High-Order Moment-Encoded Kinetic Simulation of Turbulent Flows

Wei Li, Tongtong Wang, Zherong Pan, Xifeng Gao, Kui Wu, Mathieu Desbrun

High-Order Moment-Encoded Kinetic Simulation of Turbulent Flows

WEI LI and TONGTONG WANG, LightSpeed Studios, China
ZHERONG PAN, XIFENG GAO, and KUI WU, LightSpeed Studios, USA
MATHIEU DESBRUN, Inria / Ecole Polytechnique, France

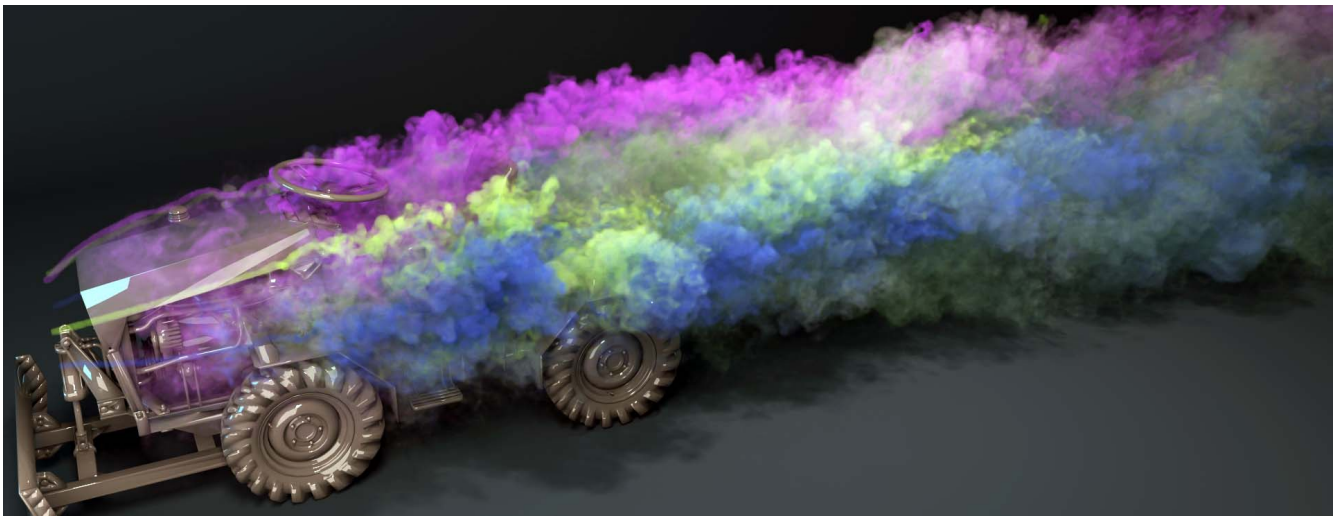


Fig. 1. **Lightweight and lightspeed LBM smulation:** We contribute a new kinetic solver for incompressible and single-phase fluid simulation interacting with complex solids. By relying on only three moments per grid node instead of the usual 27-direction velocity discretization, we reduce memory usage threefold and improve efficiency by an order of magnitude, with limited impact on accuracy. As a consequence, a high-quality visual simulation of a tractor in a wind tunnel (with passively-advected colored particles to visualize the flow) can be achieved efficiently on a commodity GPU.

Kinetic solvers for incompressible fluid simulation were designed to run efficiently on massively parallel architectures such as GPUs. While these lattice Boltzmann solvers have recently proven much faster and more accurate than the macroscopic Navier-Stokes-based solvers traditionally used in graphics, it systematically comes at the price of a very large memory requirement: a mesoscopic discretization of statistical mechanics requires over an order of magnitude more variables per grid node than most fluid solvers in graphics. In order to open up kinetic simulation to gaming and simulation software packages on commodity hardware, we propose a High-Order Moment-Encoded Lattice-Boltzmann-Method solver which we coined HOME-LBM, requiring only the storage of a few moments per grid node, with little to no loss of accuracy in the typical simulation scenarios encountered in graphics. We show that our lightweight and lightspeed fluid solver requires three times less memory and runs ten times faster than state-of-the-art kinetic solvers, for a nearly-identical visual output.

Authors' addresses: W. Li (eduardli@tencent.com): LightSpeed Studios, Shanghai, China; T. Wang (tongttwang@tencent.com): LightSpeed Studios, Shenzhen, China; Z. Rong (zrpan@global.tencent.com) and X. Gao (xifgao@global.tencent.com): LightSpeed Studios, Seattle, USA; K. Wu (kwwu@global.tencent.com): LightSpeed Studios, Los Angeles, USA; M. Desbrun (mathieu.desbrun@inria.fr): Inria Saclay/LIX, Institut Polytechnique de Paris, Palaiseau, France.

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

Additional Key Words and Phrases: Lattice Boltzmann method, velocity moments, turbulent flows, two-way coupling.

ACM Reference Format:
Wei Li, Tongtong Wang, Zherong Pan, Xifeng Gao, Kui Wu, and Mathieu Desbrun. 2023. High-Order Moment-Encoded Kinetic Simulation of Turbulent Flows. *ACM Trans. Graph.* 42, 6 (December 2023), 13 pages. https://doi.org/10.1145/3618341

1 INTRODUCTION

Lattice Boltzmann method (LBM) has become increasingly popular for the visual simulation of turbulent incompressible flows [Li et al. 2003; Thürey and Rüde 2009; Li et al. 2020; Lyu et al. 2021; Li et al. 2022]. Contrary to most early fluid solvers used in computer graphics [Foster and Metaxas 1996; Stam 1999; Mullen et al. 2009; Lentine et al. 2010] which directly seek to integrate the Navier-Stokes equations, LBM is based instead on a kinetic formulation of the flow derived from statistical mechanics: a mesoscopic description of the fluid in time, space, and velocity components is advanced in time through linear streaming (to handle advection) followed by collision (performing a local relaxation towards kinetic equilibrium). The massively-parallel nature of LBM has long been its most attractive feature, as it naturally runs on modern GPUs [Li et al. 2003; Chen et al. 2022] to offer unparalleled computational efficiency compared to conventional solvers. Moreover, the development of accurate collision models has recently led to more accurate results with less

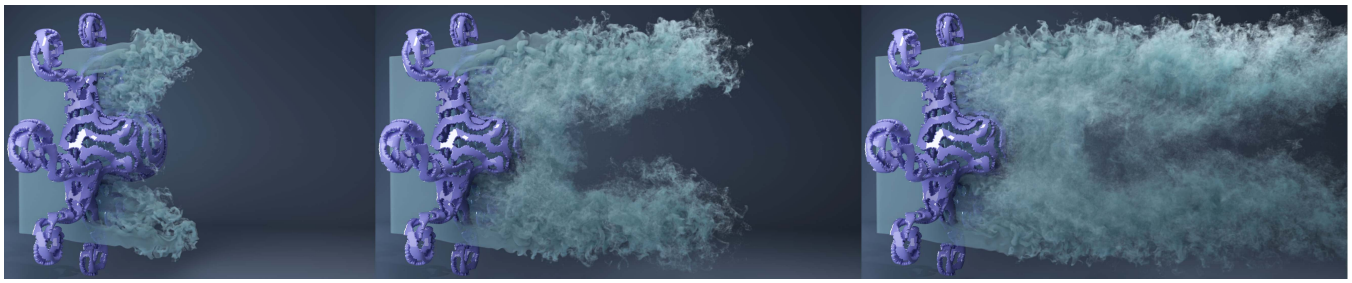


Fig. 2. **Octopus**: A complex object in the shape of an octopus with multiple serpentine grooves is placed in a turbulent wind field. A turbulent wake is formed.

dissipation and dispersion artifacts for complex flow behaviors [Li et al. 2020], in particular for low-viscosity turbulent fluids.

However, all LBM methods in graphics face a steep memory requirement: for a 27-direction velocity discretization as typically needed for simulations of high Reynolds numbers in 3D, LBM solvers often require at the very least 58 values per node (two sets of 27-direction distribution values for time integration and the first two macroscopic moments (density and linear momentum, plus temporary memory usage for conversion to moments), which prevents the use of LBM on small GPU platforms. Although recent methods using efficient swap operations can save 27 variables per node through a more involved streaming process, the stubbornly-high memory size for large simulations not only demands high-end GPUs, but also hampers efficiency by requiring intensive memory access.

In this paper, we introduce a lightweight LBM solver which significantly cuts down on memory size and improves efficiency compared to previous methods, without incurring a noticeable drop in accuracy for typical visual flow simulations. Our approach borrows from recent advances on moment-representation LBM (MR-LBM) to bypass the use of a fine directional velocity sampling, while also remedying their compressibility issues at high Reynolds numbers due to their low-order collision model. In addition, we also propose a new moment-based boundary treatment for fast two-way coupling. We demonstrate that the resulting lightspeed and lightweight LBM solver ends up being three times smaller in memory footprint and an order of magnitude faster than [Li et al. 2020] for the same resolution and a nearly-identical visual output.

2 RELATED WORK AND MOTIVATION

While fluid simulation with fluid-solid coupling has been achieved in both computer graphics (CG) and computational fluid dynamics (CFD) in a variety of ways, we briefly review the most related works to our contributions to single-phase flow simulation.

2.1 Navier-Stokes based fluid solvers

From early Eulerian grid discretiations [Foster and Metaxas 1996; Stam 1999], fluid solvers in CG developed quickly. Advanced grid-based approaches were designed [Losasso et al. 2004; Lentine et al. 2010; Zhu et al. 2013; Setaluri et al. 2014; Klingner et al. 2006; Mullen et al. 2009; Ando et al. 2013] to improve accuracy and versatility. Lagrangian particles have also offered a popular method to simulate fluids [Desbrun and Gascuel 1996; Becker and Teschner 2007; Solenthaler and Pajarola 2009; Akinci et al. 2012; Ihmsen et al. 2014; Peer et al. 2015], although simulating turbulence requires a large number of particles. Driven by the need for more vortical details, a series of vorticity-based methods [Park and Kim 2005; Golas et al. 2012; Selle et al. 2005; Weißmann and Pinkall 2010; Brochu et al. 2012; Pfaff et al. 2012; Zhang and Bridson 2014; Zhang et al. 2015] or hybrid approaches [Zhu and Bridson 2005; Raveendran et al. 2011; Jiang et al. 2015; Zhang et al. 2016] were later devised as well.

The ability to support fluid-solid coupling is also crucial to simulate complex fluid-solid interactions. Early coupling works included voxelized boundary approximations [Takahashi et al. 2002; Génevaux et al. 2003; Robinson-Mosher et al. 2008; Azevedo et al. 2016], a rigid-fluid approach [Carlson et al. 2004], a fully-Eulerian method [Teng et al. 2016], or even coupling that can treat infinitesimally thin and deformable solids [Guendelman et al. 2005] by enforcing proper solid-boundary conditions to keep the fluid incompressible and get the correct interaction force between fluid and solids. Cut-cell-based approaches [Roble et al. 2005; Batty et al. 2007; Ng et al. 2009; Gibou and Min 2012; Weber et al. 2015; Edwards and Bridson 2014; Azevedo et al. 2016; Tao et al. 2022] have also been formulated to efficiently deal with thin structures and complex geometry by subdividing them into boundary-conforming regions. However, none of these methods have demonstrated stable coupling in very turbulent flows. Particle-based solvers [Colagrossi and Landrini 2003; Solenthaler and Pajarola 2008; Akinci et al. 2012; Schechter and Bridson 2012; de Goes et al. 2015; Bender and Koschier 2016; Band et al. 2017; Koschier and Bender 2017; Becker et al. 2009; Ihmsen et al. 2013; Fang et al. 2020] typically approximate solid boundaries with a dense set of particles, but often face pressure inconsistencies [Band et al. 2018], explaining why hybrid grid-particle solvers [Zhang et al. 2016; Fei et al. 2018; Hu et al. 2018; Fei et al. 2021] usually treat boundary conditions via grid-based methods.

2.2 Boltzmann-based fluid solvers

LBM has recently been proven a great alternative to traditional incompressible Navier-Stokes based solvers for both single [Li et al. 2020; Lyu et al. 2021] and multiphase [Li et al. 2021, 2022; Li and Desbrun 2023] fluid simulation. While early works [Li et al. 2003; Thürey and Rüde 2009] were based on the low-order BGK collision model [Chen and Doolen 1998], LBM solvers in graphics significantly improved their accuracy for turbulent flows by constructing higher-order collision models. For instance, Li et al. [2020] proposed a central-moment multiple-relaxation-time (MRT) collision model where non-physical rates were optimized for reduced dispersion and dissipation. Moreover, in order to improve computational efficiency on GPU [Li et al. 2003], Chen et al. [2022] and Lehmann [2022] provided a number of detailed strategies to enable improved parallelism. As a result, current methods have reached impressive capabilities

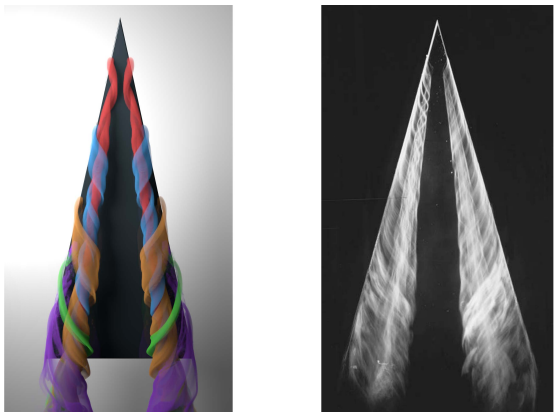


Fig. 3. **Delta wing**. The airflow over a thin-shell delta wing is simulated with HOME-LBM (left), exhibiting the stable spiral vortex structure near the leading edge of the wing seen in experiments [Délery 2001] (right).

to simulate both laminar and turbulent flows, with both numerical accuracy and efficiency surpassing even the most recent Navier-Stokes solvers. Fluid/structure coupling was also tackled in previous LBM works, using the immersed boundary method (IBM [Peskin 1972]) or the moving bounce-back boundary method (BBM [Ladd 1994]); Lyu et al. [2021] also suggested a velocity correction to allow for thin obstacles and suppress the usual ringing artifacts of BBM.

2.3 Motivation and overview

The recent upending of Navier-Stokes fluid solvers due to the emergence of high-order accurate LBM solvers has not yet permeated the realm of video games or simulation software packages using commodity hardware, because all current kinetic methods face a huge memory cost: since a D3Q27 model (i.e., a discretization of velocities in 27 directions) is needed to reliably handle turbulent flows, a typical implementation requires a minimum of 58 distribution values per grid node [Li et al. 2020], including two copies of the 27-direction distribution values for time integration and the important macroscopic quantities (density and velocity). While various streaming strategies can remove up to 27 variables per node [Latt 2007], this still is *eight to ten times more than a typical Navier-Stokes solver*, requiring high-end GPUs for detailed simulations with fine grids. Very recently, Vardhan et al. [2019] and Ferrari et al. [2023] proposed encoding a 27-direction distribution via *only its few first moments* in order to replace the memory-hungry distribution-based representation. This moment-representation LBM (MR-LBM) only uses 10 variables per node, greatly reducing memory size requirements and inducing a performance uplift between 25% and 40% on GPU due to far reduced memory access. However, this improvement in efficiency and memory size comes with a severe limitation: this lightweight approach can no longer support turbulent flows ($Re \geq 4000$) reliably, due to limited accuracy.

This paper leverages the moment-based representation of LBM initially proposed in [Vardhan et al. 2019; Ferrari et al. 2023], while addressing its current limitations. We show that one can use a lightweight moment representation to replace the large storage of direction-based distributions used in current LBM methods in graphics while enforcing a high-order accurate collision at low computational cost. We also formulate a new fluid-solid coupling scheme based on this moment-based LBM approach to generate complex two-way phenomena. Reduced memory usage along with fast moment-based collision and coupling result in a lightweight and lightspeed kinetic solver with limited accuracy loss compared to state-of-the-art LBM solvers.

3 BACKGROUND

Before introducing our new moment-based LBM scheme, we briefly review current single-phase LBM methods in graphics, and introduce existing moment-based representation LBM methods, while pointing out their stability and accuracy issues.

3.1 Lattice Boltzmann method at a glance

Continuous Boltzmann equation. In statistical mechanics, fluid dynamics is described by the time evolution of a mesoscopic distribution function $\boldsymbol{f}(\boldsymbol{v}, \boldsymbol{x}, t)$ representing the probability of particle being at position $\boldsymbol{x}$ at time t and with velocity $\boldsymbol{v}$. This is in sharp contrast to typical NS-based method modeling the flow dynamics with the time evolution of a macroscopic velocity $\boldsymbol{u}(\boldsymbol{x}, t)$. In the context of single-phase fluid dynamics, the governing kinetic equation for the evolution of the distribution function is known as the Boltzmann equation [Shan et al. 2006]:

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

where $\boldsymbol{F}$ represents external forces and Ω is the collision term which relaxes the distribution function towards the local thermodynamic equilibrium state $\boldsymbol{f}^{eq}$ [Coreixas et al. 2017] defined as

$$\boldsymbol{f}^{\mathrm{eq}} = \frac{\rho}{(2\pi r T)^{d/2}} \exp\!\left(-\frac{\|\boldsymbol{v}-\boldsymbol{u}\|^2}{2rT}\right), \tag{2}$$

where r is a gas constant, T is the thermodynamic temperature and d is the spatial dimension (2, or 3). The macroscopic quantities such as density ρ, linear momentum $\rho\boldsymbol{u}$, and the rank-two tensor $\boldsymbol{S}$ related to the stress tensor can be recovered from the distribution through:

$$\rho = \int \boldsymbol{f} \, \mathrm{d}\boldsymbol{v}, \quad \rho\boldsymbol{u} = \int \boldsymbol{v}\, \boldsymbol{f} \, \mathrm{d}\boldsymbol{v}, \quad \rho S_{\alpha\beta} = \int (\boldsymbol{v}^2 - \frac{1}{3}\delta_{\alpha\beta})\, \boldsymbol{f} \, \mathrm{d}\boldsymbol{v} \,, \tag{3}$$

where the greek indices α and β refer to tensor coordinates, i.e., $\boldsymbol{S} = \{S_{\alpha\beta}\}_{\alpha,\beta}$ for $\alpha, \beta \in \{x,y,z\}$.

The original lattice Boltzmann method (LBM) used the Bhatnagar-Gross-Krook (BGK) collision model $\Omega(\boldsymbol{f}) = -(\boldsymbol{f} - \boldsymbol{f}^{\mathrm{eq}})/\tau$ (where τ is the relaxation time determining how fast the equilibrium is being reached, thus related to the kinematic viscosity ν via $\tau = 3\nu + \frac{1}{2}$), which preserves density and first-order momentum. Even with this simplistic collision model, Eq. (1) was proven to recover Navier-Stokes equation macroscopically [Shan et al. 2006].

Lattice Boltzmann equations. With time t discretized through regular timesteps, space $\boldsymbol{x}$ discretized through a regular grid, and the mesoscopic velocity $\boldsymbol{v}$ discretized at each grid node through a lattice structure with q directions as shown in Fig. 4, one can turn the continuous Eq. (1) into lattice Boltzmann equations (LBE) in dimensionless units [Li et al. 2020] yielding:

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

where $f_i(\boldsymbol{x}, t)$ encodes the distribution f in the i-th direction at position $\boldsymbol{x}$ and time t, $\boldsymbol{c}_i$ is the discrete lattice velocity in the i-th direction (Fig. 4), Ω_i is the discretized collision operator, and F_i results from external forces projected on distribution space. Through

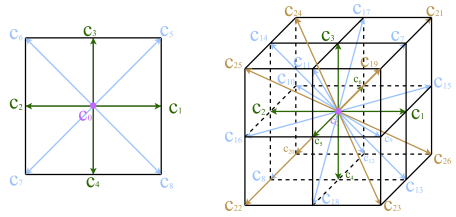


Fig. 4. **Lattice structures**: In LBM, distribution functions are often discretized on a D2Q9 lattice (left) or on a D3Q27 (right) in 3D, offering a discretization of both space via grid nodes and velocity via (orange, blue, and green) links which properly resolves turbulent flows.

operator splitting, Eqs. (4) can be solved in two steps: first, the distribution values are advected via *streaming* by computing

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

followed by a collision step expressed as

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

Macroscopic quantities are then updated using discretized versions of Eqs. (3), which, when incorporating external forces, read:

$$\rho = \sum_{i=0}^{q-1} f_i, \quad \rho\boldsymbol{u} = \sum_{i=0}^{q-1} \boldsymbol{c}_i f_i + \frac{1}{2}\boldsymbol{F}, \quad \rho S_{\alpha\beta} = \sum_{i=0}^{q-1} (\boldsymbol{c}_i^2 - \frac{1}{3}\delta_{\alpha\beta})\, f_i\,. \tag{7}$$

Collision models. While the BGK collision model described above makes an LBM solver provably equivalent to the Navier-Stokes equations for a fine enough discretization, its first-order nature was recognized as a major limitation for a fixed grid size. In recent years, many involved collision models have been formulated which offer higher-order accuracy. In particular, the non-orthogonal central-moment multiple-relaxation-time model (NOCM-MRT [De Rosis 2017; De Rosis and Luo 2019]) and the cumulant-based model (CU-MRT [Geier et al. 2015, 2017]) have both been proven to be particularly accurate collision models. For instance, Li et al. [2020] and Lyu et al. [2021] leveraged a NOCM-MRT collision model where the distribution function is mapped onto a central moment space and each moment is relaxed at a different rate before reprojecting the post-collision moments back into a distribution function; a further optimization of higher-order (non-physical) relaxation rates can also be added to offer low-diffusion and low-dispersion simulations. Very recently, Lyu et al. [2023] show that a cumulant model, which performs collision by mapping onto a cumulant space this time, can provide even higher accuracy for very high Reynolds numbers if higher-order cumulants are optimized to maximize entropy.

3.2 Towards lightweight LBM

However, all single-phase turbulent fluid simulations in computer graphics thus far need to store two distribution functions for their temporal update and the first two macroscopic quantities from Eq. (7), leading to at least 58 variables per node — without considering extra temporary memory used, for instance, for the projection from distribution to moment spaces in the evaluation of the collision terms. While more complex in-place streaming methods (such as the swap approach of Latt [2007] or the esoteric twist of Geier and Schönherr [2017]) can remove a copy of the distribution function, typical memory requirements for high-resolution simulations rule out the use of consumer GPUs.

A moment representation for LBM (MR-LBM [Vardhan et al. 2019; Gounley et al. 2022; Ferrari et al. 2023]) has recently been formulated. It requires far fewer variables per node, thus relaxing memory requirements at least threefold compared to LBM solvers in graphics such as [Li et al. 2020], seemingly ensuring its applicability to consumer GPUs. The MR-LBM consists of three steps: (a) from the three stored moments ρ, $\rho\boldsymbol{u}$, $\rho\boldsymbol{S}$, a distribution function is reconstructed via (we use Einstein notation to indicate tensor contraction):

$$f_i(\boldsymbol{x},t) = \rho(\boldsymbol{x}-\boldsymbol{c}_i,t)\, w_i \left(1 + 3\boldsymbol{u}(\boldsymbol{x}-\boldsymbol{c}_i,t)\cdot\boldsymbol{c}_i + \tfrac{1}{2c_s{}^4} S_{\alpha\beta}(\boldsymbol{x}-\boldsymbol{c}_i,t) H^{[2]}_{\alpha\beta}(\boldsymbol{c}_i)\right), \tag{8}$$

where w_i are lattice weights and $\boldsymbol{H}^{[2]} = \{H^{[2]}_{\alpha\beta}\}_{\alpha,\beta}$ represents the Hermite polynomials of order 2; (b) the reconstructed distribution function is then streamed using Eq. (5); (c) the resulting distribution is converted back to its three moments using Eqs. (7), and the BGK collision model and external forces are incorporated (effectively replacing Eq. (6)) by simply updating the current moments through:

$$\rho u_\alpha(\boldsymbol{x},t+1) = \rho u_\alpha(\boldsymbol{x},t) + \left(1-\tfrac{1}{2\tau}\right) F_\alpha\,, \tag{9}$$

$$\rho S_{\alpha\beta}(\boldsymbol{x},t+1) = (1-\tfrac{1}{\tau})\rho S_{\alpha\beta}(\boldsymbol{x},t) + \tfrac{1}{\tau}\rho u_\alpha u_\beta(\boldsymbol{x},t) + \left(1-\tfrac{1}{2\tau}\right)(F_\alpha u_\beta + F_\beta u_\alpha)(\boldsymbol{x},t). \tag{10}$$

For simplicity, in this paper, we call the first step (a) the "distribution reconstruction" step, the second step (b) "regular streaming" while the third will be denoted by the "moment-based collision" step.

3.3 Discussion

While the MR-LBM uses about a third less memory than a traditional LBM and about 50% of the memory required by an in-place implementation, it cannot handle flows with Reynolds numbers above 4000. There are two reasons for this severe limitation. First, in the distribution reconstruction step, a *second-order reconstruction* is used, which introduces truncation errors to macroscopic moments for turbulent flows. Second, the moment-based collision step assumes a BGK collision model which is well known to be very limited as it is only first-order accurate, often bringing large errors into macroscopic moments. As a result, the gain in memory size comes with a stringent limitation on the viscosity of the fluid: one cannot produce the high Reynolds number flows that a regular LBM implementation is typically successful at generating.

4 HIGH-ORDER MOMENT-ENCODED LBM

We now introduce our High-Order Moment-Encoded LBM (HOME-LBM) approach which removes most of the shortcomings of MR-LBM by modifying its first and third steps to gain substantial accuracy and stability. In this section, we will assume a D3Q27 lattice discretization in 3D (resp., D2Q9 in 2D) as typically recommended for turbulent flows to ensure numerical accuracy.

4.1 Hermite polynomials expansion

Continuous expressions. As a reminder, distribution functions in LBM are traditionally expressed through Hermite series expansions [Shan et al. 2006]:

$$f(\boldsymbol{v},\boldsymbol{x},t) = \omega(\boldsymbol{v}) \sum_{n=0}^{N} \frac{1}{n!} \boldsymbol{H}^{[n]}(\boldsymbol{v}) : \boldsymbol{a}^{[n]}(\boldsymbol{x},t)\,, \tag{11}$$

where N is the order of approximation; $\omega(\boldsymbol{v})$ is a weighting function defined as $\omega(\boldsymbol{v}) = \exp(-\|\boldsymbol{v}\|^2/2)/(2\pi)^{d/2}$; the superscript "$[n]$" indicates a rank-$n$ tensor; the operator ":" denotes full tensor contraction; and $\boldsymbol{H}^{[n]}$ is the Hermite polynomial of order n, a n-th order tensor defined as

$$\boldsymbol{H}^{[n]}(\boldsymbol{v}) = \frac{(-1)^n}{\omega(\boldsymbol{v})} \nabla^n \omega(\boldsymbol{v}). \tag{12}$$

Since Hermite polynomials form an orthonormal basis (for the L_2 inner product weighted by ω), the coefficient $\boldsymbol{a}^{[n]}(\boldsymbol{x}, t)$ of a given distribution f can be computed via a weighted inner product:

$$\boldsymbol{a}^{[n]}(\boldsymbol{x}, t) = \int \frac{f(\boldsymbol{v}, \boldsymbol{x}, t)}{\omega(\boldsymbol{v})} \boldsymbol{H}^{[n]}(\boldsymbol{v}) \mathrm{d}\boldsymbol{v} \ . \tag{13}$$

Note that the first three orders of coefficients coincide, in fact, with the three first *velocity moments* from Eq. (3) since

$$\boldsymbol{a}^{[0]} = \rho \ , \quad \boldsymbol{a}^{[1]} = \rho \boldsymbol{u} \ , \quad \text{and} \quad \boldsymbol{a}^{[2]} = \rho \boldsymbol{S} \ . \tag{14}$$

Discrete distributions. Now, when a lattice discretization of the local velocity $\boldsymbol{v}$ into q discrete microscopic velocities $\{\boldsymbol{c}_i\}_{i=0}^{q-1}$ ($q=9$ in 2D, $q=27$ in 3D, see Fig. 4) is used and using the lattice weights w_i from Eq. 8 as quadrature weights, the *discrete* distribution function corresponding to $f(\cdot, \boldsymbol{x}, t)$ are the q values $\{f_i\}_{i=0}^{q-1}$ per node and per timestep defined as:

$$f_i(\boldsymbol{x}, t) = \frac{w_i}{\omega(\boldsymbol{c}_i)} f(\boldsymbol{c}_i, \boldsymbol{x}, t), \tag{15}$$

Eq. (13) is then evaluated through Gauss-Hermite quadrature as:

$$\boldsymbol{a}^{[n]}(\boldsymbol{x}, t) = \sum_{i=0}^{q-1} f_i(\boldsymbol{x}, t) \boldsymbol{H}^{[n]}(\boldsymbol{c}_i) \ . \tag{16}$$

One can verify that the first three orders of coefficients $\boldsymbol{a}^{[0]}, \boldsymbol{a}^{[1]}$, and $\boldsymbol{a}^{[2]}$ corresponds to the discrete moments from Eqs. (7).

4.2 High-order distribution reconstruction

In the MR-LBM approach [Vardhan et al. 2019; Gounley et al. 2022; Ferrari et al. 2023], discrete distribution functions are not stored on the computational grid to avoid large memory footprint, but replaced by only the first three velocity moments ρ, $\rho\boldsymbol{u}$, and $\rho\boldsymbol{S}$ at each grid node instead. From these moments, a second-order Hermite approximation of the distribution of the form of Eq. (11) that matches these velocity moments is derived by simply choosing only three non-zero coefficients $(\boldsymbol{a}^{[0]}, \boldsymbol{a}^{[1]}, \boldsymbol{a}^{[2]})$ in the Hermite expansion and setting these coefficients to the three known moments (Eq. (14)). This low-order reconstruction of the 27-direction values f_i leads to a very simple expression of the distribution function given in Eq. (8) — and hence, a very cheap reconstruction — but introduces significant truncation errors due to its particularly low-order in the case of turbulent flows, making it ill-adapted to our needs.

Regularized distributions. Instead, we propose to leverage existing works to reconstruct a third-order Hermite approximation of the distribution which will improve the precision and numerical stability of our HOME-LBM. Latt and Chopard [2006] introduced the concept of *regularized distribution* for standard LBM, which stemmed from making sure that the BGK collision model leads to the correct macroscopic equations in the hydrodynamic limit, by *filtering* distribution functions after streaming and before collision to remove unwanted oscillations ("ghost modes") and instabilities in the simulation. From the current distribution $\boldsymbol{f}$ of a node, the first three velocity moments ρ, $\boldsymbol{u}$, and $\boldsymbol{a}^{[2]}$ are evaluated, and a particular choice of a second-order Hermite truncated series for the off-equlibrium part $\boldsymbol{f}^{\text{off}}$ (i.e., the original distribution minus the equilibrium distribution $\boldsymbol{f}^{\text{eq}}(\rho, \boldsymbol{u})$) is chosen to ensure that the regularized distribution satisfies key symmetries. Malaspinas [2015] proposed another regularization, now relying on a full sixth-order Hermite series (in 3D), where the various coefficients $\boldsymbol{a}^{[n>2]}$ are recursively evaluated from ρ, $\boldsymbol{u}$, and $\boldsymbol{a}^{[2]}$ to remove the typical numerical inaccuracies of high-order moments.

Third-order Hermite reconstruction. Compared to the previous work mentioned above, we are *not trying to filter an existing distribution, but to reconstruct one instead* — but we can use their approach to turn our three velocity moments into a valid distribution function. While regularizations are often based on second-order Hermite truncations or full sixth-order hermite series, we opt for a third-order Hermite truncation to offer a good balance between accuracy and computational efficiency. Since the moment representation of LBM stores ρ (to which we set $\boldsymbol{a}^{[0]}$), $\rho\boldsymbol{u}$ (to which we set $\boldsymbol{a}^{[1]}$), and $\rho\boldsymbol{S}$ (to which we set $\boldsymbol{a}^{[2]}$), we can use the recursive computation of third-order coefficients $\boldsymbol{a}^{[3]}$ from Malaspinas [2015] *as is* for our reconstruction of a third-order Hermite approximation of the distribution. The advantages are obvious: due to their regularization procedure, the reconstructed distribution is void of significant ghost modes; but our truncation at the third term of the Hermite expansion offers improved computational efficiency, obviously.

Reconstruction in closed-form. While the recursive evaluation of $\boldsymbol{a}^{[n>2]}$ from [Malaspinas 2015] can be used, we can also express the resulting coefficients directly in closed form as a function of the first three velocity moments ρ, $\rho\boldsymbol{u}$, and $\rho\boldsymbol{S}$, leading in 3D to the following third-order Hermite reconstruction expression for each of the distribution function f_i:

$$\begin{aligned} f_i = \rho w_i \Bigg[& 1 + \frac{\boldsymbol{c}_i \cdot \boldsymbol{u}}{c_s^2} + \frac{\boldsymbol{H}^{[2]}(\boldsymbol{c}_i) : \boldsymbol{S}}{2c_s^4} \\ & + \tfrac{1}{2c_s^6} \big(\ H^{[3]}_{xxy}(\boldsymbol{c}_i)(S_{xx}u_y + 2S_{xy}u_x - 2u_x u_x u_y) \\ & \quad + H^{[3]}_{xyy}(\boldsymbol{c}_i)(S_{yy}u_x + 2S_{xy}u_y - 2u_x u_y u_y) \\ & \quad + H^{[3]}_{xxz}(\boldsymbol{c}_i)(S_{xx}u_z + 2S_{xz}u_x - 2u_x u_x u_z) \\ & \quad + H^{[3]}_{xzz}(\boldsymbol{c}_i)(S_{zz}u_x + 2S_{xz}u_z - 2u_x u_z u_z) \\ & \quad + H^{[3]}_{yzz}(\boldsymbol{c}_i)(S_{zz}u_y + 2S_{yz}u_z - 2u_y u_z u_z) \\ & \quad + H^{[3]}_{yyz}(\boldsymbol{c}_i)(S_{yy}u_z + 2S_{yz}u_z - 2u_y u_y u_z) \\ & \quad + H^{[3]}_{xyz}(\boldsymbol{c}_i)(S_{xz}u_y + S_{yz}u_x + S_{xy}u_z - 2u_x u_y u_z) \big) \Bigg]. \end{aligned} \tag{17}$$

We also provide the formula for the 2D reconstruction of a distribution from the three moments in App. B (see Eq. (29)), as well as all the expressions of the Hermite terms in App. A to ease implementation.

4.3 High-order collision model

The original MR-LBM relies on the BGK collision model, whose first-order nature can only handle low Reynolds numbers. For turbulent flows, one needs a higher-order collision model instead. Multiple relaxation time (MRT) models, which convert the distribution function into a central-moment or cumulant space in which each component gets relaxed towards its equilibrium with an individual rate before

Fig. 5. **Wind through a fern**: A static 3D model of a fern is placed in a strong flow field, generating a complex wake (visualized with particles).

being converted back, have been successfully used in graphics in recent years, be it for single-phase [Li et al. 2020; Lyu et al. 2021, 2023] or multi-phase [Li et al. 2021, 2022; Li and Desbrun 2023] flow simulation. We incorporate an MRT moment-based collision step in HOME-LBM too in order to ensure higher accuracy.

Central-moment collision in regular LBM. An MRT-based approach to collision, named NOCM-MRT, uses non-orthogonal central moments $\boldsymbol{m}=\{m_i\}_{i=0..26}$, which are derived from distribution functions $\boldsymbol{f}=\{f_i\}_{i=0..26}$ via a linear transform $\boldsymbol{m}=\boldsymbol{M}\boldsymbol{f}$, where $\boldsymbol{M}$ is a matrix known in closed form as a function of the macroscopic velocity $\boldsymbol{u}$. By subtracting the equilibrium distribution projected into the same central moment space, one gets the off-equilibrium central moments $\boldsymbol{m}-\boldsymbol{m}^{\mathrm{eq}}$, which are relaxed by individual rates r_i, before the result is converted back to a distribution by multiplying it by $\boldsymbol{M}^{-1}$, which provides the collision terms Ω_i from Eq. (4) — see, for instance, [Li et al. 2020]. If one also incorporates external forces, the NOCM-MRT collision is expressed as:

$$\boldsymbol{\Omega} = -\boldsymbol{M}^{-1}\big(\boldsymbol{R}(\boldsymbol{m}-\boldsymbol{m}^{\mathrm{eq}}) + (\boldsymbol{I}-\tfrac{1}{2}\boldsymbol{R})\boldsymbol{K}\big), \tag{18}$$

where $\boldsymbol{\Omega}$ comprises all collision terms Ω_i, $\boldsymbol{R}$ is the diagonal matrix containing all individual relaxation rates r_i, and $\boldsymbol{K}$ represents the force terms projected into central-moment space.

Collision in HOME-LBM. In order to improve efficiency, we propose an alternate evaluation based on our velocity moments, still affording a high-order collision evaluation. After streaming the reconstructed distribution, we directly compute the three new moment ρ^*, $\rho^*\boldsymbol{u}^*$, and $\rho^*\boldsymbol{S}^*$ for each of the resulting distribution per node based on Eq. (7) *before the collision step.* (The use of the superscript asterisk indicates that these moments are temporary: they will be altered by the collision.) Knowing ρ^* and $\boldsymbol{u}^*$ on a grid node allows us to know the equilibrium distribution defined by Eq. (2). The resulting central-moment $\boldsymbol{m}^{\mathrm{eq}}=\boldsymbol{M}\boldsymbol{f}^{\mathrm{eq}}$ and the external force $\boldsymbol{F}$ are then projected into the Hermite-based form of Eq. (11): following [Li et al. 2020], we use a sixth-order Hermite expansion here, leading to mostly *zero* terms except for:

$$m_0^{\mathrm{eq}} = m_9^{\mathrm{eq}} = \rho,\ m_{17}^{\mathrm{eq}} = \tfrac{1}{3}\rho,\ m_{18}^{\mathrm{eq}} = \tfrac{1}{9}\rho\ ,\ m_{26}^{\mathrm{eq}} = \tfrac{1}{27}\rho \tag{19}$$

for the central moments, and, for the force-related terms,

$$\begin{aligned} K_1 &= F_x, & K_2 &= F_y, & K_3 &= F_z, \\ K_{10} &= \tfrac{2}{3}F_x, & K_{11} &= \tfrac{2}{3}F_y, & K_{12} &= \tfrac{2}{3}F_z, \\ K_{23} &= \tfrac{1}{9}F_x, & K_{24} &= \tfrac{1}{9}F_y, & K_{25} &= \tfrac{1}{9}F_z\ . \end{aligned} \tag{20}$$

To evaluate the actual post-streaming distribution f from the updated velocity moments, we chose a third-order Hermite expansion to improve upon previous works (which all used a second-order one) while keeping the collision evaluation simple. Indeed, the resulting effect of the collision encoded by Eq. (18) ends up being quite straightforward: the post-collision velocity moments do not require the lengthy evaluation of (and multiplications by) matrices $\boldsymbol{M}$ and its inverse, as the closed-form expressions of the three velocity moments (found with Matlab [Matlab 2023]) are

$$\rho(\boldsymbol{x}, t+1) = \rho^*\,; \tag{21}$$

$$u_\alpha(\boldsymbol{x}, t+1) = u_\alpha^* + \tfrac{1}{2\rho^*}F_\alpha\,; \tag{22}$$

$$S_{xy}(\boldsymbol{x}, t+1) = (1-\tfrac{1}{\tau})S_{xy}^* + \tfrac{1}{\tau}u_x^*u_y^* + \tfrac{2\tau-1}{2\tau\rho^*}(F_xu_y^* + F_yu_x^*)\,; \tag{23}$$

$$\begin{aligned} S_{xx}(\boldsymbol{x}, t+1) &= \tfrac{\tau-1}{3\tau}(2S_{xx}^* - S_{yy}^* - S_{zz}^*) + \tfrac{1}{3}\left(u_x^{*2} + u_y^{*2} + u_z^{*2}\right) \\ &+ \tfrac{1}{3\tau}\left(2u_x^{*2} - u_y^{*2} - u_z^{*2}\right) + \tfrac{1}{\rho^*}F_xu_x^* + \tfrac{\tau-1}{3\tau\rho^*}(2F_xu_x^* - F_yu_y^* - F_zu_z^*)\,; \\ S_{yy}(\boldsymbol{x}, t+1) &= \tfrac{\tau-1}{3\tau}(2S_{yy}^* - S_{xx}^* - S_{zz}^*) + \tfrac{1}{3}\left(u_x^{*2} + u_y^{*2} + u_z^{*2}\right) \\ &+ \tfrac{1}{3\tau}\left(2u_y^{*2} - u_x^{*2} - u_z^{*2}\right) + \tfrac{1}{\rho^*}F_yu_y^* + \tfrac{\tau-1}{3\tau\rho^*}(2F_yu_y^* - F_xu_x^* - F_zu_z^*)\,; \\ S_{zz}(\boldsymbol{x}, t+1) &= \tfrac{\tau-1}{3\tau}(2S_{zz}^* - S_{xx}^* - S_{yy}^*) + \tfrac{1}{3}\left(u_x^{*2} + u_y^{*2} + u_z^{*2}\right) \\ &+ \tfrac{1}{3\tau}\left(2u_z^{*2} - u_x^{*2} - u_y^{*2}\right) + \tfrac{1}{\rho^*}F_zu_z^* + \tfrac{\tau-1}{3\tau\rho^*}(2F_zu_z^* - F_xu_x^* - F_yu_y^*)\,. \end{aligned}$$

Note that these expressions (except for the trivial Eqs. (21)-(22) which just reflect density and linear momentum preservation of the collision process) are far simpler than having to deal with matrix $\boldsymbol{M}$ and its inverse in the usual LBM collision process: therefore, the moment-encoded LBM provides a great boost in efficiency due to these simple collision updates, see Fig. 12. For completeness, we provide the expressions for the 2D case in App. C.

Cumulant-based collision model. We can also proceed similarly to incorporate the cumulant-based collision model into our moment-encoded solver; we provide the resulting closed-form expressions in App. D. Due to the more involved nature of the cumulant model (i.e., its non-linear projection), the update rules are three times as long. Therefore, all our results use the central-moment based collision (Eqs. (21-23)), except for Fig. 8 where we test both models.

4.4 Moment-based single-node coupling

Now that we have addressed how to integrate the lattice Boltzmann equations (Eqs. (6)) in our moment-encoded LBM context, the last issue to tackle is fluid-solid coupling. One of the simplest approaches to coupling is the moving bounce-back method [Ladd 1994], which alters the streaming process near a boundary by reversing the distribution function advection against the boundary and applying a momentum exchange based on the velocity of the solid encountered. However, its first-order nature often leads to spurious oscillations especially in turbulent flows, which spurred the introduction of a hybrid velocity-correction method [Lyu et al. 2021] to offer stable two-way coupling, at the cost of larger stencils. We favor a more direct and local approach that requires less computations by leveraging our high-order distribution reconstruction.

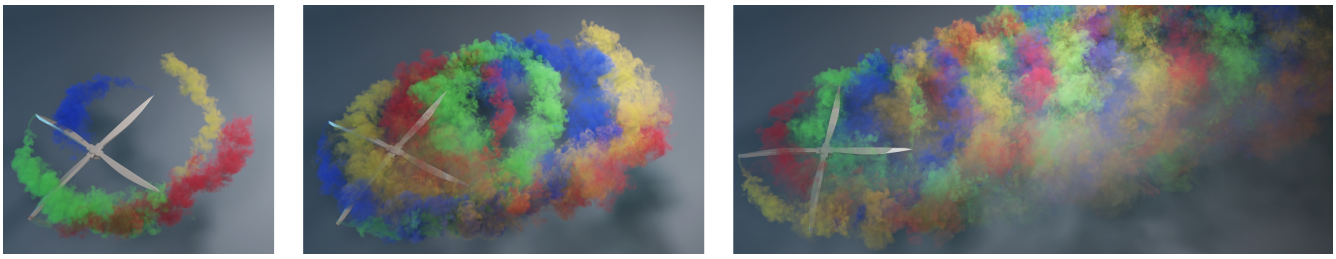
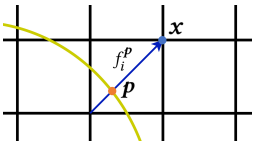


Fig. 6. **Propeller**: A spinning propeller placed in a constant flow field going towards the right creates complex vortical patterns. In this simulation, passive colored particles are emitted at the tips of the four blades purely for visualization purposes.

Streaming. Streaming is performed on all unobstructed links without any change. Now if a link to a grid node $\boldsymbol{x}$ intersects a solid boundary (a case often called a "cut-cell link"), we alter streaming so that an approximated distribution function from the solid boundary is streamed to the grid node. Let $\boldsymbol{p}$ be the point of the intersected solid on the link (see inset). Our approach simply amounts to stream a reconstructed distribution value $f_i^{\boldsymbol{p}}$ to $\boldsymbol{x}$, as we detail next, so that any type of obstacle geometry (even thin shells) can be handled.



Solid distribution. Given our three-velocity-moment setup, we can approximate $f_i^{\boldsymbol{p}}$ easily. First, since we are simulating a homoegenous weakly-compressible fluid, $\rho^{\boldsymbol{p}} = \rho^{\boldsymbol{x}}$. As for the velocity $\boldsymbol{u}^{\boldsymbol{p}}$, we simply pick it based on the motion of the solid boundary at $\boldsymbol{p}$, which is known exactly based on the linear velocity and angular velocity of the solid. We thus just need to form an approximation of the rank-2 tensor $\boldsymbol{S}^{\boldsymbol{p}}$. We know that the equilibrium tensor given the known moments $\rho^{\boldsymbol{p}}$ and $\boldsymbol{u}^{\boldsymbol{p}}$ would be $S^{\boldsymbol{p},\mathrm{eq}}_{\alpha\beta} = u^{\boldsymbol{p}}_\alpha u^{\boldsymbol{p}}_\beta$, so we now need to add an approximation of its off-equilibrium part. We simply use the off-equilibrium moment of node $\boldsymbol{x}$, resulting in the expression:

$$S^{\boldsymbol{p}}_{\alpha\beta} = u^{\boldsymbol{p}}_\alpha u^{\boldsymbol{p}}_\beta + \left(S^{\boldsymbol{x}}_{\alpha\beta} - u^{\boldsymbol{x}}_\alpha u^{\boldsymbol{x}}_\beta\right). \tag{24}$$

This approximation is in fact tantamount to assuming $\nabla \boldsymbol{u}|_{\boldsymbol{x}} = \nabla \boldsymbol{u}|_{\boldsymbol{p}}$ since we know that the off-equilibrium part of $\boldsymbol{S}$ is proportional to the local stress tensor [Malaspinas 2015]. While this may be too coarse of an approximation for very high Reynolds number (generating thin, fast varying boundary layers), we found it to be sufficient for graphics purposes — and better approximations could be derived from neighboring values of nodes on the same side of the obstacle, albeit at a higher computational cost. Now that we have the three velocity moments at $\boldsymbol{p}$, we evaluate the outcoming value $f_i^{\boldsymbol{p}}$ using Eq. (17) and stream it to $\boldsymbol{x}$.

Force on solids. In kinetic theory, momentum exchange is used to calculate the resulting force on a solid when fluid particles hit it, expressed in a Galilean-invariant way [Peng et al. 2016] as:

$$\boldsymbol{F}dt = f_{\mathrm{in}}(\boldsymbol{c}_{\mathrm{in}} - \boldsymbol{u}_{\boldsymbol{p}}) - f_{\mathrm{out}}(\boldsymbol{c}_{\mathrm{out}} - \boldsymbol{u}_{\boldsymbol{p}}), \tag{25}$$

where {in,out} refer to the velocity index of the cut link and its opposite. One can thus evaluate the impact of the fluid motion onto a solid by summing all the contributions for all nodes $\boldsymbol{x}$ whose links ℓ_k intersect the solid to get the force $\boldsymbol{F}_B$ and torque $\boldsymbol{\tau}_B$ through:

$$\boldsymbol{F}_B \equiv \sum_{\boldsymbol{x}} \sum_{\ell_k \in L_{\boldsymbol{x}}} f_{k'}(\boldsymbol{x},t)\,(\boldsymbol{c}_{k'} - \boldsymbol{u}_{\boldsymbol{p}(\boldsymbol{x})}) - f_k(\boldsymbol{x},t+1)\,(\boldsymbol{c}_k - \boldsymbol{u}_{\boldsymbol{p}(\boldsymbol{x})}), \tag{26}$$

$$\boldsymbol{\tau}_B \equiv \sum_{\boldsymbol{x}} (\boldsymbol{x} - \boldsymbol{x}_C) \times \textstyle\sum_{\ell_k \in L_{\boldsymbol{x}}} f_{k'}(\boldsymbol{x},t)\,(\boldsymbol{c}_{k'} - \boldsymbol{u}_{\boldsymbol{p}(\boldsymbol{x})}) - f_k(\boldsymbol{x},t+1)\,(\boldsymbol{c}_k - \boldsymbol{u}_{\boldsymbol{p}(\boldsymbol{x})}), \tag{27}$$

where k' is the opposite direction to k, and we use $\boldsymbol{p}(\boldsymbol{x})$ to denote the solid point $\boldsymbol{p}$ on the link associated to grid node $\boldsymbol{x}$, and $\boldsymbol{x}_C$ to denote the barycenter of the solid. As force and torque expressions are in LBM space, we further map them to physical space before sending them to the rigid-body integrator, as in [Li et al. 2020].

5 RESULTS

We now evaluate of our HOME-LBM approach by discussing a few implementation details first, then presenting a number of tests, simulations, and comparisons to previous work. All results were run on a workstation (AMD Ryzen 9 7900X3D 12-core processor) equipped with an NVIDIA GeForce RTX 4090 with 24GB of GPU memory. We also provide all relevant statistics in Tab. 1, obtained with NVIDIA Nsight to profile kernel time.

Table 1. **Statistics.** All examples timed on a NVIDIA GeForce RTX GPU.

Figure	Resolution	ν	ms/iter	#integration steps	total time (min.)
1	1200×400×400	0.0002	246	11, 500	47.2
2	600×300×300	0.0001	38	8, 280	5.3
3	660×250×330	0.003	37	10, 800	6.6
5	720×360×360	0.0001	39	6, 000	4.0
6	1170×390×624	0.0004	132	13, 500	30.0
13	250×300×250	0.0001	36	4, 800	2.9
15	900×210×480	0.0002	71	6, 000	7.1
16	400×200×400	0.0001	31	41, 600	21.5
17	400×200×400	0.00001	27	21, 200	9.5

5.1 Implementation details

We implemented our approach (see pseudocode in Alg. 1) in C++ and CUDA, using a structure-of-arrays (SOA) data structure to store 20 variables per grid node: 10 per grid node to store the velocity moments, with two copies to facilitate the time update. For our solid boundary treatment, we follow the approach of Li and Desbrun [2023] by first constructing a bounding volume hierarchy tree structure for the 3D mesh model on the GPU; for dynamic objects, we use cut-cell flags and bounding boxes to accelerate link-mesh intersection. Given the low peak memory and efficiency of our HOME-LBM scheme, turbulent flow simulation with static objects is achieved with only one GPU kernel pass which contains streaming, boundary treatment (link-mesh intersection) and collision. Though not limited to such a choice, we use 8×4×4 blocks and assign each node to one CUDA thread. To speed up the computation, we load the node data, save the reconstructed 27 distribution function values into shared memory, and fetch the data when required for the following step within the kernel. Since the streaming step requires accessing neighboring nodes, we also store one halo layer of nodes around the block in our shared memory to further accelerate GPU data access. For simulations involving dynamic objects, we use two kernel passes to improve performance, the additional one focusing

on cut-cell flag update. When realtime performance and memory size are particularly pressing requirements, we also follow [Geier et al. 2017] to improve floating-point precision by replacing f_i with $f_i - w_i$, which slightly changes the computation of ρ (we need to add $\sum_i w_i = 1$back). This trick, allowing for greater off-equilibrium components, lets us use 16-bit floats (instead of 32-bit floats) without stability or visual consequences; our realtime session from Fig. 14 was achieved with this variant, where we gained another factor two in memory and a speedup factor of 1.3 using 8×8×4 blocks this time. Finally, we can advect particles along the macroscopic velocity field with a RK3 method [Ralston 1962] for visualization purposes.

5.2 Accuracy and efficiency evaluations

3D Taylor-Green vortex. We first test our solver's numerical accuracy using the three dimensional Taylor-Green vortex (TGV) case, as in [De Rosis and Luo 2019], for a Mach number $Ma = 0.2c_s$. In a cubic periodic domain of resolution $D \times D \times D$, we initialize the density with $\rho = 1$ and the velocity field $\boldsymbol{u}$ with:

$$\begin{aligned} u_x(\mathbf{x}, 0) &= U_0 \cos(2\pi x/D) \sin(2\pi y/D) \sin(2\pi z/D), \\ u_y(\mathbf{x}, 0) &= -U_0 \sin(2\pi x/D) \cos(2\pi y/D) \sin(2\pi z/D)/2, \\ u_z(\mathbf{x}, 0) &= -U_0 \sin(2\pi x/D) \sin(2\pi y/D) \cos(2\pi z/D)/2. \end{aligned} \tag{28}$$

where $U_0 = 0.2$, $D = 128$, and $L_0 = D/(2\pi)$, and simulate the vortical flow for a duration of $100 L_0/U_0$. By testing different Reynolds numbers Re, we effectively vary the viscosity ν of the flow since $\boldsymbol{\nu} = U_0 L_0/Re$. Fig. 7 shows the normalized kinetic energy for $Re = 2,000$ and $Re = 20,000$ for different solvers. Our HOME-LBM results match the most accurate cumulant-based solver, improving drastically over the original MR-LBM for high Reynolds numbers.

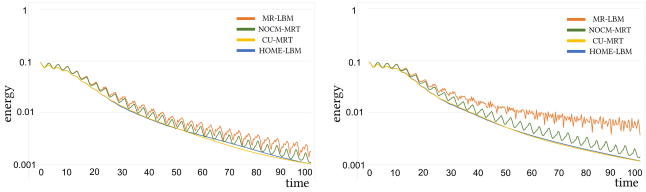


Fig. 7. **Taylor-Green vortex**: For a simulation of the Taylor-Green vortex at two different Reynolds numbers (left: $Re = 2,000$, right: $Re = 20,000$), we measure the (log of the) normalized kinetic energy in time for four different collision models: the moment representation method (MR-LBM [Ferrari et al. 2023]), the central-moment model (NOCM [Li et al. 2020]), the cumulant-based model (CU-LBM [Geier et al. 2017]), and ours (using the central-moment based model). HOME-LBM matches the state-of-the-art cumulant model, and drastically improves upon MR-LBM.

Flow past a sphere. We also evaluated our solver in Fig. 8 with different LBM-based solvers for the simulation of a flow past a spherical obstacle at a very high Reynolds number $Re = 800,000$ using a computational grid of 200×400×200. While the original MR-LBM [Ferrari et al. 2023] (a) crashes early, we see that the approach of [Li et al. 2020] which optimizes the high-order parameters of the central-moment based collision model (c) outperforms the collision model where all non-physical high-order parameters are set to 1 (generating overly smooth flow, (b)) — but ringing artifacts start creeping in at such a large Re. The state-of-the-art cumulant collision model [Geier et al. 2017], instead, is free of ringing artifacts (d). HOME-LBM, shown in (e) using the cumulant-based model and in (f) using the central-moment-based collision, approximates the

ALGORITHM 1: Pseudocode of our kinetic single-phase fluid solver.

```
t ← 0;
Initialize ρ, u and S;
while t < T do
    For all fluid nodes in GPU kernels, reconstruct distribution
      function f by Eq. (17) into shared Memory ;      // Sec. 4.2
    for each node x do
        for each direction i do
            if no intersection between node x and node x − c_i then
                Stream f_i to neighboring node x − c_i;
            else
                Apply solid boundary condition ;      // Sec. 4.4
            end
        end
    end
    Evaluate new ρ, u and S with Eq. (7) ;            // Sec. 3
    Perform moment-based collision ;                  // Sec. 4.3
    t ← t + 1
end
```

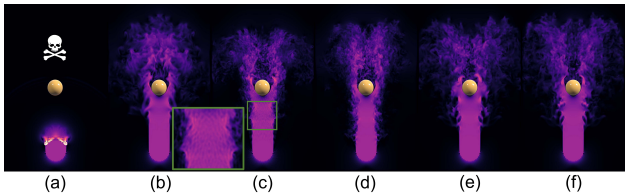


Fig. 8. **Comparisons.** For a strong flow past a sphere, we run various LBM solvers based on different collision models: (a) MR-LBM [Ferrari et al. 2023] which blows up at an early frame, (b) original NOCM-MRT [De Rosis 2017], (c) optimized NOCM-MRT [Li et al. 2020], (d) CU-MRT [iRMB 2023], followed by HOME-LBM with cumulant-based (e) and central-moment-based (f) collisions. While our approach does not capture exactly the high-order cumulant collision model (e), both our options behave similarly, without the ringing artifacts of (c) or the clear oversmoothing of (b) as exhibited by a cross-section of flow field indicating the velocity magnitude.

results of the full-LBM cumulant-based model well despite its only third-order nature, and does not exhibit ringing artifacts either.

Accuracy and robustness analysis. We also performed experiments to evaluate our solver's accuracy. We first test the accuracy of HOME-LBM using the 2D Taylor-Green vortex example (for which a closed-form solution is known) proposed in [Zehnder et al. 2018] and used in the NOCM-MRT work of [Li et al. 2020]. Compared to these two state-of-the-art Navier-Stokes vs. LBM solvers for different time steps and grid sizes, Fig. 9 demonstrates that our compression of the distribution to its first three velocity moments does not affect

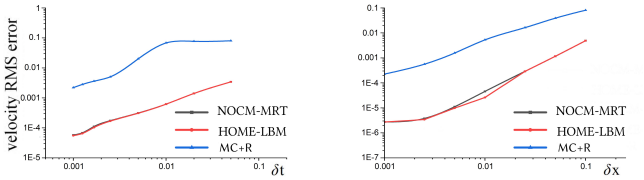


Fig. 9. **2D Taylor-Green vortex test.** We compare the reflection-advection (MC+R) [Zehnder et al. 2018], NOCM-MRT LBM [Li et al. 2020] and HOME-LBM solvers for (a) different time step sizes and (b) spatial grid sizes on the 2D Taylor-Green vortex simulation. The velocity root-mean-square error is computed based on the known analytical solution.

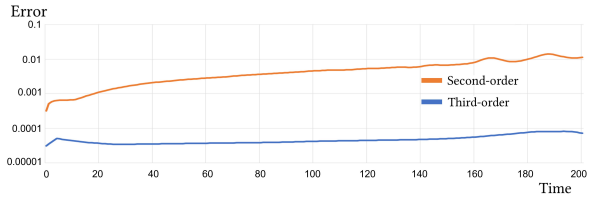
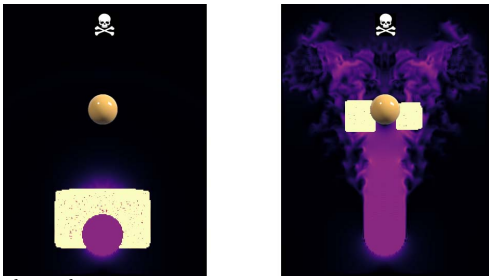


Fig. 10. **Reconstruction error.** To test second- vs. third-order Hermite reconstructions from $\{\mathbf{a}[0], \mathbf{a}[1], \mathbf{a}[2]\}$ (Eq. (8) vs. (17)), we measure the L_2 error in the reconstructed third-order moments $\mathbf{a}[3]$ w.r.t. the original ones throughout the simulation from Fig. 8 [Li et al. 2020] .

the resulting accuracy of the solver as we almost systematically match the results of [Li et al. 2020], with roughly two orders of magnitude smaller velocity root-mean-square error than [Zehnder et al. 2018] — exemplifying the current superiority of LBM solvers. Note that this result using a D2Q9 discretization is likely to be better than the actual reconstruction error generated by HOME-LBM for a D3Q27 discretization, but it provides clear evidence of the advantage that our moment-encoded approach has over common Navier-Stokes solvers. We then use a neutral LBM simulation (we picked the NOCM-MRT simulation [Li et al. 2020] of the jet flow past a sphere from Fig. 8) to test the difference between a second-order vs. a third-order Hermite reconstruction method (i.e., Eq. (8) vs. Eq. (17)): for each frame of the animation, we pick all the distribution functions, compute their first three velocity-moments, then use either Eq. (8) *or* Eq. (17) to reconstruct them. When we measure the L_2 error of the third-order moment from the reconstructions compared to the third-order moment of the original distributions, we see around one to two orders of improvement in the third-order Hermite reconstruction (Fig. 10). Finally, we prove in Fig. 11 that both our high-order distribution reconstruction *and* moment-based collision model are important: substituting one of these contributions with a lower-order approximation leads to blowups.

Performance comparison. To show our solver's efficiency, we compare the timing cost per iteration for an identical simulation, chosen to be the one from Fig. 8. We use the optimized version of NOCM-MRT from [Lyu et al. 2021] which employs LU decomposition and shared memory to accelerate the approach of [Li et al. 2020], and the GPU-based cumulant collision model from [iRMB 2023]. We run the



low-order reconstruction, high-order collision

high-order reconstruction, low-order collision

Fig. 11. **Ablation study.** To test the individual effects of the high-order distribution reconstruction and high-order collision model, we run the simulation scenario from Fig. 8 with (left) low-order reconstruction and high-order collision, vs. (right) high-order reconstruction and low-order collision. Both cases blow up (yellow regions indicate *NaN* at time of blow-up), with low-order distribution reconstruction stopping very early on.

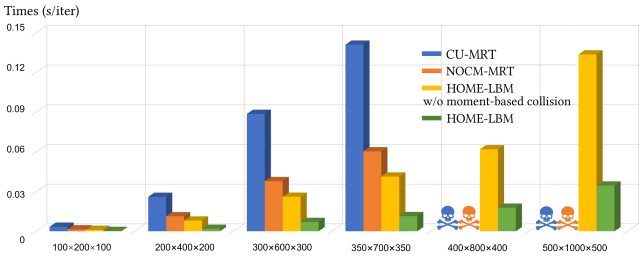


Fig. 12. **Efficiency of LBM methods**: By running the same "flow past sphere" example with three implementations (CU-LBM [iRMB 2023], NOCM-MRT [Lyu et al. 2021], and our HOME-LBM), we compare the different average times per iteration of these solvers. HOME-LBM is on average 5 times more efficient than NOCM-MRT and 10 times more efficient than CU-LBM. Moreover, our smaller memory footprint allows us to reach fine grid sizes for which the two other methods run out of memory. In particular, our moment-based collision is a key factor in our timing gains.

same simulation for 2000 iterations, count the computational time using the `clock()` function, and deduce the mean time per iteration for each method. Results are shown in Fig. 12: HOME-LBM ends up about five times faster than NOCM-MRT (in particular because ours only needs one kernel pass and significantly less memory, which significantly reduces data read/write and boosts performance) and over an order of magnitude faster than cumulant collision model; moreover, on a GPU with 24GB memory, HOME-LBM can support resolutions of up to 500×1000×500, while the two others only go up to 350×700×350 before running out of memory. Note that in particular, our novel moment-based collision is an important factor in our timing gains since using the NOCM-MRT model from [Lyu et al. 2021] on the distribution function right after streaming instead ends up being four times slower.

5.3 Offline simulations

Now, we go over different test results that we ran to illustrate our solver ability. Note that we employed Bullet [Coumans and Bai 2021] as our rigid-body solver, and final results were rendered using the GPU-accelerated 3D renderer Redshift [Maxon 2023]

3D coupling with static objects. In Fig. 1, we demonstrate a flow past a detailed mesh of a tractor. Five different colored smoke injections are used to visualize the details of the flow and wake. In Fig. 2, an octopus mesh with intricate topology (as it contains several serpentine grooves) is placed in a turbulent wind field. Smoke flowing through the complex obstacle exhibits a variety of vortex details. Fig. 5 shows a complex wake created by a fern in the wind.

3D coupling with dynamic objects. Fig. 6 shows a rotating aircraft propeller in a wind field. In this one-way coupling example, the

Fig. 13. **Leaves blown by a jet**: Several leaves of different weights are dropped on a jet flow, demonstrating two-way coupling effects.

wind entrains the vortices generated by the propeller's dynamics to the right of the frame. In order to visualize the flow details, we insert four passive smoke injectors at the tips of the four blades. Fig. 15 shows the motion of a hair comb creating fine vortices. We also demonstrate two-way coupling examples. Fig. 13 shows leaves of different weights being dropped on top of a jet flow and falling under the action of gravity. Fig. 16 shows a rotor of 2kg and 3.75m of diameter, where a torque is applied to make it rotate at 19 rad/s. This makes it lift off, but as soon as the torque stops, it falls back onto the floor. Finally, in Fig. 17, two car models of widely different weights are dropped to the floor. Due to air resistance, the cars end up with different terminal speeds and hit the ground at different times, resulting in different effects on the surrounding smoke.

5.4 Real-time simulation

We also present a realtime session entirely run on an NVIDIA GeForce RTX 4090 GPU card in Fig. 14. With this single GPU, we were able to manipulate interactively the displacement of a rigid body (a bowl in this case) and run our HOME-LBM simulator with a 196×196×196 grid, as well as an in-house volume renderer. Note that graphics fluid solvers are also able to achieve realtime through the use of large time steps and/or multigrid pressure solves, but as argued in [Li et al. 2020], the visual quality and accuracy of LBM simulation (and in particular, its ability to handle nearly inviscid fluids) are significantly higher, thus resulting in improved realism.

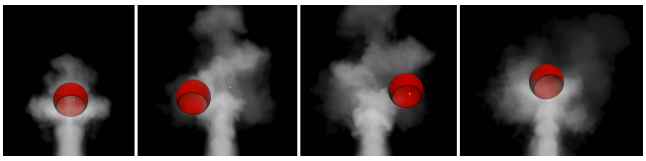


Fig. 14. **Real time demo**: a real-time session where a user moves around an object (here, a bowl) which interacts with a jet flow is entirely done on a single GPU card, volumetric rendering included.

5.5 Matching real experiments

Finally, we simulated the airflow over a delta wing with a 75°-sweep angle for an angle of attack of 20°. As Fig. 3 demonstrates, our simulation reproduces the stable spiral vortices above the wing which are responsible for aerodynamic lift, matching the experimental visualization from [Délery 2001].

6 CONCLUSION

In this paper we introduced HOME-LBM, a lightweight, yet light-speed simulator of fluid turbulent flows and rigid body coupling based on high-order moment-encoded LBM solver. It offers a fresh perspective on the LBM method: while all previous graphics approaches using kinetic solvers had distribution functions as their main variables, we show that keeping only the first three velocity moments leads to a smaller memory footprint and a far improved computational efficiency. Compared to other computational fluid dynamics approaches using the same moments to encode the kinetic variables, we introduce a higher-order reconstruction of the distribution function for the streaming step, and show that the collision step can be done with a simple closed-form update rule of the moments, improving the efficiency of the solver while offering an accuracy close to the full-blown distribution-based solvers.

This new type of solver opens up a number of possible research directions. First, our third-order Hermite-based collision model leads to relatively simple update rules for the central moment version, but the cumulant-based version (which we provide in App. D) is slightly more involved; it would be useful to see if simpler rules could be found, still ensuring at least a third-order approximation. Note that if more moments are stored, one could also go up in order — but we believe that third-order offers a good compromise for graphics as it suffices for most typical scenarios of visual fluid simulation and offers particularly simple computations for the case of single-phase fluid simulation. Moreover, the efficiency and memory footprint that we now reach gives hope that one could treat compressible and thermal flows with a kinetic solver with significantly less memory than the current $D3Q103$ lattice structures used in compressible LBM works in CFD [Coreixas et al. 2017]. It could require the addition of other macroscopic variables such as temperature terms, and maybe higher-order reconstruction and collision. Furthermore, extending our moment-based approach to LBM multiphase solvers would be an equally promising direction for future research.

ACKNOWLEDGMENTS

The tractor model in Fig. 1 is from grabcad, the octopus in Fig. 2 is from Thingi10k, the fern in Fig. 5 is from cgtrader, the leaf in Fig. 13 is from archive3d, the comb in Fig. 15 is from cgtrader, while the propeller in Figs. 6 and 16 is from free3d. Mathieu Desbrun acknowledges the generous support of Ansys and Adobe Research, as well as a Choose France Inria chair.

REFERENCES

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph.* 31, 4 (2012), 62.

Ryoichi Ando, Nils Thürey, and Chris Wojtan. 2013. Highly Adaptive Liquid Simulations on Tetrahedral Meshes. *ACM Trans. Graph.* 32, 4, Article 103 (2013).

Vinicius C Azevedo, Christopher Batty, and Manuel M Oliveira. 2016. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Trans. Graph.* 35, 4 (2016), 97.

Stefan Band, Christoph Gissler, Markus Ihmsen, Jens Cornelis, Andreas Peer, and Matthias Teschner. 2018. Pressure boundaries for implicit incompressible SPH. *ACM Trans. Graph.* 37, 2 (2018), 14.

Stefan Band, Christoph Gissler, and Matthias Teschner. 2017. Moving Least Squares Boundaries for SPH Fluids. In *VRPHYS*. 21–28.

Christopher Batty, Florence Bertails, and Robert Bridson. 2007. A Fast Variational Framework for Accurate Solid-Fluid Coupling. In *ACM SIGGRAPH*. 100–es.

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

Markus Becker, Hendrik Tessendorf, and Matthias Teschner. 2009. Direct forcing for Lagrangian rigid-fluid coupling. *IEEE Trans. Vis. Comp. Graph.* 15, 3 (2009), 493–503.

Jan Bender and Dan Koschier. 2016. Divergence-free SPH for incompressible and viscous fluids. *IEEE Trans. Vis. Comp. Graph.* 23, 3 (2016), 1193–1206.

Tyson Brochu, Todd Keeler, and Robert Bridson. 2012. Linear-Time Smoke Animation with Vortex Sheet Meshes. In *Symposium on Computer Animation*. 87–95.

Mark Carlson, Peter J. Mucha, and Greg Turk. 2004. Rigid Fluid: Animating the Interplay between Rigid Bodies and Fluid. In *ACM SIGGRAPH*. 377–384.

Shiyi Chen and Gary D Doolen. 1998. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.* 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.

Andrea Colagrossi and Maurizio Landrini. 2003. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *J. Comput. Phys.* 191, 2 (2003), 448–475.

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.

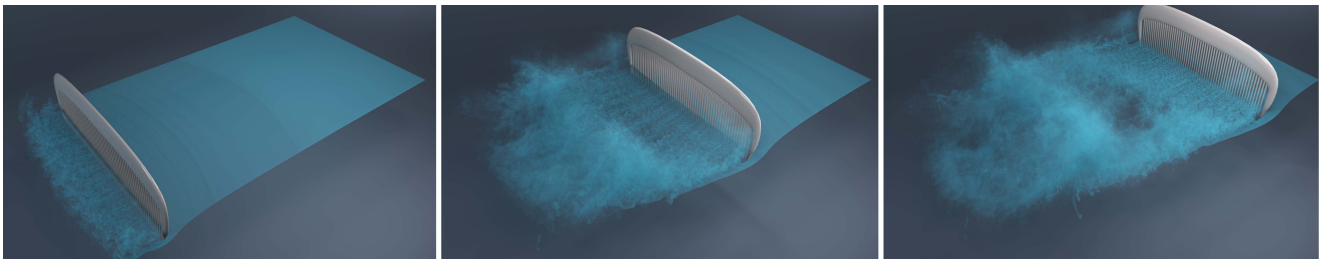


Fig. 15. **Air comb**: Inspired by the hair brush example of [Lyu et al. 2021, Fig. 4], we simulate a hair comb containing sixty teeth moving around and creating fine vortices in its wake, properly capturing the complex fluid-solid interaction engendered by this finely-detailed geometry.

Erwin Coumans and Yunfei Bai. 2016–2021. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org.

Fernando de Goes, Corentin Wallez, Jin Huang, Dmitry Pavlov, and Mathieu Desbrun. 2015. Power Particles: An Incompressible Fluid Solver Based on Power Diagrams. *ACM Trans. Graph.* 34, 4, Article 50 (2015).

Alessandro De Rosis. 2017. Nonorthogonal central-moments-based lattice Boltzmann scheme in three dimensions. *Phys. Rev. E* 95, 1 (2017), 013310.

Alessandro De Rosis and Kai H Luo. 2019. Role of higher-order Hermite polynomials in the central-moments-based lattice Boltzmann framework. *Phys. Rev. E* 99, 1 (2019), 013301.

Jean M Délery. 2001. Robert Legendre and Henri Werlé: Toward the Elucidation of Three-Dimensional Separation. *Annu. Rev. Fluid Mech.* 33, 1 (2001), 129–154.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed Particles: A New Paradigm for Animating Highly Deformable Bodies. In *Eurographics Workshop on Computer Animation and Simulation*. 61–76.

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

Yu Fang, Ziyin Qu, Minchen Li, Xinxin Zhang, Yixin Zhu, Mridul Aanjaneya, and Chenfanfu Jiang. 2020. IQ-MPM: An Interface Quadrature Material Point Method for Non-Sticky Strongly Two-Way Coupled Nonlinear Solids and Fluids. *ACM Trans. Graph.* 39, 4, Article 51 (2020).

Yun 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 (2021), 1–16.

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

Marco A. Ferrari, Waine B. de Oliveira Jr., Alan Lugarini, Admilson T. Franco, and Luiz A. Hegele Jr. 2023. A graphic processing unit implementation for the moment representation of the lattice Boltzmann method. *International Journal for Numerical Methods in Fluids* 95, 7 (2023), 1076–1089.

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

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

Martin Geier and Martin Schönherr. 2017. Esoteric Twist: An Efficient In-Place Streaming Algorithms 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. *Comput. Math. Appl.* 70, 4 (2015), 507–547.

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

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

Abhinav Golas, Rahul Narain, Jason Sewall, Pavel Krajcevski, Pradeep Dubey, and Ming Lin. 2012. Large-Scale Fluid Simulation Using Velocity-Vorticity Domain Decomposition. *ACM Trans. Graph.* 31, 6, Article 148 (2012).

John Gounley, Madhurima Vardhan, Erik W. Draeger, Pedro Valero-Lara, Shirley V. Moore, and Amanda Randles. 2022. Propagation Pattern for Moment Representation of the Lattice Boltzmann Method. *IEEE Trans. Parallel Distrib. Syst.* 33, 3 (2022), 642–653.

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

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

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

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

iRMB at Technische Universität Braunschweig. 2023. VirtualFluids. Git repository. https://git.rz.tu-bs.de/irmb/virtualfluids

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).

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

Dan Koschier and Jan Bender. 2017. Density Maps for Improved SPH Boundary Handling. In *Symposium on Computer Animation*. Article 1.

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

Jonas Latt. 2007. *How to implement your DdQq dynamics with only q variables per node (instead of 2q)*. Technical Report. Tufts University.

Jonas Latt and Bastien Chopard. 2006. Lattice Boltzmann method with regularized pre-collision distribution functions. *Math. Comput. Simul.* 72, 2 (2006), 165–168.

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), 92.

Michael Lentine, Wen Zheng, and Ronald Fedkiw. 2010. A Novel Algorithm for Incompressible Flow Using Only a Coarse Grid Projection. In *ACM SIGGRAPH*. Article 114.

Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling. *ACM Trans. Graph.* 39, 4, Article 47 (2020).

Wei Li and Mathieu Desbrun. 2023. Fluid-Solid Coupling in Kinetic Two-Phase Flow Simulation. *ACM Trans. Graph.* 42, 4, Article 123 (jul 2023).

Wei Li, Daoming Liu, Mathieu Desbrun, Jin Huang, and Xiaopei Liu. 2021. Kinetic-Based Multiphase Flow Simulation. *IEEE Trans. Vis. Comp. Graph.* 27, 7 (2021), 3318–3334.

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).

Wei Li, Xiaoming Wei, and Arie Kaufman. 2003. Implementing lattice Boltzmann computation on graphics hardware. *The Visual Computer* 19 (2003), 444–456.

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

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 (2023).

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 (dec 2021).

Orestis Malaspinas. 2015. Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization. arXiv:1505.06900 [physics.flu-dyn]

Matlab. 2023. *R2022a*. MathWorks Inc.

Maxon. 2023. The world's first fully GPU-accelerated, biased renderer. https://www.maxon.net/en/redshift.

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. 2009. Energy-Preserving Integrators for Fluid Animation. In *ACM SIGGRAPH*. Article 38.

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

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

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

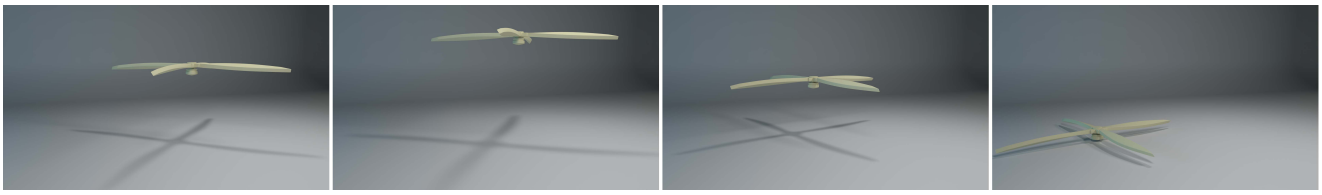


Fig. 16. **Rotor lift**: Spinning a rotor (2 kilograms, span of 3.75 meters) at 19 rad/s makes it lift off; once the applied torque stops, it falls back onto the floor.

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

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

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

Anthony Ralston. 1962. Runge-Kutta methods with minimum error bounds. *Mathematics of computation* 16, 80 (1962), 431–437.

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

Avi Robinson-Mosher, Tamar Shinar, Jon Gretarsson, Jonathan Su, and Ronald Fedkiw. 2008. Two-Way Coupling of Fluids to Rigid and Deformable Solids and Shells. In *ACM SIGGRAPH*. Article 46.

Doug Roble, Nafees bin Zafar, and Henrik Falt. 2005. Cartesian Grid Fluid Simulation with Irregular Boundary Voxels. In *ACM SIGGRAPH Sketches*. 138–es.

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

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

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

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

B. Solenthaler and R. Pajarola. 2008. Density Contrast SPH Interfaces. In *Symposium on Computer Animation*. 211–218.

B. Solenthaler and R. Pajarola. 2009. Predictive-Corrective Incompressible SPH. *ACM Trans. Graph.* 28, 3 (jul 2009).

Jos Stam. 1999. Stable Fluids. In *Annual Conference on Computer Graphics and Interactive Techniques*. 121–128.

Tsunemi Takahashi, Heihachi Ueki, Atsushi Kunimatsu, and Hiroko Fujii. 2002. The Simulation of Fluid-Rigid Body Interaction. In *ACM SIGGRAPH Sketches*. 266.

Michael Tao, Christopher Batty, Mirela Ben-Chen, Eugene Fiume, and David IW Levin. 2022. VEMPIC: particle-in-polyhedron fluid simulation for intricate solid boundaries. *ACM Trans. Graph.* 41, 4 (2022), 1–22.

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

Niels Thürey and Ulrich Rüde. 2009. Stable free surface flows with the lattice Boltzmann method on adaptively coarsened grids. *Comput. Visual Sci.* 12 (2009), 247–263.

Madhurima Vardhan, John Gounley, Luiz Hegele, Erik W. Draeger, and Amanda Randles. 2019. Moment Representation in the Lattice Boltzmann Method on Massively Parallel Hardware. In *Supercomputing*. Article 34.

Daniel Weber, Johannes Mueller-Roemer, André Stork, and Dieter Fellner. 2015. A Cut-Cell Geometric Multigrid Poisson Solver for Fluid Simulation. *Comput. Graph. Forum* 34, 2 (2015), 481–491.

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

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

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

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

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

Yongning Zhu and Robert Bridson. 2005. Animating Sand as a Fluid. In *ACM SIGGRAPH*. 965–972.

A HERMITE POLYNOMIALS

The explicit second-order Hermite tensor values we need are:

$$H^{[2]}_{\alpha,\beta}(\boldsymbol{c}_i) = c_{i,\alpha}c_{i,\beta} - \tfrac{1}{3}\delta_{\alpha\beta},$$

while the third-order Hermite tensor values are:

$$H^{[3]}_{\alpha,\beta,\gamma}(\boldsymbol{c}_i) = c_{i,\alpha}c_{i,\beta}c_{i,\gamma} - \tfrac{1}{3}\left(c_{i,\alpha}\delta_{\beta\gamma} + c_{i,\beta}\delta_{\alpha\gamma} + c_{i,\gamma}\delta_{\alpha\beta}\right).$$

B 2D MOMENT-BASED RECONSTRUCTION

For the reader's convenience, we also provide the 2D reconstruction of a distribution from its three velocity moments ρ, $\rho\boldsymbol{u}$, and $\rho\boldsymbol{S}$, in order to complement the 3D expression from Eq. (17):

$$\begin{aligned} f_i =& \rho w_i \Bigg[1 + \frac{\boldsymbol{c}_i \cdot \mathbf{u}}{c_s^2} + \frac{\boldsymbol{H}^{[2]}(\boldsymbol{c}_i) : \mathbf{S}}{2c_s^4} \\ &+ \frac{1}{2c_s^6}\big(H^{[3]}_{xxy}(\boldsymbol{c}_i)(S_{xx}u_y + 2S_{xy}u_x - 2u_xu_xu_y) \\ &\quad + H^{[3]}_{xyy}(\boldsymbol{c}_i)(S_{yy}u_x + 2S_{xy}u_y - 2u_xu_yu_y)\big)\Bigg]. \end{aligned} \tag{29}$$

C 2D CENTRAL-MOMENT-BASED COLLISION

To complement the 3D expressions from Sec. 4.3, we also provide the 2D expressions of our central-moment-based collision model. The projections into the Hermite-based form of Eq. (11) of $\boldsymbol{m}^{\text{eq}}$ and the force-related term lead to mostly zero terms, except for:

$$m_0^{\text{eq}} = \rho, \quad m_3^{\text{eq}} = 2\rho/3, \quad m_8^{\text{eq}} = \rho/9$$

and

$$K_1 = F_x, \quad K_2 = F_y, \quad K_6 = \tfrac{1}{3}F_y, \quad K_7 = \tfrac{1}{3}F_x;$$

The updates then become:

$$\begin{aligned} \rho(\boldsymbol{x}, t+1) &= \rho^*; \\ u_\alpha(\boldsymbol{x}, t+1) &= u^*_\alpha + \tfrac{1}{2\rho^*}F_\alpha; \\ S_{xy}(\boldsymbol{x}, t+1) &= (1-\tfrac{1}{\tau})S^*_{xy} + \tfrac{1}{\tau}u^*_xu^*_y + \tfrac{2\tau-1}{2\tau\rho^*}(F_xu^*_y + F_yu^*_x); \\ S_{xx}(\boldsymbol{x}, t+1) &= \tfrac{\tau-1}{2\tau}(S^*_{xx} - S^*_{yy}) + \tfrac{\tau+1}{2\tau}{u^*_x}^2 \\ &\quad + \tfrac{\tau-1}{2\tau}{u^*_y}^2 + \tfrac{1}{\rho^*}F_xu^*_x + \tfrac{\tau-1}{2\tau\rho^*}(F_xu^*_x - F_yu^*_y); \\ S_{yy}(\boldsymbol{x}, t+1) &= \tfrac{\tau-1}{2\tau}(S^*_{yy} - S^*_{xx}) + \tfrac{\tau+1}{2\tau}{u^*_y}^2 \\ &\quad + \tfrac{\tau-1}{2\tau}{u^*_x}^2 + \tfrac{1}{\rho^*}F_yu^t_y + \tfrac{\tau-1}{2\tau\rho^*}(F_yu^*_y - F_xu^*_x). \end{aligned}$$

D 3D CUMULANT-MOMENT-BASED COLLISION

Finally, we also provide the closed-form expressions one gets when using the cumulant-based approach from [Geier et al. 2017] in our moment-encoded context:

$$\begin{aligned} \rho(\boldsymbol{x}, t+1) &= \rho^*; \\ u_\alpha(\boldsymbol{x}, t+1) &= u^*_\alpha + \tfrac{1}{2\rho^*}F_\alpha; \\ S_{xy}(\boldsymbol{x}, t+1) &= (1-\tfrac{1}{\tau})S^*_{xy} + \tfrac{1}{\tau}u^*_xu^*_y + \tfrac{2\tau-1}{2\tau\rho^*}(F_xu^*_y + F_yu^*_x); \\ S_{xz}(\boldsymbol{x}, t+1) &= (1-\tfrac{1}{\tau})S^*_{xz} + \tfrac{1}{\tau}u^*_xu^*_z + \tfrac{2\tau-1}{2\tau\rho^*}(F_xu^*_z + F_zu^*_x); \\ S_{yz}(\boldsymbol{x}, t+1) &= (1-\tfrac{1}{\tau})S^*_{yz} + \tfrac{1}{\tau}u^*_yu^*_z + \tfrac{2\tau-1}{2\tau\rho^*}(F_zu^*_y + F_yu^*_z); \end{aligned}$$

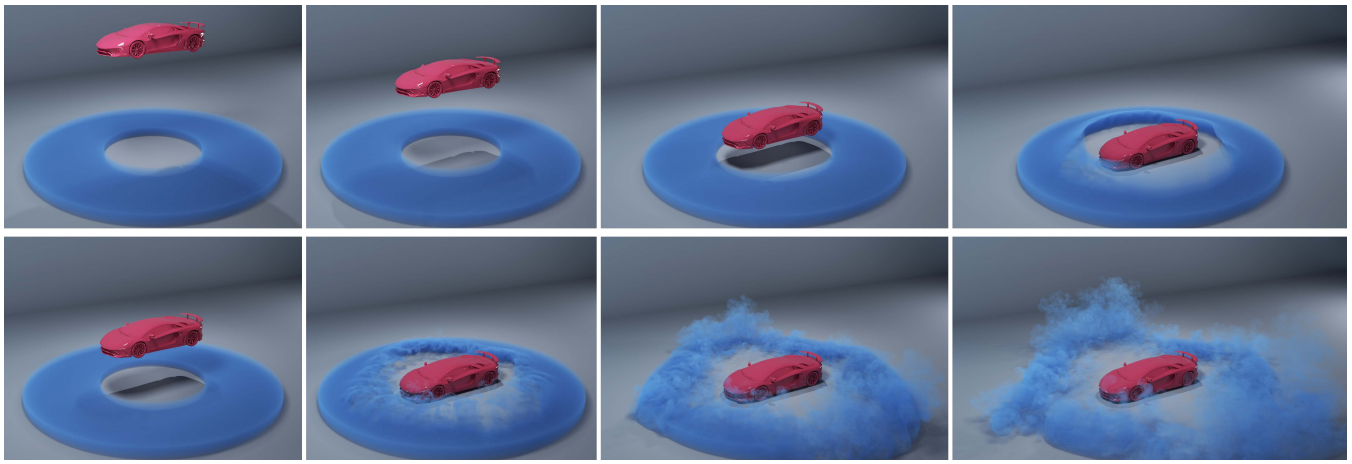


Fig. 17. **Falling cars**. In this two-way coupling example of a very light car (top) and of a heavier car (bottom) dropped on the floor from a height, the car's terminal speed ends up being very different for each case, and its wake disturbs a static smoke ring.

$$
\begin{aligned}
S_{xx}(\boldsymbol{x},t+1) = {} & \tfrac{1}{\tau^2}\big(u_x^{*4} - \tfrac{1}{2}u_y^{*4} - \tfrac{1}{2}u_z^{*4} - \tfrac{1}{4}u_x^{*2}u_y^{*2} - \tfrac{1}{4}u_x^{*2}u_z^{*2} + \tfrac{1}{2}u_y^{*2}u_z^{*2} - \tfrac{1}{2}u_x^{*2}(2S_{xx}^* - S_{yy}^* - S_{zz}^*) + \tfrac{1}{4}u_y^{*2}(2S_{yy}^* - S_{xx}^* - S_{zz}^*) \\
& + \tfrac{1}{4}u_z^{*2}(2S_{zz}^* - S_{xx}^* - S_{yy}^*) + \tfrac{1}{\rho^*}F_x(-u_x^{*3} - \tfrac{1}{4}u_x^*u_y^{*2} - \tfrac{1}{4}u_x^*u_z^{*2}) + \tfrac{1}{\rho^*}F_y(\tfrac{1}{2}u_y^{*3} + \tfrac{1}{2}u_x^{*2}u_y^* - \tfrac{1}{4}u_y^*u_z^{*2}) + \tfrac{1}{\rho^*}F_z(\tfrac{1}{2}u_z^{*3} + \tfrac{1}{2}u_x^{*2}u_z^* - \tfrac{1}{4}u_z^*u_y^{*2})\big) \\
& + \tfrac{1}{\tau}\big(-2u_x^{*4} + \tfrac{1}{4}u_y^{*4} + \tfrac{1}{4}u_z^{*4} + \tfrac{5}{4}u_x^{*2}u_y^{*2} + \tfrac{5}{4}u_x^{*2}u_z^{*2} - u_y^{*2}u_z^{*2} + \tfrac{2}{3}u_x^{*2} - \tfrac{1}{3}u_y^{*2} - \tfrac{1}{3}u_z^{*2} + \tfrac{1}{4}u_x^{*2}(8S_{xx}^* - 7S_{yy}^* - 7S_{zz}^*) \\
& + \tfrac{1}{4}u_y^{*2}(2S_{xx}^* - S_{yy}^* + 2S_{zz}^*) + \tfrac{1}{4}u_z^{*2}(2S_{xx}^* + 2S_{yy}^* - S_{zz}^*) - \tfrac{1}{3}(2S_{xx}^* - S_{yy}^* - S_{zz}^*) + \tfrac{1}{\rho^*}F_x(2u_x^{*3} + \tfrac{1}{2}u_x^*u_y^{*2} + \tfrac{1}{2}u_x^*u_z^{*2} - \tfrac{2}{3}u_x^*) \\
& + \tfrac{1}{\rho^*}F_y(-\tfrac{3}{4}u_y^{*3} - \tfrac{7}{4}u_x^{*2}u_y^* + \tfrac{1}{2}u_y^*u_z^{*2} + \tfrac{1}{3}u_y^*) + \tfrac{1}{\rho^*}F_z(-\tfrac{3}{4}u_z^{*3} - \tfrac{7}{4}u_x^{*2}u_z^* + \tfrac{1}{2}u_z^*u_y^{*2} + \tfrac{1}{3}u_z^*)\big) \\
& - \tfrac{5}{4}u_x^{*4} + \tfrac{1}{4}u_y^{*4} + \tfrac{1}{4}u_z^{*4} - u_x^{*2}u_y^{*2} - u_x^{*2}u_z^{*2} + \tfrac{1}{2}u_y^{*2}u_z^{*2} + \tfrac{1}{3}u_x^{*2} + \tfrac{1}{3}u_y^{*2} + \tfrac{1}{3}u_z^{*2} + (\tfrac{5}{4}u_x^{*2} - \tfrac{1}{4}u_y^{*2} - \tfrac{1}{4}u_z^{*2})(S_{xx}^* + S_{yy}^* + S_{zz}^*) \\
& + \tfrac{1}{3}(2S_{xx}^* - S_{yy}^* - S_{zz}^*) + \tfrac{1}{\rho^*}F_xu_x^* + \tfrac{1}{\rho^*}F_x(\tfrac{5}{4}u_x^{*3} - \tfrac{1}{4}u_x^*u_y^{*2} - \tfrac{1}{4}u_x^*u_z^{*2} - \tfrac{1}{3}u_x^*) + \tfrac{1}{\rho^*}F_y(-\tfrac{1}{4}u_y^{*3} + \tfrac{5}{4}u_x^{*2}u_y^* - \tfrac{1}{4}u_y^*u_z^{*2} - \tfrac{1}{3}u_y^*) \\
& + \tfrac{1}{\rho^*}F_z(-\tfrac{1}{4}u_z^{*3} + \tfrac{5}{4}u_x^{*2}u_z^* - \tfrac{1}{4}u_z^*u_y^{*2} - \tfrac{1}{3}u_z^*)\,; \\
S_{yy}(\boldsymbol{x},t+1) = {} & \tfrac{1}{\tau^2}\big(u_y^{*4} - \tfrac{1}{2}u_x^{*4} - \tfrac{1}{2}u_z^{*4} - \tfrac{1}{4}u_y^{*2}u_x^{*2} - \tfrac{1}{4}u_y^{*2}u_z^{*2} + \tfrac{1}{2}u_x^{*2}u_z^{*2} - \tfrac{1}{2}u_y^{*2}(2S_{yy}^* - S_{xx}^* - S_{zz}^*) + \tfrac{1}{4}u_x^{*2}(2S_{xx}^* - S_{yy}^* - S_{zz}^*) \\
& + \tfrac{1}{4}u_z^{*2}(2S_{zz}^* - S_{yy}^* - S_{xx}^*) + \tfrac{1}{\rho^*}F_y(-u_y^{*3} - \tfrac{1}{4}u_y^*u_x^{*2} - \tfrac{1}{4}u_y^*u_z^{*2}) + \tfrac{1}{\rho^*}F_x(\tfrac{1}{2}u_x^{*3} + \tfrac{1}{2}u_y^{*2}u_x^* - \tfrac{1}{4}u_x^*u_z^{*2}) + \tfrac{1}{\rho^*}F_z(\tfrac{1}{2}u_z^{*3} + \tfrac{1}{2}u_y^{*2}u_z^* - \tfrac{1}{4}u_z^*u_x^{*2})\big) \\
& + \tfrac{1}{\tau}\big(-2u_y^{*4} + \tfrac{1}{4}u_x^{*4} + \tfrac{1}{4}u_z^{*4} + \tfrac{5}{4}u_y^{*2}u_x^{*2} + \tfrac{5}{4}u_y^{*2}u_z^{*2} - u_x^{*2}u_z^{*2} + \tfrac{2}{3}u_y^{*2} - \tfrac{1}{3}u_x^{*2} - \tfrac{1}{3}u_z^{*2} + \tfrac{1}{4}u_y^{*2}(8S_{yy}^* - 7S_{xx}^* - 7S_{zz}^*) \\
& + \tfrac{1}{4}u_x^{*2}(2S_{yy}^* - S_{xx}^* + 2S_{zz}^*) + \tfrac{1}{4}u_z^{*2}(2S_{yy}^* + 2S_{xx}^* - S_{zz}^*) - \tfrac{1}{3}(2S_{yy}^* - S_{xx}^* - S_{zz}^*) + \tfrac{1}{\rho^*}F_y(2u_y^{*3} + \tfrac{1}{2}u_y^*u_x^{*2} + \tfrac{1}{2}u_y^*u_z^{*2} - \tfrac{2}{3}u_y^*) \\
& + \tfrac{1}{\rho^*}F_x(-\tfrac{3}{4}u_x^{*3} - \tfrac{7}{4}u_y^{*2}u_x^* + \tfrac{1}{2}u_x^*u_z^{*2} + \tfrac{1}{3}u_x^*) + \tfrac{1}{\rho^*}F_z(-\tfrac{3}{4}u_z^{*3} - \tfrac{7}{4}u_y^{*2}u_z^* + \tfrac{1}{2}u_z^*u_x^{*2} + \tfrac{1}{3}u_z^*)\big) \\
& - \tfrac{5}{4}u_y^{*4} + \tfrac{1}{4}u_x^{*4} + \tfrac{1}{4}u_z^{*4} - u_y^{*2}u_x^{*2} - u_y^{*2}u_z^{*2} + \tfrac{1}{2}u_x^{*2}u_z^{*2} + \tfrac{1}{3}u_y^{*2} + \tfrac{1}{3}u_x^{*2} + \tfrac{1}{3}u_z^{*2} + (\tfrac{5}{4}u_y^{*2} - \tfrac{1}{4}u_x^{*2} - \tfrac{1}{4}u_z^{*2})(S_{yy}^* + S_{xx}^* + S_{zz}^*) \\
& + \tfrac{1}{3}(2S_{yy}^* - S_{xx}^* - S_{zz}^*) + \tfrac{1}{\rho^*}F_yu_y^* + \tfrac{1}{\rho^*}F_y(\tfrac{5}{4}u_y^{*3} - \tfrac{1}{4}u_y^*u_x^{*2} - \tfrac{1}{4}u_y^*u_z^{*2} - \tfrac{1}{3}u_y^*) + \tfrac{1}{\rho^*}F_x(-\tfrac{1}{4}u_x^{*3} + \tfrac{5}{4}u_y^{*2}u_x^* - \tfrac{1}{4}u_x^*u_z^{*2} - \tfrac{1}{3}u_x^*) \\
& + \tfrac{1}{\rho^*}F_z(-\tfrac{1}{4}u_z^{*3} + \tfrac{5}{4}u_y^{*2}u_z^* - \tfrac{1}{4}u_z^*u_x^{*2} - \tfrac{1}{3}u_z^*)\,; \\
S_{zz}(\boldsymbol{x},t+1) = {} & \tfrac{1}{\tau^2}\big(u_z^{*4} - \tfrac{1}{2}u_y^{*4} - \tfrac{1}{2}u_x^{*4} - \tfrac{1}{4}u_z^{*2}u_y^{*2} - \tfrac{1}{4}u_z^{*2}u_x^{*2} + \tfrac{1}{2}u_y^{*2}u_x^{*2} - \tfrac{1}{2}u_z^{*2}(2S_{zz}^* - S_{yy}^* - S_{xx}^*) + \tfrac{1}{4}u_y^{*2}(2S_{yy}^* - S_{zz}^* - S_{xx}^*) \\
& + \tfrac{1}{4}u_x^{*2}(2S_{xx}^* - S_{zz}^* - S_{yy}^*) + \tfrac{1}{\rho^*}F_z(-u_z^{*3} - \tfrac{1}{4}u_z^*u_y^{*2} - \tfrac{1}{4}u_z^*u_x^{*2}) + \tfrac{1}{\rho^*}F_y(\tfrac{1}{2}u_y^{*3} + \tfrac{1}{2}u_z^{*2}u_y^* - \tfrac{1}{4}u_y^*u_x^{*2}) + \tfrac{1}{\rho^*}F_x(\tfrac{1}{2}u_x^{*3} + \tfrac{1}{2}u_z^{*2}u_x^* - \tfrac{1}{4}u_x^*u_y^{*2})\big) \\
& + \tfrac{1}{\tau}\big(-2u_z^{*4} + \tfrac{1}{4}u_y^{*4} + \tfrac{1}{4}u_x^{*4} + \tfrac{5}{4}u_z^{*2}u_y^{*2} + \tfrac{5}{4}u_z^{*2}u_x^{*2} - u_y^{*2}u_x^{*2} + \tfrac{2}{3}u_z^{*2} - \tfrac{1}{3}u_y^{*2} - \tfrac{1}{3}u_x^{*2} + \tfrac{1}{4}u_z^{*2}(8S_{zz}^* - 7S_{yy}^* - 7S_{xx}^*) \\
& + \tfrac{1}{4}u_y^{*2}(2S_{zz}^* - S_{yy}^* + 2S_{xx}^*) + \tfrac{1}{4}u_x^{*2}(2S_{zz}^* + 2S_{yy}^* - S_{xx}^*) - \tfrac{1}{3}(2S_{zz}^* - S_{yy}^* - S_{xx}^*) + \tfrac{1}{\rho^*}F_z(2u_z^{*3} + \tfrac{1}{2}u_z^*u_y^{*2} + \tfrac{1}{2}u_z^*u_x^{*2} - \tfrac{2}{3}u_z^*) \\
& + \tfrac{1}{\rho^*}F_y(-\tfrac{3}{4}u_y^{*3} - \tfrac{7}{4}u_z^{*2}u_y^* + \tfrac{1}{2}u_y^*u_x^{*2} + \tfrac{1}{3}u_y^*) + \tfrac{1}{\rho^*}F_x(-\tfrac{3}{4}u_x^{*3} - \tfrac{7}{4}u_z^{*2}u_x^* + \tfrac{1}{2}u_x^*u_y^{*2} + \tfrac{1}{3}u_x^*)\big) \\
& - \tfrac{5}{4}u_z^{*4} + \tfrac{1}{4}u_y^{*4} + \tfrac{1}{4}u_x^{*4} - u_z^{*2}u_y^{*2} - u_z^{*2}u_x^{*2} + \tfrac{1}{2}u_y^{*2}u_x^{*2} + \tfrac{1}{3}u_z^{*2} + \tfrac{1}{3}u_y^{*2} + \tfrac{1}{3}u_x^{*2} + (\tfrac{5}{4}u_z^{*2} - \tfrac{1}{4}u_y^{*2} - \tfrac{1}{4}u_x^{*2})(S_{zz}^* + S_{yy}^* + S_{xx}^*) \\
& + \tfrac{1}{3}(2S_{zz}^* - S_{yy}^* - S_{xx}^*) + \tfrac{1}{\rho^*}F_zu_z^* + \tfrac{1}{\rho^*}F_z(\tfrac{5}{4}u_z^{*3} - \tfrac{1}{4}u_z^*u_y^{*2} - \tfrac{1}{4}u_z^*u_x^{*2} - \tfrac{1}{3}u_z^*) + \tfrac{1}{\rho^*}F_y(-\tfrac{1}{4}u_y^{*3} + \tfrac{5}{4}u_z^{*2}u_y^* - \tfrac{1}{4}u_y^*u_x^{*2} - \tfrac{1}{3}u_y^*) \\
& + \tfrac{1}{\rho^*}F_x(-\tfrac{1}{4}u_x^{*3} + \tfrac{5}{4}u_z^{*2}u_x^* - \tfrac{1}{4}u_x^*u_y^{*2} - \tfrac{1}{3}u_x^*)\,.
\end{aligned}
$$