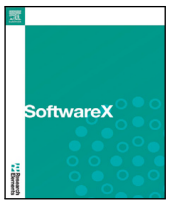
SoftwareX 27 (2024) 101871

Contents lists available at ScienceDirect

SoftwareX

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



OpenHFDIB-DEM: An extension to OpenFOAM for CFD-DEM simulations with arbitrary particle shapes

Ondřej Studeník [a,c], Martin Isoz [a,b,*], Martin Kotouč Šourek [c], Petr Kočí [c]

[a] Institute of Thermomechanics of the Czech Academy of Sciences, Dolejškova 5, Prague 182 00, Czech Republic
[b] University of Chemistry and Technology, Prague, Department of Mathematics, Informatics and Cybernetics, Technická 5, Prague 166 28, Czech Republic
[c] University of Chemistry and Technology, Prague, Department of Chemical Engineering, Technická 5, Prague 166 28, Czech Republic

ARTICLE INFO

Keywords:
Computational fluid dynamics (CFD)
Discrete element method (DEM)
Non-spherical particles
OpenFOAM

ABSTRACT

Fluid flows containing dispersed particles are abundant in both nature and industry. Simulating such flows, especially with high volumetric fractions of the dispersed solid phase, is challenging due to the complexity of solid–fluid and solid–solid interactions. In this paper, we introduce OpenHFDIB-DEM, an open-source software that couples computational fluid dynamics (CFD) with hybrid fictitious domain immersed boundary method (HFDIB) and discrete element method (DEM). The software is developed as an extension of the OpenFOAM library. The main contribution lies in a custom DEM implementation that is fully integrated in the OpenFOAM framework and allows for CFD-DEM simulations of various flows with arbitrarily shaped solids defined by triangulated surfaces in stereolithographic (STL) format.

Code metadata

Code metadata description	Metadata
Current code version	2.6
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00378
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/5797185/tree
Legal Code License	GNU-GPLv3
Code versioning system used	git
Software code languages, tools, and services used	C++, MPI, GNU Make
Compilation requirements, operating environments	OpenFOAM-v8
If available Link to developer documentation/manual	http://docs.isoz.eu/openHFDIB-DEM/2.6/
Support email for questions	openhfdib-dem@it.cas.cz

1. Motivation and significance

Software coupling computer fluid dynamics (CFD) with the discrete element method (DEM) enables high-fidelity simulations of particle-laden flows, a phenomenon abundantly present in both natural and industrial processes. Here, we are particularly interested in situations where the solids are distributed so densely that they encounter both each other and the boundaries of the containing device. Furthermore, the fluid and solid phase are mutually affected. This situation occurs, for example, in wet granulation [1], food processing [2], fluidization [3], or in catalytic material deposition in monolithic reactors [4]. Under the conditions of interest, particle morphology plays a crucial role in the integral flow behavior.

Due to their scientific and industrial potential, coupled CFD-DEM solvers have become a prominent and rapidly evolving research topic in the past decade. Currently, several solvers are available for both open-source and commercial applications. The most well-known open-source software are (i) CFDEM® project [5], which is an OpenFOAM based CFD-DEM solver utilizing LIGGGHTS software [6] for DEM, and (ii) YADE [7], a DEM solver that also can be linked to OpenFOAM. The most prominent commercial ones are (i) ANSYS-Fluent® [8] coupled with ANSYS-Rocky® [9], and (ii) Aspherix® [10] DEM and its extension

DOI of original article: https://doi.org/10.1016/j.powtec.2024.120067.
* Corresponding author at: Institute of Thermomechanics of the Czech Academy of Sciences, Dolejškova 5, Prague 182 00, Czech Republic.
E-mail address: isozm@it.cas.cz (Martin Isoz).
URL: https://www.it.cas.cz/en/d4/l041/ (Martin Isoz).

https://doi.org/10.1016/j.softx.2024.101871
Received 16 July 2024; Received in revised form 23 August 2024; Accepted 29 August 2024
Available online 9 September 2024
2352-7110/© 2024 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

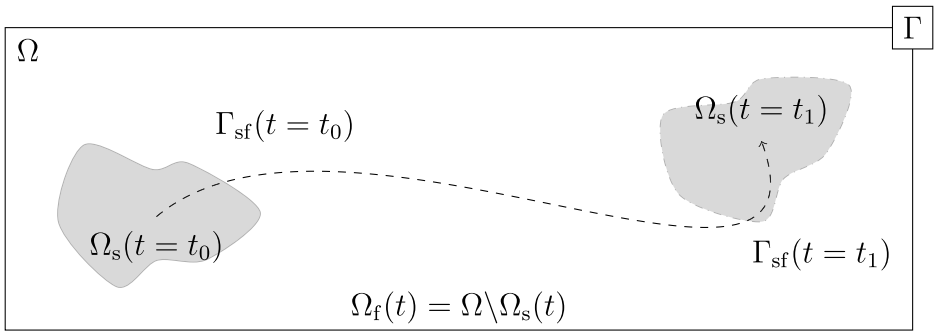


Fig. 1. Solution domain Ω comprising solid (Ω_s) and fluid (Ω_f) phases. An exemplary particle is shown at two different times $t = t_0$ and $t = t_1$. Trajectory of the particle is indicated by a dashed line.

CFDEM® Coupling, which is a commercial successor to the open-source CFDEM® project. To our knowledge, YADE software enables the application of convex polyhedral particles for DEM simulations. ANSYS-Rocky® and Aspherix® enable consideration of surface mesh base solids, however, the inner workings of the particle treatment are not publicly available. Yet, all of the listed solvers allow for similar capabilities, usually based on clusters of spheres or superquadrics, see, e.g., [5,9,10].

The CFD-DEM solver OpenHFDIB-DEM presented here is implemented as an extension to the widely recognized OpenFOAM [11] library. On the CFD side, the solver is based on a variant of the immersed boundary method called hybrid fictitious domain immersed boundary method (HFDIB), originally described in [12,13] and modified as presented in [14]. This method allows for using a static CFD mesh that is not re-arranged according to the moving solid objects. For the discrete element method, we leverage an in-house developed soft-DEM solver designed for arbitrarily shaped solids. The complete OpenHFDIB-DEM framework has the ability to simulate pure DEM applications (e.g. particle pouring) or fully coupled CFD-DEM applications (e.g. fluidized beds, coarse slurry flows) precisely without being limited by the particle morphology. Uniquely among available open-source codes, the particles are defined using stereolithographic (STL) files with no further approximations or assumptions on convexity. Furthermore, the complete CFD-DEM code is monolithic, which simplifies data transfer required by fluid–solid interaction. To demonstrate the computational efficiency and robustness of our presented extension, several OpenHFDIB-DEM library verification tests are presented, including verification benchmarks for precision and large system simulations containing thousands of particles for both DEM and CFD-DEM applications.

2. Model description

In general, we solve the balance equations for momentum and mass in a finite, open, and connected domain $\Omega \subset \mathbb{R}^3$ and mark Γ the boundary of Ω. The computational domain Ω is assumed to contain a continuous fluid phase and a dispersed and moving solid phase. The situation is illustrated in Fig. 1, where Ω_s and Ω_f represent the parts of Ω occupied by the solid and fluid phases, respectively, and Γ_{sf} is the fluid–solid interface.

The overall goal of each CFD-DEM solver is to efficiently combine the Eulerian (CFD) formulation of the balance equations in $\Omega = \Omega_s \cup \Omega_f$ with the Lagrangian (DEM) framework for particle motion, that is, for updating Ω_s. We leverage the OpenFOAM finite volume C++ library [11] for CFD computations. To insert particles into the computational domain, i.e., to perform the $\Omega = \Omega_s \cup \Omega_f$ operation, we build on our previous work with the hybrid fictitious domain immersed boundary method (HFDIB) [14]. The DEM part of the code also stems from the work [14], but was significantly reformulated and will be presented in more detail, with a special focus on solid–solid collision dynamics.

2.1. Computational fluid dynamics

We consider an incompressible flow of a Newtonian fluid laden with a dispersed solid phase undergoing an arbitrary rigid body motion. The flow is solved in the full computational domain $\Omega = \Omega_s \cup \Omega_f$ and is assumed to be governed by the standard incompressible Navier–Stokes equations

$$\begin{aligned} &\mathcal{M}(\boldsymbol{u}) = -\nabla\tilde{p} + \boldsymbol{f}_{\mathrm{ib}} \\ &\nabla\cdot\boldsymbol{u} = 0 \end{aligned} \quad , \quad \mathcal{M}(\boldsymbol{u}) = \frac{\partial \boldsymbol{u}}{\partial t} + \nabla\cdot(\boldsymbol{u}\otimes\boldsymbol{u}) - \nabla\cdot(\nu\nabla\boldsymbol{u}) \tag{1}$$

where $\boldsymbol{u}$ is the fluid velocity, ν kinematic viscosity, and $\tilde{p}$ kinematic pressure. The forcing term $\boldsymbol{f}_{\mathrm{ib}}$ is constructed to generate a fictitious representation of Ω_s inside Ω and to bend the fluid flow around the solid.

For the particular case of the hybrid fictitious domain-immersed boundary method (HFDIB) [12–14], the forcing term is defined as

$$\boldsymbol{f}_{\mathrm{ib}} = \mathrm{ceil}(\lambda)\tilde{\boldsymbol{f}}_{\mathrm{ib}}\,, \quad \tilde{\boldsymbol{f}}_{\mathrm{ib}} = \mathcal{M}(\boldsymbol{u}_{\mathrm{ib}}) + \nabla\tilde{p}, \quad \lambda = \begin{cases} 0 & \text{in} \quad \Omega_f \\ 1 & \text{in} \quad \Omega_s \\ \tilde{\lambda}\in(0,1) & \text{in} \quad \Gamma_{sf} \end{cases} \tag{2}$$

where λ is a color function defining the position of the immersed bodies in the computational domain Ω and $\boldsymbol{u}_{\mathrm{ib}}$ is the velocity of the moving solid phase.

The exact definition of $\mathcal{M}(\boldsymbol{u}_{\mathrm{ib}})$ depends on the approach used to enforce the boundary conditions on Γ_{sf}. After spatial discretization of Ω, Γ_{sf} cannot be assumed to be collocated with the resulting computational mesh, Ω^h. Therefore, to enforce the boundary conditions on Γ_{sf}, its exact position with respect to Ω^h has to be reconstructed and interpolation must be used to connect the two, i.e., to evaluate $\mathcal{M}(\boldsymbol{u}_{\mathrm{ib}})$. For details on the exact procedure applied here, the reader is referred to our previous work [14].

Eqs. (1) are discretized via the finite volume method and solved using a standard segregated approach originating from the PISO-SIMPLE (PIMPLE) family of algorithms. The final solution algorithm is obtained by plugging the DEM part into the CFD code and is described in detail in Algorithm 2 in [14].

2.2. Discrete element method

The movement of individual solids $\mathcal{B}_i$, which form the solid phase $\Omega_s = \bigcup_{i=1}^{N_\mathcal{B}} \mathcal{B}_i$, is described using the discrete element method (DEM). DEM is a finite difference method based on the Lagrangian solution of Newton's second law of motion,

$$m_i\frac{\mathrm{d}^2\boldsymbol{x}_i}{\mathrm{d}t^2} = \boldsymbol{f}_i^{\mathrm{g}} + \boldsymbol{f}_i^{\mathrm{d}} + \boldsymbol{f}_i^{\mathrm{c}}\,, \quad I_i\frac{\mathrm{d}\boldsymbol{\omega}_i}{\mathrm{d}t} = \boldsymbol{t}_i^{\mathrm{d}} + \boldsymbol{t}_i^{\mathrm{c}}\,, \tag{3}$$

where m_i is the mass of $\mathcal{B}_i$ of constant density ρ_i, and $\boldsymbol{x}_i$, $\boldsymbol{\omega}_i$, I_i are its centroid position, angular velocity and the matrix of its inertial moments at time t, respectively. By $\boldsymbol{f}$ and $\boldsymbol{t}$ in (3), we denote the forces and torques acting on $\mathcal{B}_i$, respectively. In general, the presented solver

	translation	rotation
update accelerations	$\bar{\boldsymbol{a}}^{\mathrm{o}} = \frac{1}{m}\left(\boldsymbol{f}^{\mathrm{g}}_{\mathrm{CFD}} + \boldsymbol{f}^{\mathrm{d}}_{\mathrm{CFD}} + \boldsymbol{f}^{\mathrm{c,o}}_{\mathrm{DEM}}\right)$	$\bar{\boldsymbol{\alpha}}^{\mathrm{o}} = \boldsymbol{I}_i^{-1}\left(\boldsymbol{t}^{\mathrm{d}}_{\mathrm{CFD}} + \boldsymbol{t}^{\mathrm{c,o}}_{\mathrm{DEM}}\right)$
update velocities	$\boldsymbol{u}^{*} = \boldsymbol{u}^{\mathrm{o}} + \frac{1}{2}\bar{\boldsymbol{a}}^{\mathrm{o}}\Delta t$	$\boldsymbol{\omega}^{*} = \boldsymbol{\omega}^{\mathrm{o}} + \frac{1}{2}\bar{\boldsymbol{\alpha}}^{\mathrm{o}}_i\Delta t$
		$\boldsymbol{\omega}^{*} \rightsquigarrow \omega^{*}, \boldsymbol{o}^{*}$
translate and rotate	$\Delta\boldsymbol{x} = \boldsymbol{u}^{*}\Delta t$	$\Delta\theta = \omega^{*}\Delta t$
	$\Delta\boldsymbol{x}, \Delta\theta, \boldsymbol{o}^{*} \rightsquigarrow$ new position and orientation	
update contact	get $\boldsymbol{f}^{\mathrm{c,n}}_{\mathrm{DEM}}$	get $\boldsymbol{t}^{\mathrm{c,n}}_{\mathrm{DEM}}$
update accelerations	$\bar{\boldsymbol{a}}^{\mathrm{n}} = \frac{1}{m}\left(\boldsymbol{f}^{\mathrm{g}}_{\mathrm{CFD}} + \boldsymbol{f}^{\mathrm{d}}_{\mathrm{CFD}} + \boldsymbol{f}^{\mathrm{c,n}}_{\mathrm{DEM}}\right)$	$\bar{\boldsymbol{\alpha}}^{\mathrm{n}} = \boldsymbol{I}_i^{-1}\left(\boldsymbol{t}^{\mathrm{d}}_{\mathrm{CFD}} + \boldsymbol{t}^{\mathrm{c,n}}_{\mathrm{DEM}}\right)$
update velocities	$\boldsymbol{u}^{n} = \boldsymbol{u}^{*} + \frac{1}{2}\bar{\boldsymbol{a}}^{\mathrm{n}}\Delta t$	$\boldsymbol{\omega}^{n} = \boldsymbol{\omega}^{*} + \frac{1}{2}\bar{\boldsymbol{\alpha}}^{\mathrm{n}}\Delta t$

(4)

Box I.

considers $\mathcal{B}_i$ to be affected by gravity (g), drag (d), and contact (c) with other bodies or solid walls.

DEM solver is subcycled with respect to the flow solution, that is, contacts and solids positions are repeatedly evaluated with a single set of gravity and drag forces. Integration of the DEM governing Eqs. (3) itself works in a finite difference fashion similar to the LIGGGHTS software [6]. First, the old, intermediate, and new time levels are marked by superscripts (o), (*), and (n), respectively. Next, the angular velocity $\boldsymbol{\omega}_i$ of $\mathcal{B}_i$ is split into an axis of rotation $\boldsymbol{o}_i = \boldsymbol{\omega}_i/\|\boldsymbol{\omega}_i\|$ and a scalar angular velocity $\omega_i = \|\boldsymbol{\omega}_i\|$ representing the rotation of $\mathcal{B}_i$ around $\boldsymbol{o}_i$. Using these concepts and omitting the subscript i denoting $\mathcal{B}_i$, the integration over a single time step Δt adheres to the following steps given in Box I where $\Delta\boldsymbol{x}$ and $\Delta\theta$ are incremental translation and angle of rotation of $\mathcal{B}_i$, respectively. Update of the translational position of $\mathcal{B}$ follows the relation $\boldsymbol{x}^{\mathrm{n}} = \boldsymbol{x}^{\mathrm{o}} + \Delta\boldsymbol{x}$. To update $\mathcal{B}_i$ orientation, $\Delta\theta$ and $\boldsymbol{o}^{*}$ are used to construct the Rodriguez rotation matrix [15], which is used to rotate all the points of $\mathcal{B}_i$ as described in detail in [14].

The given integration scheme splits the updates of forces, accelerations, and velocities on one side, and the changes in solids position and orientation on the other side. The former are updated with time step $\Delta t/2$ at t^{o} and t^{n}, the latter with Δt at t^{*}. Such an approach enables the use of longer time steps.

2.3. Contact treatment

We apply principles of the soft DEM in which a solid–solid overlap is possible during contact, and the contact force is scaled according to the magnitude of such an overlap [16]. The current version of OpenHFDIB-DEM solver uses reformulation of our previous model [14], detailed in Appendix A, that has a direct link to the Cundall and Strack [17] model with contact based on the Hertzian theory [18] and is extended by the dissipation term derived by Tsuji et al. [19]. The new formulation of the contact model provides parameterization and results consistent with the LIGGGHTS software [6] for contacts between two spherical particles, as well as a spherical particle and a planar wall.

3. Software description

The OpenHFDIB-DEM project comprises the main software library and two example solvers, pimpleHFDIBFoam and HFDIBDEMFoam. The latter is designed solely for DEM applications, while the former is designed for fully coupled CFD-DEM. The software can be installed on any system where the OpenFOAM-v8 library is compiled, and the software repository provides a full installation guide and basic tutorials for both solvers. The code itself is strongly object-oriented and written predominantly in C++ leveraging the features of the OpenFOAM library [11]. However, to increase the code efficiency for large arrays, the C++ standard library [20] was also utilized. To enable high performance computing (HPC), the code was optimized and fully parallelized using the message-passing interface (MPI) [21]. In the following, we provide a brief description of the main library and its integration with the pimpleHFDIBFoam solver with emphasis on the parallelization.

Library architecture. OpenHFDIB-DEM library is implemented directly in OpenFOAM and consists of the parent class `OpenHFDIBDEM` that acts as an interface for solvers and as a database for the considered solids. Each solid considered in the presented framework is defined using four intertwined classes, which are directly linked to the parent `OpenHFDIBDEM` class as illustrated in Fig. 2a. The four classes are (i) `geomModel`, responsible for operations with particle geometry; (ii) `addModel`, which takes care of registration of new solids into the computational domain; (iii) `contactModels`, a class for detection and treatment of solid–solid collisions; and (iv) `immersedBody`, accounting for the solid body movement within the computational domain, including coupling with the fluid phase. Finally, the parent `OpenHFDIBDEM` class is, for all the solids, responsible for memory management and task redistribution in parallel run.

The current `geomModel` class (brown in Fig. 2a) considers three geometry types: (i) `shapeBased` with a sole representative sphereBody for spherical solids defined via radius and center; (ii) `stlBased` further divided into convexBody and nonConvexBody defined using their surface representations (STL files); and (iii) `clusterBased`, which is a general structure applied to solids passing through cyclic boundary conditions regardless of their geometry. The `geomModel` class is used to compute particle properties (mass, center of mass and moment of inertia), particle transformations (translation and rotation), and particle projections onto the CFD computational mesh.

On top of registering newly added solids, the `addModel` class is suited to provide position and orientation for the solid when it is initialized. While several particle addition methods are available (see the green fields in Fig. 2a), the fundamental condition for a successful particle addition is to locate a vacant space where a particle can be added without collisions with other particles or walls. This is achieved iteratively; when the particle geometry is projected to the given position, it is investigated for any collisions using the `contactModels` class. When no collision is detected, the particle is added to the given position; otherwise, the particle is disregarded, or a new position is sought, depending on the selected addition method. The currently implemented addition methods contain options to switch between the particle addition according to a given STL position and orientation, and more complex modes that generate distributions with a given solid volume fraction within a user-defined sub-domain. The addition methods also enable repeated particle insertion control based on time period or required volume fraction.

The `contactModels` class (purple in Fig. 2) is divided into two derived classes to account for (i) particle–particle collisions treated by `particleContact`, and (ii) particle–wall collisions treated by `wallContact`. The physics behind the contact models is detailed in Section 2. Both the contact treatment classes are enhanced by the virtual mesh algorithm described in [22], which is contained in its own class `virtualMesh`.

Finally, the `immersedBody` class is designed to account for the body within the computational framework throughout the simulation

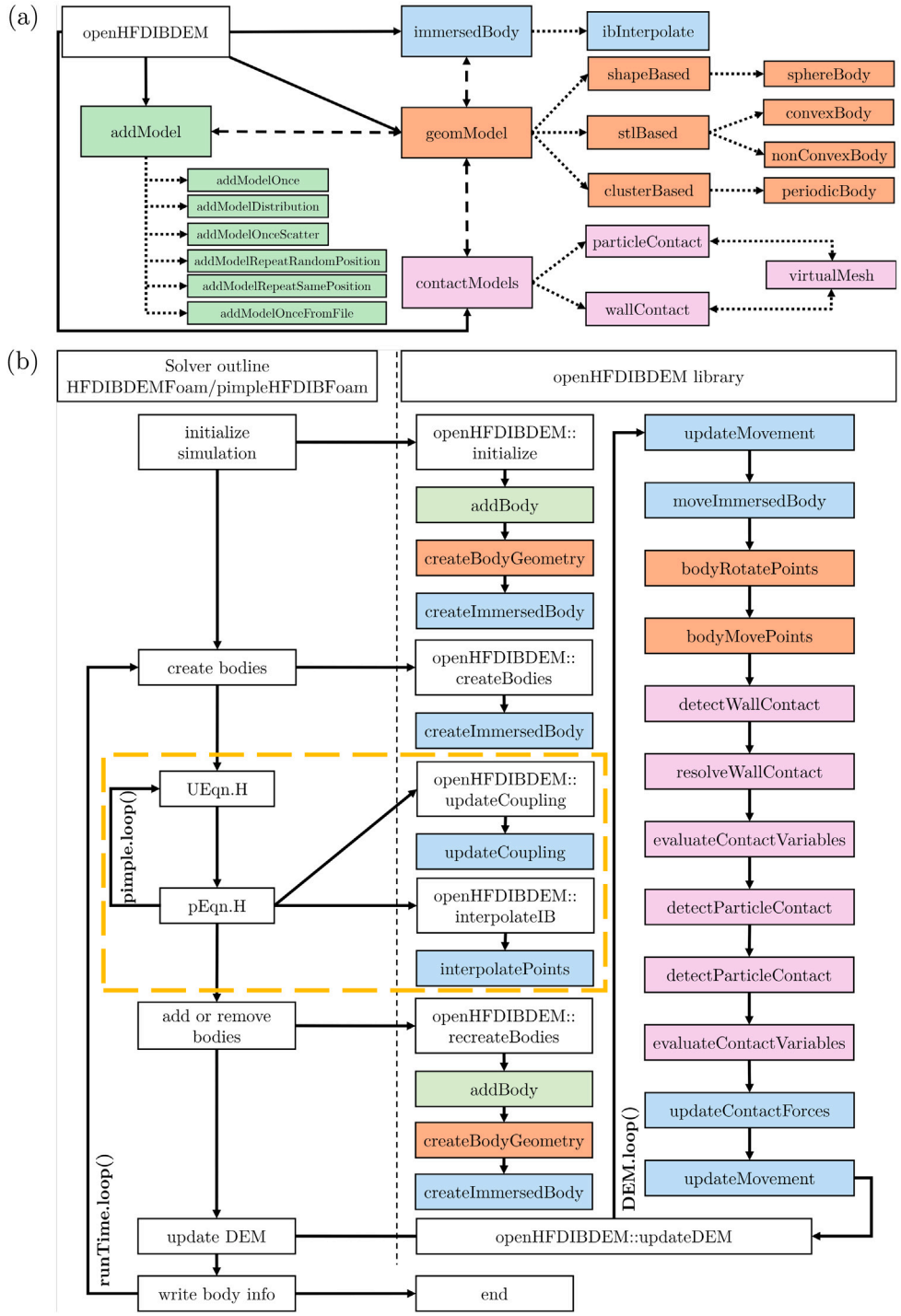


Fig. 2. (a) OpenHFDIB-DEM library structure. (b) Linking the OpenHFDIB-DEM library to the pimpleFoam solver.

time. The class manages the particle geometry and contains methods to evaluate acting forces (fluid coupling and buoyancy/gravity). The particle's position and rotation is updated by a numerical integration, see Eqs. (4). The `immersedBody` class also enables different modes of motion: fully coupled with the fluid, or prescribed particle rotation, translation or both, or a static solid body.

Linking to a CFD solver. Coupling of the OpenHFDIB-DEM library with the standard pimpleFoam solver of the OpenFOAM library is illustrated in Fig. 2b. The solver structure is shown on the left, while the relevant methods from OpenHFDIB-DEM are listed on the right and colored according to the class they appertain to. The library is linked to the solver at five different places. First, during the computation initialization, the `OpenHFDIBDEM` class needs also to be initialized. Second, at the beginning of each CFD time step, the solids are re-projected in the computational domain. Third, during the construction and solution of the pressure equation (`pEqn.H`) inside the PIMPLE loop, the fluid–solid coupling is repeatedly updated and the forcing term $\boldsymbol{f}_{\mathrm{ib}}$ in Eqs. (1) and (2) is updated. Fourth, after convergence of the PIMPLE loop, the solids are either added or removed from the computational domain as driven by the `addModel` class. Finally, at the end of each CFD time step, the DEM loop is entered to resolve the solid movements and potential solid–solid contacts.

Solver parallelization. The *OpenHFDIB-DEM* library is designed specifically to take into account arbitrary polyhedral solids defined using triangulated surfaces (STLs). The contact treatment for non-convex polyhedral solids poses high demands on computational resources, even though the applied algorithms are highly optimized, hence the solver parallelization is necessary. The OpenFOAM library provides MPI-leveraging, a domain-decomposition-based parallelization. It decomposes finite-volume domains (meshes) into subdomains, where each subdomain is assigned to one processor unit (CPU). During the simulation, CPUs work independently and information is exchanged only via processor–processor boundaries, that is, only between the adjacent subdomains. However, this approach, hereafter marked as $\mathcal{P}_{\mathrm{CFD}}$ parallelization, is not suitable for contact treatment in DEM where all contacts may be concentrated in a single subdomain. Another bottleneck of $\mathcal{P}_{\mathrm{CFD}}$ for DEM with complex STL-defined solids is data transfer between processors enforced by a solid moving from one subdomain to another.

The limitations of native OpenFOAM $\mathcal{P}_{\mathrm{CFD}}$ parallelization were overcome by building a second level of parallelization ($\mathcal{P}_{\mathrm{DEM}}$) with the goal of improving load balance when the solids and their contacts are concentrated only in some of the $\mathcal{P}_{\mathrm{CFD}}$ subdomains. To achieve these goals, a contact pool is created that contains all contacts occurring at the given time. The contacts are distributed between the processors independently of $\mathcal{P}_{\mathrm{CFD}}$ domain decomposition. To limit processor–processor communication, a new memory structure is created for $\mathcal{P}_{\mathrm{DEM}}$ in which all particle information is arranged in an n-dimensional array, with n being equal to the number of CPUs used. This way, each CPU accesses a unique set of memory addresses, shifting processor communication from individual particles to large data structures and achieving well-scalable simulations on multiple CPUs, though at the expense of somewhat increased memory requirements.

4. Verification and illustrative examples

To showcase the capabilities of the OpenHFDIB-DEM project, three examples are presented. First, the precision and accuracy of the custom DEM model are verified against the LIGGGHTS software [6]. Second, a simulated pouring of thousands of non-spherical solids is presented with an emphasis on the openHFDIB-DEM parallel computation efficiency. The third test deals with fludization of thousands of differently shaped solid particles, representing a typical large-scale CFD-DEM application. All of the tests presented are included in the repository for this article. The tests were evaluated using a computer with AMD® EPYC™ 7551 64-core CPU and 1 TB RAM. Further verification and examples of the software applications may be found in [14,22]. Finally, an overview of our code capabilities compared to standard CFD-DEM solutions is provided in Appendix B.

4.1. Comparison with LIGGGHTS for spherical particles

The DEM model presented here comprises an updated formulation of contact treatment for arbitrarily shaped solids. In contrast to the standard formulation of the Hertz contact model, the contact forces are scaled by the magnitude of the overlap volume. Nevertheless, the model is consistent with the Hertz model for spheres. LIGGGHTS [6] is a widely spread open-source DEM solver for spherical particles, generally applied as the standard for soft-DEM modeling of the granular matter. Here, we use it as the reference for verification tests of our OpenHFDIB-DEM solver. The presented comparison consists of three verification tests presented in Fig. 3, with material properties and solver settings summarized in Table 1. The solid walls possess the same material properties as the particles.

The first verification test focuses on the treatment of normal contact force. In the test, we evaluate the contact of two spheres. The bottom sphere is static, while the upper one moves towards it on the trajectory connecting the spheres centers, Fig. 3a. The magnitude of the initial velocity of the upper sphere is $u_{\mathrm{i}}^{0} = 1\,\mathrm{ms}^{-1}$.

Table 1
Material properties and simulation settings for verification against the LIGGGHTS solver.

Test No.	Y (GPa)	ε (−)	μ (−)	C_{ℓ} (−)	ν (−)	ρ (kg m^{-3})	d_{char} (cm)	Δt (µs)
1	0.1	1.0, 0.5, 0.06	0.0	0.55	0.0	2500	2.0	10
2	0.1	1.0	1.0	0.55	0.0	2500	8.0	1.0
3	0.5	0.6	0.1	0.55	0.4	2500	0.4	1.0

Table 2
Material properties and simulation settings for DEM pouring of polyhedral particles.

Test No.	Y (GPa)	ε (−)	μ (−)	C_{ℓ} (−)	ν (−)	ρ (kg m^{-3})	d_{char} (mm)	Δt (µs)
4	0.50	0.25	1.0	4	0.5	4000	3.88	1
5	0.50	0.25	1.0	4	0.5	4000	3.99	1
6-left	0.50	0.5	0.5	4	0.5	4000	4.18	1
6-right	0.05	0.25	1.0	4	0.5	5000	7.94	0.5

Fig. 3b shows a comparison between OpenHFDIB-DEM and LIGGGHTS solution for the upper sphere trajectory and different values of the coefficient of restitution. The results are practically identical. The evolution of the normal contact force for the two solvers is compared in Fig. 3c. Slight differences in the force evolution can be observed, especially in the purely elastic case with $\varepsilon = 1.0$. These differences are caused by a different evolution of the overlap volume compared to the overlap distance. However, the moments of the acting force are the same for both OpenHFDIB-DEM and LIGGGHTS, which results in virtually the same particle trajectories for both solvers.

Test 2 simulates a gravity-induced rolling of a sphere down an inclined plane, Fig. 3d. The distance traveled, sphere angular velocity, and acting tangential force at different inclination angles are provided in Fig. 3e,f,g, respectively. Again, the OpenHFDIB-DEM, LIGGGHTS, and analytical solution are practically identical for the distance traveled and angular velocity, Figs. 3e,f. The limit tangential forces also match quite well, Fig. 3g. Furthermore, the initial force oscillation artifacts are significantly reduced in the OpenHFDIB-DEM solution compared to LIGGGHTS.

Test 3 investigates a pouring of 10,000 monodisperse spherical particles into a hexahedral container, Fig. 3h. To analyze the system, we study the evolution of the packed bed porosity ε_{V}, which is practically the same for both OpenHFDIB-DEM and LIGGGHTS, including the bounce-back of spheres indicated by points (†) and (□) in Fig. 3i.

Analysis of the computation costs revealed that LIGGGHTS is approximately 45 times faster than OpenHFDIB-DEM in a DEM simulation of ideal spherical particles. This difference in the computational efficiency is caused by the intended application of OpenHFDIB-DEM for CFD-DEM simulations. Even if only the DEM solver is used in OpenHFDIB-DEM, the CFD computational mesh is still constructed and all particles are projected onto it in each time step, which is a time-consuming operation. The subsequent tests will demonstrate advanced features of our solver that justify the increased computational costs.

4.2. Pure DEM for polyhedral particles pouring

The pouring of generally shaped particles into a hexahedral container is simulated in Tests 4–6, Fig. 4. This study focuses on the solver efficiency and scaling performance with respect to DEM; no fluid is taken into account. The tests consider particles defined using surface representation (STL files). All collisions are evaluated using the virtual mesh algorithm [22], set on the refinement level 4. The material properties and solver settings are summarized in Table 2; the material of the system walls again corresponds to the particles.

We tested four types of particles: an icosahedron, a 320-sided sphere-like polyhedron, a rock-like particle generated in Blender [23],

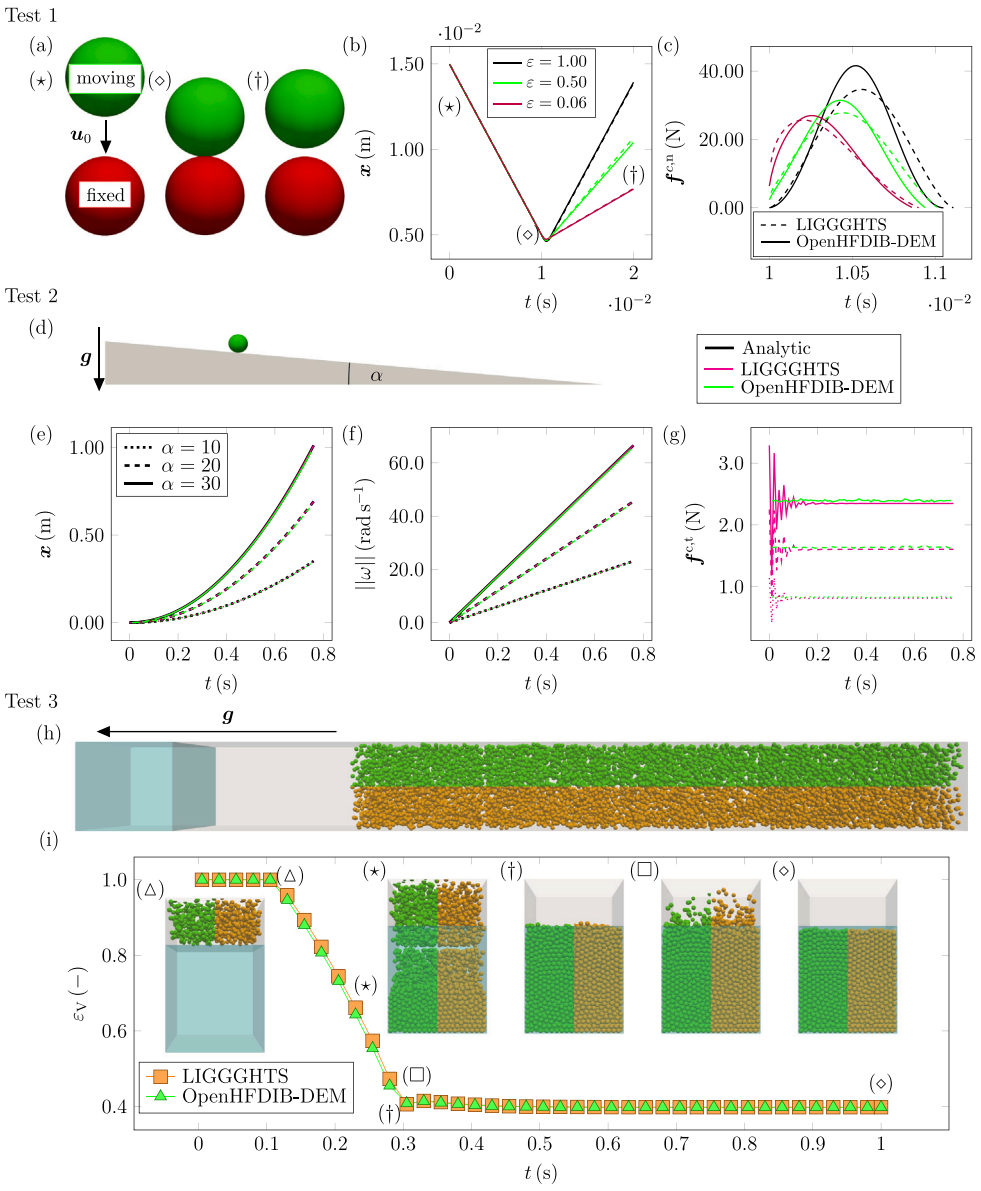


Fig. 3. Verification of OpenHFDIB-DEM against LIGGGHTS.

and a γ-alumina particle taken from [24]. Initial state of the simulations is given in Fig. 4a. Qualitative views of individual particles, together with the number of particles considered (n) and the slope of the scaling trend line (a) are displayed in Fig. 4b. The code scaling in parallelization up to 32 cores is shown in Fig. 4c. In general, the DEM part of the code scales at 60–70% of the ideal (linear) effectiveness. At the moment, the parallelization bottleneck is data transfer between the cores. Consequently, the code scales better for particles that occupy less space in memory, e.g., for icosahedrons (Test 4 in Fig. 4). To achieve good scaling, we recommend to have at least 30 particles per single core while adhering to the OpenFOAM standard of 50,000 cells per core.

4.3. Fluidization with polyhedral particles

The CFD-DEM capabilities of OpenHFDIB-DEM are illustrated in simulations of fluidization with the same icosahedrons and γ-alumina particles as used in Test 4 and Test 6-right in Fig. 4. The particles properties correspond to Table 2 except that the particle density is set to $2500\,\mathrm{kgm^{-3}}$ and restitution coefficient is 0.75. The fluid is a Newtonian liquid with kinematic viscosity $7\cdot10^{-6}\,\mathrm{m^2s^{-1}}$ and density of $1000\,\mathrm{kgm^{-3}}$.

The fluidization column itself is a 250 mm high hexahedron of a square cross section with the side of 80 mm. The CFD computational domain is discretized in 120 × 120 × 375 cells. On the walls of the column, standard no-slip boundary conditions are used. At the inlet, a uniform inlet velocity $\boldsymbol{u}_{\mathrm{inlet}} = (0, 0, 0.085)^{\mathrm{T}}\,\mathrm{ms^{-1}}$ and zero-gradient for pressure are used. At the outlet, zero-gradient for velocity and a prescribed pressure are applied. The integration step for the CFD solver is adaptive, governed by the flow Courant number, and the DEM integration step is set to 1/100 of the current CFD time step.

The results of the fluidization tests are depicted in Fig. 5a, complemented by the temporal evolution of the fluidized bed porosity (ε_{V}) in Fig. 5b. The zone of interest in which ε_{V} is evaluated is indicated by a red box in Fig. 5a. The tests start from an initial, randomly generated bed of a prescribed height in time $t = 0$ s. After roughly 1.5 s of the simulation time, all the tested packed beds reach a stable porosity corresponding to a pseudo-steady state fluidization, Fig. 5b. As expected, the resulting porosity do not depend on the number of particles (1000, 3000 or 5000 icosahedrons) but indeed it depends on the particle size and shape (icosahedrons vs. alumina particles).

The scaling of computational time with parallelization on 1–32 cores is shown in Fig. 5c. The complete CFD-DEM code scales similarly to

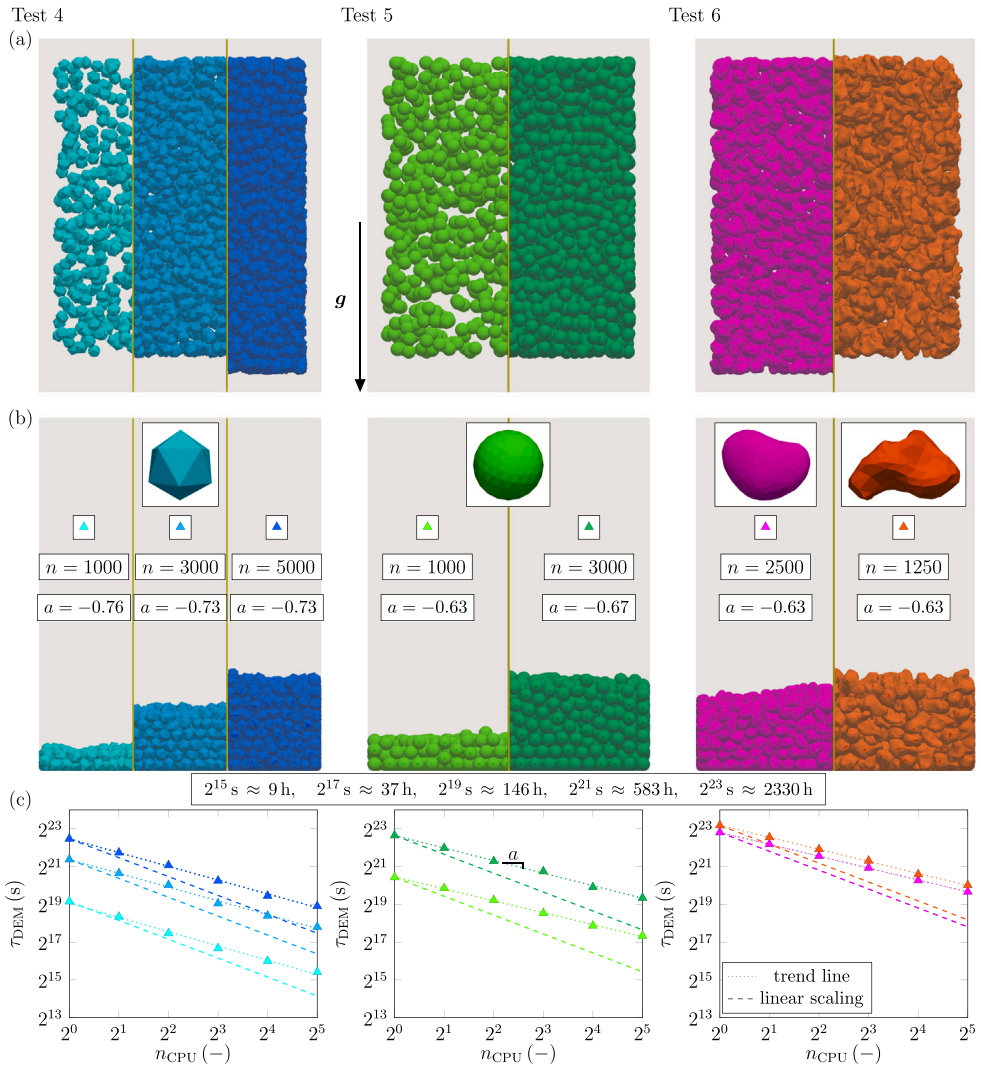


Fig. 4. DEM simulations of pouring of polyhedral particles; n denotes the number of particles, a is the observed slope of scaling in parallelization.

the DEM part of the code, that is, with $\gtrsim 60\%$ effectiveness of the ideal linear scaling. In the system with 1000 icosahedral particles, the DEM is responsible for 44% of the computational costs. With an increasing number of particles or particle complexity, the DEM costs increase. Specifically, DEM is responsible for 82% and 89% of the computational costs of fluidization with 3000 and 5000 icosahedrons, respectively; and for 69% of the computational time for the alumina particles. A conservative estimate is that the current code can be effectively used to simulate cases with $\mathcal{O}(10^4)$ non-spherical particles on a CFD mesh with $\mathcal{O}(10^1)$ cells per characteristic particle dimension.

5. Impact and conclusions

The OpenHFDIB-DEM project is an open-source extension of the OpenFOAM framework, compatible with OpenFOAMv8 and soon to be ported to OpenFOAM 2312. It is designed for particle-resolved CFD-DEM simulations with arbitrarily shaped solids. Particular attention was paid to the software architecture and to the DEM part of the code, in which the contact model was reformulated to be consistent with the standard Hertz model for spheres while keeping its generality with respect to contact between arbitrarily shaped solids. For spheres, OpenHFDIB-DEM provides results in agreement with the LIGGGHTS software. For both DEM and CFD-DEM applications with thousands of non-spherical solids, the computational time scales with the number of cores at roughly 60% of the ideal linear scaling. Due to its open and modular architecture, the OpenHFDIB-DEM library can be implemented in different OpenFOAM-based solvers to ease simulation of various processes involving flow laden with arbitrarily shaped particles.

CRediT authorship contribution statement

Ondřej Studeník: Writing – original draft, Visualization, Validation, Software, Investigation. **Martin Isoz:** Writing – review & editing, Supervision, Software, Funding acquisition, Conceptualization. **Martin Kotouč Šourek:** Validation, Software, Investigation, Formal analysis, Conceptualization. **Petr Kočí:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All the data and codes are publicly available on github.

Acknowledgments

The work was financially supported by the Czech Science Foundation project 22-12227S. Martin Isoz and Ondřej Studeník acknowledge

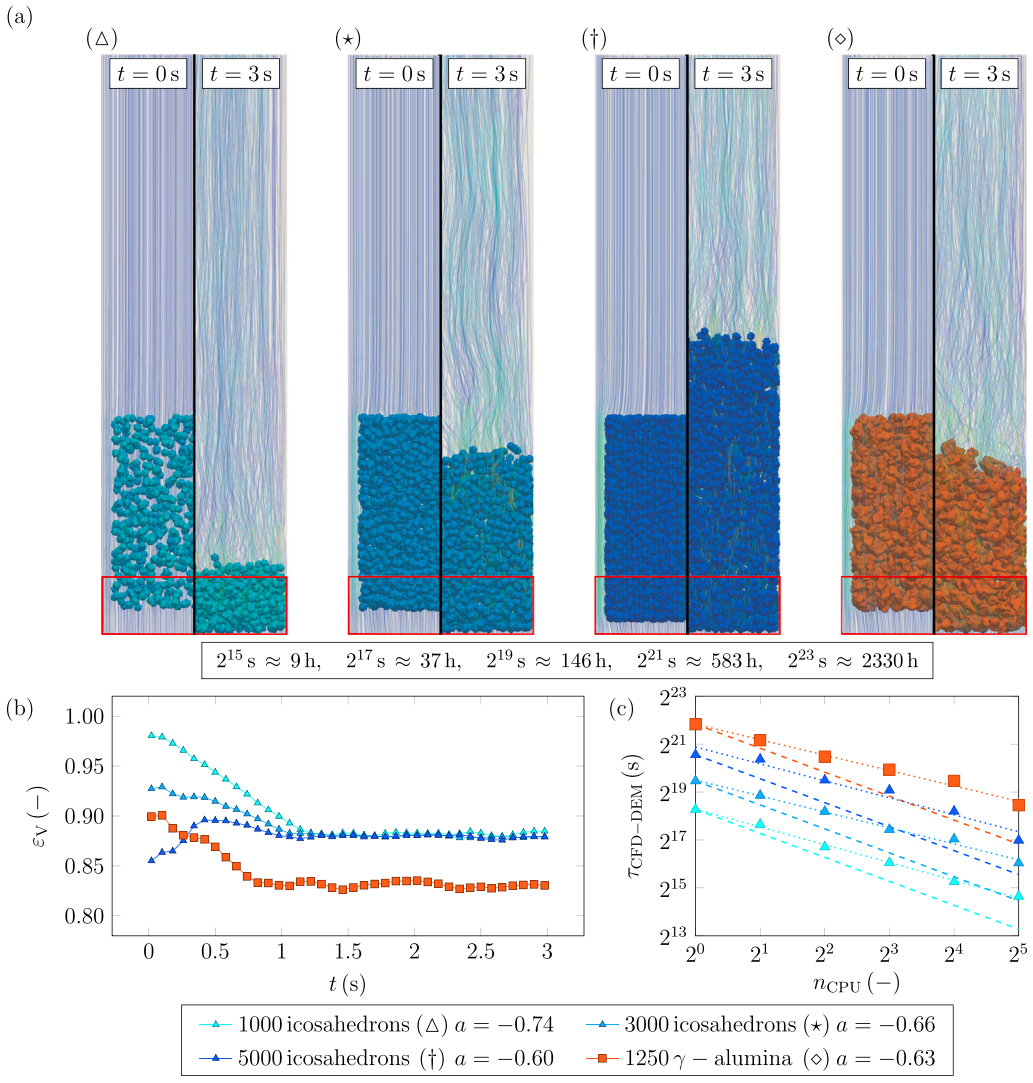


Fig. 5. CFD-DEM simulations of fluidization with differently shaped polyhedral particles; ε_V denotes porosity of the bed, a is the observed slope of scaling in parallelization.

the financial support provided by institutional support RVO:61388998, by the grant project with No. TN02000069/001N of the Technology Agency of the Czech Republic, Czechia, and by the Ministry of Education, Youth, and Sports of the Czech Republic via the project No. CZ.02.01.01/00/22_008/0004591 (Ferroic Multifunctionalities), co-funded by the European Union. Martin Kotouč Šourek thanks Johnson Matthey Technology Centre in Sonning Common for the support and feedback.

Appendix A. Detailed description of contact treatment

The contact model from our previous work [14] was updated considering the Cundall and Strack [17] model. The new formulation is further extended by the dissipation term derived by Tsuji et al. [19]. In the following, we provide a brief overview of the template contact model for spherical particles and list the modifications made to generalize it to arbitrary solids.

Base contact model for spherical particles. In the case of a collision between two spherical particles (i and j), the contact model by Cundall and Strack [17] with the dissipation term from [19] can be summarized in the following relations for the contact normal force acting on the particles ($\boldsymbol{f}^{c,n}$),

$$\boldsymbol{f}^{c,n} = \left(k^n\delta + \gamma^n\sqrt{k^n M^{red}}\,\frac{d\delta}{dt}\right)\boldsymbol{n}^c\,, \tag{A.1}$$

$$k^n = \frac{4}{3}Y^{red}\sqrt{r^{red}\delta}\,, \quad \gamma^n = \frac{-\sqrt{5}\ln(\varepsilon)}{\sqrt{\ln(\varepsilon)^2+\pi^2}}\,, \tag{A.2}$$

where k^n denotes the elastic material stiffness according to the reduced material properties of Young's modulus Y^{red}, sphere radii r^{red} and given particle overlap δ. By γ^n, we mark the damping factor evaluated according to the restitution coefficient ε with M^{red} being the reduced mass of the particles and $\boldsymbol{n}^c$ the contact normal vector. The reduced values are evaluated as a harmonic average of the individual particle properties. Furthermore, in the case of Young modulus, the reduced value is weighted by $(1-\nu^2)$, where ν is the Poisson ratio of the individual materials.

In the case of sphere–sphere collision, the overlap length (δ) is evaluated as

$$\delta = \max\left[0, (r_i + r_j) - (\boldsymbol{x}_i - \boldsymbol{x}_j)\cdot\boldsymbol{n}^c\right]\,, \quad \boldsymbol{n}^c = \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}\,. \tag{A.3}$$

However, this principle cannot be directly applied to collisions of arbitrarily shaped solids, where the overlap length does not always reflect the particle shape and overall scope of the collision.

Formulation for non-spherical particles. During the years, numerous approaches have been proposed to treat the contact of non-spherical particles; see the reviews by Feng [25] and Zhao et al. [26], or the recent work of Zhao and Zhao [27]. In the present solver, we build the contact treatment for arbitrarily shaped particles on the works of Chen

Table B.3
Overview of general capabilities of the most spread CFD-DEM codes compared to openHFDIB-DEM.

Software identifiers					
CFD	OpenFOAMv8	OpenFOAMv5	OpenFOAMv6	ANSYS Fluent	OpenFOAMv8
DEM	HFDIB-DEM	LIGGGHTS	YADE	Rocky DEM	Aspherix
Code information					
open-source	✓	✓	✓	✗	✗
architecture	monolithic	modular	modular	modular	modular
Treatable particles					
spheres	✓	✓	✓	✓	✓
clumps	✗	✗	✗	✓	✓
superquadrics	✗	✗	✗	✓	✓
convex polyhedra	✓	✗	✗	✓	✓
general polyhedra	✓	✗	✗	✓	✓
Coupling scheme					
point force	✗	✗	✓	✓	✗
volume averaged	✗	✓	✓	✓	✓
particle-resolved	✓	✗	✗	✗	✓

[28] and Matuttis and Chen [16], where the concept of overlap volume (V^{o}) as a generalization to the overlap length δ was introduced and studied in detail for polyhedral particles. The formulation of the contact normal force from [16,28] is

$$\boldsymbol{f}^{\mathrm{c,n}} = \left(\frac{Y^{\mathrm{red}} V^{\mathrm{o}}}{\ell^{\mathrm{c}}} + \gamma^* \sqrt{\frac{Y^{\mathrm{red}} M^{\mathrm{red}}}{(\ell^{\mathrm{c}})^3}} \frac{\mathrm{d}V^{\mathrm{o}}}{\mathrm{d}t} \right) \boldsymbol{n}^{\mathrm{c}}, \quad \ell^{\mathrm{c}} = 4 \frac{\|\boldsymbol{\ell}_j\| \, \|\boldsymbol{\ell}_i\|}{\|\boldsymbol{\ell}_j\| + \|\boldsymbol{\ell}_i\|}, \tag{A.4}$$

where ℓ_{c} stands for characteristic contact length with $\|\boldsymbol{\ell}\|$ as the distance between the solids center of mass to the center of mass of the overlap volume V^{o}, and γ^* is an empirical dissipation coefficient. To treat the term of the overlap volume derivative, we assume that the contact cross-sectional area A^{c} is constant during a single DEM time step ($A^{\mathrm{c}} \rightsquigarrow \bar{A}^{\mathrm{c}}$) and the time derivative of the overlap volume in (A.4) can be approximated as

$$\frac{\mathrm{d}V^{\mathrm{o}}}{\mathrm{d}t} = \bar{A}^{\mathrm{c}} \left(\frac{\mathrm{d}\boldsymbol{\ell}^{\mathrm{o}}}{\mathrm{d}t} \cdot \boldsymbol{n}^{\mathrm{c}} \right) = \bar{A}^{\mathrm{c}} \left(\boldsymbol{u}_{\mathrm{r}} \cdot \boldsymbol{n}^{\mathrm{c}} \right), \quad \boldsymbol{u}_{\mathrm{r}} = \boldsymbol{u}_i - \boldsymbol{u}_j + (\boldsymbol{\omega}_i \times \boldsymbol{\ell}_i - \boldsymbol{\omega}_j \times \boldsymbol{\ell}_j) \tag{A.5}$$

where $\boldsymbol{\ell}^{\mathrm{o}}$ is the position of the V^{o} center of mass and $\boldsymbol{u}_{\mathrm{r}}$ stands for the relative velocity of the contact pair. This contact model was implemented as a part of our previous work [14].

To replace the empirical dissipation coefficient γ^* and utilize the widely applied coefficient of restitution (ε), we reformulated the contact model in the current version of our OpenHFDIB-DEM solver. Based on a comparison of elastic terms in Eqs. (A.1) and (A.4), the new formulation is

$$\boldsymbol{f}^{\mathrm{c,n}} = \left(k^{\mathrm{n}}_* \delta^{\mathrm{o}} + \gamma^{\mathrm{n}} \sqrt{k^{\mathrm{n}}_* M^{\mathrm{red}}} \, \boldsymbol{u}_{\mathrm{r}} \cdot \boldsymbol{n}^{\mathrm{c}} \right) \boldsymbol{n}^{\mathrm{c}}, \tag{A.6}$$

$$k^{\mathrm{n}}_* = Y^{\mathrm{red}} \frac{\bar{A}^{\mathrm{c}}}{\ell^{\mathrm{c}}}, \quad \ell^{\mathrm{c}} = C_{\ell} \frac{\|\boldsymbol{\ell}_j\| \, \|\boldsymbol{\ell}_i\|}{\|\boldsymbol{\ell}_j\| + \|\boldsymbol{\ell}_i\|}, \tag{A.7}$$

where k^{n}_* is the effective elastic material stiffness for the volume-defined contact, $\delta^{\mathrm{o}} = V^{\mathrm{o}}/\bar{A}^{\mathrm{c}}$ is derived from the overlap volume and contact cross section area, and C_{ℓ} is the effective curvature of the given particle shape. Note that from the comparison of k^{n} and k^{n}_*, cf. (A.2) and (A.7), it follows that

$$k^{\mathrm{n}} \sim k^{\mathrm{n}}_* \rightsquigarrow k^{\mathrm{n}} = \frac{4}{3} Y^{\mathrm{red}} \sqrt{r^{\mathrm{red}} \delta} \sim Y^{\mathrm{red}} \frac{\bar{A}^{\mathrm{c}}}{\ell^{\mathrm{c}}} = k^{\mathrm{n}}_* \rightsquigarrow \sqrt{r^{\mathrm{red}} \delta} \sim \frac{\bar{A}^{\mathrm{c}}}{\ell^{\mathrm{c}}}. \tag{A.8}$$

The evaluation of contact-defining parameters such as the overlap volume V^{o}, cross-section area $\bar{A}^{\mathrm{c}}$, or the contact center is resolved using a custom algorithm called *virtual mesh*, which is described in detail in [22].

Treatment of tangential forces. To comply with the particles rotation, the tangential forces acting on the particles i and j in contact are (i) compensated for the rotation of the particles, and (ii) evaluated incrementally as proposed by Mindlin and Deresiewicz [29]. The compensation for particles rotation is performed by projecting the acting tangential force ($\boldsymbol{f}^{\mathrm{c,t}}$) from the previous time step ($\boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old}}$) in the direction tangential to the current contact as:

$$\text{project into new direction} \quad \boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old},*} = \boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old}} - \left(\boldsymbol{n}^{\mathrm{c}} \cdot \boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old}} \right) \boldsymbol{n}^{\mathrm{c}}, \tag{A.9}$$

$$\text{rescale to original magnitude} \quad \boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old,cor}} = \frac{\|\boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old}}\|}{\|\boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old},*}\|} \boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old},*}. \tag{A.10}$$

The increment in tangential force is, in agreement with [29] but leveraging the modifications similar as proposed for the treatment of the contact normal force, defined as

$$\begin{aligned} \Delta \boldsymbol{f}^{\mathrm{c,t}} &= k^{\mathrm{t}}_* \, \Delta \boldsymbol{\xi}^{\mathrm{t}} - 2\gamma^{\mathrm{n}} \sqrt{k^{\mathrm{t}}_* M^{\mathrm{red}}} \, \boldsymbol{u}^{\mathrm{t}}_{\mathrm{r}}, \quad k^{\mathrm{t}}_* = 8 G^{\mathrm{red}} \frac{\bar{A}^{\mathrm{c}}}{\ell^{\mathrm{c}}}, \\ \Delta \boldsymbol{\xi}^{\mathrm{t}} &= \boldsymbol{u}^{\mathrm{t}}_{\mathrm{r}} \, \Delta t, \quad \boldsymbol{u}^{\mathrm{t}}_{\mathrm{r}} = \boldsymbol{u}_{\mathrm{r}} - \left(\boldsymbol{u}_{\mathrm{r}} \cdot \boldsymbol{n}^{\mathrm{c}} \right) \boldsymbol{n}^{\mathrm{c}}, \end{aligned} \tag{A.11}$$

where k^{t}_* is the tangential elastic material stiffness, for the evaluation of which a similar argument as in (A.8) was applied to replace $\sqrt{r^{\mathrm{red}}\delta}$ by $\bar{A}^{\mathrm{c}}/\ell^{\mathrm{c}}$. The reduced shear modulus G^{red} is computed as a harmonic average of the properties of the individual particles weighted by $(2-\nu^2)$. The increment of the elastic tangential force is scaled according to the change of the tangential overlap $\Delta\boldsymbol{\xi}^{\mathrm{t}}$ while the incremental dumping is proportional to the particles relative tangential velocity $\boldsymbol{u}^{\mathrm{t}}_{\mathrm{r}}$. Finally, the new tangential contact force $\boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{new}}$ is computed in the following manner,

$$\boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{new}} = \boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{old}} + \Delta \boldsymbol{f}^{\mathrm{c,t}}. \tag{A.12}$$

Implementation of friction and contact force application. To conclude the treatment of the collision forces, the new tangential force $\boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{new}}$ is corrected to comply with the Coulomb friction with the coefficient of sliding friction μ^{fr} as

$$\text{If} \quad \|\boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{new}}\| > \mu^{\mathrm{fr}} \|\boldsymbol{f}^{\mathrm{c,n}}\| \quad \text{then} \quad \boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{new}} = \mu^{\mathrm{fr}} \frac{\|\boldsymbol{f}^{\mathrm{c,n}}\|}{\|\boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{new}}\|} \boldsymbol{f}^{\mathrm{c,t}}_{\mathrm{new}}. \tag{A.13}$$

The normal and tangential forces are summed up to evaluate the final contact force $\boldsymbol{f}^{c} = \boldsymbol{f}^{\mathrm{c,n}} + \boldsymbol{f}^{\mathrm{c,t}}$, which is applied to the right hand sides of Eq. (3) affecting the colliding particles $\mathcal{B}_i$ and $\mathcal{B}_j$:

$$\begin{aligned} \boldsymbol{f}^{c}_{i} &= \boldsymbol{f}^{\mathrm{c},}, \quad \boldsymbol{t}^{c}_{i} = \boldsymbol{\ell}_i \times \boldsymbol{f}^{c} \\ \boldsymbol{f}^{c}_{j} &= -\boldsymbol{f}^{\mathrm{c},}, \quad \boldsymbol{t}^{c}_{j} = \boldsymbol{\ell}_j \times (-\boldsymbol{f}^{c}). \end{aligned} \tag{A.14}$$

Appendix B. Available CFD-DEM codes capabilities

In Table B.3, we list general capabilities of the most used CFD-DEM codes allowing for fully coupled simulations of particle-laden flows. The codes capabilities are compared with the openHFDIB-DEM solver presented. Note that modular architecture stands for two independent codes linked via input/output interfaces, while monolithic architecture represents a single code where internal variables from CFD are available to DEM and vice versa. The listed features of ANSYS Fluent and Aspherix are based on their web presentations as no documentation is freely available.

References

[1] Börner M, Bück A, Tsotsas E. DEM-CFD investigation of particle residence time distribution in top-spray fluidised bed granulation. Chem Eng Sci 2017;161:187–97.

[2] Genovese D, Lozano J, Rao M. The rheology of colloidal and noncolloidal food dispersions. J Food Sci 2007;72(2):R11–20.

[3] Hilton J, Mason L, Cleary P. Dynamics of gas-solid fluidised beds with non-spherical particle geometry. Chem Eng Sci 2010;65(5):1584–96.

[4] Blažek M, Žalud M, Kočí P, York A, Schlepütz C, Stampanoni M, et al. Wash-coating of catalytic particulate filters studied by time-resolved X–ray tomography. Chem Eng J 2021;409:128057–1–128057–14.

[5] Kloss C, Goniva C, Hager A, Amberger S, Pirker S. Models, algorithms and validation for opensource DEM and CFD-DEM. Prog Comput Fluid Dyn 2012;12:140–52.

[6] Kloss C, Goniva C. LIGGHTS – open source discrete element simulations of granular materials based on LAMMPS. In: Supplemental proceedings: Materials fabrication, properties, characterization and modeling, vol. 2. The Minerals, Metals & Materials Society (TMS); 2011, p. 781–8. http://dx.doi.org/10.1002/9781118062142.ch94.

[7] Šmilauer V, Angelidakis V, Catalano E, Caulk R, Chareyre B, Chèvremont W, et al. Yade documentation. 3rd ed. Zenodo; 2021, http://dx.doi.org/10.5281/ZENODO.5705394.

[8] Ansys-Fluent. Ansys Fluent 12.0 theory guid. Canonsburg: ANSYS, Inc, SAS IP, Inc.; 2015, URL: https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/main_pre.htm.

[9] Ansys-Rocky. Rocky user manual, 4.0. Canonsburg: ANSYS, Inc, SAS IP, Inc.; 2024, URL: http://www.rocky-dem.com.

[10] DCS Computing GmbH. Aspherix website (n.d.). 2024, URL: https://www.aspherix-dem.com/.

[11] OpenCFD. OpenFOAM: The open source CFD toolbox. User guide version 1.4. Reading UK: OpenCFD Limited; 2007.

[12] Municchi F, Radl S. Consistent closures for Euler-Lagrange models of bi-disperse gas-particle suspensions derived from particle-resolved direct numerical simulations. Int J Heat Mass Transfer 2017;111:171–90.

[13] Municchi F, Radl S. Momentum, heat and mass transfer simulations of bounded dense mono-dispersed gas-particle systems. Int J Heat Mass Transfer 2018;120:1146–61.

[14] Isoz M, Kotouč Šourek M, Studeník O, Kočí P. Hybrid fictitious domain-immersed boundary solver coupled with discrete element method for simulations of flows laden with arbitrarily-shaped particles. Comput & Fluids 2022;244:105538.

[15] Rodrigues O. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendants des causes qui peuvent les produire. J Math Pures Appl (9) 1840;5.

[16] Matuttis H, Chen J. Understanding the discrete element method: Simulation of non-spherical particles for granular and multi-body systems. Wiley; 2014.

[17] Cundall PA, Strack ODL. A discrete numerical model for granular assemblies. Géotechnique 1979;29(1):47–65.

[18] Hertz H. Ueber die Berührung fester elastischer K orper. J Reine Angew Math 1882;1882(92):156–71.

[19] Tsuji Y, Tanaka T, Ishida T. Lagrangian numerical simulation of plug flow of cohesionless particles in a horizontal pipe. Powder Technol 1992;71(3):239–50.

[20] ISO. ISO\IEC 14882:2020: Programming languages— C++. 6th ed. pub-ISO; 2020, p. 1853, URL:.

[21] Bruck J, Dolev D, Ho C, Roşu M, Strong R. Efficient message passing interface (MPI) for parallel computing on clusters of workstations. J Parallel Distrib Comput 1997;40(1):19–34.

[22] Kotouč Šourek M, Studeník O, Isoz M, Kočí P, York AP. Viscosity prediction for dense suspensions of non-spherical particles based on CFD-DEM simulations. Powder Technol 2024;444:120067.

[23] Community BO. Blender - A 3D modelling and rendering package. Stichting Blender Foundation, Amsterdam: Blender Foundation; 2018, URL: http://www.blender.org.

[24] Ditscherlein R, Furat O, Löwer E, Mehnert R, Trunk R, Leißner T, et al. PARROT: A pilot study on the open access provision of particle-discrete tomographic datasets. Microsc Microanal 2022;28(2):350–60.

[25] Feng Y. Thirty years of developments in contact modelling of non-spherical particles in DEM: a selective review. Acta Mech Sinica 2023;39:722343.

[26] Zhao J, Zhao S, Luding S. The role of particle shape in computational modelling of granular matter. Nat Rev Phys 2023;5:505–25.

[27] Zhao S, Zhao J. Revolutionizing granular matter simulations by high-performance ray tracing discrete element method for arbitrarily-shaped particles. Comput Methods Appl Mech Engrg 2023;416:116370.

[28] Chen J. Discrete element method for 3D simulations of mechanical systems of non-spherical granular materials [Ph.D. thesis], Chōfu,Tokyo, Japan: The University of Electro-Communications; 2012.

[29] Mindlin RD, Deresiewicz H. Elastic spheres in contact under varying oblique forces. J Appl Mech 1953;20(3):327–44.