YADE - An extensible framework for the interactive simulation of multiscale, multiphase, and multiphysics particulate systems

Vasileios Angelidakis, Katia Boschi, Karol Brzeziński, Robert A Caulk, Bruno Chareyre, Carlos Andrés del Valle, Jérôme Duriez, Anton Gladky, Dingeman L H van der Haven, Janek Kozicki, et al.

Computer Physics Communications 304 (2024) 109293

Contents lists available at ScienceDirect

Computer Physics Communications

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

Computer Programs in Physics

YADE - An extensible framework for the interactive simulation of multiscale, multiphase, and multiphysics particulate systems ☆,☆☆

Vasileios Angelidakis [a,b,*], Katia Boschi [c], Karol Brzeziński [d], Robert A. Caulk [e], Bruno Chareyre [e,*], Carlos Andrés del Valle [f], Jérôme Duriez [g], Anton Gladky [h], Dingeman L.H. van der Haven [i,j], Janek Kozicki [k,l], Gerald Pekmezi [m], Luc Scholtès [n], Klaus Thoeni [o]

[a] School of Natural and Built Environment, Queen's University Belfast, BT9 5AG Belfast, United Kingdom
[b] School of Engineering, Newcastle University, NE1 7RU, Newcastle upon Tyne, United Kingdom
[c] Department of Civil and Environmental Engineering, Politecnico di Milano, 20133 Milan, Italy
[d] Faculty of Civil Engineering, Warsaw University of Technology, 00 637 Warsaw, Poland
[e] Univ. Grenoble Alpes, CNRS, Grenoble INP, 3SR, 38000 Grenoble, France
[f] Departamento de Física, Universidad Nacional de Colombia, Carrera 45 No. 26-85, Edificio Uriel Gutiérrez, Bogotá D.C., Colombia
[g] INRAE, Aix Marseille Univ, RECOVER, Aix-en-Provence, France
[h] Independent researcher, 09618 Brand-Erbisdorf, Germany
[i] Department of Materials Science & Metallurgy, University of Cambridge, CB3 0FS Cambridge, United Kingdom
[j] Oral Drug Product Process Development, Novo Nordisk A/S, 2760 Måløv, Denmark
[k] Faculty of Applied Physics and Mathematics, Gdańsk University of Technology, 80-233 Gdańsk, Poland
[l] Advanced Materials Center, Gdańsk University of Technology, 80-233 Gdańsk, Poland
[m] Department of Mechanical Engineering, The University of Alabama at Birmingham, Birmingham, AL, USA
[n] Université Clermont Auvergne, CNRS, IRD, OPGC, Laboratoire Magmas et Volcans, Clermont-Ferrand, France
[o] Centre for Geotechnical Science and Engineering, The University of Newcastle, 2308 Callaghan, Australia

ARTICLE INFO

Dataset link: https://gitlab.com/yade-dev/trunk

Keywords:
Discrete element method (DEM)
Open-source software
Granular materials
Non-spherical particles
Coupled methods
Parallel computing

ABSTRACT

This contribution presents the key elements of YADE, an extensible open-source framework for dynamic simulations. During the past 19 years, YADE has evolved from "Yet Another Dynamic Engine" to a versatile multiscale and multiphysics solver, counting a large, active, and growing community of users and developers. The computationally intense parts of the source code are written in C++, using flexible object models that allow for easy implementation of new features. The source code is wrapped in Python, equipping the software with an interactive kernel used for rapid and concise scene construction, simulation control, post-processing, and debugging. The project, including documentation and examples, is hosted on https://yade-dem.org, while the source code is freely available on GitLab. Over the last decade, YADE has expanded in terms of capabilities thanks to the contribution of many developers from different fields of expertise, including soil and rock mechanics, chemical engineering, physics, bulk material handling, and mineral processing. The rapid growth of YADE can be attributed to (1) the careful and robust design of the framework core, (2) a continuous integration pipeline with fully embedded thorough tests which are executed upon each merge request, ensuring stable compilation for various operating systems, and (3) user-friendliness, facilitated by the Python interface, detailed documentation, and rigorous user support. In this paper, we review the main features of YADE, highlighting its versatility in terms of applications, its flexibility in terms of code development, as well as recent improvements in terms of computational efficiency.

☆ The review of this paper was arranged by Prof. Andrew Hazel.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding authors.
E-mail addresses: v.angelidakis@qub.ac.uk (V. Angelidakis), bruno.chareyre@3sr-grenoble.fr (B. Chareyre).

https://doi.org/10.1016/j.cpc.2024.109293
Received 23 February 2024; Received in revised form 17 June 2024; Accepted 21 June 2024
Available online 28 June 2024

Program summary
Program Title: YADE - Yet Another Dynamic Engine
CPC Library link to program files: https://doi.org/10.17632/n4f5fw97rd.1
Developer's repository link: https://gitlab.com/yade-dev/trunk
Licensing provisions: GNU General Public License 2
Programming language: C++, Python
Nature of problem: Numerical simulation of many-particle systems requires accurate models for particle-to-particle interactions, efficient contact detection between objects of various shapes, and robust time integration. In addition, the flow of fluids, thermal effects, as well as other coupled problems in the presence of particles are found in many fundamental and practical applications and they need dedicated computational tools. YADE provides a computational framework to perform such simulations using the discrete element method and multiple extensions of it.
Solution method: YADE simulates particulate systems using the Discrete Element Method (DEM) in a flexible platform combining C++ and Python. It provides a large variety of shape and interaction models, an explicit time-integration scheme, and many post-processing tools. YADE features dedicated solvers for both fluid and heat fluxes at the pore scale, and it supports couplings with third-party software such as Escript (finite element method) and OpenFOAM (computational fluid dynamics).

1. Introduction

Particulate systems, i.e., collections of interacting discrete elements, are ubiquitous in both industry and nature. Understanding and being able to predict the behavior of particulate systems is thus of paramount importance to the chemical, pharmaceutical, and manufacturing industries as well as fundamental to every earth science concerned with either engineering applications (geotechnics, mining, natural resources management), risk assessments (landslides, earthquakes) or, more generally, geophysical processes. In this context, discontinuous numerical methods have attracted more and more attention in recent years because of their intrinsic capability to describe the true nature of particulate systems as a collection of bodies and their responses to different kinds of loading of either mechanical, hydraulic, or thermal origin.

Originating in the field of soil mechanics in the 1970s, the discrete element method (DEM) simulates individual particles as discrete entities with defined geometries, material properties, and interactions [29]. By tracking the motion, forces, and contacts between these discrete elements over time, the DEM provides a microscopic understanding of macroscopic behaviors observed in particulate systems. As a matter of fact, the DEM shares many principles with the Molecular Dynamics (MD) method in which materials are studied at the most fundamental level by simulating the interactions between individual molecules or atoms [67]. Hereafter, the term DEM refers to soft contact models and explicit time-integration, as pioneered by Cundall and Strack [29] and widely covered by both closed-source and open-source codes [63,4,77,107,110], as opposed to the implicit scheme and possibly non-smooth interactions of Contact Dynamics [57].

YADE [97] is an open-source package based on the smooth DEM. It was first introduced in 2008 [64] to provide a research platform capable of handling various numerical models and facilitating their coupling. Since then, thanks to its versatility and its open development model, YADE has benefited from the contributions of many developers to simulate multiscale and multiphysics problems of growing complexity. This continued activity is visible through numerous research projects, Ph.D. theses, and scientific articles that have relied on it over the years.[1] YADE is written mainly in object-oriented C++, but the objects are most commonly manipulated using a Python interpreter thanks to systematic Python binding. The Python layer facilitates scene construction, simulation control, and post-processing. YADE offers a stable release version available as a package in all Debian, Ubuntu, and DEB-based distributions, as well as up-to-date versions containing the latest developments available, either as a ready-to-use packaged version or as a source code that can be modified and compiled with dedicated add-ons. The developers ensure that every new contribution is both numerically and scientifically sound and try to make YADE as easy to learn as possible for new users by providing a detailed reference manual[2] as well as numerous working examples.[3] In addition, both users and developers can benefit from the support of the community thanks to the issue tracker on GitLab.[4] Finally, as YADE is fully open-source under a GNU General Public License (GNU GPL) license, all features implemented in it can be accessed and reused freely for both non-commercial and commercial purposes under consideration of the GNU GPL license terms.

In the following, we describe some key aspects and unique features of YADE. Section 2 presents YADE modeling features, i.e., the ingredients of DEM simulations that are available in the platform for simulating particulate systems; it includes various formulations for particle shape, interaction models, boundary conditions, and couplings with other methods. Section 3 describes the practical aspects of how YADE is used and developed, while Section 4 sets a deeper technical focus on some implementation choices that contribute to the versatility and computational efficiency of YADE.

2. Modeling features

2.1. Particle shapes

YADE simulates objects of various shapes. For non-spherical particles, YADE provides implementations of several integration algorithms for particle rotations, including the ones proposed by Fincham [43], Omelyan [79], and del Valle et al. [32]. The translational motion is independent of shape and uses the conventional (symplectic) finite difference scheme used in Cundall and Strack [29]. Fig. 1 illustrates the available classes to simulate rigid non-spherical objects. YADE supports the modeling of cuboidal (Box), triangular (Facet), and planar (Wall) objects that are commonly used for representing rigid boundaries and containers. Additionally, YADE provides functions for generating complex domains from mesh files in formats like gengeo, gmsh, gts, iges, stl, unv and representing them with collections of Facets.

Sharp, convex polyhedra are available using two shape classes: Polyhedra [42] and PotentialBlock [11]. The Polyhedra class utilizes computational geometry concepts and the CGAL[5] library [62] to facilitate contact detection and to compute the contact volume that is used in a volume-dependent contact force model. The PotentialBlock class uses an alternative modeling strategy, where the geometric

[1] An extensive list of references is available on https://www.yade-dem.org/doc/publications.html.

[2] https://www.yade-dem.org/doc/index-toctree.html.

[3] https://yade-dem.org/doc/tutorial-more-examples.html.

[4] https://gitlab.com/yade-dev/answers.

[5] Requires CGAL version 4.4.

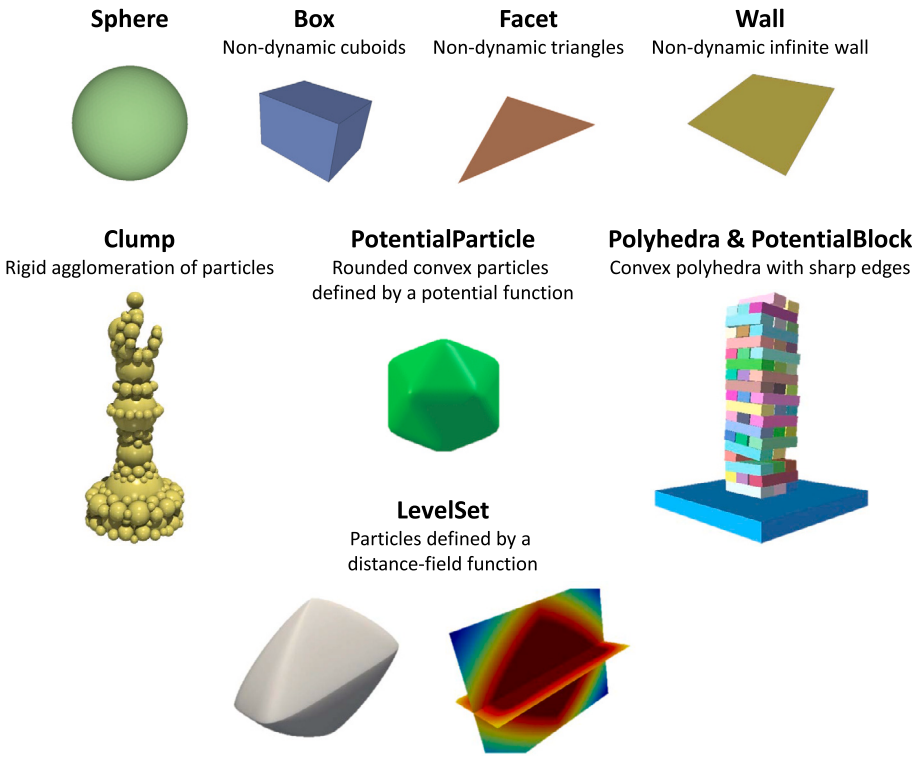

Fig. 1. YADE classes for rigid particle shapes.

plane defining each particle face is established, and the interior of each particle is defined as the shared space of the planar inequalities formed by the planes corresponding to the particle face. Contact detection is facilitated using linear programming and convex optimization techniques to detect whether a feasible region exists within the inequalities defined by the faces of two particles in contact. In particular, the contact point between two `PotentialBlocks` is calculated as the analytic center of the planar inequalities in the overlap region, whereas the contact normal vector is calculated using the gradient of an inner potential particle, for each particle, i.e. a smaller rounded version of each particle that is defined by a piece-wise implicit potential function. The penetration distance used to calculate normal contact forces is calculated using a bracketed search algorithm from the contact point to the surface of the particles along the contact normal direction, whereas another bracketed search is employed in the perpendicular direction to trace the surface of the contact area, which is used to establish non-linear force-deformation contact behavior.

The `PotentialParticle` [12] shape class represents generalized rounded convex particles described by a single piece-wise potential function, the value of which is zero on the surface and positive outside the particle. A constrained minimization method is then employed to establish contact and compute a contact point and contact normal vector, which in turn determine a penetration distance and contact area used to calculate contact forces, where the constraints for a `PotentialParticle` are curved.

YADE's `LevelSet` shape class [35] represents particles with virtually any shape. `LevelSet` uses the level-set discrete element method (LS-DEM) [59,61], which defines the particle surface implicitly using the zero contour of a scalar function, chosen here as the signed distance function expressed in a spatially discretized form (no closed-form expression is required). The latter can be obtained using the Fast Marching Method whenever the location of the particle surface is known (e.g. [93]). To detect contacts, a primary-secondary contact algorithm usually combines the signed-distance field for a particle with the surface node discretization of another particle, both being simultaneously used for distance queries. Fig. 2 illustrates such a `LevelSet` description for the case of a rock aggregate. Using surface nodes, however, may raise issues because results depend on how the surface nodes are distributed and the number of nodes [34,35]. As an alternative to surface nodes, a volume-interacting level-set discrete element method (VLS-DEM) is provided [54], which is based on the same `LevelSet` shape class but utilizes the overlap volume to determine the interparticle forces, thereby removing the need for surface nodes. The computational cost is higher, but in return, VLS-DEM can handle arbitrarily complex geometries, including those of highly concave, angular, and rough particles [54,55].

The `Clump` shape class defines rigid aggregates formed by many particles (called "members") to represent generic shapes, including concave ones (e.g. [40]). The computational advantage of a `Clump` over a group of bonded particles (see next section) is that the interactions between members of one clump are ignored, and the motion of all members is dictated by the clump itself. Comparisons between clumps and other types of non-spherical particle types have been carried out in the literature to quantify their accuracy and efficiency [39]. The interactions between clumps need no specific implementation; they simply result from interactions between their constituents. Clumps of spheres, approximating the physical shape(s) at hand, can be generated using dedicated generation algorithms [5]. The number of spheres in a clump logically controls the accuracy of the shape approximation and the increase in computational cost. Clumps are commonly used to simulate multisphere bodies, yet the clump implementation in `YADE` allows the formation of aggregates using any particle shape, e.g. convex polyhedra can be clumped to form concave polyhedra.

Polyhedra and clumps can be used to simulate crushable particles with the fragment replacement method. This approach requires adopting a failure criterion for the strength evaluation of the particles. If a particle reaches the limit state, breakage occurs, and smaller particles replace the broken one. Eliáš [42] and Gladkyy and Kuna [46] implemented two algorithms to simulate the fragmentation of sharp polyhedra, where the particle stress tensor is used in conjunction with a failure criterion to determine when particles break. The main differences between these models lie in the type of failure criterion considered and

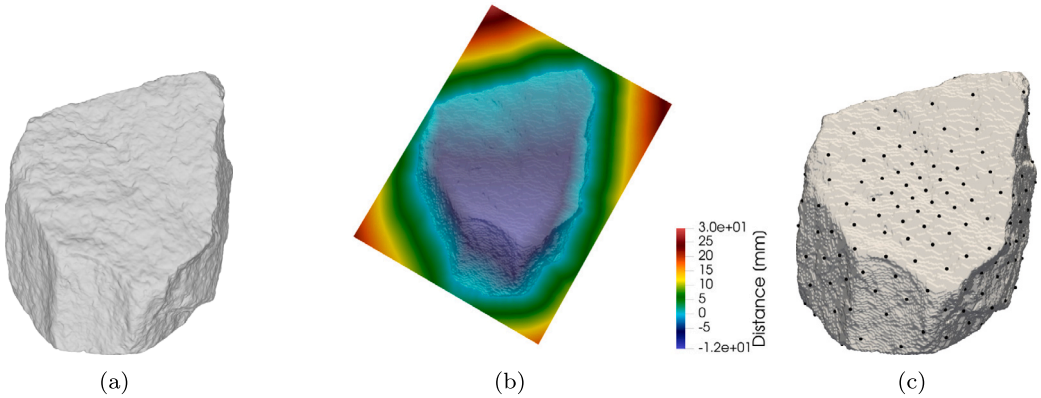

Fig. 2. Level Set description of a centimetric rock aggregate: (a) scanned 3D view of the actual particle and its counterpart in YADE with (b) distance data and (c) surface nodes (a total of 264). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

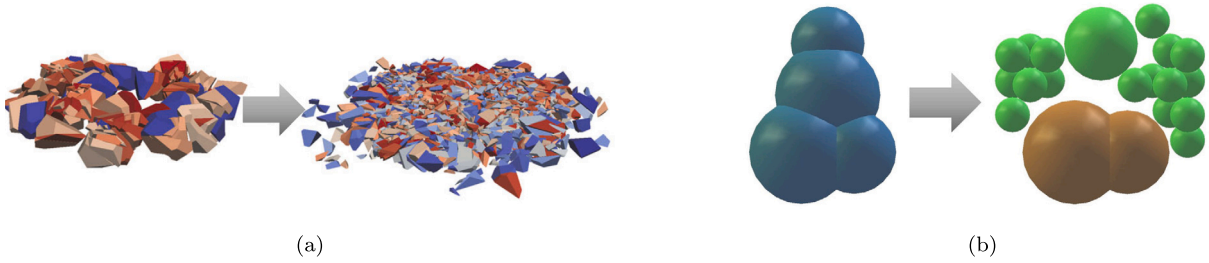

Fig. 3. Example of: (a) polyhedra breakage (modified after [46]) and (b) clump breakage.

how particle size effects are taken into account to calculate the single particle crushing strength. An example of breakage due to oedometric compression is shown in Fig. 3(a), where the initial 22 polyhedra break into more than 3 000 fragments. Brzeziński and Gladky [17] adopted the same failure criterion for a clump breakage algorithm. This method was developed for multisphere particles (i.e. clumps), but it can also be used with standalone spheres [16]. The algorithm evaluates the strength of each clump constituent. If the stress exceeds the clump member's strength, the clump is replaced by standalone spherical fragments. The volume of the new particles is adjusted so that the total volume and mass of the particles in the simulation remain unchanged. Fig. 3(b) shows an example of a clump composed of four spheres before and after breakage. The middle sphere breaks into smaller fragments during the compression, whereas the other spheres remain intact. The remaining parts of the clump split into a new clump (brown) and separated spherical particles (green).

2.2. Deformable particles and flexible structures

The simplest method for modeling deformable particles and flexible structures is to "glue" or "bond" spheres together [85]. This means that attractive forces between particles are allowed, i.e. the interaction law is cohesive. The cohesive link can have a threshold that allows the bonds to break. YADE implements several cohesive contact models that enable the simulation of deformable bodies such as rubber particles (e.g. [7,8]) and flexible structures such as wire meshes (e.g. [104,103]) and woven fabrics (e.g. [27]).

Another option to model deformable particles in YADE is the `DeformControl` feature. In this case, the deformation is achieved by expanding the radius of the spheres based on their overlap so that the volume of the material is kept constant [53]. This approach is useful for modeling the compaction of powders and other granular materials, such as metal pellets, where significant strain hardening occurs because of the additional contacts formed due to plastic deformation.

The modeling of more complex deformable structures can be achieved using the `GridConnection` and `Pfacet` shape classes [13,41]. These classes allow the representation of catenary-like structures formed as the Minkowski sum of a sphere and polyline/polytope (Fig. 4). Unlike bonded spheres, spheropolyhedra have no artificial surface bumpiness, and thus the contact force between them and other objects is a continuous function of the relative displacement (Fig. 5). This shape model is compatible with all sphere-sphere contact models since all interactions are seen as interactions between virtual spheres on both sides of the contact and moving together with the contact point. The nodes of the structures have six degrees of freedom (translational and rotational), and, as such, these classes can be used to represent complex beams, grids, shells, and membrane structures, including hollow spheropolyhedra. These structures can break, and as long as the internal interactions are assigned a failure condition (such as cohesive interactions, typically); they are otherwise purely elastic. As mentioned in Section 2.1, YADE allows any type of shape to be clumped together to form a rigid body. This also applies to `GridConnection` and `Pfacet` elements, i.e. clumping the `PFacets` of a deformable object turns it into a rigid object, which is a way to represent rounded convex and concave polyhedra.

2.3. Broad-phase and narrow-phase collision detection

For the sake of computational efficiency, YADE performs contact detection in two phases. First, a *broad-phase* detection establishes pairs of particles which may interact (a list of close neighbors with or without actual contact) - the broad phase results in a list of *potential* interactions. Then, the *narrow-phase* goes through the potential interactions to determine their geometry precisely and decide which ones are real interactions - for which forces are finally computed. All particle types undergo the same broad-phase contact detection regardless of their detailed shape, whereas the narrow-phase contact detection differs for each combination of shapes, i.e. the precise contact detection for sphere-sphere pairs is different from the one for sphere-polyhedra or polyhedra-polyhedra pairs.

The broad phase is based on a sweep-and-prune algorithm [94]: each body is assigned an axis-aligned bounding box (AABB), and the extrema

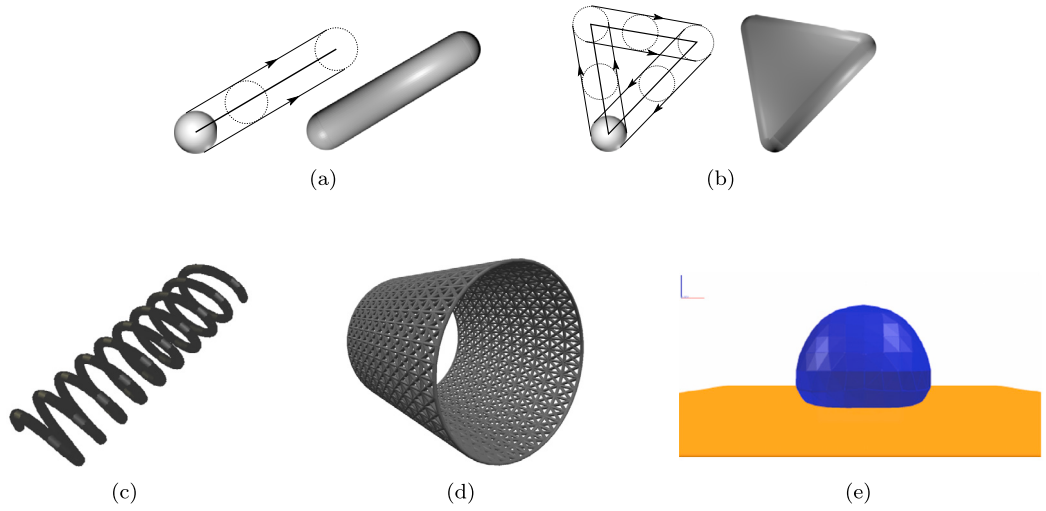

Fig. 4. Complex structures and deformable objects: (a) basic `GridConnection` (rounded cylinder) corresponding to the Minkowski sum of a sphere and a line segment, (b) basic `PFacet` corresponding to the Minkowski sum of a sphere and a triangular facet, (c) helicoidal beam made from `GridConnection`s, (d) complex grid structures made from `GridConnection`s and (e) hollow deformable sphere made from `PFacet`s.

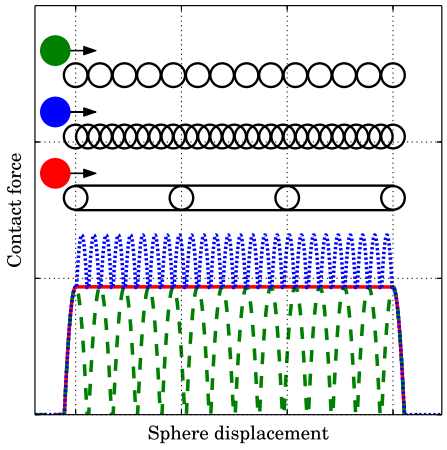

Fig. 5. Concept of rough vs. smooth interaction when moving a sphere along a sphere assembly and a `GridConnection`.

of the AABBs are sorted along all three axes to detect new overlaps. This method is known to be efficient in dynamic problems where, due to temporal coherence, each sort starts from a highly pre-ordered list. In addition, the size of the AABBs includes an extra length (often called "Verlet distance") such that it is not necessary to execute the broad phase detection at every iteration. The sorting algorithm is parallelized by dividing the list of extrema into multiple chunks.

2.4. Interaction models

2.4.1. Framework of classical interaction models

An interaction model defines the interaction force between two bodies as a function of their relative motion and, possibly, of other physical variables. On this aspect `YADE`'s framework is highly generic and flexible, allowing users to extend or introduce new models. As a matter of fact, many different models have been implemented and validated over the years [97], modeling both shear and normal force components, as well as rolling and twisting torques, linear and non-linear responses, elastic and inelastic contacts, bubble interactions, friction, adhesion/-cohesion, viscosity, and damage. Some interaction models include the effects of interstitial fluids, such as capillary or lubrication forces.

The collection of models includes, for dry contacts, linear-elastic contacts with Coulombian friction as in Cundall and Strack [29] (`Law2_ScGeom_FrictPhys_CundallStrack`), the linear spring-dashpot of Walton [106] (`Law2_ScGeom_ViscElPhys_Basic`), as well as more sophisticated variants which cancel attractive forces [92]. Non-linear, Hertzian, and optionally viscous contacts models are available with, namely, the `Law2_ScGeom_MindlinPhys_Mindlin` class, where two formulations are available, for constant [105] and velocity-dependent coefficients of restitution [76], respectively. The viscoelastic Hertz-Mindlin contact model (no-slip solution) was recently benchmarked against analytical solutions and numerical results from other DEM software packages in Dosta et al. [33], alongside most of the active open-source DEM codes, in an effort to enhance transparency within the open-source DEM developers' community. An extension of the Hertz-Mindlin formulation employing the Conical Damage Model [50] is also implemented to capture the accumulation of damage at interparticle contacts via the `Law2_ScGeom_MindlinPhysCDM_HertzMindlinCDM` class [100]. Adhesion can be simulated using either the Derjaguin-Muller-Toporov (DMT) formulation with Hertzian normal forces or a simpler linear-stiffness adhesive model [71] via the `Law2_ScGeom_LudingPhys_Basic` class. Models defining rolling and twisting torques in addition to the contact forces are available in, namely, `Law2_ScGeom6D_CohFrictPhys_CohesionMoment` (linear-elastic torques limited by a cohesive-frictional failure criterion) or `Law2_ScGeom_ImplicitLubricationPhys` (viscous torques). An exhaustive list of the contact models currently available in `YADE` and what physics are included in these models can be found in Table 1.

2.4.2. Cohesive materials with predefined planar discontinuities

Discontinuity surfaces, either isolated or organized in networks, often govern the behavior of cohesive frictional materials (e.g., fractures in rock masses). Inspired by the contact logic proposed by Mas Ivars et al. [74], `YADE` enables to overlay planar surfaces on a set of discrete elements to define such discontinuities using the Jointed Cohesive Frictional Particle Model (JCFPM) implemented in the `Law2_ScGeom_JCFpmPhys_JointedCohesiveFrictionalPM` class [90,51,36]. The planar surfaces, defined as triangulated meshed surfaces, can be created externally with a CAD software (Blender, Gmsh, etc.) and imported into `YADE` in `stl` format or defined internally using the dedicated `utils.facet()` function (Fig. 6(a)). Once incorporated into a discrete elements set, every interaction occurring between elements located on either side of the surfaces is identified (Fig. 6(b)), and their mechanical behavior is eventually adjusted to model, for instance, planes of weakness in otherwise cohesive media (e.g., fractures in rocks) or,

Table 1
Overview of interaction models in YADE.

Contact law - Class name	Non-linear elasticity	Viscosity	Moments*	Shear forces**	Normal plasticity	Adhesion/ cohesion	Additional information and references
Law2_ChCylGeom6D_CohFrictPhys_CohesionMoment			✓	✓	✓	✓	[13]
Law2_CylScGeom6D_CohFrictPhys_CohesionMoment			✓	✓		✓	[13]
Law2_CylScGeom_FrictPhys_CundallStrack				✓			[13]
Law2_GridCoGridCoGeom_FrictPhys_CundallStrack				✓			[41]
Law2_L3Geom_FrictPhys_ElPerfPl				✓			
Law2_L6Geom_FrictPhys_Linear				✓			
Law2_PolyhedraGeom_PolyhedraPhys_Volumetric	✓			✓			[42]
Law2_SCG_KnKsPBPhys_KnKsPBLaw	✓	✓		✓		✓	[11]
Law2_SCG_KnKsPhys_KnKsLaw	✓	✓		✓		✓	[12]
Law2_ScGeom6D_CohFrictPhys_CohesionMoment			✓	✓	✓	✓	[1], shear creep available
Law2_ScGeom6D_InelastCohFrictPhys_CohesionMoment			✓	✓	✓	✓	Plasticity in compression, tension, bending and twisting
Law2_ScGeom_BubblePhys_Bubble							[24]
Law2_ScGeom_CapillaryPhys_Capillarity and CapillarityEngine	✓	✓	✓	✓			Global engines for capillarity [88,37]
Law2_ScGeom_CpmPhys_Cpm				✓	✓	✓	[96]
Law2_ScGeom_FrictPhys_CundallStrack				✓			[87] and many more
Law2_ScGeom_FrictViscoPhys_CundallStrackVisco		✓		✓			[103]
Law2_ScGeom_ImplicitLubricationPhys		✓	✓	✓			Lubrication forces [28]
Law2_ScGeom_JCFpmPhys_JointedCohesiveFrictionalPM				✓	✓	✓	Bonded particles [90,91]
Law2_ScGeom_LudingPhys_Basic		✓		✓		✓	
Law2_ScGeom_MindlinPhysCDM_HertzMindlinCDM	✓			✓			Conical Damage Model [100]
Law2_ScGeom_MindlinPhys_HertzWithLinearShear	✓			✓			
Law2_ScGeom_MindlinPhys_Mindlin	✓	✓	✓	✓		✓	[75]
Law2_ScGeom_MortarPhys_Lourenco				✓	✓	✓	
Law2_ScGeom_PotentialLubricationPhys		✓	✓	✓			Lubrication forces
Law2_ScGeom_VirtualLubricationPhys		✓	✓	✓			Lubrication forces
Law2_ScGeom_ViscElCapPhys_Basic		✓	✓	✓			Capillarity
Law2_ScGeom_ViscElPhys_Basic		✓	✓	✓			
Law2_ScGeom_ViscoFrictPhys_CundallStrack		✓		✓			
Law2_ScGeom_WirePhys_WirePM	✓				✓		[104]
Law2_ScGridCoGeom_CohFrictPhys_CundallStrack			✓	✓	✓	✓	[13,41]
Law2_ScGridCoGeom_FrictPhys_CundallStrack			✓	✓	✓		[13,41]
Law2_TTetraSimpleGeom_NormPhys_Simple	✓			✓			

* Includes rolling friction.
** Includes sliding friction.

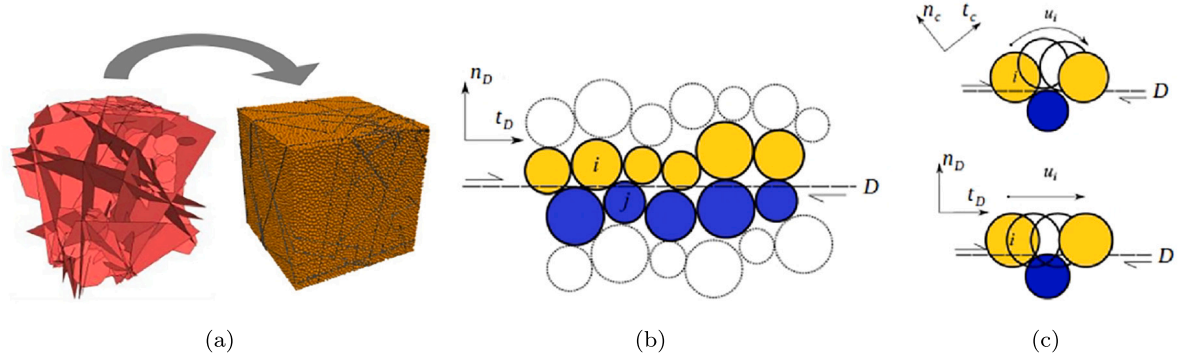

Fig. 6. Integration of discontinuity surfaces into a DEM model: (a) predefined meshed fracture network (stl format) incorporated into a particle assembly, (b) identification of contacts along the discontinuity surface D, and (c) associated contact logic: classical (top), which is according to the contact geometry, or smooth (bottom), according to the discontinuity surface geometry.

conversely, reinforcement sheets (e.g., steel mesh sheets in concrete). The JCFPM contact logic allows the reorientation of all the interactions that intersect the plane surface so that the surface is discretized into a set of contacts that share the same predefined orientation and mechanical properties, ensuring the emergent behavior is independent of the inherent roughness produced by the arrangement of spheres at the interface (Fig. 6(c)). In its current formulation, the approach allows the creation of interacting deformable and breakable blocks in a given volume of material as done, for instance, in grain-based models (GBM) [84].

2.4.3. Capillary models

Simulating surface tension in the pendular regime (small degree of saturation) has been a frequent application of the DEM for at least two decades (e.g. [69,45,68,60] and many subsequent papers). In the pendular regime, wetting liquid forms distinct bridges between particle pairs, and the mechanical contribution from the fluid mixture reduces to an attractive force in a pair interaction [69]. The popularity of this topic is reflected in the variety of capillary force models available in YADE. Several closed-form expressions are available in the ViscElCap* classes [47], while CapillarityEngine and Law2_ScGeom_CapillaryPhys_Capillarity [88,37] are based on direct

numerical resolution of the Laplace-Young equation, linking the shape of the interface between immiscible fluids to the surface tension.

2.4.4. Lubrication models

When particles immersed in a viscous fluid are in relative motion, the fluid mediates remote pair interactions that resist any motion. At close range, the associated viscous forces and torques can be computed using the *lubrication* approximation. The analytical expression of such forces is the cornerstone of the Stokesian dynamics method for rigid spheres [14]. An interaction model that combines solid contacts and lubrication is implemented in YADE's `Law2_ScGeom_ImplicitLubricationPhys`. It uses (i) the lubrication expressions from [44] and [58] to compute the remote viscous effects together with (ii) a (repulsive) linear-frictional model when the particles are in direct contact. When it is assumed that the two types of forces simultaneously contribute to some elastic deformation of the spheres, the equations for solid contact and for lubrication are intimately coupled. Following [73] and later [28], YADE's implementation integrates this coupled viscoelastic problem with an implicit scheme (motion integration remains explicit). It is to be noted that the viscous resistance by lubrication is theoretically infinite when the spheres are in contact. For this reason, many lubrication models introduce a cut-off distance related to roughness to avoid numerical issues. In contrast, the viscoelastic solver of YADE admits zero roughness since elasticity regularizes the problem, and the implicit integration is unconditionally stable.

2.5. Boundary conditions

2.5.1. Rigid and viscoelastic boundaries

YADE provides multiple methods of imposing mechanical boundary conditions. The most straightforward approach is fixing the bodies in place to create a rigid boundary for the particles. Usually, facets or walls are used for such a purpose. Nevertheless, all shapes can be constrained by reducing their degrees of freedom. Imposing stress via a rigid boundary needs a servo-mechanism that controls the position of external walls to achieve and maintain a prescribed state (stress, strain, or porosity). Imposing deformation or stress on the faces of a cuboid containing packed particles is key to simulating element tests. The engine `TriaxialStressController` helps control six rigid boundaries in this context [97]. Each direction can be assigned either a strain rate or a pressure. The pressure is maintained by servo-controlling the position of the boundaries; knowing the instantaneous total stiffness of the boundary-packing interface, the target pressure is reached almost exactly at each time iteration. The engine also returns post-processing data such as the stresses (when strain is controlled), the strains (when pressure is controlled), or the average porosity.

Alternatively, one can constrain the bodies with a viscoelastic boundary condition. In this way, the constrained body is connected to its reference position with a virtual viscoelastic element. In the current implementation, the Burgers model defines the behavior of the viscoelastic element and can be set up to represent simpler ones (Maxwell, Kelvin-Voigt, or purely elastic) [18]. Creating an array of bodies constrained by this condition is a convenient technique for modeling deformable boundaries. It can simulate foundations, particularly in geotechnical issues, pavement design, etc. Fig. 7 shows a sphere over a deformable boundary consisting of an array of cuboids with applied boundary conditions. When the ball falls on the boundary, cuboids pressed by the sphere are displaced and the boundary elements generate a reaction force depending on the amplitude and velocity of the displacement. The remaining cuboids are unaffected since viscoelastic constraints imposed on each body act independently.

2.5.2. Uniform boundaries

YADE supports uniform boundary conditions on domains of arbitrary shapes through continuous boundary surface tracking [83,82]. This surface tracking is accomplished by pairing the Alpha shape surface reconstruction algorithm of the CGAL library [30] with a power-diagram construction algorithm. The former identifies all particles on the boundary of the model, while the latter partitions the boundary surface into polygonal cells associated with each boundary particle. Power diagrams are shown in Fig. 8 for three different model topologies. Knowing both a list of the particles on the model boundary and the surface area vector associated with each boundary particle allows for the application of uniform boundary conditions. The Static Uniform Boundary Condition (SUBC) is simply imposed by applying a force vector on each boundary particle obtained by multiplying the uniform stress tensor by the area vector of that particle. Since the boundary surface is updated continuously, the force vectors are always up-to-date, even when the model is subject to large strains. Likewise, the Kinematic Uniform Boundary Condition (KUBC) is imposed by applying velocity vectors on each boundary particle obtained by multiplying the uniform velocity gradient tensor by the position vector of the centroid of the polygonal cell of the boundary particle. Orthogonal-mixed uniform boundary conditions are also possible by alternating between KUBC and SUBC and applying appropriate (non-conflicting) uniform stress and uniform strain tensors.

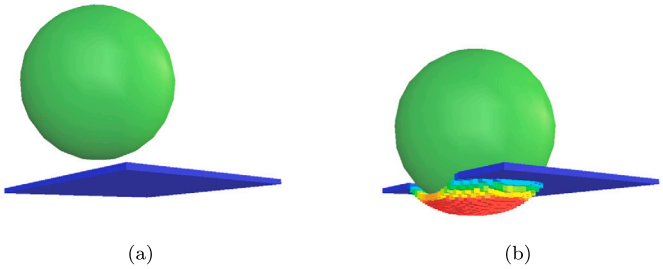

Fig. 7. Modeling a deformable boundary by applying viscoelastic boundary condition to rigid elements (cuboids): (a) initial state and (b) deformed state.

2.5.3. Periodic boundaries

Periodic boundary conditions (PBC) impose homogeneous conditions in an infinite, periodic space. This technique was originally developed for use in MD simulations [2], and it presents the advantage of avoiding boundary effects. An intuitive explanation of PBCs is that particles are modeled inside a unit cell, and if a particle leaves the cell, it re-enters immediately from the other side. Nevertheless, YADE's implementation does not constrain positions to a single periodic cell. Instead, periodicity is realized by the collision detection algorithm generating interactions between bodies and their periodic images [97].

The deformation of space is prescribed via the macro-scale velocity gradient. It is integrated over time, accumulating deformation. The velocity gradient is assigned directly by the user or servo-controlled by some engines to reach some prescribed stress (engines `PeriIsoCompressor` and `PeriTriaxController`).

In Fig. 9, a small number of particles is compacted by shrinking the space isotropically. The 3D rendering of these highlights periodic images (the wired shapes) of some particles from the reference cell. Periodic packings like this one are sometimes used to fill large volumes by placing the same elementary microstructure repeatedly on a regular grid.

2.6. Coupling approaches

2.6.1. PFV-DEM coupling: `FlowEngine` and `TwoPhaseFlowEngine`

YADE provides a set of coupled solvers for single-phase and multiphase flow based on the pore-scale finite volume (PFV) approach (Fig. 10). This technique implements a pore-network decomposition of the pore space in granular systems, and it couples fluid flow to the movements of the individual particles.

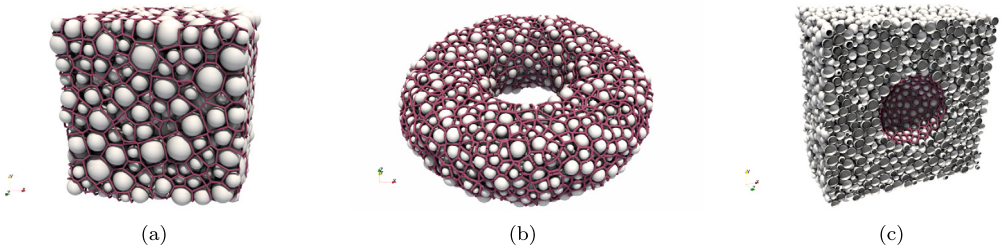

Fig. 8. Power diagrams on discrete element models with different topologies: (a) cubic, (b) toroidal and (c) internal cavity.

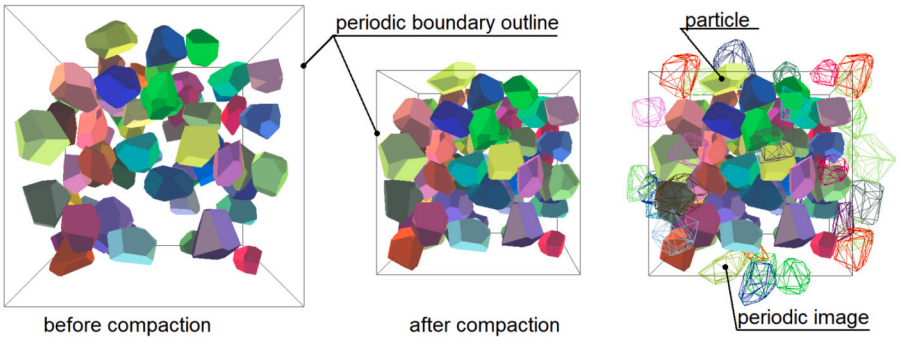

Fig. 9. Compaction of polyhedra packing by shrinking the periodic space.

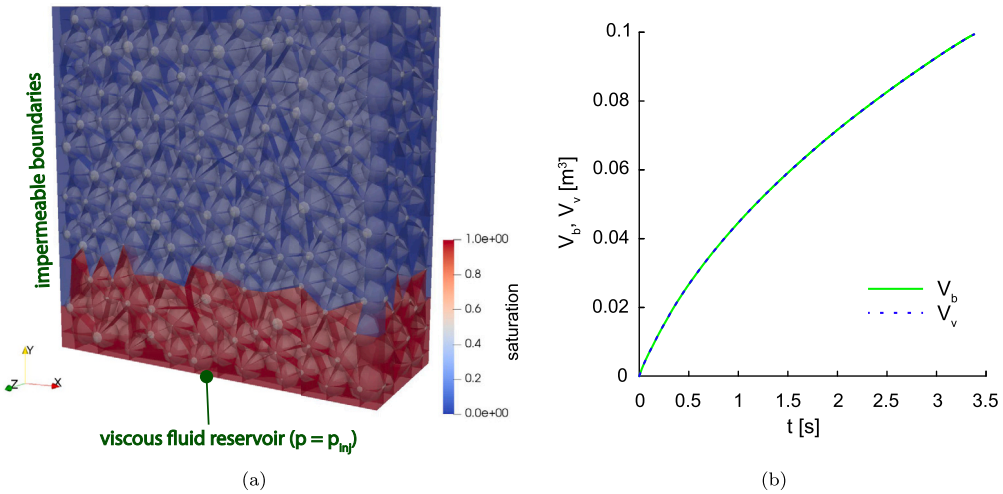

Fig. 10. Regression test of viscous fluid injections into a dry dense sphere packing: (a) PFV-DEM Model and (b) temporal evolution of V_b and V_v. V_b is the difference between the total viscous fluid volume injected from the bottom boundary and fluid volume losses due to air bubble generations, and V_v is the total viscous fluid volume flowing inside dry or partially saturated pores.

The single phase solver is `FlowEngine`. Its first version was dedicated to incompressible Stokes' flow [26,19]. It has been later extended to compressible flow [89], and combined with the JCFPM (see Section 2.4.2) to deal with permeability contrasts in fractured media [80,109].

Two-phase flow with capillarity is solved with a similar pore-space partitioning in the `UnsaturatedEngine` [108] and `TwoPhaseFlowEngine` [102] (the latest differ by the local rule controlling imbibition), enabling the simulation of partially-saturated materials without the restriction to small water content of the pendular bridge models (cf. Section 2.4.3). The algorithms track drainage-imbibition events at the pore scale when two immiscible fluids occupy the pore space (such as water and air). It handles all states from saturated to dry, tracks the formation of liquid clusters of various sizes (down to a pendular bridge), and evaluates the capillary forces on the particles in generic conditions.

2.6.2. GPU acceleration of the `FlowEngine`

`FlowEngine` was accelerated using a variety of methods, including matrix factor reuse, parallel task management, and GPU computing. Combined, these techniques yield an increase of performance by an astonishing 170x as highlighted in [23]. Instructions for exploiting GPU acceleration are detailed in an online tutorial.[6] GPU acceleration can be enabled during compilation using the `CMake` flag `CHOLMOD_GPU`.[7]

2.6.3. ThermalEngine

`ThermalEngine` opens up a wide range of Thermal-Hydraulic-Mechanical (THM) modeling possibilities, such as non-isothermal flow through a conductive sphere packing and geomechanical thermo-poro-

[6] https://yade-dem.org/doc/GPUacceleration.html.
[7] Requires `SuiteSparse` version 2.0.0.

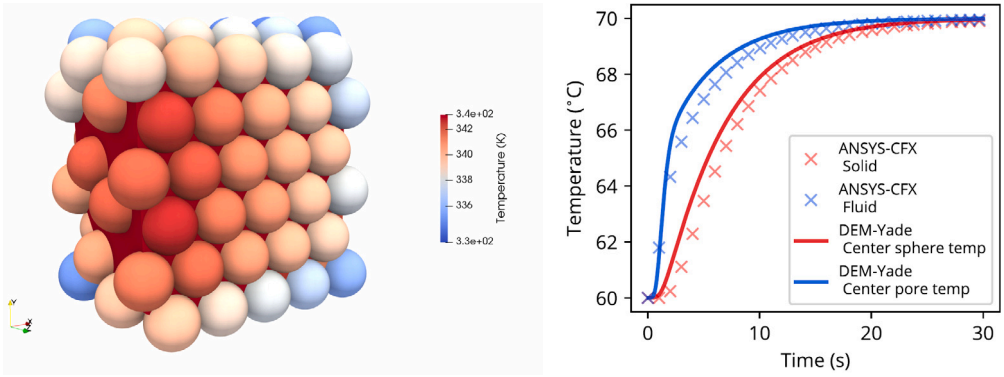

Fig. 11. Example of an advection and conduction numerical simulation of a warm fluid passing through a packing of cold spheres (modified after [20]): (a) temperature distribution in the PFV-DEM model and (b) temporal evolution of the temperature (comparison with predictions from ANSYS-CFX [6]).

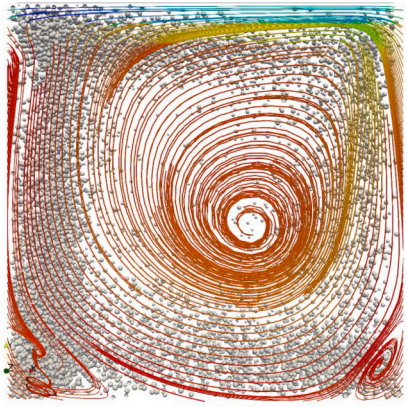

Fig. 12. Example of CFD-DEM simulation with `OpenFOAM`: 50 000 spheres in lid-driven cavity flow.

elasticity. Conductive heat transfer is simulated between particles using a connectivity network, while advective heat transfer is plugged into the fluxes given by `FlowEngine` (Fig. 11). Caulk et al. [20] describes the framework in detail and shows a complex THM experimental validation. Furthermore, `YADE`'s Python wrapping enables users to easily modify, debug, visualize, and extend the existing THM framework.

2.6.4. CFD-DEM coupling with `OpenFOAM`

`YADE` supports coupling with Computational Fluid Dynamics (CFD) solvers using `OpenFOAM`. It implements an Euler-Lagrange scheme as described in [66]. The coupling supports massively parallel execution using a Message Passing Interface (MPI) scheme for both the solid phase (`YADE`) and the fluid (`OpenFOAM`). One of the solution methods implemented uses a point-force coupling approach, whereas the other uses a volume-averaged coupling, which accounts for the particle volume fraction with Gaussian interpolation of field and particle variables, respectively. The current framework enables the future inclusion of fully resolved flow description around the particle by the Immersed Boundary Method (IBM) and full parallel coupling between `YADE-MPI` and `OpenFOAM`. An example simulation is shown in Fig. 12.

2.6.5. Hierarchical multiscale coupling

A multiscale framework is available to model boundary value problems (BVPs) with a combined FEM×DEM approach. It relies on the FEM solver Escript[8] [86] to produce a hierarchical FEM×DEM coupling [48,49]. FEM is used to discretize the large-scale problem, while DEM is used at each Gauss point to provide the local incremental response of a simulated material. This approach, pioneered by [78], eliminates the need for phenomenological assumptions at the continuum scale.

With a similar logic, the DEM can be embedded into the Material Point Method (MPM) [70]. Such an MPM×DEM coupling using `YADE` and *CB-Geo* for the MPM[9] is presented in [38].

2.6.6. Multidomain coupling

Due to its flexible design, `YADE` can easily be coupled with other numerical methods or packages. The most commonly implemented coupling approach is surface or interface coupling. Stránský [99] presented a general framework that allows coupling of `YADE` with the open-source framework OOFEM [81]. It was successfully applied to model the interaction between ballast and sleepers [98]. Thereby, the problem is physically split into two domains and solved separately in each code. Contact forces and boundary surfaces are shared and updated as the simulation progresses. More recently, `YADE` has been coupled with the transient formulation of the Boundary Element Method (BEM) [9,10]. This interface coupling allows the accurate representation of infinite domains. Several DEM regions can be embedded in the BEM domain. The latter is assumed to behave as linear elastic and is generally used to represent the far-field. The relevant BEM matrices can be pre-computed, and `YADE` handles the simulation. Another method that has been coupled with `YADE` is Peridynamics [95]. A generalized force coupling approach with a staggered time integration scheme was developed that allowed to directly implement Peridynamics in `YADE` [52].

3. General overview

3.1. User interface and live processing

`YADE`'s user interface comprises a graphical user interface (GUI) with several windows and an interactive IPython shell (Fig. 13). The GUI includes a Qt interface with basic functionalities (save/load/run/pause) and variable explorer, real-time 3D-rendering of the scene using *QGLViewer*, and a plotting window based on the python library `matplotlib`. The GUI exposes all variables for all bodies, engines, and interactions in a simulation, with read-write access in most cases and the documentation associated with these variables.

Most `YADE` users do not manipulate any C++ code. This is only necessary for the sake of performance when truly new models and features need to be implemented. Instead, the Python interpreter is the primary interface for most advanced use. It lets users combine the basic components needed for a specific simulation (the "components" being implemented in C++, typically) in a Python script. This includes custom

[8] https://github.com/esys-escript/esys-escript.github.io.

[9] https://github.com/cb-geo/mpm.

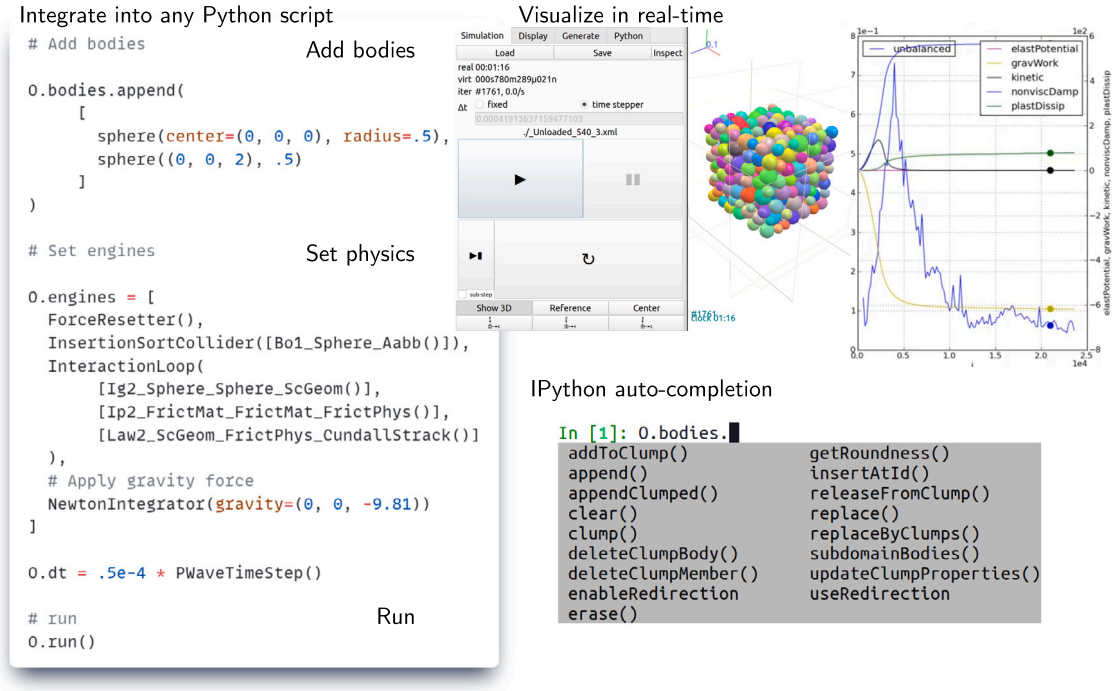

Fig. 13. Overview of the YADE user interface. Top left: the simplest python script for a complete simulation. Top right: visualization capabilities. Bottom right: tab-completion lists all available functions to use.

engine lists, dynamic time-step control, body/interaction data extraction, event-based boundary conditions, transitions between multiple stages of a simulation, live plotting, and possibly even post-processing. A simulation job described in a script is executed with the command `yade script.py`.

The interactivity of the IPython shell makes it well-suited for exploring variables and performing various operations at run time, while it also provides in-line documentation. In addition, it is possible to import YADE as a Python module and use it in another software or a custom Python program. For users running parametric sweeps with a large number of parameters, YADE offers `yade-batch` for launching and monitoring the simulations. The batch mode uses a table of parameters as input in addition to the script to be executed.

YADE has a ready-to-use module based on the `matplotlib` library for both plotting live data and saving plot data (shown in Fig. 13) into optionally compressed files (yade.plot module). In addition to the possibility of manipulating GUI views within YADE, for more advanced graphical illustrations, simulation data can readily be exported in a VTK format for post-processing and 3D rendering of simulation results with Paraview.[10]

3.2. Post-processing

3.2.1. From discrete to continuum results

Some of the processed data that can be visualized in Paraview is generated by the `TessellationWrapper`, which defines continuum scale strain and stress from DEM results. This class handles the triangulation of spheres in a scene, builds a tessellation on request, and gives access to computed quantities, such as volume, porosity, micro-strain, and micro-stress associated with individual spheres (Figs. 14(a) and 14(b)), as explained in [19]. This feature can also be used to process experimental data from X-ray computed tomography [3] (see Fig. 14(c)). Figs. 14(d) and 14(e) show results from a plate load test simulation. The strain field is directly the output of `TessellationWrapper`, the stress field is interpolated by triangulating the per-sphere data.

3.2.2. Acoustic emissions module

YADE can simulate acoustic emission (AE) events in cohesive discrete element assemblies. Following a strain energy approach, the AE module, currently implemented in the JCFPM, tracks the release of strain energy associated with interparticle bond breakages to estimate AE properties. The module enables the clustering of bonds that break close to one another which enables the realistic quantification of AE magnitude during material failure. YADE users can easily post-process and visualize the AE events using the existing data export tools in YADE (Fig. 15). Caulk [21] demonstrates the new functionality by running three-point bending tests on rock samples containing various levels of heterogeneity to ultimately show the spatial distributions and magnitudes of AE during pre- and post-failure.

3.3. Installation

Starting from the 0.60.0 version released in 2011, major releases of YADE have been systematically proposed in all Debian, Ubuntu, and other DEB-based distributions as a pre-compiled package. While the installation of YADE with *apt* (`apt install yade`) installs a stable version corresponding to the release date of one specific Linux system, binary packages reflecting the development branch daily are available in a separate personal package archive (PPA) providing `yadedaily`.[11]

Recompilation on the host system is only necessary when one needs to modify the code or implement new algorithms. For further details, the YADE documentation [97] provides extensive instructions about package dependencies and compilation on various Linux versions.

3.4. Code development, continuous integration, deployment and packaging

YADE development occurs on a dedicated `gitlab.com` repository, where a master branch evolves through reviewed merge requests from

[10] https://www.paraview.org.

[11] https://yade-dem.org/doc/installation.html#packages.

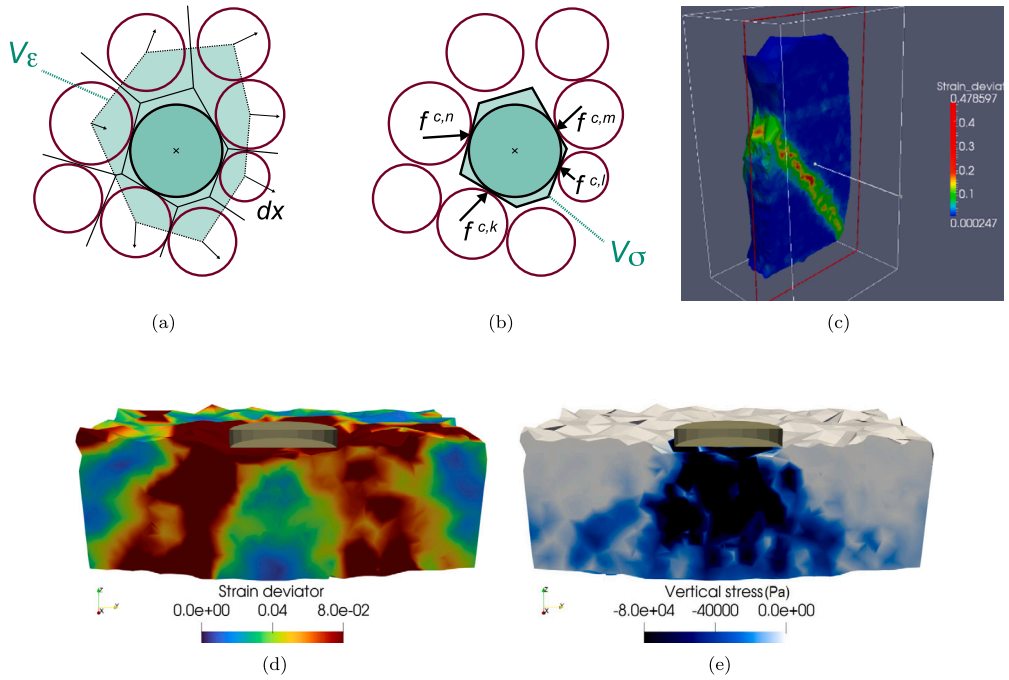

Fig. 14. From discrete to continuum results: (a) particle-centered domain for the definition of micro-strain, (b) particle-centered domain for the definition of micro-stress, (c) example of micro-strain visualized with Paraview (analyzing X-ray CT data on sand [3]), (d) example of micro-strain visualized with Paraview (results of plate load test simulation), and (e) example of micro-stress.

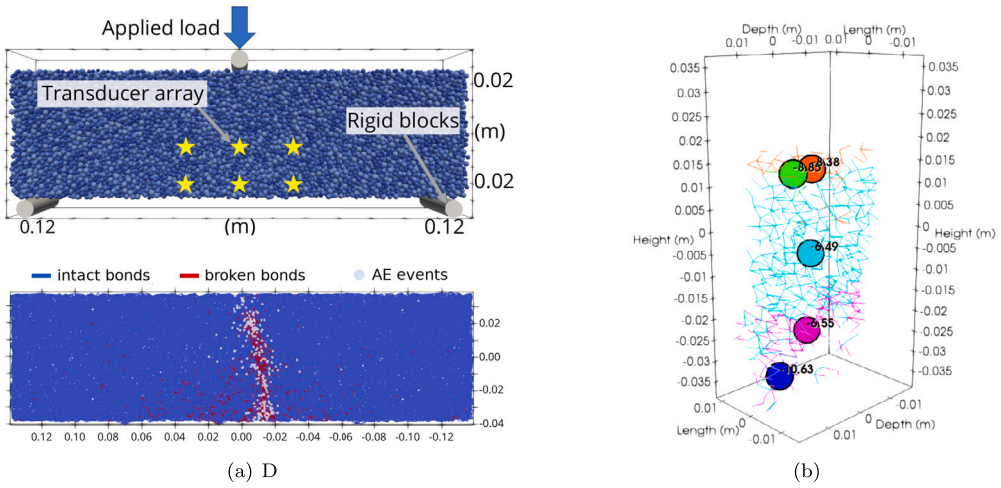

Fig. 15. Example of acoustic emission (AE) processing along the tensile fracture developing in a three-point bending test simulation (modified after [21]): (a) DEM model and (b) clustering of AE events. The AE events are clustered and labeled with their magnitudes.

developer branches, and issues are reported. To ensure a smooth, ever-going development, a Continuous Integration (CI) pipeline configured for `YADE` performs automated builds, tests, and checks for every merge request every two days on the master branch. The builds are done inside docker images of all the major releases of Debian (from Stretch to Bookworm) and Ubuntu (from 16.04 to 22.04) as well as ArchLinux and OpenSuse. The CentOS build is currently in progress. Apart from the default amd64 architecture, the automated builds are also performed inside the qemu emulator of other architectures: arm64, ppc64el, and s390x. In total, there are about 100 jobs with different Linux versions and different combinations of compilation options. CI facilitates early detection of any kind of regression or bug introduced in packages `YADE` depends on, and accurate reporting to the third-party developers.

The tests come in three different variants. First, basic tests (`yade -test`) consider the internals of `YADE`, such as serialization, mathematical functions for all numerical precisions (number of decimal places used in computations [65]) or other basic functionalities of the software. Second, more elaborate checks (`yade -check` or `yade -checkall`) run short simulations using different physical models, which correspond to realistic use cases, to check the reproducibility of results across changes to the source code. Finally, GUI tests run simple simulations and compare their screenshots with the reference screenshots to identify possible issues in the GUI.

After an accepted merge request, the Gitlab pipelines build newer `yadedaily` packages for installation/update by end-users. DEB yadedaily packages are built for the last four stable Debian versions (beginning from Debian 9 Stretch) and for the last 4 Ubuntu LTS versions (beginning from Ubuntu 16.04 Xenial). The yade-dem.org server periodically pulls the last successfully built DEB-artefacts from the `YADE` Gitlab projects, signs them with GPG-key, and puts them into the DEB-repository with the aptly package-management system. The new

Table 2
Options to build YADE from source.

Build option	Description	Reference
DISABLE_ALL	Disable all options (OFF by default).	
ENABLE_ASAN	AddressSanitizer build, please see documentation, useful for fixing memory leaks (OFF by default).	
ENABLE_CGAL	Enable Triangulation using (CGAL) (ON by default).	[56]
ENABLE_COMPLEX_MP	Requires boost >= 1.71: use boost::multiprecision for ComplexHP: (1) complex128 (2) mpc_complex (3) complex_adaptor (ON by default); Otherwise use std::complex<...>.	
ENABLE_DEFORM	Enable constant volume deformation engine (OFF by default).	[53]
ENABLE_FAST_NATIVE	Use maximum possible optimization, compiled code runs only on the same processor type; speedup about 2% with gcc compiler, and above 5% with clang compiler, which requires ENABLE_USEFUL_ERRORS=OFF (OFF by default).	[65]
ENABLE_FEMLIKE	Enable meshed solids (FEM-like) (ON by default).	
ENABLE_GL2PS	Enable postscript export of OpenGL rendering (ON by default).	
ENABLE_GTS	Enable features using the GNU Triangulated Surface (GTS) library (ON by default).	
ENABLE_GUI	Enable Graphical User Interface (ON by default).	
ENABLE_LBMFLOW	Enable Lattice Boltzmann Method for fluid flow (ON by default).	
ENABLE_LINSOLV	Use efficient linear solvers based on SuiteSparse, OpenBLAS and Metis (ON by default).	[23]
ENABLE_LIQMIGRATION	Enable liquid migration (OFF by default).	[72]
ENABLE_LOGGER	Use powerful boost::log library for logging (ON by default).	
ENABLE_LS_DEM	Enable level-set shape description (ON by default).	[35,54]
ENABLE_MASK_ARBITRARY	Enable arbitrarily large size of group mask (OFF by default)	
ENABLE_MPFR	Use https://www.mpfr.org/ library, can be used for higher precision calculations or for CGAL exact predicates (OFF by default).	[65]
ENABLE_MPI	Enable MPI communications in YADE. Used for distributed memory parallelization and YADE-OpenFOAM coupling.	[25]
ENABLE_MULTI_REAL_HP	Allow using twice, quadruple or higher precisions: RealHP<N> for N$\in\{1,2,3,4,8,10,20\}$. It is not advertised in yade.config.features because of frequent confusion with output of yade.math.usesHP() as described in https://gitlab.com/yade-dev/trunk/-/issues/247. Use yade.math.getRealHPCppDigits10() and yade.math.getRealHPPythonDigits10() to check how many decimal digits each of available precisions has. (ON by default).	[65]
ENABLE_OAR	Generate a script for oar-based task scheduler (OFF by default)	
ENABLE_OPENMP	Enable OpenMP parallelization (ON by default)	
ENABLE_PARTIALSAT	Enable the partially saturated clay engine, under construction (OFF by default).	[22]
ENABLE_PFVFLOW	Enable the FlowEngine (ON by default).	[26,19]
ENABLE_POTENTIAL_BLOCKS	Enable potential blocks option (ON by default).	[11]
ENABLE_POTENTIAL_PARTICLES	Enable potential particles option (ON by default).	[12]
ENABLE_PROFILING	Enable profiling, e.g. show some more metrics, which can help in locating bottlenecks in the code (OFF by default)	
ENABLE_SPH	Enable Smoothed Particle Hydrodynamics (OFF by default)	
ENABLE_THERMAL	Enable thermal engine (ON by default, experimental).	[20]
ENABLE_TWOPHASEFLOW	Enable TwoPhaseFlowEngine (ON by default)	[108,101]
ENABLE_USEFUL_ERRORS	Enable useful compiler errors, which help a lot in error-free development (ON by default).	
ENABLE_VTK	Enable VTK export option (ON by default).	

documentation is also built and uploaded to yade-dem.org (*html* pages, *pdf* and *epub* outputs).

Stable versions are typically released approximately once a year, sourced from the master branch. For the past eight years, these versions have followed a naming convention of YYYY.MMa-z. The letter suffix is used to indicate bug-fix releases.

3.5. Build system

The current build system is CMake, which automatically detects the build platform, searches for dependencies, performs all the necessary system checks and handles custom compilation options: debug build, optional features such as GUI, parallelization, specific solvers, etc. (see Table 2 for the complete list of build options and Fig. 16 for YADE modules and simplified dependency tree).

CMake generates various build files, including the Makefile, which is essential for compiling the source code. Additionally, CMake monitors changes in the source code, automatically regenerating the build files when necessary, and installs the binary files in the user defined location. Furthermore, CMake simplifies the generation of documentation by replacing it with a simple make doc command and concealing the complexity of the documentation generation process. The documentation is written in reStructuredText (reST) markup language and built with Sphinx [15].

4. Framework features

4.1. Serialization

YADE features robust serialization routines. Every variable and attribute in every class is wrapped and registered, which allows to:

- save and load a whole simulation to binary files (or .xml files, which is useful when porting a saved file to a different YADE version).
- browse the entire simulation and inspect the registered member variables in a QT GUI.
- read and modify the contents of the simulation from the IPython frontend by accessing the registered variables.

Additionally, upon registering variables to these three targets, each variable is accompanied by a doc help string, which is readily available from the IPython shell as mentioned in Section 3.1. These doc help strings are also used when building the YADE reference manual [97]. They are a significant part of the whole YADE manual (Chapters 2.3 and 2.4, about 596 pages therein out of a total of 906 pages).

4.2. OpenMP and MPI parallelizations

4.2.1. Shared memory

Since the work of Ŝmilauer [96], parallel execution of YADE through shared memory is supported using the OpenMP library. Typical par-

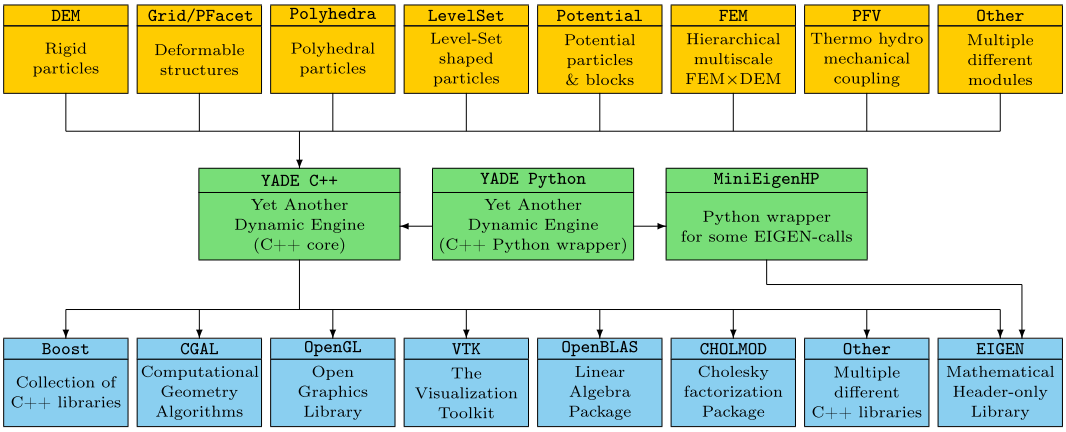

Fig. 16. Simplified dependency tree showing the interaction among YADE modules (yellow), core parts of the framework (green) and external dependencies (blue).

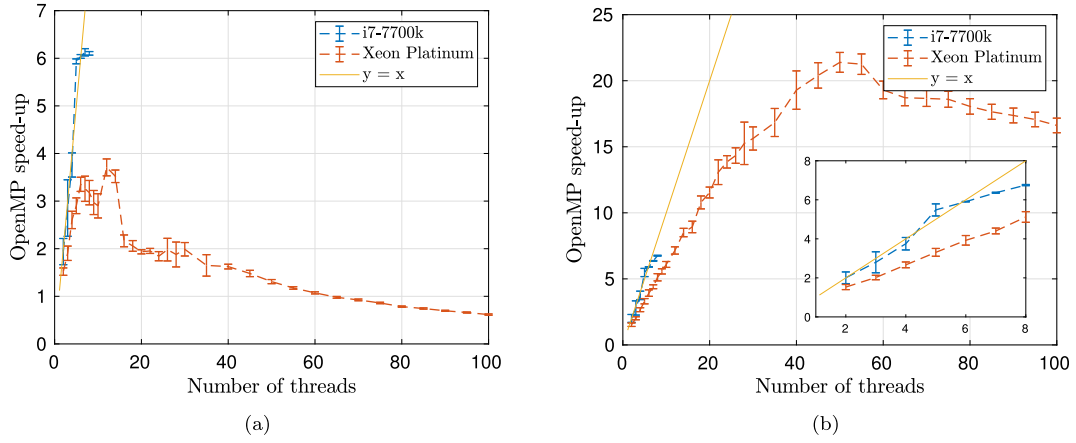

Fig. 17. OpenMP speed up for a quasi-static axisymmetric compression of a packing including 8 000 grains, using two shape descriptions of different complexities, on different machines with different Intel processors (modified after [34]): (a) simple spheres and (b) LevelSet shapes, see Section 2.1.

allel sections involve splitting the loop over bodies when integrating motion equations or over contacts when computing and summing the contact forces for each body over several threads. In many cases, the shared memory parallelization was achieved by the mere insertion of a #pragma directive in the C++ source code before an existing, purely sequential loop. In some cases, additional synchronization operations were necessary to ensure the thread safety of a naive multi-threaded sum. The accumulation of interaction forces on bodies is such a case. This parallelization technique is deep in the source code and, therefore, invisible to the user. There is no need to think about it when formulating a simulation script. There is often a limit to how far additional cores increase performance, but this limit depends greatly on the type of simulation and hardware. Fig. 17 shows the performance increase versus the number of threads, where the performance is measured by Cundall's number: the number of discrete elements times the number of time-steps divided by the wall clock time. The timing data in Fig. 17 and Fig. 18 include a small overhead from the timing module of YADE and have been obtained either on a 4-core (8 threads) Intel i7-7700 3.60 GHz (0.8–4.2 GHz) processor with 8 MB of cache memory and 2.4 GHz RAM or on two Intel Xeon Platinum 8270 2.7 GHz (4.0 GHz max) processors with 26 cores each and 36 MB of cache memory, i.e. a total of 52 cores and 104 threads, together with a 2.9 GHz RAM. With simple spheres, the best performance is obtained with 6 cores, slightly above 10^6 particles×iterations/s. This is approximately four times faster than single-threaded execution (Fig. 18). Using a more complex shape model lowers the overall performance but makes using more threads relatively more efficient.

4.2.2. Distributed memory

Distributed memory parallelism through domain decomposition has been implemented more recently to take advantage of computing clusters using the MPI protocol [25]. It is available in the *mpy* module.

Implementing a distributed memory approach is sometimes considered an intrusive technique in terms of source code (which is a problem with a large multi-author code base like YADE), yet YADE's implementation did not require substantial changes in the C++ source code. Instead, the implementation was started in Python (*mpi4py* module [31]), such that one master Python process can control multiple running instances of YADE and communicate data and instructions to them interactively. Each instance is essentially MPI-blind internally. Only a few critical functions have been translated from Python to C++ at a later stage. That was the case, namely, for functions involving multiple loops on bodies and interactions, needing to form a large message. In such cases, because of the loops and pickling overhead, the C++ versions outperformed their Python counterparts significantly. Nevertheless, the core of the code remained independent of those utility functions. Each running instance can exploit shared memory parallelism inside a node, enabling hybrid parallelism (OpenMP + MPI).

In practice, each YADE instance handles a subset of the particles, i.e. a "subdomain". These subdomains are fundamentally Lagrangian: they contain a subset of bodies regardless of their positions in space. The interactions between subdomains are detected by the same collision detection algorithm as used for conventional bodies, based on the AABB of the subset (see Fig. 19). When two subdomains' AABB's intersect, the corresponding instances communicate to each other the positions

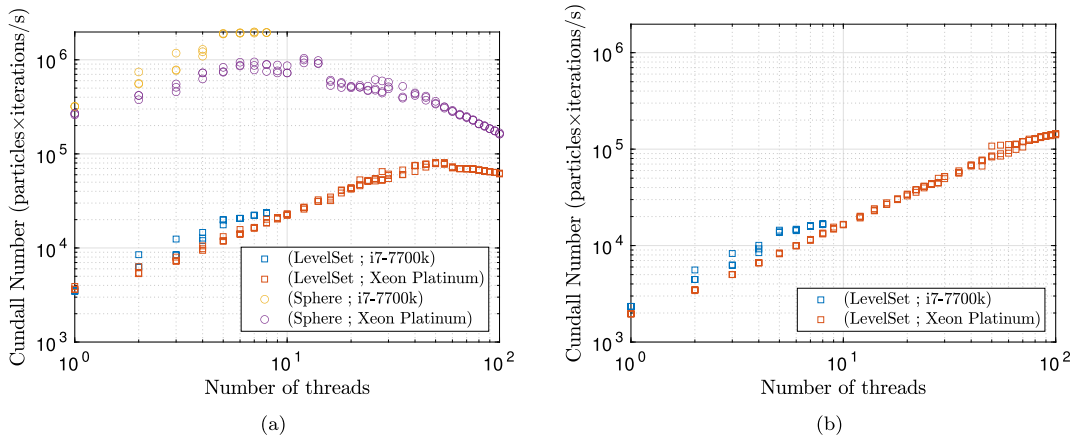

Fig. 18. Cundall Number obtained during OpenMP runs of two different simulations on different machines with different Intel processors: (a) quasi-static axisymmetric compression with two possible shape descriptors for 8 000 grains (after Fig. 17 and data from [34]) and (b) dynamic collapse of 1 000 `LevelSet`-shaped particles (after data from [35]).

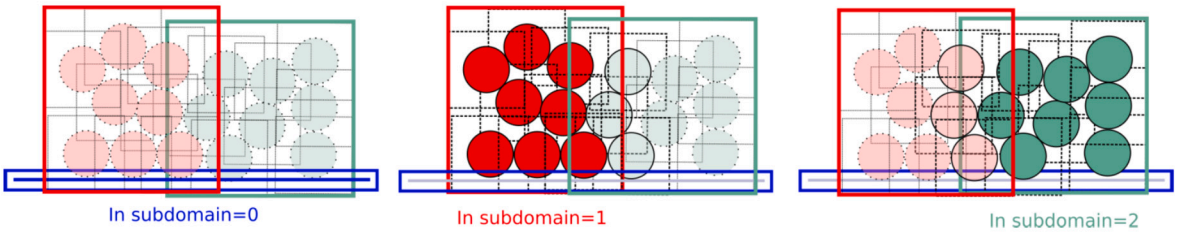

Fig. 19. Subdomains and their AABBs as defined by domain decomposition in the *mpy* (MPI) module. Subdomain = 0 owns the floor, and it passes the bounding boxes of the other subdomains to handle collision detection between subdomains. Every other subdomain controls a subset of the bodies (strong colors), and it collects the positions of bodies from other subdomains (light colors) if they are in the overlapping region to compute interactions with them.

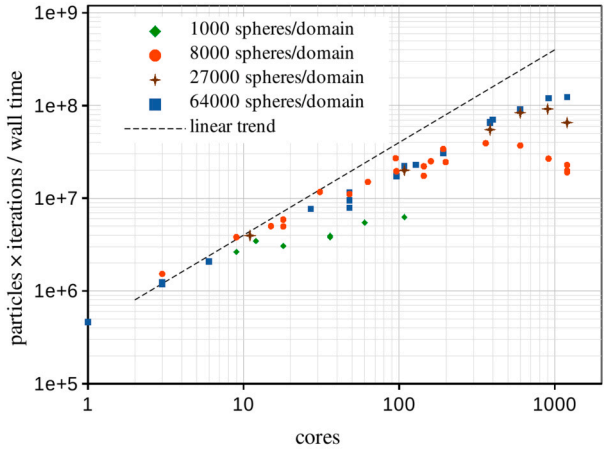

Fig. 20. `YADE`'s throughput with *mpy*'s domain decomposition and a constant number of spheres per core (weak scaling). The dotted line indicates a linear trend. The maximum number of spheres (1 200 cores with 64×10^3 spheres) is 76.8×10^6.

and velocities of particles in the intersection. They also trade bodies with one another to maximize compactness and minimize intersections in future communications and, eventually, detach when possible. The detection of collisions between subdomains, the main centralized operation, is executed in a master process which may also handle special bodies such as the floor in Fig. 19. That floor is receiving summary forces and torques from every subdomain. The master process does not know the content of any subdomains. As seen in Fig. 20, the performance in a heap problem is only slightly sublinear up to 1 200 cores.

4.3. Available floating-point precisions

`YADE` features high-precision calculations for floating point numbers [65]. It is implemented in such a way that the `YADE`-default `Real` floating type may stand for different working precisions, depending on compilation options. By default, Debian and Ubuntu packages are provided for the `Real` type to be: `double`, `long double`, `float128` and `mpfr150` through different executables, respectively `yade`, `yade-longdouble`, `yade-float128` or `yade-mpfr150`. They feature calculation precisions of 15, 18, 33, and 150 digits.

Even when working with the `Real` type to be the regular `double` type, it is still possible to use the higher precision types in the sensitive parts of the C++ algorithms: the `RealHP<N>` for $N \in \{1, 2, 3, 4, 8, 10, 20\}$ is a convenient C++ typedef which provides a floating point type with N times higher precision than the base precision `RealHP<1>` which is the regular `Real`. Additionally, when working in Python the `RealHP<2>` type is accessible inside the `yade.math.HP2` module. Other precision types can also be exported to Python by changing the `YADE_MINIEIGEN_HP` inside the `RealHPConfig.hpp`[12] file, if necessary.

5. Summary and future outlook

This paper describes the current state of the `YADE` simulation framework, focusing more specifically on aspects that have made it both a powerful and popular tool for the modeling and simulation of particulate systems. `YADE` currently includes modeling features that go far beyond the earlier version dedicated to the simulation of spherical discrete elements. `YADE` can now deal with multiple particle shapes and

[12] https://gitlab.com/yade-dev/trunk/-/blob/2023.02a/lib/high-precision/RealHPConfig.hpp#L18.

deformable particles, as well as with deformable structures. Its extended library of contact models and boundary conditions, combined with its capabilities in terms of multiscale and multiphysics couplings as well as its recent parallelization strategy, make it suitable for tackling boundary value problems in many fields of science and technology.

Moving forward, YADE has been built with the desire to constantly evolve thanks to the feedback and input of its users and contributors. Its future is thus not entirely written in stone. Nonetheless, current efforts focus on enhancing or extending some of its newest features, such as:

- consolidating the MPI strategy to support applications involving millions of particles,
- extending the thermo-hydro-mechanical coupling scheme to fractured media,
- optimizing and parallelizing the coupling with other methods such as Peridynamics and BEM,
- incorporating additional features to existing contact models for specific processes, such as fatigue in brittle materials, additive manufacturing, bio-cementation etc., and
- optimizing the current contact detection scheme for polyhedra, level-sets, and possibly other complex shapes.

CRediT authorship contribution statement

Vasileios Angelidakis: Writing – original draft. **Katia Boschi:** Writing – original draft. **Karol Brzeziński:** Writing – original draft. **Robert A. Caulk:** Writing – original draft. **Bruno Chareyre:** Writing – original draft. **Carlos Andrés del Valle:** Writing – original draft. **Jérôme Duriez:** Writing – original draft. **Anton Gladky:** Writing – original draft. **Dingeman L.H. van der Haven:** Writing – original draft. **Janek Kozicki:** Writing – original draft. **Gerald Pekmezi:** Writing – original draft. **Luc Scholtès:** Writing – original draft. **Klaus Thoeni:** Writing – original draft.

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 data created during this research is openly available from GitLab at https://gitlab.com/yade-dev/trunk.

Acknowledgements

YADE has been developed and tested by many people over the years, and the code would not exist without the support of all these people who have contributed to making it what it is today.

The authors would like to acknowledge support from the following funding sources:

- the Engineering and Physical Sciences Research Council (grant number EP/R511584/1) via the project “OCULAR: Automated Acquisition & Classification of Particles”;
- COST action CA18222 “Attosecond Chemistry”;
- the Australian Coal Association Research Programme (ACARP C19026) and The Australian Research Council (ARC DP190102407);
- the Institute for Mineral Processing Machines and Recycling Systems Technology at TU Bergakademie Freiberg;
- the French Sud region during the LS-ENROC project;
- Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) programme;
- FP7 Grant “DIGA” (HPRN-CT-2002-00220);
- FP7-people “Mumolade” (289911);
- Grenoble INP - UGA Graduate schools of Engineering and Management (“Emergence” funding and PhD grant Catalano); and
- the French C2D2 program (“Hydrofond” project).

In addition, the authors thank the Division of Theoretical Physics and Quantum Information, Faculty of Applied Physics and Mathematics, at Gdańsk University of Technology, and the support unit Grenoble Alpes Research Infrastructure for Scientific Computing and Data (UAR 3758 GRICAD) for hosting the GitLab Continuous Integration (CI) pipeline for YADE.

References

[1] R. Aboul Hosn, L. Sibille, N. Benahmed, B. Chareyre, Discrete numerical modeling of loose soil with spherical particles and interparticle rolling friction, Granul. Matter 19 (2017) 1–12, https://doi.org/10.1007/s10035-016-0687-0.
[2] B.J. Alder, T.E. Wainwright, et al., Phase transition for a hard sphere system, J. Chem. Phys. 27 (1957) 1208, https://doi.org/10.1063/1.1743957.
[3] E. Andò, S.A. Hall, G. Viggiani, J. Desrues, P. Bésuelle, Grain-scale experimental investigation of localised deformation in sand: a discrete particle tracking approach, Acta Geotech. 7 (2012) 1–13, https://doi.org/10.1007/s11440-011-0151-6.
[4] D. André, J. Charles, I. Iordanoff, J. Néauport, The GranOO workbench, a new tool for developing discrete element simulations, and its application to tribological problems, Adv. Eng. Softw. 74 (2014) 40–48, https://doi.org/10.1016/j.advengsoft.2014.04.003.
[5] V. Angelidakis, S. Nadimi, M. Otsubo, S. Utili, CLUMP: a code library to generate universal multi-sphere particles, SoftwareX 15 (2021) 100735, https://doi.org/10.1016/j.softx.2021.100735.
[6] ANSYS Inc., ANSYS CFX, https://www.ansys.com/products/fluids/ansys-cfx.
[7] M. Asadi, A. Mahboubi, K. Thoeni, Discrete modeling of sand-tire mixture considering grain-scale deformability, Granul. Matter 20 (2018) 18, https://doi.org/10.1007/s10035-018-0791-4.
[8] M. Asadi, A. Mahboubi, K. Thoeni, Towards more realistic modelling of sand-rubber mixtures considering shape, deformability and micro-mechanics, Can. Geotech. J. 60 (2023), https://doi.org/10.1139/cgj-2021-0710.
[9] G. Barros, A. Pereira, J. Rojek, J. Carter, K. Thoeni, Efficient multi-scale staggered coupling of discrete and boundary element methods for dynamic problems, Comput. Methods Appl. Mech. Eng. 415 (2023) 116227, https://doi.org/10.1016/j.cma.2023.116227.
[10] G. Barros, V. Sapucaia, P. Hartmann, A. Pereira, J. Rojek, K. Thoeni, A novel BEM-DEM coupling in the time domain for simulating dynamic problems in continuous and discontinuous media, Comput. Methods Appl. Mech. Eng. 410 (2023) 116040, https://doi.org/10.1016/j.cma.2023.116040.
[11] C. Boon, G. Houlsby, S. Utili, A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method, Comput. Geotech. 44 (2012) 73–82, https://doi.org/10.1016/j.compgeo.2012.03.012.
[12] C. Boon, G. Houlsby, S. Utili, A new contact detection algorithm for three-dimensional non-spherical particles, Powder Technol. 248 (2013) 94–102, https://doi.org/10.1016/j.powtec.2012.12.040.
[13] F. Bourrier, F. Kneib, B. Chareyre, T. Fourcaud, Discrete modeling of granular soils reinforcement by plant roots, Ecol. Eng. (2013), https://doi.org/10.1016/j.ecoleng.2013.05.002.
[14] J.F. Brady, G. Bossis, Stokesian dynamics, Annu. Rev. Fluid Mech. 20 (1988) 111–157, https://doi.org/10.1146/annurev.fl.20.010188.000551.
[15] G. Brandl, Sphinx documentation, http://sphinx-doc.org/sphinx.pdf, 2021.
[16] K. Brzeziński, P. Ciężkowski, S. Bąk, Tricking the fractal nature of granular materials subjected to crushing, Powder Technol. 425 (2023) 118601, https://doi.org/10.1016/j.powtec.2023.118601.
[17] K. Brzeziński, A. Gladky, Clump breakage algorithm for DEM simulation of crushable aggregates, Tribol. Int. 173 (2022) 107661, https://doi.org/10.1016/j.triboint.2022.107661.
[18] K. Brzeziński, A. Zbiciak, A. Gladky, Implementation of a viscoelastic boundary condition to Yade–open source DEM software, J. Theor. Appl. Mech. 61 (2023) 355–364, https://doi.org/10.15632/jtam-pl/163053.
[19] E. Catalano, B. Chareyre, E. Barthélemy, Pore-scale modeling of fluid-particles interaction and emerging poromechanical effects, Int. J. Numer. Anal. Methods Geomech. 38 (2014) 51–71, https://doi.org/10.1002/nag.2198.
[20] R. Caulk, L. Scholtès, M. Krzaczek, B. Chareyre, A pore-scale thermo–hydro-mechanical model for particulate systems, Comput. Methods Appl. Mech. Eng. 372 (2020) 113292, https://doi.org/10.1016/j.cma.2020.113292.
[21] R.A. Caulk, Modeling acoustic emissions in heterogeneous rocks during tensile fracture with the discrete element method, Open Geomech. 2 (2020), https://doi.org/10.5802/ogeo.5.
[22] R.A. Caulk, Modélisation micro-macro par la méthode des éléments discrets (DEM) du comportement à long terme des scellements de puits sous sollicitation hydraulique-gaz, Ph.D. thesis, Université Grenoble Alpes, 2022, http://www.theses.fr/2022GRALI003, thèse de doctorat dirigée par Chareyre, Bruno Matériaux, mécanique, génie civil, électrochimie Université Grenoble Alpes 2022.

[23] R.A. Caulk, E. Catalano, B. Chareyre, Accelerating Yade's poromechanical coupling with matrix factorization reuse, parallel task management, and GPU computing, Comput. Phys. Commun. 248 (2020) 106991, https://doi.org/10.1016/j.cpc.2019.106991.

[24] D. Chan, E. Klaseboer, R. Manica, Film drainage and coalescence between deformable drops and bubbles, Soft Matter 7 (2011) 2235–2264, https://doi.org/10.1039/c0sm00812e.

[25] B. Chareyre, W. Chevremont, T. Guntz, F. Kneib, J. Pourroy, Calcul distribué mpi pour la dynamique de systèmes particulaires, https://www.yade-dem.org/publi/YadeTechnicalArchive/YadeMPIhackathon.pdf.

[26] B. Chareyre, A. Cortis, E. Catalano, E. Barthélemy, Pore-scale modeling of viscous flow and induced forces in dense sphere packings, Transp. Porous Media 94 (2012) 595–615, https://doi.org/10.1007/s11242-012-0057-2.

[27] H. Cheng, H. Yamamoto, K. Thoeni, Y. Wu, An analytical solution for geotextile-wrapped soil based on insights from DEM analysis, Geotext. Geomembr. 45 (2017) 361–376, https://doi.org/10.1016/j.geotexmem.2017.05.001.

[28] W. Chèvremont, H. Bodiguel, B. Chareyre, Lubricated contact model for numerical simulations of suspensions, Powder Technol. 372 (2020) 600–610, https://doi.org/10.1016/j.powtec.2020.06.001.

[29] P. Cundall, O. Strack, A discrete numerical model for granular assemblies, Geotechnique (1979) 47–65, https://doi.org/10.1680/geot.1979.29.1.47.

[30] T.K.F. Da, S. Loriot, M. Yvinec, 3D alpha shapes, in: CGAL Editorial Board (Ed.), CGAL User and Reference Manual. 4.11.3, 2018, https://doc.cgal.org/4.11.3/Manual/packages.html#PkgAlphaShapes3Summary.

[31] L. Dalcin, Y.L.L. Fang, mpi4py: status update after 12 years of development, Comput. Sci. Eng. 23 (2021) 47–54, https://doi.org/10.1109/mcse.2021.3083216.

[32] C.A. del Valle, V. Angelidakis, S. Roy, J.D. Muñoz, T. Pöschel, SPIRAL: An efficient algorithm for the integration of the equation of rotational motion, Comput. Phys. Commun. 297 (2024) 109077, https://doi.org/10.1016/j.cpc.2023.109077.

[33] M. Dosta, D. Andre, V. Angelidakis, R. Caulk, M. Celigueta, B. Chareyre, J.F. Dietiker, J. Girardot, N. Govender, C. Hubert, R. Kobyłka, A. Moura, V. Skorych, D. Weatherley, T. Weinhart, Comparing open-source DEM frameworks for simulations of common bulk processes, Comput. Phys. Commun. 296 (2024) 109066, https://doi.org/10.1016/j.cpc.2023.109066.

[34] J. Duriez, S. Bonelli, Precision and computational costs of Level Set-Discrete Element Method (LS-DEM) with respect to DEM, Comput. Geotech. 134 (2021) 104033, https://doi.org/10.1016/j.compgeo.2021.104033.

[35] J. Duriez, C. Galusinski, A Level Set-Discrete Element Method in YADE for numerical, micro-scale, geomechanics with refined grain shapes, Comput. Geosci. 157 (2021) 104936, https://doi.org/10.1016/j.cageo.2021.104936.

[36] J. Duriez, L. Scholtès, F.V. Donzé, Micromechanics of wing crack propagation for different flaw properties, Eng. Fract. Mech. 153 (2016) 378–398, https://doi.org/10.1016/j.engfracmech.2015.12.034.

[37] J. Duriez, R. Wan, Contact angle mechanical influence for wet granular soils, Acta Geotech. 12 (2017) 67–83, https://doi.org/10.1007/s11440-016-0500-6.

[38] S. Duverger, A multi-scale, MPMxDEM, numerical modelling approach for geotechnical structures under severe loading, Ph.D. thesis, Aix-Marseille Université (AMU), 2023, https://theses.hal.science/tel-04101270.

[39] S. Duverger, V. Angelidakis, S. Nadimi, S. Utili, S. Bonelli, P. Philippe, J. Duriez, Investigation techniques and physical aspects of the angle of repose of granular matter, Granul. Matter 26 (2024) 20, https://doi.org/10.1007/s10035-023-01378-z.

[40] Sacha Duverger, Jérôme Duriez, Pierre Philippe, Stéphane Bonelli, Rattlers´ involvement for possibly looser critical states under higher mean stress, in: Proc. of Powders & Grains 2021, EPJ Web Conf. 249 (2021) 11002, https://doi.org/10.1051/epjconf/202124911002.

[41] A. Effeindzourou, B. Chareyre, K. Thoeni, A. Giacomini, F. Kneib, Modelling of deformable structures in the general framework of the discrete element method, Geotext. Geomembr. 44 (2016) 143–156, https://doi.org/10.1016/j.geotexmem.2015.07.015.

[42] J. Eliáš, Simulation of railway ballast using crushable polyhedral particles, Powder Technol. 264 (2014) 458–465, https://doi.org/10.1016/j.powtec.2014.05.052.

[43] D. Fincham, Leapfrog rotational algorithms, Mol. Simul. 8 (1992) 165–178, https://doi.org/10.1080/08927029208022474.

[44] N. Frankel, A. Acrivos, On the viscosity of a concentrated suspension of solid spheres, Chem. Eng. Sci. 22 (1967) 847–853, https://doi.org/10.1016/0009-2509(67)80149-0.

[45] J.A. Gili, E.E. Alonso, Microstructural deformation mechanisms of unsaturated granular soils, Int. J. Numer. Anal. Methods Geomech. 26 (2002) 433–468, https://doi.org/10.1002/nag.206.

[46] A. Gladkyy, M. Kuna, DEM simulation of polyhedral particle cracking using a combined Mohr–Coulomb–Weibull failure criterion, Granul. Matter 19 (2017) 41, https://doi.org/10.1007/s10035-017-0731-8.

[47] A. Gladkyy, R. Schwarze, Comparison of different capillary bridge models for application in the discrete element method, Granul. Matter (2014) 1–10, https://doi.org/10.1007/s10035-014-0527-z.

[48] N. Guo, Multiscale characterization of the shear behavior of granular media, Ph.D. thesis, Hong Kong University of Science and Technology, 2014, https://doi.org/10.14711/thesis-b1334193.

[49] N. Guo, J. Zhao, A coupled FEM/DEM approach for hierarchical multiscale modelling of granular media, Int. J. Numer. Methods Eng. 99 (2014) 789–818, https://doi.org/10.1002/nme.4702.

[50] J. Harkness, A. Zervos, L. Le Pen, S. Aingaran, W. Powrie, Discrete element simulation of railway ballast: modelling cell pressure effects in triaxial tests, Granul. Matter 18 (2016) 1–13, https://doi.org/10.1007/s10035-016-0660-y.

[51] B. Harthong, L. Scholtès, F.V. Donzé, Strength characterization of rock masses, using a coupled DEM–DFN model, Geophys. J. Int. 191 (2012) 467–480, https://doi.org/10.1111/j.1365-246X.2012.05642.x.

[52] P. Hartmann, K. Thoeni, J. Rojek, A generalised multi-scale Peridynamics-DEM framework and its application to rigid-soft particle mixtures, Comput. Mech. 71 (2023) 107–126, https://doi.org/10.1007/s00466-022-02227-1.

[53] M. Haustein, A. Gladkyy, R. Schwarze, Discrete element modeling of deformable particles in YADE, SoftwareX 6 (2017) 118–123, https://doi.org/10.1016/j.softx.2017.05.001.

[54] D.L.H. van der Haven, I.S. Fragkopoulos, J.A. Elliott, A physically consistent discrete element method for arbitrary shapes using volume-interacting level sets, Comput. Methods Appl. Mech. Eng. 414 (2023) 116165, https://doi.org/10.1016/j.cma.2023.116165.

[55] D.L.H. van der Haven, I.S. Fragkopoulos, J.A. Elliott, Volume-interacting level set discrete element method: the porosity and angle of repose of aspherical, angular, and concave particles, Powder Technol. 433 (2024) 119295, https://doi.org/10.1016/j.powtec.2023.119295.

[56] C. Jamin, S. Pion, M. Teillaud, 3D triangulations, in: CGAL Editorial Board (Ed.), CGAL User and Reference Manual. 5.6, 2023, https://doc.cgal.org/5.6/Manual/packages.html#PkgTriangulation3.

[57] M. Jean, The non-smooth contact dynamics method, Comput. Methods Appl. Mech. Eng. 177 (1999) 235–257, https://doi.org/10.1016/S0045-7825(98)00383-1.

[58] D. Jeffrey, Y. Onishi, The forces and couples acting on two nearly touching spheres in low-Reynolds-number flow, Z. Angew. Math. Phys. 35 (1984) 634–641, https://doi.org/10.1007/bf00952109.

[59] A.X. Jerves, R.Y. Kawamoto, J.E. Andrade, Effects of grain morphology on critical state: a computational analysis, Acta Geotech. 11 (2016) 493–503, https://doi.org/10.1007/s11440-015-0422-8.

[60] M. Jiang, S. Leroueil, J. Konrad, Insight into shear strength functions of unsaturated granulates by DEM analyses, Comput. Geotech. 31 (2004) 473–489, https://doi.org/10.1016/j.compgeo.2004.07.001.

[61] R. Kawamoto, E. Andò, G. Viggiani, J.E. Andrade, Level set discrete element method for three-dimensional computations with triaxial case study, J. Mech. Phys. Solids 91 (2016) 1–13, https://doi.org/10.1016/j.jmps.2016.02.021.

[62] L. Kettner, 3D polyhedral surface, in: CGAL Editorial Board (Ed.), CGAL User and Reference Manual.4.11.3, 2018, http://doc.cgal.org/4.11.3/Manual/packages.html#PkgPolyhedronSummary.

[63] C. Kloss, C. Goniva, A. Hager, S. Amberger, S. Pirker, Models, algorithms and validation for opensource DEM and CFD-DEM, Prog. Comput. Fluid Dyn. 12 (2012) 140–152, https://doi.org/10.1504/pcfd.2012.047457.

[64] J. Kozicki, F. Donzé, A new open-source software developed for numerical simulations using discrete modeling methods, Comput. Methods Appl. Mech. Eng. 197 (2008) 4429–4443, https://doi.org/10.1016/j.cma.2008.05.023.

[65] J. Kozicki, A. Gladky, K. Thoeni, Implementation of high-precision computation capabilities into the open-source dynamic simulation framework YADE, Comput. Phys. Commun. 270 (2022) 108167, https://doi.org/10.1016/j.cpc.2021.108167.

[66] D. Kunhappan, B. Harthong, B. Chareyre, G. Balarac, P.J.J. Dumont, Numerical modeling of high aspect ratio flexible fibers in inertial flows, Phys. Fluids 29 (2017) 093302, https://doi.org/10.1063/1.5001514.

[67] B. Leimkuhler, C. Matthews, Molecular Dynamics, Springer, Cham, 2015, https://doi.org/10.1007/978-3-319-16375-8.

[68] X. Li, Effective stress in unsaturated soil: a microstructural analysis, Géotechnique 53 (2003) 273–277, https://doi.org/10.1680/geot.2003.53.2.273.

[69] G. Lian, C. Thornton, M.J. Adams, A theoretical study of the liquid bridge forces between two rigid spherical bodies, J. Colloid Interface Sci. 161 (1993) 138–147, https://doi.org/10.1006/jcis.1993.1452.

[70] W. Liang, J. Zhao, Multiscale modelling of large deformation in geomechanics, Int. J. Numer. Anal. Methods Geomech. 43 (2019) 1080–1114, https://doi.org/10.1002/nag.2921.

[71] S. Luding, Cohesive, frictional powders: contact models for tension, Granul. Matter 10 (2008) 235–246, https://doi.org/10.1007/s10035-008-0099-x.

[72] R. Mani, D. Kadau, H.J. Herrmann, Liquid migration in sheared unsaturated granular media, Granul. Matter 15 (2012) 447–454, https://doi.org/10.1007/s10035-012-0387-3.

[73] D. Marzougui, B. Chareyre, J. Chauchat, Microscopic origins of shear stress in dense fluid–grain mixtures, Granul. Matter 17 (2015) 297–309, https://doi.org/10.1007/s10035-015-0560-6.

[74] D. Mas Ivars, M.E. Pierce, C. Darcel, J. Reyes-Montes, D.O. Potyondy, R. Paul Young, P.A. Cundall, The synthetic rock mass approach for jointed rock mass modelling, Int. J. Rock Mech. Min. Sci. 48 (2011) 219–244, https://doi.org/10.1016/j.ijrmms.2010.11.014.

[75] C. Modenese, Numerical study of the mechanical properties of lunar soil by the discrete element method, Ph.D. thesis, Oxford University, UK, 2013, https://ora.ox.ac.uk/objects/uuid:c8908ef8-9652-4e8d-9b2f-49770f3ce815.

[76] P. Müller, T. Pöschel, Collision of viscoelastic spheres: compact expressions for the coefficient of normal restitution, Phys. Rev. E 84 (2011) 021302, https://doi.org/10.1103/PhysRevE.84.021302.
[77] D. Nishiura, M.Y. Matsuo, H. Sakaguchi, ppohDEM: Computational performance for open source code of the discrete element method, Comput. Phys. Commun. 185 (2014) 1486–1495, https://doi.org/10.1016/j.cpc.2014.02.014.
[78] M. Nitka, G. Combe, C. Dascalu, J. Desrues, Two-scale modeling of granular materials: a DEM-FEM approach, Granul. Matter 13 (2011) 277–281, https://doi.org/10.1007/s10035-011-0255-6.
[79] I.P. Omelyan, Algorithm for numerical integration of the rigid-body equations of motion, Phys. Rev. E 58 (1998) 1169–1172, https://doi.org/10.1103/PhysRevE.58.1169.
[80] E. Papachristos, L. Scholtès, F. Donzé, B. Chareyre, Intensity and volumetric characterizations of hydraulically driven fractures by hydro-mechanical simulations, Int. J. Rock Mech. Min. Sci. 93 (2017) 163–178, https://doi.org/10.1016/j.ijrmms.2017.01.011.
[81] B. Patzák, Oofem — an object-oriented simulation tool for advanced modeling of materials and structures, Acta Polytech. 52 (2012), https://doi.org/10.14311/1678.
[82] G. Pekmezi, B. Chareyre, D. Littlefield, Uniform boundary conditions on models of spherical particles through alpha shape surface tracking and Laguerre–Voronoi diagrams, Comput. Phys. Commun. 301 (2024) 109214, https://doi.org/10.1016/j.cpc.2024.109214.
[83] G. Pekmezi, D. Littlefield, B. Chareyre, Statistical distributions of the elastic moduli of particle aggregates at the mesoscale, Int. J. Impact Eng. 139 (2020) 103481, https://doi.org/10.1016/j.ijimpeng.2019.103481.
[84] D.O. Potyondy, A grain-based model for rock: approaching the true microstructure, in: Proceedings of Rock Mechanics in the Nordic Countries, 2010, pp. 9–12.
[85] D.O. Potyondy, The bonded-particle model as a tool for rock mechanics research and application: current trends and future directions, Geosyst. Eng. 18 (2015) 1–28, https://doi.org/10.1080/12269328.2014.998346.
[86] R. Schaa, L. Gross, J. du Plessis, PDE-based geophysical modelling using finite elements: examples from 3D resistivity and 2D magnetotellurics, J. Geophys. Eng. 13 (2016) S59–S73, https://doi.org/10.1088/1742-2132/13/2/S59.
[87] L. Scholtès, B. Chareyre, F. Darve, Micromechanics of granular materials with capillary effects, Int. J. Eng. Sci. 47 (2009) 64–75, https://doi.org/10.1016/j.ijengsci.2008.07.002.
[88] L. Scholtès, P.Y. Hicher, F. Nicot, B. Chareyre, F. Darve, On the capillary stress tensor in wet granular materials, Int. J. Numer. Anal. Methods Geomech. 33 (2009) 1289–1313, https://doi.org/10.1002/nag.767.
[89] L. Scholtès, B. Chareyre, H. Michallet, E. Catalano, D. Marzoughi, Modeling wave-induced pore pressure and effective stress in a granular seabed, Contin. Mech. Thermodyn. 27 (2015) 305–323, https://doi.org/10.1007/s00161-014-0377-2.
[90] L. Scholtès, F.V. Donzé, Modelling progressive failure in fractured rock masses using a 3d discrete element method, Int. J. Rock Mech. Min. Sci. 52 (2012) 18–30, https://doi.org/10.1016/j.ijrmms.2012.02.009.
[91] L. Scholtès, F.V. Donzé, A DEM model for soft and hard rocks: role of grain interlocking on strength, J. Mech. Phys. Solids 61 (2013) 352–369, https://doi.org/10.1016/j.jmps.2012.10.005.
[92] T. Schwager, T. Pöschel, Coefficient of restitution and linear–dashpot model revisited, Granul. Matter 9 (2007) 465–469, https://doi.org/10.1007/s10035-007-0065-z.
[93] J. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, 1999, https://math.berkeley.edu/~sethian/Books/hold_sethian_book.pdf.
[94] M.I. Shamos, D. Hoey, Geometric intersection problems, in: 17th Annual Symposium on Foundations of Computer Science (sfcs 1976), IEEE, 1976, pp. 208–215, https://doi.org/10.1109/sfcs.1976.16.
[95] S. Silling, Reformulation of elasticity theory for discontinuities and long-range forces, J. Mech. Phys. Solids 48 (2000) 175–209, https://doi.org/10.1016/s0022-5096(99)00029-0.
[96] V. Šmilauer, Cohesive Particle Model using the Discrete Element Method on the Yade Platform, Ph.D. thesis, Czech Technical University in Prague, Faculty of Civil Engineering & Université Grenoble I – Joseph Fourier, 2010, https://tel.archives-ouvertes.fr/tel-00502402/document.
[97] V. Šmilauer, V. Angelidakis, E. Catalano, R. Caulk, B. Chareyre, W. Chèvremont, S. Dorofeenko, J. Duriez, N. Dyck, J. Elias, B. Er, A. Eulitz, A. Gladky, N. Guo, C. Jakob, F. Kneib, J. Kozicki, D. Marzougui, R. Maurin, C. Modenese, G. Pekmezi, L. Scholtès, L. Sibille, J. Stransky, T. Sweijen, K. Thoeni, C. Yuan, Yade documentation, in: The Yade Project, 3rd ed., 2021, http://yade-dem.org/doc/.
[98] W. Song, B. Huang, X. Shu, J. Stránský, H. Wu, Interaction between railroad ballast and sleeper: a DEM-FEM approach, Int. J. Geomech. 19 (2019), https://doi.org/10.1061/(asce)gm.1943-5622.0001388.
[99] J. Stránský, Mesoscale Discrete Element Model for Concrete and Its Combination with FEM, Ph.D. thesis, Czech Technical University in Prague, 2018, https://dspace.cvut.cz/handle/10467/75647.
[100] B. Suhr, W.A. Skipper, R. Lewis, K. Six, DEM modelling of railway ballast using the conical damage model: a comprehensive parametrisation strategy, Granul. Matter 24 (2022) 40, https://doi.org/10.1007/s10035-021-01198-z.
[101] T. Sweijen, S.M. Hassanizadeh, B. Chareyre, L. Zhuang, Dynamic pore-scale model of drainage in granular porous media: the pore-unit assembly method, Water Resour. Res. 54 (2018) 4193–4213, https://doi.org/10.1029/2017WR021769.
[102] T. Sweijen, E. Nikooee, S.M. Hassanizadeh, B. Chareyre, The effects of swelling and porosity change on capillarity: DEM coupled with a pore-unit assembly method, Transp. Porous Media 113 (2016) 207–226, https://doi.org/10.1007/s11242-016-0689-8.
[103] K. Thoeni, A. Giacomini, C. Lambert, S. Sloan, J. Carter, A 3D discrete element modelling approach for rockfall analysis with drapery systems, Int. J. Rock Mech. Min. Sci. 68 (2014) 107–119, https://doi.org/10.1016/j.ijrmms.2014.02.008.
[104] K. Thoeni, C. Lambert, A. Giacomini, S. Sloan, Discrete modelling of hexagonal wire meshes with a stochastically distorted contact model, Comput. Geotech. 49 (2013) 158–169, https://doi.org/10.1016/j.compgeo.2012.10.014.
[105] C. Thornton, S.J. Cummins, P.W. Cleary, An investigation of the comparative behaviour of alternative contact force models during inelastic collisions, Powder Technol. 233 (2013) 30–46, https://doi.org/10.1016/j.powtec.2012.08.012.
[106] O.R. Walton, Numerical simulation of inclined chute flows of monodisperse, inelastic, frictional spheres, Mech. Mater. 16 (1993) 239–247, https://doi.org/10.1016/0167-6636(93)90048-V, special Issue on Mechanics of Granular Materials.
[107] T. Weinhart, L. Orefice, M. Post, M.P. van Schrojenstein Lantman, I.F. Denissen, D.R. Tunuguntla, J. Tsang, H. Cheng, M.Y. Shaheen, H. Shi, P. Rapino, E. Grannonio, N. Losacco, J. Barbosa, L. Jing, J.E. Alvarez Naranjo, S. Roy, W.K. den Otter, A.R. Thornton, Fast, flexible particle simulations – an introduction to MercuryDPM, Comput. Phys. Commun. 249 (2020) 107129, https://doi.org/10.1016/j.cpc.2019.107129.
[108] C. Yuan, B. Chareyre, A pore-scale method for hydromechanical coupling in deformable granular media, Comput. Methods Appl. Mech. Eng. 318 (2017) 1066–1079, https://doi.org/10.1016/j.cma.2017.02.024.
[109] L. Zhang, L. Scholtès, F. Donzé, Discrete element modeling of permeability evolution during progressive failure of a low-permeable rock under triaxial compression, Rock Mech. Rock Eng. 54 (2021) 6351–6372, https://doi.org/10.1007/s00603-021-02622-9.
[110] S. Zhao, J. Zhao, SudoDEM: unleashing the predictive power of the discrete element method on simulation for non-spherical granular particles, Comput. Phys. Commun. 259 (2021) 107670, https://doi.org/10.1016/j.cpc.2020.107670.