnel

To validate our virtual testing facility, we perform three increasingly difficult cases for which experimental data acquired in real-life wind tunnels exist to compare our results both qualitatively and quantitatively: drag crisis for a sphere, a golf ball in flight, and finally, three variants of a benchmark car model (that we each test with or without the wheels being allowed to spin) which are typically used to validate a solver's accuracy, and for which we can thus better compare performance and accuracy.

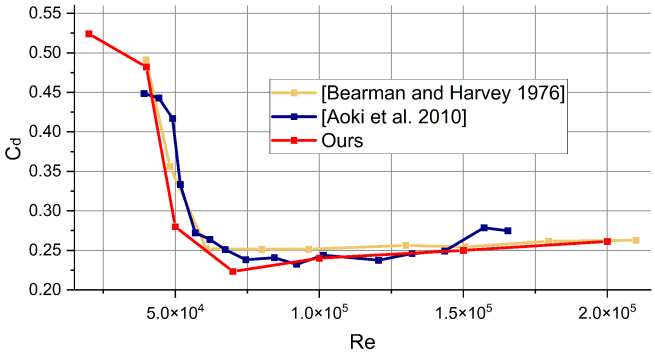


Fig. 11. **Drag of golf ball vs. Reynolds number.** Compared to the real-world experiments reported in [Bearman and Harvey 1976] and [Aoki et al. 2010] for the drag coefficient of golf ball as a function of the Reynolds number Re, our wind tunnel simulator provides very similar evaluations. Note that the drop in drag coefficient (drag crisis) happens far earlier than in the case of a smooth sphere, as expected — see Fig. 10.

Drag crisis for a sphere. As mentioned earlier, the phenomenon called drag crisis refers to the seemingly paradoxical fact that the drag coefficient of, say, a sphere, suddenly drops at a critical Reynolds number and only partially recovers as the Reynolds number keeps increasing, which is very challenging to correctly predict for a numerical solver without turbulence modeling. Because this drag crisis happens at high Re and depends on the surface property of the sphere (the effect occurs at lower Reynolds numbers when the sphere is rough than when it is smooth), experimental data is exceedingly difficult to gather accurately (in fact, Fig. 5(c) was obtained at a lower Reynolds number by adding a trip wire to artificially create a higher effective Reynolds number), explaining the discrepancy between experiments such as [Morrison 2013] and [Barati et al. 2014]. Our virtual wind tunnel manages to qualitatively reproduce the sudden drop in drag coefficient for a sphere as illustrated in Fig. 10 at about the correct Reynolds number, and with drag coefficients closer to the experimental values in general beyond the critical Reynolds number as compared to an LBM solver using the boundary treatment of [Tao et al. 2018] without entropy-based cumulant collision model. Note that the drag prediction from our method is slightly shifted from the critical drag-crisis region, producing a larger deviation from the experimental data at Re=200,000. In this critical region (where even the two experimental curves differ), the numerical treatment of the surface — and in particular of its roughness — can easily alter drag prediction as discussed in [Geier et al. 2017b]. One should also point out that, like many other state-of-the-art fluid flow solvers, we over-estimate the drag coefficient values for higher Reynolds numbers: while experiments seem to show a drop down to $C_d = 0.1$, we reach 0.175; but it remains better than the results of [Tao et al. 2018] which go down to 0.2 only.

Flight of a golf ball. We also performed tests on a golf ball in a high-speed flow with our virtual wind tunnel. The golf ball model we used has 362 dimples (The PGA tour uses balls with a minimum of 322 and a maximum of 376 dimples), with a ratio of dimple depth to ball diameter equal to 0.007. As Fig. 11 demonstrates, our evaluated drag coefficients are comparable to the published experimental values from [Bearman and Harvey 1976] and [Aoki et al. 2010]. We also use this golf ball case to show the ability of our solver to resolve small geometric features: as expected, the flows at $Re = 100,000$ around a golf (*dimpled*) ball vs. a ping-pong (*smooth*) ball are drastically different, since dimples generate very small boundary layer vortices that interact with the surrounding flow, reducing the attachment of the laminar boundary layer to the surface of the golf ball. This makes the overall flow separate farther around the back side of the golf ball, thereby forming a converging shape of the wake with reduced drag, compared to the wider wake with larger vortices for a smooth ball due to laminar boundary layer separation as depicted in Fig. 12 with our solver via passively-advected colored particles. This converging shape is very similar to the wake of a ball close to drag crisis, explaining the drag reduction mechanism by a production of small-scale vortices around the solid boundary. Note that our multiresolution simulation is key here in capturing this kind of details: a single resolution with the same total number of grid nodes fails to resolve the thin turbulent boundary layer created by the small dimples, forming different separation points of boundary layer flows and wake shapes, and thus grossly over-estimating the underlying drag coefficients, see Fig. 13 (right).

Aerodynamics of DrivAer car model. Finally, we ran our virtual testing facility on three variants of the DrivAer model, a benchmark car model created at Technische Universität München to investigate automotive aerodynamics and validate numerical simulations [Heft et al. 2011, 2012a]. We tested three rear-end forms — called Fastback, Notchback and Estateback respectively — in our experiments. All meshes are digital models of a real car down to a 3mm accuracy, with detailed underbody, mirrors, but no engine bay flow, while the domain resolution is chosen to balance accuracy and efficiency. For each model, we tested the aerodynamic characteristics at $57.6km/h$ based on our simulator in two typical scenarios used in industrial tests: without or with ground simulation (GS) — that is, in the first case we assume that the ground is fixed and the wheels do not spin, while in the second case, the car has been put on a conveyor belt to simulate the road moving under the car, and the wheels are rotating on it. These two scenarios provide great insights on the various sources of numerical issues, showing that even the seemingly innocuous effects of a rotating wheel or a fixed ground can greatly affect wind tunnel testing and the resulting aerodynamic coefficients. We plot the resulting drag curves in time in Fig. 14, showing that the drag begins to stabilize once the transient effects early in the simulation attenuates past the first second. The actual drag coefficient estimates are then obtained by averaging the values between $1.2s$ and $2s$, and compared to ground-truth values from [Heft et al. 2012b] after a final calibration via mean shift as listed in Tab. 1. Note that this mean shift was measured by computing the average deviation between a set of computed drag coefficients and their corresponding experimental data over different setups, which is a common engineering method to compensate for limited resolution and the resulting systematic failure to resolve the turbulent flow between the car body and the ground. Mean velocity and pressure corresponding to the same averaging range as above are also shown in Fig. 15, where pressure on wheels are shown separately as insets to highlight the differences between the simulation with and without ground simulation — clearly indicating that rotating

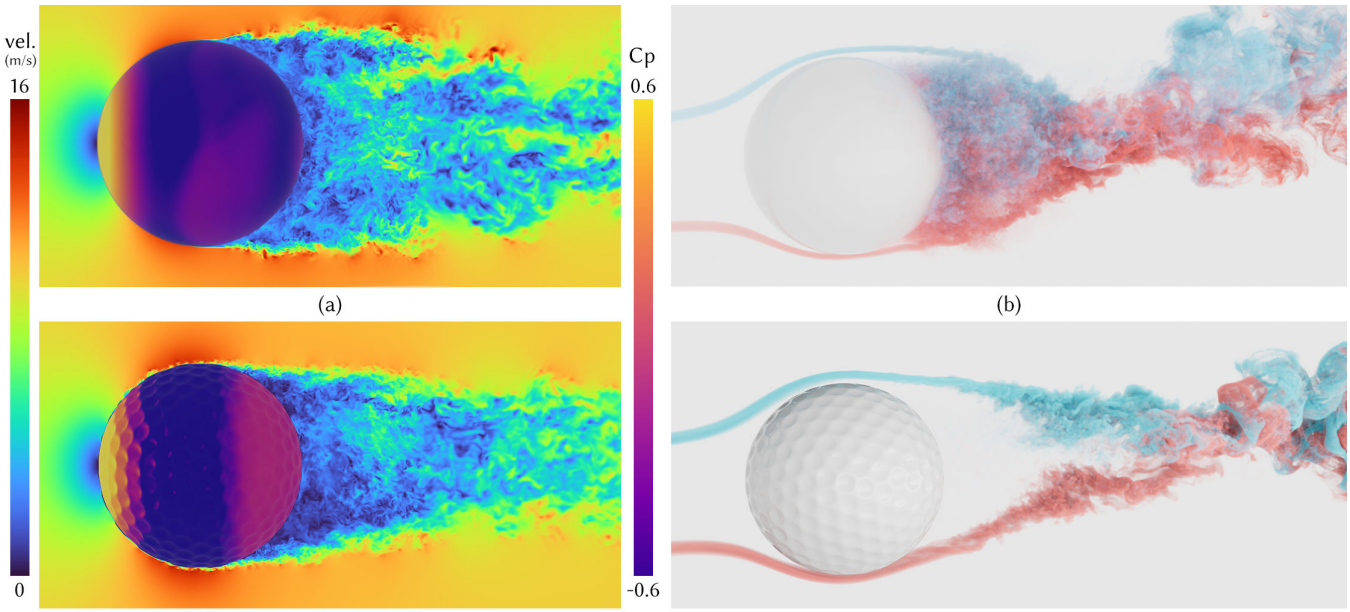


Fig. 12. **Ping-pong vs. golf ball at Re=100,000.** While a ping-pong ball (top) differs (up to scale) from a golf ball (bottom) only in the absence of tiny dimples on its surface, testing these two balls in our wind tunnel exhibits very different velocity and surface pressure (C_p) fields (left); consequently, the flows visualized via passively-advected dyed particles are dramatically different (right), providing a good intuition of why golf balls can travel much further.

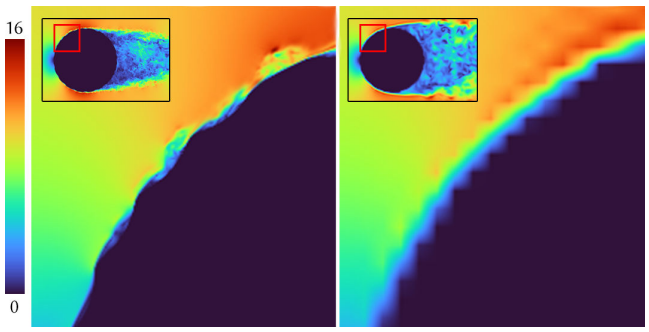


Fig. 13. **Multi- vs. single-resolution simulation.** A multiresolution simulation better resolves the boundary layer flow going within the small dimples of a golf ball (left), while a single-resolution simulation with the same total number of grid nodes cannot (right). As a consequence, we witness a very different behavior of the turbulent wake when the velocity magnitude (with a colormap indicating its value in m/s) is visualized.

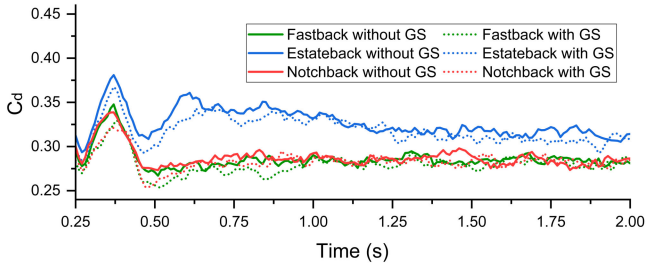


Fig. 14. **Car drag over time.** We plot the simulated drag coefficient in time for different DrivAer car configurations, with or without ground simulation (GS, meaning ground motion and rotating wheels are simulated).

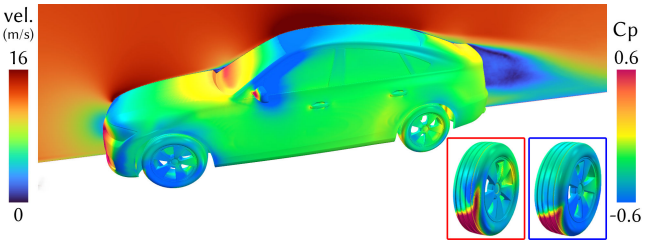


Fig. 15. **DrivAer Fastback aerodynamic simulation.** A vertical cross-section shows the magnitude of the mean velocity flow around the DrivAer fastback benchmark car model, while the mean pressure over the model surface colors the mesh. Mean pressure distributions without (red inset) and with (blue inset) ground simulation are also visualized on the wheels.

wheels reduce drag due to their generation of small boundary layer vortices. Expectedly, the case with no ground simulation is easier, and our virtual wind tunnel captures the drag coefficient values almost perfectly with an average error of 0.4%, which is, based on the car companies we communicated with, far better than the industry standards requiring less than 3% error in this case. With ground simulation, our maximum error reaches 3.17%, still safely below the limit tolerated in the automotive industry, which is of 5% in this more challenging case.

Comparisons with industrial software suites. As mentioned early on, Siemens' StarCCM+ [Siemens PLM Software 2023] and Dassault Systèmes' PowerFLOW [Exa/3DS 2023] are two well-established commercial tools implementing respectively an FVM-based NS solver and an LBM-based solver, both running on CPU clusters for industrial users. Because time-dependent unsteady flow simulation with StarCCM+ can take up to weeks of computations (having the

Table 1. **Drag estimation accuracy.** We compare our estimates of the drag coefficient C_d for different DrivAer configurations with experimental data, with or without ground simulation (GS, meaning ground motion and rotating wheels are simulated).

Car configurations	*Predicted* C_d	*Measured* C_d	*Relative Error*
Fastback w/o GS	0.2849	0.284	0.32%
Notchback w/o GS	0.2851	0.286	-0.31%
Estateback w/o GS	0.316	0.318	-0.63%
Fastback w/ GS	0.2811	0.275	2.22%
Notchback w/ GS	0.283	0.277	2.17%
Estateback w/ GS	0.3089	0.319	-3.17%

wheels rotating require dynamic remeshing for such a solver), users often settle for a steady-state solution as a compromise, which does not closely match the real physical measurement process because boundary approximations of rotating objects are often employed. PowerFLOW overcomes these difficulties and affords higher performance, although at the cost of relying on a larger CPU cluster than Siemens' solver due to the increase in the number of nodes needed, which often curbs its practical appeal. Instead, our virtual wind tunnel system inherits the benefits of kinetic solvers (and thus the advantages of PowerFLOW), but runs on GPU with far less stringent hardware requirements: case in point, with an NVIDIA A100 GPU having 6,912 CUDA cores, the simulation for drag prediction of the DrivAer Fastback model without ground simulation took only around 3 hours to complete, which is 10,356 GPU core hours per second of simulation. James et al. [2018] recently reported a similar drag prediction on a Notchback model using PowerFLOW and StarCCM+, which require approximately 80 hours on 96 CPUs and 13 hours on 128 CPUs, respectively, to get a converged result. This indicates that our efficiency is likely to be better using the same number of processor cores, assuming that each of their CPU had at least 8 cores — although comparing performance across CPU clusters and GPUs is surely difficult. More strikingly, the drag coefficients reported for PowerFLOW do not show an obvious difference with or without ground simulation, which contradicts experimental data. Other results have approximately around 3% error compared to the experimental measurements reported in [Heft et al. 2012b]. We note that they used a model without side mirrors while we used a full car model, and no calibration information was provided. Thus, formulating a more precise statement on how accurate our solver is in comparison to PowerFLOW and StarCCM+ still requires further testing, but our initial analysis of performance indicates that we are at least not worse, and likely better than existing commercial software — and our maximal 3.17% error on drag prediction with ground simulation and with both side mirrors and a detailed underbody (which naturally produce stronger turbulence) proves that our virtual testing facility clearly meets current industry standards.

4.3 Other simulation results

In order to demonstrate our applicability not only to industrial design purposes but also as a tool for VFX animation of exquisitely complex flow patterns around models, we have performed a number of additional simulations. For instance, we conducted a high-resolution aerodynamic simulation of a scaled Being 787 aircraft model at an angle of attack of 8 degrees at relatively low speed (0.16 Mach) in Fig. 17. The model has a length of 6.16 meters and a wing

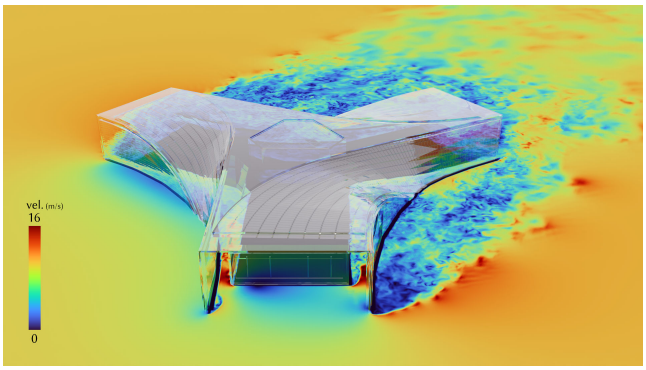


Fig. 16. **Aerodynamics of an architectural model.** Our virtual wind tunnel can simulate the airflow passing through a building structure containing covered passages inside. Visualized here is the velocity field magnitude for a horizontal cross-section, where the internal flow is clearly visible.

span of 3.2 meters. The simulation was conducted on two NVIDIA A100 GPUs, with the finest grid capturing details of 3.75 millimeters near the model boundary. It took around 1.8 hours to run a simulation of 1.7 seconds. We also simulated the airflow within a complex scaled architectural model containing covered hallways, where node flooding was necessary to select all the valid grid nodes, see Fig. 6. The model has a bounding box of size $4.34m \times 1.06m \times 3.76m$, within a domain of size $25m \times 8m \times 25m$. It took 1.9 hours to complete a simulation of 3.5 seconds, see the resulting cross-section visualization of velocity in Fig. 16. To further demonstrate the ability of our multiresolution testing facility to handle high geometric and topological complexity, we simulated a flow passing through an irregular pipe with nozzles on its surface (Fig. 19), where the nozzles connect the internal fluid region with the exterior. The pipe model has a bounding box of size $1.3m \times 2m \times 0.83m$, within a domain of size $7.84m \times 12m \times 4.96m$. The simulation took 0.9 hours to produce 6 seconds of simulation. Visualized with smoke particles, it is clear that the flow goes into the pipe through the bottom left inlet and leaves the pipe through the nozzles as expected, highlighting our ability to automatically construct multiresolution grids on very complex models. We hope to be able to quantitatively compare some of these results (plane, building) by acquiring higher quality model meshes together with validation datasets in the near future, in addition to the current visual demonstrations.

4.4 Discussions and limitations

While the state-of-the-art CG methods recently proposed [Li et al. 2020; Lyu et al. 2021] have significantly raised the bar for both efficiency and accuracy compared to other CG fluid solvers, they still have not reached a level of accuracy, scale, and even stability needed for industrial design simulations — see Fig. 18 for a comparison in a very simple case at relatively low Reynolds number but with multiresolution simulation, in which previous LBM methods proposed in CG blow up. Conversely, methods in CFD have managed to propose ever-more accurate collision models and boundary treatment methods, but they mostly focused on the solver itself, ignoring many other important factors in a virtual wind tunnel system such as the non-trivial automatic multiresolution grid construction or efficient geometry handling, which are key to practical use in industrial

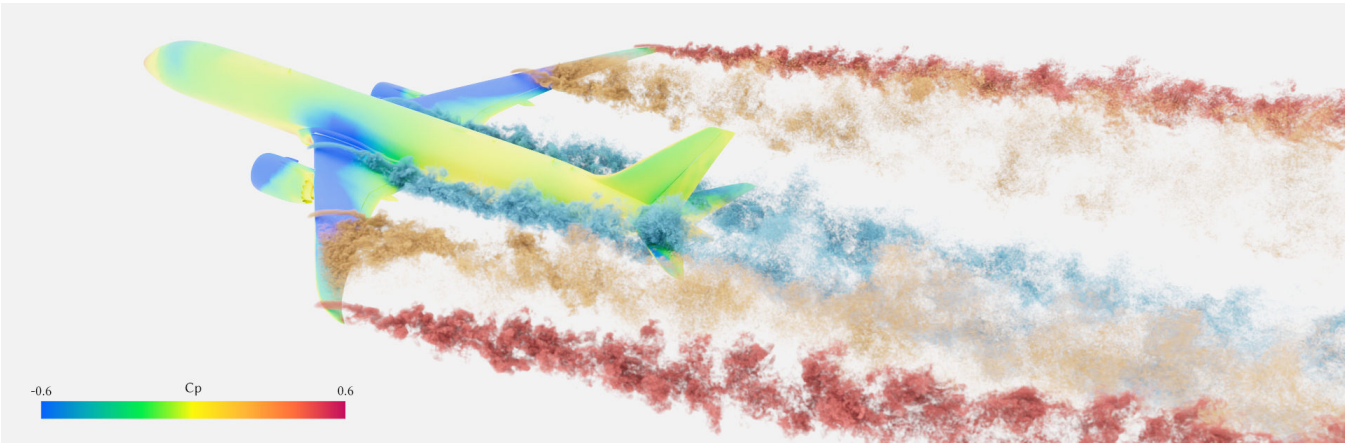


Fig. 17. **Aerodynamics of a Boeing-787 passenger aircraft.** We conducted a high-resolution aerodynamic simulation on dual GPUs for a scaled aircraft model of Being 787 at an angle of attack of 8°. The pressure field is color-mapped over the aircraft body surface, while passively-advected dyed particles injected from the six different locations along the leading edge of the main wing are visualized.

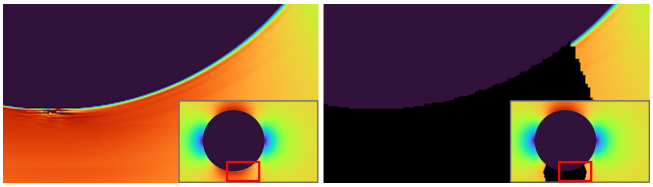


Fig. 18. **Typical blowup of latest LBM solvers in CG.** Existing LBM methods in CG mostly rely on central-moment collision models [Li et al. 2020; Lyu et al. 2021]. Due to the entangling of various orders of moments and improper high-order relaxation rates, they often blow up at low Reynolds numbers (here, Re=100,000) when simulating even a simple flow around the sphere with multiresolution grids, see black region (right), due to spurious velocities near the solid boundary (left).

design. Our work not only bridges the gap in scale (through multiresolution) and accuracy (through our entropy-driven cumulant model and enhanced boundary treatment) necessary to reach engineering applications, but also significantly broadens the type of simulation that the VFX industry can benefit from: fluid flows over large-scale scenes and/or zoomed-in shots are easily handled with much higher efficiency than current NS solvers. Importantly, the ability to simulate low, high, or very high Reynolds numbers is a crucial benefit in making sequences that capture the proper scaling of turbulence, a feature that has often been lacking in previous graphics techniques.

Limitations. Our method is not without limitations. First, as we have seen previously, our drag coefficient predictions are far better for a static setup (without GS) than for a dynamic one (with GS); while this is true of many other solvers, how to improve dynamic boundary condition to have predictions of physical quantities with consistent error margins deserves further research. Second, our handling of data structure for irregular domains is currently not optimal, so we are guessing that further improvements in terms of performance are possible. Lastly, LBM solvers usually have higher memory usage than the corresponding NS solvers with similar accuracy — albeit for a far accrued efficiency. How to reduce memory consumption is another aspect that needs further developments.

5 CONCLUSION

In this paper, we developed a new multiresolution LBM-based fluid flow simulation technique to build an efficient and accurate virtual wind tunnel for physical evaluation of digital models. The improved accuracy and efficiency brought forth by our contributions based on state-of-the-art techniques from both CG and CFD afford a unique opportunity to not only perform visual simulation of complex fluid dynamics for VFX, but also to design, preview, and assess the design of cars, planes, or buildings at subsonic weakly-compressible speeds prior to the construction of their full-scale mockup: our results allow for fine spatial resolution of flow fields near model boundaries to be made early in the design process for analysis and feedback, and consequently, it helps accelerate the design process.

Our work calls for a number of follow-up works. We handled dynamic objects in one-way coupling with restricted rotations and translations to build a virtual wind tunnel; pushing further the treatment of dynamic objects and allowing two-way coupling would offer a much more versatile toolbox for both engineers and animators to simulate quickly and accurately large-scale scenes and/or zoomed-in shots with much higher efficiency than many current NS solvers in CG field. Currently we do not allow dynamic grid reconstruction, although it would keep a low memory usage — but at the cost of slowing down the entire simulation. Creating a *virtual water tunnel*, maybe by building over the recent work of [Li et al. 2022], would also be a great challenge to generate even more complex simulations. On the engineering side, we were able prove that a few key improvements made over recent LBM solvers presented in CG suffice to now open them up to industrial design audiences. While further advancing towards industrial-strength computational tools for flow simulation may bring us outside of the realm of CG, it is an exciting avenue for a number of reasons. First, it represents a *rara avis* in CG, whose industrial impact outside of the entertainment industry has only sporadically happened. Second, it may also offer a number of opportunities to incorporate advanced graphics practices (e.g., in visualization) into industrial practices. Finally, it is an additional hint that the usual gap between graphics simulation

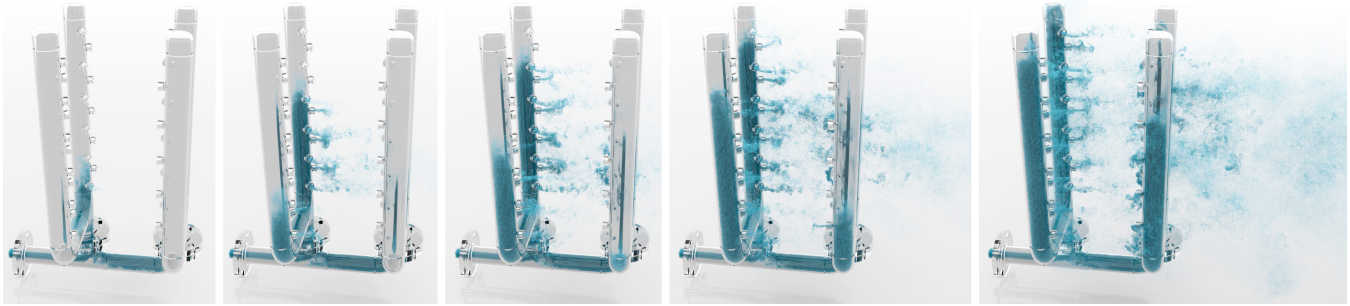


Fig. 19. **Airflow passing through a pipe with nozzles.** With our multiresolution solver, we can simulate a flow passing through an irregular transparent pipe with various nozzles on its surface. By injecting smoke particles at the inlet of the pipe, it becomes clear that the flow gradually fills up the pipe while exiting from the nozzles. Note that the surrouding air was given an initial constant velocity, which blows the smoke rightward after it comes out.

and computational science & engineering is further eroding, offering opportunity for collaborations across the two fields in the near future.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for helping us improve exposition. This work was supported by the National National Science Foundation of China (No. 62072310) and ShanghaiTech University. MD acknowledges the generous support of a Choose France Inria chair. 3D models were provided by GrabCAD users Fawaz Bukht Majmader (Fig. 1), Ferro (Fig. 16), and William Heidt (Fig. 19), as well as Sketchfab user mudkipz321 (Fig. 17).

REFERENCES

Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. 2007. Adaptively Sampled Particle Fluids. In *ACM SIGGRAPH 2007 Papers*. 48. https://doi.org/10.1145/1275808.1276437

Ronojoy Adhikari and Sauro Succi. 2008. Duality in Matrix Lattice Boltzmann Models. *Phys. Rev. E* 78 (2008), 066701. Issue 6.

Giancarlo Alfonsi. 2009. Reynolds-Averaged Navier-Stokes Equations for Turbulence Modeling. *Applied Mechanics Reviews* 62, 4 (2009).

Alexis Angelidis and Fabrice Neyret. 2005. Simulation of Smoke Based on Vortex Filament Primitives. In *Symposium on Computer Animation*. 87–96.

Sandosh Ansumali, Ilya V. Karlin, and Hans C. Öttinger. 2003. Minimal Entropic Kinetic Models for Hydrodynamics. *Europhysics Letters (EPL)* 63, 6 (2003), 798–804.

Katsumi Aoki, Koji Muto, and Hiroo Okanaga. 2010. Aerodynamic characteristics and flow pattern of a golf ball with rotation. *Procedia Engineering* 2, 2 (2010), 2431–2436. The Engineering of Sport 8 - Engineering Emotion.

Thomas Astoul, Gauthier Wissocq, Jean-François Boussuge, Alois Sengissen, and Pierre Sagaut. 2021. Lattice Boltzmann Method for Computational Aeroacoustics on Non-uniform Meshes: a direct grid coupling approach. *J. Comp. Phys.* 447 (2021), 110667.

Matthew Aultman, Rodrigo Auza-Gutierrez, Kevin Disotell, and Lian Duan. 2022. Effects of Wheel Rotation on Long-Period Wake Dynamics of the DrivAer Fastback Model. *Fluids* 7, 1 (2022).

Peter Bailey, Joe Myre, Stuart D.C. Walsh, David J. Lilja, and Martin O. Saar. 2009. Accelerating Lattice Boltzmann Fluid Flow Simulations Using Graphics Processors. In *2009 International Conference on Parallel Processing*. 550–557.

Reza Barati, Seyed Ali Akbar Salehi Neyshabouri, and Goodarz Ahmadi. 2014. Development of empirical models with high accuracy for estimation of drag coefficient of flow around a smooth sphere: An evolutionary approach. *Powder Technology* 257 (2014), 11–19.

Peter W. Bearman and John K. Harvey. 1976. Golf Ball Aerodynamics. *Aeronautical Quarterly* 27, 2 (1976), 112–122.

Markus Becker and Matthias Teschner. 2007. Weakly Compressible SPH for Free Surface Flows. In *Symposium on Computer Animation*. 209–217.

Jan Bender and Dan Koschier. 2015. Divergence-Free Smoothed Particle Hydrodynamics. In *Symposium on Computer Animation*. 147–155.

M'hamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. 2001. Momentum Transfer of a Lattice-Boltzmann Fluid with Boundaries. *Physics of Fluids* 13, 11 (2001), 3452–3459.

Jeremiah U. Brackbill and Hans M. Ruppel. 1986. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *J. Comput. Phys.* 65, 2 (1986), 314–343.

John R. Chawner and Nigel J. Taylor. 2019. *Progress in Geometry Modeling and Mesh Generation Toward the CFD Vision 2030*. 2945.

Hudong Chen, Olga Filippova, James Hoch, Kim Molvig, Richard Shock, Christopher Teixeira, and Raoyang Zhang. 2006. Grid Refinement in Lattice Boltzmann Methods Based on Volumetric Formulation. *Physica A: Statistical Mechanics and its Applications* 362, 1 (2006), 158–167. https://doi.org/10.1016/j.physa.2005.09.036

Hudong Chen, Pradeep Gopalakrishnan, and Raoyang Zhang. 2014. Recovery of Galilean Ivariance in Thermal Lattice Boltzmann Models for Arbitrary Prandtl Number. *International Journal of Modern Physics C* 25, 10 (2014), 1450046.

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. 2022. GPU Optimization for High-Quality Kinetic Fluid Simulation. *IEEE Trans. Vis. Comp. Graph.* 28, 9 (2022), 3235–3251. https://doi.org/10.1109/TVCG.2021.3059753

Yong Chen, Xiangyang Wang, and Hanhua Zhu. 2021. A General Single-node Second-order Boundary Condition for the Lattice Boltzmann Method. *Physics of Fluids* 33, 4 (2021), 043317.

Nuttapong Chentanez and Matthias Müller. 2014. Mass-Conserving Eulerian Liquid Simulation. *IEEE Trans. Vis. Comput. Graph.* 20, 1 (2014), 17–29.

Byoungjin Chun and Anthony J.C. Ladd. 2007. Interpolated Boundary Condition for Lattice Boltzmann Simulations of Flows in Narrow Gaps. *Physical Review E* 75, 6 (2007), 066705.

Christophe Coreixas, Gauthier Wissocq, Guillaume Puigt, Jean-François Boussuge, and Pierre Sagaut. 2017. Recursive Regularization Step for High-order Lattice Boltzmann Methods. *Phys. Rev. E* 96 (2017), 033306. Issue 3.

Paul J. Dellar. 2001. Bulk and shear viscosities in lattice Boltzmann equations. *Phys. Rev. E* 64 (2001), 031203. Issue 3.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Comp. Anim. & Simul.* 61–76.

Rahul Deshpande, Vivek Kanti, Aditya Desai, and Sanjay Mittal. 2017. Intermittency of laminar separation bubble on a sphere during drag crisis. *Journal of Fluid Mechanics* 812 (2017), 815–840.

Dominique D'Humières. 1992. Generalized Lattice-Boltzmann Equations. In *Rarefied Gas Dynamics: Theory and Simulations*. 450–458.

Dominique D'Humières, Irina Ginzburg, Manfred Krafczyk, Pierre Lallemand, and Li-Shi Luo. 2002. Multiple-relaxation-time Lattice Boltzmann Models in Three Dimensions. *Phil. Trans. R. Soc. A: Math. Phys. Eng. Sci.* 360, 1792 (2002), 437–451.

Alexandre Dupuis and Bastien Chopard. 2003. Theory and Applications of an Alternative Lattice Boltzmann Grid Refinement Algorithm. *Phys. Rev. E* 67 (2003), 066707. Issue 6.

Jack G.M. Eggels. 1996. Direct and Large-eddy Simulation of Turbulent Fluid Flow using the Lattice-Boltzmann Scheme. *International Journal of Heat and Fluid Flow* 17, 3 (1996), 307–323.

Georg Eitel-Amor, Matthias Meinke, and Wolfgang Schröder. 2013. A lattice-Boltzmann method with hierarchically refined meshes. *Computers & Fluids* 75 (2013), 127–139.

Exa/3DS. 2023. *SIMULIA PowerFLOW*. Dassault Systèmes.

Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. 2000. Finite volume methods. In *Solution of Equation in Rn (Part 3), Techniques of Scientific Computing (Part 3)*. Handbook of Numerical Analysis, Vol. 7. Elsevier, 713–1018.

Table 2. **Statistics.** Parameters and performance statistics of our results, where "*Sim. duration*" refers to the duration of flow simulation measured in real-life seconds, while "*Wall-clock per sim. s.*" refers to the actual wall-clock time it took to simulate one second of a flow, and since the grid construction is only done once, we consider it as preprocessing. Note that for some of the simulations (Figs. 12 and 16) we use Reynolds number equivalence to rescale the physical parameters by adjusting the viscosity, as typically done in the LBM literature to offer normalized parameters without affecting the dynamic behavior.

Cases	*Domain Size* (m^3)	*#Grids*	*Finest* Δx	*Viscosity* (v)	*Flow Speed*	*#GPUs*	*Sim. duration*	*Grid construction*	*Wall-clock per sim. s.*
Fig. 1	15×4×6	5	4.0 mm	2.0×10^{-5}	41.67 m/s	1	2.05 $sec.$	1.58 $min.$	27.59 $min.$
Fig. 12 (a)	17.6×17.6×17.6	6	3.125 mm	1.6×10^{-4}	10 m/s	1	5 $sec.$	3.18 $min.$	31.53 $min.$
Fig. 12 (b)	17.6×17.6×17.6	6	3.125 mm	1.6×10^{-4}	10 m/s	1	5 $sec.$	3.13 $min.$	30.01 $min.$
DrivAer Fastback w/o GS	40×20×16	7	4.0 mm	1.57×10^{-5}	16 m/s	1	2 $sec.$	6.03 $min.$	49.29 $min.$
DrivAer Notchback w/o GS	40×20×16	7	4.0 mm	1.57×10^{-5}	16 m/s	1	2 $sec.$	7.43 $min.$	83.78 $min.$
DrivAer Estateback w/o GS	40×20×16	7	4.0 mm	1.57×10^{-5}	16 m/s	1	2 $sec.$	7.27 $min.$	79.29 $min.$
DrivAer Fastback w/ GS	40×20×16	7	4.0 mm	1.57×10^{-5}	16 m/s	1	2 $sec.$	5.93 $min.$	90.73 $min.$
DrivAer Notchback w/ GS	40×20×16	7	4.0 mm	1.57×10^{-5}	16 m/s	1	2 $sec.$	7.83 $min.$	126.40 $min.$
DrivAer Estateback w/ GS	40×20×16	7	4.0 mm	1.57×10^{-5}	16 m/s	1	2 $sec.$	8.00 $min.$	120.73 $min.$
Fig. 16	25×8×25	6	5.0 mm	1.6×10^{-4}	10 m/s	1	3.5 $sec.$	2.82 $min.$	32.57 $min.$
Fig. 17	25×12×20	7	3.75 mm	2.0×10^{-5}	55.56 m/s	2	1.7 $sec.$	15.70 $min.$	63.09 $min.$
Fig. 19	7.84×12×4.96	5	3.5 mm	1.6×10^{-5}	5 m/s	1	6 $sec.$	1.72 $min.$	8.80 $min.$

Yun (Raymond) Fei, Qi Guo, Rundong Wu, Li Huang, and Ming Gao. 2021. Revisiting Integration in the Material Point Method: A Scheme for Easier Separation and Less Dissipation. *ACM Trans. Graph.* 40, 4, Article 109 (2021).

Joel H. Ferziger, Milovan Perić, and Robert L Street. 2002. *Computational methods for fluid dynamics*. Vol. 3. Springer.

Olga Filippova and Dieter Hänel. 1998. Grid Refinement for Lattice-BGK Models. *J. Comp. Phys.* 147, 1 (1998), 219–228. https://doi.org/10.1006/jcph.1998.6089

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

Chuyuan Fu, Qi Guo, Theodore Gast, Chenfanfu Jiang, and Joseph Teran. 2017. A Polynomial Particle-in-Cell Method. *ACM Trans. Graph.* 36, 6, Article 222 (2017).

Martin Geier, Andreas Greiner, and Jan G. Korvink. 2006. Cascaded Digital Lattice Boltzmann Automata for High Reynolds Number Flow. *Phys. Rev. E* 73 (2006), 066705. Issue 6.

Martin Geier, Andreas Greiner, and Jan G. Korvink. 2009. A Factorized Central Moment Lattice Boltzmann Method. *The European Physical Journal Special Topics* 171, 1 (2009), 55–61.

Martin Geier, Andrea Pasquali, and Martin Schönherr. 2017a. Parametrization of the Cumulant Lattice Boltzmann Method for Fourth Order Accurate Diffusion Part I: derivation and validation. *J. Comp. Phys.* 348 (2017), 862–888.

Martin Geier, Andrea Pasquali, and Martin Schönherr. 2017b. Parametrization of the cumulant lattice Boltzmann method for fourth order accurate diffusion part II: Application to flow around a sphere at drag crisis. *J. Comput. Phys.* 348 (2017), 889–898.

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

Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. 2015. The Cumulant Lattice Boltzmann Equation in Three Dimensions: theory and validation. *Computers & Mathematics with Applications* 70, 4 (2015), 507–547.

Félix Gendre, Denis Ricot, Guillaume Fritz, and Pierre Sagaut. 2017. Grid refinement for aeroacoustics in the lattice Boltzmann method: A directional splitting approach. *Phys. Rev. E* 96 (2017), 023311. Issue 2.

Irina Ginzburg and Dominique D'Humières. 2003. Multireflection Boundary Conditions for Lattice Boltzmann Models. *Phys. Rev. E* 68 (2003), 066614. Issue 6.

Ehsan Goraki Fard. 2015. *Cumulant LBM approach for Large Eddy Simulation of Dispersion Microsystems*. Ph. D. Dissertation. Technische Universität Braunschweig.

Joseph L. Greathouse and Mayank Daga. 2014. Efficient Sparse Matrix-Vector Multiplication on GPUs Using the CSR Storage Format. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 769–780. https://doi.org/10.1109/SC.2014.68

Francis H Harlow. 1962. *The particle-in-cell method for numerical solution of problems in fluid dynamics*. Technical Report. Los Alamos National Lab, USA.

Manuel Hasert, Kannan Masilamani, Simon Zimny, Harald Klimach, Jiaxing Qi, Jörg Bernsdorf, and Sabine Roller. 2014. Complex Fluid Simulations with the Parallel Tree-based Lattice Boltzmann Solver Musubi. *J. Comp. Sci.* 5, 5 (2014), 784–794.

Angelina Heft, Thomas Indinger, and Nikolaus Adams. 2011. *Investigation of Unsteady Flow Structures in the Wake of a Realistic Generic Car Model*.

Angelina I. Heft, Thomas Indinger, and Nikolaus A. Adams. 2012a. Experimental and Numerical Investigation of the DrivAer Model *(Fluids Engineering Division Summer Meeting, Vol. Volume 1: Symposia, Parts A and B)*. American Society of Mechanical Engineers, 41–51.

Angelina I Heft, Thomas Indinger, and Nikolaus A Adams. 2012b. *Introduction of a new realistic generic car model for aerodynamic investigations*. Technical Report. SAE Technical Paper.

Shuling Hou, James Sterling, Shiyi Chen, and Gary D. Doolen. 1994. A Lattice Boltzmann Subgrid Model for High Reynolds Number Flows. arXiv:comp-gas/9401004.

Yu Hou, David Angland, Alois Sengissen, and Aline Scotto. 2019. Lattice-Boltzmann and Navier-Stokes simulations of the partially dressed, cavity-closed nose landing gear benchmark case. In *25th AIAA/CEAS aeroacoustics conference*. 2555.

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

Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. 2014b. SPH Fluids in Computer Graphics. In *Eurographics - State of the Art Reports*, Sylvain Lefebvre and Michela Spagnuolo (Eds.).

Martin Imre, Jun Tao, and Chaoli Wang. 2017. Efficient GPU-accelerated computation of isosurface similarity maps. In *2017 IEEE Pacific Visualization Symposium*. 180–184.

Peter Irwin, Roy Denoon, and David Scott. 2013. *Wind Tunnel Testing of High-Rise Buildings*. Routledge.

Jérôme Jacob, Orestis Malaspinas, and Pierre Sagaut. 2018. A New Hybrid Recursive Regularised Bhatnagar-Gross-Krook Collision Model for Lattice Boltzmann Method-based Large Eddy Simulation. *Journal of Turbulence* 19, 11-12 (2018), 1051–1076.

Taryn James, Neil Lewington, Lothar Krueger, Manfred Lentzen, Karel Chalupa, Burkhard Hupertz, and Sudesh Woodiga. 2018. Development and Initial Testing of a Full-Scale DrivAer Generic Realistic Wind Tunnel Correlation and Calibration Model. *SAE Int. J. Passenger Cars-Mechanical Systems* 11, 5 (2018), 353–368.

Hrvoje Jasak and Tessa Uroić. 2020. Practical computational fluid dynamics with the finite volume method. In *Modeling in Engineering Using Innovative Numerical Methods for Solids and Fluids*. Springer, 103–161.

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

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.

I. V. Karlin, F. Bösch, and S. S. Chikatamarla. 2014. Gibbs' Principle for the Lattice-kinetic Theory of Fluid Dynamics. *Phys. Rev. E* 90 (2014), 031302. Issue 3.

I. V. Karlin, A. Ferrante, and H. C. Öttinger. 1999. Perfect Entropy Functions of the Lattice Boltzmann Method. *Europhysics Letters (EPL)* 47, 2 (1999), 182–188.

Tero Karras. 2012. Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees. In *Proceedings of the ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*. 33–37.

ByungMoon Kim, Yingjie Liu, Ignacio Llamas, and Jarek Rossignac. 2005. FlowFixer: Using BFECC for Fluid Simulation. In *Eurographics Workshop on Natural Phenomena*. https://doi.org/10.2312/NPH/NPH05/051-056

Manfred Krafczyk, Jonas Tölke, and Li-Shi Luo. 2003. Large-eddy simulations with a multiple-relaxation-time LBE model. *International Journal of Modern Physics B* 17, 01-02 (2003), 33–39.

Andreas Krämer, Dominik Wilde, Knut Küllmer, Dirk Reith, and Holger Foysi. 2019. Pseudoentropic derivation of the regularized lattice Boltzmann method. *Phys. Rev. E* 100 (2019), 023302. Issue 2.

Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlen M. Viggen. 2016. *The Lattice Boltzmann Method: Principles and Practice*. Springer.

Adrian Kummerländer, Márcio Dorn, Martin Frank, and Mathias J. Krause. 2023. Implicit propagation of directly addressed grids in lattice Boltzmann methods. *Concurrency and Computation: Practice and Experience* 35, 8 (2023), e7509.

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.

Daniel Lagrava. 2012. *Revisiting grid refinement algorithms for the lattice Boltzmann method.* Ph. D. Dissertation. Université de Genève.

Daniel Lagrava, Orestis Malaspinas, Jonas Lätt, and Bastien Chopard. 2012. Advances in Multi-domain Lattice Boltzmann Grid Refinement. *J. Comp. Phys.* 231, 14 (2012), 4808–4822.

Pierre Lallemand and Li-Shi Luo. 2000. Theory of the Lattice Boltzmann Method: dispersion, dissipation, isotropy, Galilean invariance, and stability. *Phys. Rev. E* 61 (2000), 6546–6562. Issue 6.

Pierre Lallemand, Li-Shi Luo, Manfred Krafczyk, and Wen-An Yong. 2021. The lattice Boltzmann method for nearly incompressible flows. *J. Comp. Phys.* 431 (2021), 109713. https://doi.org/10.1016/j.jcp.2020.109713

Jonas Lätt and Bastien Chopard. 2006. Lattice Boltzmann Method with Regularized Pre-collision Distribution Functions. *Mathematics and Computers in Simulation* 72, 2 (2006), 165–168.

Jonas Lätt, Orestis Malaspinas, Dimitrios Kontaxakis, Andrea Parmigiani, Daniel Lagrava, Federico Brogi, Mohamed Ben Belgacem, Yann Thorimbert, Sébastien Leclaire, Sha Li, Francesco Marson, Jonathan Lemus, Christos Kotsalos, Raphaël Conradin, Christophe Coreixas, Rémy Petkantchin, Franck Raynaud, Joël Beny, and Bastien Chopard. 2021. Palabos: Parallel Lattice Boltzmann Solver. *Computers & Mathematics with Applications* 81 (2021), 334–350.

Sylvain Lefebvre and Hugues Hoppe. 2006. Perfect Spatial Hashing. *ACM Trans. Graph.* 25, 3 (2006), 579–588.

Moritz Lehmann. 2022. Esoteric Pull and Esoteric Push: Two Simple In-Place Streaming Schemes for the Lattice Boltzmann Method on GPUs. *Computation* 10, 6 (2022).

Wei Li, Kai Bai, and Xiaopei Liu. 2019. Continuous-Scale Kinetic Fluid Simulation. *IEEE Trans. Vis. Comput. 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 Trans. Graph.* 39, 4 (2020).

Wei Li, Yihui Ma, Xiaopei Liu, and Mathieu Desbrun. 2022. Efficient Kinetic Simulation of Two-Phase Flows. *ACM Trans. Graph.* 41, 4, Article 114 (2022), 17 pages.

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. Comp. Phys.* 304 (2016), 424–440.

Daniel S. Lo. 2014. *Finite element mesh generation.* CRC Press.

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.

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 (2021).

E. Lévêque, F. Toschi, L. Shao, and J.-P. Bertoglio. 2007. Shear-improved Smagorinsky model for large-eddy simulation of wall-bounded turbulent flows. *Journal of Fluid Mechanics* 570 (2007), 491–502.

Miles Macklin and Matthias Müller. 2013. Position Based Fluids. *ACM Trans. Graph.* 32, 4, Article 104 (2013).

Orestis Malaspinas. 2015. Increasing Stability and Accuracy of the Lattice Boltzmann Scheme: recursivity and regularization. arXiv:1505.06900 [physics.flu-dyn].

Orestis Malaspinas and Pierre Sagaut. 2011. Advanced large-eddy simulation for lattice Boltzmann methods: The approximate deconvolution model. *Physics of Fluids* 23, 10 (2011), 105103.

O. Malaspinas and P. Sagaut. 2014. Wall model for large-eddy simulation based on the lattice Boltzmann method. *J. Comput. Phys.* 275 (2014), 25–40.

Francesco Marson. 2022. *Directional Lattice Boltzmann Boundary Conditions.* Ph. D. Dissertation. University of Geneva.

Francesco Marson, Yann Thorimbert, Bastien Chopard, Irina Ginzburg, and Jonas Lätt. 2021. Enhanced Single-node Lattice Boltzmann Boundary Condition for Fluid Flows. *Phys. Rev. E* 103 (2021), 053308. Issue 5.

Keijo K. Mattila, Paulo C. Philippi, and Luiz A. Hegele. 2017. High-order Regularization in Lattice-Boltzmann Equations. *Physics of Fluids* 29, 4 (2017), 046103.

Maxon. 2023. Redshift renderer. (2023). https://www.redshift3d.com/product

Renwei Mei, Dazhi Yu, Wei Shyy, and Li-Shi Luo. 2002. Force evaluation in the lattice Boltzmann method involving curved geometry. *Phys. Rev. E* 65 (2002), 041203. Issue 4. https://doi.org/10.1103/PhysRevE.65.041203

Florian R. Menter. 1994. Two-equation Eddy-viscosity Turbulence Models for Engineering Applications. *AIAA Journal* 32, 8 (1994), 1598–1605.

Markus Mohrhard, Gudrun Thäter, Jakob Bludau, Bastian Horvat, and Mathias J. Krause. 2019. Auto-vectorization friendly parallel lattice Boltzmann streaming scheme for direct addressing. *Computers & Fluids* 181 (2019), 1–7.

Faith A. Morrison. 2013. *An introduction to fluid mechanics.* Cambridge University Press.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-Based Fluid Simulation for Interactive Applications. In *Symposium on Computer Animation.* 154–159.

Patrick Nathen, Marc Haussmann, Mathias J. Krause, and Nikolaus A. Adams. 2018. Adaptive filtering for the simulation of turbulent flows with lattice Boltzmann methods. *Computers & Fluids* 172 (2018), 510–523.

X. B. Nie, X. Shan, and H. Chen. 2008. Galilean invariance of lattice Boltzmann models. *Europhysics Letters* 81, 3 (2008), 34005.

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

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

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

Tobias Pfaff, Nils Thuerey, and Markus Gross. 2012. Lagrangian Vortex Sheets for Animating Fluids. *ACM Trans. Graph.* 31, 4, Article 112 (2012), 8 pages.

Kannan N. Premnath, Martin J. Pattison, and Sanjoy Banerjee. 2009. Dynamic subgrid scale modeling of turbulent flows using lattice-Boltzmann method. *Physica A: Statistical Mechanics and its Applications* 388, 13 (2009), 2640–2658.

Ziyin Qu, Minchen Li, Fernando De Goes, and Chenfanfu Jiang. 2022. The Power Particle-in-Cell Method. *ACM Trans. Graph.* 41, 4, Article 118 (2022), 13 pages.

Ziyin Qu, Xinxin Zhang, Ming Gao, Chenfanfu Jiang, and Baoquan Chen. 2019. Efficient and Conservative Fluids Using Bidirectional Mapping. *ACM Trans. Graph.* 38, 4, Article 128 (2019).

Martin Rohde, Drona Kandhai, Jos J. Derksen, and Harry E. A. van den Akker. 2006. A Generic, Mass Conservative Local Grid Refinement Technique for Lattice-Boltzmann Schemes. *International Journal for Numerical Methods in Fluids* 51, 4 (2006), 439–468.

Gianluca Romani, Edoardo Grande, Francesco Avallone, Daniele Ragni, and Damiano Casalino. 2022. Performance and noise prediction of low-Reynolds number propellers using the Lattice-Boltzmann method. *Aerospace Science and Technology* 125 (2022), 107086.

Pierre Sagaut. 2010. Toward Advanced Subgrid Models for Lattice-Boltzmann-based Large-eddy Smulation: theoretical formulations. *Computers & Mathematics with Applications* 59, 7 (2010), 2194–2199.

Florian Schornbaum and Ulrich Rüde. 2016. Massively Parallel Algorithms for the Lattice Boltzmann Method on NonUniform Grids. *SIAM J. Sci. Comput.* 38, 2 (2016), C96–C126.

Florian Schornbaum and Ulrich Rüde. 2018. Extreme-Scale Block-Structured Adaptive Mesh Refinement. *SIAM Journal on Scientific Computing* 40, 3 (2018), C358–C387.

Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35, 2 (2008), 350–371.

Andrew Selle, Nick Rasmussen, and Ronald Fedkiw. 2005. A Vortex Particle Method for Smoke, Water and Explosions. In *Proceedings of ACM SIGGRAPH.* 910–914.

Jung Hee Seo and Rajat Mittal. 2011. A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. *J. Comp. Phys.* 230, 19 (2011), 7347–7363.

Xiaowen Shan. 2019. Central-moment-based Galilean-invariant Multiple-relaxation-time Collision Model. *Phys. Rev. E* 100 (2019), 043308. Issue 4.

Xiaowen Shan and Hudong Chen. 2007. A General Multiple-relaxation-time Boltzmann Collision Model. *International Journal of Modern Physics C* 18, 4 (2007), 635–643.

Ratnesh K Shukla, Mahidhar Tatineni, and Xiaolin Zhong. 2007. Very high-order compact finite difference schemes on non-uniform grids for incompressible Navier-Stokes equations. *J. Comput. Phys.* 224, 2 (2007), 1064–1094.

Siemens PLM Software. 2023. *Simcenter STAR-CCM+ 2210.* Siemens.

Dassault Systèmes Simulia Corp. 2023. *3DEXPERIENCE SIMULIA.* Dassault Systèmes.

Giovanni Solari. 2019. *Wind Science and Engineering: Origins, Developments, Fundamentals and Advancements.* Springer.

B. Solenthaler and R. Pajarola. 2009. Predictive-Corrective Incompressible SPH. In *Proceedings of ACM SIGGRAPH.* Article 40, 6 pages.

Jos Stam. 1999. Stable Fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques.* 121–128.

John C. Strikwerda. 2004. *Finite Difference Schemes and Partial Differential Equations, Second Edition.* SIAM.

Xiangshuo Tang, Yue Yu, and Alparslan Oztekin. 2022. Asymptotic Method for Entropic Multiple Relaxation Time Model in Lattice Boltzmann Method. *Phys. Rev. E* 106 (2022), 015303. Issue 1.

Shi Tao, Qing He, Baiman Chen, Xiaoping Yang, and Simin Huang. 2018. One-point Second-order Curved Boundary Condition for Lattice Boltzmann Simulation of Suspended Particles. *Computers & Mathematics with Applications* 76, 7 (2018), 1593–1607.

Shi Tao, Junjie Hu, and Zhaoli Guo. 2016. An investigation on momentum exchange methods and refilling algorithms for lattice Boltzmann simulation of particulate flows. *Computers & Fluids* 133 (2016), 1–14.

Shashank S. Tiwari, Eshita Pal, Shivkumar Bale, Nitin Minocha, Ashwin W. Patwardhan, Krishnaswamy Nandakumar, and Jyeshtharaj B. Joshi. 2020. Flow past a single stationary sphere, 2. Regime mapping and effect of external disturbances. *Powder Technology* 365 (2020), 215–243.

Jiyuan Tu, Guan Heng Yeoh, and Chaoqun Liu. 2018. *Computational fluid dynamics: a practical approach.* Butterworth-Heinemann.

Milton Van Dyke. 1982. *An album of fluid motion.* The Parabolic Press.

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

Steffen Weißmann and Ulrich Pinkall. 2010. Filament-Based Smoke with Vortex Shedding and Variational Reconnection. In *Proceedings of ACM SIGGRAPH*. Article 115.

Kui Wu, Nghia Truong, Cem Yuksel, and Rama Hoetzlein. 2018. Fast Fluid Simulations with Sparse Volumes on the GPU. *Computer Graphics Forum* 37, 2 (2018), 157–167.

Dazhi Yu, Renwei Mei, and Wei Shyy. 2003. A Unified Boundary Treatment in Lattice Boltzmann Method. In *Aerospace Sciences Meeting and Exhibit*. 953.

Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An Advection-Reflection Solver for Detail-Preserving Fluid Simulation. *ACM Trans. Graph.* 37, 4, Article 85 (2018), 8 pages.

Raoyang Zhang, Xiaowen Shan, and Hudong Chen. 2006. Efficient Kinetic Method for Fluid Simulation beyond the Navier-Stokes Equation. *Phys. Rev. E* 74 (2006), 046703. Issue 4.

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

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

Zheyan Zhang, Yongxing Wang, Peter K Jimack, and He Wang. 2020. MeshingNet: A new mesh generation method based on deep learning. In *International Conference on Computational Science*. Springer, 186–198.

Weifeng Zhao, Juntao Huang, and Wen-An Yong. 2019. Boundary Conditions for Kinetic Theory Based Models I: lattice Boltzmann models. *Multiscale Modeling & Simulation* 17, 2 (2019), 854–872.

Weifeng Zhao and Wen-An Yong. 2017. Single-node Second-order Boundary Schemes for the Lattice Boltzmann Method. *J. Comp. Phys.* 329 (2017), 1–15.

Yongning Zhu and Robert Bridson. 2005. Animating Sand as a Fluid. *ACM Trans. Graph.* 24, 3 (2005), 965–972.

Olgierd C. Zienkiewicz, Robert L. Taylor, and J.Z. Zhu. 2013. *The Finite Element Method: its Basis and Fundamentals*. Butterworth-Heinemann.