



HAL
open science

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

Jérôme Duriez, Cédric Galusinski

► **To cite this version:**

Jérôme Duriez, Cédric Galusinski. A Level Set-Discrete Element Method in YADE for numerical, micro-scale, geomechanics with refined grain shapes. *Computers & Geosciences*, 2021, 157, pp.1-43/104936. 10.1016/j.cageo.2021.104936 . hal-03359788

HAL Id: hal-03359788

<https://hal.inrae.fr/hal-03359788>

Submitted on 30 Sep 2021

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

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

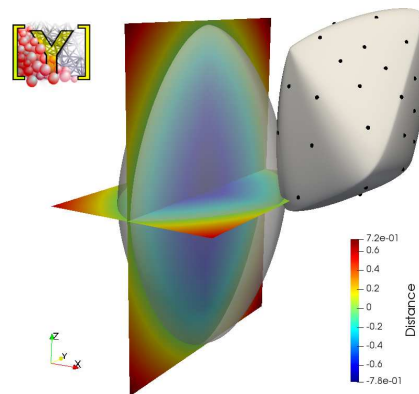


Distributed under a Creative Commons Attribution 4.0 International License

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)

¹Contribution: Writing - Original Draft, Software, Funding

²Contribution: 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

23 assumption (adopted e.g. by Duriez et al., 2011; Guo and Zhao, 2013; Duriez
24 et al., 2018; Aboul Hosn et al., 2017), since such spheres constitute a very
25 strong simplification of material particles while particle’s shape has a known
26 influence on the macroscopic behavior (e.g. Cho et al., 2006). Therefore, vari-
27 ous DEM strategies towards a better shape description have been introduced,
28 such as the use of rigid aggregates of spheres, so-called clumps, that should
29 mimic real shapes (e.g. Garcia et al., 2009; Mede et al., 2018); or the di-
30 rect consideration of polyhedra (e.g. Eliáš, 2014; Gladkyy and Kuna, 2017).
31 Clumps offer the advantage to accommodate straightforward and compu-
32 tationally cheap contact algorithms designed for spheres, but still present
33 some unrealistic local roundness. On the other hand, polyhedra resort to
34 more complex algorithms, which still remain restricted, most often, to con-
35 vex surfaces (Dubois, 2011). One can also note the potential particles (PP)
36 or potential blocks (PB) approaches by Houlsby (2009); Boon et al. (2012,
37 2013), which both describe particles’ surfaces resorting to the zero-level of
38 a so-called potential. Each scalar potential is given in a set of closed-form
39 expressions with a variable number of shape parameters, leading to rounded
40 (PP-case) or angular (PB-case) surfaces that are necessarily convex. Then,
41 the so-called Level Set Discrete Element Method (LS-DEM) has been recently
42 proposed by Kawamoto et al. (2016), in 3D, as another DEM extension to-
43 wards realistic shapes. In LS-DEM, and with a limited similarity to PP and
44 PB approaches, every DE’s surface is implicitly described as the zero-level
45 set of the specific signed distance function to that surface. Contributing to
46 its generality, no closed-form equation or convexity assumptions are required
47 in LS-DEM since Level Set and Fast Marching Methods (Osher and Sethian,
48 1988; Sethian, 1996, 1999) are available to construct distance fields for ar-
49 bitrary, possibly concave, surfaces and a wide class of scientific applications

50 (e.g. Yang et al., 2019). Kawamoto et al. (2016, 2018) actually illustrated
51 the capabilities of the LS-DEM to describe real soil grains, shapes being ac-
52 quired through X-ray computed tomography, as well as its promising features
53 to reproduce observed behaviors both qualitatively and quantitatively.

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

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

69 **2. Level Set and Fast Marching methods**

70 *2.1. Level set formalism*

71 Level set approaches (Osher and Sethian, 1988; Sethian, 1999) see in-
72 terfaces $\mathcal{S}(t)$ as the zero-level set of a function $\phi^t(\vec{x}, t)$ being defined from
73 $\mathbb{R}^d \times \mathbb{R}$ into \mathbb{R} , with \mathbb{R}^d covering the whole space of a d dimensionnality.
74 Evolving contours (resp. surfaces) can then be described for $d = 2$ (resp.
75 $d = 3$). While the interfaces evolve, propagating with a normal velocity

76 $\vec{v} = F(\vec{x}, t) \vec{n}$ (where \vec{n} stands for the outwards normal), all level sets evolve
77 with an extended velocity parallel to the gradient of the level set function
78 $\phi^t(\vec{x}, t)$. Since $\phi^t(\vec{x}, t)$ is constant along $\mathcal{S}(t)$ (equal to 0 $\forall t$), the nullity of
79 the material derivative along the interface front leads to the following level
80 set equation:

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

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

$$H^t(\phi_x^t) = F |\phi_x^t| \quad \text{for } d = 1 \quad (2)$$

$$H^t(\phi_x^t, \phi_y^t) = F \sqrt{\phi_x^{t2} + \phi_y^{t2}} \quad \text{for } d = 2 \quad (3)$$

$$H^t(\phi_x^t, \phi_y^t, \phi_z^t) = F \sqrt{\phi_x^{t2} + \phi_y^{t2} + \phi_z^{t2}} \quad \text{for } d = 3 \quad (4)$$

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

86 The signed (shortest) distance to $\mathcal{S}(t)$ is a typical choice for the function
87 ϕ^t , with the convention of a negative, resp. positive, distance when being
88 inside, resp. outside, of $\mathcal{S}(t)$. Doing so, and for a constant and uniform speed
89 $F(\vec{x}, t) = F$, one can relate ϕ^t to $T(\vec{x})$, the arrival time of \mathcal{S} at \vec{x} :

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

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

$$F \|\vec{\nabla} T\| = 1 \quad (6)$$

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

95 Doing so, the current interface $\mathcal{S}(t)$ is the t -level set of T and no time variable
 96 enters the partial differential equation (6). That stationary perspective can
 97 be finally complemented by the consideration of $\phi(\vec{x})$, the distance to $\mathcal{S}(t =$
 98 $0)$:

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

99 with the following form for the Eikonal equation:

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

100 Similar to Eqs. (1)-(4), Eq. (8) can be cast in the form of a Hamiltonian
 101 $H(\phi_x, \dots) = 1$ with:

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

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

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

102 2.2. A Fast Marching Method for the stationary perspective

103 Looking for the distance field ϕ to a given, constant, surface \mathcal{S} , the Eikonal
 104 equation (8) can be efficiently solved using a so-called Fast Marching Method
 105 (FMM, Sethian, 1996, 1999). Space being discretized on a grid, the Eikonal
 106 equation makes the ϕ -value at some gridpoint \vec{x}_i being directly dependent
 107 upon surrounding ϕ -values at adjacent gridpoints, as can be seen from finite
 108 difference expressions for the spatial derivatives in Eqs. (9) to (11). Account-
 109 ing for the monotonous nature of ϕ , which strictly increases (in absolute
 110 value) when \vec{x} goes away of \mathcal{S} , the FMM eventually gives the full discrete
 111 field $\phi(\vec{x}_i)$ starting from an initial set of gridpoints being along, or close to,
 112 the surface and serving as boundary conditions. In more details, the FMM
 113 recursively applies Eq. (8), in the form of Eqs. (9) or (10) or (11) depending
 114 on gradient's dimensionality, and adopting gradient expressions decentred

115 to low and known ϕ -values. Recursive applications actually go in a down-
 116 wind direction away from the surface, until the whole spatial grid has been
 117 handled. The key point of the FMM is to go through the grid points in the
 118 right order, following at each step the minimal value of distance.

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

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

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

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

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

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

135 3. LS-DEM formulation

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

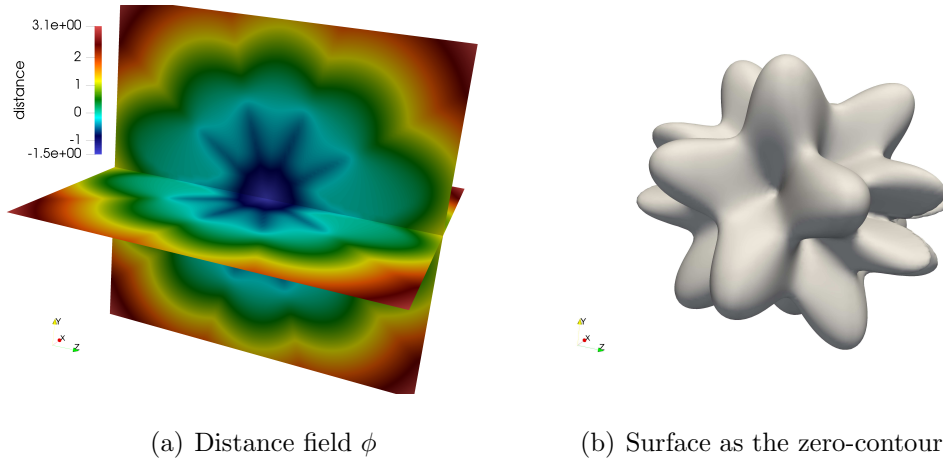


Figure 1: Level Set description of a flake-like surface defined by Eq. (13), with $(R; \Delta R) = (3; 1.5)$, after executing a FMM on a 0.1-spaced isotropic grid

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

141 3.1. LS-DEM shape description

142 Following Kawamoto et al. (2016, 2018), a discrete signed distance field
 143 on a body-centered regular grid, possibly obtained from the previous FMM,
 144 is the first LS-DEM ingredient. That (grid ; distance field) pair is in-
 145 dependently defined for every DE in a local coordinate system and first
 146 serves for defining the DE inertial quantities (mass m and inertia matrix
 147 $\mathbf{I} = I_{\alpha\beta}, \alpha, \beta \in \{x, y, z\}$) summing contributions from grid voxels v making

148 up the body's volume V as per the following discrete-form equations:

$$m = \rho \sum_{v \in V} V_v = \rho N_{vox} V_v \quad (14)$$

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

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

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

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

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

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

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

149 In the above equations, ρ is the material mass density, $\vec{x}_v = (x_v, y_v, z_v)$
 150 the middle point of a voxel and $\vec{x} = (x, y, z)$ the body's center of mass. Eqs.
 151 (15),(19)-(21) serve for verification purposes since the body-attached local
 152 frame is expected to be inertial and $\vec{x}, I_{xy}, I_{xz}, I_{yz}$ to be nil. By a simple
 153 convention, a grid voxel v of volume V_v is herein said to be part of the
 154 body's volume V when its lowest corner is inside the surface, showing a zero
 155 or negative distance value. While smoother choices have been proposed by
 156 Kawamoto et al. (2016, 2018), it will be verified in Section 5.2 that the present
 157 choice does not inhibit precision for grids being fine enough, i.e. showing a
 158 spacing g_{grid} at least ten times smaller than a grain's characteristic size l_{grain} .

159 For the purpose of LS-DEM contact algorithms that will be described in
 160 Section 3.2 below, a second LS-DEM ingredient adds to the distance field, in
 161 the form of a set of N_n boundary nodes $\{N_i, i \in [0; N_n - 1]\}$ discretizing each
 162 body's surface \mathcal{S} . Generally speaking, boundary nodes should count in the

163 order of thousands and their positions are defined at the intersection of \mathcal{S} i.e.
 164 $\phi(\vec{x}) = 0$ and N_n half-lines i.e. rays $\lambda\vec{v}$, with \vec{v} a direction and λ a positive
 165 abscissa, that stem from the center of mass. Due to the adopted tri-linear
 166 interpolation of the discrete distance field within the grid extents, $\phi(\vec{x} = \lambda\vec{v})$
 167 is a cubic polynomial in λ whose coefficients depend upon grid distance val-
 168 ues and ray tracing boundary nodes corresponds to solve its positive roots
 169 (see e.g. Lin and Ching, 1996). Since rays should provide an appropriate dis-
 170 cretization of spherical angles (θ, φ) , the corresponding directions (θ, φ) are
 171 chosen to follow a spiral path instead of a simple rectangular discretization of
 172 $[0; \pi] \times [0; 2\pi]$ in order to avoid a possible (shape-dependent) concentration
 173 of nodes at the poles $\theta = 0[\pi]$. More details about the spiral path or the
 174 choice of boundary nodes number are given by Duriez and Bonelli (2021) and
 175 in the next sections.

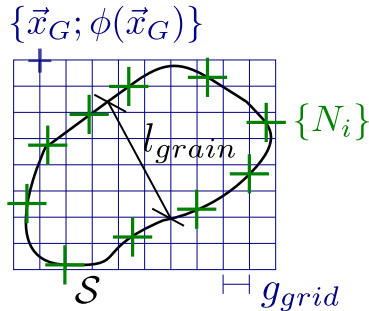


Figure 2: Shape description in LS-DEM: a regular grid $\{\vec{x}_G\}$ carrying the distance field ϕ , together with boundary nodes $\{N_i\}$ (2D view for clarity)

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

180 *3.2. Contact law*

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

$$u_n = -\min(\phi_2(\overrightarrow{ON}_i), \overrightarrow{ON}_i \in \mathcal{S}_1) = -\phi_2(\overrightarrow{ON}_c) \geq 0 \quad (22)$$

$$\vec{n} = \vec{\nabla}\phi_1(\overrightarrow{ON}_c) \quad (23)$$

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

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

$$\vec{F}_n = k_n u_n \vec{n} \quad (24)$$

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

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

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

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

215 3.3. Motion integration

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

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

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

220 The above Newton-Euler equations are classically damped using a numerical
 221 coefficient D , which modifies the resultant force and torque so that kinetic

222 energy always decreases (or is led to increase by a smaller extent) as soon as
 223 $\vec{F} \neq \vec{0}$. Eq. (28), relating the variation in $\vec{\omega}$ with $\vec{\Gamma}$, is expressed in local axes
 224 where the inertia tensor \mathbf{I} is constant. Denoting $\mathbf{R}(t)$ the rotation matrix
 225 passing from local axes to global ones, and $\mathbf{\Omega}$ the antisymmetric matrix such
 226 that $\mathbf{\Omega} \vec{x} = \vec{\omega} \wedge \vec{x}, \forall \vec{x}$, Eq. (28) is finally supplemented with:

$$\frac{d\mathbf{R}}{dt} = \mathbf{R}\mathbf{\Omega} \quad (29)$$

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

230 4. Proposed implementation

231 4.1. Source code

232 The present C++ and Python implementation inserts LS-DEM into the
 233 2020.01a version, i.e. the `git` commit 9964f53, of the YADE platform (Šmilauer
 234 et al., 2015). Figure 3 illustrates the LS-DEM workflow exposed in the previ-
 235 ous section together with the most noticeable new (or modified) C++ classes
 236 responsible for execution.

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

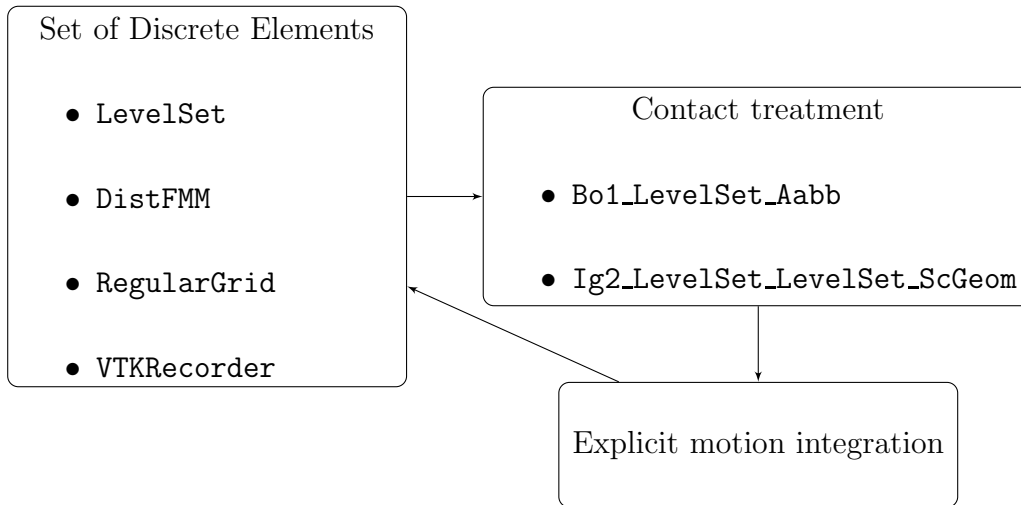


Figure 3: New or modified C++ classes for a LS-DEM workflow in YADE

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

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

263 Visualization of `LevelSet`-shaped bodies relies on `vtk` exports of the dis-
264 crete distance field for each DE in current configuration, thanks to a modified
265 version of files `pkg/dem/VTKRecorder.*pp`. Actual display is typically done
266 from Paraview software (Ayachit, 2019), using its Python interface and a pro-
267 vided `pvVisu` function defined in `examples/levelSet/pvVisu.py`. For the
268 purpose of alternate visualization methods at the user’s discretion, a `Lev-`
269 `elSet.marchingCubes` method is also available from YADE interface and
270 gives a triangulated description of a particle’s surface as per the Marching
271 Cubes algorithm (Lorensen and Cline, 1987).

272 The files `pkg/dem/LevelSetInteraction.*pp` finally implement the con-
273 tact algorithms described in previous Section 3.2. The `Bo1_LevelSet_Aabb`
274 is first responsible to compute an axis-aligned bounding box (`Aabb`) used
275 in YADE for a first, crude and fast, detection of possible contacts. At the
276 beginning of a simulation, that class first loops over the whole distance field
277 to compute the 8 corners of the corresponding `Aabb` in local axes, stored
278 in `LevelSet.corners`. Then, the current `Aabb` in model (global) axes is
279 easily determined following rigid transformations. In case of overlap between
280 `Aabb`, precise contact detection subsequently resorts to the other class `Ig2_-`
281 `LevelSet_LevelSet_ScGeom`, that implements the master-slave contact de-
282 tection based on boundary nodes, identifying the contact point, if any, and
283 the associated kinematic variables (normal vector, interpenetration depth)
284 between two `LevelSet`-shaped bodies. Similar classes enable contact inter-
285 action between a `LevelSet`-shaped body and existing `Wall` or `Box` shapes,
286 often adopted in YADE to simulate rigid boundaries.

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

290 also correspond to the pre-existing `FriictPhys` contact model in YADE. This
291 enables LS-DEM simulations to adopt a pre-existing contact law, namely
292 `Law2_ScGeom_FriictPhys_CundallStrack`. Motion integration as described
293 in previous Section 3.3 has also been readily available from the `NewtonIn-`
294 `tegrator` class.

295 *4.2. Code usage*

296 The (modified or classical) YADE platform starts in the form of a Python3
297 interactive interface, invoked from `install/bin/yadelevelSet` in the pro-
298 posed installation procedure (see the “Computer code availability” section).
299 Instead of an interactive session, scripts prepared beforehand can be as
300 well passed as argument and launched in the same manner than classical
301 Python scripts. Examples of YADE scripts using the new LS-DEM fea-
302 tures can be found in the source code at `lsYade/examples/levelSet/*.py`
303 (`levelSetBody.py` in particular) and also `lsYade/scripts/checks-and-`
304 `tests/checks/checkLSdem.py`. The latter actually serves as a new regres-
305 sion test into the YADE platform (Haustein et al., 2017), to insure stability of
306 the LS-DEM features in the future. These examples illustrate the definition
307 of `LevelSet` bodies through a new `levelSetBody()` YADE function. That
308 function proposes level set descriptions of pre-defined analytical shapes (from
309 boxes and spheres to superellipsoids, see next Section 5), together with the
310 possibility of a direct assignment of the regular grid with its distance field.
311 The latter enables users to directly insert any distance field they would have
312 otherwise acquired, for instance from computed tomography (Vlahinić et al.,
313 2014). In all cases, grid spacing g_{grid} is input through a `spacing` attribute
314 while a `nNodes` attribute of `levelSetBody()` controls the boundary nodes
315 number N_n .

316 Documentation can be obtained for any class or attribute in the usual

317 interactive Python manner, typing e.g. `LevelSet?` or `levelSetBody?`. An
 318 HTML version of the documentation can also be built executing `make doc`
 319 from the compilation folder.

320 4.3. Code validation

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

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

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

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

$$\|\vec{\nabla}\phi\| = \|\vec{\nabla}f\| \quad (32)$$

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

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

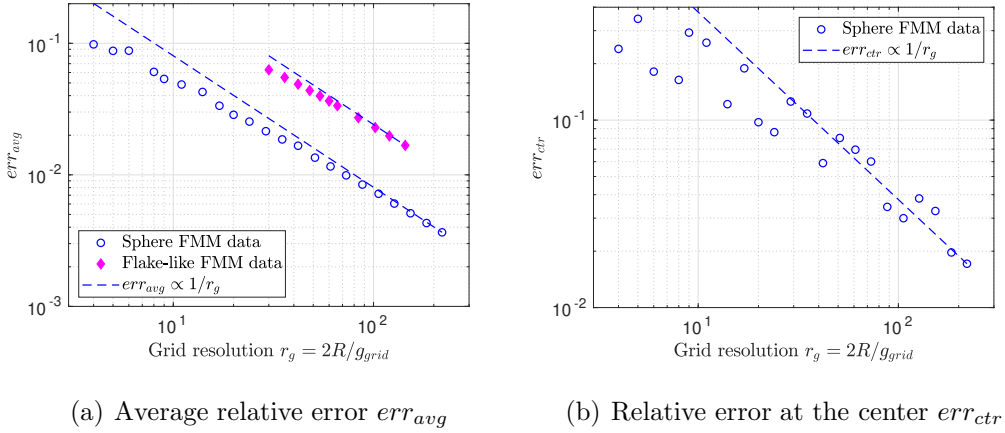


Figure 4: Influence of the grid resolution on the FMM precision, with reference to spherical or flake-like (Figure 1) surfaces

339 Associated time costs are depicted in Figure 5. They refer to distance
340 computations, i.e. solving Eq. (8) only, as per the present sequential FMM
341 being executed on a workstation having one 4 cores, 8 threads, Intel i7-
342 7700, 0.8 - 4.2GHz processor with 8 MB of cache memory, as well as 64
343 GB of 2.4 GHz RAM. Each case is run between 3 and 9 times (all depicted
344 on the Figure) to account for possible variations in time cost, and after
345 using the Linux command `cpufreq` and its `performance` governor set at 4.0
346 GHz. Denoting N_{gp} the total number of gridpoints, with $N_{gp} = \mathcal{O}(r_g^3)$, a
347 $\mathcal{O}(r_g^6) = \mathcal{O}(N_{gp}^2)$ complexity appears, in accordance with classical Level
348 Set Methods. Sethian (1996) actually proposed a lighter complexity for the
349 FMM, through adopting a heap sort when searching the minimum ϕ -value
350 for propagating the distance field. For the purpose of LS-DEM, the FMM
351 will apply only once per DE, at the very beginning of a simulation and the
352 present time cost in the order of a second for few tens of grid voxel per particle
353 length is actually acceptable, considering the final time cost of a complete
354 LS-DEM simulation. Figure 5 finally illustrates that the FMM computation

355 of distance for the more complex, non-spherical, flake-like shape logically
 356 shows the same time costs.

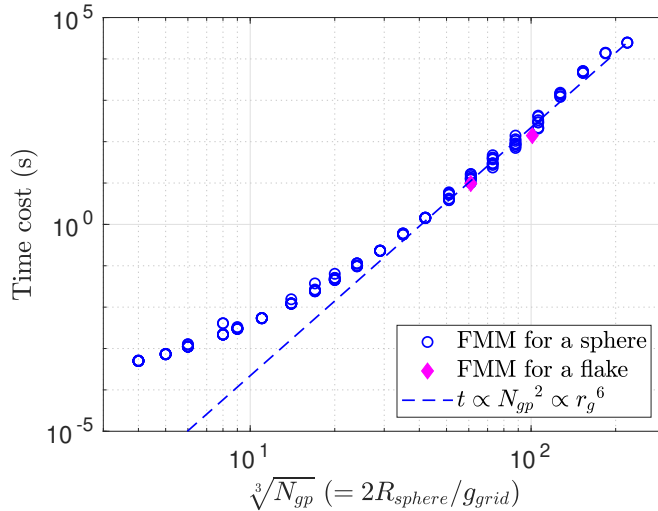


Figure 5: Time cost of the FMM according to the number of gridpoints per space axis $\sqrt[3]{N_{gp}}$, with N_{gp} the total number

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

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

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

368 *5.1. Superellipsoids surfaces*

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

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

374 Figure 6 illustrates five different superellipsoids, with their corresponding
 375 shape parameters presented in Table 1. Table 2 also details their volume and
 376 inertia properties, as obtained from closed form expressions given by Barr
 377 (1995). One can here observe how the ϵ_n exponent modifies the z -variation of
 378 cross-sections in (x, y) planes. For instance, adopting $\epsilon_n \rightarrow 0$ induces fairly
 379 constant cross-sections and a wider distribution of matter for extreme values
 380 along the “north-south” axis \vec{z} , see Shapes A or C. On the other hand, the
 381 $\epsilon_n = 1$ case corresponds to a rounded variation of these cross sections when
 382 progressing along \vec{z} (Shape B). Some singularity, i.e. a sharpness at $z = 0$,
 383 would appear for $\epsilon_n \geq 2$, alongside concavity in a plane tangent to \vec{z} for
 384 $\epsilon_n > 2$. For a given ϵ_n , ϵ_e controls the contour’s roundness in the (x, y) plane
 385 of these cross-sections. While $\epsilon_e = 1$ corresponds to perfectly round (circles
 386 or ellipses) contours, decreasing ϵ_e towards 0 induce edges that tend to align
 387 with the \vec{x} and \vec{y} axes, see Shape A vs C. Alternate edges and sharpnesses
 388 would be obtained in the (x, y) plane at $\epsilon_e = 2$, just before concavity in that
 389 plane, for $\epsilon_e > 2$.

390 *5.2. LS-DEM description of superellipsoids*

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

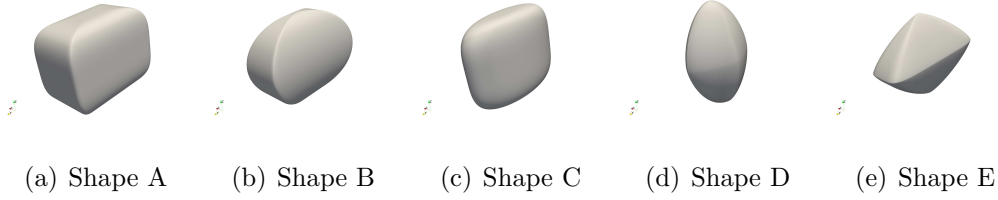


Figure 6: Five possible superquadric shapes

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_{xx}^{th}/ρ	I_{yy}^{th}/ρ	I_{zz}^{th}/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)

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

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

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

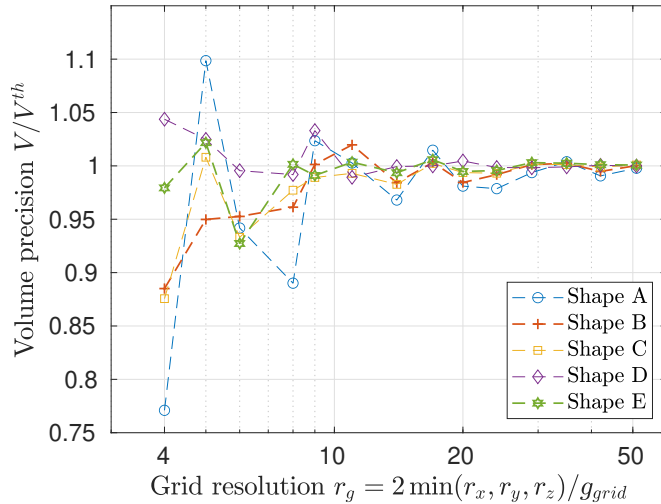


Figure 7: LS-DEM precision in describing superellipsoid volumes, with reference to Figure 6

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

423 5.3. Time costs in comparison with convex polyhedra

424 LS-DEM time costs are now briefly illustrated in comparison with the use
 425 of convex polyhedra as initially implemented in the YADE platform by Eliáš
 426 (2014). Describing such `Polyhedra` shapes in YADE relies on the CGAL
 427 library, used here in its 4.11 version (Kettner, 2018). That external library
 428 determines for instance a possible overlapping volume between two convex
 429 polyhedra for the purpose of contact treatment.

430 Such shapes may actually also apply to the present five superellipsoids,
 431 after locating the polyhedra’s vertices along the superquadric surface. Pre-
 432 viously determined LS-DEM boundary nodes (with $r_g = 50$) can be used for
 433 such a purpose. These vertices, through their connecting edges and plane

434 portions (facets) making the polyhedra's surface, govern the precision in de-
 435 scribing a superellipsoid shape even though, by the present construction, the
 436 obtained particles volumes are always smaller than the exact volumes of the
 437 considered (convex) superellipsoids. Figure 8 illustrates how the number of
 438 vertices controls the obtained volume and the necessity to use hundreds of
 439 polyhedra vertices in order to limit the error on the volume below few %.

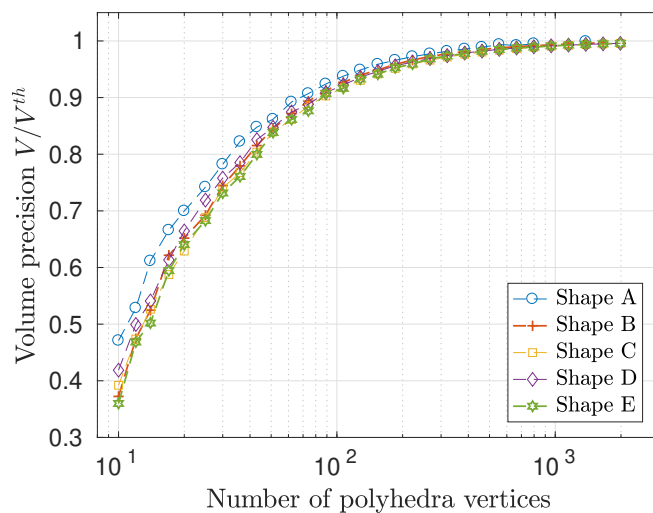


Figure 8: Precision in describing superellipsoid volumes when using convex polyhedra, with reference to Figure 6

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

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

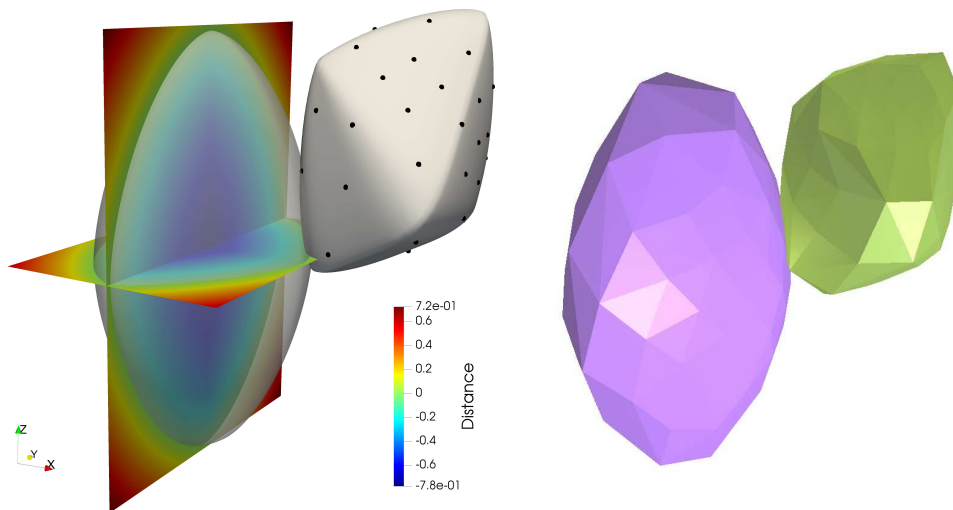


Figure 9: Two contacting superellipsoids described using LS-DEM (left, with 51 boundary nodes) or convex polyhedra (right, with 107 vertices per body)

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

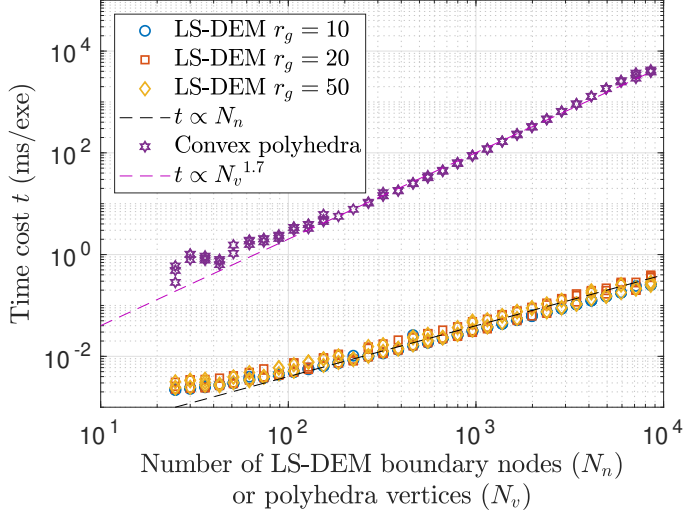


Figure 10: Time costs for computing the single contact of Figure 9 using LS-DEM or convex polyhedra, expressed in milliseconds per execution of `InteractionLoop` in one DEM iteration (see text)

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

477 **6. Discharge example**

478 *6.1. Simulation setup*

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

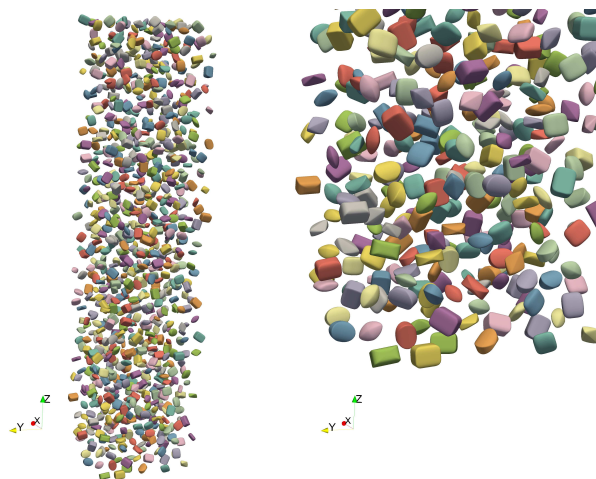


Figure 11: Views of the initial cloud of superellipsoids (left: in whole, right: close-up), with lateral and ground walls of the container not shown

486 In the present simulation, particles initially adopt random orientations
487 and form a cloud with no contacts: initial porosity is $n_0 \approx 0.96$ in a $L_x \times$
488 $L_y \times L_z = 0.23^2 \times 0.91 \text{ m}^3$ volume. This initial set up is kept the same
489 for all presented simulations. Table 3 lists the simulation's parameters, with
490 contact parameters arbitrarily chosen among classical DEM choices, e.g. $k_n \in$
491 $\{3 \times 10^4; 3 \times 10^6\} \text{ N/m}$ 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)	(μ s)	(-)
10^5	0.7	$\tan(25^\circ)$ 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

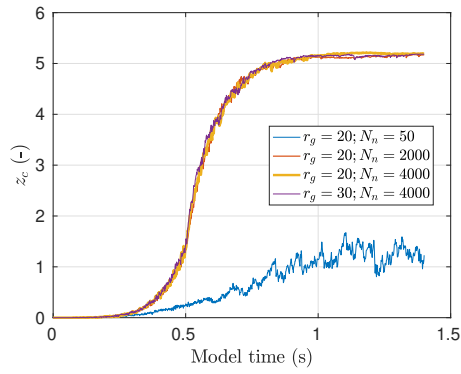
492 6.2. Results

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

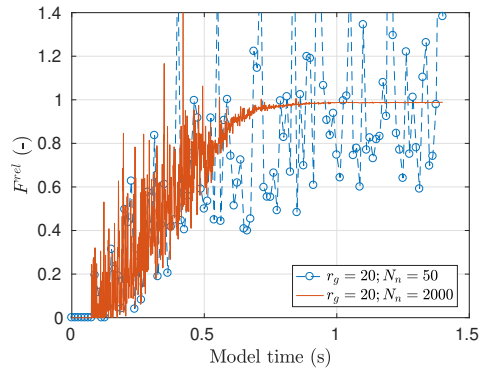
$$F^{rel} = \frac{F}{\rho \|\vec{g}\| \sum_{i=1}^{n_{DE}} V^{th}(i)} \quad (34)$$

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

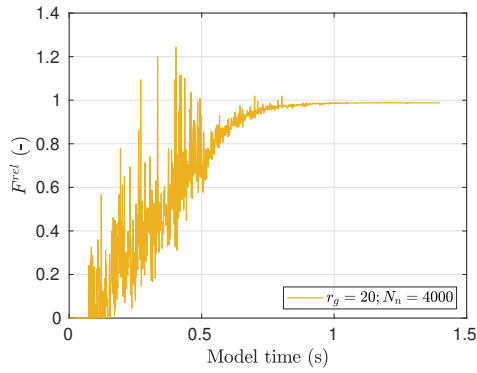
501 Figure 12 also illustrates the possible influence of LS-DEM discretization
502 parameters N_n and r_g . Using just $N_n = 50$ boundary nodes, together with
503 $r_g = 20$, for instance prevents stabilization because contacts are hardly de-
504 tected and too easily lost. On the other hand, choosing ($r_g = 20$; $N_n = 2000$)
505 here appears as optimal since finer particle descriptions eventually lead to
506 the same results though with higher computational costs, as discussed in the
507 following.



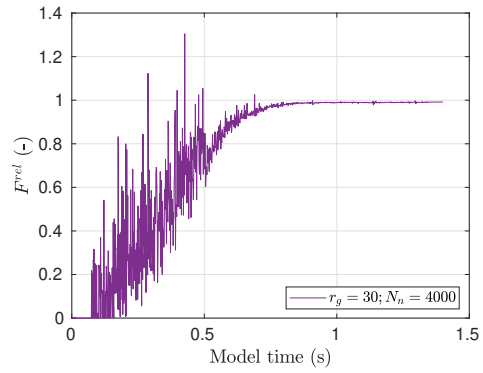
(a) z_c



(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)

508 *6.3. Computational costs*

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

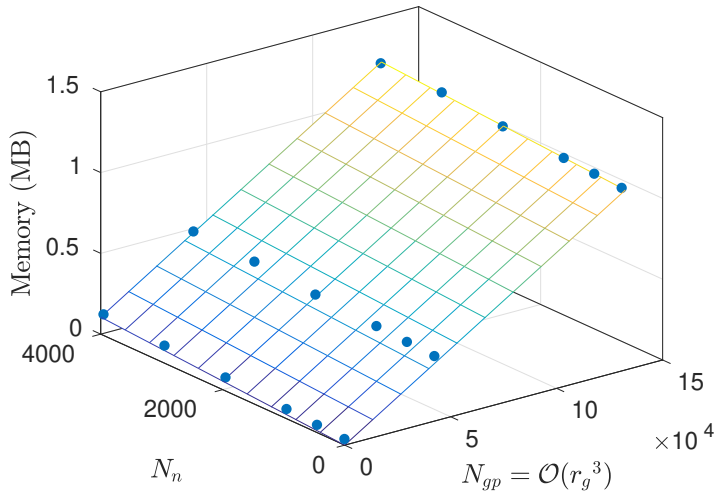


Figure 13: LS-DEM memory requirements per discrete element definition. Each data point is obtained from a different discharge simulation. The planar fit is colored according to memory (also on the z -axis) and is obtained after bilinear regression, following the expression $a \times N_{gp} + b \times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

520 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

521 lighter LS-DEM costs with respect to polyhedra, even though those costs
522 would logically be even more reduced with ideal spherical shapes (Duriez
523 and Bonelli, 2021). Time costs can anyway be significantly decreased using
524 simple OpenMP parallel computing in a shared memory paradigm. Doing
525 so, loops over interactions (for contact treatment in `InteractionLoop`) or
526 bodies (for motion integration in `NewtonIntegrator`) are split into differ-
527 ent OpenMP threads which are simultaneously executed by different CPU
528 cores. With respect to the sequential case, additional supervisory opera-
529 tions become necessary in order to avoid simultaneous access to the same
530 variable in memory from different threads. Nevertheless, OpenMP execution
531 of the present discharge simulation, using the optimal choices $r_g = 20$ and
532 $N_n = 2000$, appears as very beneficial, with a significant, linear and nearly
533 optimal, speedup as depicted in Figure 14. Speedup is here measured re-
534 peating 3 times each parallel execution as well as the sequential one, on a
535 server machine with two Intel Xeon Platinum 8270, 2.7 GHz, processors with
536 26 cores and 36 MB of cache memory each, i.e. a total of 52 cores and 104
537 threads, together with 1.5 TB 2.9 GHz RAM. From all these simulations, 9
538 parallel / sequential timing ratios are computed and depicted in Figure 14
539 through their average and standard deviation.

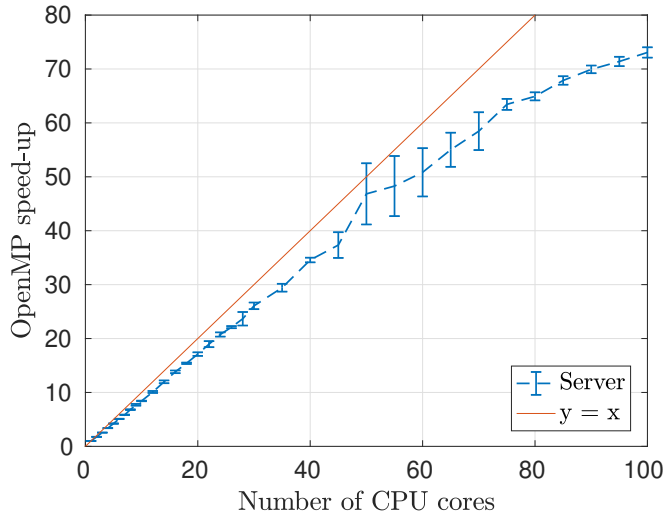


Figure 14: OpenMP scalability of the LS-DEM discharge simulation for $r_g = 20$ and $N_n = 2000$. Sequential time cost is $28696 \text{ s} \pm 106 \text{ s}$ ($\approx 8 \text{ h}$) from average and standard deviation on 3 runs, being reduced to $392 \text{ s} \pm 5 \text{ s}$ ($\approx 6.5 \text{ min}$)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

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

544 7. Conclusions

545 Extending DEM for what concerns shape description, LS-DEM has been
 546 included in the YADE open-source platform for mechanical simulations of
 547 granular soils and other discrete systems. With distance-to-surface fields
 548 serving in a discrete fashion as a primary ingredient of the method, the pro-
 549 posed implementation also includes a Fast Marching Method to construct
 550 such fields for a wide class of surfaces with an analytical description. The
 551 versatility of the method is evident from the direct application to superellip-
 552 soids. On the other hand, significant computational costs are inherent to the

553 method, be it in terms of memory or execution time. Time costs are never-
554 theless beneficial with respect to a polyhedral description of complex shapes,
555 as already available in YADE, and they can be furthermore reduced through
556 OpenMP parallel computing with a significant speed-up. As for the memory
557 requirements, these could also decrease in the future using a more appro-
558 priate data structure than the current regular grid (Duriez and Galusinski,
559 2020).

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

565 **Appendix**

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

570 **Conflict of Interest**

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

574 **Acknowledgements**

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

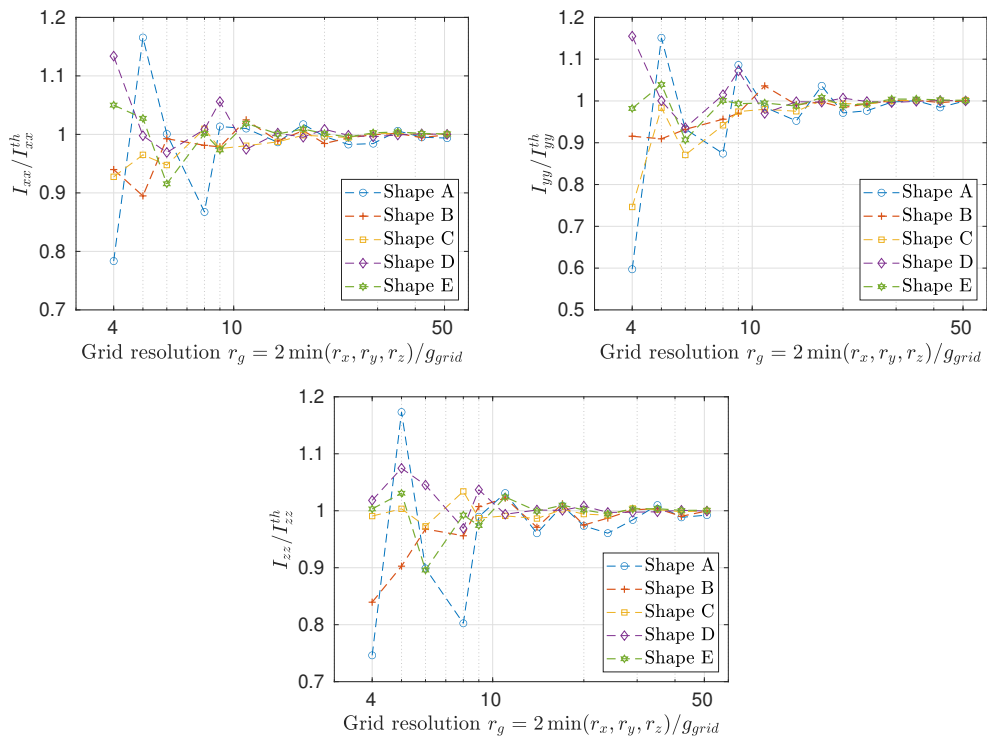


Figure 15: LS-DEM precision in describing inertia components for the five superellipsoids of Figure 6

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

582 **Computer code availability**

583 The present LS-DEM code is released under the GNU General Public
584 License v2. It has been developed by Jérôme Duriez (jerome.duriez@inrae.fr),
585 the contacting author of the manuscript, and made first available in January
586 2021.

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

591 A bash script `install.sh` is for instance available at [https://gitlab.](https://gitlab.com/jduriez/lsYade)
592 [com/jduriez/lsYade](https://gitlab.com/jduriez/lsYade) and in the manuscript submission, in order to down-
593 load source code and trigger compilation. After a correct installation, execut-
594 ing `install/bin/yadelevelSet --check` should include running: `checkLS-`
595 `dem.py [...] Status: success` in its output.

596 YADE LS-DEM simulations are realistically possible on computing-oriented,
597 multi-core (clock speed higher than 2.5 GHz) personal desktops, with signifi-
598 cant RAM: several tens of GB are for instance necessary for simulating Rep-
599 resentative Elementary Volumes of granular soils with an adequate precision.
600 Visualization of the simulations builds upon the free, open-source, Paraview
601 software and its Python interface. Compilation dependencies include e.g.
602 `cmake`, `g++`, `boost`, `Qt`, `freeglut3`, `libQGLViewer`, `eigen`, `gdb`, `sqlite3`, `Loki`,

603 VTK, Python3 including numpy, sphinx, IPython, matplotlib on Ubuntu
604 18.04 or 20.04 (see the Prerequisites section of `install.sh`. Note that the
605 Paraview Python interface is provided by `paraview-python`, resp. `python3-`
606 `paraview`, package on Ubuntu 18.04, resp. 20.04).

607 **References**

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

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

612 Barr, A. H., 1981. Superquadrics and angle-preserving transformations. *IEEE*
613 *Computer Graphics and Applications* 1 (1), 11 – 23.

614 URL [https://authors.library.caltech.edu/9756/1/BARIEEECGA81.](https://authors.library.caltech.edu/9756/1/BARIEEECGA81.pdf)
615 **pdf**

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

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

621 Boon, C., Houlsby, G., Utili, S., 2013. A new contact detection algorithm
622 for three-dimensional non-spherical particles. *Powder Technology* 248, 94
623 – 102.

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

- 627 Cundall, P., Strack, O., 1979. A discrete numerical model for granular as-
628 semblies. *Géotechnique* 29, 47–65.
- 629 Deluzarche, R., Cambou, B., 2006. Discrete numerical modelling of rock-
630 fill dams. *International Journal for Numerical and Analytical Methods in*
631 *Geomechanics* 30, 1075–1096.
- 632 Dubois, F., 2011. Numerical modeling of granular media composed of polyhe-
633 dral particles. In: Radjai, F., Dubois, F. (Eds.), *Discrete-element Modeling*
634 *of Granular Materials*. ISTE-Wiley, pp. 233–262.
- 635 Duriez, J., Bonelli, S., 2021. Precision and computational costs of Level Set-
636 Discrete Element Method (LS-DEM) with respect to DEM. *Computers*
637 *and Geotechnics* 134, 104033.
- 638 Duriez, J., Darve, F., Donzé, F.-V., 2011. A discrete modeling-based consti-
639 tutive relation for infilled rock joints. *International Journal of Rock Me-*
640 *chanics & Mining Sciences* 48 (3), 458–468.
- 641 Duriez, J., Galusinski, C., 2020. Level set representation on octree for gran-
642 ular material with arbitrary grain shape. In: Šimurda, D., Bodnár, T.
643 (Eds.), *Proceedings Topical Problems of Fluid Mechanics 2020*. Prague,
644 pp. 64–71.
645 URL <http://www2.it.cas.cz/fm/im/im/proceeding/2020/9>
- 646 Duriez, J., Wan, R., Pouragha, M., Darve, F., 2018. Revisiting the existence
647 of an effective stress for wet granular soils with micromechanics. *Inter-*
648 *national Journal for Numerical and Analytical Methods in Geomechanics*
649 42 (8), 959–978.
- 650 Eliáš, J., 2014. Simulation of railway ballast using crushable polyhedral par-
651 ticles. *Powder Technology* 264, 458 – 465.

- 652 Garcia, X., Latham, J.-P., Xiang, J., Harrison, J., 2009. A clustered over-
653 lapping sphere algorithm to represent real particles in discrete element
654 modelling. *Géotechnique* 59 (9), 779–784.
- 655 Gladkyy, A., Kuna, M., 2017. DEM simulation of polyhedral particle crack-
656 ing using a combined Mohr–Coulomb–Weibull failure criterion. *Granular*
657 *Matter* 19 (3), 41.
- 658 Guo, N., Zhao, J., 2013. The signature of shear-induced anisotropy in gran-
659 ular media. *Computers and Geotechnics* 47, 1–15.
- 660 Guo, N., Zhao, J., 2014. A coupled FEM/DEM approach for hierarchical
661 multiscale modelling of granular media. *International Journal for Numer-*
662 *ical Methods in Engineering* 99 (11), 789–818.
- 663 Hagenmuller, P., Chambon, G., Naaim, M., 2015. Microstructure-based mod-
664 eling of snow mechanics: a discrete element approach. *The Cryosphere*
665 9 (5), 1969–1982.
- 666 Haustein, M., Gladkyy, A., Schwarze, R., 2017. Discrete element modeling
667 of deformable particles in YADE. *SoftwareX* 6, 118 – 123.
- 668 Houlsby, G., 2009. Potential particles: a method for modelling non-circular
669 particles in DEM. *Computers and Geotechnics* 36 (6), 953 – 959.
- 670 Kawamoto, R., Andò, E., Viggiani, G., Andrade, J. E., 2016. Level set
671 discrete element method for three-dimensional computations with triax-
672 ial case study. *Journal of the Mechanics and Physics of Solids* 91, 1–13.
- 673 Kawamoto, R., Andò, E., Viggiani, G., Andrade, J. E., 2018. All you need
674 is shape: Predicting shear banding in sand with LS-DEM. *Journal of the*
675 *Mechanics and Physics of Solids* 111, 375–392.

- 676 Kettner, L., 2018. 3D polyhedral surface. In: CGAL User and Reference
677 Manual, 4.11.3 Edition. CGAL Editorial Board.
678 URL [http://doc.cgal.org/4.11.3/Manual/packages.html#](http://doc.cgal.org/4.11.3/Manual/packages.html#PkgPolyhedronSummary)
679 [PkgPolyhedronSummary](http://doc.cgal.org/4.11.3/Manual/packages.html#PkgPolyhedronSummary)
- 680 Li, L., Marteau, E., Andrade, J. E., 2019. Capturing the inter-particle force
681 distribution in granular material using LS-DEM. *Granular Matter* 21 (3),
682 43.
- 683 Lin, C.-C., Ching, Y.-T., 1996. An efficient volume-rendering algorithm with
684 an analytic approach. *The Visual Computer* 12 (10), 515–526.
- 685 Lorensen, W. E., Cline, H. E., 1987. Marching cubes: A high resolution 3D
686 surface construction algorithm. In: *Proceedings of the 14th Annual Con-*
687 *ference on Computer Graphics and Interactive Techniques. SIGGRAPH*
688 *87. p. 163169.*
- 689 Mede, T., Chambon, G., Hagenmuller, P., Nicot, F., 2018. A medial axis
690 based method for irregular grain shape representation in DEM simulations.
691 *Granular Matter* 20 (1), 16.
- 692 Miehe, C., Dettmar, J., Zäh, D., 2010. Homogenization and two-scale simu-
693 lations of granular materials for different microstructural constraints. *In-*
694 *ternational Journal for Numerical Methods in Engineering* 83 (8-9), 1206–
695 1236.
- 696 Osher, S., Sethian, J. A., 1988. Fronts propagating with curvature-dependent
697 speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of*
698 *Computational Physics* 79 (1), 12– 49.
- 699 Pirnia, P., Duhaime, F., Ethier, Y., Dub, J.-S., 2019. ICY: An interface

700 between COMSOL multiphysics and discrete element code YADE for the
701 modelling of porous media. *Computers & Geosciences* 123, 38 – 46.

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

705 Sethian, J., 1996. A fast marching level set method for monotonically ad-
706 vancing fronts. *Proceedings of the National Academy of Sciences* 93 (4),
707 1591–1595.

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

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

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

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

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

724 Yang, L., Hyde, D., Grujic, O., Scheidt, C., Caers, J., 2019. Assessing and vi-
725 sualizing uncertainty of 3D geological surfaces using level sets with stochas-
726 tic motion. *Computers & Geosciences* 122, 54 – 67.