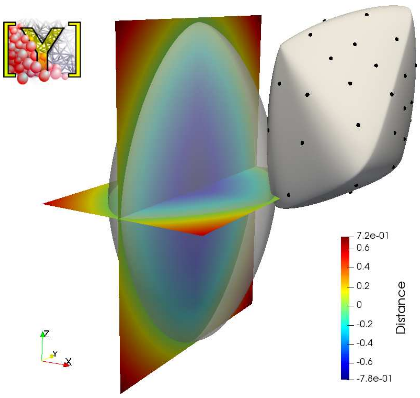
A Level Set-Discrete Element Method in YADE for numerical, micro-scale, geomechanics with refined grain shapes

Jérôme Duriez, Cédric Galusinski

Graphical Abstract

A Level Set-Discrete Element Method in YADE for numerical, micro-scale, geomechanics with refined grain shapes

Jérôme Duriez, Cédric Galusinski



A Level Set-Discrete Element Method in YADE for numerical, micro-scale, geomechanics with refined grain shapes

Jérôme Duriez[a,1,*], Cédric Galusinski[b,2]

[a]INRAE, Aix Marseille Univ, RECOVER, Aix-en-Provence, France
[b]IMATH, Université de Toulon, CS 60584, 83041 Toulon Cedex 9, France

Abstract

A C++-Python package is proposed for 3D mechanical simulations of granular geomaterials, seen as a collection of particles being in contact interaction one with another while showing complex grain shapes. Following the so-called Level Set-Discrete Element Method (LS-DEM), the simulation workflow stems from a discrete field for the signed distance function to every particle, with its zero-level set corresponding to a particle's surface. A Fast Marching Method is proposed to construct such a distance field for a wide class of surfaces. In connection with dedicated contact algorithms and Paraview visualization procedures, this shape description eventually extends the YADE platform for discrete simulations. Its versatility is illustrated on superquadric particles i.e. superellipsoids. On computational aspects, memory requirements possibly exceed one megabyte (MB) per particle when using a double numeric precision, and time costs, though also significant, appear to

*Corresponding author
Email address: `jerome.duriez@inrae.fr` (Jérôme Duriez)
URL: `https://www6.paca.inrae.fr/recover/membres-du-laboratoire/pages-personnelles/jerome-duriez` (Jérôme Duriez)
[1]Contribution: Writing - Original Draft, Software, Funding
[2]Contribution: Writing - Review & Editing, Formal analysis

be lighter than the use of convex polyhedra and can be drastically reduced using a simple, OpenMP, parallel execution.

Keywords: Discrete Element Method (DEM), Level Set-DEM (LS-DEM), particle's shape, Fast Marching Method (FMM)

1. Introduction

Geomaterials very often show a discrete nature which controls their solid-like strains or fluid-like strain rates while being under stress, e.g. granular soils. A proper description of that mechanical behavior is of interest to countless geo-engineering problems, e.g. the safe design of large rockfill dams (Deluzarche and Cambou, 2006), possibly rising in the order of hundreds of meters after piling up decimetric pieces of rock, or the forecast of snow mechanical stability and avalanches (Hagenmuller et al., 2015). Unlike the equivalent continuum descriptions classically used in engineering practice, numerical modelling approaches based on the Discrete Element Method (DEM, Cundall and Strack, 1979) duly respect this granular nature by describing the time evolution of a discrete set of particles, the so-called Discrete Elements (DE), in mechanical interaction. While DEM approaches often serve for qualitative studies in discrete geomechanics (e.g. Guo and Zhao, 2013; Duriez et al., 2018), they also are more and more often deployed for quantitative modelling (e.g. Aboul Hosn et al., 2017), possibly in a multiscale framework where a DEM description of a Representative Elementary Volume eventually substitutes constitutive relations in a FEM-like model (Miehe et al., 2010; Guo and Zhao, 2014).

On that quantitative point of view, the predictive abilities of DEM may appear as variable depending on the loading conditions (Aboul Hosn et al., 2017). As a matter of fact, they certainly often suffer from a spherical shape assumption (adopted e.g. by Duriez et al., 2011; Guo and Zhao, 2013; Duriez et al., 2018; Aboul Hosn et al., 2017), since such spheres constitute a very strong simplification of material particles while particle's shape has a known influence on the macroscopic behavior (e.g. Cho et al., 2006). Therefore, various DEM strategies towards a better shape description have been introduced, such as the use of rigid aggregates of spheres, so-called clumps, that should mimic real shapes (e.g. Garcia et al., 2009; Mede et al., 2018); or the direct consideration of polyhedra (e.g. Eliáš, 2014; Gladkyy and Kuna, 2017). Clumps offer the advantage to accommodate straightforward and computationally cheap contact algorithms designed for spheres, but still present some unrealistic local roundness. On the other hand, polyhedra resort to more complex algorithms, which still remain restricted, most often, to convex surfaces (Dubois, 2011). One can also note the potential particles (PP) or potential blocks (PB) approaches by Houlsby (2009); Boon et al. (2012, 2013), which both describe particles' surfaces resorting to the zero-level of a so-called potential. Each scalar potential is given in a set of closed-form expressions with a variable number of shape parameters, leading to rounded (PP-case) or angular (PB-case) surfaces that are necessarily convex. Then, the so-called Level Set Discrete Element Method (LS-DEM) has been recently proposed by Kawamoto et al. (2016), in 3D, as another DEM extension towards realistic shapes. In LS-DEM, and with a limited similarity to PP and PB approaches, every DE's surface is implicitly described as the zero-level set of the specific signed distance function to that surface. Contributing to its generality, no closed-form equation or convexity assumptions are required in LS-DEM since Level Set and Fast Marching Methods (Osher and Sethian, 1988; Sethian, 1996, 1999) are available to construct distance fields for arbitrary, possibly concave, surfaces and a wide class of scientific applications (e.g. Yang et al., 2019). Kawamoto et al. (2016, 2018) actually illustrated the capabilities of the LS-DEM to describe real soil grains, shapes being acquired through X-ray computed tomography, as well as its promising features to reproduce observed behaviors both qualitatively and quantitatively.

The present contribution then proposes an independent and original implementation of LS-DEM into the existing YADE open-source platform (Šmilauer et al., 2015), which is often used for geo-mechanical simulations (e.g. Duriez et al., 2011; Boon et al., 2013; Duriez et al., 2018; Aboul Hosn et al., 2017; Pirnia et al., 2019). Example usages are furthermore provided for complex, superquadric shapes, alongside discussing computational costs in comparison with the polyhedral shape description.

Section 2 first recalls Level Set and Fast Marching Methods serving to establish distance fields for arbitrary surfaces. Then, Section 3 describes how LS-DEM uses the particles' distance fields for DEM simulations of granular soils, usually following here the initial guidelines of Kawamoto et al. (2016) or Duriez and Bonelli (2021). An original LS-DEM code is proposed accordingly and summarized in Section 4. Section 5 and 6 present a direct application to non-spherical, superquadric, shapes with illustrative simulations and a computational comparison with the use of convex polyhedra.

2. Level Set and Fast Marching methods

2.1. Level set formalism

Level set approaches (Osher and Sethian, 1988; Sethian, 1999) see interfaces $\mathcal{S}(t)$ as the zero-level set of a function $\phi^t(\vec{x}, t)$ being defined from $\mathbb{R}^d \times \mathbb{R}$ into $\mathbb{R}$, with $\mathbb{R}^d$ covering the whole space of a d dimensionnality. Evolving contours (resp. surfaces) can then be described for $d = 2$ (resp. $d = 3$). While the interfaces evolve, propagating with a normal velocity

$\vec{v} = F(\vec{x}, t)\, \vec{n}$ (where $\vec{n}$ stands for the outwards normal), all level sets evolve with an extended velocity parallel to the gradient of the level set function $\phi^t(\vec{x}, t)$. Since $\phi^t(\vec{x}, t)$ is constant along $\mathcal{S}(t)$ (equal to 0 $\forall t$), the nullity of the material derivative along the interface front leads to the following level set equation:

$$\frac{\partial \phi^t}{\partial t} + F||\vec{\nabla}\phi^t|| = 0 \tag{1}$$

Eq. (1) conforms the formalism of Hamilton-Jacobi partial derivative equations (Osher and Sethian, 1988), with the Hamiltonian H^t, as a function of the spatial derivative(s) of ϕ^t, being equal to:

$$H^t(\phi^t_x) = F|\phi^t_x| \qquad \text{for } d = 1 \tag{2}$$

$$H^t(\phi^t_x, \phi^t_y) = F\sqrt{{\phi^t_x}^2 + {\phi^t_y}^2} \qquad \text{for } d = 2 \tag{3}$$

$$H^t(\phi^t_x, \phi^t_y, \phi^t_z) = F\sqrt{{\phi^t_x}^2 + {\phi^t_y}^2 + {\phi^t_z}^2} \qquad \text{for } d = 3 \tag{4}$$

where f_x, f_y, f_z stand for the spatial derivatives of any scalar function f with respect to x, y, z.

The signed (shortest) distance to $\mathcal{S}(t)$ is a typical choice for the function ϕ^t, with the convention of a negative, resp. positive, distance when being inside, resp. outside, of $\mathcal{S}(t)$. Doing so, and for a constant and uniform speed $F(\vec{x}, t) = F$, one can relate ϕ^t to $T(\vec{x})$, the arrival time of $\mathcal{S}$ at $\vec{x}$:

$$\phi^t(\vec{x}, t) = F\,(T(\vec{x}) - t) \tag{5}$$

Inserting Eq. (5) into the level set equation (1), one easily re-obtains the so-called Eikonal equation:

$$F\,||\vec{\nabla}T|| = 1 \tag{6}$$

With respect to the use of ϕ^t and the level set Eq. (1), the consideration of T and the Eikonal Eq. (6) forms another description of evolving interfaces, adapted to the case of a constant and uniform sign for the normal velocity.

Doing so, the current interface $\mathcal{S}(t)$ is the t-level set of T and no time variable enters the partial differential equation (6). That stationary perspective can be finally complemented by the consideration of $\phi(\vec{x})$, the distance to $\mathcal{S}(t = 0)$:

$$\phi(\vec{x}) = \phi^t(\vec{x}, 0) = F\,T(\vec{x}) \tag{7}$$

with the following form for the Eikonal equation:

$$||\vec{\nabla}\phi|| = 1 \Leftrightarrow H(\phi_x, ..) = 1 \tag{8}$$

Similar to Eqs. (1)-(4), Eq. (8) can be cast in the form of a Hamiltonian $H(\phi_x, ..) = 1$ with:

$$H(\phi_x) = |\phi_x| \qquad \text{for } d = 1 \tag{9}$$

$$H(\phi_x, \phi_y) = \sqrt{{\phi_x}^2 + {\phi_y}^2} \qquad \text{for } d = 2 \tag{10}$$

$$H(\phi_x, \phi_y, \phi_z) = \sqrt{{\phi_x}^2 + {\phi_y}^2 + {\phi_z}^2} \qquad \text{for } d = 3 \tag{11}$$

2.2. A Fast Marching Method for the stationary perspective

Looking for the distance field ϕ to a given, constant, surface $\mathcal{S}$, the Eikonal equation (8) can be efficiently solved using a so-called Fast Marching Method (FMM, Sethian, 1996, 1999). Space being discretized on a grid, the Eikonal equation makes the ϕ-value at some gridpoint $\vec{x}_i$ being directly dependent upon surrounding ϕ-values at adjacent gridpoints, as can be seen from finite difference expressions for the spatial derivatives in Eqs. (9) to (11). Accounting for the monotonous nature of ϕ, which strictly increases (in absolute value) when $\vec{x}$ goes away of $\mathcal{S}$, the FMM eventually gives the full discrete field $\phi(\vec{x}_i)$ starting from an initial set of gridpoints being along, or close to, the surface and serving as boundary conditions. In more details, the FMM recursively applies Eq. (8), in the form of Eqs. (9) or (10) or (11) depending on gradient's dimensionnality, and adopting gradient expressions decentred to low and known ϕ-values. Recursive applications actually go in a downwind direction away from the surface, until the whole spatial grid has been handled. The key point of the FMM is to go through the grid points in the right order, following at each step the minimal value of distance.

The FMM for instance directly applies to any surface $\mathcal{S}$ showing a scalar inside/outside function $f(\vec{x})$, being positive (resp. negative) for $\vec{x}$ located outside (resp. inside) the surface and null along the surface. In such a case, boundary conditions gridpoints are easily identified as all gridpoints being outside of the surface and having a grid neighbor inside and they can be assigned the following ϕ-value:

$$\phi(\vec{x}) = \frac{f(\vec{x})}{||\vec{\nabla} f(\vec{x})||} \text{ for } \vec{x} \text{ close to } \mathcal{S} \tag{12}$$

By construction, Eq. (12) is a first order approximation to ϕ, obviously obeying $\phi = 0$ along $\mathcal{S}$ and also verifying the Eikonal equation (8) close to $\mathcal{S}$, provided that $\vec{\nabla}(1/||\vec{\nabla} f||)$ is finite. This constitutes the initialization of the distance function on the grid points close to the interface, before applying the recursive operations of the FMM.

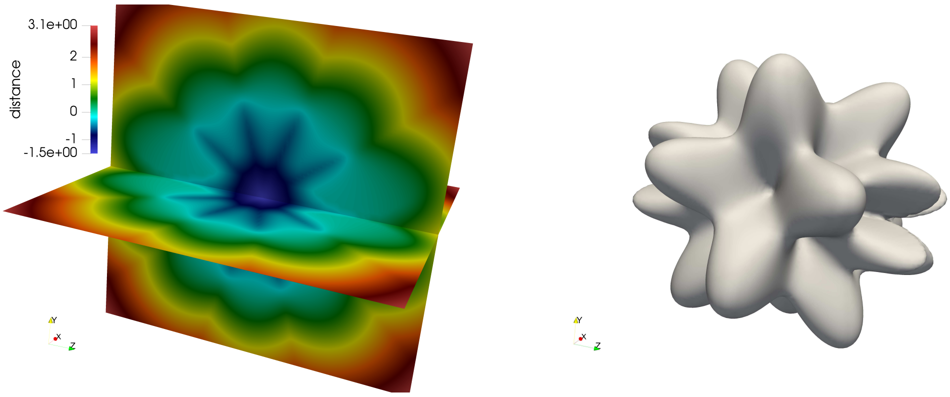
Figure 1 illustrates the distance output of such a FMM procedure, herein implemented in a `DistFMM` C++ class presented in Section 4.1, when applied to the following "flake-like" inside/outside function:

$$f(\vec{x}) = r - [R + \Delta R \sin(5\theta) \sin(4\varphi)] \tag{13}$$

In Eq. (13), (r, θ, φ) refer to spherical coordinates with $\theta \in [0; \pi]$ measured from $\vec{z}$ axis and $\varphi \in [0; 2\pi]$ measured in $(\vec{x}, \vec{y})$ plane.

3. LS-DEM formulation

For any DEM mechanical simulation to progress in time, it is first necessary to describe the shapes of the bodies, i.e. DEs, and detect their pos-



(a) Distance field ϕ (b) Surface as the zero-contour of ϕ

Figure 1: Level Set description of a flake-like surface defined by Eq. (13), with $(R; \Delta R) = (3; 1.5)$, after executing a FMM on a 0.1-spaced isotropic grid

sible contact interactions with neighbors. Then, contact-scale constitutive relationships express forces and torques so that rigid motion equations can finally be integrated. The following sections detail these three steps.

3.1. LS-DEM shape description

Following Kawamoto et al. (2016, 2018), a discrete signed distance field on a body-centered regular grid, possibly obtained from the previous FMM, is the first LS-DEM ingredient. That (grid ; distance field) pair is independently defined for every DE in a local coordinate system and first serves for defining the DE inertial quantities (mass m and inertia matrix $\boldsymbol{I} = I_{\alpha\beta}, \alpha, \beta \in \{x, y, z\}$) summing contributions from grid voxels v making up the body's volume V as per the following discrete-form equations:

$$m = \rho \sum_{v \in V} V_v = \rho\, N_{vox}\, V_v \tag{14}$$

$$\vec{x} = \frac{1}{N_{vox}} \sum_{v \in V} \vec{x}_v \tag{15}$$

$$I_{xx} = \rho \sum_{v \in V} \left[(y_v - y)^2 + (z_v - z)^2 \right] V_v \tag{16}$$

$$I_{yy} = \rho \sum_{v \in V} \left[(x_v - x)^2 + (z_v - z)^2 \right] V_v \tag{17}$$

$$I_{zz} = \rho \sum_{v \in V} \left[(x_v - x)^2 + (y_v - y)^2 \right] V_v \tag{18}$$

$$I_{xy} = -\rho \sum_{v \in V} (x_v - x) \times (y_v - y) V_v \tag{19}$$

$$I_{xz} = -\rho \sum_{v \in V} (x_v - x) \times (z_v - z) V_v \tag{20}$$

$$I_{yz} = -\rho \sum_{v \in V} (y_v - y) \times (z_v - z) V_v \tag{21}$$

In the above equations, ρ is the material mass density, $\vec{x}_v = (x_v, y_v, z_v)$ the middle point of a voxel and $\vec{x} = (x, y, z)$ the body's center of mass. Eqs. (15),(19)-(21) serve for verification purposes since the body-attached local frame is expected to be inertial and $\vec{x}, I_{xy}, I_{xz}, I_{yz}$ to be nil. By a simple convention, a grid voxel v of volume V_v is herein said to be part of the body's volume V when its lowest corner is inside the surface, showing a zero or negative distance value. While smoother choices have been proposed by Kawamoto et al. (2016, 2018), it will be verified in Section 5.2 that the present choice does not inhibit precision for grids being fine enough, i.e. showing a spacing g_{grid} at least ten times smaller than a grain's characteristic size l_{grain}.

For the purpose of LS-DEM contact algorithms that will be described in Section 3.2 below, a second LS-DEM ingredient adds to the distance field, in the form of a set of N_n boundary nodes $\{N_i,\ i \in [0; N_n - 1]\}$ discretizing each body's surface $\mathcal{S}$. Generally speaking, boundary nodes should count in the order of thousands and their positions are defined at the intersection of $\mathcal{S}$ i.e. $\phi(\vec{x}) = 0$ and N_n half-lines i.e. rays $\lambda\,\vec{v}$, with $\vec{v}$ a direction and λ a positive abscissa, that stem from the center of mass. Due to the adopted tri-linear interpolation of the discrete distance field within the grid extents, $\phi(\vec{x} = \lambda\vec{v})$ is a cubic polynomial in λ whose coefficients depend upon grid distance values and ray tracing boundary nodes corresponds to solve its positive roots (see e.g. Lin and Ching, 1996). Since rays should provide an appropriate discretization of spherical angles (θ, φ), the corresponding directions (θ, φ) are chosen to follow a spiral path instead of a simple rectangular discretization of $[0;\pi] \times [0;2\pi]$ in order to avoid a possible (shape-dependent) concentration of nodes at the poles $\theta = 0[\pi]$. More details about the spiral path or the choice of boundary nodes number are given by Duriez and Bonelli (2021) and in the next sections.

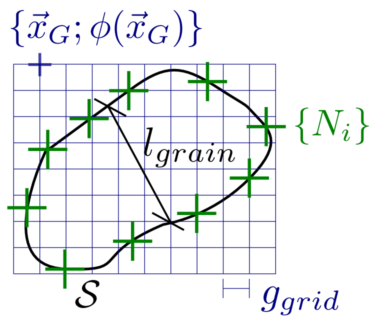


Figure 2: Shape description in LS-DEM: a regular grid $\{\vec{x}_G\}$ carrying the distance field ϕ, together with boundary nodes $\{N_i\}$ (2D view for clarity)

Figure 2 illustrates these LS-DEM ingredients which all refer to a constant shape-related (inertial) local frame and never need to be updated. Assuming rigid particles, the subsequent contact algorithm easily switches between global and local frames during simulations.

3.2. Contact law

From the distance fields and the set of boundary nodes, contact detection between two bodies 1 and 2 relies on a master-slave algorithm whereby nodes N_i^1 of body 1 are tested in the distance field ϕ_2 of body 2 (see also Figure 9). In order to increase precision, the body 1 is chosen as the smallest one in volume, which enables one to explore distance fields with the greatest surface density in nodes. Contact is detected as soon as one node N_i^1 verifies $\phi_2(N_i^1) \leqslant 0$. LS-DEM belongs to the wide class of "soft" DEM whereby small overlaps, $\phi_2(N_i^1) < 0$, are possible: these overlaps would in reality materialize through slight changes in shape which are neglected in soft DEM approaches. After identifying the set of contacting boundary nodes, a unique contact point is herein chosen from the node N_c showing the greatest interpenetration depth u_n, which also gives the contact normal as the local gradient of ϕ_1:

$$u_n = -\min(\phi_2(\overrightarrow{ON_i}),\ \overrightarrow{ON_i} \in \mathcal{S}_1) = -\phi_2(\overrightarrow{ON_c}) \geqslant 0 \tag{22}$$

$$\vec{n} = \vec{\nabla}\phi_1(\overrightarrow{ON_c}) \tag{23}$$

That final consideration of a unique contacting point, also adopted by Li et al. (2019), currently restricts the proposed LS-DEM implementation to convex shapes. For a pair of contacting bodies with concave shapes, multiple contact points would occur but these could be easily detected with the same master-slave algorithm. As such, Kawamoto et al. (2016, 2018) also addressed concave shapes by defining a mechanical interaction at each contacting boundary node. While being more general, this choice nevertheless poses the risk to make the macroscopic behavior, e.g. the bulk stiffness, to directly depend upon the chosen number of boundary nodes in case a physically unique contact area would involve more than one boundary node.

A classical contact law for cohesionless materials finally expresses the interaction force after decomposing the latter in a normal, along $\vec{n}$, and a tangential component. A repulsive normal force $\vec{F}_n$ first arises due to the interpenetration depth u_n, as per a linear elastic model with a k_n stiffness:

$$\vec{F}_n = k_n \, u_n \, \vec{n} \tag{24}$$

Along the tangential direction, a linear elastic-plastic relationship governs the shear force variations. Denoting k_t the shear stiffness and μ the friction coefficient, the following Eqs. (25)-(26) describe the variations of the shear force $\vec{F}_t$, starting from $\vec{0}$:

$$d\vec{F}_t = d\left(||\vec{F}_t||\frac{\vec{F}_t}{||\vec{F}_t||}\right) = ||\vec{F}_t||d\left(\frac{\vec{F}_t}{||\vec{F}_t||}\right) + d(||\vec{F}_t||)\frac{\vec{F}_t}{||\vec{F}_t||} \tag{25}$$

$$d(||\vec{F}_t||)\frac{\vec{F}_t}{||\vec{F}_t||} = k_t \, d\vec{u}_t \quad \text{enforcing } ||\vec{F}_t|| \leqslant \mu||\vec{F}_n|| \tag{26}$$

while updates in the shear force direction, $\vec{F}_t/||\vec{F}_t||$, are applied in order to follow changes in the tangent plane's orientation, e.g. a change in contact normal (Šmilauer et al., 2015).

3.3. Motion integration

As for general DEM, the translation and rotation of each DE in space, under resultant force $\vec{F}$ and torque $\vec{\Gamma}$ (computed at the center of mass), finally follow Newton-Euler equations for rigid bodies with $\vec{v}$ and $\vec{\omega}$ the linear and angular velocities:

$$m\frac{d\vec{v}}{dt} = (1 \pm D)\vec{F} \tag{27}$$

$$\boldsymbol{I}\frac{d\vec{\omega}}{dt} + \vec{\omega} \wedge \boldsymbol{I}\vec{\omega} = (1 \pm D)\vec{\Gamma} \tag{28}$$

The above Newton-Euler equations are classically damped using a numerical coefficient D, which modifies the resultant force and torque so that kinetic energy always decreases (or is led to increase by a smaller extent) as soon as $\vec{F} \neq \vec{0}$. Eq. (28), relating the variation in $\vec{\omega}$ with $\vec{\Gamma}$, is expressed in local axes where the inertia tensor $\boldsymbol{I}$ is constant. Denoting $\boldsymbol{R}(t)$ the rotation matrix passing from local axes to global ones, and $\boldsymbol{\Omega}$ the antisymmetric matrix such that $\boldsymbol{\Omega}\,\vec{x} = \vec{\omega} \wedge \vec{x}$, $\forall \vec{x}$, Eq. (28) is finally supplemented with:

$$\frac{d\boldsymbol{R}}{dt} = \boldsymbol{R}\boldsymbol{\Omega} \tag{29}$$

These equations (27)-(29) are then integrated over the time steps through an explicit algorithm common to any non-spherical shape in YADE (Šmilauer et al., 2015).

4. Proposed implementation

4.1. Source code

The present C++ and Python implementation inserts LS-DEM into the 2020.01a version, i.e. the `git` commit 9964f53, of the YADE platform (Šmilauer et al., 2015). Figure 3 illustrates the LS-DEM workflow exposed in the previous section together with the most noticeable new (or modified) C++ classes responsible for execution.

Looking from the `lsYade` root folder of the proposed source code, the files `pkg/dem/LevelSet.*pp` introduce the new shape descriptor `LevelSet`. That class includes the discrete distance field as a `LevelSet.distField` attribute. The regular grid carrying the distance field is `LevelSet.lsGrid`, which is an instance of the `RegularGrid` class. Boundary nodes are stored in `LevelSet.boundNodes` and computed (once, at the beginning of a simulation) solving for cubic roots during the ray tracing procedure mentioned in the above Section 3.1. A Newton-Raphson algorithm proposed by the external `Boost.Math` library is adopted for this purpose, being preferred over canonical formulae for numerical stability. Moreover, the distance cubic polynomial

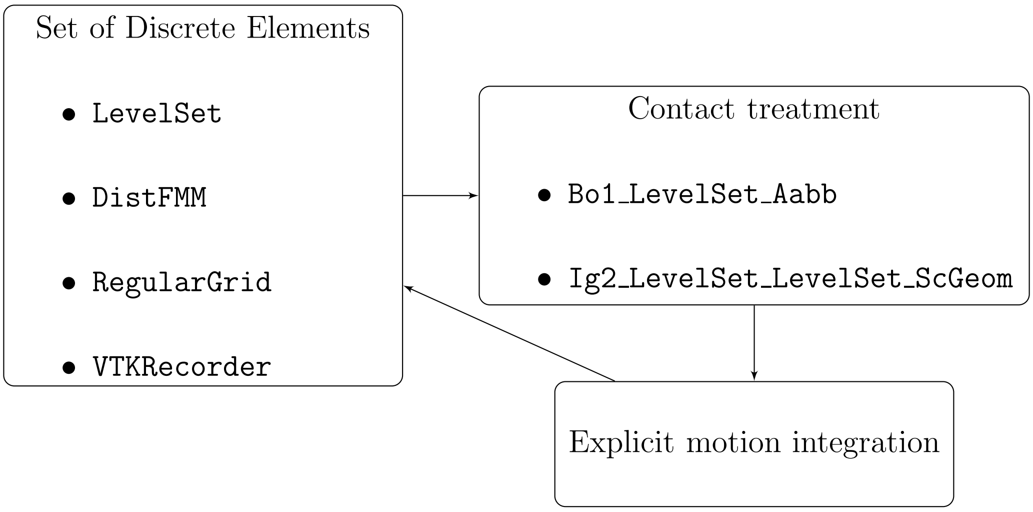


Figure 3: New or modified `C++ classes` for a LS-DEM workflow in YADE

is first turned dimensionless (with respect to the grid spacing) for the relative magnitude of its coefficients to be unaffected by the unit system, insuring a constant behavior of the root finding algorithm whatever the user's choice in this aspect. It is recalled the distance field, its grid and the boundary nodes all refer to a reference local (inertial) frame for each particle.

The distance field at the heart of the shape descriptor can be directly passed from the user (see next section) or also obtained from the Fast Marching Method proposed in `DistFMM` class. In the latter case, `DistFMM.phi()` is to execute once the grid and the boundary conditions are defined as `grid` and `phiIni` class attributes, respectively. Various predefined functions are proposed to build appropriate boundary conditions expected in `phiIni`. In addition to a `distIniSE` function intended to compute the distance to superquadric shapes detailed in Section 5.1, a versatile `PhiIniPy` function may be based upon any user-defined Python function that discriminates between the inside and the outside of a surface and outputs boundary condition values for the FMM, such as shown in Figure 1.

Visualization of LevelSet-shaped bodies relies on vtk exports of the discrete distance field for each DE in current configuration, thanks to a modified version of files pkg/dem/VTKRecorder.*pp. Actual display is typically done from Paraview software (Ayachit, 2019), using its Python interface and a provided pvVisu function defined in examples/levelSet/pvVisu.py. For the purpose of alternate vizualisation methods at the user's discretion, a LevelSet.marchingCubes method is also available from YADE interface and gives a triangulated description of a particle's surface as per the Marching Cubes algorithm (Lorensen and Cline, 1987).

The files pkg/dem/LevelSetInteraction.*pp finally implement the contact algorithms described in previous Section 3.2. The Bo1_LevelSet_Aabb is first responsible to compute an axis-aligned bounding box (Aabb) used in YADE for a first, crude and fast, detection of possible contacts. At the beginning of a simulation, that class first loops over the whole distance field to compute the 8 corners of the corresponding Aabb in local axes, stored in LevelSet.corners. Then, the current Aabb in model (global) axes is easily determined following rigid transformations. In case of overlap between Aabb, precise contact detection subsequently resorts to the other class Ig2_LevelSet_LevelSet_ScGeom, that implements the master-slave contact detection based on boundary nodes, identifying the contact point, if any, and the associated kinematic variables (normal vector, interpenetration depth) between two LevelSet-shaped bodies. Similar classes enable contact interaction between a LevelSet-shaped body and existing Wall or Box shapes, often adopted in YADE to simulate rigid boundaries.

In the end, the set of kinematic variables for a LS-DEM interaction is equivalent in nature to those used for spheres in general DEM and it can be stored in the existing ScGeom class. Constitutive properties k_n, k_t and μ

also correspond to the pre-existing `FrictPhys` contact model in YADE. This enables LS-DEM simulations to adopt a pre-existing contact law, namely `Law2_ScGeom_FrictPhys_CundallStrack`. Motion integration as described in previous Section 3.3 has also been readily available from the `NewtonIntegrator` class.

4.2. Code usage

The (modified or classical) YADE platform starts in the form of a Python3 interactive interface, invoked from `install/bin/yadelevelSet` in the proposed installation procedure (see the "Computer code availability" section). Instead of an interactive session, scripts prepared beforehand can be as well passed as argument and launched in the same manner than classical Python scripts. Examples of YADE scripts using the new LS-DEM features can be found in the source code at `lsYade/examples/levelSet/*.py` (`levelSetBody.py` in particular) and also `lsYade/scripts/checks-and-tests/checks/checkLSdem.py`. The latter actually serves as a new regression test into the YADE platform (Haustein et al., 2017), to insure stability of the LS-DEM features in the future. These examples illustrate the definition of `LevelSet` bodies through a new `levelSetBody()` YADE function. That function proposes level set descriptions of pre-defined analytical shapes (from boxes and spheres to superellipsoids, see next Section 5), together with the possibility of a direct assignment of the regular grid with its distance field. The latter enables users to directly insert any distance field they would have otherwise acquired, for instance from computed tomography (Vlahinić et al., 2014). In all cases, grid spacing g_{grid} is input through a `spacing` attribute while a `nNodes` attribute of `levelSetBody()` controls the boundary nodes number N_n.

Documentation can be obtained for any class or attribute in the usual interactive Python manner, typing e.g. `LevelSet?` or `levelSetBody?`. An HTML version of the documentation can also be built executing `make doc` from the compilation folder.

4.3. Code validation

The implementation is first validated for what concerns the FMM in `DistFMM` class. Applying the procedure on a sphere of radius R with a known distance field $\phi^{th}(\vec{x}) = r - R$, numerical precision can be quantified, looking e.g. at the average relative error on all gridpoints (excluding those with $\phi^{th} = 0$) or at the relative error at the center, as follows:

$$err_{avg} = \text{average}\left(\left\{\left|\frac{\phi(\vec{x}_i) - \phi^{th}(\vec{x}_i)}{\phi^{th}(\vec{x}_i)}\right| \ , \vec{x}_i \mid \phi^{th}(\vec{x}_i) \neq 0\right\}\right) \tag{30}$$

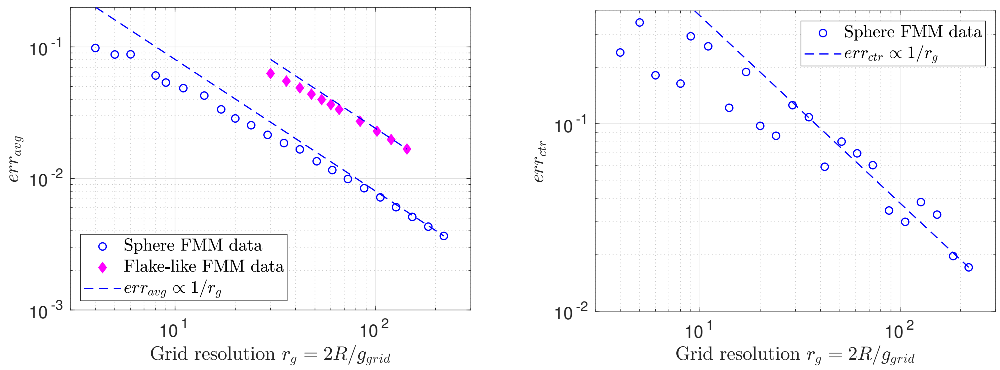
$$err_{ctr} = \frac{\min(\phi) + R}{R} \tag{31}$$

Considering Eq. (30), another average error is also analyzed for the more complex flake-like shape previously presented in Figure 1. While no exact distance field is known for such a surface, one could attempt a reconstruction of its inside/outside function f, Eq. (13), solving another variant of the Eikonal equation, with a non-unit speed, i.e.:

$$||\vec{\nabla}\phi|| = ||\vec{\nabla}f|| \tag{32}$$

By initializing the FMM, close to $\mathcal{S}$, with values of f: $\phi(\vec{x}_i) = f(\vec{x}_i)$ and solving for Eq. (32), one should indeed await $\phi^{th} = f$ as an exact solution.

For these two examples of a FMM application, Figure 4 illustrates how the FMM results approach their respective ϕ^{th} with a decreasing grid spacing g_{grid} i.e. an increasing grid resolution $r_g = 2R/g_{grid}$. While the precision is somewhat worse for the flake-like surface, in line with an increasing complexity of the problem, it always linearly scales with the grid resolution, in accordance with the first order expression of $\vec{\nabla}\phi$ in the numerical method.



(a) Average relative error err_{avg} (b) Relative error at the center err_{ctr}

Figure 4: Influence of the grid resolution on the FMM precision, with reference to spherical or flake-like (Figure 1) surfaces

Associated time costs are depicted in Figure 5. They refer to distance computations, i.e. solving Eq. (8) only, as per the present sequential FMM being executed on a workstation having one 4 cores, 8 threads, Intel i7-7700, 0.8 - 4.2GHz processor with 8 MB of cache memory, as well as 64 GB of 2.4 GHz RAM. Each case is run between 3 and 9 times (all depicted on the Figure) to account for possible variations in time cost, and after using the Linux command `cpufreq` and its `performance` governor set at 4.0 GHz. Denoting N_{gp} the total number of gridpoints, with $N_{gp} = \mathcal{O}({r_g}^3)$, a $\mathcal{O}({r_g}^6) = \mathcal{O}({N_{gp}}^2)$ complexity appears, in accordance with classical Level Set Methods. Sethian (1996) actually proposed a lighter complexity for the FMM, through adopting a heap sort when searching the minimum ϕ-value for propagating the distance field. For the purpose of LS-DEM, the FMM will apply only once per DE, at the very beginning of a simulation and the present time cost in the order of a second for few tens of grid voxel per particle length is actually acceptable, considering the final time cost of a complete LS-DEM simulation. Figure 5 finally illustrates that the FMM computation of distance for the more complex, non-spherical, flake-like shape logically shows the same time costs.

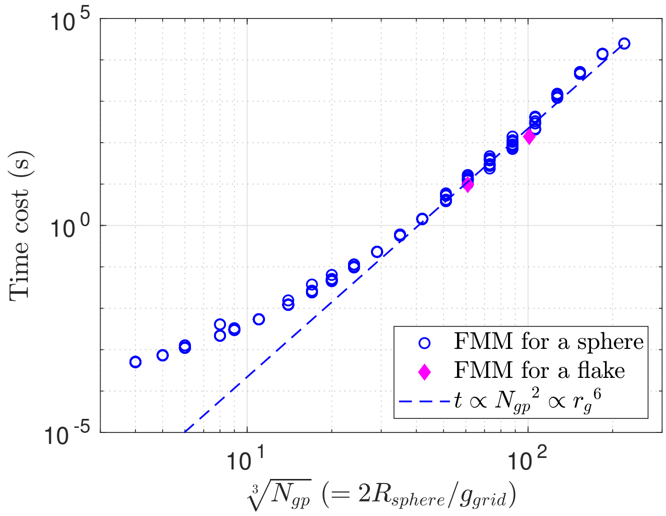


Figure 5: Time cost of the FMM according to the number of gridpoints per space axis $\sqrt[3]{N_{gp}}$, with N_{gp} the total number

FMM distance computations aside, code implementation was previously validated checking LS-DEM simulations of spherical particles did correspond with classical DEM simulations, provided that grid resolution and boundary nodes are appropriately chosen (Duriez and Galusinski, 2020; Duriez and Bonelli, 2021).

5. A direct application of LS-DEM to superquadric shapes

The versatility of LS-DEM to address complex shapes is now illustrated on superellipsoids, also known as superquadric ellipsoids. These surfaces are first presented from an analytical point of view before that their LS-DEM description is introduced with its corresponding precision and eventually compared with the possible use of convex polyhedra.

5.1. Superellipsoids surfaces

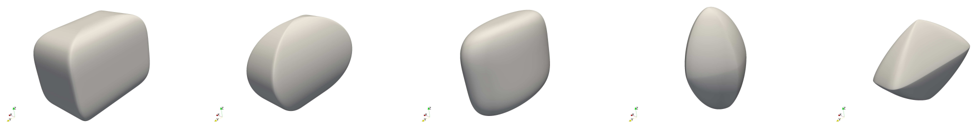
Superellipsoids (Barr, 1981, 1995) form a versatile class of surfaces which can be used as more complex shape models of granular soils (see e.g. Wang et al., 2019). They generalize ellipsoids through two additional exponents ϵ_e and ϵ_n that enter their surface equation together with three different radii r_x, r_y, r_z. In a local frame, the surface equation namely reads:

$$f(x,y,z) = \left(\left| \frac{x}{r_x} \right|^{\frac{2}{\epsilon_e}} + \left| \frac{y}{r_y} \right|^{\frac{2}{\epsilon_e}} \right)^{\frac{\epsilon_e}{\epsilon_n}} + \left| \frac{z}{r_z} \right|^{\frac{2}{\epsilon_n}} - 1 = 0 \tag{33}$$

Figure 6 illustrates five different superellipsoids, with their corresponding shape parameters presented in Table 1. Table 2 also details their volume and inertia properties, as obtained from closed form expressions given by Barr (1995). One can here observe how the ϵ_n exponent modifies the z-variation of cross-sections in (x, y) planes. For instance, adopting $\epsilon_n \to 0$ induces fairly constant cross-sections and a wider distribution of matter for extreme values along the "north-south" axis $\vec{z}$, see Shapes A or C. On the other hand, the $\epsilon_n = 1$ case corresponds to a rounded variation of these cross sections when progressing along $\vec{z}$ (Shape B). Some singularity, i.e. a sharpness at $z = 0$, would appear for $\epsilon_n \geqslant 2$, alongside concavity in a plane tangent to $\vec{z}$ for $\epsilon_n > 2$. For a given ϵ_n, ϵ_e controls the contour's roundness in the (x, y) plane of these cross-sections. While $\epsilon_e = 1$ corresponds to perfectly round (circles or ellipses) contours, decreasing ϵ_e towards 0 induce edges that tend to align with the $\vec{x}$ and $\vec{y}$ axes, see Shape A vs C. Alternate edges and sharpnesses would be obtained in the (x, y) plane at $\epsilon_e = 2$, just before concavity in that plane, for $\epsilon_e > 2$.

5.2. LS-DEM description of superellipsoids

Previous DEM descriptions of superellipsoids have already been proposed by Podlozhnyuk et al. (2017) or Weinhart et al. (2020), for instance. In those



(a) Shape A (b) Shape B (c) Shape C (d) Shape D (e) Shape E

Figure 6: Five possible superquadric shapes

Shape	Half-extents (cm)			Curvature exponents	
	r_x	r_y	r_z	ϵ_e	ϵ_n
A	0.58	1	0.83	0.1	0.5
B	0.42	1	0.83	0.1	1
C	same as Shape B			1	0.5
D	0.5	0.7	1	1.4	1.2
E	0.4	1	0.8	0.4	1.6

Table 1: Shape parameters of the five superellipsoids shown in Figure 6

Shape	Volume (cm^3)	Inertia components (cm^5)		
	V^{th}	I^{th}_{xx}/ρ	I^{th}_{yy}/ρ	I^{th}_{zz}/ρ
A	3.353	1.649	0.9751	1.358
B	1.852	0.7456	0.3417	0.5770
C	1.914	0.7996	0.4389	0.5153
D	1.093	0.2773	0.2350	0.1304
E	1.086	0.1283	0.3184	0.2625

Table 2: Geometric properties of the considered superellipsoid shapes. Inertia components are obtained following Barr (1995)

studies, contact detection involves a minimization procedure that endows the shape equation (33) with an approximated distance nature, following the potential approach by Houlsby (2009). Such a minimization is then performed by an iterative numerical method, at each DEM iteration. On the other hand, the generic workflow of LS-DEM is herein proposed to directly apply to superquadrics, considering true distance quantities and avoiding the need for an iterative procedure, outside the consideration of boundary nodes.

The LS-DEM description of a superellipsoid particle nevertheless logically shows a finite precision, with for instance the inertial quantities depending on the chosen resolution for the grid carrying ϕ, as per the above Section 3.1. Quantifying now the grid resolution as $r_g = 2\ \min(r_x, r_y, r_z)/g_{grid}$, Figure 7 then compares the obtained LS-DEM volume with the expected volume presented in Table 2. It shows that using at least ten grid cells per particle's length leads to satisfactory results with an error on the volume being smaller than few %. A similar precision is achieved for inertia components, as shown in Figure 15 in the Appendix. While this analysis is merely geometric in nature, a direct connection between errors in describing particles' volumes and bias in mechanical results was proposed by Mede et al. (2018) when using clumps.

The influence of grid spacing on inertial quantities directly relates to the voxellised nature of the present description of particle's volume, in connection with the sign of discrete ϕ-values $\phi(\vec{x}_i)$. Section 4.3 previously illustrated how the grid spacing also affects the precision in the actual values of those, after solving through a FMM the Eikonal equation. A last impact of grid spacing onto the LS-DEM precision exists through the tri-linear interpolation used to evaluate distance at any location other than a gridpoint, such as a boundary node for the purpose of contact detection. From the present and past results

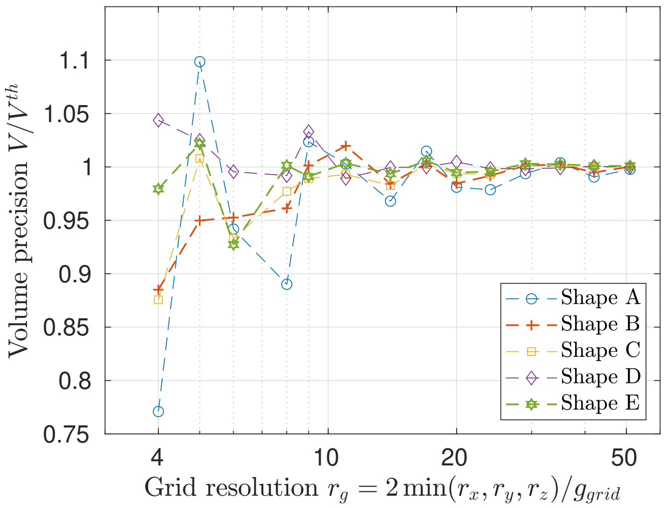


Figure 7: LS-DEM precision in describing superellipsoid volumes, with reference to Figure 6

(Duriez and Galusinski, 2020; Duriez and Bonelli, 2021), using r_g in the order of few tens (10 to 50) appears to be an adequate compromise between precision and computational (memory) costs, on all aspects.

5.3. Time costs in comparison with convex polyhedra

LS-DEM time costs are now briefly illustrated in comparison with the use of convex polyhedra as initially implemented in the YADE platform by Eliáš (2014). Describing such `Polyhedra` shapes in YADE relies on the CGAL library, used here in its 4.11 version (Kettner, 2018). That external library determines for instance a possible overlapping volume between two convex polyhedra for the purpose of contact treatment.

Such shapes may actually also apply to the present five superellipsoids, after locating the polyhedra's vertices along the superquadric surface. Previously determined LS-DEM boundary nodes (with $r_g = 50$) can be used for such a purpose. These vertices, through their connecting edges and plane portions (facets) making the polyhedra's surface, govern the precision in describing a superellipsoid shape even though, by the present construction, the obtained particles volumes are always smaller than the exact volumes of the considered (convex) superellipsoids. Figure 8 illustrates how the number of vertices controls the obtained volume and the necessity to use hundreds of polyhedra vertices in order to limit the error on the volume below few %.

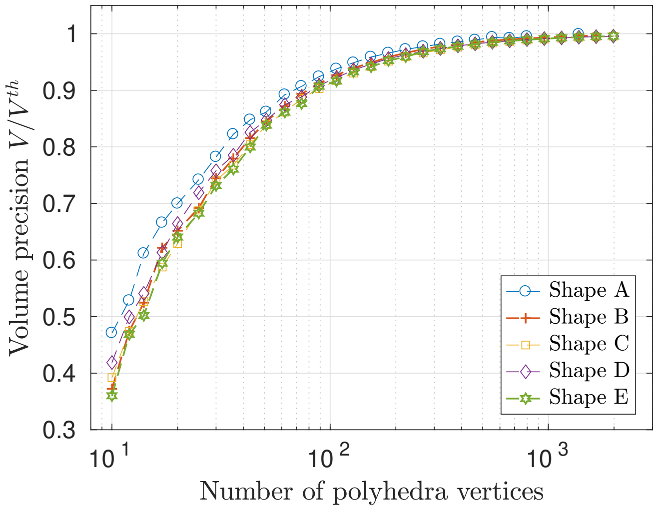


Figure 8: Precision in describing superellipsoid volumes when using convex polyhedra, with reference to Figure 6

Comparing the YADE use of `Polyhedra` or `LevelSet` shapes is actually not direct since the shape precision in LS-DEM both depends upon grid resolution and boundary nodes number, with associated computational costs being different in nature: memory requirements only (excluding time cost at DE creation) for the former, and time cost mostly for the latter. Convex polyhedra on the other hand are solely defined by their number of vertices N_v and show virtually no memory requirements. Also, the use of LS-DEM with N_n boundary nodes may more often miss contacts than the use of convex polyhedra with $N_v = N_n$, if one thinks e.g. to possible face-to-face contacts.

An imperfect comparison is still proposed looking at the time costs during YADE contact treatment in both approaches, i.e. the execution of `InteractionLoop` that embeds either the LS-DEM `Ig2_LevelSet_LevelSet_ScGeom` or the CGAL-enabled `Ig2_Polyhedra_Polyhedra_PolyhedraGeom` for polyhedra, both being responsible for virtually all time cost of each case. Looking at the lone pair of two fixed superquadrics illustrated in Figure 9, contact being detected in all cases, associated time costs are depicted according to boundary nodes or vertices numbers in Figure 10. Those sequential time costs are measured on the same workstation used in Section 2.2, repeating 3 times each case and excluding initialization costs that appear in particular at the first execution of `Ig2_Polyhedra_Polyhedra_PolyhedraGeom`. Corresponding scripts are provided at `lsYade/examples/levelSet/seContact.py` and `lsYade/examples/polyhedra/seContact.py`.

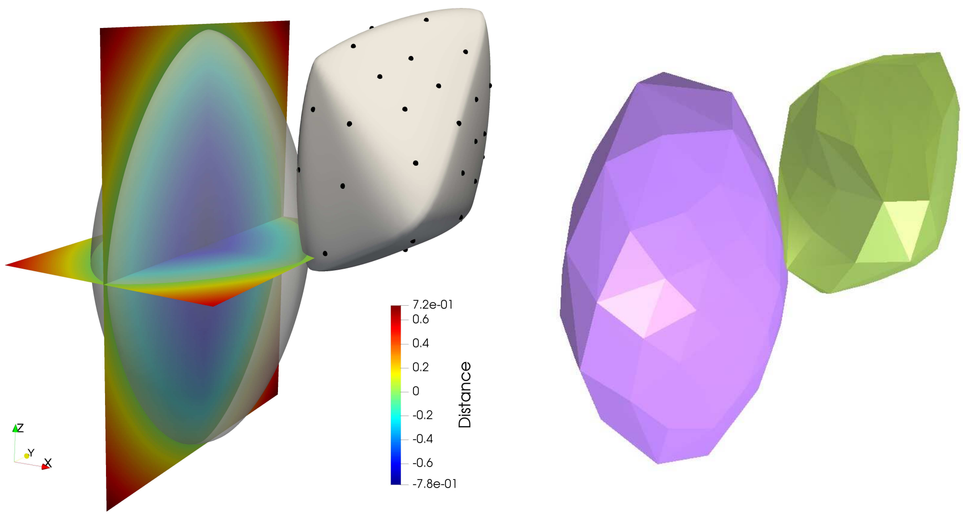


Figure 9: Two contacting superellipsoids described using LS-DEM (left, with 51 boundary nodes) or convex polyhedra (right, with 107 vertices per body)

LS-DEM timing data first show a logical proportionality between N_n and time cost t. Furthermore, the LS-DEM time cost is again shown to be interestingly insensitive to the grid resolution, even though the latter contributes

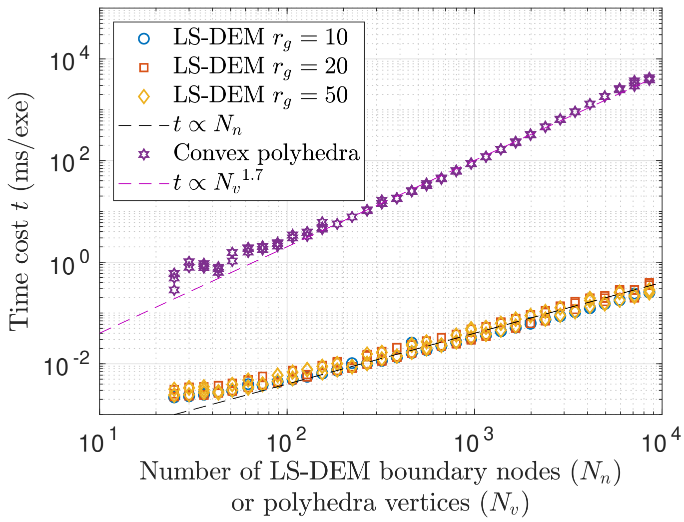


Figure 10: Time costs for computing the single contact of Figure 9 using LS-DEM or convex polyhedra, expressed in milliseconds per execution of `InteractionLoop` in one DEM iteration (see text)

to a greater precision. As for the use of convex polyhedra, the corresponding time cost appears as proportional to ${N_v}^{1.7}$, then close to $\mathcal{O}({N_v}^2)$. With N_v being checked to be itself proportional to the number of edges, N_e, or planar facets, N_f, making up each polyhedral surface, this $\mathcal{O}({N_v}^2) = \mathcal{O}({N_e}^2)$ time complexity is actually consistent with the consideration of all possible edge pairs adopted by Eliáš (2014) for that contact algorithm. Mostly, the polyhedral time cost is several orders of magnitude higher than its LS-DEM counterpart for $N_n = N_v$. In spite of the incomplete equivalence between N_n and N_v, these important differences in time costs clearly suggest LS-DEM might be lighter to use in terms of time, especially if a high fidelity is desired at the particle scale since this would here require hundreds of polyhedra vertices (Figure 8).

6. Discharge example

6.1. Simulation setup

A final illustration of LS-DEM is proposed in `examples/levelSet/discharge.py` as the discharge under gravity ($\vec{g} = -g\vec{z}$, with $g = 9.8\text{m/s}^2$) and into a rigid container ($L_x \times L_y \times L_z = 0.25^2 \times \infty$ m^3) of $n_{DE} = 1000$ superellipsoids with equal proportions of the previous five shapes A to E (Figure 11). Similar dynamic simulations could serve to study rock falls and slides up to an obstacle, or the angle of repose of granular geomaterials conveyed in industrial processes.

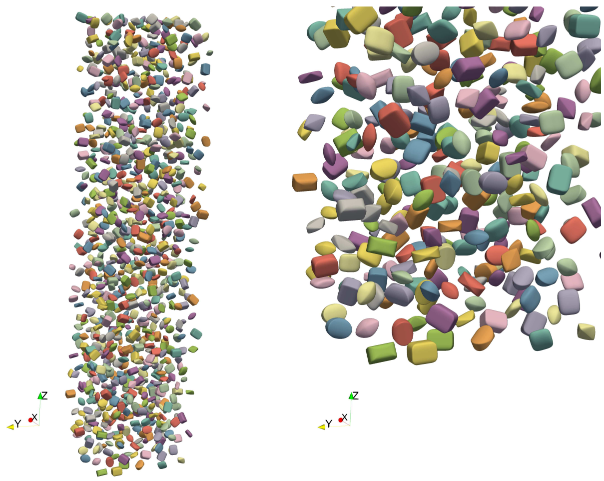


Figure 11: Views of the initial cloud of superellipsoids (left: in whole, right: close-up), with lateral and ground walls of the container not shown

In the present simulation, particles initially adopt random orientations and form a cloud with no contacts: initial porosity is $n_0 \approx 0.96$ in a $L_x \times L_y \times L_z = 0.23^2 \times 0.91$ m^3 volume. This initial set up is kept the same for all presented simulations. Table 3 lists the simulation's parameters, with contact parameters arbitrarily chosen among classical DEM choices, e.g. $k_n \in \{3 \times 10^4; 3 \times 10^6\}$ N/m in (Kawamoto et al., 2016, 2018).

Contact properties			Density	Timestep	Damping
k_n	k_t/k_n	μ	ρ	Δt	D
(N/m)	(-)	(-)	(kg/m^3)	(μs)	(-)
10^5	0.7	$\tan(25°)$ or 0 (lateral walls)	2650	25 $\approx 0.15\sqrt{\frac{m_{min}}{k_{max}}}$	0.3

Table 3: LS-DEM parameters for the discharge simulation

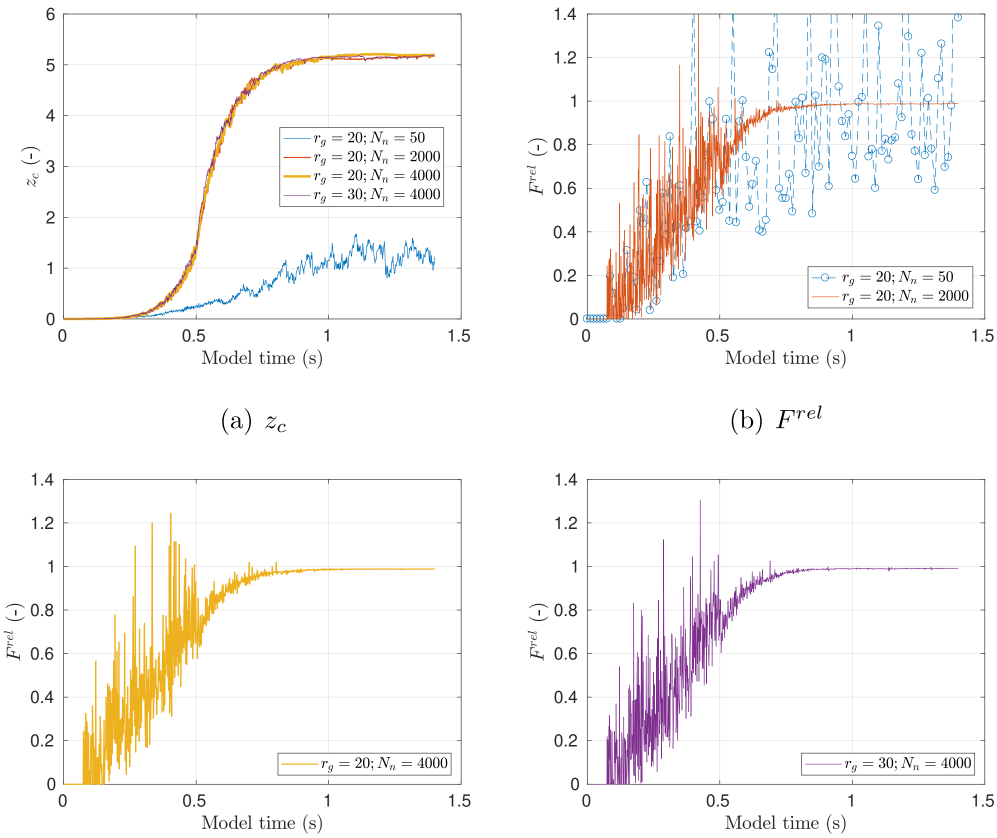
6.2. Results

After executing 56 000 DEM iterations over 1.4 s of model time, a final equilibrium state can be observed in Figure 12, for what concerns the average coordination number z_c or the vertical load exerted on the ground wall F, compared in a F^{rel} ratio with the expected weight F^{th} that corresponds to the theoretical solid volumes of all particles (Table 2):

$$F^{rel} = \frac{F}{\rho||\vec{g}|| \sum\limits_{i=1}^{n_{DE}} V^{th}(i)} \tag{34}$$

In this illustrative simulation, most dissipation of the initial gravitational energy is artificial, coming from the numerical damping mentioned in the above Section 3.3 used with $D = 0.3$.

Figure 12 also illustrates the possible influence of LS-DEM discretization parameters N_n and r_g. Using just $N_n = 50$ boundary nodes, together with $r_g = 20$, for instance prevents stabilization because contacts are hardly detected and too easily lost. On the other hand, choosing ($r_g = 20; N_n = 2000$) here appears as optimal since finer particle descriptions eventually lead to the same results though with higher computational costs, as discussed in the following.



(a) z_c

(b) F^{rel}

(c) F^{rel} (cont.)

(d) F^{rel} (cont.)

Figure 12: Dynamics of the discharge illustration (with only a fraction of datapoints for the $r_g = 20; N_n = 50$ case on (b), for readability)

6.3. Computational costs

Memory (RAM) requirements for LS-DEM simulations are first quantified calling the `resource.getrusage` Python function before and after defining all DEs. In accordance with the double precision of the present YADE simulations, used memory is verified in Figure 13 to follow (within a 15% margin) a 8 bytes requirement for each scalar value: one for the distance at each gridpoint and three for each boundary node (its coordinates). Significant memory costs are obtained, being in the order of MB per DE definition. Total values for the whole simulation in the four cases of Figure 12 are also listed in Table 4. It is to note though that those memory requirements could be reduced in the future, adopting octree structures to carry the distance field instead of regular grids (Duriez and Galusinski, 2020).

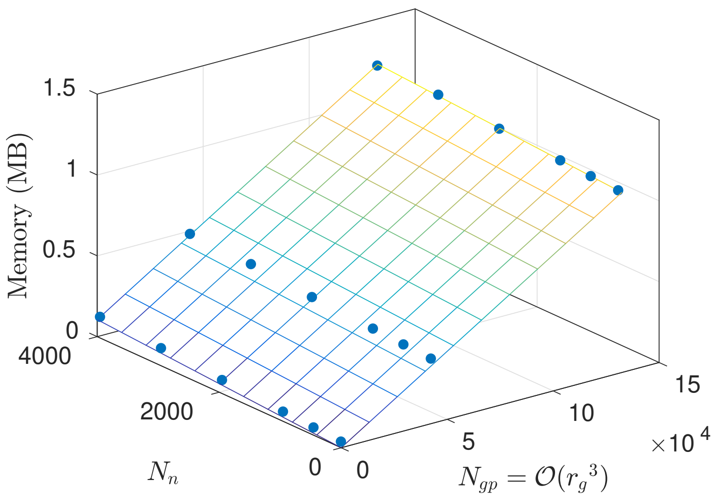


Figure 13: LS-DEM memory requirements per discrete element definition. Each data point is obtained from a different discharge simulation. The planar fit is colored according to memory (also on the z-axis) and is obtained after bilinear regression, following the expression $a \times N_{gp} + b \times N_n$ with $a = 0.0841 \times 10^{-4}$ MB and $b = 0.2637 \times 10^{-4}$ MB

As for execution time, users may expect from the previous Section 5.3

Simulation	RAM usage (MB)
$r_g = 20; N_n = 50$	578
$r_g = 20; N_n = 2000$	625
$r_g = 20; N_n = 4000$	672
$r_g = 30; N_n = 4000$	1391

Table 4: Total (whole simulation) RAM usage for the discharge simulations of Figure 12

lighter LS-DEM costs with respect to polyhedra, even though those costs would logically be even more reduced with ideal spherical shapes (Duriez and Bonelli, 2021). Time costs can anyway be significantly decreased using simple OpenMP parallel computing in a shared memory paradigm. Doing so, loops over interactions (for contact treatment in `InteractionLoop`) or bodies (for motion integration in `NewtonIntegrator`) are split into different OpenMP threads which are simultaneously executed by different CPU cores. With respect to the sequential case, additional supervisory operations become necessary in order to avoid simultaneous access to the same variable in memory from different threads. Nevertheless, OpenMP execution of the present discharge simulation, using the optimal choices $r_g = 20$ and $N_n = 2000$, appears as very beneficial, with a significant, linear and nearly optimal, speedup as depicted in Figure 14. Speedup is here measured repeating 3 times each parallel execution as well as the sequential one, on a server machine with two Intel Xeon Platinum 8270, 2.7 GHz, processors with 26 cores and 36 MB of cache memory each, i.e. a total of 52 cores and 104 threads, together with 1.5 TB 2.9 GHz RAM. From all these simulations, 9 parallel / sequential timing ratios are computed and depicted in Figure 14 through their average and standard deviation.

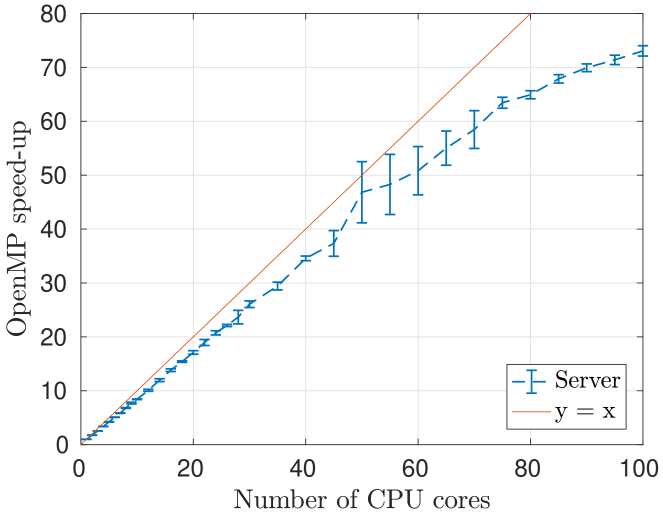


Figure 14: OpenMP scalability of the LS-DEM discharge simulation for $r_g = 20$ and $N_n = 2000$. Sequential time cost is 28696 s $\pm$ 106 s ($\approx$ 8 h) from average and standard deviation on 3 runs, being reduced to 392 s $\pm$ 5 s ($\approx$ 6.5 min) using 100 CPU cores

In the case of a quasistatic simulation being executed on the same machine, Duriez and Bonelli (2021) evidenced a linear behavior up to 50 cores approximately and with a corresponding speedup of more than 20, before that speedup may level off and even decrease.

7. Conclusions

Extending DEM for what concerns shape description, LS-DEM has been included in the YADE open-source platform for mechanical simulations of granular soils and other discrete systems. With distance-to-surface fields serving in a discrete fashion as a primary ingredient of the method, the proposed implementation also includes a Fast Marching Method to construct such fields for a wide class of surfaces with an analytical description. The versatility of the method is evident from the direct application to superellipsoids. On the other hand, significant computational costs are inherent to the method, be it in terms of memory or execution time. Time costs are nevertheless beneficial with respect to a polyhedral description of complex shapes, as already available in YADE, and they can be furthermore reduced through OpenMP parallel computing with a significant speed-up. As for the memory requirements, these could also decrease in the future using a more appropriate data structure than the current regular grid (Duriez and Galusinski, 2020).

Perspectives lie in user-friendly LS-DEM simulations in YADE for multiscale investigations in granular mechanics. A particular multiscale avenue is formed by the hierarchical modelling approaches where the DEM serves as an alternative to phenomenological (e.g. elasto-pastic) stress-strain constitutive relations in structure-scale FEM simulations (e.g. Guo and Zhao, 2014).

Appendix

Confirming the analysis made on volumes in Section 5.2 (Figure 7), Figure 15 illustrates how LS-DEM achieves to describe inertia components of superellipsoids with a very good precision, provided the grid resolution is fine enough i.e. includes more than 10 grid voxels per particle length.

Conflict of Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

Acknowledgements

The authors acknowledge financial support of the French Sud region to the LS-ENROC project; Stéphane Bonelli (INRAE, Aix Marseille Univ, RE-

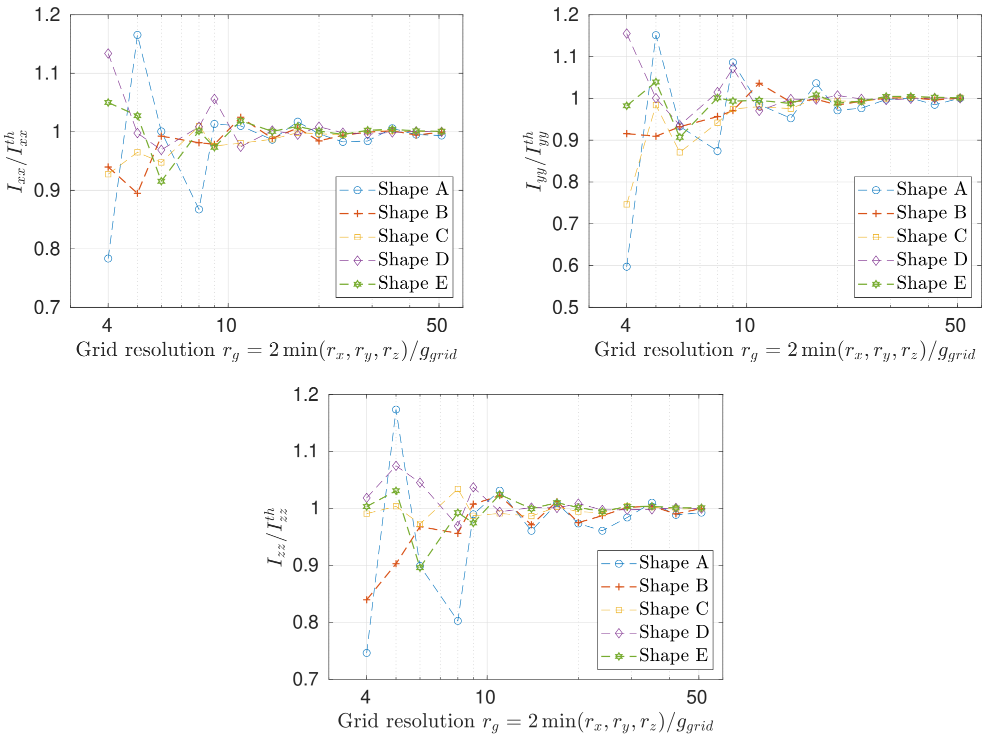


Figure 15: LS-DEM precision in describing inertia components for the five superellipsoids of Figure 6

COVER) and Frédéric Golay (Université de Toulon, IMATH) for fruitful discussions; as well as the mailing list of the French CNRS group in Scientific Computing ("Calcul"). Other YADE developers, passed and present (`yade-dev` GitLab team), are also acknowledged for setting up and maintaining the platform grounding the proposed implementation.

Computer code availability

The present LS-DEM code is released under the GNU General Public License v2. It has been developed by Jérôme Duriez (jerome.duriez@inrae.fr), the contacting author of the manuscript, and made first available in January 2021.

Source code can be currently found at `https://gitlab.com/jduriez/lsYade`. Insertion into the `master` branch of the YADE platform at `https://gitlab.com/yade-dev/trunk` is planned after publication, in addition to the classical deposit at `https://github.com/CAGEO`.

A bash script `install.sh` is for instance available at `https://gitlab.com/jduriez/lsYade` and in the manuscript submission, in order to download source code and trigger compilation. After a correct installation, executing `install/bin/yadelevelSet --check` should include `running: checkLS-dem.py [...] Status: success` in its output.

YADE LS-DEM simulations are realistically possible on computing-oriented, multi-core (clock speed higher than 2.5 GHz) personal desktops, with significant RAM: several tens of GB are for instance necessary for simulating Representative Elementary Volumes of granular soils with an adequate precision. Visualization of the simulations builds upon the free, open-source, Paraview software and its Python interface. Compilation dependencies include e.g. cmake, g++, boost, Qt, freeglut3, libQGLViewer, eigen, gdb, sqlite3, Loki, VTK, Python3 including numpy, sphinx, IPython, matplotlib on Ubuntu 18.04 or 20.04 (see the Prerequisites section of `install.sh`. Note that the Paraview Python interface is provided by paraview-python, resp. python3-paraview, package on Ubuntu 18.04, resp. 20.04).

References

Aboul Hosn, R., Sibille, L., Benahmed, N., Chareyre, B., 2017. Discrete numerical modeling of loose soil with spherical particles and interparticle rolling friction. Granular Matter 19 (4), 11–12.

Ayachit, U., 2019. The ParaView Guide. Kitware.

Barr, A. H., 1981. Superquadrics and angle-preserving transformations. IEEE Computer Graphics and Applications 1 (1), 11 – 23.
URL https://authors.library.caltech.edu/9756/1/BARieeecga81.pdf

Barr, A. H., 1995. Rigid physically based superquadrics. In: Kirk, D. (Ed.), Graphics Gems III. Academic Press, pp. 137–159.

Boon, C., Houlsby, G., Utili, S., 2012. A new algorithm for contact detection between convex polygonal and polyhedral particles in the discrete element method. Computers and Geotechnics 44, 73 – 82.

Boon, C., Houlsby, G., Utili, S., 2013. A new contact detection algorithm for three-dimensional non-spherical particles. Powder Technology 248, 94 – 102.

Cho, G.-C., Dodds, J., Santamarina, J. C., 2006. Particle shape effects on packing density, stiffness, and strength: Natural and crushed sands. Journal of Geotechnical and Geoenvironmental Engineering 132 (5), 591–602.

Cundall, P., Strack, O., 1979. A discrete numerical model for granular assemblies. Géotechnique 29, 47–65.

Deluzarche, R., Cambou, B., 2006. Discrete numerical modelling of rockfill dams. International Journal for Numerical and Analytical Methods in Geomechanics 30, 1075–1096.

Dubois, F., 2011. Numerical modeling of granular media composed of polyhedral particles. In: Radjai, F., Dubois, F. (Eds.), Discrete-element Modeling of Granular Materials. ISTE-Wiley, pp. 233–262.

Duriez, J., Bonelli, S., 2021. Precision and computational costs of Level Set-Discrete Element Method (LS-DEM) with respect to DEM. Computers and Geotechnics 134, 104033.

Duriez, J., Darve, F., Donzé, F.-V., 2011. A discrete modeling-based constitutive relation for infilled rock joints. International Journal of Rock Mechanics & Mining Sciences 48 (3), 458–468.

Duriez, J., Galusinski, C., 2020. Level set representation on octree for granular material with arbitrary grain shape. In: Šimurda, D., Bodnár, T. (Eds.), Proceedings Topical Problems of Fluid Mechanics 2020. Prague, pp. 64–71.
URL `http://www2.it.cas.cz/fm/im/im/proceeding/2020/9`

Duriez, J., Wan, R., Pouragha, M., Darve, F., 2018. Revisiting the existence of an effective stress for wet granular soils with micromechanics. International Journal for Numerical and Analytical Methods in Geomechanics 42 (8), 959–978.

Eliáš, J., 2014. Simulation of railway ballast using crushable polyhedral particles. Powder Technology 264, 458 – 465.

Garcia, X., Latham, J.-P., Xiang, J., Harrison, J., 2009. A clustered overlapping sphere algorithm to represent real particles in discrete element modelling. Géotechnique 59 (9), 779–784.

Gladkyy, A., Kuna, M., 2017. DEM simulation of polyhedral particle cracking using a combined Mohr–Coulomb–Weibull failure criterion. Granular Matter 19 (3), 41.

Guo, N., Zhao, J., 2013. The signature of shear-induced anisotropy in granular media. Computers and Geotechnics 47, 1–15.

Guo, N., Zhao, J., 2014. A coupled FEM/DEM approach for hierarchical multiscale modelling of granular media. International Journal for Numerical Methods in Engineering 99 (11), 789–818.

Hagenmuller, P., Chambon, G., Naaim, M., 2015. Microstructure-based modeling of snow mechanics: a discrete element approach. The Cryosphere 9 (5), 1969–1982.

Haustein, M., Gladkyy, A., Schwarze, R., 2017. Discrete element modeling of deformable particles in YADE. SoftwareX 6, 118 – 123.

Houlsby, G., 2009. Potential particles: a method for modelling non-circular particles in DEM. Computers and Geotechnics 36 (6), 953 – 959.

Kawamoto, R., Andò, E., Viggiani, G., Andrade, J. E., 2016. Level set discrete element method for three-dimensional computations with triaxial case study. Journal of the Mechanics and Physics of Solids 91, 1–13.

Kawamoto, R., Andò, E., Viggiani, G., Andrade, J. E., 2018. All you need is shape: Predicting shear banding in sand with LS-DEM. Journal of the Mechanics and Physics of Solids 111, 375–392.

Kettner, L., 2018. 3D polyhedral surface. In: CGAL User and Reference Manual, 4.11.3 Edition. CGAL Editorial Board.
URL http://doc.cgal.org/4.11.3/Manual/packages.html#PkgPolyhedronSummary

Li, L., Marteau, E., Andrade, J. E., 2019. Capturing the inter-particle force distribution in granular material using LS-DEM. Granular Matter 21 (3), 43.

Lin, C.-C., Ching, Y.-T., 1996. An efficient volume-rendering algorithm with an analytic approach. The Visual Computer 12 (10), 515–526.

Lorensen, W. E., Cline, H. E., 1987. Marching cubes: A high resolution 3D surface construction algorithm. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH 87. p. 163169.

Mede, T., Chambon, G., Hagenmuller, P., Nicot, F., 2018. A medial axis based method for irregular grain shape representation in DEM simulations. Granular Matter 20 (1), 16.

Miehe, C., Dettmar, J., Zäh, D., 2010. Homogenization and two-scale simulations of granular materials for different microstructural constraints. International Journal for Numerical Methods in Engineering 83 (8-9), 1206–1236.

Osher, S., Sethian, J. A., 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. Journal of Computational Physics 79 (1), 12– 49.

Pirnia, P., Duhaime, F., Ethier, Y., Dub, J.-S., 2019. ICY: An interface between COMSOL multiphysics and discrete element code YADE for the modelling of porous media. Computers & Geosciences 123, 38 – 46.

Podlozhnyuk, A., Pirker, S., Kloss, C., 2017. Efficient implementation of superquadric particles in Discrete Element Method within an open-source framework. Computational Particle Mechanics 4, 101–118.

Sethian, J., 1996. A fast marching level set method for monotonically advancing fronts. Proceedings of the National Academy of Sciences 93 (4), 1591–1595.

Sethian, J., 1999. Level set methods and fast marching methods. Cambridge University Press.

Vlahinić, I., Andò, E., Viggiani, G., Andrade, J. E., 2014. Towards a more accurate characterization of granular media: extracting quantitative descriptors from tomographic images. Granular Matter 16 (1), 9–21.

Šmilauer, V., et al., 2015. Yade Documentation 2nd ed. The Yade Project, http://yade-dem.org/doc/.

Wang, X., Tian, K., Su, D., Zhao, J., 2019. Superellipsoid-based study on reproducing 3D particle geometry from 2D projections. Computers and Geotechnics 114, 103131.

Weinhart, T., Orefice, L., Post, M., van Schrojenstein Lantman, M. P., Denissen, I. F., Tunuguntla, D. R., Tsang, J., Cheng, H., Shaheen, M. Y., Shi, H., Rapino, P., Grannonio, E., Losacco, N., Barbosa, J., Jing, L., Alvarez Naranjo, J. E., Roy, S., den Otter, W. K., Thornton, A. R., 2020. Fast, flexible particle simulations – an introduction to MercuryDPM. Computer Physics Communications 249, 107129.

Yang, L., Hyde, D., Grujic, O., Scheidt, C., Caers, J., 2019. Assessing and visualizing uncertainty of 3D geological surfaces using level sets with stochastic motion. Computers & Geosciences 122, 54 – 67.