Computer Physics Communications 320 (2026) 109997

Contents lists available at ScienceDirect

Computer Physics Communications

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

Computer Programs in Physics

Advanced strategies for discrete simulations with three-dimensional *R*-shapes in rockable framework

Vincent Richefeu [a,*], Gaël Combe [a], Lhassan Amarsid [b], Raphaël Prat [b], Jean-Mathieu Vanson [b], Saied Nezamabadi [c,d], Patrick Mutabaruka [e], Jean-Yves Delenne [d], Farhang Radjaï [c]

[a] CNRS, Grenoble INP, Univ. Grenoble Alpes, 3SR 38000, Grenoble, France
[b] DES, IRESNE, CEA, F-13108, Saint Paul-Lez-Durance, Cadarache, France
[c] LMGC, CNRS, University of Montpellier, 34090, Montpellier, France
[d] IATE, INRAE, Institut Agro, University of Montpellier, 34000, Montpellier, France
[e] DYNECO/DHYSED, IFREMER 2 Plouzané, 29280, France

ARTICLE INFO

J Ballantyne

Keywords:
Discrete element method
Complex shapes
Contact detection
Computing performance

ABSTRACT

The Discrete Element Method (DEM) is widely used to simulate the mechanical behavior of granular materials across a broad range of applications and industrial domains. Particle shape is a key feature playing a crucial role for physics-fidelity of DEM simulations. However, accurately representing complex particle shapes within DEM frameworks presents significant challenges such as defining unambiguous contact normals or managing geometric singularities. Rigid particles are often modeled as convex polyhedra, which inherently suffer from ill-defined outward normal vectors at sharp edges and vertices. To represent non-convex geometries, these polyhedra must typically be combined, further increasing the computational and geometric complexity. In this work, we adopt an efficient and robust strategy to overcome these limitations by using *R*-shapes, defined as rounded-edge shapes, also known as sphero-polyhedra, obtained by sweeping a sphere of radius *R* along the edges and faces of a base polyhedral shape. This construction results in smooth surface transitions and circumvents common issues associated with traditional polygonal representations. This paper provides a detailed presentation of the implementation, structure, and advantages of *R*-shapes in DEM simulations. The proposed solutions are implemented in a fully open-source software package called `Rockable`, developed in C++, which integrates state-of-the-art numerical techniques and shared-memory parallelization for enhanced performance. Beyond the geometric modeling aspects, we also address several methodological challenges, including the treatment of contact elasticity and the numerical integration scheme. The combined contributions of this work offer a practical and efficient framework for simulating complex particle shapes in DEM with high physics fidelity and computational efficiency.

Program summary

Program Title: `Rockable`
CPC Library link to program files: (to be added by Technical Editor)
Developer's repository link:
https://github.com/richefeu/rockable
Licensing provisions: CeCILL-B
Programming language: C++11
Supplementary material:
Nature of problem(approx. 50–250 words):
The open-source software `Rockable` addresses key challenges in simulating the mechanical behavior of granular materials using the Discrete Element Method (DEM), widely applied in both industrial applications and academic studies particularly where particle shape plays a critical role. Accurate modeling of the diversity of particle shapes in DEM remains non-trivial, due in part to ambiguities in defining contact normals. Rigid particles are often represented as convex polyhedra, which suffer from poorly defined outward normals at edges and vertices. Additionally, modeling non-convex shapes typically requires aggregating multiple convex elements,

* Corresponding author.
E-mail address: vincent.richefeu@univ-grenoble-alpes.fr (V. Richefeu).

https://doi.org/10.1016/j.cpc.2025.109997
Received 27 June 2025; Received in revised form 15 October 2025; Accepted 11 December 2025
Available online 17 December 2025

increasing complexity. Rockable resolves these issues through the use of *R*-shapes, or sphero-polyhedra, obtained by sweeping a sphere of radius *R* along the edges and faces of a polyhedral base. Developed in C++, Rockable integrates state-of-the-art numerical techniques and supports shared-memory parallelism for high-performance simulations. It also addresses key physical modeling challenges, including contact elasticity and particle breakage.

Solution method(approx. 50–250 words):

Rockable implements a geometry-based contact detection algorithm for *R*-shapes, relying on the classification of contacts into four elementary types: vertex-vertex, vertex-edge, vertex-face, and edge-edge. This systematic approach provides a unified framework for computing contact points and their associated local frames. A key advantage of this method is its ability to reduce complex contact scenarios, such as face-face interactions, into combinations of these fundamental cases. This modular treatment not only improves robustness and computational efficiency but also extends naturally to non-convex and hollowed shapes. The framework is designed to be extensible, making Rockable a versatile tool for DEM-based modeling of irregular and evolving particle geometries.

1. Introduction

Since the emergence of discrete element simulations in the 1980s, numerous improvements have been made to the Discrete Element Method (DEM). In fact, there is a wide variety of discrete approaches [*e.g.,* 1–7], each with its own pros and cons in different fields of interest. However, the most widely used approach is the one introduced by Cundall [8]. Since the advent of DEM, significant efforts have been made to model force laws between circular or spherical particles. Although Cundall introduced polygonal shapes [9], the focus has only recently shifted to the shape of particles [10].

The widespread adoption of DEM for modeling granular materials highlights its versatility in various fields of application. However, as the geometrical and physical intricacies of the problems increase, one crucial factor that often becomes challenging is the accurate representation of the shapes of the particles. In many scenarios, the difficulties associated with handling the geometric attributes of particles within a DEM-based code become apparent. Certain features, such as non-convexity or the unambiguous definition of contact normals, can complicate the simulation process. As a result, addressing these challenges becomes imperative to achieve more accurate and realistic simulations of granular materials in various applications.

As we begin this exploration, let us first envision the world of *R*-shapes in action. The illustrations in Fig. 1 capture rotating drums containing *R*-polyhedra at increasing rotation speed. These simulations provide an example of the well-known flow regimes investigated many authors such as in Orozco [11], Orozco et al. [12,13]. Fig. 2(left) shows a clear break from conventional particle shapes, with the simulation now featuring hexapods [14,15]. These hexapods, with their elaborate design and numerous protrusions, present a significant challenge to particle-based modeling. Their presence within the rotating drums introduces a greater level of sophistication. The intricate interactions of these hexapodal elements within the drum demonstrate the adaptability of Rockable, highlighting its flexibility in addressing emerging challenges in DEM.

Expanding the scope of our exploration into the capabilities of this framework, we encounter in Fig. 2(right) a novel and compelling scenario – a mixing device equipped with a 3D-modeled tool. This tool, a single non-convex polyhedron, is presented as a STL file, pushing the boundaries of particle-based modeling even further. This figure not only showcases the flexibility of Rockable but also highlights its adaptability to incorporate intricate external geometries. The multi-level interactions between the non-convex tool and the particle system inside the mixing device reveal another potential of this framework to bridge the gap between sophisticated physical phenomena and numerical simulations in industrial applications. In this visual narrative, we observe the fusion of precision engineering and computational achievement.

As a final illustrative example, we turn to the field of rockfall simulations. Fig. 3 captures a scenario where both the terrain and the falling blocks defy the world of non-convex polyhedral shapes. In this realistic depiction, the interplay between rugged terrain and non-convex blocks reflects the challenges encountered in real-world situations. This simulation highlights the versatility of Rockable, which seamlessly manages the collision and interaction of irregular shapes [*e.g.,* 17–24].

The primary objective of this paper is to provide a comprehensive and detailed explanation, including practical aspects, of the technical and geometric strategies used in implementing *R*-shapes within the DEM. Typically, such technical details are not extensively covered in studies that focus on specific physical phenomena. Consequently, this paper concentrates on elucidating the computational aspects. The emphasis lies on thoroughly addressing the technical difficulties associated with the implementation of *R*-shapes in DEM simulations.

2. Simulation process

The software Rockable enables the modeling of discrete elements with irregular, concave, and possibly perforated geometries. Unlike traditional discrete element methods that typically use spherical or regular-shaped particles, the proposed approach allows for the inclusion of particles with these more intricate shapes. This enhancement significantly increases the realism of the simulations, as it accounts for the true shapes and interactions of particles that are often encountered in real-world granular materials.

In addition to these intricate geometries, a key feature of the software is its ability to model the assembly of particles that are glued together in such a manner that there are no voids between them. This cohesive assembly allows the formation of aggregate structures, where individual particles act as elementary entities. These structures are capable of being fractured or “cut” under loading, which enables the simulation of material failure or rupture. To simulate the failure process, the software utilizes predefined rupture surfaces, which indicate potential failure zones within the particle. The rupture criterion relies on stress thresholds and/or energy release rates. This process replicates the fracture behavior of real materials under stress.

Based on the material behavior and failure mechanisms, three situations can be distinguished as illustrated in Fig. 4:

(a) **Pre-cracked continuum**. In this situation, the material is continuous but contains predefined surfaces or zones that can potentially open or slide under stress, leading to cracks and fractures. These predefined surfaces act as weak points within the material, simulating the initiation and propagation of fractures. This model is useful for studying how materials with inherent flaws or pre-existing cracks behave under load, such as in geological formations or composite materials.

(b) **Cemented granular assembly**. Here, the material consists of rigid particles that are bonded together with solid cement-like bonds. The entire sample can fracture into smaller, granular pieces (like sand)

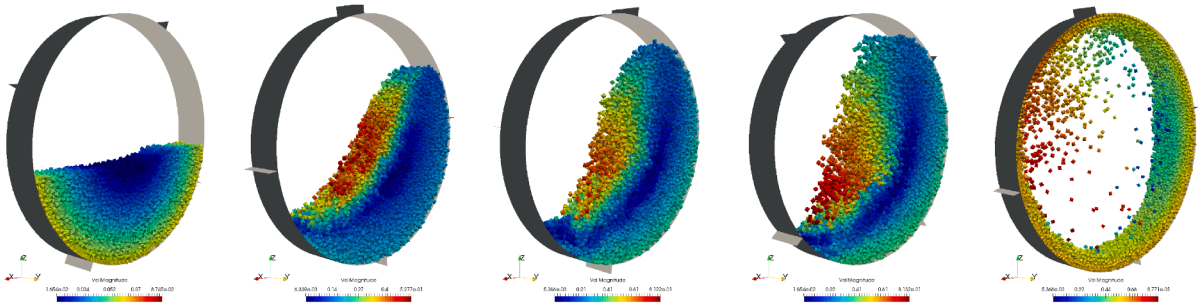

Fig. 1. Simulation of granular flows composed of *R*-polyhedra in a rotating drum by means of the `Rockable` code. The visual representation offers insight into different dynamic regimes: (from left to right) surging, rolling, cascading, cataracting and centrifuging.

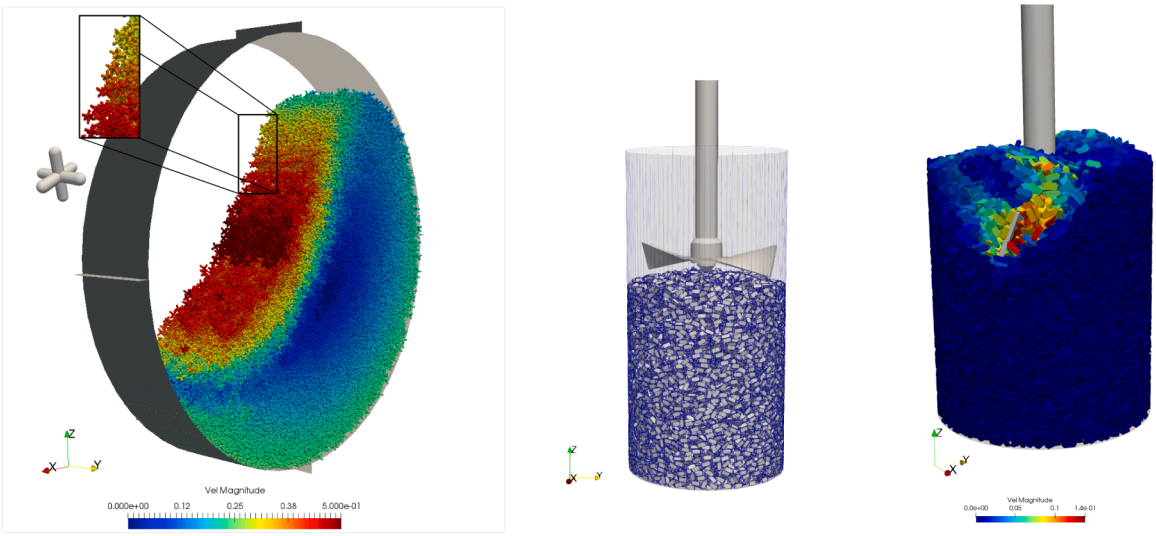

Fig. 2. Complex shapes for 3D-modeled mixing tool. (left) Intricate hexapodal particles navigate the dynamic environment within rotating drums, emphasizing the ability of `Rockable` to handle interactions between complex particle geometries [16]. (right) Example of a simulation using the so-called FT4 powder rheometer. The blade is represented as a single non-convex polyhedron, seamlessly integrates into the mixing device, demonstrating the framework adaptability to incorporate intricate external geometries.

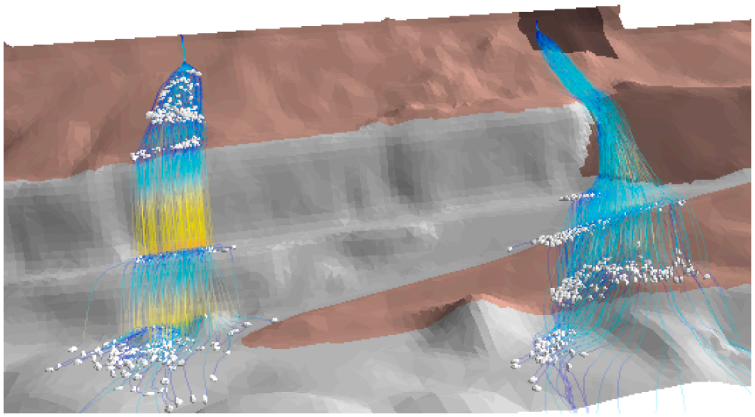

Fig. 3. Non-convex terrain and block interactions in rockfall simulations. Solid lines represent the trajectories of the blocks with their velocities in color map.

when the bonds are broken. This model emphasizes the cohesive strength provided by the bonds between particles. This approach is suitable for simulating materials like concrete or sedimentary rocks, where the failure process involves the breakdown of bonds between particles.

(c) **Hybrid fracture model**. This situation combines elements of both the pre-cracked continuum and the cemented granular assembly. It features a material with both predefined crack surfaces and bonded particles, allowing for a more complex and realistic simulation of failure processes. This hybrid model can be applied to materials with mixed failure modes, such as certain types of composites or geological materials that exhibit both brittle fracture and granular disintegration.

3. Definition of *R*-shapes

When the shape of the discrete elements is of primary importance – just as in rockfall models, for instance – it must be taken into account

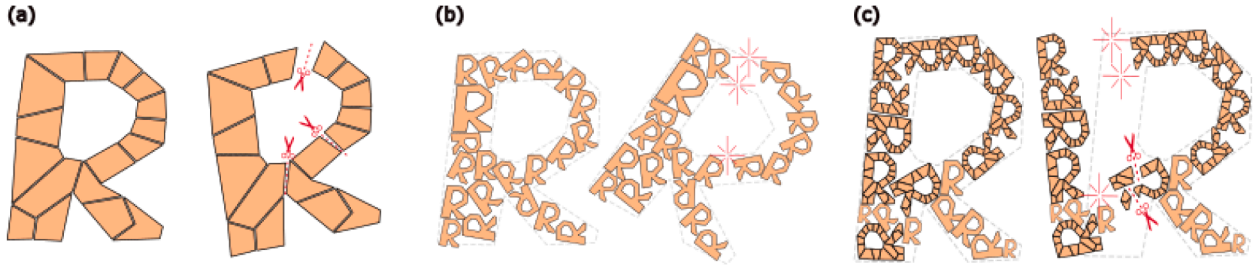

Fig. 4. A 2D diagram depicting various failure mechanisms applicable in the software `Rockable`: (a) pre-cracked continuum, (b) cemented granular assembly, (c) hybrid fracture model. Each rigid element is represented as an *R*-shape, which may feature concavities and voids, similar to shape of the letter R.

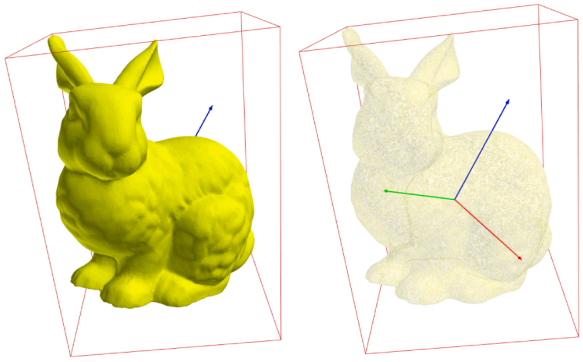

Fig. 5. Modelling Standfordt Bunny as a sphero-polyhedron. The rigid body consists of a clustering of three shapes: spheres for the vertices, cylinders for the edges and 3D thick-polygons for the facets. The surrounding box is the smallest Oriented Bounding Box (OBB) that best fits the shape of the rabbit. The visible frame on the transparent representation is the eigen-frame of the shape, where the inertia matrix is diagonal. It should be noted that the axes and the center of the OBB do not necessarily match the eigen-axes of inertia and the center of mass, respectively.

explicitly in the model. Different strategies are possibles (*e.g.*, polyhedra, clumps) but one can choose sphero-polyhedra which has several advantages including a highly simplified contact detection. This ability, which is rarely taken into account in other codes, is provided by the software `Rockable`. In practice, the principle of *R*-shapes is similar to that of clumps, where spheres are assembled to form a rigid body with arbitrary shape. But, in addition to spheres, two supplementary elementary shapes are considered: cylinders to form the edges and polygonal plans to form the faces; the vertices are formed by spheres.

Fig. 5 gives an example of *R*-shape that represents the Standfordt Bunny that has the particularity of being non-convex. The shape is defined by a set of vertices interconnected by edges and faces. The vertex positions of a body i are given by reference to the center of mass C_i in the body frame $\mathcal{R}_i$ defined by the main directions of the inertial tensor (see Section 4). The shape of the body is then defined by sweeping a sphere of radius R along each point of its edges and faces. For this reason, the term "*R*-shape", that is more concise than sphero-polyhedron, is preferred. The term is also more precise as it allows for an explicit definition of the shape, such as *R*-cube, *R*-triangle, *R*-line, and even *R*-point, which simply refers to a sphere. From a mathematical viewpoint, the resulting shape can be seen as the Minkowsky sum of a polyhedron with a sphere. In simple terms, the Minkowski sum involves combining two shapes by adding the points of one shape to the points of another.

4. Mass/inertia properties

Inertia properties are prerequisites for the calculation of the dynamics of each particle. These are the centers of mass, the masses, the principal directions of inertia, and the corresponding moments of inertia

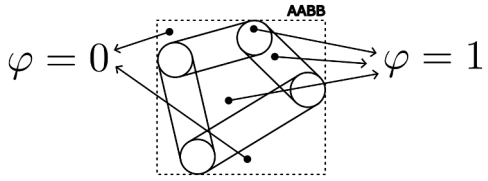

Fig. 6. Schematic illustration in two dimensions of the function φ as defined in Eq. (2).

(eigenvalues). In order to calculate these properties for shapes that can have any geometry (concave, convex, holes), a numerical integration technique has been chosen. This technique is the Monte Carlo (MC) method (see Numerical Recipes in C++, [25]), which is based on the approximation of the integral of a function f on a volume V:

$$\int_V f \mathrm{d}V \simeq V\langle f\rangle \pm \sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N}} \tag{1}$$

To improve accuracy and compution speed, the MC integrations are based on Sobol sequences to pick points in a semi-random pattern rather than on a regular grid. The integration procedure is performed by first setting an axis-aligned bounding box (AABB) that surrounds and fits the shape. A position within this AABB is randomly generated by means of a Sobol sequence. To assess whether this point is inside the polyhedron, we use an algorithm based on the oddness of the number of intersections between a semi-infinite ray (starting from the point) and each face. A function $\varphi_{\text{polyh}}(\boldsymbol{x})$ indicates, by returning 0 or 1, whether the point at coordinate $\boldsymbol{x}$ is inside a polyhedron or not. The radius of the sphere used in the Minkowski sum is also taken into account by checking the proximity of the point to the body boundary, namely the nodes, edges, and faces.

$$\varphi(\boldsymbol{x}) = \left| \sum_{\text{vertexes}} \varphi_{\text{sph}}(\boldsymbol{x}) + \sum_{\text{edges}} \varphi_{\text{cyl}}(\boldsymbol{x}) + \sum_{\text{faces}} \varphi_{\text{polyg}}(\boldsymbol{x}) + \varphi_{\text{polyh}}(\boldsymbol{x}) \right|^{0/1} \tag{2}$$

where the symbol $|\cdots|^{0/1}$ turns any nonzero value into 1, and all $\varphi_{\bullet}$ are binary indicator functions associated to the corresponding elementary shape (sphere, cylinder, thick polygon or polyhedron).

Although Eq. (2) involves a somewhat unconventional operator, the function $\varphi(\boldsymbol{x})$ is readily computable and provides a clear indication of material presence at position $\boldsymbol{x}$, as depicted in Fig. 6. Accordingly, the summations in this equation should be interpreted as logical 'OR' operations, reflecting the presence of material at any of the considered positions. It is also possible to treat the shape as a surface or an empty (hollow) structure; in such cases, the term φ_{polyh} is not included in Eq. (2).

In the precalculation procedure for the mass properties, the volume center and mass center of the shape are first estimated. These properties are required to perform a second integration set for the estimate of the matrix of inertia (divided by the body mass). The eigenvectors and eigenvalues are extracted from this matrix to define the shape frame.

The node positions are then expressed if this body frame and the position and orientation of the body are set. Then, it becomes possible to numerically integrate any quantity in volume V^{AABB} using the scheme shown in Eq. (1). Thus, the volume of a block will be measured using the following expression:

$$V = \int_{V^{\text{AABB}}} \varphi(\boldsymbol{x}) \mathrm{d}V \simeq \langle \varphi(\boldsymbol{x}) \rangle V^{\text{AABB}}. \tag{3}$$

By assuming a uniformly distributed volume density ρ, the mass of a shape is $m = \rho V$, and its inertial center $\boldsymbol{x}_G$ can also be obtained by means of MC integration:

$$\begin{aligned} \boldsymbol{x}_G &= \frac{1}{V} \int_{V^{\text{AABB}}} \boldsymbol{x}\, \varphi(\boldsymbol{x}) \mathrm{d}V \\ &\simeq \frac{\langle \boldsymbol{x}\, \varphi(\boldsymbol{x}) \rangle V^{\text{AABB}}}{V} = \frac{\langle \boldsymbol{x}\, \varphi(\boldsymbol{x}) \rangle}{\langle \varphi(\boldsymbol{x}) \rangle} \end{aligned} \tag{4}$$

The symmetric matrix of inertia relative to the point $\boldsymbol{x}_G$ is in turn computed using MC integrations for each of the 6 components:

$$\begin{aligned} I_{xx}(\boldsymbol{x}_G) &= \gamma \langle \varphi(\boldsymbol{x}) (\delta y^2 + \delta z^2) \rangle \\ I_{yy}(\boldsymbol{x}_G) &= \gamma \langle \varphi(\boldsymbol{x}) (\delta x^2 + \delta z^2) \rangle \\ I_{zz}(\boldsymbol{x}_G) &= \gamma \langle \varphi(\boldsymbol{x}) (\delta x^2 + \delta y^2) \rangle \\ I_{xy}(\boldsymbol{x}_G) &= -\gamma \langle \varphi(\boldsymbol{x}) \delta x \delta y \rangle \\ I_{xz}(\boldsymbol{x}_G) &= -\gamma \langle \varphi(\boldsymbol{x}) \delta x \delta z \rangle \\ I_{yz}(\boldsymbol{x}_G) &= -\gamma \langle \varphi(\boldsymbol{x}) \delta y \delta z \rangle \end{aligned} \tag{5}$$

where $(\delta x,\ \delta y,\ \delta z)^T = (\boldsymbol{x} - \boldsymbol{x}_G)$ and the common prefactor is $\gamma = (mV^{\text{AABB}})/V$.

In order to save memory space and computation time, only the eigenvalues extracted from this matrix will be stored as $I_1^\star$, $I_2^\star$ and $I_3^\star$. The positions of the vertices are expressed in the eigenframe given by the eigenvectors of the inertia matrix, with respect to the center of mass of the R-shape. The position and orientation of the shape will thus be defined.

5. Contact detection

A major advantage of R-shapes is to allow for an unambiguous determination of the contact points and associated local frameworks. In fact, all contact configurations between R-shapes are reduced to a set of four elementary types of contacts: vertex-vertex, vertex-edge, vertex-face, and edge-edge; see Fig. 7. It it noteworthy that in this representation a face-face contact is **not** an elementary contact in the sense that it can be described as a combination of the four above elementary contacts. For instance, there may be several vertex-face elementary contacts or a combination of edge-face and vertex-face contacts. Fig. 8 presents a number of contact configurations (not exhaustive) between two R-triangles. As mentioned above, concave shapes and even hollow geometries (*e.g.*, chain links) may well be modeled with this approach. An added benefit of R-shapes is that the contact normals are obtained in a unique way, whereas with conventional polyhedra there would exist a multitude of possible directions for the contacts which do not involve a planar part (*e.g.*, vertex-vertex or vertex-edge).

Each elementary contact type employs distinct geometric principles to determine the contact position $\boldsymbol{c}$, the contact normal $\boldsymbol{n}$, and the normal distance d_n (with negative value when overlap occurs), ensuring physically accurate simulations. Details are presented below, while Fig. 9 provides a graphical support for the interpretation of the corresponding equations.

- **For a vertex-vertex configuration**, just like for a pair of spheres, the contact geometry is defined by the relative positions of the vertices $\boldsymbol{x}_{C\bullet}$ and their radii $R_\bullet$. The contact normal vector $\boldsymbol{n}$ is computed as the unit vector from the vertex of particle j to the vertex of particle i:

$$\boldsymbol{n} = \frac{\mathbf{x}_{Ci} - \boldsymbol{x}_{Cj}}{\|\boldsymbol{x}_{Ci} - \boldsymbol{x}_{Cj}\|} \tag{6}$$

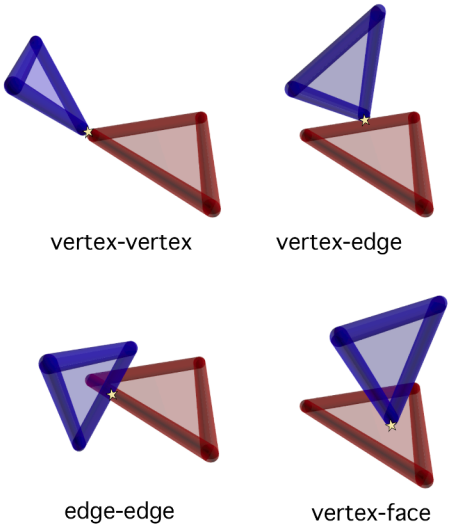

Fig. 7. The four distinct elementary contact configurations between two R-triangles. This figure shows the unique ways in which these shapes can interact at their most basic contact level.

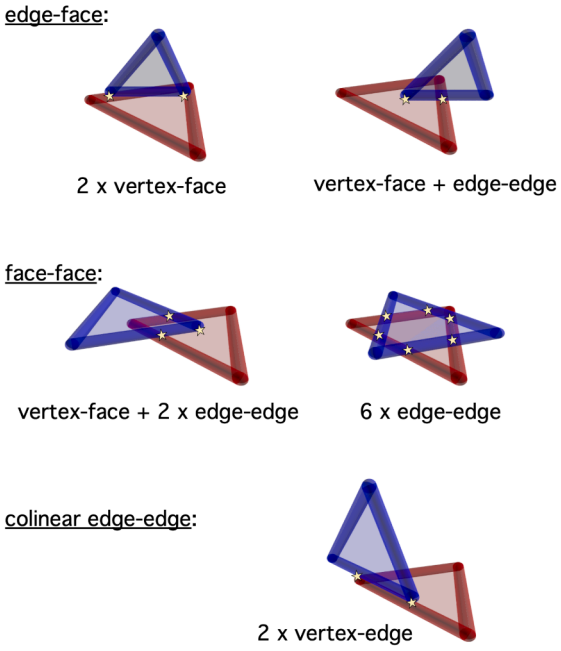

Fig. 8. Examples of complex contact configurations arising from various combinations of elementary contacts between R-shapes. These illustrations highlight a few of the many possible arrangements that can result from combining the four basic contact types.

The overlap distance d_n is derived by subtracting the sum of the Minkowski radii R_i and R_j from the distance between the vertices:

$$d_n = \|\boldsymbol{x}_{Ci} - \boldsymbol{x}_{Cj}\| - (R_i + R_j) \tag{7}$$

The contact point $\boldsymbol{c}$ is positioned along the normal vector, offset by half the overlap distance:

$$\boldsymbol{c} = \boldsymbol{x}_{Ci} - (R_i + d_n/2)\boldsymbol{n} \tag{8}$$

- **For a vertex-edge configuration**, the contact point is determined by projecting the vertex onto the edge. The projection parameter ξ is calculated as:

$$\xi = \frac{(\boldsymbol{x}_{Ci} - \boldsymbol{x}_{Aj}) \cdot \boldsymbol{E}_j}{\boldsymbol{E}_j \cdot \boldsymbol{E}_j} \tag{9}$$

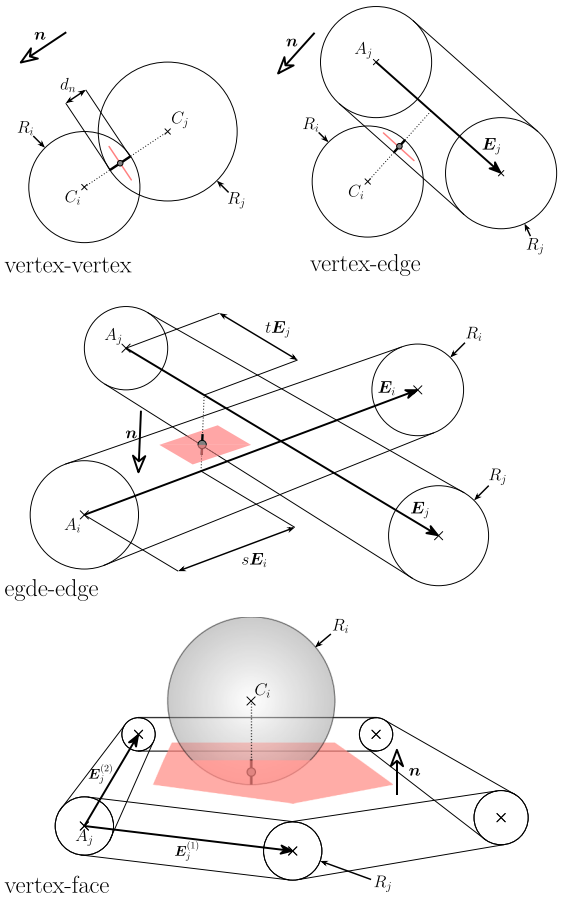

Fig. 9. Illustration of the geometric principles underlying the elementary contact types. Vertex-vertex and vertex-edge contacts are represented in two dimensions, whereas edge-edge and vertex-face contacts are depicted in three dimensions.

where $\boldsymbol{E}_j = \boldsymbol{x}_{Bj} - \boldsymbol{x}_{Aj}$ is the edge vector. If ξ lies within the interval $[0, 1]$, the vertex projects onto the edge, and the contact normal $\boldsymbol{n}$ is computed as:

$$\boldsymbol{n} = \frac{\boldsymbol{D}}{\|\boldsymbol{D}\|} \text{ with } \boldsymbol{D} = \boldsymbol{x}_{Ci} - (\boldsymbol{x}_{Aj} + \xi\, \boldsymbol{E}_j) \tag{10}$$

The overlap distance d_n is:

$$d_n = |(\boldsymbol{x}_{Ci} - \boldsymbol{x}_{Aj}) \cdot \boldsymbol{n}| - (R_i + R_j) \tag{11}$$

and the contact point $\boldsymbol{c}$ is derived similarly to the vertex-vertex case; see Eq. (8).

- **For an edge-edge configuration**, the interactions are resolved by computing the shortest distance between two line segments. The parameters s and t locate the closest points on each edge:

$$s = \frac{e\,b - a\,d}{f} \quad \text{and} \quad t = \frac{c\,b - e\,a}{f} \tag{12}$$

where $a = \boldsymbol{E}_i \cdot \boldsymbol{v}$, $b = \boldsymbol{E}_j \cdot \boldsymbol{v}$, $c = \boldsymbol{E}_i \cdot \boldsymbol{E}_i$, $d = \boldsymbol{E}_j \cdot \boldsymbol{E}_j$, $e = \boldsymbol{E}_i \cdot \boldsymbol{E}_j$, and $f = c\,d - e^2$.

If f is zero within a specified tolerance, the edges are parallel, and the interaction is discarded; the interaction is then managed using vertex-edge contact pairs, as illustrated in the final configuration case of Fig. 8. Otherwise, the contact normal $\boldsymbol{n}$ is computed as the unit vector between the closest points on the edges:

$$\boldsymbol{n} = \frac{\boldsymbol{D}}{\|\boldsymbol{D}\|} \quad \text{with} \quad \boldsymbol{D} = (\boldsymbol{x}_{Ai} + s\,\boldsymbol{E}_i) - (\boldsymbol{x}_{Aj} + t\,\boldsymbol{E}_j) \tag{13}$$

The overlap distance d_n and contact point $\boldsymbol{c}$ are then derived as in previous cases.

$$\begin{cases} d_n &= \|\boldsymbol{D}\| - (R_i + R_j) \\ \boldsymbol{c} &= (\boldsymbol{x}_{Ai} + s\,\boldsymbol{E}_i) - (R_i + d_n/2)\boldsymbol{n} \end{cases} \tag{14}$$

- **For a vertex-face configuration**, the vertex is projected onto the plane defined by the face. The face normal $\boldsymbol{n}$ is computed as the cross product of two edge vectors of the face:

$$\boldsymbol{n} = \frac{\boldsymbol{E}_j^1 \times \boldsymbol{E}_j^2}{\|\boldsymbol{E}_j^1 \times \boldsymbol{E}_j^2\|} \tag{15}$$

where $\boldsymbol{E}_j^1$ and $\boldsymbol{E}_j^2$ are vectors defining the face. The distance h from the vertex to the face is calculated as:

$$h = (\boldsymbol{x}_{Ci} - \boldsymbol{x}_{Aj}) \cdot \boldsymbol{n} \tag{16}$$

If h is negative, the normal is inverted to ensure it points from the face to the vertex. The projected point $\boldsymbol{P}$ is then checked for inclusion within the face using the **crossing number algorithm** (odd-even rule) that is a well-established method in computational geometry for determining whether a point lies inside a polygon. If the point lies inside the face, the overlap distance d_n is computed as:

$$d_n = h - (R_i + R_j) \tag{17}$$

The contact point $\boldsymbol{c}$ is positioned along the normal vector, offset by half the overlap distance; Eq. (8).

For all interaction types, the relative velocity $\mathcal{V}_{ij}$ at the contact point is computed as:

$$\mathcal{V}_{ij} = (\boldsymbol{v}_j + \boldsymbol{\Omega}_j \times (\boldsymbol{c} - \boldsymbol{x}_{Cj})) - (\boldsymbol{v}_i + \boldsymbol{\Omega}_i \times (\boldsymbol{c} - \boldsymbol{x}_{Ci})) \tag{18}$$

where $\boldsymbol{v}_\bullet$ and $\boldsymbol{\Omega}_\bullet$ are, respectively, the linear and angular velocities of the particles, and $x_{C\bullet}$ are the mass centers.

The interaction force models can then be described using the local parameters, namely the normal distance d_n, the normal vector $\boldsymbol{n}$, the relative velocity $\mathcal{V}_{ij}$, and possibly other parameters.

The geometric framework described above provides computationally efficient and physically accurate contact detection. It is nevertheless essential to ensure geometric consistency, *i.e.*, the interpenetration between R-shapes must remain smaller than the radius R, as excessive overlap could lead to interlocking of the shapes. To prevent this situation, it is advisable to estimate the expected orders of magnitude of overlaps in the simulation so that they remain significantly smaller than R. This can be anticipated through the definition of the relative stiffness κ [*e.g.*, 2], commonly used in DEM simulations for granular physics, which accounts for both the contact normal stiffnesses and the pressure experienced by the granular assembly. An optional practical technique in `Rockable` can be used to prevent excessive interpenetrations. It consists in scaling the contact normal stiffness k_n by the factor $d_{\max}/(d_{\max} + d_n)$, so that the stiffness increases sharply as d_n approaches $-d_{\max}$, while choosing $d_{\max} > R$ ensures that k_n remains finite when $d_n = R$.

6. Contact elasticity for R-shapes

Modeling contact elasticity is a critical aspect of aspherical particle packing, as it plays a crucial role in accurately capturing the physics of granular media. The elastic properties of the contacts between the particles significantly influence the overall elasticity of granular materials, including their mechanical response to external forces and their ability to transmit stresses. Therefore, a proper understanding and modeling of contact elasticity are essential for developing accurate and reliable simulations of granular materials. In this section, we discuss the approach used in `Rockable` to model contact elasticity when multiple contact points are involved.

In order to model the contacts between sphero-polyhedra, a total of four elementary interactions are used as explained in Section 5. Consequently, a face-face contact can consist of a minimum of three sub-contacts, but may involve many more (see the second case of Fig. 8 as

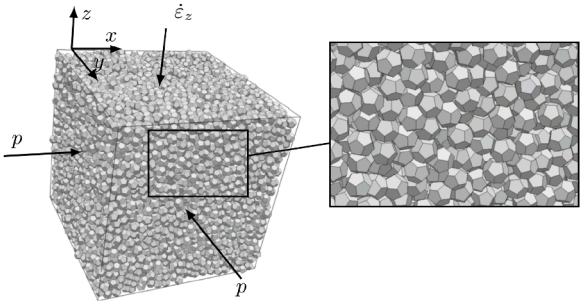

Fig. 10. A snapshot of the sample of dodecahedral particles in the isotropic state, used in the work of Vu et al. [26].

an illustration). To regulate the elastic properties of such compound or multiple contacts, it is crucial to apply a weight (or penalty) to each force (or equivalently, each stiffness) associated with the involved subcontacts. This approach enables accurate control of the elastic behavior of the contacts, and thereby the overall mechanical response of the granular material.

The default approach is to define the penalty factor as the inverse of the number of contact points involved in an interaction between two rigid bodies. With this approach, the resulting linear elastic contact force between the two bodies exhibits an apparent stiffness equivalent to that of a single contact. Alternative methods, which have been implemented but not fully tested and used, involve defining the penalty factor based on the overlaps and/or the surface area of the contact.

In analyzing the elasticity of packings of polyhedral particles, in order to accurately identify and manage the multiple contacts, Vu et al. introduced in Ref. Vu et al. [26] a parameter that reflects the number of face-face and edge-face contacts, and the anisotropy of the contact orientations induced by compression. They predicted the evolution of elastic moduli during compression and showed that our computational approach provides a novel way to manage multiple contacts in polyhedral particle packings. In this study, it was observed that while the number of contact points can be large depending on the number of edges in a face, the number of independent *constraints* remained constant. Taking into account these additional factors, the computational approach provides a comprehensive and efficient way to manage multiple contacts in polyhedral particle packings. This approach allows for an accurate prediction of the behavior of granular materials and extracts valuable information about their connectivity and anisotropy from measurements of elastic moduli [26].

7. Force laws

This section develops a selection of force laws for visco-elasto-adhesive frictional contact to demonstrate their implementation in the code. Based on [27], the model integrates linear elasticity, van der Waals adhesion, Coulomb friction, and viscosity to capture inelastic collisions via an indirect restitution coefficient. Here, the objective is to illustrate the implementation of force laws and to demonstrate that, owing to *R*-shapes, the procedure closely resembles that used for spherical particles. More advanced or specialized laws fall outside the scope of this paper.

The total interaction force $\boldsymbol{f}$ between two particles is the sum of normal and tangential components f_n and $\boldsymbol{f}_t$, respectively:

$$\boldsymbol{f} = f_n \boldsymbol{n} + \boldsymbol{f}_t \tag{19}$$

where $\boldsymbol{n}$ is the contact normal as defined in Fig. 9 together with the position of the contact point $\boldsymbol{c}$. The interaction force depends on local variables, such as the normal distance, as discussed in Section 5.

For the elasto-adhesive part of the contact law, the normal force is the sum of a linear elastic repulsion force $f_n^e = -k_n d_n$, where k_n is the normal stiffness, and a constant adhesion force $-f_c$:

$$f_n = \begin{cases} f_n^e - f_c = -k_n d_n - f_c & \text{for } d_n < 0 \\ 0 & \text{for } d_n \geq 0 \end{cases} \tag{20}$$

If the adhesion force f_c is assumed to represent a van der Waals force, it may be approximated as:

$$f_c = \frac{3}{8}\pi\gamma\langle D\rangle \tag{21}$$

where A is the Hamaker constant, $\langle D\rangle$ is the mean particle diameter, and γ is the surface energy given by

$$\gamma = \frac{A}{18\pi h_0^2} \tag{22}$$

where h_0 is the minimal gap distance between the interacting surfaces of the two particles allowed by surface roughness. This adhesion law illustrates that, beyond usual local variables such as the normal distance, additional parameters can be incorporated into the derivation of a force law. For example, in this case, the total force acting between two particles at a face-face contact may also be approximated as the product of the surface energy by the interacting area, which can be calculated from the geometry of contact. The appropriate expression depends on the material and surface roughness of the particles.

For the tangential force, we used a linear elastic law together with a Coulomb dry friction criterion:

$$\boldsymbol{f}_t = \begin{cases} -k_t \boldsymbol{d}_t & \text{for } \|\boldsymbol{f}_t\| \leq \mu\left(f_n + f_c\right) \\ -\mu\left(f_n + f_c\right)\frac{\dot{\boldsymbol{d}}_t}{\|\dot{\boldsymbol{d}}_t\|} & \text{otherwise} \end{cases} \tag{23}$$

where μ is friction coefficient, $\boldsymbol{d}_t$ is the cumulative tangential displacement, and k_t is tangential stiffness. Note that, as compared to the Coulomb criterion $\|\boldsymbol{f}_t\| \leq \mu f_n$ for cohesionless contacts, here the Coulomb cone is shifted to account for the adhesion force added to the normal force. In other word, only the repulsive part $f_n^e = f_n + f_c$ of the normal force comes into play.

In Eq. (23), the tangential components of the relative velocity and distance vectors are computed as follows:

$$\begin{cases} \dot{\boldsymbol{d}}_t &= \boldsymbol{\mathcal{V}} - \boldsymbol{\mathcal{V}} \cdot \boldsymbol{n} \cdot \boldsymbol{n} \\ \boldsymbol{d}_t &= \sum_{t_0^c}^{t} \dot{\boldsymbol{d}}_t \, \Delta t \end{cases} \tag{24}$$

where the relative velocity is determined using Eq. (18), Δt denotes the time step increment, and t_0^c represents the time at which the contact occurs between two particles.

For energy dissipation, in addition to frictional sliding and adhesion breaking as two possible mechanisms of dissipation, inelastic collisions can also occur. They are characterized by a restitution coefficient $\epsilon_n < 1$. In cohesionless contacts, the value of ϵ_n is controlled by adding a viscous normal force f_n^v to the normal force:

$$f_n^v = -2\alpha_n \sqrt{k_n m_{\text{eff}}}\, \dot{d}_n \tag{25}$$

where $m_{\text{eff}} = m_i m_j/(m_i + m_j)$ is the effective mass of the interacting particles. With this parametrization of the damping force, α_n is simply given by:

$$\alpha_n = -\frac{\ln\left(\epsilon_n\right)}{\sqrt{\ln^2\left(\epsilon_n\right) + \pi^2}}. \tag{26}$$

A similar viscous damping parameter can be added to the expression of the friction force to account for the corresponding tangential restitution coefficient.

In all the expressions forces acting between two particles, the stiffness and damping parameters can depend on the overlap to take care of nonlinear force laws such as Hertz contact. In `Rockable`, all parameter values may be individualized to specified groups of particles, making it possible to simulate multicomponent granular materials.

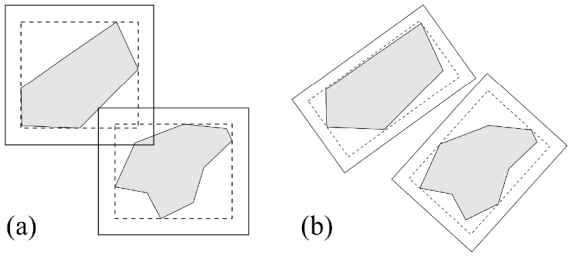

Fig. 11. A two-dimensional sketch of the (a) axis-aligned and (b) oriented bounding boxes; respectively AABB and OBB. To anticipate contacts, the dashed boxes that fit the shapes are inflated with half an alert distance.

8. Inventory of nearby particles

The purpose of the procedures described in this section is to establish a list of potential contacts, i.e. a list of neighbors as an inventory of nearby particles, in the most efficient manner. In many situations, updating this list has a significant cost on the global calculation time (although its role is precisely to optimize the calculations). During the development and use of `Rockable`, it was imperative to optimize this part of the calculation, because we identified that it was excessively time-consuming, particularly for the fall of complex-shaped blocks in a finely defined topography [17–19,28].

The simplest way to build a list of neighbors is to test the proximity of all possible pairs of blocks, which leads to a number of operations equal to the square of the number of elements. In terms of Big-O notation, this is an algorithm with $\mathcal{O}(N^2)$ complexity; however, as long as the list update can be done with a time interval greater than the time step, a modest gain can still be achieved, but it is clearly not sufficient for most applications. This is a critical aspect that explains why a great amount of effort has been devoted to adapting more sophisticated strategies, often borrowed from the world of 3D video games. These strategies focus, on the one hand, on the volumes surrounding the particles and, on the other hand, on the algorithm itself by targeting a $\mathcal{O}(N)$ complexity.

8.1. Oriented bounding boxes (OBB)

The choice of a bounding box shape has a significant influence on the efficiency of the reconstruction, especially if the objects have an elongated or anisometric shape Fig. 11. Both aligned bounding boxes (AABB) and oriented bounding boxes (OBB) were tested. For the latter, a geometric algorithm was developed and used to determine the box with the smallest volume. This algorithm is not described here, but it should be noted that the axes of the optimal OBB do not necessarily correspond to the axes of inertia of the *R*-shape. Testing the proximity of two OBBs is slightly more expensive than testing the proximity of two AABBs, however, for purely geometric reasons, using OBBs for non-spherical shapes is generally more efficient, as it limits the range of objects in all directions. Thus, with dense flow simulations, the computation times can be divided by ∼7 when OBBs are used.

8.2. Linked-cells

The naive strategy allows for a significant reduction in the duration of the simulation, but since the complexity is $\mathcal{O}(N^2)$, the computation time grows rapidly with the number of elements. To overcome this problem, the linked-cells strategy (see Fig. 12) has been implemented. It consists of dividing the whole system (*i.e.*, the AABB box that surrounds all the elements of the simulation) into structured cells (a grid) associated with a list of elements it contains and a dummy cell associated with elements that are too large to be contained in the structured cell, such as infinite planes. For a given object, the proximity test is performed only with the other objects in its own cell, as well as those in neighboring cells. The number of cells involved in the search for neighbors of a 3D object is constant and equal to 27. Fig. 12 gives a 2D representation of the cells involved in the search for neighbors of an element; in this case, there are 9 cells tested. Given that the number of elements per cell is limited, the number of operations per element is almost constant, and the complexity of the algorithm is limited to $\mathcal{O}(N)$ in the worst case.

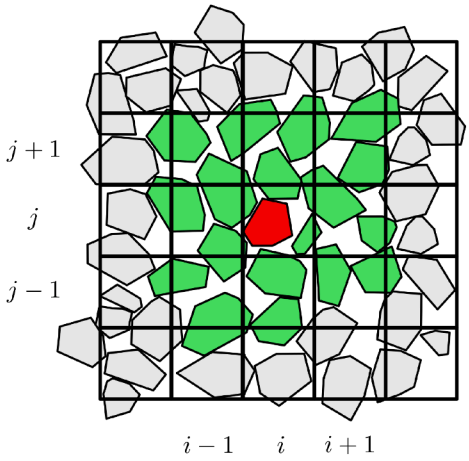

Fig. 12. Link-cell strategy sketched in two dimensions: the proximity test of the red element in the cell (i, j) is performed only with the green elements in the nearby cells. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

A further improvement consists in limiting the linked-cells to the mobile blocks rather than to the whole system, which includes the terrain. From a computational point of view, the terrain is a fixed *R*-shape which has the particularity of being very large and comprising a very large number of sub-elements. The whole terrain will be seen by all the mobile blocks by defining a special cell in the linked-cells grid.

8.3. OBB binary trees

When the proximity of two *R*-shapes is tested, the number of tests can be tremendous and contribute excessively to increase the calculation time. Therefore, a solution has been implemented to overcome this problem: the use of **binary OBB-trees**. The principle is to subdivide the OBB of the elements (including the terrain, if any) into sub-OBBs in a hierarchical manner: the level 0 box contains two boxes at level 1 which cover two parts of the terrain, then the level 1 boxes themselves contain two boxes each, and so on up to the desired level. The OBB tree is constructed by first representing each geometric sub-element (faces, edges, and vertices) as individual leaf nodes, each tightly bounded by an OBB computed from its defining points. These leaf OBBs are then recursively grouped into a hierarchical binary tree. At each level, the set of OBBs is divided along the axis of maximum spatial variance in two pieces, determined from the principal eigenvector of the covariance matrix of the points. The bundles are projected onto this axis, sorted, and divided at the median, forming two subsets that become the two child nodes. This principal-axis, median-split strategy produces a spatially coherent and balanced hierarchy, enabling efficient collision detection by quickly excluding large portions of the shape that cannot intersect.

Fig. 13 shows the sub-OBBs obtained at levels 0, 3, 6 and 12. We can clearly see that the boxes become smaller and smaller and therefore contain fewer and fewer sub-elements (spheres, tubes, and thick 3D-polygons). When searching for contacts, identifying the deepest level OBBs in the binary tree means that the tests can be limited to the sub-elements contained within them.

To get an idea of the gains that can be relaized with this new functionality, let us imagine a situation where a search for contacts is carried out between a cubic block made up of 20 sub-elements (6 faces, 6 edges and 8 vertices) and the terrain made up of $200 \times 200 = 40\,000$ polygons

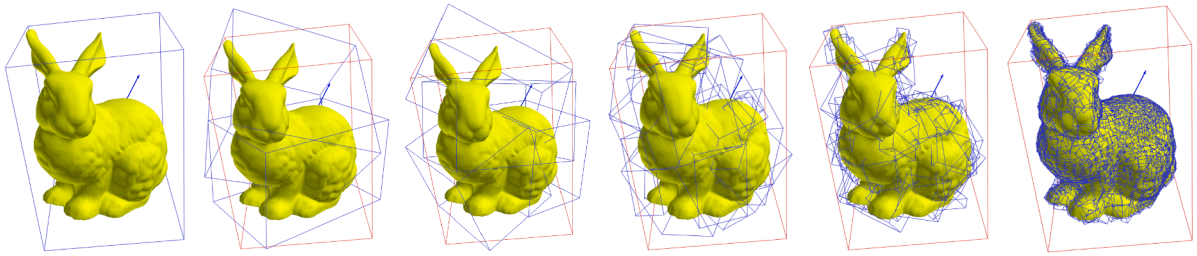

Fig. 13. Presentation of the OBB trees of Stanford Bunny from trunk (single OBB around the entire element) to leaves (OBB around each sub-element), with two levels of branches.

– this configuration corresponds to the rather coarse definition of a topography. The number of tests that must be performed without an OBB binary tree is $20 \times 40\,000 = 800\,000$ tests. Using a depth-6 tree, the smallest boxes would contain around $40\,000/26 = 625$ elements, and assuming a single OBB overlap between the block and a depth-6 box in the terrain, the number of operations would be reduced to $20 \times 625 = 12\,500$ tests, corresponding to 64 times fewer tests. This improvement means substantial savings in simulation time. This gain is vital for the applicability of the DEM approach on realistic terrain, as it conditions the hope of reaching the end of a simulation in a reasonable time.

9. Explicit time integration

Since a R-shape is a rigid body, only the time evolution of the position of mass center and overall rotation must be computed. The movement of the entities that make up the body (with primitive shapes) is governed by the relations of rigid motion. In other words, the movement is time-integrated in the same manner as a classical clump.

9.1. Selection of an integration scheme

Four integration schemes implemented in Rockable have been tested in one dimension for damped oscillating masses. Their very different patterns of complexity are detailed in Appendix A from the simplest to the most complex: the Forward **Euler** scheme, the **velocity-Verlet** scheme, the **Beeman** scheme, and the 4th-order Runge-Kutta-Nyström (**RKN4**) scheme. The second-order ordinary differential equation (ODE) to be solved is:

$$m\,\ddot{x} + \nu\,\dot{x} + k\,x = F_0, \tag{27}$$

where the number of top-dots indicates the order of the time derivative of a displacement δx, m is the mass, ν is the damping coefficient of a dashpot, and k is the stiffness of the spring.

[This part has been moved to Appendix A]

To compare the accuracy of the four schemes, we have computed the error in the mass position. This error is calculated as the sum over N_t discrete time steps of the absolute difference between the numerical solution $x(t)$ obtained using each scheme and the analytical solution $x_{\text{sol}}(t)$:

$$E_{\text{accuracy}} = \frac{1}{N_t}\sum_{i=0}^{N_t} \left|x(t_i) - x_{\text{sol}}(t_i)\right|, \tag{28}$$

where $x_{\text{sol}}(t)$ is the analytical solution of Ordinary Differential Eq. (27) with the initial conditions $x(0) = 0$ and $\dot{x}(0) = 0$. Although there are various methodologies for evaluating the accuracy and stability of integration schemes, such comparisons are beyond the scope of this paper. However, to demonstrate the versatility of `Rockable` in accommodating various integration schemes, Fig. 14 provides an investigation into the effect of time step variations on computational accuracy.

To provide a clear and consistent evaluation, the following dimensionless forms of parameters were used for all integration schemes: unit mass ($m = 1$), unit stiffness ($k = 1$), and two different values for the viscous parameter $\nu = 0$ (no dissipation case) and $\nu = 1$ (*i.e.*, half the critical viscosity $2\sqrt{mk}$). An external unit force ($F_0 = 1$) was applied and the simulation was run for a maximum time of $t_{\max} = 20$. For each integration scheme, the time step was progressively reduced, starting from a value of $\Delta t_c = N_s\sqrt{m/k} = N_s$, which is in the order of the critical time step. To facilitate comparison between schemes, the time step Δt used in each case is presented in Fig. 14 relative to the critical increment Δt_c, and also relative to the number N_s of acceleration computations per time step.

Based on the results presented in Fig. 14, it is clear that the explicit Euler integration scheme does not perform well, as expected. The error in the displacement is quite large, even for relatively small time steps, and the scheme is inaccurate for both the undamped and half-critically damped cases. On the other hand, the Beeman scheme performs very well for the smallest time-steps, with an error E that is several orders of magnitude smaller than the other schemes. However, as the time-step size increases, the error also increases, and the scheme becomes less accurate than some of the other schemes, particularly for the damped case. Among the schemes considered, the velocity-Verlet scheme performs well for both the non-damped and viscous-damped cases, and for a wide range of time steps. In particular, for the largest time steps, the velocity-Verlet scheme appears to be the most accurate (with a relative error that is smaller than the other schemes). Upon reducing the time step, both the Beeman and RKN4 schemes exhibit superior accuracy compared to the velocity-Verlet scheme. Our analysis (results not shown here) confirmed that the computational time across these schemes did not significantly differ. Therefore, the widespread popularity of the velocity-Verlet scheme cannot be attributed solely to its accuracy.

In general, the choice of an integration scheme depends on the particular requirements of the simulation, which include not only the level of accuracy required, but also the stability of the scheme. To assess the stability of each scheme, we introduce the Root Mean Squared Error, denoted as $E_{\text{stability}}$. This metric is derived from the total energy of the system and is calculated as follows:

$$E_{\text{stability}} = \sqrt{\frac{1}{N_t}\sum_{i=0}^{N_t}\Big(E_{\text{tot}}(t_i) - E_{\text{tot}}(t_0)\Big)^2}, \tag{29}$$

where E_{tot} represents the total energy at time t, and is defined as:

$$E_{\text{tot}}(t) = \frac{k\big(x(t)\big)^2}{2} + \frac{m\big(v(t)\big)^2}{2}. \tag{30}$$

Fig. 15 illustrates the progression of this error as a function of time step, being similarly normalized. The aim is to achieve a constant value for the error $E_{\text{stability}}$ when the time step is changed, indicating stability. The Euler scheme fails to attain this stability within the tested time step range when viscosity is absent. Conversely, when viscosity is active, the Euler scheme exhibits superior stability compared to the RKN4 and Beeman schemes. For the remaining schemes, stability is achieved with increasingly larger time step, irrespective of whether viscosity is active or not. It is noteworthy that the Beeman scheme maintains stability over larger time steps as compared with the RKN4 scheme. Intriguingly,

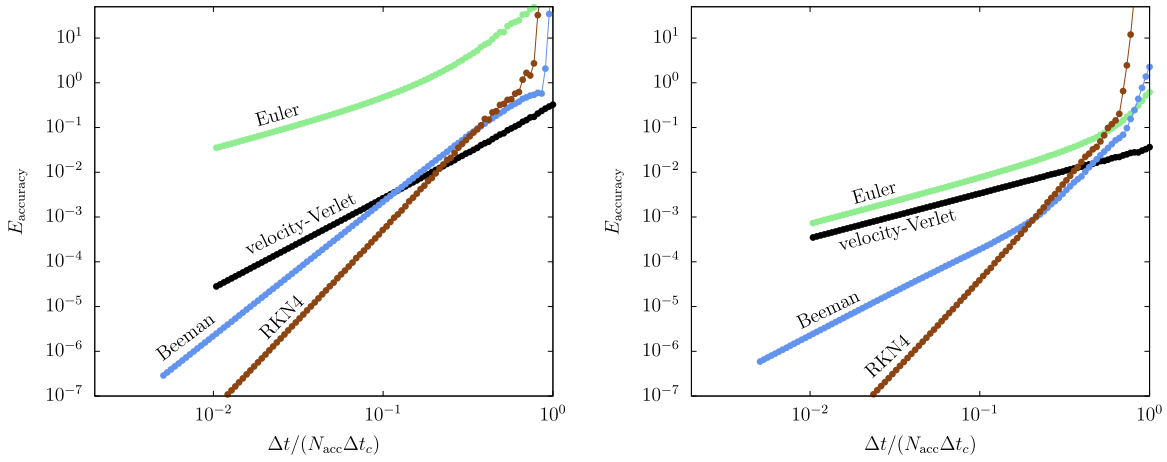

Fig. 14. Comparison of the computational accuracy of various integration schemes for different values of time step. The kinematics error E_{accuracy} is plotted against the time step relative to the critical time step and the number N_{acc} of acceleration computation per time step, for two different viscous parameters: (left) $\nu = 0$, (right) $\nu = \sqrt{mk}$.

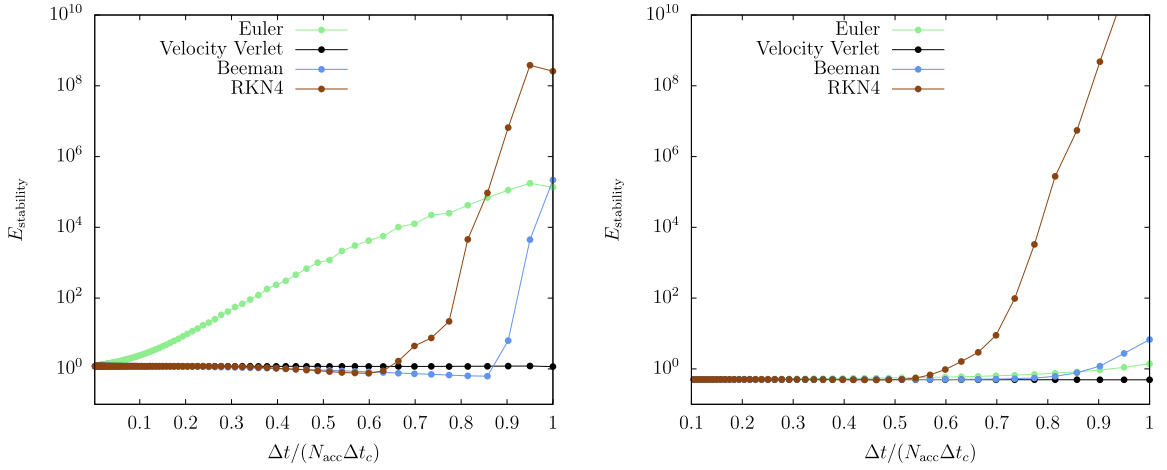

Fig. 15. Comparison of the computational stability of various integration schemes for different time-step sizes. The Root Mean Squared Error $E_{\text{stability}}$ is plotted against the time step relative to the critical time step and the number N_{acc} of acceleration computation per time step, for two different viscous parameters: (left) $\nu = 0$, (right) $\nu = \sqrt{mk}$.

the velocity-Verlet scheme demonstrates stability across the largest time steps, regardless of the activation of viscous damping. These numerical experiments demonstrate the *symplectic* nature of the velocity-Verlet scheme, which means that it is robust against the variations of time integration parameters and is therefore well suited for long-term simulations.

Given the observations presented in Figs. 14 and 15, the velocity-Verlet scheme emerges as a favorable choice for a broad spectrum of situations. This is due to its combination of accuracy, stability, and relative simplicity in implementation compared to higher-order schemes. Consequently, the velocity-Verlet scheme is adopted as the default scheme to be used in `Rockable`, although the other three are also available. Appendix B outlines the 3D implementation of the velocity-Verlet scheme, accounting for non-spherical geometries and rotational effects.

[This part has been moved to Appendix B]

10. Software architecture and extensibility

The software `Rockable` is designed with a strong emphasis on **academic usability**, ensuring that it remains accessible, modular, and extensible. The core philosophy guiding its development is twofold: (1) minimizing dependencies on external libraries to simplify deployment and usage, and (2) enabling straightforward extensions through direct modifications to the C++ codebase. This approach ensures that researchers and developers can adapt the software to their specific needs without encountering unnecessary complexity.

Unlike many modern simulation frameworks that rely on interpreted languages (*e.g.*, Python, Lua or Julia) or Domain Specific Language (DSL), this software adopts a lightweight design. The configuration of simulations and the management of periodic backups are handled through a dedicated, human-readable format. This format is thoroughly documented,[1] allowing users to quickly familiarize themselves with the tool and set up simulations efficiently. By avoiding interpreted layers, the software reduces overhead and potential performance bottlenecks, while preserving clarity in the simulation setup.

For advanced users seeking to implement custom interaction models or extend existing functionalities, the software provides a robust inheritance-based framework. Key components of the simulation pipeline are designed as abstract base classes, which users can derive to introduce new behaviors. The most notable extensible components include:

[1] https://richefeu.github.io/rockable/

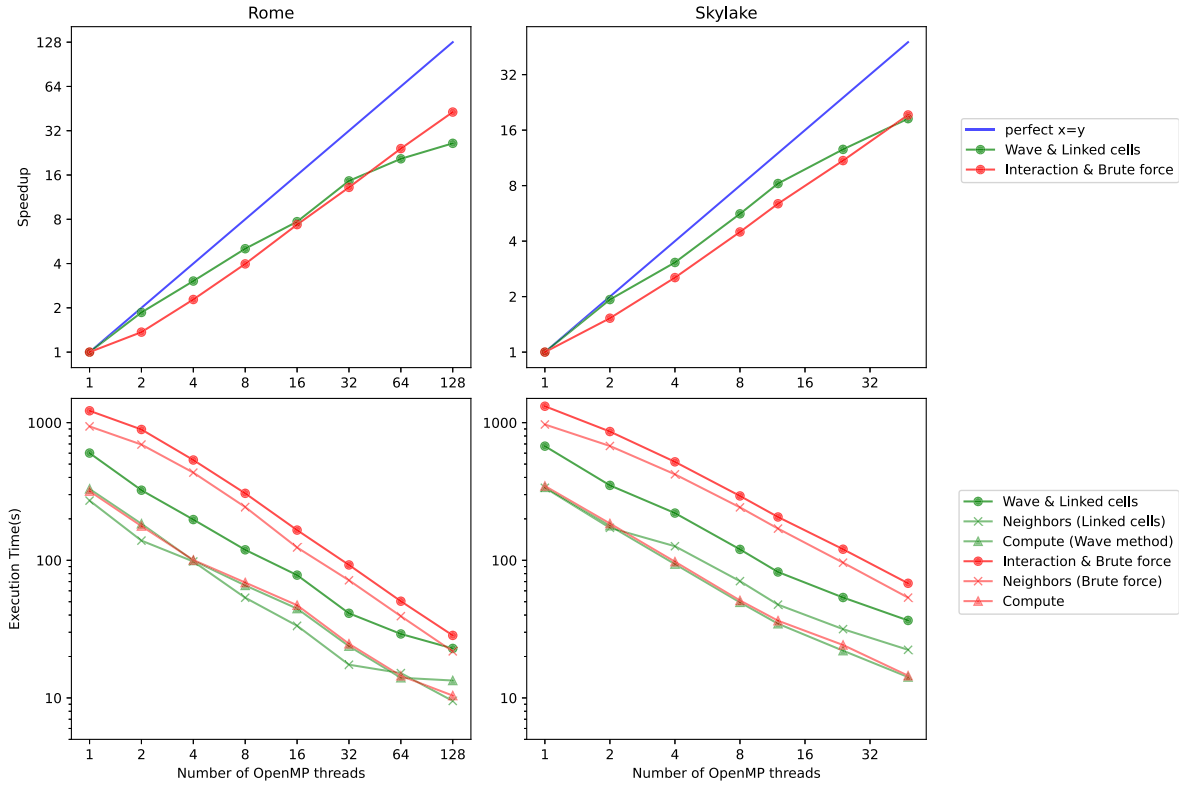

Fig. 16. Speedup (higher is better) and execution time in s (lower is better) for a simulation of 99 995 polyhedra over 500 timesteps in strong scaling. Simulations are run on two type of nodes, Rome: a bi-processor AMD Rome (Epyc) of 64 cores per processor, and Skylake: a bi-processor Intel Skylake 8168 of 24 cores per processor.

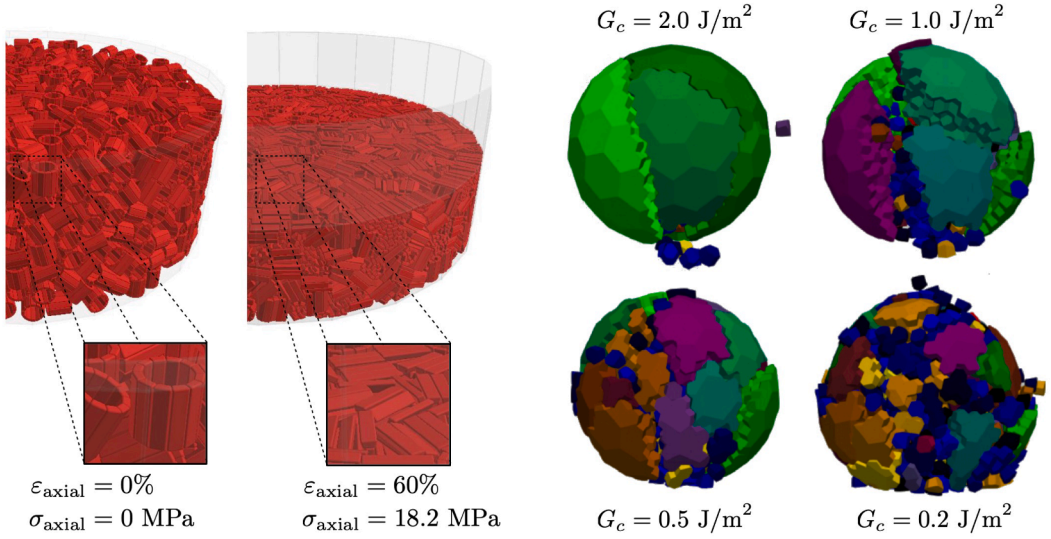

Fig. 17. (Left) Axially compressed cylindrical sample with a constant diameter of 35 cm and an initial height of 12.2 cm composed of 1924 breakable shells (12 sectors); after [39]. (Right) Fragmentation of a particle impacting a rigid wall for different values of fracture energy G_c. The simulations were carried out with an impact velocity of 4.5 m/s; after [40].

- `ForceLaw`: this class serves as the foundation for defining interaction models, including cohesive models discussed earlier. By deriving from `ForceLaw`, users can implement custom force laws tailored to their research requirements.
- `BodyForce`: this component allows the application of volumetric forces, such as gravitational fields directed toward a specific point or between discrete elements. Custom implementations can model complex physical phenomena, such as non-uniform gravitational fields or electromagnetic interactions.
- `Boundary`: this class manages geometric boundaries and boundary conditions. Users can extend it to introduce custom geometries or enforce specialized constraints, such as rotating drums as shown in Fig. 1.
- `PreproCommand`, `DataExtractor`, and `PostProcessor`: these classes facilitate data analysis and post-processing. Users can derive from them to implement custom preprocessing steps, extract specific simulation metrics, or perform advanced post-processing tasks, such as statistical analysis or visualization.

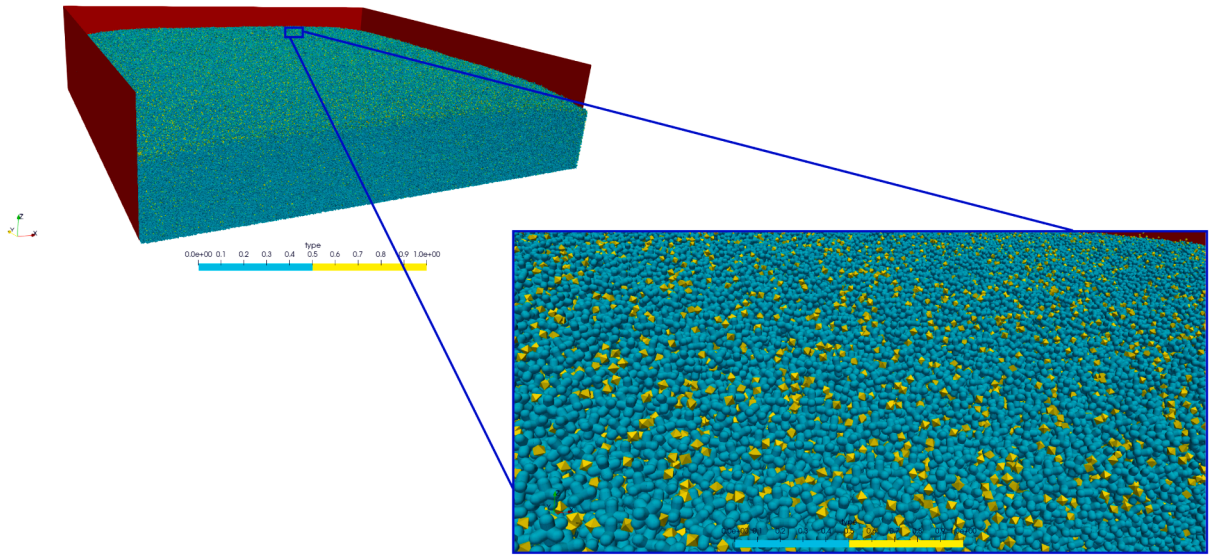

Fig. 18. Five millions octahedra (yellow) mixed with five millions hexapods (blue) deposited into a box. This simulation has been realized with ExaDEM over 4096 cores on CCRT/Topaze. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

This design choice prioritizes performance and consistency, as it avoids runtime interpretation overhead. Additionally, the modular structure of the codebase promotes **collaborative development**, as individual components can be tested and validated independently. By adhering to these principles, the software not only serves as a powerful tool for discrete element simulations but also as a flexible platform for innovation in computational research.

11. Code performance

`Rockable`, like most DEM codes, aims to offer decent performance on multi-core machines while maintaining a low memory footprint for running complex simulations in reasonable timescales. It has two main performance bottlenecks: contact detection and contact computation, while other sections are usually cheap, such as resetting data, force storage, or computing the time scheme. The implementation integrates state-of-the-art algorithms to speed up contact detection, such as the linked-cell method coupled with the Verlet list method (avoid triggering contact detection at each time step) when the number of polyhedra is high. To improve the detection of polyhedra with fine-grained geometry, an OBB binary tree algorithm is used. When dealing with a small number of particles, the straightforward approach involves testing each particle i against every other particle j with $j > i$. This method, which involves $N(N-1)/2$ tests, that is, a complexity $\mathcal{O}(N^2)$, referred to as the "brute force" algorithm in `Rockable`, is commonly employed. These aspects with the implemented solutions to improve performance were detailed in previous sections.

Beyond these technical improvements, the solver takes also advantage of an OpenMP thread parallelization for the most computationally intensive sections. Whereas thread parallelization for contact detection can be simply done by iterating on cells for the linked-cell method or on particles for the trivial algorithm because computations are independent, this is not the case for contact computation because the contact forces applied from a particle A to a particle B are the opposite of the forces applied by particle B to particle A. In order to preserve precious computation cycles, this value is computed once and incremented for both particles. However, by iterating over iterations, two OpenMP threads can update the same data at the same time. To overcome this issue, thread mutexes (locks) are added, but these synchronizations can drastically decrease the overall performance.

To mitigate this difficulty, `Rockable` proposes two lock-free algorithms, the first one consists of reordering the "force" updates and iterating over particles instead of interactions. Another lock-free algorithm is based on the wave method (used in a Molecular Dynamics code [29]) that consists in iterating over cells and waves; a wave is a list of cells. Cells are classified in 8 or 27 waves such as two threads processing two different cells of the same wave cannot update data of the same particles. Note that this last algorithm needs to be used for a very large number of particles to achieve better performance than other algorithms.

Fig. 16 shows speedups according to OpenMP parallelization strategies for the simulation composed of 99 995 polyhedra over 500 time steps. We observe that for both machines, using the "brute force" algorithm achieved better speedup for this scenario; however, the sequential time is twice as long as the linked-cell method (see Execution Time). As a consequence, the linked-cell strategy should be preferred in the case of large numbers of polyhedra.

12. Perspectives and conclusions

In this section, we discuss potential improvements and future developments in the `Rockable` framework. These envisioned features represent exciting directions for further expanding the capabilities and applications of the simulation tool. It is worth noting that many of them have already been at least partially implemented and tested, demonstrating the ongoing progress and potential of the tool. Some of these developments are the following:

- **Periodic boundary conditions**. Introducing periodic boundary conditions [30] allows simulations to model systems with repeating patterns or structures, which is often seen in various scientific and engineering contexts. This feature, partly implemented in `Rockable` as shown in Fig. 10 – but also in Figs. 1 and 2(a) in the single direction of the drum axis – enables simulation of larger systems while reducing edge effects and is particularly valuable in the context of concurrent double-scale simulations such as FEM×DEM [*e.g.,* 31,32] or MPM×DEM [*e.g.,* 33–35].
- **Specialized boundary conditions**. Beyond periodic boundaries, specialized boundary conditions are developed to model scenarios with more complex external influences. This might include specific interactions with walls, constraints, or environmental conditions relevant to the target applications. Fig. 1 illustrates such

a specialized boundary, specifically a perfectly cylindrical rotating drum (not faceted). Fig. 3 presents another specialized boundary, featuring a terrain R-surface optimized for rock fall simulations.

- **Coupling with fluids**. Combining DEM with computational fluid dynamics, such as the Lattice Boltzmann Method (LBM), could open up new avenues for modeling complex systems where the interaction between particles and fluids is critical [36]. This is particularly relevant in scenarios like sediment transport, granular flows inside fluids, or other multi-physics problems [37,38].
- **Particle fracture**. The initial framework for modeling crushable hollow particles in `Rockable` was predicated on a yield criterion that integrates both normal and tangential force components [39]. The packing is capable of releasing internal void space upon particle fracture, which consequently exhibits high compressibility as illustrated in Fig. 17(Left). A Bonded Cell Method for particle breakage has already been introduced [40]; see Fig. 17(Right). It models the particle as an aggregate of polyhedral cells (R-shapes) where the bonds are governed by the Griffith criterion of fracture. A crack propagates if the elastic energy released exceeds the fracture energy. Compressive forces do not contribute to fracture. For a cell-to-cell interface of finite area, a "coarse-grained" energy criterion is used: when the energy release rate G meets or exceeds the fracture energy G_c at a face-face contact between two cells, the cohesive interface fails, and the contact becomes frictional.
- **Soft/compliant particles**. Expanding the framework to handle flexible particles would be a significant advancement. This would allow simulation of particle materials that can deform under various conditions, such as soft particles, polymers, or biological tissues. The objective is not to determine the kinematic field with a large quantity of degrees of freedom but rather to make certain ones possible.
- **Non-Smooth Contact Dynamics**. The Contact Dynamics approach, which has been used for a long time by our research group [3], will be shortly added in `Rockable`. The whole theoretical modeling is ready, and the next step is to implement it following the approach used in [41,42] for soft particles modeled with the material point method.
- **High Performance Computing (HPC)**. A simulation with `Rockable` can efficiently manage around 100 000 sphero-polyhedral particles over the course of a few weeks. However, simulating millions of particles required the use of HPC. To preserve the clarity and simplicity of `Rockable`'s source code, we opted to develop HPC capabilities as a separate effort. As a result, a new HPC code, named ExaDEM [43], was created. It uses a subset of `Rockable`'s functionalities to explore very large-scale simulations; see Fig. 18. The primary goal is to improve polyhedron-intensive simulations on modern HPC platforms. The plan is to continuously integrate R-shape functionalities into the ExaNBODY platform [44], allowing the combination of these features with the latest hybrid parallelization techniques, including MPI + OpenMP and GPU acceleration using CUDA or HIP backends.

These novel features would not only enhance the versatility of `Rockable` but also broaden its applicability across a wide range of scientific and engineering domains. It highlights the software development team's commitment to continuous improvement and adaptability in meeting the evolving needs of the simulation community.

In conclusion, the framework `Rockable` provides a versatile solution designed to simplify the implementation of sphero-polyhedra, or R-shapes, in DEM simulations. Particle-based modeling is at the heart of various scientific and engineering activities, providing crucial information about granular systems and their behavior. As we venture further into the intricacies of these simulations, the importance of managing the complexity of particle shape becomes increasingly apparent. Interestingly, `Rockable`, despite having been in development for a decade, is now available in the spirit of open collaboration and accessibility. In this paper, we have introduced `Rockable` and shown some features of its capabilities. The adaptability of the framework is revealed when it effortlessly handles non-convex shapes and complex scenarios, effectively bridging the gap between real physical phenomena and numerical simulations.

Key aspects of `Rockable`, such as the definition of R-shapes, mass property calculations, contact detection mechanisms, integration schemes, and performance optimization strategies, were discussed. In particular, we showed that the integration of linked-cell structures and binary OBB trees has significantly enhanced the framework's scalability and computational efficiency, enabling more extensive simulations. Finally, we would like to underline that `Rockable` serves a different purpose compared to `YADE` [45] – a pioneering DEM framework that originated from Laboratoire 3SR and excels in its designated domain. Instead, `Rockable` enriches the DEM world by offering a versatile solution to specific challenges associated with sphero-polyhedral representation of particle shapes and dynamic simulations of their interactions.

CRediT authorship contribution statement

Vincent Richefeu: Writing – original draft, Software, Methodology, Conceptualization; **Gaël Combe:** Writing – review & editing, Methodology, Conceptualization, Software, Methodology, Conceptualization; **Lhassan Amarsid:** Software, Conceptualization; **Raphaël Prat:** Writing – original draft, Software, Methodology, Conceptualization; **Jean-Mathieu Vanson:** Software, Methodology, Conceptualization; **Saied Nezamabadi:** Methodology, Conceptualization; **Patrick Mutabaruka:** Methodology, Conceptualization; **Jean-Yves Delenne:** Writing – review & editing, Software, Methodology, Conceptualization; **Farhang Radjaï:** Writing – review & editing, Methodology, Conceptualization.

Uncited Citation

Missing citation Fig.11.

Data availability

Data will be made available on request.

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.

Acknowledgment

This work was performed using HPC resources from CCRT funded by the CEA/DEs simulation program.

Appendix A. Integration schemes in `Rockable`

The tested schemes are listed below from the simplest to the most complex. They correspond to the schemes that are implemented in `Rockable`.

- The **Forward Euler scheme** is regarded as the most fundamental explicit first-order approach. Despite its reputation for not providing an optimal solution, it is included only for testing purposes. It can be summarised at each time increment Δt by the following sequence:

$$\begin{cases} x(t+\Delta t) & \leftarrow x(t) + \dot{x}(t)\Delta t \\ \dot{x}(t+\Delta t) & \leftarrow \dot{x}(t) + \ddot{x}(t)\Delta t \\ \ddot{x}(t+\Delta t) & \leftarrow \mathbf{ACC}\Big(x[,\ \dot{x}],\ t\Big) \end{cases} \tag{A.1}$$

where $\mathrm{acc}\Big(x[,\ \dot{x}],\ t\Big)$ is a function that expresses the acceleration of the mass at a time t, as a function of the current position x, and that depends optionally on the current velocity $\dot{x}$. In this computational context, the term "current" refers to the numerical value stored within the computer's memory.

- The **velocity-Verlet scheme**, which is particularly popular for dynamical systems, is a second-order method that operates according to the following steps:

$$\begin{cases} x(t+\Delta t) & \leftarrow x(t) + \dot{x}(t)\Delta t + \ddot{x}(t)\dfrac{\Delta t}{2} \\ \dot{x}(t+\frac{\Delta t}{2}) & \leftarrow \dot{x}(t) + \ddot{x}(t)\dfrac{\Delta t}{2} \\ \dot{x}^\star & \leftarrow \frac{1}{2}\Big(\dot{x}(t-\frac{\Delta t}{2}) + \dot{x}(t+\frac{\Delta t}{2})\Big) \\ \ddot{x}(t+\Delta t) & \leftarrow \mathbf{ACC}\Big(x[,\ \dot{x}^\star],\ t\Big) \\ \dot{x}(t+\Delta t) & \leftarrow \dot{x}(t+\frac{\Delta t}{2}) + \ddot{x}(t+\Delta t)\dfrac{\Delta t}{2} \end{cases} \tag{A.2}$$

It is important to note that the velocity at the moment of computing the acceleration is time-shifted. Therefore, if required, the velocity needs to be estimated at the appropriate time step to ensure the accuracy of the calculations. This can be done using the interpolated value $\dot{x}^\star$ just before computing the acceleration. By carefully considering the time-shifted velocity and applying appropriate estimation techniques, the velocity-Verlet scheme can provide an effective and accurate solution for dynamical systems.

- The explicit variant of **Beeman scheme** is another popular method for solving dynamical systems that provides a higher order of accuracy compared to the velocity-Verlet scheme. It is a fourth-order method that uses a predictor-corrector approach to calculate the position, velocity, and acceleration of a system at each time step. This scheme proceeds as follows:

$$\begin{cases} \ddot{x}(t) & \leftarrow \mathbf{ACC}\Big(x[,\ \dot{x}],\ t\Big) \\ x(t+\Delta t) & \leftarrow x(t) + \dot{x}(t)\Delta t \\ & \quad + \frac{1}{6}\Big(4\ddot{x}(t) - \ddot{x}(t-\Delta t)\Big)\Delta t^2 \\ \dot{x}^\star & \leftarrow \dot{x}(t) + \frac{1}{6}\Big(2\ddot{x}(t) + 5\ddot{x}(t) \\ & \quad - \frac{1}{2}\ddot{x}(t-\Delta t)\Big)\Delta t \\ \ddot{x}(t+\Delta t) & \leftarrow \mathbf{ACC}\Big(x[,\ \dot{x}^\star],\ t+\Delta t\Big) \\ \dot{x}(t+\Delta t) & \leftarrow \dot{x}(t) + \frac{1}{12}\Big(5\ddot{x}(t+\Delta t) + 8\ddot{x}(t) \\ & \quad - \ddot{x}(t-\Delta t)\Big)\Delta t \end{cases} \tag{A.3}$$

We can see that the scheme requires more computational effort due to the predictor-corrector approach. In particular, it requires $N_s = 2$ estimations of acceleration within a time step. However, this additional effort can be justified in cases where high accuracy is required. Unlike the velocity-Verlet scheme, the velocity $\dot{x}^\star$ is not an average velocity estimate. Instead, it is a *predicted* velocity used in the second calculation of acceleration. The velocity is then *corrected* at the conclusion of a step.

- The **4th-order Runge-Kutta-Nyström scheme** (RKN4) is a powerful multistep method for solving dynamical systems that provides a high order of accuracy [46,47]. It is a method that uses four function evaluations per step to calculate the acceleration at each time step.

The RKN4 scheme proceeds as follows:

$$\begin{cases} \ddot{x}_1 & \leftarrow \mathbf{ACC}\Big(x[,\ \dot{x}],\ t\Big) \\ x & \leftarrow x(t) + \dot{x}(t)\dfrac{\Delta t}{2} + \ddot{x}_1\dfrac{\Delta t^2}{8} \\ \dot{x} & \leftarrow \dot{x}(t) + \ddot{x}_1\dfrac{\Delta t}{2} \\ \ddot{x}_2 & \leftarrow \mathbf{ACC}\Big(x[,\ \dot{x}],\ t\Big) \\ \dot{x} & \leftarrow \dot{x}(t) + \ddot{x}_2\dfrac{\Delta t}{2} \\ \ddot{x}_3 & \leftarrow \mathbf{ACC}\Big(x[,\ \dot{x}],\ t\Big) \\ x & \leftarrow x(t) + \dot{x}(t)\Delta t + \ddot{x}_3\dfrac{\Delta t^2}{2} \\ \dot{x} & \leftarrow \dot{x}(t) + \ddot{x}_3\Delta t \\ \ddot{x}_4 & \leftarrow \mathbf{ACC}\Big(x[,\ \dot{x}],\ t\Big) \\ x(t+\Delta t) & \leftarrow x(t) + \dot{x}(t)\Delta t \\ & \quad + (\ddot{x}_1 + \ddot{x}_2 + \ddot{x}_3)\dfrac{\Delta t^2}{6} \\ \dot{x}(t+\Delta t) & \leftarrow \dot{x}(t) \\ & \quad + (\ddot{x}_1 + 2\ddot{x}_2 + 2\ddot{x}_3 + \ddot{x}_4)\dfrac{\Delta t}{6} \end{cases} \tag{A.4}$$

With an even more computational effort, the scheme requires $N_s = 4$ estimations of acceleration within a time step.

B. Detailed implementation of 3D velocity-verlet

Classically, a motion computation in discrete elements models consists in alternating two operations for each body: (1) the calculation of the resultant force and moment at the center of mass from the forces (and moments) of contact and gravity; (2) the integration of Newton's second law (for translational motions) and Euler's equations (for rotational motions). For this purpose, the velocity-Verlet scheme, which seems to achieve a good compromise between accuracy, stability and also memory saving, is used. For translational movements, the algorithm can be formalized in 3 dimensions as follows:

$$\begin{cases} \boldsymbol{x}_G(t+\Delta t) \leftarrow \boldsymbol{x}_G(t) + \boldsymbol{v}_G(t)\Delta t + \frac{1}{2}\boldsymbol{a}_G(t)\Delta t^2 \\ \boldsymbol{v}_G(t+\Delta t) \leftarrow \boldsymbol{v}_G(t) + \frac{1}{2}\big[\boldsymbol{a}_G(t) + \boldsymbol{a}_G(t+\Delta t)\big]\Delta t \end{cases} \tag{B.1}$$

with

$$\boldsymbol{a}_G(t) = \frac{\boldsymbol{F}(t)}{m} + \boldsymbol{g} \tag{B.2}$$

It should be noted here that the velocity is calculated in a stepwise fashion by estimating the accelerations twice at times t and $t+\Delta t$. In order to get closer to the computer implementation, the symbol $\leftarrow$ has been used; it signifies that the variable on the left-hand side is updated with the value of the variable on the right-hand side within the computer's memory.

For rotational movements, it seems important to use exactly the same scheme, which unfortunately is not always the case in 3D implementations. For convenience, the angular position of the blocks is given by a unitary quaternion which has some advantages, including better numerical stability. A quaternion is a mathematical tool that extends the concept of complex numbers so that:

$$q_i i + q_j j + q_k k + s \quad \text{with} \quad i^2 = j^2 = k^2 = -1 \tag{B.3}$$

Talking about a unit quaternion means that $q_i^2 + q_j^2 + q_k^2 + s^2 = 1$. By analogy with complex numbers, quaternions are defined by a real scalar s and an imaginary vector $\boldsymbol{i} = (i, j, k)^T$, which facilitates the introduction of this conventional notation of quaternions:

$$\breve{\boldsymbol{q}} \equiv [\![s \mid \boldsymbol{i}]\!] \tag{B.4}$$

The values of s and $\boldsymbol{i}$ to represent an orientation have, unfortunately, nothing intuitive (unlike Euler angles for example), it is easy to calculate a matrix of rotation or any desired angle using simple mathematical relations. When the orientation of a rigid body is given by a unit quaternion $\breve{\boldsymbol{q}}$, an interesting property is that the time derivative of the

quaternion $\dot{\breve{q}}$ can be computed using the instantaneous velocity vector $\boldsymbol{\Omega}$ of the body as follows:

$$\dot{\breve{q}}(\boldsymbol{\Omega}) = \frac{1}{2}\breve{\boldsymbol{\Omega}} \star \breve{q} \quad \text{where} \quad \breve{\boldsymbol{\Omega}} \equiv [\![0 \mid \boldsymbol{\Omega}]\!] \tag{B.5}$$

where the operator $\star$ denotes the *Hamilton product*, which expands as follows:

$$[\![s_1 \mid \boldsymbol{i}_1]\!] \star [\![s_2 \mid \boldsymbol{i}_2]\!] = [\![s_1 s_2 - \boldsymbol{i}_1 \cdot \boldsymbol{i}_2 \mid s_1 \boldsymbol{i}_2 + s_2 \boldsymbol{i}_1 + \boldsymbol{i}_1 \times \boldsymbol{i}_2]\!] \tag{B.6}$$

It follows from Eq. (B.5) that the second derivative of the quaternion can be written:

$$\ddot{\breve{q}}(\boldsymbol{\Omega},\ \dot{\boldsymbol{\Omega}}) = \frac{1}{2}\dot{\breve{\boldsymbol{\Omega}}} \star \breve{q} + \frac{1}{4}\breve{\boldsymbol{\Omega}} \star (\breve{\boldsymbol{\Omega}} \star \breve{q}) \tag{B.7}$$

where $\dot{\breve{\boldsymbol{\Omega}}} \equiv [\![0 \mid \dot{\boldsymbol{\Omega}}]\!]$. By following the velocity-Verlet scheme, the orientation given by a quaternion $\breve{\boldsymbol{Q}}$ is thus written – formally and in a manner similar to Eq. (B.1):

$$\begin{cases} \breve{\boldsymbol{Q}}(t+\Delta t) & \leftarrow \breve{\boldsymbol{Q}}(t) + \dot{\breve{\boldsymbol{Q}}}(t)\Delta t + \frac{1}{2}\ddot{\breve{\boldsymbol{Q}}}(t)\Delta t^2 \\ \boldsymbol{\Omega}(t+\Delta t) & \leftarrow \boldsymbol{\Omega}(t) + \frac{1}{2}\big(\dot{\boldsymbol{\Omega}}(t) + \dot{\boldsymbol{\Omega}}(t+\Delta t)\big)\Delta t \end{cases} \tag{B.8}$$

where, for the sake of brevity, the dependence of quaternion derivatives on rotational velocity and acceleration has not been explicitly noted, but $\dot{\breve{\boldsymbol{Q}}}(t) \equiv \dot{\breve{\boldsymbol{Q}}}\big(\boldsymbol{\Omega}(t),\ t\big)$ and $\ddot{\breve{\boldsymbol{Q}}}(t) \equiv \ddot{\breve{\boldsymbol{Q}}}\big(\boldsymbol{\Omega}(t),\ \dot{\boldsymbol{\Omega}}(t),\ t\big)$

The angular accelerations are calculated using the Euler equations:

$$\begin{cases} \dot{\Omega}_1^\bullet &= \big(M_{G1}^\bullet - (I_3^\bullet - I_2^\bullet)\,\Omega_2^\bullet\,\Omega_3^\bullet\big)/I_1^\bullet \\ \dot{\Omega}_2^\bullet &= \big(M_{G2}^\bullet - (I_1^\bullet - I_3^\bullet)\,\Omega_3^\bullet\,\Omega_1^\bullet\big)/I_2^\bullet \\ \dot{\Omega}_3^\bullet &= \big(M_{G3}^\bullet - (I_2^\bullet - I_1^\bullet)\,\Omega_1^\bullet\,\Omega_2^\bullet\big)/I_3^\bullet \end{cases} \tag{B.9}$$

However, it is important to notice that these relations are expressed in the principal axes of the considered block (*i.e.*, the framework defining the principal moments of inertia). This is recalled by the exponent $\bullet$ in the Eqs. (B.9). To express the resultant moment $\boldsymbol{M}_G$ and the angular velocity $\boldsymbol{\Omega}$ in the principal framework, it is necessary to apply a rotation opposite to the block orientation (given by a quaternion $\breve{\boldsymbol{Q}}$). To do so, the conjugate quaternion ($\text{conj}(\breve{\boldsymbol{Q}}) \equiv [\![s_Q \mid -\boldsymbol{i}_Q]\!]$) is used, *i.e.*, with an opposite sign of the imaginary part:

$$\boldsymbol{M}_G \xrightarrow{\text{conj}(\breve{\boldsymbol{Q}})} \boldsymbol{M}_G^\bullet \quad \text{and} \quad \boldsymbol{\Omega} \xrightarrow{\text{conj}(\breve{\boldsymbol{Q}})} \boldsymbol{\Omega}^\bullet \tag{B.10}$$

$$\dot{\boldsymbol{\Omega}}^\bullet \xrightarrow{\breve{\boldsymbol{Q}}} \dot{\boldsymbol{\Omega}} \tag{B.11}$$

References

[1] S. Roux, Quasi-static contacts, in: Physics of Dry Granular Media, Springer, 1998, pp. 267–284.
[2] J.-N. Roux, Geometric origin of mechanical properties of granular materials, Phys. Rev. E 61 (6) (2000) 6802.
[3] F. Radjaï, V. Richefeu, Contact dynamics as a nonsmooth discrete element method, Mech. Mater. 41 (6) (2009) 715–728.
[4] S. Van Baars, Discrete element modelling of granular materials, Heron 41 (2) (1996) 139–157.
[5] S. Luding, S. McNamara, How to handle the inelastic collapse of a dissipative hard-sphere gas with the TC model, Granular Matter 1 (1998) 113–128.
[6] J.J. Moreau, Sorne numerical methods in multibody dynamics: application to granular materials, Eur. J. Mech.-A/Solids 13 (4-suppl) (1994) 93–114.
[7] J. Fortin, G. de Saxcé, Modélisation numérique des milieux granulaires par l'approche du bipotentiel, Comptes Rendus de l'Académie des Sci. Ser. IIB-Mech. Phys. Astron. 327 (8) (1999) 721–724.
[8] P.A. Cundall, O.D.L. Strack, A discrete numerical model for granular assemblies, Geotechnique 29 (1) (1979) 47–65.
[9] P.A. Cundall, UDEC-A generalized distinct element program for modelling jointed rock, US Army European Research Office and Defence Nuclear Agency, Contract Report DAJA 37 (1980).
[10] F. Radjaï, Horizons in Dry Granular Modeling: beyond DEM, 2022, (https://hal.science/hal-04968358). IFPRI General Meeting, Jun 2022, Brussel, Belgium.
[11] L.F. Orozco, Modélisation Numérique Et RhÉologie Des Milieux Granulaires À Particules Fragmentables En Vue D'application Aux Broyeurs À Boulets, Phd thesis, Doctoral school ISS, 2019.
[12] L.F. Orozco, D.-H. Nguyen, J.-Y. Delenne, P. Sornay, F. Radjaï, Discrete-element simulations of comminution in rotating drums: effects of grinding media, Powder Technol. 362 (2020) 157–167.
[13] L.F. Orozco, J.-Y. Delenne, P. Sornay, F. Radjaï, Rheology and scaling behavior of cascading granular flows in rotating drums, J. Rheol. 64 (2020) 915–931.
[14] D.C. Vu, Simulation numérique discrète du broyage / mélange de poudres cohésives, Phd thesis, Doctoral school ISS, 2023.
[15] D.C. Vu, L. Amarsid, J. Delenne, V. Richefeu, F. Radjaï, Rheology and scaling behavior of polyhedral particle flows in rotating drums, Powder Technol. 434 (2024) 119338.
[16] T.D. Tran, Comportement mécanique d'agglomérats et leur aptitude à l'écoulement : approches expérimentale et numérique, Phd thesis, Doctoral school ISS, 2023.
[17] V. Richefeu, P. Villard, Modeling Gravity Hazards from Rockfalls to Landslides, ISTE press, 2016.
[18] V. Richefeu, G. Mollon, D. Daudon, P. Villard, Dissipative contacts and realistic block shapes for modeling rock avalanches, Eng. Geol. 149–150 (2012) 78–92.
[19] G. Mollon, V. Richefeu, P. Villard, D. Daudon, Numerical simulation of rock avalanches: influence of local dissipative contact model on the collective behavior of granular flows, J. Geophys. Res. 117 (2012) F02036.
[20] D. Daudon, P. Villard, G. Richefeu, V. Mollon, Influence of the morphology of slope and blocks on the energy dissipations in a rock avalanche, C.R. Mecanique 343 (2) (2014) 166–177.
[21] P. Bottelin, D. Jongmans, D. Daudon, A. Mathy, A. Helmstetter, V. Bonilla-Sierra, H. Cadet, D. Amitrano, V. Richefeu, L. Lorier, L. Baillet, P. Villard, F. Donzé, Seismic and mechanical studies of the artificially triggered rockfall at the mount néron (French alps, December 2011), Nat. Hazards Earth Syst. Sci. 14 (2) (2014) 3175–3193.
[22] G. Mollon, V. Richefeu, P. Villard, D. Daudon, Discrete modelling of rock avalanches: sensitivity to block and slope geometries, Granular Matter 17 (5) (2015) 645–666.
[23] Y.S. Cuervo, Modélisation des éboulements rocheux par la méthode des éléments discrets : application aux évènements réels, Ph.D. thesis, Doctoral school IMEP2, 2015.
[24] B. Garcia, V. Richefeu, J. Baroth, D. Daudon, P. Villard, Collision of shaped boulders with sand substrate investigated by experimental, stochastic and discrete approaches, JGR - Earth Surf. 125 (11) (2020).
[25] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C: the Art of Scientific Computing, Cambridge University Press, 1992.
[26] D.C. Vu, L. Amarsid, J. Delenne, V. Richefeu, F. Radjaï, Macro-elasticity of granular materials composed of polyhedral particles, Granular Matter 26 (6) (2024).
[27] M. Sonzogni, J.-M. Vanson, K. Ioannidou, Y. Reynier, S. Martinet, F. Radjaï, Dynamic compaction of cohesive granular materials: scaling behavior and bonding structures, Soft Matter 20 (2024) 5296. https://doi.org/10.1039/d3sm01116j
[28] B. Garcia, Analyse des mécanismes d'interaction entre un bloc rocheux et un versant de propagation : application à l'ingénierie, Phd thesis, Doctoral school IMEP2, 2019.
[29] R. Prat, L. Colombet, R. Namyst, Combining task-based parallelism and adaptive mesh refinement techniques in molecular dynamics simulations, in: Proceedings of the 47th International Conference on Parallel Processing, 2018, pp. 1–10.
[30] F. Radjaï, C. Voivret, Periodic boundary conditions, in: F.R. .F. Dubois (Ed.), Discrete-element Modeling of Granular Materials, Wiley-Iste, 2011, pp. 181–198. https://hal.science/hal-00690049.
[31] A. Argilaga Claramunt, FEMxDEM double scale approach with second gradient regularization applied to granular materials modelization, Theses, Université Grenoble Alpes, 2016. https://theses.hal.science/tel-01626295.
[32] J. Desrues, A. Argilaga, D. Caillerie, G. Combe, T.K. Nguyen, V. Richefeu, S. dal Pont, From discrete to continuum modelling of boundary value problems in geomechanics: an integrated FEM-DEM approach, Int. J. Numer. Anal. Methods Geomech. 43 (5) (2019) 919–955.
[33] S. Duverger, A multi-scale, MPMxDEM, numerical modelling approach for geotechnical structures under severe loading, Theses, Aix-Marseille Université (AMU), 2023. https://theses.hal.science/tel-04101270.
[34] O. Ozenda, G. Chambon, V. Richefeu, Multiscale MPMxDEM method for granular flows with cohesive non-spherical particles, applications to snow avalanches, 2022, (https://euromech.org/conferences/EMMC/EMMC18). 18th European Mechanics of Materials Conference (EMMC18), 4–6 april 2022, Oxford, UK.
[35] O. Ozenda, V. Richefeu, G. Chambon, P. Hagenmuller, A multiscale MPMxDEM model for simulating snowpack deformation and failure, 2023, (https://www.egu23.eu). EGU General Assembly 2023, 23–28 april 2023, Vienna, Austria.
[36] P. Mutabaruka, Modélisation Numérique Des Milieux Granulaires Immergés : Initiation Et Propagation Des Avalanches Dans Un Fluide, Ph.D. thesis, Doctoral school ISS, 2013.
[37] L. Amarsid, J.-Y. Delenne, P. Mutabaruka, Y. Monerie, F. Perales, F. Radjaï, Viscoinertial regime of immersed granular flows, Phys. Rev. E 96 (2017) 012901.
[38] P. Mutabaruka, J.-Y. Delenne, K. Soga, F. Radjaï, Initiation of immersed granular avalanches, Phys. Rev. E Stat. Nonlinear Soft Matter Phys. 89 (5) (2014) 052203.
[39] M. Stasiak, G. Combe, V. Richefeu, P. Villard, G. Armand, J. Zghondi, High compression of granular assemblies of brittle hollow particles. Investigation through a 3D discrete element model, Comp. Part. Mech. (2022). https://doi.org/10.1007/s40571-021-00449-3
[40] D.-C. Vu, L. Amarsid, J.-Y. Delenne, V. Richefeu, F. Radjaï, Particle fracture regimes from impact simulations, Phys. Rev. E 109 (4) (2024) 044907.
[41] S. Nezamabadi, F. Radjaï, J. Averseng, J.-Y. Delenne, Implicit frictional-contact model for soft particle systems, J. Mech. Phys. Solids 83 (2015) 72–87.
[42] S. Nezamabadi, X. Frank, J.-Y. Delenne, J. Averseng, F. Radjaï, Parallel implicit contact algorithm for soft particle systems, Comput. Phys. Commun. 237 (2019) 17–25.

[43] R. Prat, T. Carrard, L. Amarsid, V. Richefeu, C.-e. Doncecchi, P. Lafourcade, G. Latu, J.-M. Vanson, ExaDEM: a HPC application based on exaNBody targeting scalable DEM simulations with complex particle shapes, J. Open Source Softw. 10 (106) (2025) 7484. https://doi.org/10.21105/joss.07484

[44] T. Carrard, R. Prat, G. Latu, K. Babilotte, P. Lafourcade, L. Amarsid, L. Soulard, ExaNBody: a HPC framework for N-Body applications, in: European Conference on Parallel Processing, Springer, 2023, pp. 342–354.

[45] Yade Community, YADE – Open Source Discrete Element Method, 2024, (https://example.com). Version 2024-02-27.git-6fc6806.

[46] J.R. Dormand, P.J. Prince, New Runge-Kutta algorithms for numerical simulation in dynamical astronomy, Celestial Mech. 18 (3) (1978) 223–232.

[47] E. Fehlberg, Classical Seventh-, Sixth-, And Fifth-Order Runge-Kutta-NystrÖM Formulas With Stepsize Control For General Second-Order Differential Equations, Technical Report, Marshall Space Flight Center, AL: National Aeronautics and Space Administration, 1974.