Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Assessment of numerical methods for fully resolved simulations of particle-laden turbulent flows

J.C. Brändle de Motta[a,b,*], P. Costa[c,d], J.J. Derksen[e], C. Peng[f], L.-P. Wang[f,g], W.-P. Breugem[c], J.L. Estivalezes[h,a,j], S. Vincent[i], E. Climent[a], P. Fede[a], P. Barbaresco[j], N. Renon[j]

[a] Institut de Mécanique des Fluides de Toulouse (IMFT), Université de Toulouse, CNRS, Toulouse, France
[b] COmplexe de Recherche Interprofessionnel en Arothermochimie (CORIA), Université de Rouen Normandie, CNRS, INSA de Rouen, Saint-Étienne du Rouvray, France
[c] Laboratory for Aero and Hydrodynamics, Delft University of Technology, Delft, The Netherlands
[d] KTH, Department of Mechanics, Stockholm SE-100 44, Sweden
[e] School of Engineering, University of Aberdeen, Aberdeen, UK
[f] Department of Mechanical Engineering, University of Delaware, Newark, Delaware, USA
[g] Department of Mechanics and Aerospace Engineering, Southern University of Science and Technology, China
[h] ONERA, The French Aerospace Lab, Toulouse, France
[i] Laboratoire Modélisation et Simulation Multi Echelle (MSME), CNRS, Université Paris-Est Marne-la-Vallée, Marne-la-Vallée, France
[j] CALMIP, Université de Toulouse, CNRS, Toulouse, France

ARTICLE INFO

Article history:
Received 21 May 2018
Revised 24 September 2018
Accepted 12 October 2018
Available online 13 October 2018

Keywords:
Particle-laden flows
Finite-size particles
Turbulence
Direct numerical simulations

ABSTRACT

During the last decade, many approaches for resolved-particle simulation (RPS) have been developed for numerical studies of finite-size particle-laden turbulent flows. In this paper, three RPS approaches are compared for a particle-laden decaying turbulence case. These methods are, the Volume-of-Fluid Lagrangian method, based on the viscosity penalty method (VoF-Lag); a direct forcing Immersed Boundary Method, based on a regularized delta function approach for the fluid/solid coupling (IBM); and the Bounce Back scheme developed for Lattice Boltzmann method (LBM-BB). The physics and the numerical performances of the methods are analyzed. Modulation of turbulence is observed for all the methods, with a faster decay of turbulent kinetic energy compared to the single-phase case. Lagrangian particle statistics, such as the velocity probability density function and the velocity autocorrelation function, show minor differences among the three methods. However, major differences between the codes are observed in the evolution of the particle kinetic energy. These differences are related to the treatment of the initial condition when the particles are inserted in an initially single-phase turbulence. The averaged particle/fluid slip velocity is also analyzed, showing similar behavior as compared to the results referred in the literature. The computational performances of the different methods differ significantly. The VoF-Lag method appears to be computationally most expensive. Indeed, this method is not adapted to turbulent cases. The IBM and LBM-BB implementations show very good scaling.

1. Introduction

Particle-laden flows are ubiquitous in many applications, ranging for example from sediment transport in rivers to droplet generation in clouds. Moreover, the understanding of the interaction between particles and the fluid flow is crucial for many industrial applications such as fluidized beds or droplet distribution in combustion chambers.

Particle-laden flows have been studied numerically with different point-wise and Eulerian approaches during the last 5 decades [1–3]. These approaches are based on different models describing the force exerted on the particles by the fluid. Such models depend on parameters such as the slip velocity between the particles and the fluid in the immediate surroundings and the solid mass fraction. These approaches have been applied to many applications [4].

However, depending on the flow regime and physical parameters, the applicability of these models may be compromised. Indeed, the main assumption of such models is that the flow length scales are much larger than the particles size. The solution is to develop approaches treating the solid-fluid interface explicitly. These

* Corresponding author.
E-mail address: jorge.brandle@coria.fr (J.C. Brändle de Motta).

https://doi.org/10.1016/j.compfluid.2018.10.016
0045-7930/© 2018 Elsevier Ltd. All rights reserved.

resolved particle simulations (RPS) do not involve any model assumptions concerning the size and shape of the particles [5].

In recent years many, methods have been proposed to carry out RPS. The first one is the so-called body-fitted approach. In the body-fitted approach, the mesh is adapted to deal with the changing fluid domain at each time step. This approach has been given up for 3D simulations because of the remeshing computational cost; see for example [6] for a discussion of the numerical efforts needed for this kind of simulations. In order to avoid this cost, different approaches have been proposed, where the flow is solved on a fixed Eulerian grid or lattice. These methods have become appealing because they are more efficient and easier to implement in existing parallel codes.

During the last decade, these fully resolved simulations have been used to treat:

- turbulent flows where Kolmogorov length scale of the turbulent carrier fluid is smaller than the particle radius, with homogeneous isotropic turbulence [7–11] or channel flow turbulence [12,13],
- turbulence enhancement by settling particles [14],
- fluidized beds [15], and
- sediment transport on bed load [16,17].

Each method has been validated against several academic cases, and therefore its accuracy has been addressed. Still, the applications are more complex than these academic cases where the fluid flow is more or less canonical. While these methods have a very high degree of maturity and are used in several studies, the authors typically use one particular method, and do not compare their results directly against other approaches for a 4-way coupling case with many particles. The differences between the RPS approaches can have an impact on the solution obtained in this complex cases. In order to ensure that the RPS approaches reproduce the same physical solutions, it is important to build a well-defined benchmark case closer to the applications and to compare different codes. The purpose of this paper is to analyze a benchmark test case comparing different RPS approaches in order to ensure the reliability of the solution for complex cases.

To the authors' knowledge, benchmarks for numerical simulations of particle-laden flows are scarce. For the point-wise approaches, a collaborative benchmarking was performed in the case of a wall-bounded turbulence [18]. In this benchmark, non-negligible differences on the statistics obtained from the different codes have been observed. For the RPS approaches, a systematic comparison was performed recently between the Lattice-Boltzmann bounce-back and the Direct forcing-fictitious domain method for turbulent channel flow laden with finite-size particles by Wang and co-workers [19,20]. They concluded that all results are the same qualitatively, but there are noticeable quantitative differences. The present paper goes further in this direction studying a specific turbulent case and comparing 3 different approaches.

In addition to the physical analysis, this paper will discuss the numerical performance of these methods.

Indeed, the RPS simulations consume millions of CPU hours. Thus, it is imperative to develop more efficient approaches to reduce the computational cost. Even if many papers present the speed-up of each method, the CPU time consumption have to be compared with other codes. Potentially, it is possible to develop a very slow code that scales linearly in parallel. A second purpose of this paper is to provide a reliable dataset of the CPU consumption of a given case.

The present paper is the result of a collaboration initially between the supercomputer center CALMIP and the IMFT laboratory. The primary objective was to benchmark different numerical methods for fully resolved particle-laden turbulent flows by running simulations for the very same flow case on the very same supercomputer. The intercomparison pertained both to the simulation results and the computational efficiency of the methods. Other laboratories joined the initial collaboration in order to benchmark their own in-house codes. The list of methods used are:

- The VoF-Lag method developed by IMFT and MSME laboratories [21].
- The Immersed Boundary Method (IBM) developed at the Laboratory for Hydro and Aerodynamics, TU Delft [22].
- The lattice-Boltzmann method based on an improved interpolated bounce-back scheme (LBM-BB), developed at the University of Delaware (UD) [10].

A similar code has also been included during this benchmark. The Lattice Boltzmann method-immersed boundary method (LBM-IBM), developed at the Alberta University and now at the University of Aberdeen [16]. Here, only a subset of results will be presented for this method.

The benchmark consists of many particles seeded in a homogeneous turbulent flow. As cited before, many groups have worked on particle-turbulence interactions with different codes [7–13]. Nevertheless, the differences on the configurations, such as the particle size of the turbulent parameters, do not permit a rigorous comparison between the codes. Here the initial turbulent flow and the position of the particles were shared among all the groups participating in the benchmark study. These conditions can be shared again upon request by contacting the corresponding author.

This paper is organized as follows. Section 2 presents the the governing equations for particle-laden flows and the RPS methods implemented. In Section 3 the benchmark case is presented and the single-phase turbulent flow is analyzed comparing the different codes. In Section 4 the comparisons between the different methods for the particle-laden flow are given. Finally, a comparison of numerical performance is provided in Section 5.

2. Numerical approaches

2.1. Governing equations

The fluid flow simulation in this work is based on the incompressible Navier–Stokes equations. The discretized physical variables are the pressure, p, and the velocity field, $\mathbf{u}$. The mass conservation and momentum equations in the fluid domain Ω_f, is given as

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = \frac{1}{\rho} \nabla \cdot \boldsymbol{\sigma} + \mathbf{g} \tag{2}$$

are solved, where ρ is the fluid density and $\boldsymbol{\sigma}$ is the stress tensor based on the constant dynamic viscosity μ:

$$\boldsymbol{\sigma} = -p\mathbf{I} + \nabla \cdot \left(\mu\left(\nabla \mathbf{u} + \nabla^t \mathbf{u}\right)\right) \tag{3}$$

The solid particles are considered as rigid, i.e., no deformation is taken into account. Thus, we can write the velocity at any point $\mathbf{M}$ of the ith particle domain, Ω_s^i as:

$$\mathbf{u}_i(M) = \mathbf{U}_i + \boldsymbol{\omega}_i \times (\mathbf{M} - \mathbf{O}_i) \tag{4}$$

where $\mathbf{U}_i$ and $\boldsymbol{\omega}_i$ are the velocity and angular velocity vectors of the ith particle and $\mathbf{O}_i$ the mass center position.

The time evolution of each particle is given by the Newton-Euler equations:

$$\begin{aligned} m_i \frac{d\mathbf{U}_i}{dt} &= \mathbf{F}_i + m_i \mathbf{g} + \mathbf{F}_{coll} \\ I_i \frac{d\boldsymbol{\omega}_i}{dt} &= \mathbf{T}_i + \mathbf{T}_{coll} \end{aligned} \tag{5}$$

Here, m_i and I_i are the mass and the moment of inertia of the ith particle, $\mathbf{F}_i = \int_{\Gamma_i} \boldsymbol{\sigma} \cdot \mathbf{n} dA$ is the force exerted by the fluid on the particle, and $\mathbf{T}_i = \int_{\Gamma_i} \mathbf{r} \times (\boldsymbol{\sigma} \cdot \mathbf{n}) dA$ is the hydrodynamic torque, where $\mathbf{r}$ is the vector connecting the center of mass to the surface infinitesimally small area, dA. The forces $\mathbf{F}_{coll}$ and $\mathbf{T}_{coll}$ are the collision forces and torques among particles. In this benchmark study, the collision torque is not taken into account. The particles are considered as spherical.

In order to couple both phases, a no-slip and no-penetration velocity condition is considered. On any point M at the surface of the ith particle, $\Omega_s^i \cap \Omega_f$, the fluid velocity is considered to be

$$\mathbf{u}(M) = \mathbf{u}_i(M) \tag{6}$$

where $\mathbf{u}_i(M)$ is given by Eq. (4).

The different methods for solving these coupled equations are given in the following section.

2.2. Methods for fully resolved particle simulations

Many methods exist for fully resolved simulation of particles; see [5] for a recent review.

The body-fitted methods, also known as Arbitrary Lagrangian Eulerian method (ALE) have been developed for this application [23]. The main benefit of this method is that the accuracy of the boundary layer can be controlled. In this method, an unstructured grid is adapted to the fluid domain. At each time step, the forces are computed on the particle surface, then each particle is advected and the grid is updated. This method generates some problems such as the interpolation of the variables in the updated mesh, the meshing of the inter-particulate gap, and the dynamic evolution of the connectivity on the unstructured mesh. Nevertheless, the main reason why this method is not often used is that, even with the recent efforts, remeshing is still very expensive and often complex.

Another solution to maintain a body-fitted resolution of the particle boundary layer is the overset grid approach, also known as chimera approach [24,25]. This method has been recently extended to moving particles [26]. In this method, two meshes are considered: a fixed mesh covering all the physical domain and a mesh of the spherical domain around the particle. At each time step both meshes exchange information in order to converge the fluid solution. When the solution is found, the forces on the particle are computed and the grid associated to each particle moves. In this method, solvers for structured meshes can be used. This method becomes more complex when many particles have to be considered. Thus, the main limitation is the distance between the particles. In the method presented in [26] at least ten grid points are required in the particles gap.

Finally, the majority of methods used in today's applications are based on fixed Cartesian Eulerian grids. In these methods, a structured mesh covers the domain and the particles are implemented with different approaches. In some of them, the so-called fictitious domain approaches, the Navier–Stokes equations are solved in the entire domain, including the solid region. Among these methods the Physalis method considers the analytical solution near the particle interface in order to impose the no-slip condition [27,28]. This method has an original treatment of the particle boundary condition and is currently used for many applications. Other popular methods, which have been used in the present work, are described in the next subsections.

2.3. VoF-lag method

The VoF-Lag method is a viscosity penalty method based on the assumption that the Navier-Stokes equation (Eq. (2)) converges to

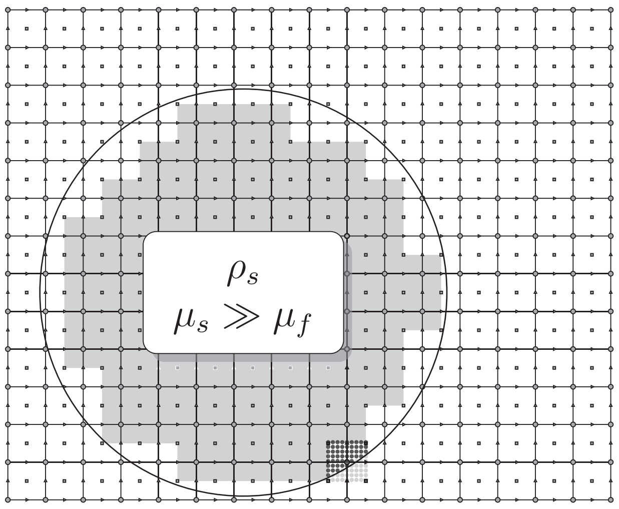


Fig. 1. Density and viscosity of the VoF-Lag approach applied to a staggered grid. Nodes are represented with: circles (pressure), triangles (velocity) and squares (transverse viscosity nodes).

the solid body dynamics (Eq. (4)), when the viscosity tends to infinity [21]. The basic idea is to use a large viscosity for the solid region in order to ensure the solid behavior, typically, in the present work, the solid viscosity is 300 times larger than the fluid viscosity. An interesting feature is that the VoF-Lag method solves simultaneously the solid and fluid velocity fields.

For this approach, three major problems have to be addressed. First of all, the physical fields such as the viscosity and density have to be accurately computed. Secondly, the Navier–Stokes solver needs to be robust and deal with high viscosity ratios. Finally, the particle transport and collision have to be treated.

2.3.1. Physical parameters

The density and the equivalent viscosity have to be computed. To do so, the solid fraction is computed at each time step, after the update of the position of the particles.

In order to obtain the solid volume fraction, C, on the volume cells containing both solid and fluid, a straightforward method is used: 25^3 points are regularly distributed in the cell. Knowing the particle's centroid position and radius, the number of points inside the particle is counted. An accurate value of the solid fraction is thus computed by averaging the number of points inside the particle divided by the total number of points, see Fig. 1. This method has been shown to be too expensive; see Section 5.

The density of the particle is directly obtained by an arithmetic average using the solid volume fraction:

$$\tilde{\rho} = C\rho_p + (1 - C)\rho \tag{7}$$

For the viscosity some additional computations are needed. In the method, two viscosity nodes are considered in order to enhance the spatial discretization order [21,29]. The phase indicator function is updated on the corresponding volume cell and a geometric average is used:

$$\tilde{\mu} = \frac{\mu\mu_s}{C\mu + (1 - C)\mu_s} \tag{8}$$

where, μ_s is the fictitious solid viscosity. This value is discussed in [21] and set to $\mu_s = 300\mu$.

2.3.2. Augmented Lagrangian solver

The Navier–Stokes equations are solved with iterative augmented Lagrangian approach [30]. This algorithm considers an iterative solution for the velocity and pressure fields, at each time step ($\mathbf{u}^{*,m}$, $p^{*,m}$). The iterations start with the velocity and pressure field of the previous time step n: $\left(\mathbf{u}^{*,0}, p^{*,0}\right) = (\mathbf{u}^n, p^n)$ and make the following iterative steps until the divergence-free condition is ensured $\|\nabla \cdot \mathbf{u}^{*,m}\| \ll \epsilon$:

$$\begin{array}{ll} \tilde{\rho}\left(\frac{\mathbf{u}^{*,m}-\mathbf{u}^n}{\Delta t} + \mathbf{u}^{*,m-1} \cdot \nabla \mathbf{u}^{*,m}\right) = -r\nabla(\nabla \cdot \mathbf{u}^{*,m}) & (a) \\ -\nabla p^{*,m-1} + \tilde{\rho}\mathbf{g} + \nabla\left(\tilde{\mu}\left(\nabla \mathbf{u}^{*,m} + \nabla^t \mathbf{u}^{*,m}\right)\right) & : \\ p^{*,m} = p^{*,m-1} - r\nabla \cdot \mathbf{u}^{*,m} & (b) \end{array} \tag{9}$$

where, r is the augmented Lagrangian parameter and m the iteration number. The converged velocity provides the velocity field at the next time step $\mathbf{u}^{n+1} = \mathbf{u}^{*,m}$.

BiCGStab II solver, coupled with a Modified and Incomplete LU (MILU) preconditioner, is implemented to solve the linear system for $\mathbf{u}^{*,m}$. At the end, the Augmented Lagrangian solver is very efficient in solving finite-size particle flows with various density and viscosity ratios while simultaneously satisfying time the incompressibility constraint. No pressure Poisson equation need to be solved. The main disadvantage of the approach is that it hardly scales under MPI parallel computations beyond several thousands of processors. Full details of the method are given in [30] and [21].

2.3.3. Lagrangian tracking

In order to update the positions of the particles the VoF-Lag method uses the velocity field obtained from the Navier–Stokes solution. In total, six points are used at the interior of each particle, 2 in each direction on either side of the center of the particle, after which the solid velocity field is interpolated. Then, the velocity and angular velocity are computed, $\mathbf{U}_i^{n+1}$ and $\boldsymbol{\omega}_i^{n+1}$. Using a second-order time integration scheme, the position of each particle is updated.

Before each time step, and with the new position and velocities, a parallel algorithm is used in order to detect collisions between particles. The particles are tracked in parallel with a master-slave algorithm where each processor only tracks the particles in its computational subdomain. A collision force is then computed and distributed over all the solid domain. This force is computed with the solid-solid interaction model [31]. Each collision is treated with a spring and damping coefficient in order to ensure that the numerical collision time takes 8 Navier–Stokes solver time steps. During these 8 time steps the particles overlap. Lubrication corrections are not included in order to ensure compatibility with the other codes used in the present benchmark study. The computed collision force becomes a source term in Eq. (9) (a).

This method has been validated for simple academic cases (sedimentation, rotation, shear) and has been used to study particle-turbulence interactions [11] and fluidized bed [15].

2.4. Immersed-boundary method

2.4.1. Numerical method

The method combines a standard second-order finite-volume pressure-correction scheme with a direct forcing IBM, as described in [22]. The IBM uses two grids, a 3D Eulerian grid, and a quasi-2D Lagrangian grid. The Eulerian grid discretizes the fluid phase, in a regular, Cartesian, marker-and-cell collocation of velocity and pressure nodes; the Lagrangian grid discretizes the surface of the spherical particles.

The idea of the direct forcing IBM can be briefly described as follows. First, the fluid prediction velocity is *interpolated* from the Eulerian to a Lagrangian grid. There the force required in each Lagrangian node for satisfying no-slip and no-penetration condition is computed. Subsequently, the force is *spread* back to the Eulerian grid. A regularized Dirac delta function with support of 3 grid cells is used to perform interpolation and spreading operations [32,33]; see Fig. 2. These forces on Lagrangian nodes for each particle are integrated in order to obtain the force $\mathbf{F}_i$ and torque $\mathbf{M}_i$ needed to update the particle velocity and angular velocity, see Eq. (5).

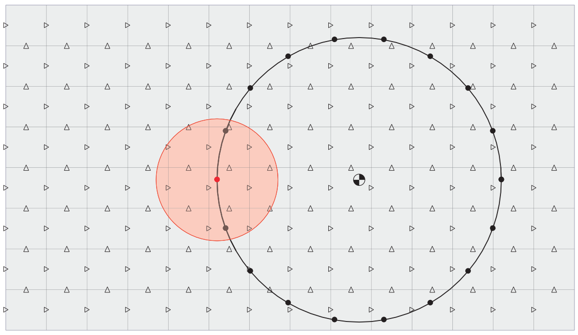


Fig. 2. Illustration of the IBM discretization in 2D. A regular Eulerian grid discretizes the fluid phase in the entire domain (triangles denote the collocation of the two velocity components). The particle surface is discretized with a distribution of Lagrangian grid points (solid black circles). A discrete regularized Dirac delta function with support of three cells (highlighted in red) is used to perform interpolation/spreading operations.

Regularization of the particle-fluid interface can result in a loss of spatial accuracy to first-order. In [22] it is shown that slight inward retraction of the Lagrangian grid by a factor $\approx \Delta x/3$ (while the particle governing equations are still solved considering its physical radius) circumvents this issue and allows for second-order spatial accuracy.

The support of the interpolation kernel is such that the same Eulerian grid point can be forced due to neighboring Lagrangian grid points, reducing the accuracy of the velocity forcing. Errors in penetration velocity arising from this are mitigated with a multi-direct forcing scheme [34], which improves the calculation of the force distribution by iterating the forcing scheme.

Finally, the method developed in [35] is used to compute collision forces between particles at contact. The forces are modeled by a soft-sphere collision model, which stretches the collision time to $\mathcal{O}(10)$ time steps of the Navier–Stokes solver. This choice is computationally attractive and physically realistic, as long as the prescribed collision time is much smaller than the characteristic time scale of particle motion.

2.4.2. Computational implementation

The algorithm is implemented in a distributed-memory parallelization framework. The three-dimensional regular Eulerian grid is divided into several computational subdomains. In most steps of the numerical algorithm, these share the total length of the domain in one direction, being of equal or smaller size than the domain length in the other directions. This configuration is commonly denoted as two-dimensional *pencil*-like decomposition. Following common practice, halo cells are used to store a copy of data pertaining to the boundary of an adjacent subdomain, in order to comply to the 2-cell width of the finite-difference stencil.

The numerical algorithm takes advantage of a direct, FFT-based solver for the finite-difference Poisson equation for the correction pressure [36]. To perform the Fourier transforms, the data distribution is transposed, such that it is shared in the direction of interest. Data transpose routines from the highly-scalable 2DECOMP&FFT library [37] are used to achieve this.

The particles are parallelized with a master-slave technique, conceptually similar to the one in [38]. The load due to particle-related computations is spread to the computational subdomains (tasks) containing the Eulerian data required for interpolation and spreading operations, which is – like the fluid velocity data – distributed in a 2D pencil configuration. The *master* process of a certain particle corresponds to the computational subdomain containing its centroid, and *slaves* to other subdomains crossing the particle-fluid interface (also accounting for the support of the IBM interpolation kernel).

Of all the operations required when including particles in the computation, the IBM forcing scheme is the most intensive. Implementing it in a distributed-memory parallelization requires some communication, as data required to perform interpolation/spreading operations can be distributed over different computational subdomains. In the present simulations, the data is communicated in a *Lagrangian* framework, in five-steps: (I) for the interpolation step, each task computes the partial sum for the interpolated velocity pertaining to Eulerian grid points in its subdomain; (II) the partial sums are communicated to the master process; (III) the master process then accumulates the sums, thereby computing the interpolated velocity and computes the resulting discrete IBM force at each Lagrangian grid point; (IV) the master process communicates the total force to the different slave processes; and (V) each process spreads the force back onto the Eulerian grid; see [39].

Recent improvements in the parallelization of the forcing scheme have been performed, see [39]. The underlying idea is to cover the support of the stencil of the IBM kernel through a 2-cell halo region. This way, interpolation and spreading operations can be performed solely by the computational subdomain containing a certain Lagrangian grid point. The advantage of this Eulerian parallelization of the IBM forcing scheme is that the communication load is known a priori, and decreases monotonically with increasing number of subdomains. This approach resulted in a very large speedup of the particle treatment (e.g. a speedup of more than a factor 2 of the particle treatment for simulations of suspensions at 20% solid volume fraction), but was not yet implemented during the course of this work.

2.5. LBM-BB method

The LBM-BB approach is based on the studies reported in [9,10]. For the fluid flow evolution, the multiple-relaxation-time (MRT) lattice Boltzmann method [40] is implemented in order to resolve the Navier–Stokes equations. The LBM solves the evolution of lattice-particle distribution functions at fixed nodes in the fluid region only. While the MRT collision model is computationally more expensive than the single-relaxation-time or BGK collision model, due to the calculation of the moments, MRT LBM provides greater control over relaxation parameters leading to a better numerical stability. The lattice velocity model is the standard D3Q19, from which 19 independent moments can be constructed at each node [40]. Compared to the conventional Navier–Stokes solvers, certainly more variables at each node location are solved, but the benefits include a much simpler (*i.e.*, quasi-linear) governing equation for the lattice-particle distribution functions when compared to the Navier–Stokes equations, more flexible handling of complex geometry, and local data communication suitable for massive scalable implementation.

When applying the LBM-BB to turbulent flow simulations, several additional considerations are necessary. First, since the LBM is formulated based on weakly compressible flow equations, caution is taken to make sure that the local flow Mach number is small (typically less than 0.3). In the present simulations, the local maximum *Ma* at the initial time is about 0.25. This amounts to specification of hydrodynamic velocity scale in the lattice units. Second, previous experience has shown that roughly twice the grid resolution is needed when compared to the pseudo-spectral method [10]. This in fact is a rather fortunate outcome due to the fact that LBM has very low numerical dissipation since the advection in the Boltzmann equation is linear and can be handled essentially exactly. The grid resolution also must resolve the viscous boundary layers on the solid particles.

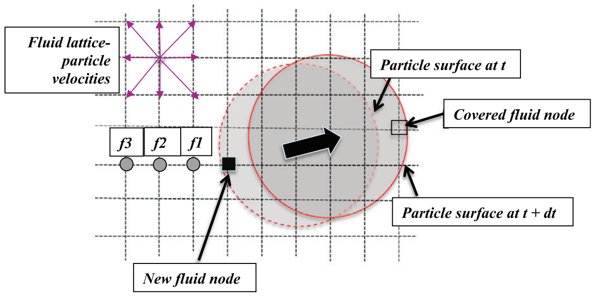


Fig. 3. Sketch to illustrate the key ideas for treating the fluid-solid interface in LBM-BB. The interpolated bounce-back scheme constructs an unknown distribution at a boundary node $f1$, at time t, in terms of known distributions at $f1$ and other nearby fluid nodes (say $f2$ and $f3$) as needed. The refilling would create distribution functions at the new fluid node. The momentum exchange algorithm then sums up the net momentum exchange at the all boundary nodes with links cutting through the surface of a solid particle.

Solid particles overlap with and move relative to the fixed fluid lattice nodes. In LBM-BB, no lattice-particle distributions functions are solved for any node inside a solid particle at any given time. When a solid particle moves relative to the fixed lattice grid during a time step, some lattice fluid nodes may be covered, and some nodes inside the solid may be uncovered. The distribution functions at the covered nodes are discarded, while the distribution functions at the uncovered nodes (or fresh fluid nodes) need to be constructed (Fig. 3). The no-slip boundary condition and hydrodynamic force $\mathbf{F}_i$ / torque $\mathbf{M}_i$ acting on ith solid particle have to be considered, see Eqs. (5) and (6).

2.5.1. Implementation

When solid particles are inserted into the flow and interact with the flow field, three issues have to be considered carefully [41]. The first aspect is how to realize the no-slip boundary condition on a moving curved wall. The current LBM-BB approach uses an interpolated bounce-back scheme presented in [42], which is a sharp solid-fluid interface treatment. Compared to the immersed boundary method (IBM) which can be viewed as a smoothed solid-fluid interface treatment, the LBM-BB is found to be more accurate [43] but at the same time the LBM-BB tends to be numerically less stable. It is found that part of the reasons for numerical instability with the LBM-BB is associated with the refilling scheme, which is the second aspect for moving solid-particle simulation. The refilling step constructs the lattice-particle distributions at new fluid nodes. The LBM-BB approach utilizes a constrained extrapolation scheme for refilling [41] which was found to be numerically more stable for turbulent particle-laden flow simulation.

The third aspect concerns the computation of hydrodynamic force and torque acting on the moving solid particle. The desired method here is the momentum-exchange method which simply sums up exchanges of momentum of fluid-lattice particles when bouncing back from the solid particle surface. There have been various implementations of the momentum-exchange method in the literature [41], some of them do not satisfy the property of Galilean invariance. The LBM-BB adopts the specific version of the momentum-exchange method introduced in [44] which is shown to be suitable for accurate representation of moving solid particles.

Finally, when performing direct simulation of turbulent particle-laden flow with the moving fluid-solid interfaces directly resolved, an efficient scalable code implementation is necessary. The LBM-BB code uses two-dimensional domain decomposition to partition the field data for scalable implementation using MPI. In the last few years, the team developing LBM-BB method has optimized their code by incorporating the following code optimiza-

Table 1
Carrier flow parameters.

ρ [kg/m^3]	ν [m^2/s]	λ [m]	η [m]	τ_k [s]	$u^0_{r.m.s}$ [m/s]	T^0_e [s]	Re_λ [-]
1.0	$1.0\,10^{-3}$	$13.7\,10^{-2}$	$74.4\,10^{-4}$	$55.2\,10^{-3}$	$64.0\,10^{-2}$	0.8	87.6

tion techniques [45]. First, the collision substep and the streaming substep are fused together using the two-array method, as discussed in [45] along with other fusing algorithms. Another key optimization concerns data communication for fluid-solid lattice links when a solid particle occupies more than one sub-domain. A novel direct-request data communication is designed to transfer the minimum data set for fluid-solid interactions between subdomains [45]. It is found that the above optimizations reduced the CPU time by a factor of 4 to 8.5, when compared to the pre-optimization code, in the direct simulation of a turbulent particle-laden flow [45]. Further details of the LBM-BB approach can be found in [9,10,41,45].

3. Benchmark description

3.1. Physical parameters

Particle-laden flows in a homogeneous isotropic turbulence (HIT) have been studied both experimentally and numerically. On the one hand, the relative simplicity of this case in comparison to the industrial applications provides a perfect framework to understand many phenomena such as the preferential concentration, the particle distribution, and the turbulence modification by the dispersed phase. On the other hand, these issues have not been completely understood because of the large number of parameters concerned (turbulence level, density ratio, size of particles, solid volume fraction) and the different ways of analyzing the results. In particular, the effect of the size of the particles is a relatively recent topic and has only been studied during the last two decades, to some limited extent, starting with the work of ten Cate et al. [7]. Many of theses studies were carried out using RPS approaches. Due to these reasons, we decided to use an HIT flow to compare the different approaches.

Turbulence shows chaotic behavior, thus, the solution could differ from one code to another. In order to reduce the degrees of freedom associated to the modeling, some choices have been addressed.

The initial turbulent flow field was generated using a spectral code with 1024^3 modes. The forcing scheme proposed by Eswaran and Pope [46], was used to obtain a statistically stationary flow by adding a stochastic force on the spectral modes. After the flow reaches statistical stationary conditions, the forcing is shut down in order to study decaying turbulence. A short transient phase was computed in order to finally obtain a solution independent of the forcing scheme. This velocity field was used as the initial condition of the present benchmark study. The spectral solution had a Reynolds number based on the Taylor scale of $Re_\lambda = 87.6$, which is large enough to obtain an inertial range in the spectrum. The largest wave number treated is compared to the Kolmogorov length scale in order to ensure that the full spectrum is solved, [47], here $\kappa_{max}\eta = 3.81 > 1.5$. The initial eddy turnover time is $T^0_e = 0.8\,s$. Table 1 summarizes the parameters of this initial flow field.

In each code, the spectral solution was interpolated at the location of the velocity nodes. To allow better comparison the considered simulation is a decaying turbulence simulation, since the implementation of a forcing method increases the differences between the codes.

Table 2
Particle conditions for the benchmark.

case	α_v [%]	N_p [-]	ρ_p [kg/m^3]	ρ_p/ρ [-]	D [m]	D/η [-]	D/λ [-]	St_k [-]
512	3.0	4450	4.0	4.0	$14.7\,10^{-2}$	19.8	1.08	87.2
1024	3.0	35602	4.0	4.0	$73.6\,10^{-3}$	9.90	0.54	21.8

For the dispersed phase, we consider two cases depending on the mesh resolution. The first case is simulated with 512^3 grid nodes and the second with 1024^3 nodes. In both cases, the solid volume fraction is set to 3 %. This value was chosen as a compromise between the two extremes: it is dense enough to ensure a convergence in the statistics and at the same time the case is sufficiently dilute in order to be not dominated by collisions. In addition, in order to reduce the effect of collisions, only elastic collisions were implemented without taking into account any lubrication corrections when particles are very near to each other.

The initial positions of the particles are chosen randomly without any particle-particle spatial overlap, and these same positions were shared among the codes. At the beginning of the simulation, the ith particle velocity $\mathbf{U}_i$ was fixed as the fluid velocity at its center $\mathbf{O}_i$. The velocity was interpolated from the spectral solution. The initial angular velocity was set to zero for IBM, LBM-BB and LBM-IBM methods, $\boldsymbol{\omega}_i(t=0) = 0$.

The initial velocity and angular velocity are treated differently for the VoF-Lag method. Indeed, the particle momentum equations (5) are not solved. The solid region is solidified and yields the linear velocity and angular velocity of the particles. The initial velocity is only used for the Lagrangian tracking that needs the velocity at the previous time step.

For both cases, the ratio of the particle diameter to grid length was fixed to 12 in order to ensure a good resolution of the particle-fluid interfaces. Table 2 provides the particle parameters. Because the ratio between the particle diameter and the Kolmogorov length scale is 19.7 for the first case and 9.86 for the second case, one can expect finite-size effects. This ratio decreases with time as the Kolmogorov scale increases when the turbulent kinetic energy decreases. The finite size effect will be studied later in this paper. Even if for this case the Stokes number based on the Kolmogorov time scale, $St_k = \frac{\frac{\rho_p}{\rho}\frac{D^2}{18\nu}}{\tau_k}$, could be considered not very meaningful [48], we provide it only as a reference.

The density ratio between the particles and the fluid has been set to 4 due to our intention to have particles with moderate inertia. In addition, even if the codes considered here could take into account neutrally buoyant particles, some methods presented in the literature are not stable for density ratios below 1.2 [33].

A snapshot of the 1024^3 IBM simulation with the turbulent structures and particles positions is provided in Fig. 4. In this figure, one can observe a high degree of flow field details and confirm that the particle size is of the same order of magnitude as the turbulent structures as suggested by the D/λ ratio, see Table 2. This ratio decreases with time as the turbulent kinetic energy decreases.

3.2. Single-phase flow

The generated turbulent field is averaged in each code to obtain the turbulent statistics. The first comparison between different codes is done for the single-phase (i.e. unladen) case.

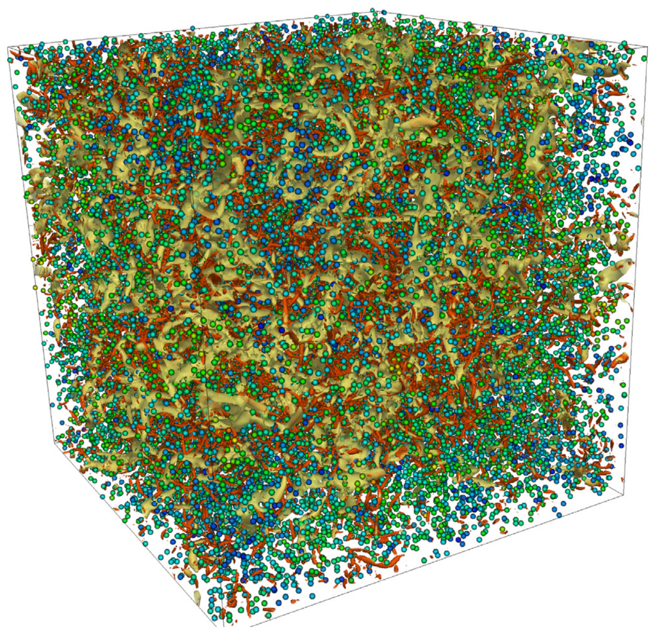


Fig. 4. Visualization of particle-laden decaying HIT. Particles are colored by their linear velocity (green-high and blue-low). Red denotes iso-surfaces of constant Q-criterion, while translucent yellow represents iso-surfaces of low pressure regions. Case 1024 simulated with IBM code, at time $1.25\,T_e^0 = 1\,s$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

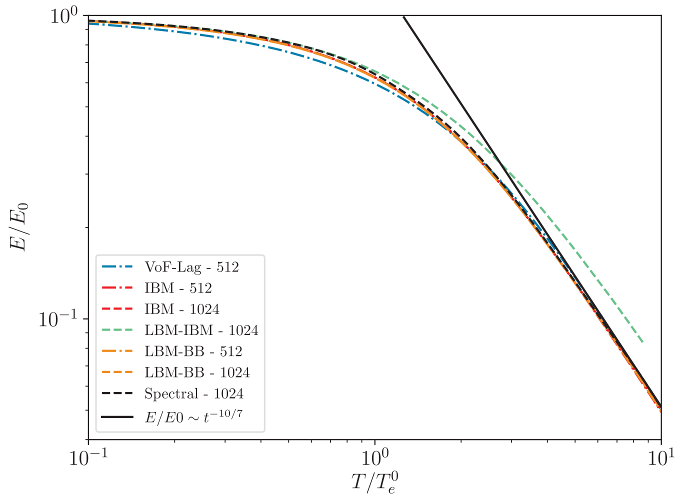
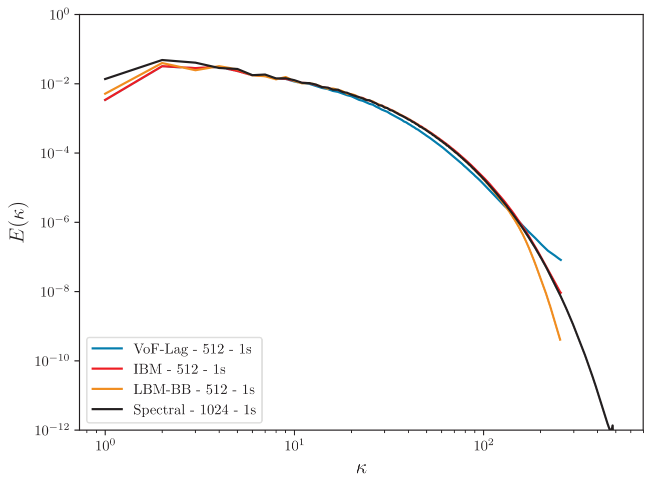


Fig. 5. Decaying fluid kinetic energy of single-phase flow. E_0 and T_0^e denote the values of kinetic energy and eddy turnover time at $T = 0$, respectively.

In Fig. 5 the time-dependent total turbulent kinetic energy is shown for each code. The total simulated time amounts nearly $10\,s = 12.5\,T_e^0$, and has been chosen in order to ensure that the total energy is still significant. In the present simulations the total energy at the end of the simulation is 2% of its initial value.

The dashed black line is the energy decay of turbulence obtained from the single-phase spectral code. It could be considered as the reference case. As expected, the energy decay is proportional to $t^{-10/7}$ [49]. All the codes reach this slope but there are some small differences. The VoF-Lag method seems to shift the initial energy level downwards, which explains the shift observed up to $t/T_0^e = 1$ in comparison to the other methods. This effect could be caused by the initial interpolation. Other difference could be seen for the LBM-IBM simulation. which is the slope is reached later than for the other methods. That is because for LBM approaches



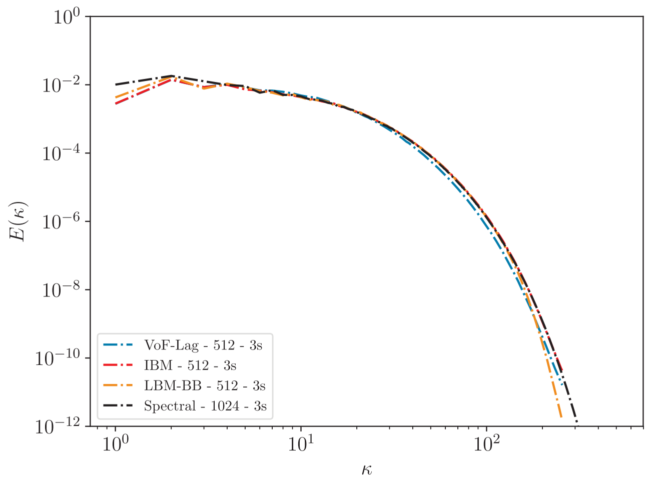


Fig. 6. Spectra for single-phase case for two given times. Top: $t = 1.25T_e^0$ ($1\,s$); Bottom: $t = 3.75T_e^0$ ($3\,s$).

the initial condition has to be carefully computed. For simulation with the LBM-BB code authors took the necessary precautions in order to obtain the appropriate initial distribution functions that are fully consistent with the macroscopic initial conditions [50]. For IBM and LBM-BB, both 512 and 1024 cases are presented. In the figure no difference can be seen. This result shows that even for the coarse mesh the turbulence decay is adequately resolved.

The spectra are now analyzed for the coarse mesh. These are computed from

$$E(\kappa) = \frac{1}{2} \sum_{|k-k_0/2|<|\chi|\leq|k+k_0/2|} \tilde{\mathbf{u}}(\chi) \cdot \tilde{\mathbf{u}}(\chi)^*, \tag{10}$$

where $\tilde{\mathbf{u}}$ is the Fourier transform of the velocity field, and $\kappa_0 = \pi/\Delta x$ is the largest wave number.

The spectra are given in Fig. 6 for two given times, with those computed from the spectral code given as reference.

The main differences appear for large wave numbers. Where the IBM solution collapses with the spectral solution, LBM-BB and VoF-Lag solutions slightly differ. The LBM-BB turbulent kinetic energy is below the energy provided by the spectral and IBM methods for both times. However, the authors have checked that the spectral solution is recovered for the LBM-BB finer mesh resolution. The finer results are not shown in the figure. Concerning the VoF-Lag method, it overpredicts turbulent kinetic energy at large wave numbers for $t = 1.25\,T_e^0$. At $t = 3.75\,T_e^0$, the result is in better agreement with the spectral method. Due to computational

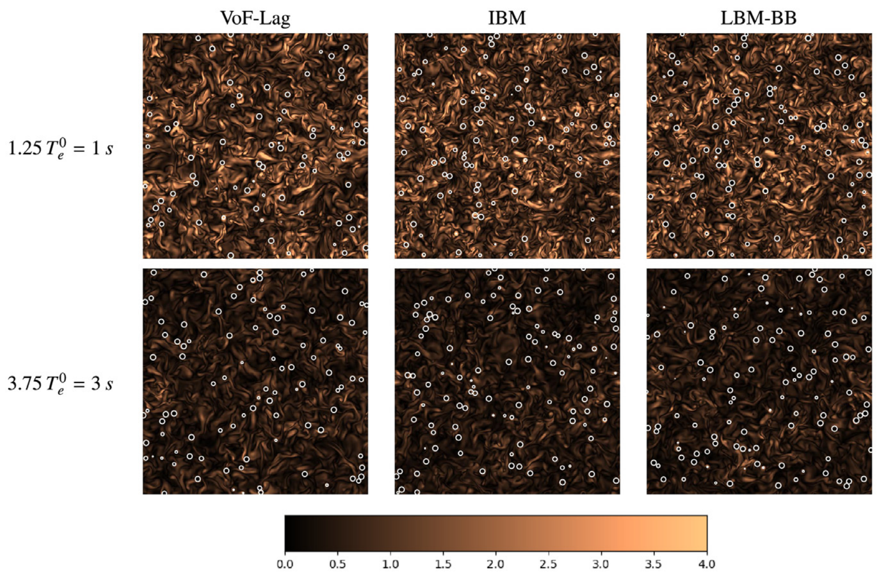


Fig. 7. Vorticity field for the $x - y$ plane and $z = 0$ obtained with each method for the 512^3 case. The vorticity magnitude is divided by the averaged value for $t = 1.25\,T_e^0 = 1\,s$.

cost, the finer mesh simulation (1024^3) has not been considered with the VoF-Lag method to check improvement of the solution at $t = 1.25\,T_e^0$.

4. Comparisons of particle-laden flow results

4.1. Carrier flow analysis

In Fig. 7 the vorticity is shown for each approach at two given times for the 512^3 resolution. It is clear that not only the vorticity levels decrease but also the structures become larger with time. If we compare carefully the turbulent structures for $t = 1.25\,T_e^0$ (top panels of Fig. 7) they remain similar among the different codes. Nevertheless, the results from different codes diverge for the later time presented in the figure (bottom panels). This quantitative code-to-code comparison is completed in this paragraph by analyzing the carrier fluid statistics.

It has been shown in many finite-size particle studies that the fluid kinetic energy decreases faster when particles are present; see for example [8,9,51]. In the present simulations this phenomenon is confirmed. Fig. 8 shows the evolution of the particle-laden case. The spectral solution for single-phase flow is given for comparison. On comparing Figs. 5 and 8, it can be observed that the fluid kinetic energy decreases faster in the two-phase flow case. In the case of single-phase flow, the fluid kinetic energy obtained with the VoF-Lag, IBM and LBM-BB methods follows the reference solution (spectral code) when in the two-phase flow the kinetic energy of these methods is below the spectral code solution. The LBM-IBM solution also decreases faster than its equivalent single-phase simulation. Turbulent modulation is weaker as compared to the cases cited above; in these papers [8,9,51], the solid volume fraction is 10%, whereas in the current study it is chosen to be 3%. It is to be noted that the 512^3 and 1024^3 cases have the same volume fraction. It can be seen in Fig. 8 that for IBM and LBM-BB methods the turbulence modulation is equivalent for both cases. It could be concluded that the main factor for the energy dissipation is not the ratio of particle diameter to Kolmogorov length ratio but the solid volume fraction. In the extensive study Lucci et al. [8] a similar conclusion is drawn. The volume fraction is highlighted as an important factor for the turbulence modulation. In [8] the effect of the diameter is also pointed out. The percentage of reduction of the turbulent kinetic energy decreases when the

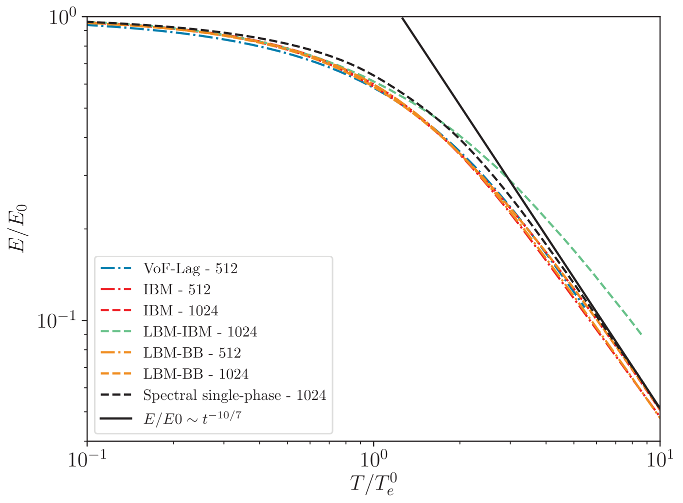
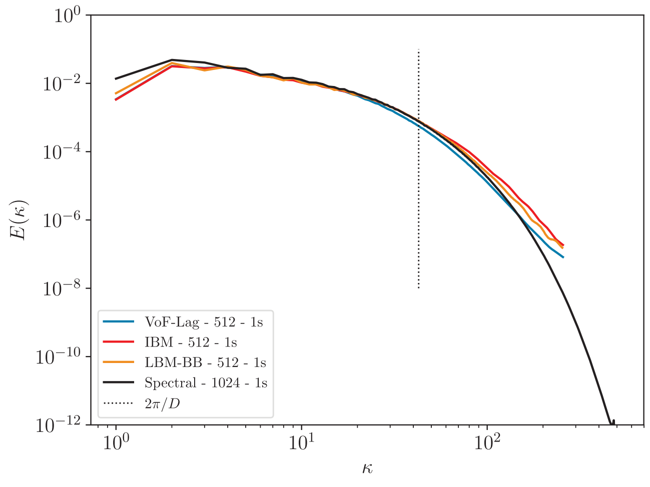


Fig. 8. Decaying fluid kinetic energy of two-phase flow. E_0 and T_0^e denote the values of kinetic energy and eddy turnover time at $T = 0$, respectively.

diameter increases. The present results are in contradiction with those presented in [8] because for the 512^3 and 1024^3 cases similar reduction is observed even though the diameter is different. In order to clarify this discrepancy, it is important to highlight that the diameter increases at constant Eulerian mesh resolution in [8]. In their study $D/\Delta x$ increases with D from 8 to 17. Here, we keep $D/\Delta x = 12$ constant and we double the mesh resolution. This results point out that resolution of particles could have an important impact on the turbulent kinetic energy modulation. This is a numerical effect since physically the particle size effect should depend on D/η rather than $D/\Delta x$. The only way to confirm the effect of particle diameter on turbulence modulation is to do a mesh convergence study. With the increase of the computer resources this kind of study will be affordable in the near future.

The analysis of the turbulent spectra, Fig. 9, provides additional information on the turbulence modulation. The discrepancies among codes on single-phase spectra have been discussed in Section 3.2. Here, we focus on the turbulence modulation by particles. In all the codes the spectra increase for wave numbers larger than the wave number corresponding to the particles' diameter,



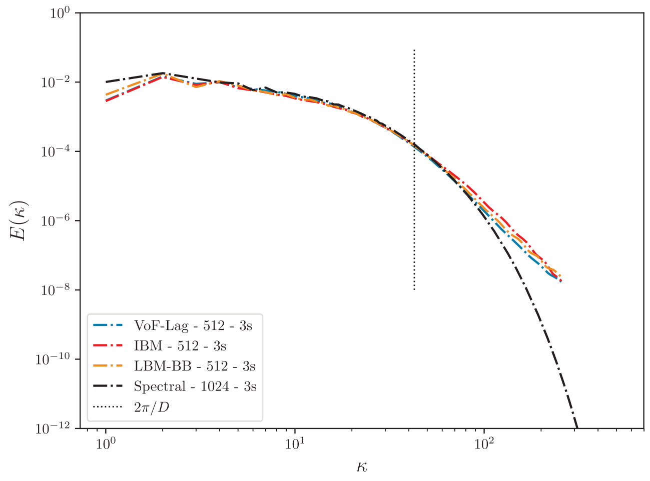


Fig. 9. Spectra for two-phase case for two given times. Top: $t = 1.25T_e^0$ ($1\,s$); Bottom: $t = 3.75T_e^0$ ($3\,s$). The single-phase spectral solution is given for reference. The vertical line corresponds to particle diameter.

$\kappa = 2\pi/D$. The energy increase level is of the same order of magnitude for all the methods used.

It is important to recall that the spectra are computed for the entire domain, including the volume occupied by the particles. For larger volume fractions some oscillations can appear on the spectra [9–11]. That is because of the computation of the spectra inside the solid region, as explained in [8]. Here, these oscillations are clearly visible for the IBM and LBM-BB approaches at $t = 1.25\,T_e^0$.

4.2. Dispersed phase statistics

Many classical results on particle-laden flow are of particle statistics. These results are shown here for the present methods.

First of all, the particle positions given by different codes are compared in Fig. 7. The particle positions remain similar between different codes at $t = 1.25\,T_e^0$ but are different at $t = 3.75\,T_e^0$. Nevertheless, even at $t = 1.25\,T_e^0$ the position of the VoF-Lag particles is significantly different, compared to the positions provided by LBM and IBM codes. This discrepancy is an effect of the initial condition that is treated differently in the VoF-lag code. This point will be discussed later in this section.

At $t = 1.25\,T_e^0$ the probability density function (p.d.f.) of the particle velocity reaches the classical Gaussian distribution, see Fig. 10. No significant discrepancy is observed among different codes. This figure allows us to consider that the number of particles for the coarse case $N_p = 4450$ is large enough to converge our statistics.

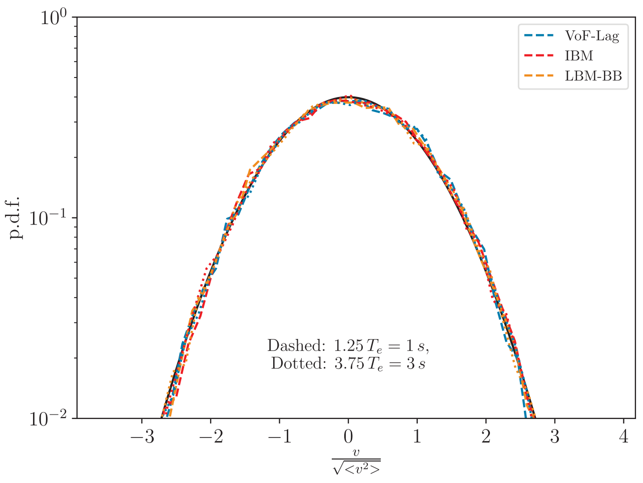


Fig. 10. P.D.F. of article velocity averaged over 3 velocity components.

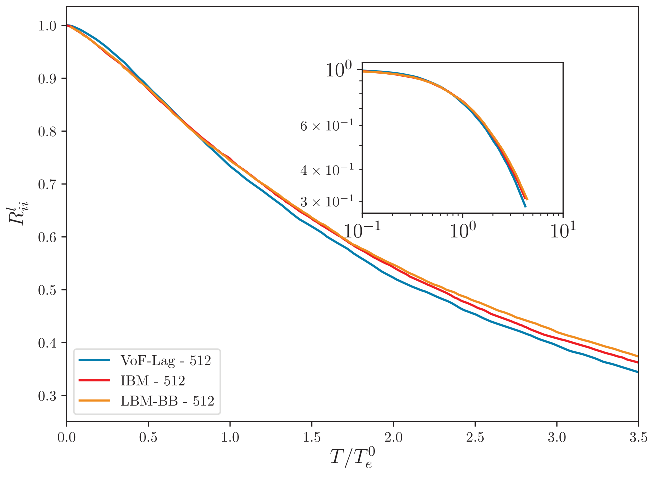


Fig. 11. Lagrangian velocity autocorrelation autocorrelation function starting at $t_0 = 1.25\,T_e^0 = 1\,s$.

In order to study the particle dispersion the velocity autocorrelation function given by,

$$R_{ii}^l(t) = \frac{\sum_{n=0}^{N_p} \mathbf{U}_i(t_0)\cdot\mathbf{U}_i(t_0+t)}{\sqrt{\sum_{n=0}^{N_p} \mathbf{U}_i(t_0)\cdot\mathbf{U}_i(t_0)}\sqrt{\sum_{n=0}^{N_p} \mathbf{U}_i(t_0+t)\cdot\mathbf{U}_i(t_0+t)}} \tag{11}$$

is analyzed. Fig. 11 shows this function for the different codes. Two major differences can be highlighted. First of all, the autocorrelation function with VoF-Lag is larger than the two other ones at early times. This difference is an effect of the initial slope of this function observed with the VoF-Lag method that is smaller compared to the other codes. This result is common for inertial particles and means that the particles are strongly correlated for small times. The second difference is that the R_{ii}^l function is smaller for larger times for the VoF-Lag simulations and larger for the LBM-BB simulations. In all the cases, the slope of the autocorrelation function recovers the same slope for larger times, see inset plot in Fig. 11.

In order to go further on the analysis of the dispersion a truncated particle autocorrelation time T^l is computed by

$$T^l = \int_0^3 R_{ii}^l(t)dt. \tag{12}$$

It cannot be directly called the *autocorrelation time* for two reasons: the integration is not done until infinity and we consider

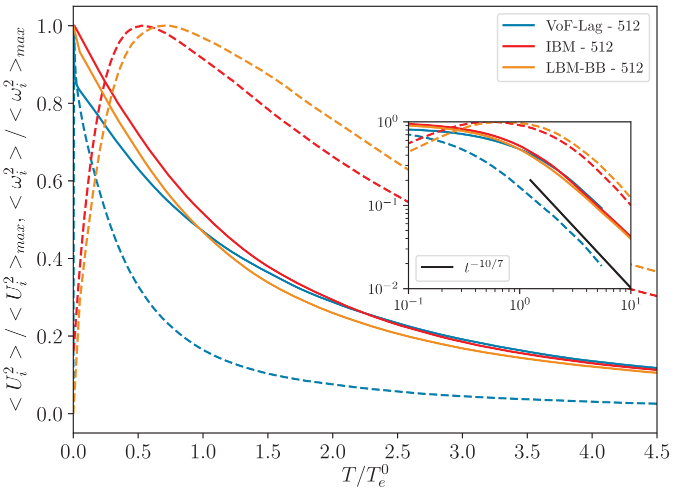


Fig. 12. Particles translational kinetic energy $< U_i^2 >$ (solid line) and angular kinetic energy $< \omega_i^2 >$ (dashed line).

a decaying turbulence. The three methods provides similar T^l: $2.23\,T_e^0$ for VoF-Lag and $2.26\,T_e^0$ for IBM and LBM-BB. The differences obtained here on the dispersion of particles are relatively small.

Based on these results, we can conclude that the dispersion is not affected by the different methods used to take into account the finite-size particles.

In order to continue the analysis of the particle statistics the particle kinetic energy is now analyzed.

The translational and angular kinetic energy ($< \mathbf{U}_i^2 >= \frac{\sum_{n=1}^{N_p} \mathbf{U}_i \cdot \mathbf{U}_i}{3N_p}$ and $< \boldsymbol{\omega}_i^2 >= \frac{\sum_{n=1}^{N_p} \boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i}{3N_p}$ respectively) is given in Fig. 12. As the turbulence is not sustained the particle kinetic energy decreases exponentially. The exponential factor of the particle decaying energy is near the $-10/7$ given for the turbulent decaying energy (see the inset plot). This global behavior is reproduced by all the methods.

The main differences observed come from the initial condition. The initial translational kinetic energy drops about 10% of the initial value for the VoF-Lag method in the first time steps. For this method, the Newton–Euler Eq. (5) are not solved explicitly. The Navier–Stokes equations ensure this fluid-solid interaction. For this reason, as soon as the initial carrier fluid region is replaced by a solid region, the equivalent-fluid inside the particle is *solidified*. That affects all the region around through the Augmented Lagrangian iteration. The velocities are then reduced inside the particles, thus the translational energy of the particles is affected. For the LBM-BB a reduction of 5% of the initial translational kinetic energy is also seen for the first iterations. This drop can be due to fact that the particles have zero angular velocity in the beginning, so there are discontinuities on the fluid-particle interfaces that induce large dissipation to the translational particle kinetic energy. The treatment of initial condition is different among different methods. The evidence is that given zero particle rotation at $t = 0$, at the very short time $t = 0.02\,s = 0.025\,T_e^0$ the angular kinetic energy recovered by the IBM method is 12 times larger than the one obtained by the LBM-BB method. The hydrodynamic torque is large for the IBM method for small times. The IBM forcing scheme achieves a more smooth velocity on the interfaces at the first iteration, thus the IBM shows no initial drop of translational kinetic energy. This could explain the discrepancies between IBM and LBM-BB.

If we compare the average velocity of particles, $< |\mathbf{U}_i| >= \frac{\sum_{n=1}^{N_p} \sqrt{\mathbf{U}_i \cdot \mathbf{U}_i}}{N_p}$, at $1.25\,T_e^0$ and $3.75\,T_e^0$, the mean velocity remains the same for all the codes, see Table 3. Indeed, we can conclude that even this initial effect does not modify the final translational kinetic energy.

The solidification has a strong effect on the angular kinetic energy. Contrary to the other methods, in the VoF-Lag method the particles recover angular velocity directly. This angular velocity is obtained inside the particle after the *solidification* and could be seen as an integration of the angular velocity inside the particle region. The angular velocity is at its maximum at the initial time step. This angular kinetic energy decreases fast at the beginning of the simulation reaching the exponential decay observed for the large times. The IBM and LBM-BB methods do not have this *solidification* effect. The angular kinetic energy starts from zero since the particles are initialized without rotation. Because of the moment of inertia, the particles take $0.53\,T_e^0$ and $0.72\,T_e^0$ to reach their maximum for IBM and LBM-BB respectively. The angular kinetic energy contained in rotation is 10% larger for the IBM method than for the LBM-BB method. This difference is also an effect of the initialization. Indeed, the IBM particles have a stronger angular acceleration during the first iterations. If we compare the angular kinetic energy without dividing by its maximum we observe than it is larger for the IBM than for LBM-BB until $t = 1.25\,T_e^0$. The averaged angular velocity, $< |\boldsymbol{\omega}_i| >= \frac{\sum_{n=1}^{N_p} \sqrt{\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_i}}{N_p}$, at $1.25\,T_e^0$ and $3.75\,T_e^0$ are provided in Table 3. Nevertheless, for all the methods, we reach the same exponential decay for the angular kinetic energy. That confirms the assumption that discrepancies on this quantity are the result of the initial condition treatment.

To go into more detail, we will now analyze the local slip velocity around the particles.

4.3. Local slip velocity

In order to compare the behavior of each code close to the particles, the average slip velocity is computed. This kind of analysis has been presented in previous papers [11,52,53]. The algorithm used by the different authors makes use of different ways to average the velocity around the particles. The main difference is how the particle frame of reference is considered for each particle. Here a different algorithm is used. The algorithm is described below.

- Loop through particles:
  - interpolate fluid velocity to a spherical surface with radius $R_{av} = 4R_p$, and determine the intrinsic velocity of the pth particle: $\mathbf{U}_p^f = \sum_l \Phi_l \mathbf{U}_l^f / \sum_l \Phi_l$, where l denotes a Lagrangian grid in the spherical surface, and Φ a phase-indicator function;
  - compute the particle-to-fluid (apparent) slip velocity $\mathbf{U}_p^s = \mathbf{U}_p^f - \mathbf{U}^p$;
  - define a spherical averaging volume, with axis of symmetry aligned with $\mathbf{U}_p^s$, and interpolate the fluid velocity to this grid, obtaining $\mathbf{U}_{p,r,\theta,\phi}^f$, with indexes $(r,\ \theta,\ \phi)$ denoting the radial, polar and azimuthal directions, respectively;
- compute intrinsic average of fluid slip velocity in the spherical volumes $\mathbf{U}^s(r,\theta) = \sum_{p,\phi} \Phi_{p,r,\theta,\phi}(\mathbf{U}_{p,r,\theta,\phi}^f - \mathbf{U}^p) / \sum_{p,\phi} \Phi_{p,r,\theta,\phi}$. Note that the sum is performed over all the particles and over the (statistically homogeneous) azimuthal direction.

Fig. 13 provides the averaged slip velocity, $\mathbf{U}^s(r,\ \theta)$, for $t = 1\,s$. This slip velocity is divided by the averaged particle velocity $< |\mathbf{U}_i| >$, given in Table 3. Even though the slip velocities are relatively small, it can be seen that for all the codes there is no fore-aft symmetry as in Stokes flow around a sphere. This asymmetry is even present for tracers [52] and is an effect of the conditional averaging of the flow in a moving frame of reference.

Table 3
Dimensionless particle averaged statistics.

Method	Case	Time	$\sqrt{<\mathbf{U}_i^2>}/u^0_{r.m.s.}$	$<\lvert\mathbf{U}_i\rvert>/u^0_{r.m.s.}$	$\sqrt{<\boldsymbol{\omega}_i^2>}D/u^0_{r.m.s.}$	$<\lvert\boldsymbol{\omega}_i\rvert>D/u^0_{r.m.s.}$
VoF-Lag	512	$1.25\,T_e^0$	0.64	1.03	0.29	0.45
IBM	512	$1.25\,T_e^0$	0.64	1.05	0.20	0.31
LBM-BB	512	$1.25\,T_e^0$	0.63	1.02	0.20	0.30
VoF-Lag	512	$3.75\,T_e^0$	0.38	0.61	0.15	0.23
IBM	512	$3.75\,T_e^0$	0.36	0.60	0.13	0.20
LBM-BB	512	$3.75\,T_e^0$	0.36	0.58	0.14	0.21

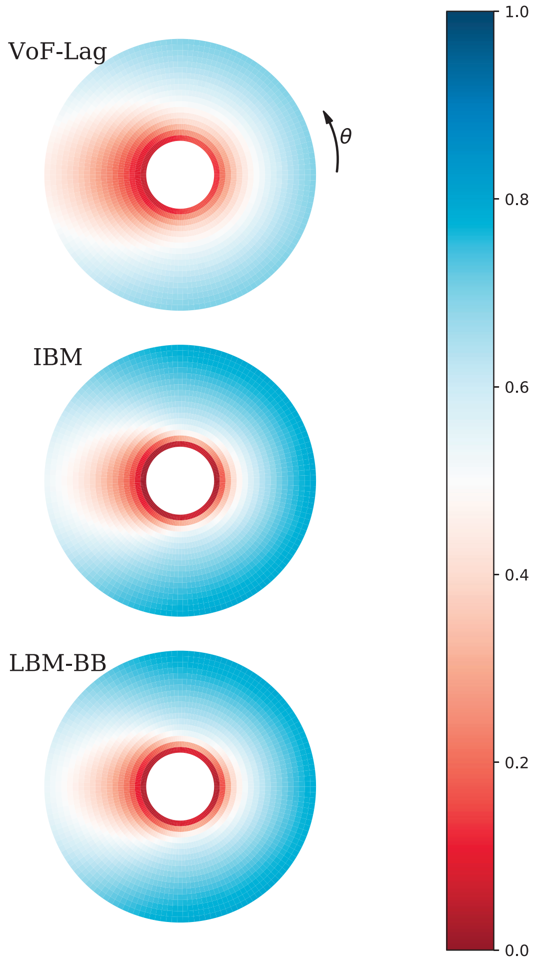


Fig. 13. Dimensionless conditionally-averaged fluid velocity for $t = 1.25T_e^0$ (1s).

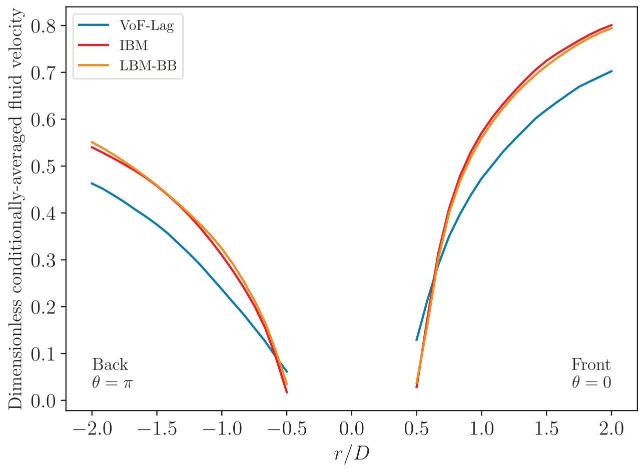


Fig. 14. Dimensionless conditionally-averaged fluid velocity for $t = 1.25T_e^0$ (1s) among the axis.

The differences between the codes are more evident in Fig. 14 where the slip velocity is reported on the axial direction, $\theta = 0$ and $\theta = \pi$. The dimensionless slip velocity is smaller than the unity for $r = 2D$. That means that the particle velocities are correlated to the surrounding fluid. That could be also linked to the two-point correlation for turbulent cases.

For the VoF-Lag method, the slip velocity for $r = 2D$ is smaller than for the other codes that could be seen as a stronger correlation between the particles and the fluid.

The averaging approach does not ensure that the slip velocity is zero at the particle's surface for the VoF-Lag method. As soon as we use an interpolation of the fluid to a spherical shell we take information inside the particle when r is small. This difference is purely an effect of the post-treatment that has been adapted to the IBM approach. Indeed, in [11] a different averaging approach is proposed where only external points are encountered. The velocity is then closer to zero.

5. Computational performance

The Vof-Lag, IBM and LBM-IBM simulations of the present work have been made on the Supercomputer EOS of the Toulouse University Computing Center. This Supercomputer is a Bullx Cluster made of 612 compute nodes interconnected thanks to Infiniband technology (FDR 56Gb/s) in a full fat-tree topology. Each nodes is made of two 10-cores socket intelÂ®Ivybridge (2680v2) with 64 Gb of Shared memory (namely a ratio of 3.2 GB per core). With 12240 cores, EOS reaches #183 rank at TOP500 in June 2014 with 93% of efficiency at the High Performance Linpack (i.e: 255 TF Rmax 274 TF Rpeak) [54].

We have taken the opportunity of the installation of EOS system, and the pre-production operation associated with, to allow the system to be used in a more dedicated way. In operation, a system with a large amount of users, may not be properly suited for benchmarking. Though this is not required in terms of application performance, at least it can be in the amount of resources available and/or waiting time to use these resources.

More precisely, for this benchmarking process, up to 128 nodes (2560 physical cores) had been dedicated for each run with a maximum of elapsed time of 3 days, again per run. We would like to point out that computing resources have been granted for each run in an exclusive manner. That is important to minimize possible interactions due to others jobs running on the system. Moreover the interconnection topology, so-called full fat-tree, has the property to minimize the worst latency and keep the maximum bandwidth for any given set of compute nodes. Hence locality effect should not play a significant role in the application performance (i.e. the performance should remain the same, irrespective of in which part of the system the codes run). Eventually, even if I/O is a very big issue in nowadays high-performance computing, it was not relevant to the present work. So it had been reduced to a minimum and not

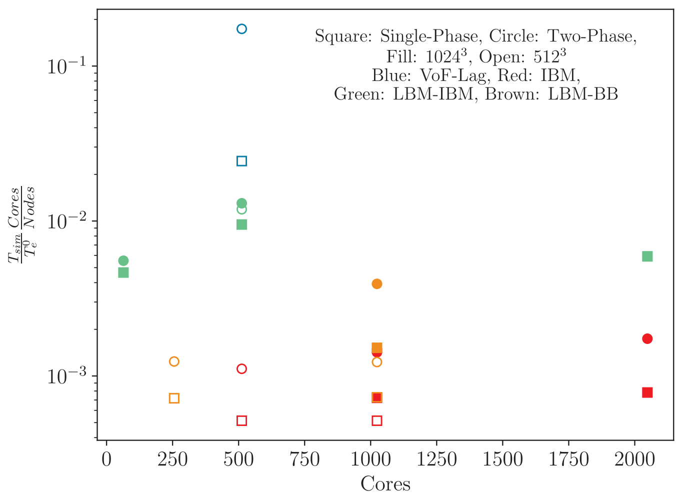


Fig. 15. Total consumption on EOS supercomputer for the different cases. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

taken into account in performance analysis. As a whole, in a period of three months, around 2 millions of cpu hour on Supercomputer EOS had been consumed.

During this benchmark the researchers and the CALMIP administrators worked together in order to enhance the implementation of the codes on this machine. In particular, for this benchmark, the LBM-IBM method was also parallelized. Some experience was obtained thanks to this collaboration. Some test were done in order to ensure that the distribution of the cores on the cluster, the choice of the compiler and the compiler options were the best choice for each code.

The LBM-BB team joined the consortium later and did not run on CALMIP computer. The University of Delaware team used the National Center for Atmospheric Research's (NCAR) supercomputer Yellowstone equipped with 2.6-GHz Intel Xeon E5-2670 (Sandy Bridge) processors [45]. This computer has similar performances as the EOS supercomputer. For this reason we decided to include the performance of this code for comparison.

Fig. 15 gives the CPU time, T_{sim}, needed to simulate a physical fluid initial turnover time T_e^0 for each code and simulations. In order to provide both weak and strong scaling this time is made dimensionless with the number of CPU cores and mesh nodes.

The VoF-Lag simulations were only run on the 512 case and were too expensive to reach the other codes on the 1024 test case. As we can see in Fig. 15 the CPU time was too high compared to other codes. In this case the single-phase case takes more than 50 thousand CPU hours while the two-phase flow more than 300 thousand CPU hours per T_e. The high computational cost for this method could be explained by different reasons. First of all, the semi-implicit iterative solver used to solve the mass and momentum equations is more expensive than the time splitting used in classical Navier–Stokes solvers or the LBM methods. The advantage of this solver is that we can utilize larger time steps for two-phase flows and we are not limited by the viscous CFL number. Nevertheless, in this case we do not take profit of this solver because the turbulent flow requires a small advective time step. In addition, when the particle-laden case is considered, the CPU time is one order of magnitude higher. This increase is explained by two factors. First of all, for stability reasons the time step was divided by a factor of two (from $0.0125\,T_e^0$ to $0.00625\,T_e^0$) increasing the computational time. The time spent on the Navier–Stokes solver, which is the part in common with single-phase simulation, is multiplied by $2.3\sim2$. The second reason is that the update of the physical characteristics takes 67% of the simulation. That includes the transport of the particles and the update of solid volume fraction, density and viscosity fields. Later studies explain that the algorithm used to update the solid volume fraction was the weakest link. After the simulations presented here this algorithm was improved by limiting the search of solid grid cells for particles' neighbors and reducing the number of points used to compute the solid fraction in intermediate grid cells. These modifications reduce the CPU time of this part of the code by 60%. In the VoF-Lag implementation the time spent to treat collisions takes 3%.

The IBM and LBM-IBM methods provide a better implementation compared to VoF-Lag method. The time of the particle-laden case is one order of magnitude larger than VoF-Lag for the 512 case: 26 thousand CPU hours per turnover time. Even if the parallel implementation was developed for the benchmark purposes it shows a remarkable speed-up. Indeed, in Fig. 15, if we compare the green filled squares we can see that the CPU time remain in the same order of magnitude and is even reduced for the simulation with 2024 CPU cores. That shows that the LBM-IBM implementation provides an adequate weak scaling factor. In the same figure, if we compare the filled and open circles at 512 CPU cores we can observe that they are similar, showing that the strong scaling is also respected. This result confirms the idea that LBM-IBM Navier-Stokes solvers could be easily parallelized and provide a good scaling. The particle-laden case increases the CPU time by 19% with 64 CPU cores and 37% with 512 cores. This overhead is slightly large compared to other LBM methods. Indeed, [9] found a computational overhead between 20% and 26% for a test case with more particles and volume fraction than the present one.

The TU Delft IBM implementation provides the best performances compared with the other two codes. The CPU time is one order of magnitude smaller than the LBM-IBM approach and two orders of magnitude smaller than the VoF-Lag method even for the single-phase flow. In Fig. 15, one can also verify that the strong and weak scaling of this implementation are really good for single and two-phase case: for the strong scaling compare the same red symbols and for weak scaling compare fill with open symbols.

Nevertheless, the particle-laden cases are much more expensive than their equivalent in single-phase. The CPU time increases, for the best case, 87% compared to same case in single-phase flow. For the worst case, the increasing of CPU consumption reach 188%. That is explained by the time taken by the IBM algorithms of interpolation and spreading that takes from 39% to 55% of the CPU consumption for the particle-laden flows simulations. In these simulations 10% of the CPU were spent in short-range interactions (collisions), integration of the Newton–Euler equations, Eq. (5), and re-initialization of particle-related arrays needed for the parallel implementation. TU Delft group has continued to improve their parallel implementation, as described in the last paragraph of Section 2.4.2 and in more detail in Section 2.5 of [39].

The time data from LBM-BB code have been added even though the processor's used was not exactly the same. We can see that the performances are similar to these of the IBM approach. The weak scaling is well recovered for the 512^3 case (compare no-fill squares in Fig. 15). Nevertheless, the strong scaling is not well recovered. The computational cost of the particles case seems coherent with other codes. The large overhead for the particle-laden case in 1024^3 is mainly because at the time when the simulation was run, the particles information (position, velocity, angular velocity, forces, ...) were shared by all the processors. Since 1024^3 case has 8 times more particles compared to 512^3, this implementation slows down the simulation. Some improvements of the LBM implementation for finite-size particles was proposed recently by the developers of the LBM-BB method [45].

The computational performance study shows that the IBM implementation is much better than the other implementation, see

Fig. 15. Nevertheless, these results have to be taken in perspective and should be considered as a snapshot. The evolution of each code and the evolution of supercomputers and compilers could change this picture in a short term. In addition, the physical parameters, as the solid volume fraction, the number of particles or the Reynolds numbers, could modify the balance between codes.

6. Discussion and conclusion

Many recent studies based on RPS approaches are used to treat particle-laden flows. The present paper provides an extensive comparison of different RPS approaches for a turbulent carrier flow case. Since they yield qualitatively similar physical results, this comparison adds confidence in the approaches.

The turbulent carrier flow is modulated by the particles. The energy decays faster in the particle-laden flow and the energy spectra increase for large wave numbers. Here an open question remains when we study the effect of the diameter on the turbulent modulation. Indeed, IBM and LBM-BB provide the same result quantitatively: the diameter has no major effect on the modulation when the volume fraction remains constant. This result is different from the conclusion provided by Lucci et al. [8] where the diameter has an effect on the modulation. A future study could provide an answer to this discrepancy.

The statistics of the dispersed phase show classical results. The p.d.f. of particle velocity follows the Gaussian distribution. The autocorrelation function is slightly different for different codes. Nevertheless, these differences are minor. Finally, the particle kinetic energy follows the trend of the decaying turbulence. The differences between the codes are sometimes significant but they are mostly related to the different initial treatments of the interior volumes of the particles. The non-physical adjustment of the solution at the first time steps is the main reason for the discrepancies.

Averaging the fluid velocity around the particles provides information about the slip velocity. The results obtained are similar to those proposed by previous authors. The main differences are near the solid-liquid interface where the VoF-Lag method does not tend to zero. That is because the averaging method is not adapted to the VoF-Lag method: it interpolates with points inside the particle. For future works, it is important to ensure the consistency between the averaging post-treatment approach and the numerical approach. Here, the same post-treatment algorithm is used for all the codes in order to have equivalent data.

The physical study was completed by an analysis of the computational performances. The methods implemented were completely different. When the simulations were performed the IBM method was the fastest method, followed by the LBM-IBM and then the Vof-Lag method. The LBM-BB approach has not run on the same supercomputer, but shows very good computational performances. One of the main results here was that the Augmented Lagrangian Method was not adapted to this kind of simulations. For the turbulence simulation the time step Δt is similar for semi-implicit or explicit time integration scheme. The semi-implicit time step used by the VoF-Lag method is more expensive than an explicit scheme.

Thanks to the benchmark each group has continued its developments and many improvements have been done after the simulations. The results obtained from the benchmark were very useful but should be considered as a snapshot done at a given time.

The present paper provides an extensive comparison for a given turbulent flow. The main purpose was to point out the numerical and physical differences between the approaches. Unfortunately, the comparisons were limited to the benchmark participants. For future comparisons, the initial condition and the algorithms done for the post-treatments could be shared upon request, by contacting the corresponding author.

Acknowledgements

This work was granted access to the HPC resources of CALMIP and the National Center for Atmospheric Researchs (NCAR) supercomputing centers. P. Costa acknowledges the funding from the Portuguese Foundation for Science and Technology under grant no. SFRH/BD/85501/2012. L.-P. Wang acknowledges the funding from the U.S. National Science Foundation (NSF) under grants CBET-1706130.

References

[1] Ishii M. Thermo-fluid dynamic theory of two-phase flow, 22 Direction des études et recherches délectricité de France. Paris: Eyrolles; 1975.
[2] Crowe CT. Review—numerical models for dilute gas-particle flows. J Fluids Eng 1982;104(3):297. doi:10.1115/1.3241835.
[3] Crowe CT. The state-of-the-art in the development of numerical models for dispersed phase flows. In: Proceedings of international conference on multiphase flows, Tsukuba, 3; 1991. p. 49–60.
[4] Crowe CT, Schwarzkopf JD, Sommerfeld M, Yutaka T. Multiphase flows with droplets and particles. 2nd ed. CRC Press; 2012. ISBN 978-1-4398-4050-4.
[5] Maxey M. Simulation methods for particulate flows and concentrated suspensions. Annu Rev Fluid Mech 2017;49(1):171–93. doi:10.1146/annurev-fluid-122414-034408.
[6] Johnson A, Tezduyar T. 3d simulation of fluid-particle interactions with the number of particles reaching 100. Comput Methods Appl Mech Eng 1997;145(3-4):301–21.
[7] ten Cate A, Derksen JJ, Portela LM, Van Den Akker HEA. Fully resolved simulations of colliding monodisperse spheres in forced isotropic turbulence. J Fluid Mech 2004;519:233–71. doi:10.1017/S0022112004001326.
[8] Lucci F, Ferrante A, Elghobashi S. Modulation of isotropic turbulence by particles of Taylor length-scale size. J Fluid Mech 2010;650:5–55.
[9] Gao H, Li H, Wang L-P. Lattice Boltzmann simulation of turbulent flow laden with finite-size particles. Comput Math Appl 2013;65(2):194–210. doi:10.1016/j.camwa.2011.06.028.
[10] Wang L-P, Ayala O, Gao H, Andersen C, Mathews KL. Study of forced turbulence and its modulation by finite-size solid particles using the lattice Boltzmann approach. Comput Math Appl 2014;67(2):363–80. doi:10.1016/j.camwa.2013.04.001.
[11] Brändle de Motta JC, Estivalezes J-L, Climent E, Vincent S. Local dissipation properties and collision dynamics in a sustained homogeneous turbulent suspension composed of finite size particles. Int J Multiphase Flow 2016;85:369–79. doi:10.1016/j.ijmultiphaseflow.2016.07.003.
[12] Picano F, Breugem W-P, Brandt L. Turbulent channel flow of dense suspensions of neutrally buoyant spheres. J Fluid Mech 2015;764:463–87.
[13] Costa P, Picano F, Brandt L, Breugem W-P. Universal scaling laws for dense particle suspensions in turbulent wall-bounded flows. Phys Rev Lett 2016;117:134501. doi:10.1103/PhysRevLett.117.134501.
[14] Uhlmann M. Interface-resolved direct numerical simulation of vertical particulate channel flow in the turbulent regime. Phys Fluids 2008;20(5). 053305–27
[15] Özel A, Brändle de Motta JC, Abbas M, Fede P, Masbernat O, Vincent S, Estivalezes J-L, Simonin O. Particle resolved direct numerical simulation of a liquid-solid fluidized bed: comparison with experimental data. Int J Multiphase Flow 2017;89:228–40. doi:10.1016/j.ijmultiphaseflow.2016.10.013.
[16] Derksen JJ. Simulations of granular bed erosion due to laminar shear flow near the critical shields number. Phys Fluids 2011;23(11):113303. doi:10.1063/1.3660258.
[17] Kidanemariam AG, Uhlmann M. Formation of sediment patterns in channel flow: minimal unstable systems and their temporal evolution. J Fluid Mech 2017;818:716–43.
[18] Marchioli C, Soldati A, Kuerten J, Arcen B, Tanire A, Goldensoph G, Squires K, Cargnelutti M, Portela L. Statistics of particle dispersion in direct numerical simulations of wall-bounded turbulence: Results of an international collaborative benchmark test. Int J Multiphase Flow 2008;34(9):879–93. doi:10.1016/j.ijmultiphaseflow.2008.01.009.
[19] Wang L-P, Peng C, Guo Z, Yu Z. Flow modulation by finite-size neutrally buoyant particles in a turbulent channel flow. J Fluids Eng 2016;138(4):041306. doi:10.1115/1.4031691.
[20] Wang L-P, Peng C, Guo Z, Yu Z. Lattice Boltzmann simulation of particle-laden turbulent channel flow. Comput Fluids 2016;124:226–36. doi:10.1016/j.compfluid.2015.07.008.
[21] Vincent S, Brändle de Motta JC, Sarthou A, Estivalezes J-L, Simonin O, Climent E. A lagrangian VOF tensorial penalty method for the DNS of resolved particle-laden flows. J Comput Phys 2014;256:582–614. doi:10.1016/j.jcp.2013.08.023.
[22] Breugem W-P. A second-order accurate immersed boundary method for fully resolved simulations of particle-laden flows. J Comput Phys 2012;231(13):4469–98. doi:10.1016/j.jcp.2012.02.026.
[23] Hu H, Joseph D, Crochet M. Direct simulation of fluid particle motions. Theor Comput Fluid Dyn 1992;3(5):285–306.
[24] Burton TM, Eaton JK. Fully resolved simulations of particle-turbulence interaction. J Fluid Mech 2005;545:67–111. doi:10.1017/S0022112005006889.

[25] Vreman AW. Particle-resolved direct numerical simulation of homogeneous isotropic turbulence modified by small fixed spheres. J Fluid Mech 2016;796:40–85. doi:10.1017/jfm.2016.228.

[26] Vreman A. A staggered overset grid method for resolved simulation of incompressible flow around moving spheres. J Comput Phys 2017;333:269–96. doi:10.1016/j.jcp.2016.12.027.

[27] Prosperetti A, Ogũz H. Physalis: a new o(n) method for the numerical simulation of disperse systems: Potential flow of spheres. J Comput Phys 2001;167(1):196–216. doi:10.1006/jcph.2000.6667.

[28] Takagi S, Ogũz H, Zhang Z, Prosperetti A. PHYSALIS: a new method for particle simulation: Part II: two-dimensional Navier–Stokes flow around cylinders. J Comput Phys 2003;187(2):371–90.

[29] Vincent S, Caltagirone J-P. Numerical solving of incompressible Navier–Stokes equations using an original local multigrid refinement. Comptes Rendus de l'Acadmie des Sci Ser IIB Mech Physics- Astron 2000;328:73–80. doi:10.1016/S1287-4620(00)88419-7.

[30] Vincent S, Sarthou A, Caltagirone J-P, Sonilhac F, Février P, Mignot C, Pianet G. Augmented Lagrangian and penalty methods for the simulation of two-phase flows interacting with moving solids. Application to hydroplaning flows interacting with real tire tread patterns. J Comput Phys 2011;230:956–83.

[31] Brändle de Motta JC, Breugem W-P, Gazanion B, Estivalezes J-L, Vincent S, Climent E. Numerical modelling of finite-size particle collisions in a viscous fluid. Phys Fluids 2013;25(8):083302. doi:10.1063/1.4817382.

[32] Roma AM, Peskin CS, Berger MJ. An adaptive version of the immersed boundary method. J Comput Phys 1999;153(2):509–34.

[33] Uhlmann M. An immersed boundary method with direct forcing for the simulation of particulate flows. J Comput Phys 2005;209(2):448–76. doi:10.1016/j.jcp.2005.03.017.

[34] Luo K, Wang Z, Fan J, Cen K. Full-scale solutions to particle-laden flows: multidirect forcing and immersed boundary method. Phys Rev E 2007;76(6):066709.

[35] Costa P, Boersma BJ, Westerweel J, Breugem W-P. Collision model for fully resolved simulations of flows laden with finite-size particles. Phys Rev E 2015;92(5):053012.

[36] Swarztrauber PN. The methods of cyclic reduction, fourier analysis and the FACR algorithm for the discrete solution of Poissons equation on a rectangle. Siam Rev 1977;19(3):490–501.

[37] Li N, Laizet S. 2decomp & fft-a highly scalable 2d decomposition library and fft interface. In: Cray user group 2010 conference; 2010. p. 1–13.

[38] Uhlmann M. Simulation of particulate flows on multi-processor machines with distributed memory. Ciemat; 2004.

[39] Costa P. Interface-resolved simulations of dense turbulent suspension flows. Tu-Delft; 2017. Ph.D. thesis. http://repository.tudelft.nl.

[40] d'Humières D. Multiple–relaxation–time lattice Boltzmann models in three dimensions. Philos Trans R Soc Lond A 2002;360(1792):437–51.

[41] Peng C, Teng Y, Hwang B, Guo Z, Wang L-P. Implementation issues and benchmarking of lattice boltzmann method for moving rigid particle simulations in a viscous flow. Comput Math Appl 2016;72(2):349–74.

[42] Bouzidi M, Firdaouss M, Lallemand P. Momentum transfer of a Boltzmann–lattice fluid with boundaries. Phys Fluids (1994-present) 2001;13(11):3452–9.

[43] Peng Y, Luo L-S. A comparative study of immersed-boundary and interpolated bounce-back methods in LBE. Prog Comput Fluid Dyn Int J 2008;8(1-4):156–67.

[44] Wen B, Zhang C, Tu Y, Wang C, Fang H. Galilean invariant fluid–solid interfacial dynamics in lattice Boltzmann simulations. J Comput Phys 2014;266:161–70.

[45] Geneva N, Peng C, Li X, Wang L-P. A scalable interface-resolved simulation of particle-laden flow using the lattice Boltzmann method. Parallel Comput 2017;67:20–37. doi:10.1016/j.parco.2017.07.005.

[46] Eswaran V, Pope SB. An examination of forcing in direct numerical simulations of turbulence. Comput Fluids 1988;16(3):257–78. doi:10.1016/0045-7930(88)90013-8.

[47] Pope SB. Turbulent flows. Cambridge University Press; 2000. ISBN 0521598869.

[48] Lucci F, Ferrante A, Elghobashi S. Is stokes number an appropriate indicator for turbulence modulation by particles of taylor-length-scale size? Physics of Fluids 2011;23(2):025101. doi:10.1063/1.3553279.

[49] Kolmogorov AN. On the degeneration of isotropic turbulence in an incompressible viscous fluid. In: Doklady Akademii Nauk SSSR, 31; 1941. p. 319–23. Translation by V. M. Tikhomirov in: Selected Works of A. N. Kolmogorov, 1991.

[50] Mei R, Luo L-S, Lallemand P, d'Humières D. Consistent initial conditions for lattice Boltzmann simulations. Comput. Fluids 2006;35(8):855–62. Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science. doi: 10.1016/j.compfluid.2005.08.008.

[51] Zhang Z, Prosperetti A. A second-order method for three-dimensional particle simulation. J Comput Phys 2005;210(1):292–324. doi:10.1016/j.jcp.2005.04.009.

[52] Cisse M, Homann H, Bec J. Slipping motion of large neutrally buoyant particles in turbulence. J Fluid Mech 2013;735. doi:10.1017/jfm.2013.490.

[53] Schneiders L, Meinke M, Schröder W. Direct particlefluid simulation of Kolmogorov-length-scale size particles in decaying isotropic turbulence. J Fluid Mech 2017;819:188–227. doi:10.1017/jfm.2017.171.

[54] Top 500, June 2014. http://www.top500.org.