High-Order Moment-Encoded Kinetic Simulation of Turbulent Flows
Wei Li, Tongtong Wang, Zherong Pan, Xifeng Gao, Kui Wu, Mathieu Desbrun

To cite this version:
Wei Li, Tongtong Wang, Zherong Pan, Xifeng Gao, Kui Wu, et al.. High-Order Moment-Encoded Kinetic Simulation of Turbulent Flows. ACM Transactions on Graphics, 2023, 42 (6), 10.1145/3618341. hal-04223237

HAL Id: hal-04223237
https://inria.hal.science/hal-04223237
Submitted on 29 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
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.
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


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; Azvedo 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; Azvedo 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.
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 ≥ 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 \( f(\mathbf{v}, t) \) representing the probability of particle being at position \( \mathbf{x} \) at time \( t \) and with velocity \( \mathbf{v} \). This is in sharp contrast to typical NS-based method modeling the flow dynamics with the time evolution of a macroscopic velocity \( \mathbf{u}(\mathbf{x}, t) \). 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]:

\[
\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f = \Omega(f) + F \cdot \nabla f,
\]

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

\[
f^{eq} = \frac{\rho}{(2\pi \tau T)^{d/2}} \exp\left(-\frac{||\mathbf{v} - \mathbf{u}||^2}{2\tau T}\right),
\]

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

\[
\rho = \int f \, d\mathbf{v}, \quad \rho \mathbf{u} = \int \mathbf{v} \, f \, d\mathbf{v}, \quad \rho S_{a\beta} = \int (\mathbf{v}^2 - \frac{1}{3} \delta_{a\beta}) \, f \, d\mathbf{v},
\]

where the greek indices \( a \) and \( \beta \) refer to tensor coordinates, i.e., \( S = (S_{a\beta})_{a,\beta} \) for \( a, \beta \in \{x,y,z\} \).

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

**Lattice Boltzmann equations.** With time \( t \) discretized through regular timesteps, space \( x \) discretized through a regular grid, and the mesoscopic velocity \( \mathbf{v} \) 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:

\[
f_i(x + c_i \cdot t + 1) - f_i(x,t) = \Omega_i + F_i, \tag{4}
\]

where \( f_i(x,t) \) encodes the distribution \( f \) in the \( i \)-th direction at position \( x \) and time \( t \), \( c_i \) is the discrete lattice velocity in the \( i \)-th direction (Fig. 4), \( \Omega_i \) 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

$$f_i^{n+1}(x, t) = f_i(x - c_i, t), \quad (5)$$

followed by a collision step expressed as

$$f_i(x, t+1) = f_i^n(x, t) + F_i. \quad (6)$$

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

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

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; another 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 $\rho, \rho u, \rho S$, a distribution function is reconstructed via (we use Einstein notation to indicate tensor contraction):

$$f_i(x, t) = \rho(x - c_i, t) \omega_i \left( 1 + 3u(x - c_i, t) \cdot c_i + \frac{1}{2 c_i S_{\alpha\beta}(x - c_i, t)} H_{\alpha\beta}^{[2]}(c_i) \right), \quad (8)$$

where $\omega_i$ are lattice weights and $H_{\alpha\beta}^{[2]} = [H_{\alpha\beta}^{[2]}]_{\alpha\beta}$ 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:

$$\rho u_a(x, t + 1) = \rho u_a(x, t) + \left( 1 - \frac{1}{\tau_a} \right) F_a, \quad (9)$$

$$\rho S_{\alpha\beta}(x, t + 1) = \left( 1 - \frac{1}{\tau_a} \right) \rho S_{\alpha\beta}(x, t) + \frac{1}{\tau_a} \rho u_a u_{\alpha\beta}(x, t) + \left( 1 - \frac{1}{\tau_a} \right) \left( F_{\alpha\beta} u_a + F_{\beta\alpha} u_a \right)(x, t). \quad (10)$$

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]:

$$f(\boldsymbol{v}, x, t) = \omega(\boldsymbol{v}) \sum_{n=0}^{N} \frac{1}{n!} H_n^{[1]}(\boldsymbol{v}) : a_n(\boldsymbol{x}, t), \quad (11)$$
where $N$ is the order of approximation; $\omega(\mathbf{v})$ is a weighting function defined as $\omega(\mathbf{v}) = \exp(-|\mathbf{v}|^2/(2\pi)^{d/2})$; the superscript $[n]$ indicates a rank-$n$ tensor; the operator $\cdot^n$ denotes full tensor contraction; and $H[n]$ is the Hermite polynomial of order $n$, a $n$-th order tensor defined as

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

(12)

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

$$a[n](x,t) = \int [\mathbf{v}, x, t] H[n](\mathbf{v})\,d\mathbf{v}.$$  

(13)

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

$$a[0] = \rho, \quad a[1] = \rho \mathbf{u}, \quad a[2] = \rho\mathbf{S}.$$  

(14)

**Discrete distributions.** Now, when a lattice discretization of the local velocity $\mathbf{v}$ into a discrete microscopic velocities $\{c_i\}_{i=0}^{q-1}$ ($q=9$ 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 $f(\cdot, x, t)$ are the $q$ values $f_i(x, t)$ per node and per timestep defined as:

$$f_i(x, t) = \frac{w_i}{\omega(c_i)} f(c_i, x, t).$$

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

$$a[n](x, t) = \int_i f_i(x, t) H[n](c_i).$$  

(16)

One can verify that the first three orders of coefficients $a[0], a[1],$ and $a[2]$ 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 $\rho, \mathbf{u},$ and $\mathbf{S}$ 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 $a[0], a[1], a[2]$ in the Hermite expansion and setting these coefficients to the three known moments (Eq. (14)). This low-order reconstruction of the 27-direction values $\mathbf{f}_i$ 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 $f$ of a node, the first three velocity moments $\rho, \mathbf{u},$ and $a[2]$ are evaluated, and a particular choice of a second-order Hermite truncated series for the off-equilibrium part $\mathbf{f}^{eq}$ (i.e., the original distribution minus the equilibrium distribution $\mathbf{f}^{eq}(\rho, \mathbf{u})$) 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 $a[n>2]$ are recursively evaluated from $\rho, \mathbf{u},$ and $a[2]$ 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 $\rho$ (to which we set $a[0]$), $\mathbf{u}$ (to which we set $a[1]$, and $\rho \mathbf{S}$ (to which we set $a[2]$), we can use the recursive computation of third-order coefficients $a[3]$ 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 $a[n>3]$ 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 $\rho, \rho \mathbf{u},$ and $\rho \mathbf{S}$, leading in 3D to the following third-order Hermite reconstruction expression for each of the distribution function $f_i$:

$$f_i = \rho \omega_i \left[ 1 + \frac{c_i \cdot \mathbf{u}}{a_i^2} + \frac{H[2](c_i) \cdot \mathbf{S}}{2a_i^4} \right. \left. + \frac{1}{a_i^6} \left( H[3]_{XXY}(c_i)(S_{XX} u_y + 2S_{XY} u_x - 2u_x u_x u_y) + H[3]_{XYY}(c_i)(S_{YY} u_x + 2S_{XY} u_y - 2u_y u_y u_x) + H[3]_{XXZ}(c_i)(S_{XX} u_z + 2S_{XZ} u_x - 2u_x u_x u_z) + H[3]_{XZZ}(c_i)(S_{ZZ} u_x + 2S_{XZ} u_z - 2u_z u_z u_x) + H[3]_{YZZ}(c_i)(S_{YY} u_z + 2S_{YZ} u_y - 2u_y u_y u_z) + H[3]_{YYZ}(c_i)(S_{YY} u_x + 2S_{XY} u_y - 2u_x u_y u_y) + H[3]_{ZZZ}(c_i)(S_{ZZ} u_z + 2S_{ZS} u_z - 2u_z u_z u_z) \right). \right)$$

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

$$\Omega = -M^{-1}(R(m - m^{eq}) + (I - \frac{1}{2}R)K),$$

where $\Omega$ comprises all collision terms $\Omega_i$, $R$ is the diagonal matrix containing all individual relaxation rates $r_i$, and $K$ 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 $\rho^*$, $u^*$, and $S^*$ 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 $\rho^*$ and $u^*$ on a grid node allows us to know the equilibrium distribution defined by Eq. (2). The resulting central-moment $m^{eq} = M f^{eq}$ and the external force $F$ 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:

$$m_0^{eq} = m_0 = \rho, \quad m_1^{eq} = \frac{1}{2} \rho, \quad m_1^{eq} = \frac{1}{2} \rho, \quad m_6^{eq} = \frac{1}{27} \rho\,$$

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

$$K_1 = \frac{1}{2} F_x, \quad K_2 = F_y, \quad K_3 = F_z,$$

$$K_{10} = \frac{1}{3} F_x, \quad K_{11} = \frac{1}{3} F_y, \quad K_{12} = \frac{1}{3} F_z,$$

$$K_{23} = \frac{1}{3} F_x, \quad K_{24} = \frac{1}{3} F_y, \quad K_{25} = \frac{1}{3} F_z.$$

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 $M$ and its inverse, as the closed-form expressions of the three velocity moments (found with Matlab [Matlab 2023]) are:

$$\rho(x, t + 1) = \rho^*; \quad u_\alpha(x, t + 1) = \frac{1}{\rho^*} F_\alpha;$$

$$S_{xy}(x, t + 1) = \left(1 - \frac{1}{3} S_{xy}^* - \frac{2}{3} \frac{x^2 y^2}{\rho^*} \right) + \frac{1}{\rho^*};$$

$$S_{xx}(x, t + 1) = \left(1 - \frac{1}{3} S_{xx}^* - \frac{2}{3} \frac{x^2}{\rho^*} \right) + \frac{1}{\rho^*};$$

$$S_{yy}(x, t + 1) = \left(1 - \frac{1}{3} S_{yy}^* - \frac{2}{3} \frac{y^2}{\rho^*} \right) + \frac{1}{\rho^*};$$

$$S_{zz}(x, t + 1) = \left(1 - \frac{1}{3} S_{zz}^* - \frac{2}{3} \frac{z^2}{\rho^*} \right) + \frac{1}{\rho^*};$$

Note that these expressions (except for the trivial Eqs. (21-22) which just reflect density and linear momentum preservation of 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 in our 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 $m = \{ m_i \}_{i=0, \ldots, 26}$, which are derived from distribution functions $f = \{ f_i \}_{i=0, \ldots, 26}$ via a linear transform $m = Mf$, where $M$ is a matrix known in closed form as a function of the macroscopic velocity $u$. By subtracting the equilibrium distribution projected into the same central moment space, one gets the off-equilibrium central moments $m - m^{eq}$, which are relaxed by individual rates $r_i$, before the result is converted back to a distribution by multiplying it by $M^{-1}$, which provides the collision terms $\Omega_i$ from Eq. (4) – see, for instance, [Li et al. 2020]. If one also incorporates external forces, the NOCM-MRT collision is expressed as:

$$\Omega = -M^{-1}(R(m - m^{eq}) + (I - \frac{1}{2}R)K),$$

where $\Omega$ comprises all collision terms $\Omega_i$, $R$ is the diagonal matrix containing all individual relaxation rates $r_i$, and $K$ 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 $\rho^*$, $u^*$, and $S^*$ 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 $\rho^*$ and $u^*$ on a grid node allows us to know the equilibrium distribution defined by Eq. (2). The resulting central-moment $m^{eq} = M f^{eq}$ and the external force $F$ 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:

$$m_0^{eq} = m_0 = \rho, \quad m_1^{eq} = \frac{1}{2} \rho, \quad m_1^{eq} = \frac{1}{2} \rho, \quad m_6^{eq} = \frac{1}{27} \rho\,$$

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

$$K_1 = \frac{1}{2} F_x, \quad K_2 = F_y, \quad K_3 = F_z,$$

$$K_{10} = \frac{1}{3} F_x, \quad K_{11} = \frac{1}{3} F_y, \quad K_{12} = \frac{1}{3} F_z,$$

$$K_{23} = \frac{1}{3} F_x, \quad K_{24} = \frac{1}{3} F_y, \quad K_{25} = \frac{1}{3} F_z.$$

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 $M$ and its inverse, as the closed-form expressions of the three velocity moments (found with Matlab [Matlab 2023]) are:

$$\rho(x, t + 1) = \rho^*; \quad u_\alpha(x, t + 1) = \frac{1}{\rho^*} F_\alpha;$$

$$S_{xy}(x, t + 1) = \left(1 - \frac{1}{3} S_{xy}^* - \frac{2}{3} \frac{x^2 y^2}{\rho^*} \right) + \frac{1}{\rho^*};$$

$$S_{xx}(x, t + 1) = \left(1 - \frac{1}{3} S_{xx}^* - \frac{2}{3} \frac{x^2}{\rho^*} \right) + \frac{1}{\rho^*};$$

$$S_{yy}(x, t + 1) = \left(1 - \frac{1}{3} S_{yy}^* - \frac{2}{3} \frac{y^2}{\rho^*} \right) + \frac{1}{\rho^*};$$

$$S_{zz}(x, t + 1) = \left(1 - \frac{1}{3} S_{zz}^* - \frac{2}{3} \frac{z^2}{\rho^*} \right) + \frac{1}{\rho^*};$$

Note that these expressions (except for the trivial Eqs. (21-22) which just reflect density and linear momentum preservation of 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. 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).
Weakly-compressible fluid, This approximation is in fact tantamount to assuming $f$ value

A solid by summing all the contributions for all nodes expressed in a Galilean-invariant way [Peng et al. 2016] as:

$$S^P = \sum_{i} (u^P_i - u_i^P).$$  (24)

This approximation is in fact tantamount to assuming $\nabla u|_x = \nabla u|_p$ since we know that the off-equilibrium part of $S$ 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 expressions 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 $p$, we evaluate the outcome value $f^P_i$ using Eq. (17) and stream it to $x$.

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:

$$F dt = f_{in}(c_m - u_p) - f_{out}(c_{out} - u_p).$$  (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 $x$ whose links $\ell_x$ intersect the solid to get the force $F_B$ and torque $\tau_B$ through:

$$F_B = \sum_{x} f^{\ell}_k(x, t) (c_{k} - u_p(x)) = f_{k}(x, t+1) (c_{k} - u_p(x)).$$  (26)

$$\tau_B = \sum_{x} (x - x_{C}) \times \sum_{\ell_x} f^{\ell}_k(x, t) (c_{k} - u_p(x))$$

where $k$ is the opposite direction to $k$, and we use $p(x)$ to denote the solid point $p$ on the link associated to grid node $x$, and $x_C$ 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 7 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>
<thead>
<tr>
<th>Figure</th>
<th>Resolution</th>
<th>$\tau$</th>
<th>$\delta$</th>
<th>Integration steps</th>
<th>Total time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1280x900x300</td>
<td>0.0002</td>
<td>0.0027</td>
<td>246</td>
<td>11,500</td>
</tr>
<tr>
<td>2</td>
<td>600x300x300</td>
<td>0.0001</td>
<td>0.0004</td>
<td>38</td>
<td>8,250</td>
</tr>
<tr>
<td>3</td>
<td>640x300x300</td>
<td>0.0001</td>
<td>0.0003</td>
<td>37</td>
<td>10,800</td>
</tr>
<tr>
<td>5</td>
<td>720x360x360</td>
<td>0.0001</td>
<td>0.0003</td>
<td>39</td>
<td>6,600</td>
</tr>
<tr>
<td>6</td>
<td>1178x990x624</td>
<td>0.0004</td>
<td>0.0013</td>
<td>132</td>
<td>13,500</td>
</tr>
<tr>
<td>13</td>
<td>250x300x120</td>
<td>0.0001</td>
<td>0.0004</td>
<td>36</td>
<td>4,600</td>
</tr>
<tr>
<td>15</td>
<td>960x210x480</td>
<td>0.0002</td>
<td>0.0007</td>
<td>71</td>
<td>6,600</td>
</tr>
<tr>
<td>16</td>
<td>400x200x400</td>
<td>0.0001</td>
<td>0.0004</td>
<td>31</td>
<td>41,600</td>
</tr>
<tr>
<td>17</td>
<td>400x200x400</td>
<td>0.0001</td>
<td>0.0001</td>
<td>27</td>
<td>21,200</td>
</tr>
</tbody>
</table>

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 \( \rho \) (we need to add \( \sum w_i = 1 \) 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 \( 8 \times 8 \times 4 \) 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 \( \mathbf{u} \) with:

\[
\begin{align*}
    u_x(x, 0) &= U_0 \cos(2\pi x / D) \sin(2\pi y / D) \sin(2\pi z / D), \\
    u_y(x, 0) &= -U_0 \sin(2\pi x / D) \cos(2\pi y / D) \sin(2\pi z / D) / 2, \\
    u_z(x, 0) &= -U_0 \sin(2\pi x / D) \sin(2\pi y / D) \cos(2\pi z / D) / 2.
\end{align*}
\]

where \( U_0 = 0.2, D = 128, \) and \( L_0 = D / (2\pi) \), and simulate the vortical flow for a duration of \( 100L_0 / U_0 \). By testing different Reynolds numbers \( Re \), we effectively vary the viscosity \( \nu \) of the flow since \( \nu = U_0L_0 / Re \). 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.

Flow past a sphere. We also evaluated our solver in Fig. 8 with different LBM solvers based on different collision models: (a) MR-LBM [Ferrari et al. 2023]; (b) central-moment model (NOCM [Li et al. 2020], (c) 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.

![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.

![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] (b) central-moment model (NOCM [Li et al. 2020]), (c) 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.](image)

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.](image)
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 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 $500 \times 1000 \times 500$, while the two others only go up to $350 \times 700 \times 350$ 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
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 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 real-time 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 real-time 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.

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 [Delery 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
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.


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.


\[ S_{xx}(x, t + 1) = \frac{1}{\rho} \left( u_x^4 - 4 u_x^3 u_y + 6 u_x^2 u_y^2 - 4 u_x u_y^3 + u_y^4 \right) + \frac{1}{\rho} F_x \left( u_x u_y^3 - 4 u_x^2 u_y u_z^2 + 6 u_x u_y^2 u_z^2 - 4 u_x u_y u_z^3 + u_y u_z^4 \right) + \frac{1}{\rho} F_y \left( 4 u_x^3 - 3 u_x^2 u_y^2 + 3 u_x u_y^3 + u_y^4 \right) \]

\[ + \frac{1}{\rho} F_z \left( 8 u_x^3 u_y + 24 u_x^2 u_y^2 - 24 u_x u_y^3 + 8 u_y^4 \right) \]

\[ S_{yy}(x, t + 1) = \frac{1}{\rho} \left( u_y^4 - 4 u_y^3 u_x + 6 u_y^2 u_x^2 - 4 u_y u_x^3 + u_x^4 \right) + \frac{1}{\rho} F_x \left( u_x u_y^3 - 4 u_x^2 u_y u_z^2 + 6 u_x u_y^2 u_z^2 - 4 u_x u_y u_z^3 + u_y u_z^4 \right) + \frac{1}{\rho} F_y \left( 4 u_x^3 - 3 u_x^2 u_y^2 + 3 u_x u_y^3 + u_y^4 \right) \]

\[ + \frac{1}{\rho} F_z \left( 8 u_x^3 u_y + 24 u_x^2 u_y^2 - 24 u_x u_y^3 + 8 u_y^4 \right) \]

\[ S_{zz}(x, t + 1) = \frac{1}{\rho} \left( u_z^4 - 4 u_z^3 u_x + 6 u_z^2 u_x^2 - 4 u_z u_x^3 + u_x^4 \right) + \frac{1}{\rho} F_x \left( u_x u_y^3 - 4 u_x^2 u_y u_z^2 + 6 u_x u_y^2 u_z^2 - 4 u_x u_y u_z^3 + u_y u_z^4 \right) + \frac{1}{\rho} F_y \left( 4 u_x^3 - 3 u_x^2 u_y^2 + 3 u_x u_y^3 + u_y^4 \right) \]

\[ + \frac{1}{\rho} F_z \left( 8 u_x^3 u_y + 24 u_x^2 u_y^2 - 24 u_x u_y^3 + 8 u_y^4 \right) \]