Computer Physics Communications 259 (2021) 107670

Contents lists available at ScienceDirect

Computer Physics Communications

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

SudoDEM: Unleashing the predictive power of the discrete element method on simulation for non-spherical granular particles☆,☆☆

Shiwei Zhao [a,b,*], Jidong Zhao [a]

[a] Department of Civil and Environmental Engineering, Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong
[b] State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou, China

ARTICLE INFO

Article history:
Received 14 May 2020
Received in revised form 30 August 2020
Accepted 6 October 2020
Available online 22 October 2020

Keywords:
Discrete element
Granular
Non-spherical
Particle shape
Superellipsoid
Polyhedron
SudoDEM
Open-source

ABSTRACT

This paper presents a novel open-source discrete element code, *SudoDEM*, for efficient modeling of both 2D and 3D non-spherical particles under a GPL v3 or later license. Built upon a popular open-source code *YADE*, our code inherits the core of a classic DEM framework empowered by OpenMP acceleration, and further offers unique features of a rich library of prime particle shapes, including poly-superellipsoids, superellipsoids, cylinders, cones, polyhedrons for 3D and disks and superellipses for 2D. Unlimited choices of more complex particle shapes can be readily generated by clumping these prime shapes. Efficient modeling of complex shaped particles hinges on contact detection. In *SudoDEM*, we have developed three generic and efficient contact detection algorithms, the parametric common normal (PCN) algorithm, the Gilbert–Johnson–Keerthi (GJK) algorithm, and the hybrid PCN–GJK algorithm, to handle contacts among complex-shaped particles during a typical DEM simulation. The new DEM code is validated and further showcased by multiple examples, including granular packing, triaxial compression, and landslide, its robustness, efficiency and versatility in providing realistic solutions to granular mechanics problems. The project is hosted at an open-source page at https://sudodem.github.io, while the source codes are freely available at a GitHub repository (https://github.com/SudoDEM). We foresee a great capability and potential for *SudoDEM* in advancing future progress in granular physics and granular mechanics and in fostering advanced simulations of critical engineering and industrial processes pertaining to granular media.

Program summary

Program title: SudoDEM
CPC Library link to program files: http://dx.doi.org/10.17632/brpk4g28zn.1
Developer's repository link: http://github.com/SudoDEM/SudoDEM, http://sudodem.github.io
Licensing provisions: GNU General Public License 3
Programming language: C++, Python
Nature of problem: Grain shape underpins important aspects of the physical and mechanical behaviors of granular media. The inability of realistic and robust modeling of particle shape has been a major obstacle for discrete-based numerical methods in solving practical problems of granular materials.
Solution method: SudoDEM implements three generic algorithms of contact detection among non-spherical particles in Discrete Element Method, which provides a rich library of particle shapes available for DEM modeling of granular media.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Granular media are second most processed materials on the earth (next to water). Understanding their physical and mechanical behavior is critical for many engineering and industrial sectors, such as grain storage in the agricultural industry, powder processing in chemical engineering, tablet medicine manufacturing, packing and transport in pharmaceutical industry, mineral extraction and handling in mining engineering, and earthworks in civil and geotechnical engineering. Micromechanics-based approaches, represented by the discrete element method (DEM) [1],

☆ The review of this paper was arranged by Prof. D.P. Landau.

☆☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author at: Department of Civil and Environmental Engineering, Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong.
E-mail addresses: ceswzhao@ust.hk, swzhao@scut.edu.cn (S. Zhao).

https://doi.org/10.1016/j.cpc.2020.107670
0010-4655/© 2020 Elsevier B.V. All rights reserved.

respect the discontinuous nature of granular media at particle level and derive their collective responses based on particle-scale physics and mechanics. They have prevailed for over three decades in providing effective and efficient solutions to engineering and industrial problems pertaining to granular materials [2–4].

Collective behavior of a granular assembly originates from the microstructural characteristics at grain scales. Among many, particle morphology (especially particle shape) is one of the most important features that dictates various key facets of granular responses, including deformation, strength, failure and flow, as highlighted by numerous experimental observations, e.g., [5] and [6] in geomechanics, and [7] in other disciplines. DEM has been conventionally and predominantly based on spherical particles (3D) or circular disks (2D), due primarily to the physical convenience and computational efficiency it may offer. Sphere or circular disk based DEM has largely employed two general strategies to further consider the effect of particle shape. One resorts to the use of spheres with artificially applied rolling resistance models (e.g., [8]), and the other is the so-called clumped (or glued) sphere technique (e.g., [9]). The rolling resistance approach can help, to certain extent, to capture the resistance of particle rotation due to irregularity in shape. It is in all essence a micromechanically phenomenological remedy that more likely may cause unrealistic fabric and fail in the analysis of micro–macro bridging. The clumped sphere approach is in line with the discrete particle concept of DEM itself, but bears two major drawbacks that are hard to overcome, namely, unwanted surface roughness (or unrealistic multi-contacts) [10] and a dramatic decrease in computational efficiency with increasing number of clumped spheres [11]. Indeed, the past decade has witnessed an increasing interest in directly modeling non-spherical particles with various shapes that are mathematically feasible to express and handle, including ellipsoid [12,13], super-ellipsoid [14,15], polyhedron [16,17], non-uniform rational basis spline (NURBS) [18], among many others. Accordingly, different contact detection algorithms have been developed to accommodate these different shapes in DEM simulations. For example, a variety of methodologies and algorithms have been developed for contact detection of convex polyhedrons (Nezami et al. [19], Boon et al. [16], Wachs et al. [20], Eliáš [21], Zheng et al. [22]). More recently, Kawamoto et al. [23] employed the level set method to handle contact detection among realistic particles in a brute-force manner by using lookup tables. Zhao and Zhao [24] proposed a novel and efficient poly-superellipsoid-based approach for modeling non-spherical particles based on hybrid Levenberg–Marquardt (LM) and Gilbert–Johnson–Keerthi (GJK) algorithms, by which a wide range of shape features (including elongation, flatness, angularity, and asymmetry) for real particles in nature can be well captured and efficiently modeled in DEM.

Practical implementation of contact detection for non-spherical particles in DEM is significantly complicated than its counterpart for the sphere case, especially when the end users expect an uncompromised computational efficiency. Although there have been published algorithms, there are rather scarce open-source DEM codes ever developed for non-spherical particles. In answering the surging needs from the large scientific and engineering community, we present an open-source DEM code, *SudoDEM*, for large-scale simulation of granular media problems with non-spherical particles. *SudoDEM* is derived from a basic DEM framework of another open-source DEM code, YADE [25,26], by retaining some fundamental features such as hybrid programming of Python and C++, OpenMP acceleration and running on a Unix-like operating system. The inheritance also enables new users and experienced YADE users to access the abundant documents and supports from the YADE community in the first place. Based on the essential components, we further develop unique and useful features into it with a rich library of prime non-spherical particles that are either analytically expressible or amiable for implementation of contact detection and computationally manageable, and can be further enriched to algebraically form more complicated grain shapes to suit various special needs in practical modeling of granular media. This paper aims to present a concise introduction to *SudoDEM* on its essential framework, workflow and algorithmic features, while skipping details on numerical implementation to avoid unnecessary distractions. For interested readers, we provide a quick guide accompanying the source codes for more details. Note that all materials mentioned in this paper are available online at our open-source project page (https://sudodem.github.io).

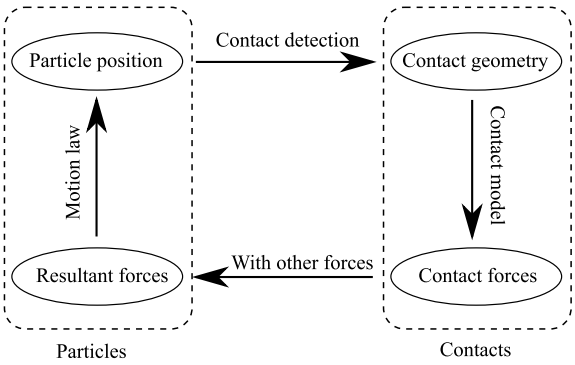

Fig. 1. Computation flowchart of a single iteration of DEM.

2. Essential ingredients of *SudoDEM*

SudoDEM inherits the core framework of YADE that has been well documented in [26]. This section only presents the critical ingredients required for the completeness of the presentation.

2.1. Computation flowchart

Fig. 1 shows the computation flowchart of DEM for a single iteration. The DEM computation is associated with two solid bodies, i.e., particle and contact, referring to the dashed rectangles in Fig. 1. In a particulate system, the particle position varies with the generation or vanishing of contact, implying a one-to-one mapping between the topology of particle arrangement and the contact distribution. Specifically, given the particle positions, the corresponding information of contact geometry (state) can be obtained by performing a procedure of contact detection. The contact force at each contact is then computed with a given contact model that offers a relationship between contact force and displacement. The resultant force (or torque), by summing up all contact forces and other body forces (e.g., gravitational forces) acting on a given particle, drives the change of the particle state according to a law of motion (i.e., the Newton second law). When all states of the particle under consideration are updated, it is ready to proceed for another DEM iteration. The above flowchart is applicable for all particles in an assembly of granular body.

2.2. The motion of particle

The motion of a particle in a granular assembly is governed by the following Newton–Euler equations:

$$F_i^{(b)} + f_i^{(d)} + \sum_{c=1}^{N} F_i^{(c)} = m\frac{\mathrm{d}v_i}{\mathrm{d}t} \tag{1a}$$

$$T_i^{(d)} + \sum_{c=1}^{N} M_i^{(c)} = I_i\frac{\mathrm{d}\omega_i}{\mathrm{d}t} - (I_j - I_k)\omega_j\omega_k \tag{1b}$$

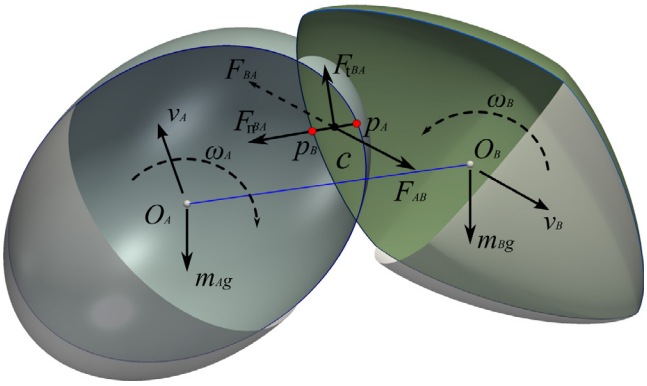

Fig. 2. Three-dimensional illustration of two touching particles with an exaggerated overlap [15].

where i, j, k are sequential indices; m is the particle mass; v_i and ω_i are the translational and angular velocities, respectively; N is the number of contacts; $F_i^{(c)}$ is the contact force at contact c; $F_i^{(b)}$ is the body force; I_i is the principal moment of inertia; $M_i^{(c)}$ is the torque around the mass center at contact c; $f_i^{(d)}$ and $T_i^{(d)}$ are damping force and torque, respectively, which are artificially introduced to facilitate the dissipation of kinetic energy in the system. Two commonly-used damping schemes, local damping and viscous damping, applied to the particle body and the contact, respectively, are given by the following equations,

$$a = -\alpha_d a Sign(v, a) \tag{2a}$$

$$f = 2\bar{v}\beta_d\sqrt{\bar{m}k} \tag{2b}$$

where a and v are the acceleration and the velocity of the particle, respectively; $Sign$ is the signum function; f and k are the viscous force and the contact stiffness at the contact, respectively; $\bar{m}$ is the equivalent mass of the two contacting particles, often given by $m_A m_B/(m_A + m_B)$ (m_A and m_B are the masses of the two particles, respectively); $\bar{v}$ is the relative velocity of the two particles at the contact; α_d and β_d are the coefficients of local damping and viscous damping, respectively. Note that the viscous force f is often calculated by the two orthogonal components, normal and tangential parts, in line with the calculation of the contact force (see Contact Model in Section 2.3); the tangential force is no longer damped once the contact slides (i.e., reaching the maximum friction) for cohesionless granular materials. It is worth noting that the Verlet scheme is adopted as the default time integrator in the simulations presented in the demos, but other elegant and complicated time integrators are available from YADE.

2.3. Contact model

Fig. 2 presents a snapshot of two contacting particles. For the sake of convenience, the contact force is generally split into two orthogonal components, the normal contact force $\boldsymbol{F}_n$ and the tangential contact force $\boldsymbol{F}_t$, which are calculated using a force–displacement law [1] given as

$$\boldsymbol{F}_n = K_n \boldsymbol{d} \tag{3a}$$

$$\boldsymbol{F}_t = \boldsymbol{F}'_t - k_t \Delta \boldsymbol{u} \tag{3b}$$

where K_n and k_t are contact normal-secant and shear-tangent stiffnesses, respectively; $\boldsymbol{d}$ is contact penetration depth; $\Delta\boldsymbol{u}$ is the incremental tangential displacement during the current time step, and $\boldsymbol{F}'_t$ is the tangential contact force at the previous time step which has been rotated to the current frame. Note that K_n and k_t depend on the material properties, contact deformation, and loading history, which may collectively yield complicated contact constitutive behaviors that can be described by a non-linear contact model, for example, the well-known Hertz–Mindlin model as follows

$$K_n = \frac{4\pi}{3\alpha}\sqrt{\frac{\mathbb{E}}{\mathbb{K}^3(A+B)}}G^*\|\boldsymbol{d}\|^{\frac{1}{2}} \tag{4a}$$

$$k_t = \pi b\left(\left(\mu_1\mathbb{K} - \mu_2\frac{\mathbb{K}-\mathbb{E}}{e^2}\right)^{-1} + \left(\mu_1\mathbb{K} + \mu_2\frac{(1-e^2)\mathbb{K}-\mathbb{E}}{e^2}\right)^{-1}\right) \tag{4b}$$

where α, b and e are the geometrical parameters associated with the contact profile between the two contacting particles; A and B are the relative curvatures that can be calculated from the principal curvatures of the two particle surfaces and the contact profile; $\mathbb{K}$ and $\mathbb{E}$ are the complete elliptic integrals of the first kind and the second kind of argument e, respectively; G^* is the equivalent contact shear modulus determined by the particle shear modulus and Poisson's ratio; μ_1 and μ_2 are the parameters introduced in terms of the particle shear modulus and Poisson's ratio. More details on the Hertz–Mindlin model for non-spherical particles are outlined in the literature [27]. For the sake of computational efficiency, however, a linear-spring model (i.e., constant K_n and k_t) is frequently used in practical simulations. Zhao et al. [27] reported that using a linear-spring model can yield rather similar microscopic and macroscopic responses as using the Hertz–Mindlin model for quasi-static simulations. In addition to the normal and shear contact force–displacement relationships, the Coulomb friction model is applied to each contact, i.e., $\|\boldsymbol{F}_t\| = \min\{\|\boldsymbol{F}_t\|, \mu\|\boldsymbol{F}_n\|\}$ where μ is the coefficient of friction. For more details about these ingredients of DEM, interested readers are referred to the literature [2].

Rolling resistance models are commonly introduced to provide an additional anti-rotational torque to resist particle rotation for spherical particles as a make up for their shape simplicity [8,28]. The anti-rotational torques M_r in the rolling direction and M_t in the twisting direction are considered for three-dimensional cases, as shown in Fig. 3(a). Referring to Fig. 3(b), a typical elasto-plastic rolling resistance model may be defined as [29]

$$\Delta\mathbf{M}_r = 0.25\beta^2 k_n r_1 r_2 \Delta\boldsymbol{\theta}_r, \quad \mathbf{M}_r \le 0.25\xi\beta f_n r \tag{5a}$$

$$\Delta\mathbf{M}_t = 0.5\beta^2 k_t r_1 r_2 \Delta\boldsymbol{\theta}_t, \quad \mathbf{M}_t \le 0.65\beta\mu f_n r \tag{5b}$$

where k_n and k_t are the normal and tangential contact stiffness, respectively; r_1 and r_2 are radii (equivalent radii for non-spheres) of the two contacting particles, respectively; $\Delta\boldsymbol{\theta}_r$ and $\Delta\boldsymbol{\theta}_t$ are incremental rotational angles in the rolling and twisting directions, respectively; f_n is the normal contact force; r is the (equivalent) radius of the smaller one of the two contacting particles; μ is the inter-particle coefficient of friction, and β and ξ are the model parameters, where β can be regarded as a shape parameter related to contact radius contributing to anti-rotational stiffness, while ξ describes the effects of local asperity crushing determining the maximum resistance that the contact can provide and related to the hardness of particle material [8]. In *Sudo-DEM*, the rolling resistance model can be installed at contact for non-spherical particles, by which smaller-scale details of particle shape (e.g., roughness) might be captured quantitatively.

2.4. Contact detection

Contact detection, also known as collision detection in graphic simulations, takes a dominant portion of running time in a simulation, e.g., up to 97% for non-spherical particles [24]. Following

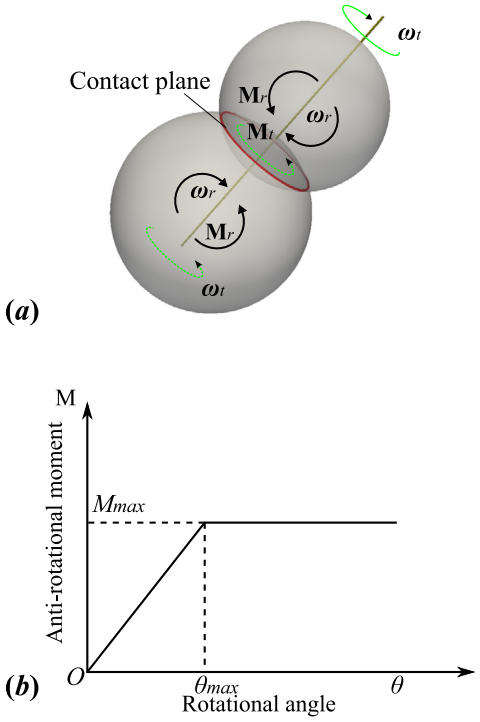

Fig. 3. Schematic of the rolling resistance model [29].

the conventional implementation, a combination of approximate and exact detection phases (also called broad and narrow phases) is adopted to reduce the computational pressure of contact detection at each time step. At the broad phase, the AABB (axis-aligned bounding box) algorithm [30] is employed to rule out most of the particles that are not touching one another. As for the narrow phase, *SudoDEM* provides three generic contact detection algorithms, as depicted in Sections 4, 5, and 6, respectively.

3. Major features of *SudoDEM*

3.1. Two- and three-dimensional packages

SudoDEM includes two packages *SudoDEM*^{2D} and *SudoDEM*^{3D} for two- and three-dimensional simulations, respectively [15,24,27,29,31,32]. *SudoDEM*^{3D} inherits similar data structures from YADE, while the third-dimension data has been removed in *SudoDEM*^{2D}. For example, in *SudoDEM*^{2D} the three-dimensional vector *Vector3r* is replaced by a two-dimensional one *Vector2r*, and the particle orientation and rotation are represented by *Rotation2D* instead of quaternions. In so doing, not only a better performance in computational efficiency but also computer-memory saving can be achieved, compared with the general implementation that one may fix the third-dimensional degree of freedom in 3D to simulate 2D cases. In addition, user-friendly GUIs are provided for both packages, as shown in Fig. 4, which helps the modelers to have a quick simulation setting and visualization in the first place. Benefited from the hybrid programming of C++ and Python, where the computationally heavy kernel is written in C++ for efficiency in computation while the frontend exposed to modelers is the wrapped C++ functions in Python, it is efficient for the modelers to set up a simulation. Listing 1 demonstrates a featured snippet in Python for a quick setup of a simulation with all ingredients of a general DEM simulation included, where we simulate a sphere falling freely towards the ground under gravity pull.

```
from sudodem import utils # module utils has some auxiliary functions
# define and append a material into the simulation
mat = RolFrictMat(Kn=1e8,Ks=7e7,frictionAngle=0,density=2650)
O.materials.append(mat)
# add a sphere and an infinite plane (the ground) into the simulation
O.bodies.append(sphere((0,0,10.0),1.0, material = mat))
O.bodies.append(utils.wall(0,axis=2,sense=1, material = mat))
# define the engines that will be executed at each time step
O.engines=[
ForceResetter(), # reset the force container
# broad phase of contact detection by AABBs.
InsertionSortCollider([Bo1_Sphere_Aabb(), Bo1_Wall_Aabb()]),
# narrow phase of contact detection
InteractionLoop( # loop all potentially contacting pairs of particles
[Ig2_Wall_Sphere_ScGeom()], # contact geometric info
[Ip2_RolFrictMat_RolFrictMat_RolFrictPhys()], # contact physical info
[RollingResistanceLaw(use_rolling_resistance=False)] # contact force
),
# Integration of particle motion
NewtonIntegrator(damping = 0.1,gravity=(0.,0.0,-9.8))
]
O.dt = 1e-5 # set a time step
O.run() # run the simulation
```

Listing 1: A Python snippet of modeling a sphere falling freely towards the ground under gravity.

3.2. Particle shape

SudoDEM provides a rich library of prime particle shapes, including superellipsoid, poly-superellipsoid, cylinder, cone, polyhedron for 3D, and disk, superellipse for 2D. A superellipsoid can be defined by the following surface equation

$$\left(\left|\frac{x}{r_x}\right|^{\frac{2}{\epsilon_1}}+\left|\frac{y}{r_y}\right|^{\frac{2}{\epsilon_1}}\right)^{\frac{\epsilon_1}{\epsilon_2}}+\left|\frac{z}{r_z}\right|^{\frac{2}{\epsilon_2}}=1 \tag{6}$$

where r_x, r_y and r_z are referred to as the semi-major axis lengths in the direction of x, y, and z axes, respectively; and $\epsilon_i (i = 1, 2)$ control the surface squareness. With the capability of representing a broad range of particle shapes as shown in Fig. 5, superellipsoids have been employed popularly in graphic simulations and robotic science.

Based on superellipsoids, we further propose a novel and versatile shape, coined as poly-superellipsoid, which can capture more shape features including elongation, flatness, angularity, and asymmetry. A poly-superellipsoid is an assemblage of eight pieces of superellipsoids for the eight octants, mathematically controlled by the following surface function [24,33]

$$\left(\left|\frac{x}{r_x}\right|^{\frac{2}{\epsilon_1}}+\left|\frac{y}{r_y}\right|^{\frac{2}{\epsilon_1}}\right)^{\frac{\epsilon_1}{\epsilon_2}}+\left|\frac{z}{r_z}\right|^{\frac{2}{\epsilon_2}}=1 \tag{7}$$

with

$$r_x = r_x^+ \quad \text{if} \quad x \geq 0 \quad \text{else} \quad r_x^- \tag{8a}$$

$$r_y = r_y^+ \quad \text{if} \quad y \geq 0 \quad \text{else} \quad r_y^- \tag{8b}$$

$$r_z = r_z^+ \quad \text{if} \quad z \geq 0 \quad \text{else} \quad r_z^- \tag{8c}$$

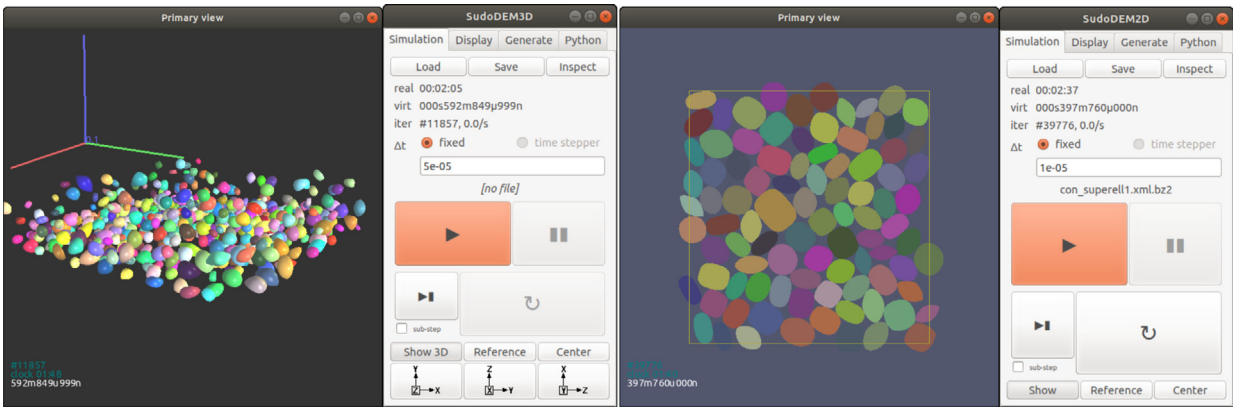

Fig. 4. GUIs for *SudoDEM*^{3D} (*left*) and *SudoDEM*^{2D} (*right*).

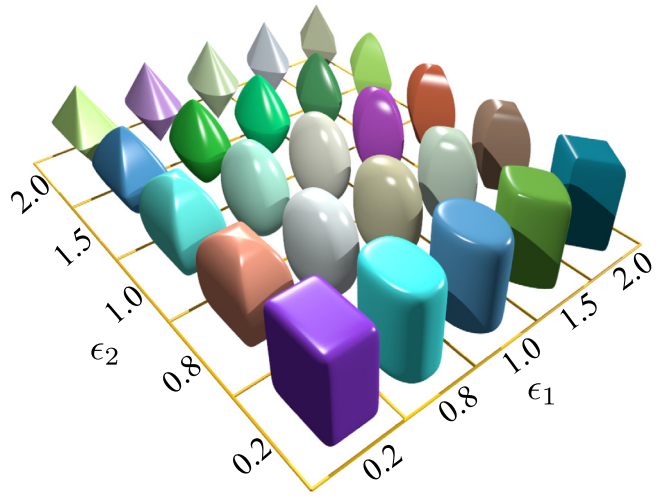

Fig. 5. Superellipsoids with $r_x = 1.0, r_y = 1.5, r_z = 2.0$ and varying ϵ_1, ϵ_2 [24].

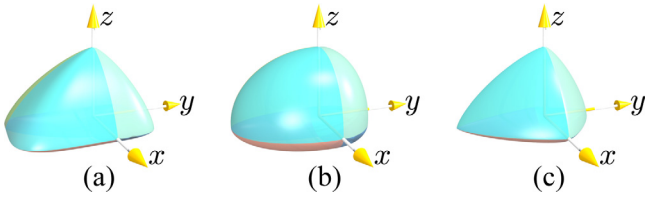

Fig. 6. Poly-superellipsoids with $r_x^+ = 1.0, r_x^- = 0.5, r_y^+ = 0.8, r_y^- = 0.9, r_z^+ = 0.4, r_z^- = 0.6$ and (a) $\epsilon_1 = 0.4, \epsilon_2 = 1.5$, (b) $\epsilon_1 = \epsilon_2 = 1.0$, (c) $\epsilon_1 = \epsilon_2 = 1.5$ [33].

where r_x^+, r_y^+, r_z^+ and r_x^-, r_y^-, r_z^- are the principal elongation along the positive and negative directions of x, y and z axes, respectively; ϵ_1 and ϵ_2 control the squareness or blockiness of particle surface, and their possible values are within $(0, 2)$ for convex shapes as exemplified in Fig. 6. To quantify particle shape, two shape descriptors are introduced such that

$$r_x^+ = l_x e_x, \quad r_x^- = l_x(1 - e_x) \tag{9a}$$
$$r_y^+ = l_y e_y, \quad r_y^- = l_y(1 - e_y) \tag{9b}$$
$$r_z^+ = l_z e_z, \quad r_z^- = l_z(1 - e_z) \tag{9c}$$

where l_x, l_y and l_z are principal lengths; e_x, e_y and e_z are principal eccentricities.

With respect to cylinder, cone and polyhedron, their definitions are not presented here for brevity. Fig. 7 shows a series of granular packings with built-in particle shapes in *SudoDEM*^{3D}. As for the two-dimensional particle shapes, the corresponding definitions can be readily obtained by degenerating from the three-dimensional cases. For example, a superellipse can be given by

$$\left|\frac{x}{r_x}\right|^{\frac{2}{\epsilon}} + \left|\frac{y}{r_y}\right|^{\frac{2}{\epsilon}} = 1 \tag{10}$$

where r_x and r_y are referred to as the semi-major axis lengths in the direction of x and y axes, respectively; ϵ is for the surface squareness. Furthermore, with a variation of the semi-major axis lengths r_x and r_y given by

$$r_x = r_x^+ \quad \text{if} \quad x \geq 0 \quad \text{else} \quad r_x^- \tag{11a}$$
$$r_y = r_y^+ \quad \text{if} \quad y \geq 0 \quad \text{else} \quad r_y^- \tag{11b}$$

a poly-superellipse can be defined, where r_x^+, r_y^+ and r_x^-, r_y^- are the principal elongation along the positive and negative directions of x and y axes, respectively.

3.3. Flexible membranes

A flexible membrane can be constructed by deformable triangular facets, i.e., constant strain triangles in the finite element method (FEM). The deformation of a facet is computed by the nodal motion, where each node has a lumped mass and equally-distributed contact forces from its associated facets. The contact between an FE facet and a particle is a degenerate case for the general inter-particle contact that is depicted in Sections 4–6. Fig. 8 demonstrates the deformation of a flexible member impacted by a granular column of poly-superellipsoidal particles falling freely under gravity.

3.4. Visualization tools

In addition to the integrated view in the GUI powered by OpenGL, *SudoDEM* also provides modules to export the simulation scene into vtk and pov files that can be rendered by using the third-party software such as *Paraview* and *POV-Ray*. For example, two auxiliary *POV-Ray* macros are defined as listed in Listing 2 for the visualization of poly-superellipsoids with *POV-Ray*. Fig. 9 shows an example of high-quality simulation visualization by using *POV-Ray* with the export interface of *SudoDEM*.

3.5. Contact detection algorithms

Two generic and efficient contact-detection algorithms, the parametric common normal (PCN) algorithm and the Gilbert–Johnson–Keerthi (GJK) algorithm, are employed in *SudoDEM*. The

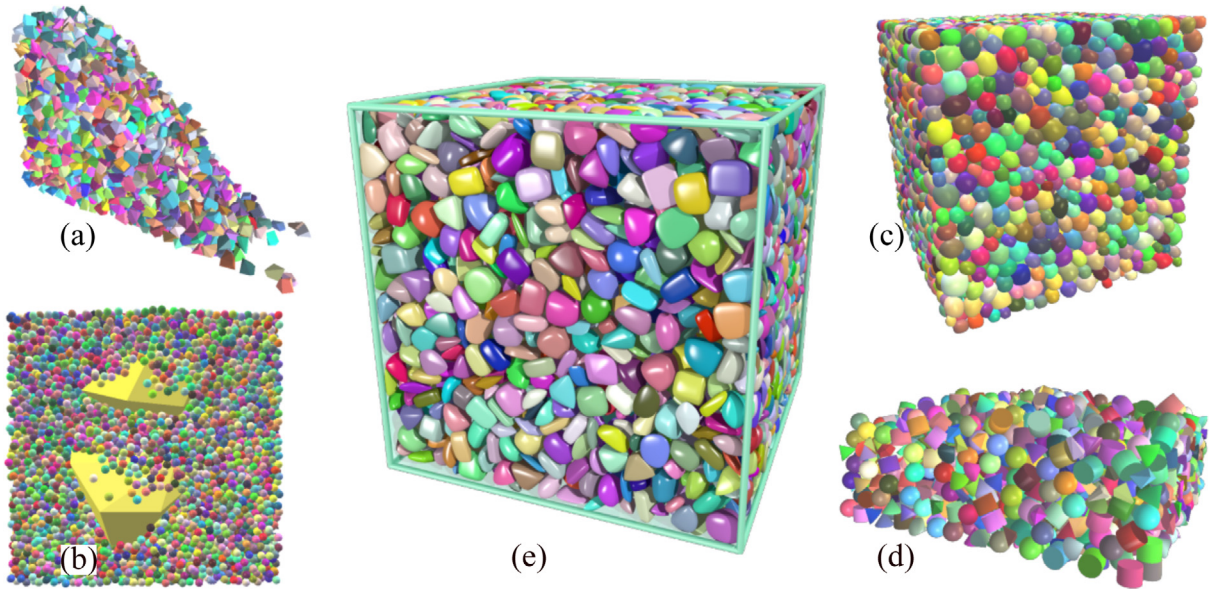

Fig. 7. Packings with different particle shapes: (a) polyhedrons; (b) mixture of polyhedrons and spheres; (c) super-ellipsoids; (d) mixture of polyhedrons, cones, cylinders and spheres; (e) poly-superellipsoids.

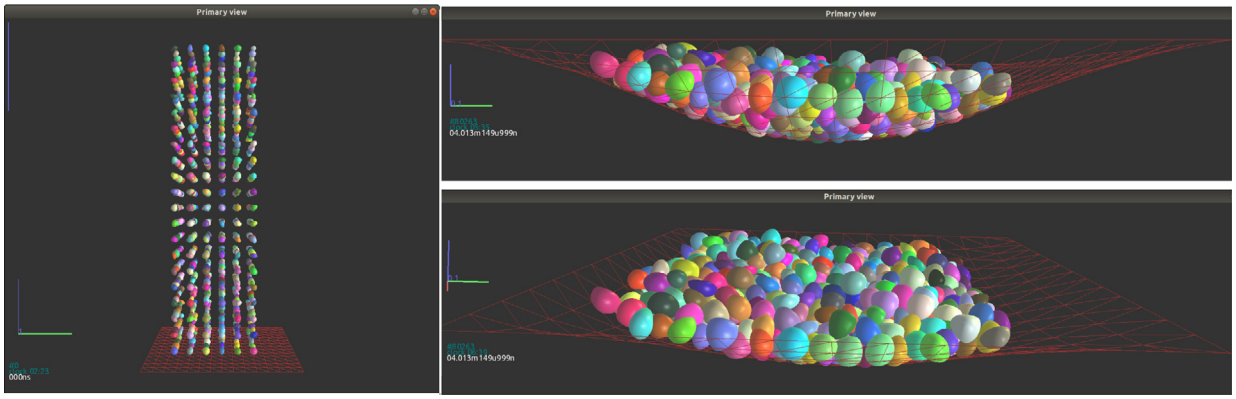

Fig. 8. A granular column of poly-superellipsoidal particles falling freely towards a flexible membrane: the initial state (*left*) and the final state with different views (*right*).

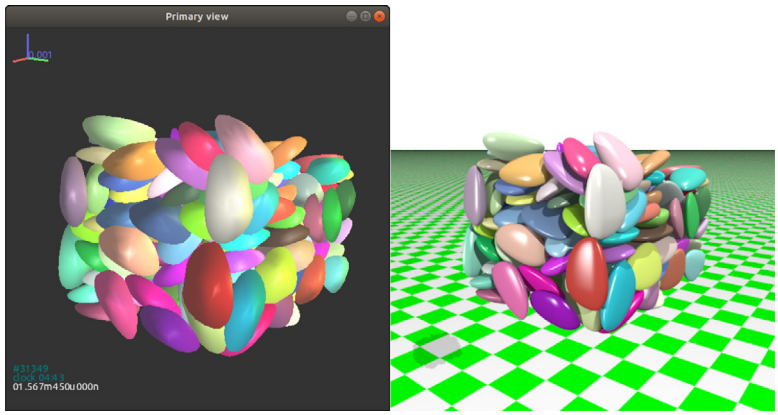

Fig. 9. Visualization of simulation: a quick real-time view integrated in the GUI (*left*) and a post-processing high-quality view with the export interface to third-party tools (*right*).

detailed implementation of PCN and GJK algorithms is referred to the literature, e.g., [15,24]. For the current released version, the built-in prime shapes such as superellipsoid, poly-superellipsoid, and superellipse are handled with PCN, while GJK handles the other shapes including polyhedron, cylinder, and cone. In addition, a hybrid algorithm between PCN and GJK is proposed as a complementary one, which has been implemented for poly-superellipsoid in our recent work [24]. We note here that *SudoDEM* provides a generic interface to extend the application of PCN and GJK to other possible customized particle shapes.

```
//
#macro octantSuper(ep1,ep2,rx,ry,rz,s1,s2,s3)
superellipsoid{ <ep1,ep2>
clipped_by{plane{-s1*x,0}}
clipped_by{plane{-s2*y,0}}
clipped_by{plane{-s3*z,0}}
scale <rx,ry,rz>}
#end
#macro polySuperEllipsoid(ep1,ep2,a1,a2,b1,b2,
    c1,c2,t1,t2,t3,r1,r2,r3)
union{
octantSuper(ep1,ep2,a2,b2,c2,-1,-1,-1)
octantSuper(ep1,ep2,a1,b2,c2,1,-1,-1)
octantSuper(ep1,ep2,a2,b1,c2,-1,1,-1)
octantSuper(ep1,ep2,a1,b1,c2,1,1,-1)
octantSuper(ep1,ep2,a2,b2,c1,-1,-1,1)
octantSuper(ep1,ep2,a1,b2,c1,1,-1,1)
octantSuper(ep1,ep2,a2,b1,c1,-1,1,1)
octantSuper(ep1,ep2,a1,b1,c1,1,1,1)
rotate<r1,r2,r3>
translate<t1,t2,t3>}
#end
```

Listing 2: Two auxiliary macros for the visualization of poly-superellipsoid in POV-Ray.

4. Parametric common normal algorithm

4.1. Parametric surface and outward normal

For an arbitrary smooth convex closed surface $\mathbb{S}$, the local coordinate $\boldsymbol{x}$ (fixed in $\mathbb{S}$) of a given surface point $\boldsymbol{p}$ can be defined as a function of the outward normal $\boldsymbol{n}$ of $\mathbb{S}$:

$$\boldsymbol{x} = S(\boldsymbol{n}) \tag{12}$$

The outward normal $\boldsymbol{n}$ can be parameterized by two angles in a local spherical coordinate system, i.e.,

$$\boldsymbol{n}(\alpha, \beta) = \cos\alpha\cos\beta\boldsymbol{i} + \sin\alpha\cos\beta\boldsymbol{j} + \sin\beta\boldsymbol{k} \tag{13}$$

where $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$ are unit base vectors of the global Cartesian coordinate system. Note that the axes of the local coordinate system coincide with that of the global one except a shift between the two corresponding origins.

In consideration of particle position and orientation, the surface point $\boldsymbol{p}$ is given by

$$\boldsymbol{p}(\alpha, \beta) = \boldsymbol{T}^{-1} S(\boldsymbol{T}\boldsymbol{n}(\alpha, \beta)) + \boldsymbol{r} \tag{14}$$

where $\boldsymbol{T}$ is the rotation matrix of a particle from the global coordinate system to the local with respect to the particle center, and $\boldsymbol{T}^{-1}$ is its inverse matrix, i.e., the inverse transformation; $\boldsymbol{r}$ is the center location of particle mass, i.e., the particle position.

4.2. Candidate penetration and common normal

Given two candidate contact points $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ (see Fig. 10, where the superscripts A and B stand for the two adjacent particles A and B hereafter), the candidate penetration $\boldsymbol{d}$ of a potential contact is defined by $\boldsymbol{d} = (\boldsymbol{p}^B - \boldsymbol{p}^A)$. By introducing the common-normal concept [14,34], as shown in Fig. 10, the outward normal $\boldsymbol{n}^A$ at point $\boldsymbol{p}^A$ shares a common normal with the contact normal $\boldsymbol{c}$, while the outward normal $\boldsymbol{n}^B$ at point $\boldsymbol{p}^B$ has an anti-direction of $\boldsymbol{c}$, i.e.,

$$\boldsymbol{n}^A = -\boldsymbol{n}^B = \boldsymbol{c} \tag{15}$$

Considering Eqs. (14) and (15), the contact points $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ are functions of the candidate contact normal $\boldsymbol{c}$, given by

$$\boldsymbol{p}^A = \boldsymbol{T}_A^{-1} S^A(\boldsymbol{T}_A \boldsymbol{c}(\alpha, \beta)) + \boldsymbol{r}^A \tag{16a}$$

$$\boldsymbol{p}^B = \boldsymbol{T}_B^{-1} S^B(-\boldsymbol{T}_B \boldsymbol{c}(\alpha, \beta)) + \boldsymbol{r}^B \tag{16b}$$

where the symbols are mentioned in Eq. (14). Therefore, the candidate penetration $\boldsymbol{d}$ is a function of the parametric angles α, β, i.e.,

$$\boldsymbol{d} = \boldsymbol{T}_B^{-1} S^B(-\boldsymbol{T}_B \boldsymbol{c}(\boldsymbol{m})) + \boldsymbol{r}^B - \boldsymbol{T}_A^{-1} S^A(\boldsymbol{T}_A \boldsymbol{c}(\boldsymbol{m})) - \boldsymbol{r}^A \tag{17}$$

with

$$\boldsymbol{m} = [\alpha, \beta]^T \tag{18}$$

In addition to the constraints given in Eq. (14), the candidate penetration $\boldsymbol{d}$ is subject to such a constraint that $\boldsymbol{d}$ is anti-parallel to the contact normal $\boldsymbol{c}$. However, to consider both touching and non-touching cases, as shown in Fig. 10, a weaker constraint needs to be applied, i.e.,

$$\boldsymbol{d} \times \boldsymbol{c} = \boldsymbol{0} \tag{19}$$

4.3. Iterative PCN

The penetration depth d can be obtained by solving the following unconstrained optimization problem with a parameter vector $\boldsymbol{m}$, i.e.,

$$\min_{\boldsymbol{m}} |d| = \min_{\alpha,\beta} \|\boldsymbol{p}^B - \boldsymbol{p}^A\| \tag{20}$$

where iterative algorithms such as the Nelder–Mead simplex algorithm [35] and the Levenberg–Marquardt algorithm [36] are applicable to the optimization problem. Note that Eq. (19) is fulfilled when Eq. (20) reaches a global minimum [14]. With solutions of the optimized parameter $\boldsymbol{m}$, other contact quantities such as contact point $\boldsymbol{p}$ inside the inter-particle overlap can be readily solved. In the above contact detection algorithm, it is most important to obtain a parametric normal-surface relation in Eq. (12) that establishes an explicit function between the candidate contact penetration $\boldsymbol{d}$ and the searching parameter $\boldsymbol{m}$. Thus, we coined a name of this algorithm as parametric common normal (PCN) algorithm, which is applicable to contact detection for any particle shape with a normal-surface relation. We present the pseudo-codes of the contact detection algorithm in Algorithm 1.

At each iteration, as listed at Line 3 in Algorithm 1, it is useful to check the following condition

$$\boldsymbol{d} \cdot \boldsymbol{c} > 0 \tag{21}$$

If this condition is true, then the two particles are not touching each other, referring to Fig. 10(b) so that the routine can be terminated. Meanwhile, a boolean flag *touching* is returned, which can be further used in other parts of DEM computation. With such a lightweight check, a large number of particle pairs can be efficiently ruled out. Besides, the PCN routine will be terminated at other proper conditions. For example, if either the angle between $\boldsymbol{d}$ and $\boldsymbol{c}$ (see Line 5 in Algorithm 1) or the change in $\boldsymbol{m}$ (see Line 15 in Algorithm 1) is sufficiently small, then we can terminate the optimization routine accordingly. Interested readers are referred to the source codes for more details.

4.4. Applied to superellipsoids and poly-superellipsoids

Taking superellipsoid as an example, the parametric function of a superellipsoid is given as

$$\boldsymbol{x}(\theta, \phi) = \begin{bmatrix} \mathrm{Sign}(\cos\theta) r_x |\cos\theta|^{\epsilon_1} |\cos\phi|^{\epsilon_2} \\ \mathrm{Sign}(\sin\theta) r_y |\sin\theta|^{\epsilon_1} |\cos\phi|^{\epsilon_2} \\ \mathrm{Sign}(\sin\phi) r_z |\sin\phi|^{\epsilon_2} \end{bmatrix} \tag{22}$$

with $\theta \in [0, 2\pi)$, $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ where r_x, r_y, and r_z are semi-length along the principal directions at the body-fixed coordinate system; ϵ_i is in $(0, 2)$. The term $\mathrm{Sign}(x)$ is the signum function. Given a normal vector (n_x, n_y, n_z) on the surface, the corresponding

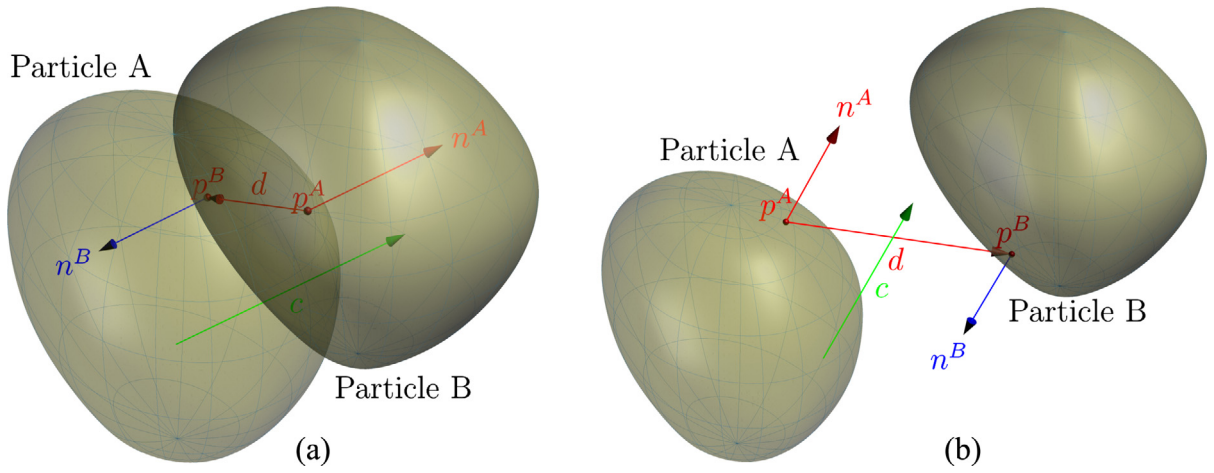

Fig. 10. Candidate penetrations and common normals for two (a) touching and (b) non-touching particles.

Algorithm 1: Parametric common normal algorithm.

Input: Particle positions $\boldsymbol{r}^A$ and $\boldsymbol{r}^B$, surface functions S^A and S^B, rotation matrices $\boldsymbol{T}_A$ and $\boldsymbol{T}_B$; cached parameters α_0 and β_0.
Output: Parameter $\boldsymbol{m}$; flag *touching*; penetration depth d; contact point $\boldsymbol{p}_c$.

$\boldsymbol{m} = [\alpha_0, \beta_0]^T$;
$\boldsymbol{d} := \boldsymbol{T}_B^{-1} S^B(-\boldsymbol{T}_B \boldsymbol{c}(\boldsymbol{m})) + \boldsymbol{r}^B - \boldsymbol{T}_A^{-1} S^A(\boldsymbol{T}_A \boldsymbol{c}(\boldsymbol{m})) - \boldsymbol{r}^A$;
for ($k := 0$; $k <$ *maxIteration* and not *stop*; *++k*) **do**
  if $\boldsymbol{d} \cdot \boldsymbol{c} > 0$ **then**
    stop := *true*; *touching* := *false*; break; //particles do not touch each other.
  if $|\boldsymbol{d} \cdot \boldsymbol{c}| / \|\boldsymbol{d}\| >$ *threshold* **then**
    stop := *true*; break; //the angle between $\boldsymbol{d}$ and $\boldsymbol{c}$ is sufficiently small.
  Compute the Jacobian matrix $\boldsymbol{J}$ of $\boldsymbol{d}$ with respect to $\boldsymbol{m}$;
  $\boldsymbol{H} := \boldsymbol{J}^T \boldsymbol{J}$; $\boldsymbol{G} := -\boldsymbol{J}^T \boldsymbol{d}$;
  if ($\|\boldsymbol{G}\|_\infty \leq \epsilon_1$) **then**
    stop := *true*; break;
  if *k==0* **then**
    compute initial damping factor $\mu := \tau \max([\boldsymbol{G}]_{ii})$;
  while true **do**
    Solve $(\boldsymbol{H} + \mu \boldsymbol{I})\delta_m + \boldsymbol{G} = \boldsymbol{0}$;
    if $\boldsymbol{m}$ has a small change $\delta_m = [\Delta\alpha, \Delta\beta]^T$, *e.g.*, $\|\delta_m\| < 10^{-3}$ **then**
      stop := *true*; break;
    $\boldsymbol{c} := \boldsymbol{c}(\boldsymbol{m} + \delta_m)$;
    $\boldsymbol{d}' := \boldsymbol{T}_B^{-1} S^B(-\boldsymbol{T}_B \boldsymbol{c}) + \boldsymbol{r}^B - \boldsymbol{T}_A^{-1} S^A(\boldsymbol{T}_A \boldsymbol{c}) - \boldsymbol{r}^A$;
    $\lambda := (\|\boldsymbol{d}\| - \|\boldsymbol{d}'\|)/(\delta_m^T(\mu\delta_m + \boldsymbol{G}))$;
    if $\lambda > 0$ **then**
      $\lambda := 1 - (2\lambda - 1)^3$; $\mu := \mu \max(\lambda, \frac{1}{3})$; $\nu := 2$;
      $\boldsymbol{m} := \boldsymbol{m} + \delta_m$ $\boldsymbol{d} := \boldsymbol{d}'$; break;
    $\mu := \mu\nu$; $\nu := 2\nu$;
$d := -\boldsymbol{d} \cdot \boldsymbol{c}$;
$\boldsymbol{p}_c := \frac{1}{2}(\boldsymbol{T}_B^{-1} S^B(-\boldsymbol{T}_B \boldsymbol{c}(\boldsymbol{m})) + \boldsymbol{r}^B + \boldsymbol{T}_A^{-1} S^A(\boldsymbol{T}_A \boldsymbol{c}(\boldsymbol{m})) + \boldsymbol{r}^A)$;

local spherical coordinate (θ, ϕ) is obtained through the following function

$$\theta = \text{atan2}(\text{Sign}(n_y)|r_y n_y|^{\frac{1}{2-\epsilon_1}}, \text{Sign}(n_x)|r_x n_x|^{\frac{1}{2-\epsilon_1}}) \tag{23a}$$

$$\phi = \text{atan2}(\text{Sign}(n_z)|r_z n_z|\cos(\theta)|^{2-\epsilon_2}|^{\frac{1}{2-\epsilon_2}}, |r_x n_x|^{\frac{1}{2-\epsilon_2}}) \tag{23b}$$

where the term atan2(x, y)[1] is the arctangent function of $\frac{x}{y}$ producing results in the range $(-\pi, \pi]$.

Consequently, the normal-surface relation can be established by combining Eqs. (22) and (23), i.e., $\boldsymbol{n} \rightarrow (\theta, \phi) \rightarrow \boldsymbol{x}$. Some important geometric properties (e.g., volume, the moment of inertia, and curvature) of a superellipsoid are not presented here for brevity, but we refer the reader to the literature [15,27]. Fig. 11 demonstrates three packing states of superellipsoidal particles falling freely into a box using *SudoDEM*.

As for poly-superellipsoids [24], the principal elongation along the positive and negative directions of x, y and z axes r_x^+, r_y^+, r_z^+ and r_x^-, r_y^-, r_z^- are properly selected to substitute r_x, r_y and r_z in Eqs. (22) and (23) according to the following relations

$$r_x = r_x^+ \quad \text{if} \quad n_x \geq 0 \quad \text{else} \quad r_x^- \tag{24a}$$

$$r_y = r_y^+ \quad \text{if} \quad n_y \geq 0 \quad \text{else} \quad r_y^- \tag{24b}$$

$$r_z = r_z^+ \quad \text{if} \quad n_z \geq 0 \quad \text{else} \quad r_z^- \tag{24c}$$

Clearly, the normal-surface relation for a poly-superellipsoid is readily established with Eq. (24) based on that for a superellipsoid, which implies that the contact detection among poly-superellipsoids holds almost the same computational efficiency as that among superellipsoids, but poly-superellipsoids provide a broader range of particle shapes.

5. The Gilbert–Johnson–Keerthi algorithm

The Gilbert–Johnson–Keerthi (GJK) algorithm [37] is an efficient algorithm for contact detection of convex bodies, especially convex polyhedrons, which has been popularly applied in computer games and graphic simulations [38], e.g., open-source physics engines such as Bullet (https://pybullet.org/) and Chrono (https://projectchrono.org/). As a sophisticated extension, we introduce the GJK algorithm with some variants into granular modeling with DEM. Since the underlying mathematics of GJK is complicated in detail, only the concept and its implementation are briefly depicted in this section.

[1] It is the four-quadrant arctangent function as implemented in many programming languages (see https://en.wikipedia.org/wiki/Atan2).

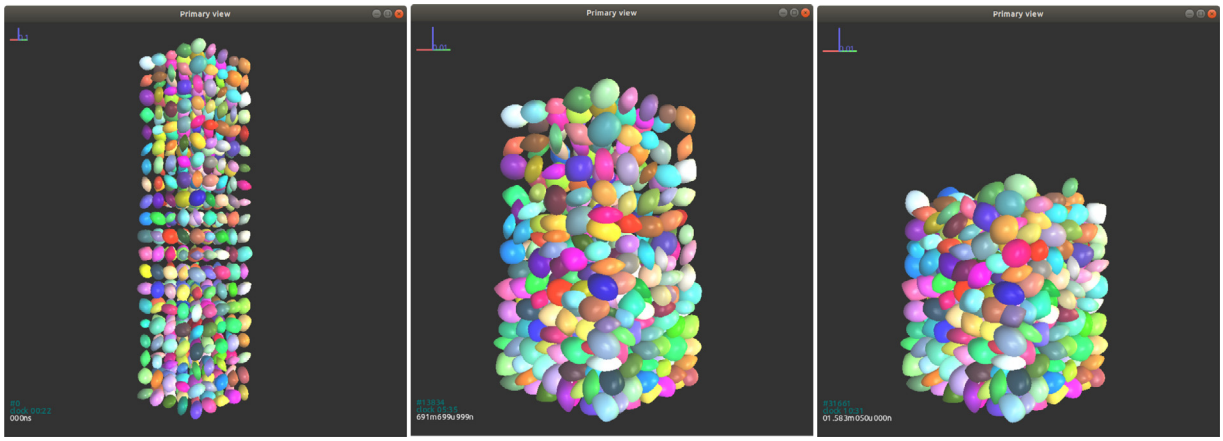

Fig. 11. Configurations of superellipsoids during packing under gravity.

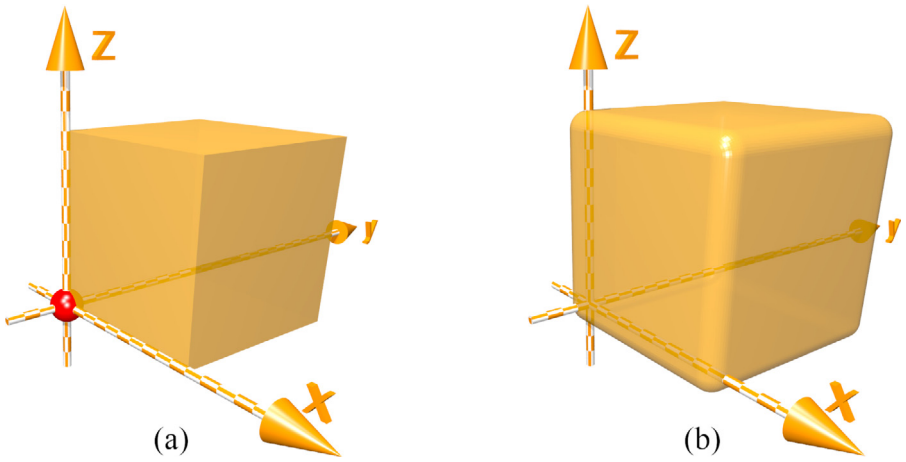

Fig. 12. (a) A cube $\mathbb{A}$ and a sphere $\mathbb{B}$ and (b) their Minkowski sum $\mathbb{C}$ (a rounded cube).

5.1. Minkowski sum and difference

Given two point sets $\mathbb{A}$ and $\mathbb{B}$ in dimension $\mathbb{R}^n$, the Minkowski sum $\mathbb{C}$ of $\mathbb{A}$ and $\mathbb{B}$ is given by

$$\mathbb{C} = \mathbb{A} \oplus \mathbb{B} = \{a + b | a \in \mathbb{A}, b \in \mathbb{B}\} \tag{25}$$

whilst the Minkowski difference $\mathbb{D}$ of $\mathbb{A}$ and $\mathbb{B}$ is defined as

$$\mathbb{D} = \mathbb{A} \ominus \mathbb{B} = \{a - b | a \in \mathbb{A}, b \in \mathbb{B}\} \tag{26}$$

Note that Minkowski sum is commutative while Minkowski difference is not, i.e., $\mathbb{A} \oplus \mathbb{B} = \mathbb{B} \oplus \mathbb{A}$ and $\mathbb{A} \ominus \mathbb{B} \neq \mathbb{B} \ominus \mathbb{A}$. The Minkowski difference $\mathbb{D}$ can be regarded as the Minkowski sum of $\mathbb{A}$ and the mirror of $\mathbb{B}$, i.e., $\mathbb{A} \ominus \mathbb{B} = \mathbb{A} \oplus (-\mathbb{B})$ where '$-$' denotes the mirror of $\mathbb{B}$ with respect to the origin. Intuitively, as shown in Fig. 12(a), given a cube $\mathbb{A}$ at the first octant of the coordinate system and a sphere $\mathbb{B}$ centered at the origin, the Minkowski sum $\mathbb{C}$ of $\mathbb{A}$ and $\mathbb{B}$ can be obtained by sweeping the sphere on the surface of the cube, yielding a rounded cube as shown in Fig. 12(b).

The Minkowski difference of two arbitrary convex bodies holds an exciting property, which offers a quick check whether the two bodies (convex sets) are touching with each other (with overlap) or not. This property says [38], if the two bodies $\mathbb{A}$ and $\mathbb{B}$ share an overlap, then their Minkowski difference $\mathbb{D}$ encloses the origin, vice versa, as shown in Fig. 13(a); if there is no overlap between the two bodies $\mathbb{A}$ and $\mathbb{B}$, then the Minkowski difference $\mathbb{D}$ does not enclose the origin and vice versa, as shown in Fig. 13(b). The GJK algorithm provides an efficient approach for checking whether the Minkowski difference $\mathbb{D}$ is enclosing the origin or not regardless of body shapes of $\mathbb{A}$ and $\mathbb{B}$.

5.2. Support point and support function

Given an arbitrary direction $\boldsymbol{v}$, a support point $\boldsymbol{p}$ is defined as the furthest point within a particle (a convex set) along $\boldsymbol{v}$, i.e., $S(\boldsymbol{v}) = \sup\{\boldsymbol{v} \cdot \boldsymbol{p} | \boldsymbol{p} \in \Gamma\}$ where Γ is the particle surface. Such a relation is denoted as a support function $\hat{S}(\boldsymbol{v})$. For a smooth surface, e.g., a poly-superellipsoid, as shown in Fig. 14, the support function is a one-to-one function between the surface point $\boldsymbol{p}$ and the surface outward normal $\boldsymbol{n}$, i.e., $\boldsymbol{p} = S(\boldsymbol{n})$. In contrast, for a non-smooth surface, e.g., a polyhedron, there might be several potential support points at a facet or edge for a given direction $\boldsymbol{v}$ or only single support point at a vertex for several directions. Special attention should be paid to the first case, which occurs when the direction $\boldsymbol{v}$ is perpendicular to a facet or edge of the polyhedron. In general, the facet center or edge middle is taken as the support point for the sake of numerical implementation. It is worth noting that there might be a non-smooth shift of force point for some special cases, but both the magnitude and direction of the force can be smooth, which will not cause numerical issues.

The support function $\hat{S}_{\mathbb{C}}(\boldsymbol{v})$ for a Minkowski sum $\mathbb{C}$ of two convex sets $\mathbb{A}$ and $\mathbb{B}$ is given as

$$\hat{S}_{\mathbb{C}}(\boldsymbol{v}) = \hat{S}_{\mathbb{A}}(\boldsymbol{v}) + \hat{S}_{\mathbb{B}}(\boldsymbol{v}) \tag{27}$$

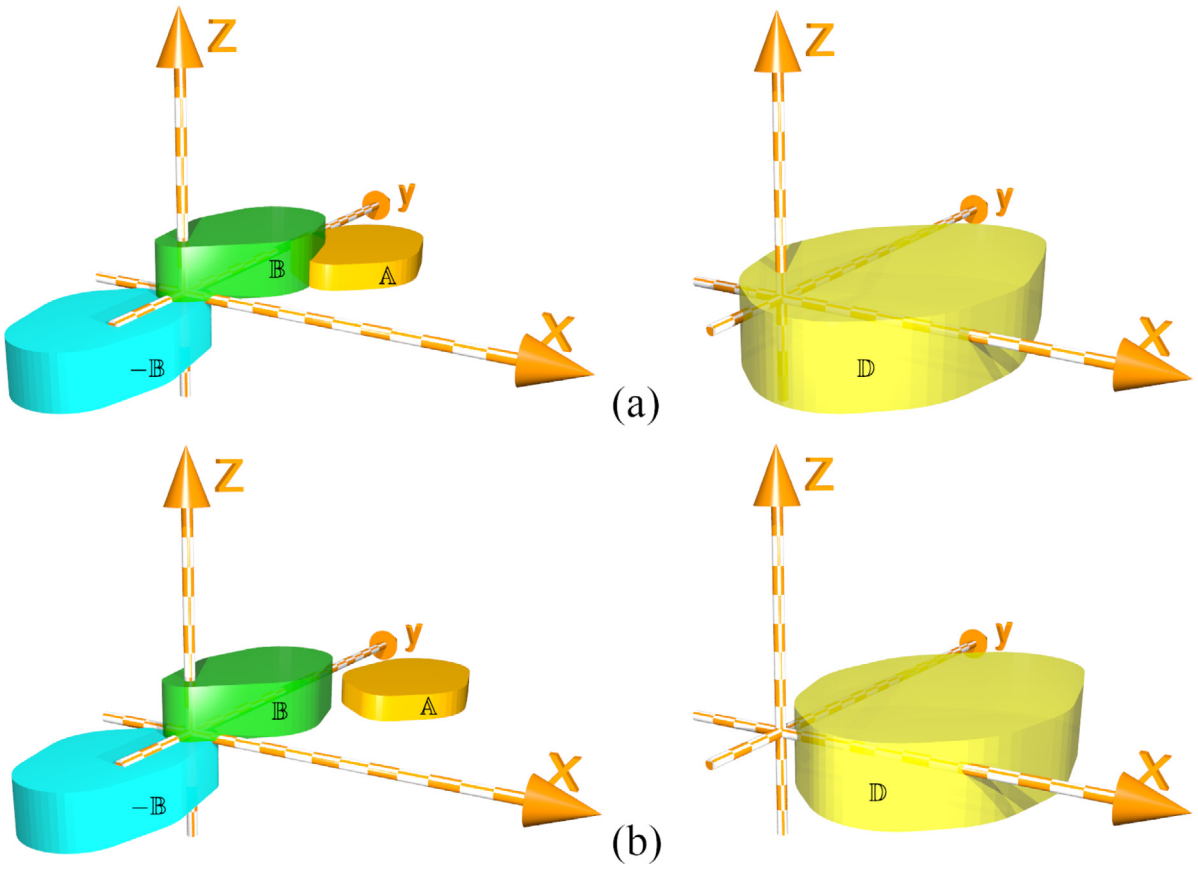

Fig. 13. Minkowski difference $\mathbb{D} = \mathbb{A} \ominus \mathbb{B}$ between two arbitrary convex shapes $\mathbb{A}$ and $\mathbb{B}$ (a) with contact and (b) without contact, where $-\mathbb{B}$ is the mirror of $\mathbb{B}$ with respect to the origin.

For the mirror $-\mathbb{B}$ of $\mathbb{B}$ with respect to the origin, its support point reads

$$\hat{S}_{-\mathbb{B}}(\boldsymbol{v}) = -\hat{S}_{\mathbb{B}}(-\boldsymbol{v}) \tag{28}$$

Therefore, the support function $\hat{S}_{\mathbb{D}}(\boldsymbol{v})$ for a Minkowski difference $\mathbb{D}$ of $\mathbb{A}$ and $\mathbb{B}$ can be obtained by

$$\hat{S}_{\mathbb{D}}(\boldsymbol{v}) = \hat{S}_{\mathbb{A}}(\boldsymbol{v}) - \hat{S}_{\mathbb{B}}(-\boldsymbol{v}) \tag{29}$$

5.3. Iterative GJK

The GJK algorithm works in a simplex-algorithm manner by iteratively searching the closest point within a Minkowski difference to the origin. For two contacting particles $\mathbb{A}'$ and $\mathbb{B}'$, as shown in Fig. 15, it converges fast in general to find a simplex enclosing the origin within the Minkowski difference $\mathbb{D}'$, indicating that the GJK is efficient to query two contacting particles. Nevertheless, the shortest distance (referred to as penetration depth hereafter) between the two particles cannot be obtained directly, which instead is often solved by the so-called expanding-polytope algorithm (EPA) [38]. The EPA expands the simplexes iteratively in an inverse-like sequence of GJK, which significantly increases the computational intensity. Thus, the EPA is not adopted here to avoid such an additional heavy computation. Nevertheless, a special particle shape is introduced as a Minkowski sum of a kernel shape and a sweeping sphere. Researchers coined such a shape as dilated polyhedron [39] or spheropolyhedron [40] when the kernel shape is a polyhedron. As shown in Fig. 15(a), particles $\mathbb{A}'$ and $\mathbb{B}'$ are constructed by enlarging the kernel particles $\mathbb{A}$ and $\mathbb{B}$ with a sweeping sphere, respectively. The radii of sweeping spheres are sufficiently large

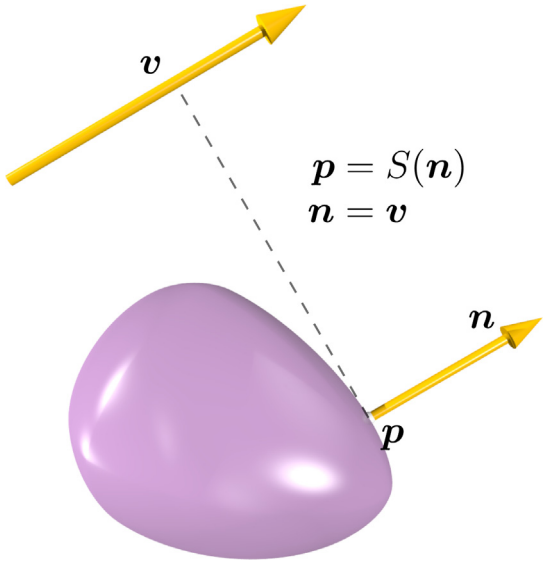

Fig. 14. A support point $\boldsymbol{p}$ on a particle surface with a given direction vector v and a support function $S(\boldsymbol{n})$. Note: n is the outward normal of particle surface at the support point $\boldsymbol{p}$ [24].

to ensure that the possible overlap occurs only in the sweeping volume. That being said, the Minkowski difference $\mathbb{D}'$ of the enlarged particles encloses the origin, while the Minkowski difference $\mathbb{D}$ of the kernel particles does not, as shown in Fig. 15(b). Therefore, the penetration depth d between particles $\mathbb{A}'$ and $\mathbb{B}'$ is

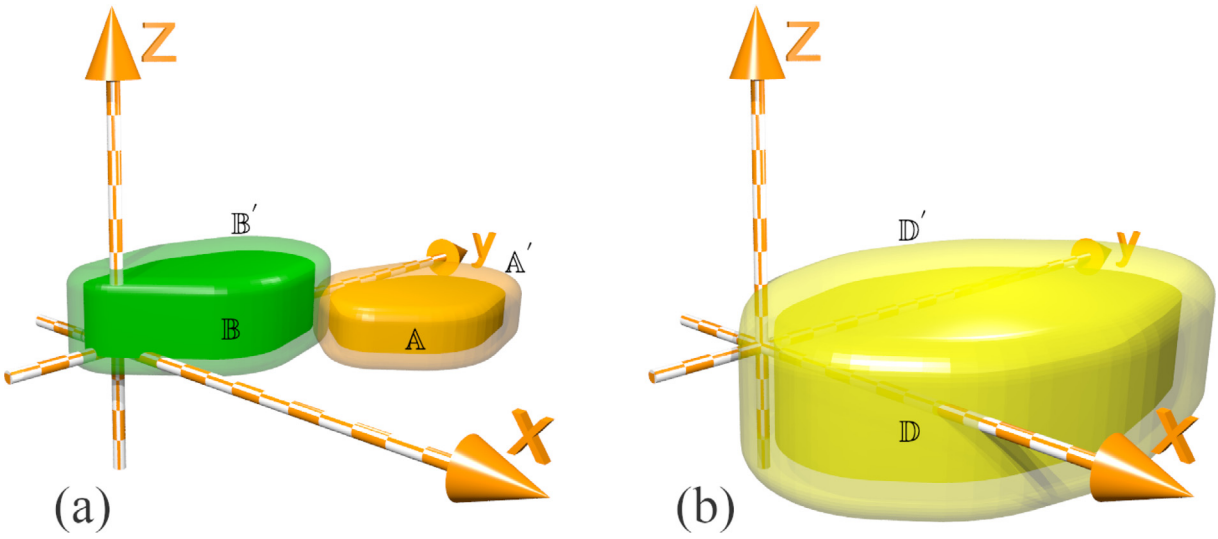

Fig. 15. (a) Particles $\mathbb{A}'$ and $\mathbb{B}'$ enlarged by a sphere sweeping over $\mathbb{A}$ and $\mathbb{B}$, respectively; (b) Minkowski difference $\mathbb{D} = \mathbb{A} \ominus \mathbb{B}$ and $\mathbb{D}' = \mathbb{A}' \ominus \mathbb{B}'$.

given by

$$d = \delta_{A'} + \delta_{B'} - d_{AB} \tag{30}$$

where $\delta_{A'}$ and $\delta_{B'}$ are the radii of the sweeping spheres for particles $\mathbb{A}'$ and $\mathbb{B}'$, respectively; d_{AB} is the shortest distance between the two kernel particles $\mathbb{A}$ and $\mathbb{B}$.

Algorithm 2: The GJK algorithm for searching contact penetration depth.

Input: Kernel particles $\mathbb{A}$ and $\mathbb{B}$; sweeping spheres $\delta_{A'}$ and $\delta_{B'}$; support functions S^A and S^B, rotation matrices $\boldsymbol{T}^A$ and $\boldsymbol{T}^B$; cached contact normal $\boldsymbol{c}$.

Output: Contact points $\boldsymbol{p}^{A'}$ and $\boldsymbol{p}^{B'}$; penetration depth d; flag *touching*.

1 $d := 0$; $l := 0$; $\boldsymbol{v} := -\boldsymbol{c}$; $\boldsymbol{C} := \emptyset$;
2 **do**
3 $\boldsymbol{v} := \boldsymbol{v}/\|\boldsymbol{v}\|$;
4 $\boldsymbol{\delta}^A = \boldsymbol{v}\delta_{A'}$; $\boldsymbol{\delta}^B = \boldsymbol{v}\delta_{B'}$;
5 $\boldsymbol{p} = \boldsymbol{T}_A^{-1} S^A(-\boldsymbol{T}_A \boldsymbol{v}) + \boldsymbol{r}^A$;//support point of kernel $\mathbb{A}$
6 $\boldsymbol{q} = \boldsymbol{T}_B^{-1} S^B(\boldsymbol{T}_B \boldsymbol{v}) + \boldsymbol{r}^B$;//support point of kernel $\mathbb{B}$
7 $\boldsymbol{w} = \boldsymbol{p} - \boldsymbol{q}$;//support point of Minkowski difference $\mathbb{D}$
8 $l := \max(\|\boldsymbol{w}\|, l)$;
9 **if** $\boldsymbol{v} \cdot \boldsymbol{w} > 0$ and $\|\boldsymbol{w}\| > (\delta_{A'} + \delta_{B'})$ **then**
10 touching:= false;//no contact break;
11 **if** $\boldsymbol{w}$ is already a simplex vertex or $d^2 - \boldsymbol{v} \cdot \boldsymbol{w} \le d^2 \epsilon_1$ **then**
12 compute points $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ on particles $\mathbb{A}$ and $\mathbb{B}$, respectively;
13 $\boldsymbol{p}^{A'} := \boldsymbol{p}^A - \boldsymbol{\delta}^A$; $\boldsymbol{p}^{B'} := \boldsymbol{p}^B + \boldsymbol{\delta}^B$;
14 touching:= true;
15 break;
16 $\boldsymbol{C} := \boldsymbol{C} \cup \{\boldsymbol{w}\}$;//add another simplex vertex
17 $\boldsymbol{v}$:= the closest point in the simplex $\boldsymbol{C}$;
18 $\boldsymbol{C}$:= the smallest subset of $\boldsymbol{C}$ with its convex hull containing $\boldsymbol{v}$;
19 $d := \|\boldsymbol{v}\|$;
20 **while** $\|\boldsymbol{v}\| > \epsilon l$;
21 $\boldsymbol{c} := -\boldsymbol{v}/\|\boldsymbol{v}\|$; $d := (\boldsymbol{p}^{A'} - \boldsymbol{p}^{B'}) \cdot \boldsymbol{c}$;

With Eq. (30), it is clear that finding the penetration depth d of two particles $\mathbb{A}'$ and $\mathbb{B}'$ is equivalent to finding the shortest distance d_{AB} between their kernel particles $\mathbb{A}$ and $\mathbb{B}$. In addition, d_{AB} is equal to the shortest distance of the Minkowski difference $\mathbb{D}$ to the origin that can be searched iteratively by using the GJK algorithm. Algorithm 2 summarizes the pseudo-codes of solving the penetration depth between particles $\mathbb{A}'$ and $\mathbb{B}'$. During the iteration, the set of simplex vertexes $\boldsymbol{C}$ is updated with the support point $\boldsymbol{w}$ of the Minkowski difference $\mathbb{D}$ with respect to a searching direction $-\boldsymbol{v}$. Referring to Lines 16–18 in Algorithm 2, $\boldsymbol{C}$ is initialized to an empty set and subsequently filled with 1, 2, 3, or up to 4 points corresponding to a simplex of a point, a line segment, a triangle, or a tetrahedron, respectively; it is, then, straightforward to compute the closest point $\boldsymbol{v}$ in the simplex to the origin; the simplex is further reduced to be the smallest subset of $\boldsymbol{C}$ such that its convex hull contains $\boldsymbol{v}$. Clearly, the point $\boldsymbol{v}$ is generally closer to the origin after each iteration. Nevertheless, the support point $\boldsymbol{w}$ may become stationary once $\boldsymbol{w}$ is already a simplex vertex at the last iteration. In addition, the change in $\boldsymbol{v}$ is likely to be relatively small with sufficient iterations. Thus, we terminate the searching routine for both cases as listed at Line 11 of Algorithm 2. The routine also stops either when there is no contact between particles $\mathbb{A}'$ and $\mathbb{B}'$ or when $\|\boldsymbol{v}\|$ is significantly small, as listed at Lines 9 and 15 of Algorithm 2.

5.4. Applied to cones, cylinders and polyhedrons

The critical ingredient of the GJK algorithm is its support functions of the two kernel particles. With respect to the support function, we group the kernel particle shapes into two broad categories of primitive shapes, e.g., cones, cylinder, and polyhedrons, with/without analytical support functions, respectively. For shapes with analytical support functions, a cone, a cylinder, and a box are exemplified here, as shown in Fig. 16.

Given a cone with base radius r and height h, as shown in Fig. 16(a), the support function $S(\boldsymbol{v})$ is given by

$$S(\boldsymbol{v}) = \begin{cases} \frac{3h}{4}\boldsymbol{v}_h, & \text{if } (\boldsymbol{v}_h \cdot \boldsymbol{v})\sqrt{r^2+h^2} \ge r, \\ -\frac{h}{4}\boldsymbol{v}_h + r\frac{\boldsymbol{v}-(\boldsymbol{v}_h\cdot\boldsymbol{v})\boldsymbol{v}_h}{\|\boldsymbol{v}-(\boldsymbol{v}_h\cdot\boldsymbol{v})\boldsymbol{v}_h\|}, & \text{else if } (\boldsymbol{v}_h \cdot \boldsymbol{v})\sqrt{r^2+h^2} < r \text{ and } \boldsymbol{v}_h \times \boldsymbol{v} \ne \boldsymbol{0}, \\ -\frac{h}{4}\boldsymbol{v}_h, & \text{otherwise.} \end{cases} \tag{31}$$

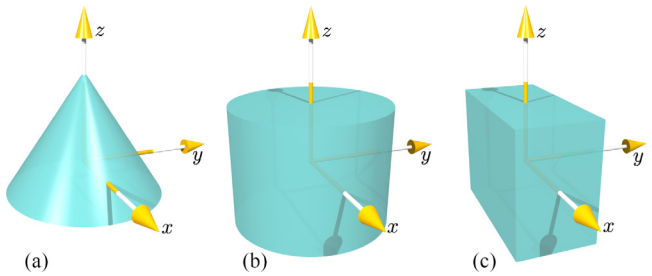

Fig. 16. (a) A cone, (b) a cylinder, and (c) a box with fixed local Cartesian coordinate systems at their centers of mass.

where $\boldsymbol{v}_h$ is a direction vector pointing from the base center to the apex of the cone. By contrast, the support function $S(\boldsymbol{v})$ of a cylinder can be more simplified due to an extra centrosymmetric property, as shown in Fig. 16(b), which can be given by

$$S(\boldsymbol{v}) = \begin{cases} \frac{h}{2}(\boldsymbol{v}_h \cdot \boldsymbol{v})\boldsymbol{v}_h + r\frac{\boldsymbol{v}-(\boldsymbol{v}_h\cdot\boldsymbol{v})\boldsymbol{v}_h}{\|\boldsymbol{v}-(\boldsymbol{v}_h\cdot\boldsymbol{v})\boldsymbol{v}_h\|}, & \text{if } \boldsymbol{v}_h \times \boldsymbol{v} \neq \boldsymbol{0}, \\ \frac{h}{2}(\boldsymbol{v}_h \cdot \boldsymbol{v})\boldsymbol{v}_h, & \text{otherwise.} \end{cases} \tag{32}$$

where $\boldsymbol{v}_h$ is a direction vector pointing from the bottom center to the top center of the cylinder; r and h are the base radius and height, respectively. So far, it has been clear that for a facet (e.g., the base of a cone) the candidate support point can only be from the edge of the center of the facet. Thus, for a polyhedron, its vertexes, edges, centers of edges, and centers of facets hold the candidate support point for a given searching direction $\boldsymbol{v}$. That is to say, the support function $S(\boldsymbol{v})$ cannot be expressed analytically for a general polyhedron. Hence, special attention should be paid to constructing the data structure for the adjacency in facets, edges, and vertexes, thereby achieving an efficient search of the support point. Nevertheless, for simple polyhedrons, the support function can be readily obtained. Taking a box as an example, referring to Fig. 16(a), the support function is given by

$$S(\boldsymbol{v}(v_1, v_2, v_3)) = \frac{1}{2}(\text{Sign}(v_1)l_1, \text{Sign}(v_2)l_2, \text{Sign}(v_3)l_3) \tag{33}$$

where l_1, l_2 and l_3 are the length, width, and height of the box, respectively; Sign() is the signum function.

As a demonstration, we simulate the granular packings of monodisperse cones, cylinders, and cubes under gravity in *SuDoDEM*. The initial and final packing states of cones, cylinders, and cubes are shown in Fig. 17, where the cubic container is not shown for better visualization.

6. Hybrid algorithm of PCN and GJK

6.1. Bridging PCN and GJK

The candidate contact points $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ in Eq. (16) can be rewritten as

$$\boldsymbol{p}^A = \hat{\boldsymbol{S}}^A(\boldsymbol{c}), \quad \boldsymbol{p}^B = \hat{\boldsymbol{S}}^B(-\boldsymbol{c}) \tag{34}$$

such that the candidate penetration $\boldsymbol{d}$ is given as

$$\boldsymbol{d} = \hat{\boldsymbol{S}}^B(-\boldsymbol{c}) - \hat{\boldsymbol{S}}^A(\boldsymbol{c}) = -\hat{\boldsymbol{S}}_{A-B}(\boldsymbol{c}) \tag{35}$$

where $\hat{\boldsymbol{S}}_{A-B}$ is the support function of Minkowski difference $\mathbb{D}$ between particles $\mathbb{A}$ and $\mathbb{B}$.

The optimization problem introduced in Eq. (20) can be recast into the following unconstrained optimization problem:

$$\min_{\boldsymbol{m}} |d| = \min_{\alpha,\beta} \|\boldsymbol{p}^B - \boldsymbol{p}^A\| = \min_{\boldsymbol{c}} \|\hat{\boldsymbol{S}}_{A-B}(\boldsymbol{c})\| \tag{36}$$

which suggests that the PCN and GJK algorithms can be utilized to search the same contact penetration alternatively. That is to say, both algorithms can be seamlessly switched from each other during the searching routine. In general, the LM algorithm involved in the PCN algorithm has an excellent performance of convergence (most often less than ten iterations). However, the searching routine may fail or converge significantly slowly either when the computed contact point from one particle surface is outside the other particle, or when the LM encounters an extremely small step in $\|\boldsymbol{m}\|$ (e.g., 10^{-5} rad for very flat face-to-face contacts).

Algorithm 3: The hybrid algorithm of PCN and GJK.

Input: Particle positions $\boldsymbol{r}^A$ and $\boldsymbol{r}^B$, surface functions S^A and S^B, erosion $\delta_{A'}$ and $\delta_{B'}$, rotation matrices $\boldsymbol{T}_A$ and $\boldsymbol{T}_B$; cached parameters α_0 and β_0.
Output: Parameter $\boldsymbol{m}$; flag *touching*; penetration depth d; contact point $\boldsymbol{p}_c$.

$\boldsymbol{m} = [\alpha_0, \beta_0]^T$;
$\boldsymbol{d} := \boldsymbol{T}_B^{-1}S^B(-\boldsymbol{T}_B\boldsymbol{c}(\boldsymbol{m})) + \boldsymbol{r}^B - \boldsymbol{T}_A^{-1}S^A(\boldsymbol{T}_A\boldsymbol{c}(\boldsymbol{m})) - \boldsymbol{r}^A$;
the main routine of PCN: Lines 2-21 of Algorithm 2;
if PCN converges slowly **then**
  $d := 0$; $l := 0$; $\boldsymbol{v} := -\boldsymbol{c}$; $\boldsymbol{C} := \emptyset$;
  do
    $\boldsymbol{v} := \boldsymbol{v}/\|\boldsymbol{v}\|$;
    $\boldsymbol{\delta}^A = \boldsymbol{v}\delta_{A'}$; $\boldsymbol{\delta}^B = \boldsymbol{v}\delta_{B'}$;
    $\boldsymbol{p} = \boldsymbol{T}_A^{-1}S^A(-\boldsymbol{T}_A\boldsymbol{v}) + \boldsymbol{r}^A + \boldsymbol{\delta}^A$;//support point of eroded $\mathbb{A}$
    $\boldsymbol{q} = \boldsymbol{T}_B^{-1}S^B(\boldsymbol{T}_B\boldsymbol{v}) + \boldsymbol{r}^B - \boldsymbol{\delta}^B$;//support point of eroded $\mathbb{B}$
    $\boldsymbol{w} = \boldsymbol{p} - \boldsymbol{q}$;//support point of Minkowski difference $\mathbb{D}$
    $l := \max(\|\boldsymbol{w}\|, l)$;
    check the convergence of GJK: Lines 11-15 of Algorithm 2;
    update the simplex: Lines 16-18 of Algorithm 2;
    $d := \|\boldsymbol{v}\|$;
  while $\|\boldsymbol{v}\| > \epsilon l$;
  $\boldsymbol{c} := -\boldsymbol{v}/\|\boldsymbol{v}\|$; $\boldsymbol{d} := (\boldsymbol{p}^{A'} - \boldsymbol{p}^{B'})$;
$d := -\boldsymbol{d} \cdot \boldsymbol{c}$;
$\boldsymbol{p}_c := \frac{1}{2}(\boldsymbol{T}_B^{-1}S^B(-\boldsymbol{T}_B\boldsymbol{c}(\boldsymbol{m})) + \boldsymbol{r}^B + \boldsymbol{T}_A^{-1}S^A(\boldsymbol{T}_A\boldsymbol{c}(\boldsymbol{m})) + \boldsymbol{r}^A)$;

Therefore, a hybrid approach of PCN and GJK is proposed to achieve a more robust and efficient solution, which has been depicted in our previous work [24]. Algorithm 3 summarizes the pseudo-codes of the hybrid algorithm. Note that the searching direction $\boldsymbol{v}$ in GJK is a position vector of a point in the updated simplex, which is anti-parallel to the contact normal (direction) $\boldsymbol{c}$ that is assumed to point from particle $\mathbb{A}$ to particle $\mathbb{B}$. Besides, it is noteworthy that particles $\mathbb{A}$ and $\mathbb{B}$ in PCN will be eroded by $\delta_{A'}$ and $\delta_{B'}$, respectively, before searching with GJK, referring to Lines 8 and 9 in Algorithm 3. Consequently, the entire searching routine is implemented with GJK only, as mentioned in Section 5. Nevertheless, the erosion procedure is not an inverse of sweeping a sphere over a kernel particle, which may cause a side effect on the eroded surface of a non-smooth particle. Taking a polyhedron as an example, the joining vertex of two adjacent edges is not identical after eroding along their corresponding inward normals. Therefore, the hybrid algorithm is only applicable to smooth particles.

6.2. Applied to poly-superellipsoids

For the sake of computational efficiency, the support function $S(\boldsymbol{n})$ of a poly-superellipsoid can be written without explicitly

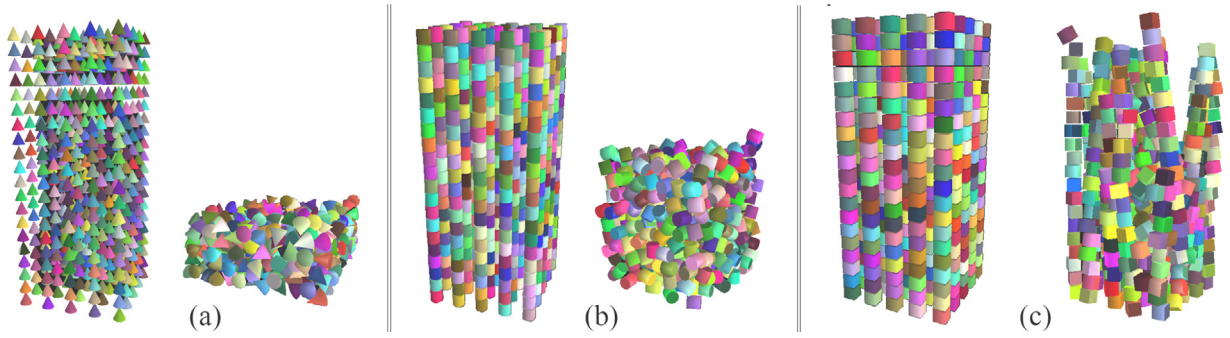

Fig. 17. Initial and final packing configurations of (a) cones, (b) cylinders and (c) cubes.

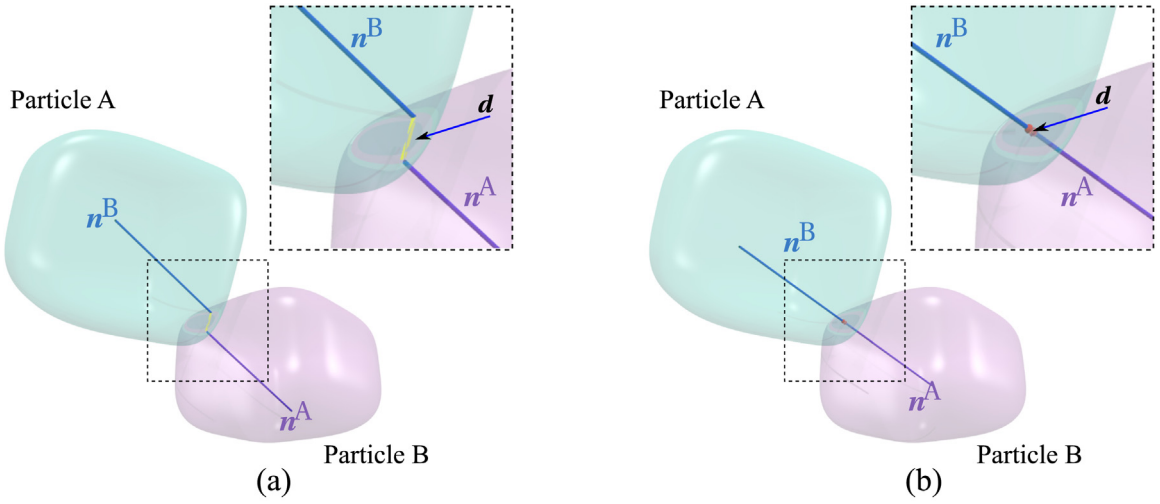

Fig. 18. Example of two particles with a flat contact: (a) PCN and (b) the hybrid algorithm of PCN and GJK [24].

computing θ and ϕ in Eq. (23), given as [24]

$$x = \frac{1}{2}\mathrm{Sign}(n_x)\left[(1+\mathrm{Sign}(n_x))r_x^+ + (1-\mathrm{Sign}(n_x))r_x^-\right]\alpha_1^{\epsilon_1/2}\alpha_2^{\epsilon_2/2} \tag{37a}$$

$$y = \frac{1}{2}\mathrm{Sign}(n_y)\left[(1+\mathrm{Sign}(n_y))r_y^+ + (1-\mathrm{Sign}(n_y))r_y^-\right] \times (1-\alpha_1)^{\epsilon_1/2}(1-\alpha_2)^{\epsilon_2/2} \tag{37b}$$

$$z = \frac{1}{2}\mathrm{Sign}(n_z)\left[(1+\mathrm{Sign}(n_z))r_z^+ + (1-\mathrm{Sign}(n_z))r_z^-\right](1-\alpha_2)^{\epsilon_2/2} \tag{37c}$$

where α_1 and α_2 are equal to $\cos^2(\theta)$ and $\cos^2(\phi)$, respectively, given as follows

$$\alpha_1 = \begin{cases} \left(1+\left|\frac{r_y n_y}{r_x n_x}\right|^{\frac{2}{2-\epsilon_1}}\right)^{-1}, & \text{if } n_x \neq 0, \\ 0, & \text{otherwise.} \end{cases}$$

and

$$\alpha_2 = \begin{cases} \left(1+\left|\frac{r_z n_z}{r_x n_x}\right|^{\frac{2}{2-\epsilon_2}}\left|\alpha_1\right|^{\frac{2-\epsilon_1}{2-\epsilon_2}}\right)^{-1}, & \text{if } n_x \neq 0, \\ \left(1+\left|\frac{r_z n_z}{r_y n_y}\right|^{\frac{2}{2-\epsilon_2}}\left|1-\alpha_1\right|^{\frac{2-\epsilon_1}{2-\epsilon_2}}\right)^{-1}, & \text{else if } n_y \neq 0, \\ 0, & \text{otherwise.} \end{cases}$$

As aforementioned, the computation by the GJK part is triggered when the PCN converges slowly in the hybrid algorithm of PCN and GJK. To offer a glance at the improvement of the hybrid algorithm, we set up a test of two particles with a flat contact, as shown in Fig. 18. Note that the initial guess of the contact normal $\boldsymbol{c}$ is the vector joining the mass centers of the two particles. The PCN with the LM algorithm fails due to slow convergence after 23 calls of the support function (Fig. 18(a)), while the hybrid algorithm succeeds after 12 calls of the support function (Fig. 18(b)). Notably, the hybrid algorithm has a much better performance than the PCN, which inherits merits from both PCN and GJK, thereby more robust and efficient. The major pros and cons of the three generic algorithms are summarized in Table 1.

Table 1
Major pros and cons of the three generic algorithms for contact detection.

Algorithm	Pros	Cons
PCN (with LM)	Robust and efficient	Prescribed $\boldsymbol{n} \to S(\boldsymbol{n})$
GJK (sphero-shape)	Any convex kernel shapes	Less efficient than PCN
The hybrid PCN–GJK	More robust and efficient than PCN	Prescribed $\boldsymbol{n} \to S(\boldsymbol{n})$

7. Simulation demos

Three examples are demonstrated here to show the robustness and versatility of *SudoDEM*. Interested readers can find more examples and their corresponding details of simulation setup in the quick guide [41] or our other published work, e.g., [15,24,27]. All demos run on an Ubuntu 18.04 Desktop with an Intel Core I7-6700 CPU (4 cores at 3.4 GHz, and 8 logic cores) and 16 GB RAM.

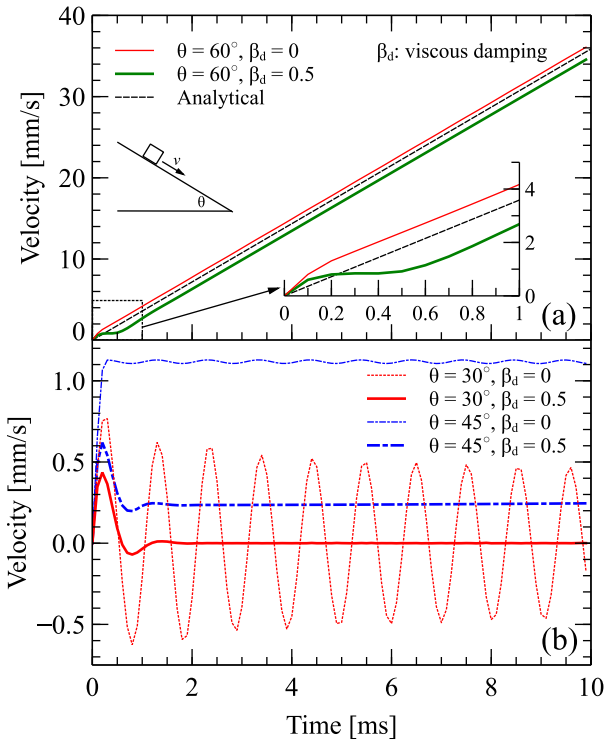

Fig. 19. Evolution of velocity of a cube sliding along a slope under gravity for different slope angles and viscous damping β_d.

7.1. Test of the friction law

The friction law is tested first by releasing a cube to possibly slide on a slope with an incline angle of θ under gravity. The cube has a length of 10 mm, mass density of 2650 kg/m^3. The coefficient of friction μ is set to 1.0, corresponding to a critical angle of $\theta_c = 45°$. Both the normal and tangential contact stiffnesses are set to 1×10^5 N/m. Besides, viscous damping with a coefficient $\beta_d = 0$ or 0.5 is applied to the contact, where $\beta_d = 0$ means no damping. Three slope angles $\theta = 30°$, 45° and 60° are selected to run the test under a gravitational acceleration g of 9.8 m/s^2. A total of six simulations are conducted for 10 ms each (the time step is 1×10^{-6} s). The velocity of the cube along the slope has been monitored with a sampling interval of 100 time steps, as shown in Fig. 19.

For $\theta > \theta_c$, i.e., $\theta = 60°$ in Fig. 5(a), the cube slides with an increasing velocity as expected. Moreover, the cube experiences a decrease in acceleration due to the development of the friction at the very beginning, and the acceleration becomes stationary after the friction reaching its maximum, i.e., $\mu G \cos(\theta)$ (G is the weight of the cube). The analytical acceleration of the cube is given by $a = g(sin\theta - \mu cos\theta)$. Notably, however, the simulated friction is calculated in terms of the relative tangential displacement of the cube with respect to the slope. It needs time for the cube to develop the friction from zero to the maximum. Hence, it is not surprising to see a larger acceleration than the analytical one at the very beginning. Moreover, the velocity of the cube is reduced to a smaller value at the beginning for the damped case. Changing the damping coefficient β_d can indeed render a match of the velocity with the analytical one, which is, however, not the focus of this work.

For $\theta < \theta_c$, i.e., $\theta = 30$ in Fig. 5(b), the cube will stay stationary on the slope in reality, while an oscillation can be observed in the simulation. The oscillation has a small amplitude but stays appreciable during the entire simulation for the non-damped case. The velocity can be practically damped quickly to zero if viscous damping is applied. It is worth noting that there are two critical factors contributing to the oscillation: a similar oscillation in the normal direction (normal oscillation) and the computational manner of the friction as aforementioned. Note that the normal oscillation makes the pressure oscillate around $Gsin(\theta)$ such that the numerical maximum friction oscillates as well.

For $\theta = \theta_c$, the cube maintains an initial velocity without acceleration in reality. However, the simulated cube may obtain a non-zero velocity due to the numerical development of friction (see $\theta = 45$ in Fig. 5(b)), even though the cube has no initial velocity at all. For the non-damped case, a small oscillation can be observed during the entire simulation, while the oscillation can be reduced quickly with the damped velocity much closer to zero for the damped case.

7.2. Heap formation

The test of heap formation is conducted to further prove the validity of the implementation of inter-particle friction. As shown in Fig. 20(a), an open channel with one end closed is set up under a particle generation box. A heap forms and gradually develops with continuous feeding of particles from the generation box. For each feeding, only a few particles are generated with random locations and orientations in the generation box before being released into the channel under gravity. In the simulation, the particles have an equivalent radius (i.e., the radius of a sphere with the same volume as the given particle) of 10 mm, mass density of 2650 kg/m^3, and friction coefficient of 0.5. The normal and tangential contact stiffnesses are set to 1×10^4 and $7 \times$

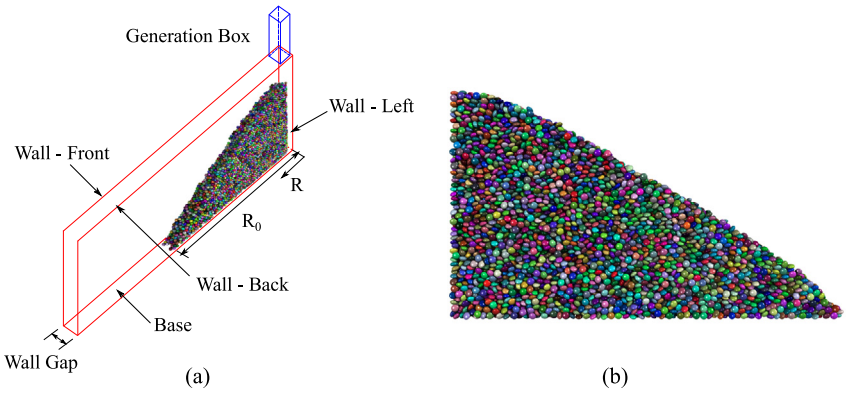

Fig. 20. (a) Simulation setup of heap formation and (b) a heap of superellipsoidal particles at rest, after [42].

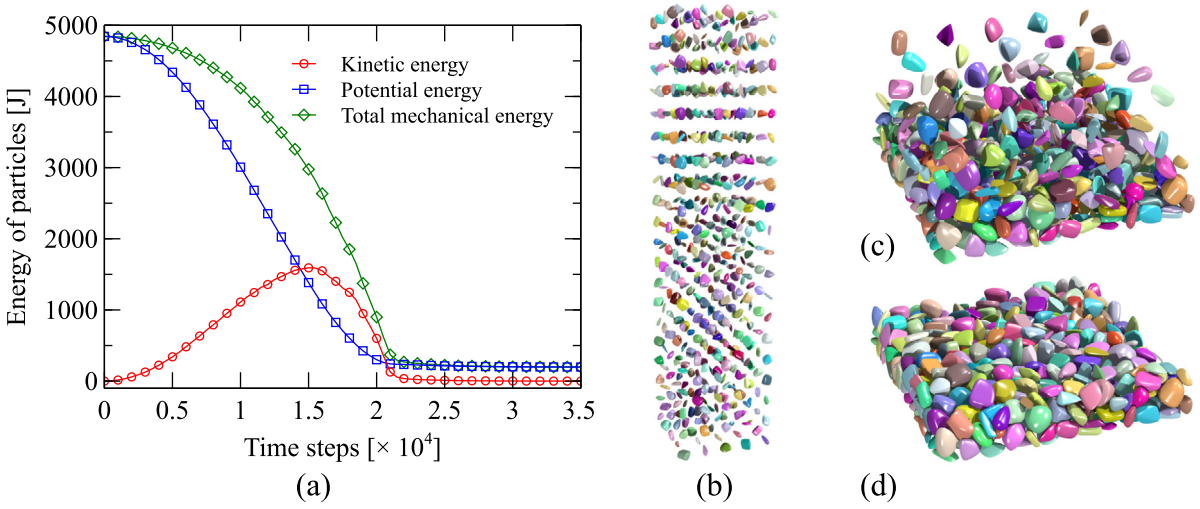

Fig. 21. Free falling of five hundred poly-superellipsoids under gravity: (a) variations of kinetic, potential and total energy of particles; snapshots at initial packing (b), after 15 000 (c) and 35 000 (d) time steps, after [24]. Note: the box is not shown for a better visualization.

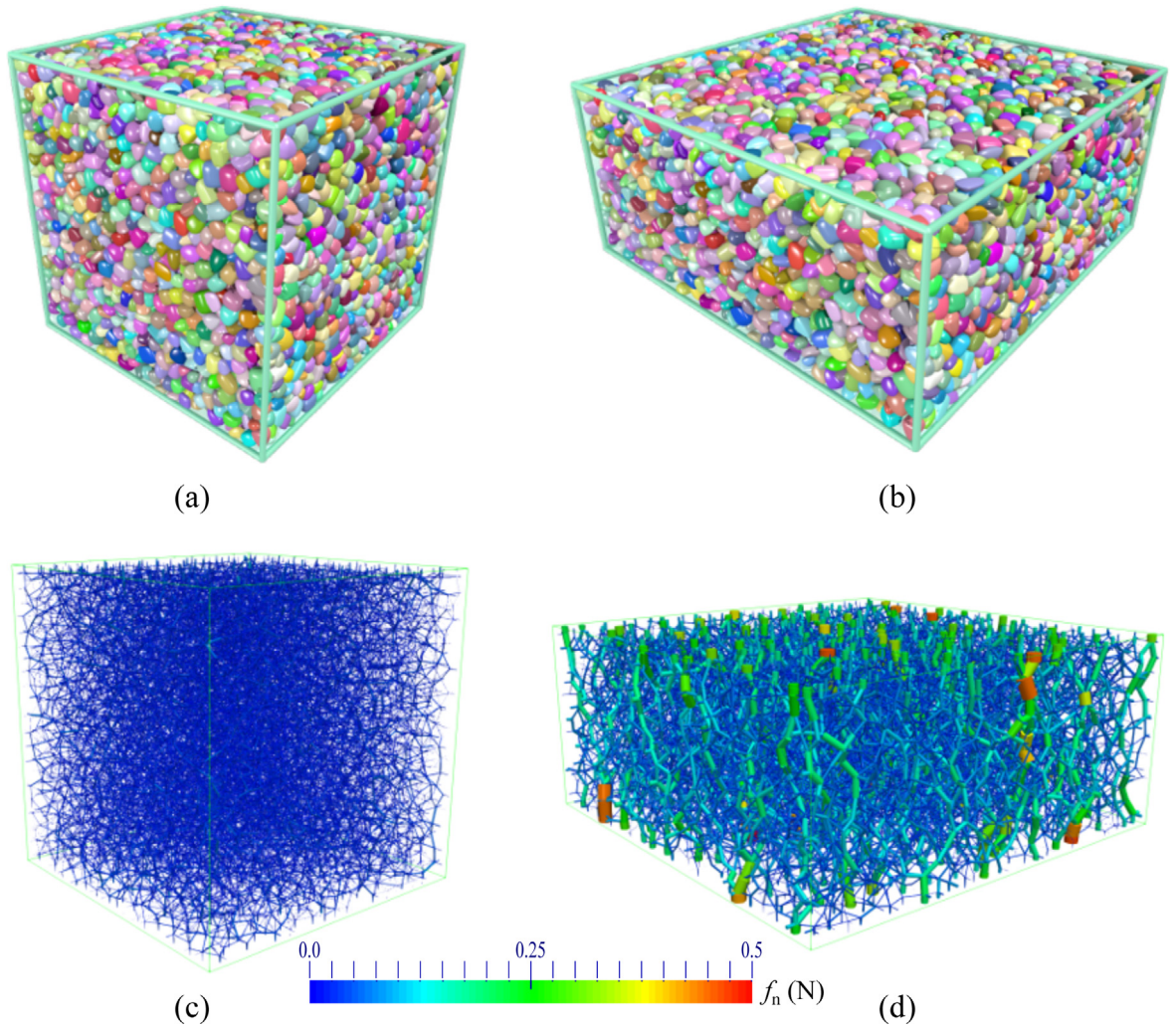

Fig. 22. Snapshots of a cubic specimen composed of 10 000 poly-superellipsoids at the initial (a) and final (b) states during triaxial compression, and its corresponding normal contact force chains at the initial (c) and final (d) states. Note: the color bar is for the magnitude of normal contact force f_n. The animation is available at https://sudodem.github.io/docs/images/triaxialtest-1.gif.

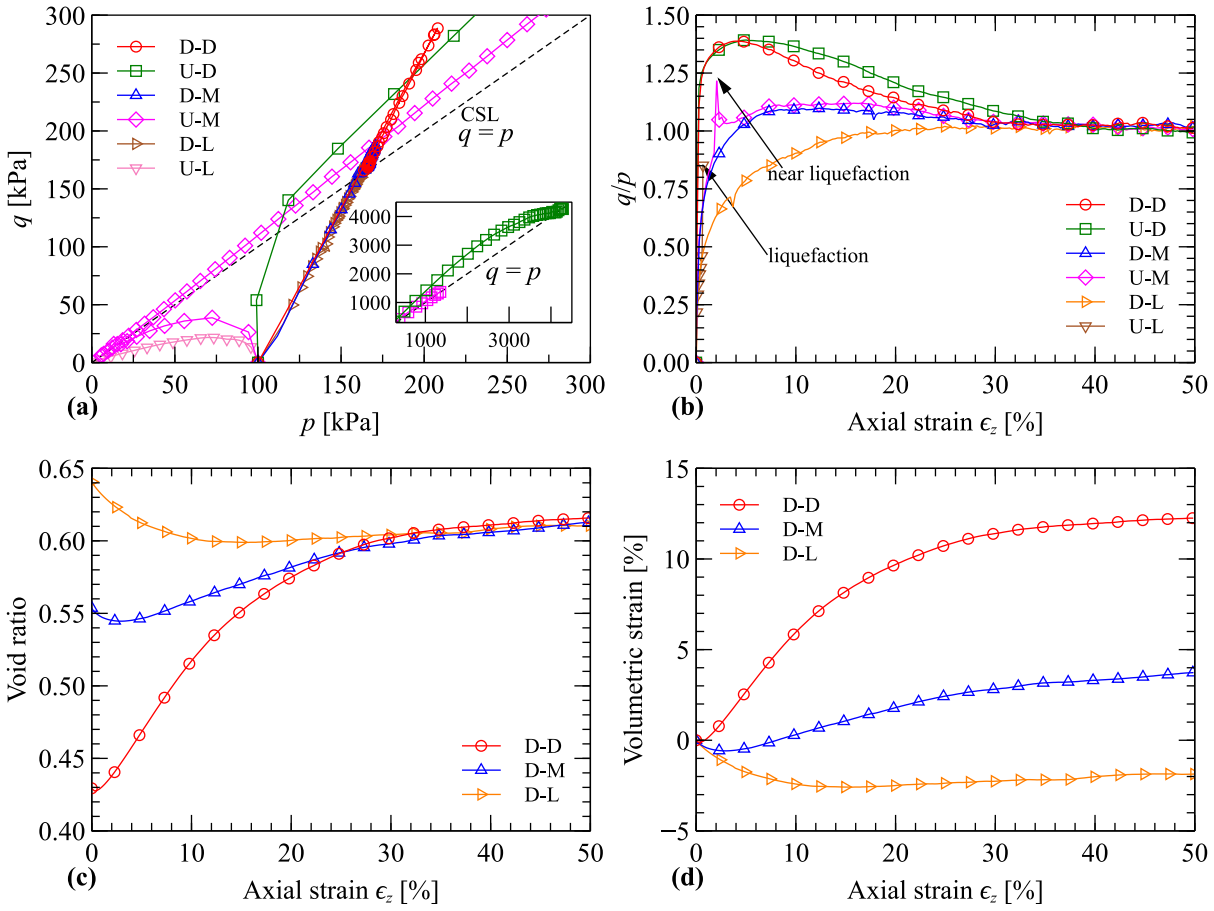

Fig. 23. Loading paths (a) and deviatoric stress ratio q/p (b) for the dense, medium-dense and loose specimens during drained and undrained triaxial compression; variation of void ratio (c) and volumetric strain (d) for the dense, medium-dense and loose specimens during drained triaxial compression. CSL: critical state line. *Source:* From [33].

10^3 N/m, respectively. The channel has the same stiffness as particles for all boundary walls. The coefficient of friction is 1.0 for the base and zero for the side walls.

Fig. 20(b) shows a heap of 8000 oblate superellipsoidal particles at rest, where an approximately straight slope can be observed as expected, proving the validity of the numerical implementation. It is worth noting that the ratio of kinetic energy E_k to potential energy E_p can be tracked for checking whether particles come to rest. For example, the heap slope stays almost stationary for the energy ratio E_k/E_p below 10^{-7} so that it can be considered reaching a relative equilibrium state. More details on the analysis of particle shape effect on a heap are reported in [42] by using *SudoDEM*.

7.3. Granular packing

As a demonstration, we simulate granular packing with poly-superellipsoidal particles falling freely into a cubic box (1 m × 1 m × 4 m) under gravity, which has already been done to test the robustness of *SudoDEM* in our previous work [24]. In the simulation, various particle shapes are generated with shape parameters randomly selected in prescribed intervals, i.e., $\epsilon_1, \epsilon_2 \in [0.4, 1.6]$, $l_x = 0.1$ m, $l_y, l_z \in [0.02, 0.1]$ m, $e_x, e_y, e_z \in [0.2, 0.8]$. The material properties and other numerical parameters follow the literature [24]: the particle mass density is set to 2650 kg/m^3; the normal and tangential contact stiffnesses are set to 1×10^5 and 7×10^4 N/m, respectively, and the coefficient of friction is set to 0.1. At the initial configuration, 500 poly-superellipsoids with random orientations are positioned at a 5-by-5-by-20 grid in a lattice manner. During the course of deposition under gravity, the free particles impact the top of a packing sequentially, and particles may collide with their neighbors (particles or container walls) and bounce back and forth. Fig. 21 shows the variation of the energy of particles in conjunction with several snapshots during packing, where the kinetic energy is $\sum(m_i v_i^2/2 + \boldsymbol{I}_i \boldsymbol{\omega}_i^2/2)$, and the total mechanical energy includes the kinetic energy and the gravitational potential energy defined as $\sum(m_i g h_i)$ (h is the height of particle mass center from the box bottom).

7.4. Triaxial compression

Three numerical specimens composed of 10 000 poly-superellipsoidal particles with random orientations and positions are prepared within a cubic container. The parameters of particle shapes are randomly selected in the prescribed intervals, i.e., $\epsilon_1, \epsilon_2 \in [0.5, 1.4]$, $l_x, l_y, l_z \in [0.25, 0.75]$ mm, $e_x, e_y, e_z \in [0.2, 0.8]$. The material properties follow the literature [33]: the contact stiffness is empirically set as $k_n = k_s = r \times 100$ MPa (where r is the average particle size); the coefficient of friction μ is set to 0.3. The specimens have initial void ratios of 0.429, 0.554 and 0.641 after consolidation with a confining stress σ_0 of 100 kPa, corresponding to three different states, i.e., dense (D), medium-dense (M) and loose (L), respectively. After that, the top and bottom plates move towards at a constant axial strain rate $\dot{\epsilon}_z$ to trigger a continuous shear process. All specimens are subjected to two typical loading conditions (commonly encountered in soil mechanics) during the course of shearing: drained and undrained. Each test is labeled as A-B: A is for the loading conditions,

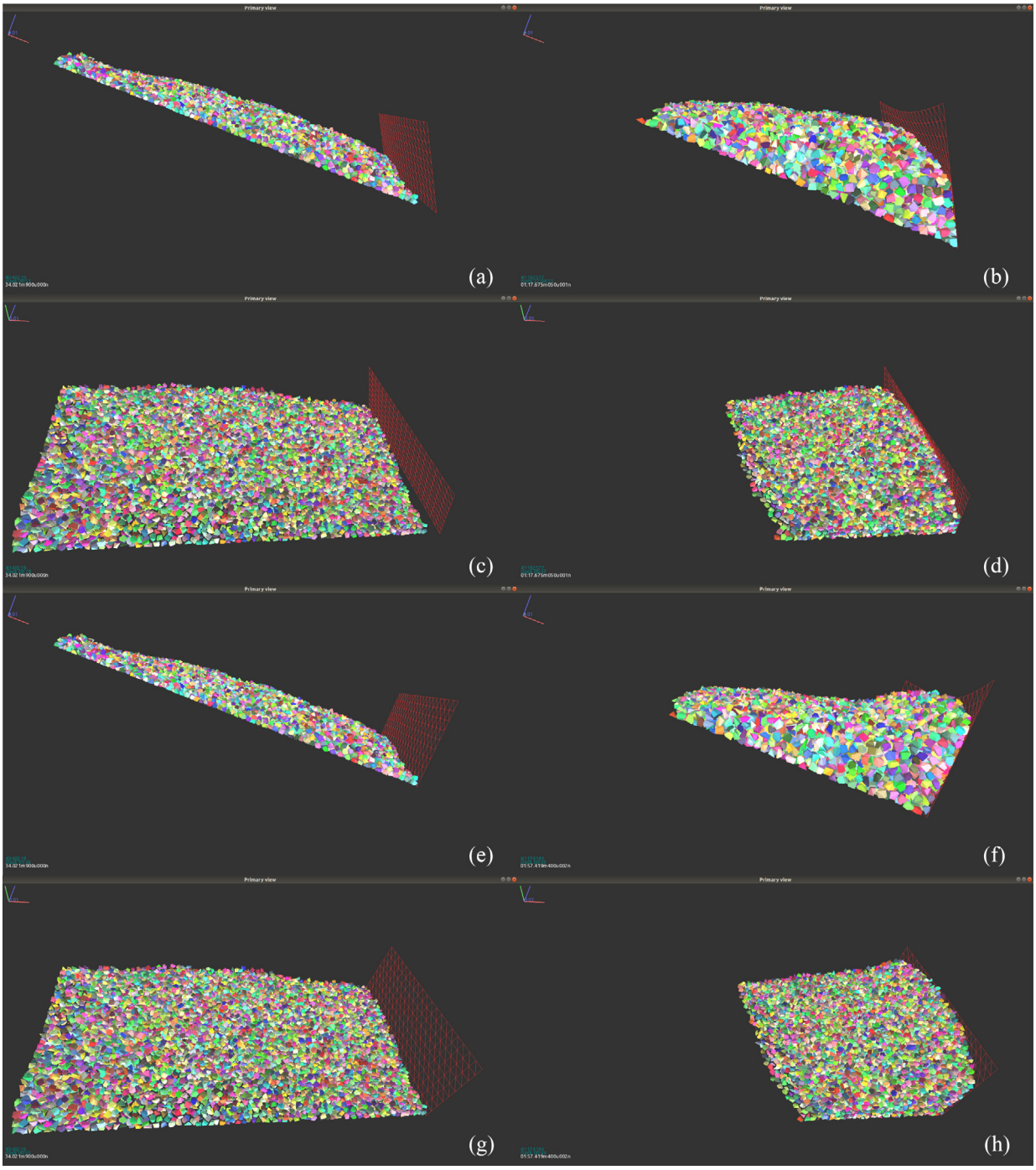

Fig. 24. Different views of landslides impacting flexible barriers: initial states (*left*) and final states (*right*) for the flexible barrier perpendicular to the horizon (a–d) or the channel bottom (e–h). Note: the channel walls are not visualized.

drained (D) or undrained (U); B is for the initial state, dense (D), medium-dense (M) or loose (L). For the drained case, the four side plates remain a constant confining stress σ_0 during shearing; for the undrained case, a specimen remains a constant volume by adjusting the positions of the four side plates according to $\dot{\epsilon_x} = \dot{\epsilon_y} = -\dot{\epsilon_z}/2$. It is worth noting that particles in the simulations are dry, and the constant volume condition is imposed to mimic the undrained case in the laboratory [3]. To ensure quasi-static shear, the axial strain rate $\dot{\epsilon_z}$ is set to a small value of 0.01/s to fulfill the criterion that the inertia number $I = \dot{\epsilon}_z \langle d \rangle \sqrt{\rho/\sigma_0} \leq 10^{-3}$ [27], where $\langle d \rangle$ is the average particle diameter, and ρ is the material mass density. More details on the numerical shear can be found in [27]. Specimens subjected to the undrained loading except for the loose ones are compressed to a sufficiently large level of axial strain $\epsilon_z = 50\%$ (where $\epsilon_z = \ln(H_0/H)$, H_0 and H are the specimen height at the initial and sheared states respectively), at which the specimens reach the steady flow regime (or critical state in soil mechanics). Fig. 22 shows snapshots and normal contact force chains of the dense specimen at the initial and final states during the drained triaxial compression. In the presented demo, it takes about 24 h for each simulation with a single CPU-thread.

The granular stress tensor can be defined in a volume-averaged form as [43]

$$\sigma_{ij} = \frac{1}{V}\sum_{c\in V} f_i^c l_j^c \tag{38}$$

where V is the total volume of the assembly; f^c is the contact force at the contact c, and l^c is the branch vector joining the centers of the two contacting particles at contact c. The mean stress p and the deviatoric stress q are defined as:

$$p = \frac{1}{3}\sigma_{ii}, \quad q = \sqrt{\frac{3}{2}\sigma'_{ij}\sigma'_{ij}} \tag{39}$$

where σ'_{ij} is the deviatoric part of stress tensor σ_{ij}. The volumetric strain ϵ_v is given by $\epsilon_v = \ln(V/V_0)$, where V_0 and V are the specimen volume at the initial and sheared states, respectively, and positive values represent dilation. Note that the volume-averaged stress is identical to the stress directly computed from the boundary walls for quasi-static simulations. Hence, it is reasonable to use the volume-average stress and the strain directly computed from the boundary to show the stress–strain relation of a granular material during triaxial shearing.

Fig. 23 shows the macroscopic mechanical responses of all specimens during drained and undrained triaxial compression. All specimens subjected to different loading paths reach a unique critical state line (CSL), i.e., maintaining a stationary q/p with loading. For the drained tests (D–D, D–M and D–L), the dense specimen D–D behaves in a strain-softening manner with volumetric dilation, whilst the loose one D–L has a strain-hardening trend with volumetric contraction. Moreover, the volume or void ratio also reaches a unique value at the critical state. For the undrained tests (U–D, U–M and U–L), it can be seen that both p and q are likely to approach zero for loose specimens, especially U-L, i.e., reaching the so-called liquefaction state in soil mechanics. The specimen can no longer undertake any external loading after reaching the liquefaction state. Overall, the simulated results are qualitatively in agreement with the well-known experimental observation on sand in soil mechanics, proving the validity and robustness of numerical implementation in *SudoDEM*.

7.5. Landslides impacting a flexible barrier

In this demo, we simulate a more engineering system where a landslide of rocks impacts a flexible resisting barrier, where the rocks are simulated by DEM while the barrier is constructed by FEs (finite elements). A granular packing is prepared in a cubic container following the same protocol as depicted in Section 7.3, which is composed of 10 000 polyhedral particles. Each polyhedral particle is a convex hull of eight vertexes randomly selected from the surface of a sphere with a radius of 0.15 m. We note here that an advanced procedure of particle shape generation may be deserved for an elegant simulation, which is however not the focus of this demo. The dimension of the container bottom is 6 m $\times$ 3 m, while the thickness of the granular packing (hereafter referred to as the rock layer) is approximately 0.6 m after reaching an equilibrium state. To trigger a slide, the rock layer is further laid on the bottom of an inclined channel with two side walls and one bottom wall confined, referring to the left column in Fig. 24. All walls of the channel are assumed frictionless, and the inter-particle coefficient of friction is set to 0.3. A linear spring contact model is applied, where both normal and tangential contact stiffnesses are set to 1×10^5 N/m for all particles and walls. The Young's modulus and Poisson's ratio of the flexible barrier are set to 1 MPa and 0.3, respectively.

As a demonstration, only one case of angle of incline (i.e., 30°) is considered, while two different installations of the barrier are taken into account: perpendicular to the horizon or the channel bottom. Besides, the top-boundary nodes (excluding those situated at the two sides) of the barrier are free, while the other nodes at the boundary are fixed in displacement. The rock layer slides and impacts the barrier under gravity. For the computing time, it takes about 12 h for each with a single CPU-thread. The final states of the system at an equilibrium state are shown in the right column of Fig. 24. It can be seen that the granular surface is not straight but even subtly rounded, which is in agreement with the experimental observations, e.g., in [44]. The reason why the surface is not straight as observed in the heap formation (e.g., with particles flowing out a hopper) is associated to the sliding-induced impact against the barrier. A more detailed analysis of the response of the system is beyond the scope of this paper.

8. Summary

In this paper, we presented an open-source code *SudoDEM* that is designed explicitly for modeling of non-spherical particles for both 2D and 3D in DEM. In addition to the built-in prime shapes including poly-superellipsoid, superellipsoid, cylinder, cone, polyhedron for 3D, and disk, superellipse for 2D, it is straightforward to customize much more shapes based on the generic interface of the robust contact detection algorithms, i.e., PCN, GJK, and the hybrid PCN–GJK. Benefiting from the hybrid programming of Python and C++, *SudoDEM* can be flexibly coupled with other codes for further considering multi-phase or hierarchical modeling by coupling FEMxDEM or MPMxDEM [45]. The source code is hosted in public domain at the GitHub repository (https://github.com/SudoDEM) under the GNU public license (GPL v3 or later). More details on the project are available at the home page (https://sudodem.github.io). Interested researchers are welcome to use and/or make a contribution to *SudoDEM*.

Declaration of competing interest

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

Acknowledgments

This work was financially supported by the Hong Kong Scholars Program (2018), the National Natural Science Foundation of China (by Project Nos. 51909095, 51679207 and 11972030), Guangdong Basic and Applied Basic Research Foundation (2020A1515011525), the Fundamental Research Funds for Central Universities, China (D2192710), Research Grants Council of Hong Kong (by GRF Project No. 16207319, TBRS Project No. T22-603/15N and CRF Project No. C6012-15G). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the financial bodies.

References

[1] P.A. Cundall, O.D. Strack, Géotechnique 29 (1) (1979) 47–65.
[2] C. O'Sullivan, Particulate Discrete Element Modelling: A Geomechanics Perspective, Taylor & Francis, 2011.
[3] N. Guo, J. Zhao, Comput. Geotech. 47 (2013) 1–15.
[4] J. Zhao, N. Guo, Acta Mech. Solida Sin. 27 (1) (2014) 1–14.
[5] H. Shin, J. Santamarina, J. Geotech. Geoenviron. Eng. 139 (2) (2012) 353–355.
[6] M. Payan, A. Khoshghalb, K. Senetakis, N. Khalili, Comput. Geotech. 72 (2016) 28–41.

[7] A.G. Athanassiadis, M.Z. Miskin, P. Kaplan, N. Rodenberg, S.H. Lee, J. Merritt, E. Brown, J. Amend, H. Lipson, H.M. Jaeger, Soft Matter 10 (1) (2013) 48–59.
[8] M. Jiang, Z. Shen, J. Wang, Comput. Geotech. 65 (2015) 147–163.
[9] C.-x. Miao, J.-j. Zheng, R.-j. Zhang, L. Cui, Comput. Geotech. 81 (2017) 249–261.
[10] D. Höhner, S. Wirtz, H. Kruggel-Emden, V. Scherer, Powder Technol. 208 (3) (2011).
[11] S. Zhao, T.M. Evans, X. Zhou, in: X. Li, Y. Feng, G. Mustoe (Eds.), Proceedings of the 7th International Conference on Discrete Element Methods, in: Springer Proceedings in Physics, no. 188, Springer Singapore, 2016, http://dx.doi.org/10.1007/978-981-10-1926-5_11.
[12] T.-T. Ng, Int. J. Numer. Anal. Methods Geomech. 33 (4) (2009) 511–527.
[13] Z.-Y. Zhou, R.-P. Zou, D. Pinson, A.-B. Yu, Ind. Eng. Chem. Res. 50 (16) (2011) 9787–9798.
[14] C. Wellmann, C. Lillie, P. Wriggers, Eng. Comput. 25 (5) (2008) 432–442.
[15] S. Zhao, N. Zhang, X. Zhou, L. Zhang, Powder Technol. 310 (2017) 175–186.
[16] C. Boon, G. Houlsby, S. Utili, Comput. Geotech. 44 (2012) 73–82.
[17] S. Zhao, X. Zhou, W. Liu, Granul. Matter 17 (6) (2015) 793–806.
[18] K.-W. Lim, J.E. Andrade, Int. J. Numer. Anal. Methods Geomech. 38 (2) (2014) 167–188.
[19] E. G. Nezami, Y. MA Hashash, D. Zhao, J. Ghaboussi, Int. J. Numer. Anal. Methods Geomech. 30 (8) (2006) 783–801.
[20] A. Wachs, L. Girolami, G. Vinay, G. Ferrer, Powder Technol. 224 (2012) 374–389.
[21] J. Eliáš, Powder Technol. 264 (2014) 458–465.
[22] F. Zheng, Y.-Y. Jiao, M. Gardner, N. Sitar, Comput. Geotech. 87 (2017) 76–85.
[23] R. Kawamoto, E. Andò, G. Viggiani, J.E. Andrade, J. Mech. Phys. Solids 91 (2016) 1–13.
[24] S. Zhao, J. Zhao, Int. J. Numer. Anal. Methods Geomech. 43 (13) (2019) 2147–2169.
[25] J. Kozicki, F. Donzé, Eng. Comput. 26 (7) (2009) 786–805.
[26] V. Šmilauer, E. Catalano, B. Chareyre, S. Dorofeenko, J. Duriez, A. Gladky, J. Kozicki, C. Modenese, L. Scholtès, L. Sibille, et al., 2010, http://yade-dem.org/doc/.
[27] S. Zhao, T. Evans, X. Zhou, Géotechnique 68 (12) (2018) 1085–1098.
[28] J. Ai, J.-F. Chen, J.M. Rotter, J.Y. Ooi, Powder Technol. 206 (3) (2011) 269–282.
[29] S. Zhao, T.M. Evans, X. Zhou, Int. J. Solids Struct. 150 (2018) 268–281.
[30] G.v.d. Bergen, J. Graph. Tools 2 (4) (1997) 1–13.
[31] S. Zhao, T.M. Evans, X. Zhou, Powder Technol. 323 (2018) 323–336.
[32] S. Zhao, X. Zhou, Granul. Matter 19 (2) (2017) 38.
[33] S. Zhao, J. Zhao, N. Guo, Phys. Rev. E 101 (1) (2020) 012906.
[34] K. Johnson, Contact Mechanics, Cambridge University Press, London, 1985.
[35] J.C. Lagarias, J.A. Reeds, M.H. Wright, P.E. Wright, SIAM J. Optim. 9 (1) (1998) 112–147.
[36] M.I. Lourakis, Found. Res. Technol. 4 (1) (2005) 1–6.
[37] E.G. Gilbert, D.W. Johnson, S.S. Keerthi, IEEE J. Robot. Autom. 4 (2) (1988) 193–203.
[38] G. Van Den Bergen, Collision Detection in Interactive 3D Environments, CRC Press, 2003.
[39] S. Ji, S. Sun, Y. Yan, Procedia Eng. 102 (2015) 1793–1802.
[40] S. Galindo-Torres, D. Pedroso, Phys. Rev. E 81 (6) (2010) 061303.
[41] S. Zhao, J. Zhao, 2019, [Online; accessed March, 2020]. URL https://sudodem.github.io/download.html.
[42] H. Chen, S. Zhao, X. Zhou, Particuology 50 (2020) 53–66.
[43] J. Christoffersen, M. Mehrabadi, S. Nemat-Nasser, J. Appl. Mech. 48 (2) (1981) 339–344.
[44] Y.-J. Jiang, Y. Zhao, I. Towhata, D.-X. Liu, Powder Technol. 270 (2015) 53–67.
[45] S. Zhao, J. Zhao, Y. Lai, Comput. Methods Appl. Mech. Engrg. 367 (2020) 113100.