Granular Matter (2009) 11:317–329
DOI 10.1007/s10035-009-0139-1

An efficient algorithm for granular dynamics simulations with complex-shaped objects

Fernando Alonso-Marroquín · Yucang Wang

Received: 29 June 2008 / Published online: 16 July 2009

Abstract One of the most difficult aspect of the realistic modeling of granular materials is how to capture the real shape of the particles. Here we present a method to simulate two-dimensional granular materials with complex-shaped particles. The particle shape is represented by the classical concept of a Minkowski sum, which permits the representation of complex shapes without the need to define the object as a composite of spherical or convex particles. A well defined interaction force between these bodies is derived. The algorithm for identification of neighbor particles reduces force calculations to $O(N)$, where N is the number of particles. The method is much more efficient, accurate and easier to implement than other models. We prove that the algorithm is consistent with energy conservation, which is numerically verified using non-dissipative granular dynamics simulations. Biaxial test simulations on dissipative granular systems demonstrate the relevance of shape in the strength and stress fluctuations at the critical state.

Keywords Granular systems · Dynamics and kinematics of rigid bodies · Molecular dynamics methods

F. Alonso-Marroquín (✉)
MoSCoS, School of Mathematics and Physics,
The University of Queensland, St Lucia,
Brisbane, QLD 4072, Australia
e-mail: fernando@physics.uq.edu.au

Y. Wang
CSIRO Exploration and Mining, Technology Court,
Pullenvale, QLD 4069, Australia

Y. Wang
PO Box 883, Kenmore, QLD 4069, Australia

1 Introduction

Rapid advances in computer simulations have led to many new developments in the modeling of particulate systems. These systems represent real physical systems at different scales, such as the small scale of biomolecules [1], geological scales of snow and debris flow [2], and the astronomical scales of the planetary rings dynamics [3]. Although particle shape plays an important role, most theoretical and numerical developments have been restricted to particles with spherical or circular shape. These simplification lead in some cases to unrealistic properties. In dissipative granular systems such as sand piles or fault gouges, disks of spheres tend to roll more easily than non spherical particles, leading to unrealistic angles of repose and very low bulk friction coefficients [4].

Four different approaches have been presented to model the real shape of particulate materials. In the first approach the shape is represented as a scalar functions. This methods can be used to represent specific shapes, such as ellipses [5], ellipsoids [6] and superquadric [7]. The main drawback of those methods is that the calculations required in the contact force are much more expensive than in spherical (or circular) particles.

In the second approach the non-spherical particle is represented by aggregates or clumps of disks and spheres bonded together [8,9]. In this approach crushing and fracture of aggregates can be easily modeled. The disadvantage is that a large number of spheres are required to represent a single particle.

The third approach is to represent the complex shape using polygons in 2D, [10–14] or polyhedrons in 3D [15,16]. Different methods are proposed to speed up simulations, such as linked cells [10], space triangulation [13,14] and contact plane technique [16]. The most difficult aspect for the simulations of these objects is the handling of contact interactions. Nowadays, the interaction is resolved by decomposing them in convex pieces, and applying penalty methods, impulse-based methods or dynamic constraints in the interaction between these pieces. Impulse-based formulations belong to the class of event-driven methods [17] They allow real-time simulations, but they cannot handle permanent or lasting contacts [15]. On the other hand, constraint methods, also known as contact dynamics methods, can handle lasting contacts with infinite stiffness [18]. However, simulations with contact dynamics are computationally expensive and lead in some cases to indeterminacy in the solution of contact forces [19]. This indeterminacy is removed by using penalty methods such as soft molecular dynamics [20], where the bodies are allowed to interpenetrate each other and the force is calculated in terms of their overlap. However, the determination of such contact force using polygons is still heuristic and lacks physical correctness, because the interactions do not comply with energy conservation [21].

A fourth approach to this problem for 3D simulations were propose by Pournin and Liebling using the mathematical concept of spheropolyhedrons [22,23]. These objects are generated from the Minkowski sum of a polyhedrons with an sphere, which is nothing more than the object resulting from sweeping an sphere around the polyhedrons. This simple concept can be used to generate very complex shapes, including non-convex bodies, without the need to decompose them into spherical or convex parts. An improved interaction model and shape-definition for complex shaped particles was presented in [24] using spheropolygons (i.e. the Minkowski sum between a polygon and a disk). By considering multiple contacts between the particles, our interaction model is different to the original Pournin & Liebling method [22,23], which assumed a single contact per particle pair. Some of the advantages and improvements involve (i) an elegant way to determine the contact (normal and tangential), (ii) the possibility to simulate non-convex particles, (iii) the enhancement of the stability and a reduction of the numerical errors in the energy conservation.

Here we present a detailed description of this method. In Sect. 2 we describe the basic components of the numerical model. In Sect. 3 we prove that the interaction model complies with the physical laws of conservative systems. Section 4 shows that the strength and the fluctuations at critical state of dense dissipative granular materials depend strongly on particle shape. Then in Sect. 5 we demonstrate that the model is much more efficient than the clump-of-disks models.

2 Model

Systems with different particle shapes are modeled using the concept of the Minkowski sum of a polygon (or polyline) with a disk. This mathematical operation is explained in Section 2.1. Section 2.2 deals with the numerical calculation of mass properties. The interaction between two particles is obtained from the individual interactions between each vertex of one particle and each edge of other, as explained in Sect. 2.3. In Sect. 2.4 we show that the number of floating point operations used to calculate interaction forces is drastically reduced by using a Verlet list and a contact list for each pair of neighbor particles. In Sect. 2.5 we present the architecture of the code used to implement the numerical model.

2.1 Minkowski sum

Given two sets of points P and Q in an Euclidean space, their Minkowski sum is given by $P + Q = \{\vec{x} + \vec{y} \mid \vec{x} \in P,\ \vec{y} \in Q\}$. This operation is geometrically equivalent to the sweeping of one set around the profile of the other without changing the relative orientation, see Fig. 1. A special case is the sum of a polygon with a disk, which is defined here as spheropolygon. Other examples of a Minkowski sum are the spherocyllinder (sphere + line segment) [25], the spherosimplex (sphere + simplex) [22] and the spheropolyhedron (sphere + polyhedron) [23], which are used in simulations of particulate systems.

The main advantage of the spheropolygons is that they allow us to represent any shape in 2D, from rounded to angular particles, and from convex to non-convex shapes. As we will see in Sect. 2.3, the Minkowski sum does not need to be explicitly calculated to determine the particle interaction. The mass properties are calculated numerically, but it does not affect much the simulation time because they are evaluated only at the beginning of the simulation.

2.2 Computational geometry

Before solving the dynamics of spheropolygons we need to introduce some useful concepts and algorithms. Given a

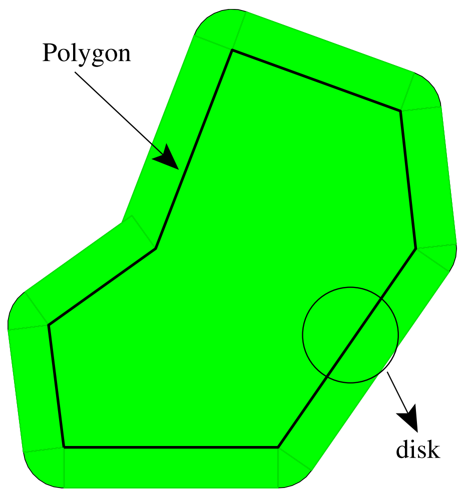

Fig. 1 Minkowski sum of a polygon with a disk

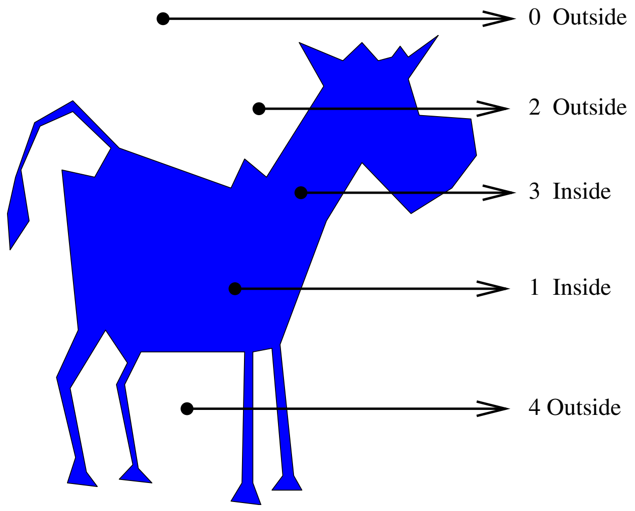

Fig. 2 Crossing test to determine whether a point is inside a polygon. The point is inside the polygon if the number of intersections of the ray with the polygon's edges is odd

spheropolygon $SP = P + S$, the polygon P will be called the skeleton; and the radius r of the disk S sphero-radius.

2.2.1 Point is inside test

An important numerical algorithm to calculate the mass properties of the spheropolygons is to determine whether a point lies on the interior of a polygon. Franklin [26] proposes a method to determine whether a point (x, y) is inside of a polygon, see Fig. 2. First, a semi-infinite ray is run horizontally (increasing x, fixed y) out from the test point, and count how many edges it crosses. At each crossing, the ray switches between inside and outside.

The essence of the ray-crossing method is as follows. Think of standing inside a field with a fence representing the polygon. Then walk west. If you need to jump the fence you know you are now outside the polygon. If you have to jump the fence again you know you are now inside again; i.e., if you were initially inside the field, the total number of fence jumps you would make will be odd, whereas if you were outside the jumps will be even. Then we can now write an outline of pseudo-code for this problem into Algorithm 1.

```
int crossings = 0;
for each line segment of the polygon do
    if ray down from (x,y) crosses segment then
        crossings++;
    end
end
if crossings is odd then
    return (inside);
else
    return (outside);
end
```

Algorithm 1: Algorithm to determine whether a point is inside of a polygon.

2.2.2 Point-inside-spheropolygon test

The distance $d(\vec{X}, P)$ from a test point $\vec{X}$ to the skeleton is defined as follows: If the point is outside the polygon, it is given by the minimum distance between the point and the edges of the polygon; If the point is inside the polygon, we let $d(\vec{X}, P) = 0$. Then the point $\vec{X}$ is inside of the spheropolygon when it satisfies $d(\vec{X}, P) < r$.

2.2.3 Calculation of the mass properties

The point-inside-spheropolygon test combined with a basic Monte Carlo method is used to evaluate the integral expressions for mass, center of mass and the moment of inertia. The numerical integration is performed by taking a quasi-random set of points $\vec{X}_i$ uniformly distributed in a rectangular box containing the spheropolygon. Then the integral over the area enclosed by the spheropolygon of any function $f(\vec{X})$ is calculated as:

$$\mathbf{M} = \int_{SP} f(\vec{X}) da \approx \frac{A_{\text{box}}}{N_p} \sum_{i=1}^{N_p} \chi(\vec{X}_i) f(\vec{X}_i), \qquad (1)$$

where A_{box} is the area of the rectangular box, $\vec{X}_i$ is a quasi-random point inside A_{box}, and $N_p =$ is the number of points. $\chi(\vec{X})$ is the characteristic function, which returns one if $\vec{X}$ is inside the spheropolygon and zero otherwise. Replacing $f(\vec{X})$ by $\sigma, \sigma\vec{X}\sigma||\vec{X}||^2$ results in $\mathbf{M} = m, m\vec{r}, I + m||\vec{r}||^2$ respectively, where σ is the density, and m, $\vec{r}$ and I are the mass, center of mass and moment of inertia.

2.3 Interaction force

To determine the interaction between spheropolygons we consider all vertex-edge distances between the polygons base. We consider two spheropolygons SP_i and SP_j with their respective polygons base P_i and P_j and sphero-radii r_i and r_j. Each polygon is defined by the set of vertices $\{V_i\}$ and edges $\{E_j\}$. The overlapping length between each pair of vertex-edge (V, E) is defined as

$$\delta(V, E) = \langle r_i + r_j - d(V, E) \rangle, \qquad (2)$$

where $d(X, E) = ||\vec{Y} - \vec{X}||$ is the Euclidean distance from the vertex V to the segment E. Here $\vec{X}$ is the position of the vertex V and $\vec{Y}$ is its closest point on the edge E. The ramp function $\langle x \rangle$ returns x if $x > 0$ and zero otherwise. The overlapping length in (2) is equivalent to the interpenetration between the disks of radii r_i and r_j centered on $\vec{X}$ and $\vec{Y}$.

The force $\vec{F}_{ij}$ acting on particle i by the particle j is defined by:

$$\vec{F}_{ij} = -\vec{F}_{ji} = \sum_{V_i E_j} \vec{F}(V_i, E_j) + \sum_{V_j E_i} \vec{F}(V_j, E_i), \qquad (3)$$

where $F(V, E)$ represent the force between the vertex V and the edge E. If the vertex-edge pair do not overlap, $F(V, E) = 0$. Different of vertex-edge forces can be included in the model: linear dashpots, non-linear Hertzian laws, damping forces proportional to the relative normal and tangential velocities, sliding friction, rolling resistance, cohesive forces, etc. The force of (3) is applied to each particle in the middle point of the overlap region between the vertex and the edge:

$$\vec{R}(V, E) = \vec{X} + \left(r_i + \frac{1}{2}\delta(V, E)\right) \frac{\vec{X} - \vec{Y}}{||\vec{Y} - \vec{X}||}, \tag{4}$$

so that the resulting torque on particle i given by j is

$$\tau_{ij} = \sum_{V_i E_j} \left(\vec{R}(V_i, E_j) - \vec{r}_i\right) \times \vec{F}(E_i, V_j) + \sum_{V_j E_i} \left(\vec{R}(V_j, E_i) - \vec{r}_i\right) \times \vec{F}(E_j, V_i), \tag{5}$$

where $\vec{r}_i$ is the center of mass of particle i.

The evolution of $\vec{r}_i$ and the orientation φ_i of the particle is governed by the equations of motion:

$$m_i \ddot{\vec{r}}_i = \sum_j \vec{F}_{ji} - m_i g \hat{y}, \quad I_i \ddot{\varphi}_i = \sum_j \tau_{ji}. \tag{6}$$

Here m_i and I_i are the mass and moment of inertia of the particle. The sum is over all particles interacting with this particle; g is the gravity; and $\hat{y}$ is the unit vector along the vertical direction.

2.4 Efficient calculation of forces

The efficiency of the dynamics simulation is mainly determined by the method of contact detection. In a system of N particles, each one with M edges, the number of operations required to update the positions of the particle in each time step is in the order of $O(NM)$, whereas the number of calculations for contact detection is $O(N^2M^2)$. Simulations therefore become very slow when either the number of particles or the number of vertices is large.

The first step to speed up the simulations is to execute the force calculation only over neighbor particles. With this aim we introduce the *Verlet list*, which is the collection of pair particles whose distance between them is less than 2δ (the distance between two particles is defined as the minimum of all vertex-edge distances). The parameter δ is equivalent to the *Verlet distance* used in simulations with spherical particles [21]. As shown in the Fig. 3, the Verlet method is equiv-

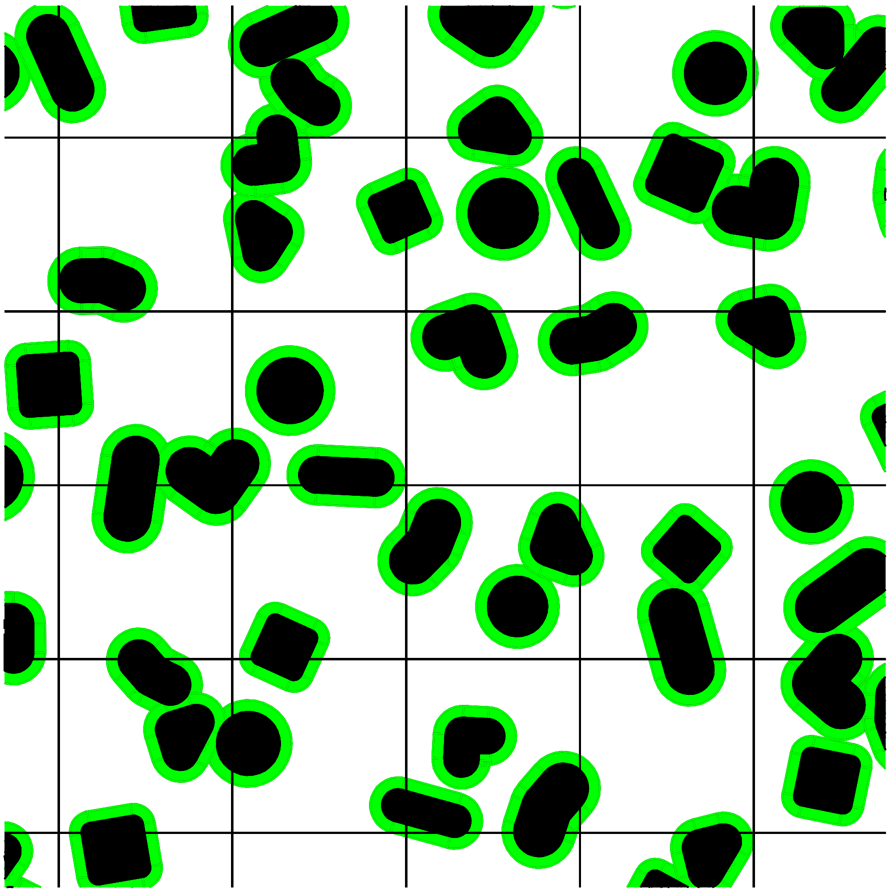
Fig. 3 Method for identification of Verlet list: the space domain is divided by square cells. Then the potential neighbor of the particles are those hosted in the same cells, or in the adjacent cells. Each particle has as *skin* of thickness δ. If the skins of two potential neighbors overlap, they are included in the neighbor list

alent to surround the particles by a *skin*, so that the Verlet list consists of all particles pairs whose skins overlap.

A *link cell* algorithm [21] is used to allow rapid calculation of this Verlet list: First, the space occupied by the particles is divided in cells of side $D + \delta$, where D is the maximal diameter of the particles. Then the link cell list is defined as the list of particles hosted in each cell. Finally, the candidates to being neighbors of a particle are searched only in the cell occupied by this particle, and its eight neighbor cells.

The Verlet list is calculated at the beginning of the simulation, and it is updated when the following condition is satisfied:

$$\max_{1 \le i \le N} \{\Delta x_i + R_i \Delta\theta_i\} > \delta. \tag{7}$$

Δx_i and $\Delta\theta_i$ are the maximal displacement and rotation of the particle after the last neighbor list update. R_i the maximal distance from the points on the particle to its center of mass. After each update Δx_i and $\Delta\theta_i$ are set to zero. The update condition is checked in each time step.

The efficiency of the simulations is very sensitive to the parameter δ. Increasing the value of δ makes updating of the list less frequent, but increases the size of the list, and hence the number of force calculations and the memory used by the simulation. Therefore, the parameter δ must be chosen by making a compromise between the storage (size of the neighbor list) and the computing time (frequency of list updates and number of force calculations).

The Verlet list reduces the amount of calculations to $O(NM^2)$, where M is the number of edges of the particles. Therefore the simulations are still very expensive when particles consist of a large number of vertices. Further reduction of the number of calculations between neighbor particles can be done by identifying which part of a particle is neighbor to the other. This idea is implemented as follows: for each element of the Verlet list, we create a *contact list*, which consists of those vertex-edge pairs whose distance between them is less than $r_i + r_j + 2\delta$, where r_i and r_j are the sphero-radii. In each time step, only these vertex-edge pairs are involved in the contact force calculations. Overall, neighborhood identification requires a Verlet list with all pair of neighbor particles, and one contact list for each pair of neighbors. These lists require little memory storage, and they reduce the amount of calculations of contact forces to $O(N)$. We therefore reduce the complexity of the algorithm to calculate contact forces at the same level as in simulations with circular particles.

2.5 Object-oriented implementation

The numerical model is implemented in C++ using an object-oriented design. The objects are organized in three classes: Particles, Interactions and Tags. The particle class contains the dynamics variables of the particle (position, orientation, mass, etc) and the information about its geometry (vertices of the skeleton and sphero-radius). The interaction class contains the variables of the elastic deformation at the contacts and the interaction parameters (stiffnesses, coefficients of friction and rolling, etc). The tag class store information about physical constraints on one particle, such as the external force imposed on the particle, or the value of the velocity fixed on the particle.

As shown in Fig. 4, three data structures are defined: (1) The particle generator contains a set of functions to generate the geometry of the particles, (2) The simulator updates the state of particles and interactions in each time step according to the interaction laws and physical constraints of the system. The simulator updates also the Verlet list when the neighbor update condition is satisfied, (3) The graphical interface use the data from the simulator to generate graphical output by writing either Matlab or Gnuplot instructions.

For the data structure of the simulator we use of the Standard Template Library (STL) of the C++ language [27]. The containers of this library provide us efficient methods to manage the objects of the model (particles, interactions, physical constrains and neighbor tables). Four containers are defined in the simulator:

- *Particle Array* contains a STL-list of particles. and a set of operators to access or modify their state.

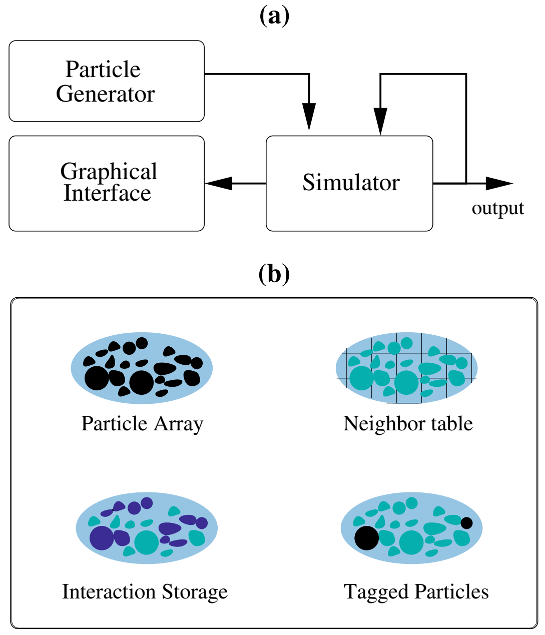

Fig. 4 *Top* data structure of the code. *Particle Generator* creates the particles and they are send to *Simulator*. There the particles are updated in each time step. The simulator allows access to the state the system by using an output of raw data, or by producing a graphical output using a *Graphical Interface*. *Bottom* The simulator consists of four containers to store the data of particles, neighborhood, interactions and physical constraints

- *Interaction Storage* allows access to the interactions between particles by means of key/value STL-map. The key consists of the indexes of the particles of the interaction. The value corresponds to the interaction itself, which contains the physical parameters of the contacts (deformation and stiffnesses). The interaction storage container uses a binary tree for rapid access to a contact given the indexes of the particles. It contains also a set of operators to access and modify the state of the interactions.
- *Tags List* stores a list of tags. Each tag contains the information required to impose physical constraints (fixed displacement or imposed force) to a particle. The Tags List contains also a set of operators to access or change the state of the tagged particles.
- *Neighbor table* contains the link cell structure, a list of all pointers to particles, and a list of the pointer of those particles marked as *big*. Walls are examples of *big particles* and they are not taken into account into the link cells. The neighbor table contains also a function which returns the list of all potential neighbors of a given particle. These potential neighbors include: (i) Particles living in the same cell, (ii) particles living in an adjacent cells, (iii) all big particles. The potential neighbors are used by the simulator to calculate the Verlet list.

To access the elements of these container, we use the powerful *for_each* operator. This operator takes as an argument a method of the class, and apply this method to all elements of the container. This is a very useful operator because it allows to access, process, or modify the elements without the need to write the *for-loops* of classical programming.

The main task of the simulator is to update the state of the system in each time step. The function *oneStep* uses a four order predictor-corrector integrator algorithm [21]. A pseudocode with the basic procedures of this function is shown in Algorithm 2. The predictor method calculates the position (center of mass and orientation) of each particle and its derivatives using a Taylor expansion. Next, the vertices of the all polygons are updated according to the predicted positions of the particles. If the neighbor update condition of (7) is satisfied, the link cell is calculated, and then it is used to update the Verlet list and the contact list of each one of its elements. Next the contact forces and torques are calculated. Then force and/or position of the tagged particles is modified according to the constraints prescribed in the Tags list. Finally, forces and torques are used to correct the position of the particles and their derivatives. The algorithm is basically the same as that used with polygons [20], except that the force is calculated using (3) and (5). Note that the efficiency of the code is based on the simplicity of the contact force calculation, and in the fact that the Minkowski sum does not need to be calculated during the time integration.

```
Input: state of the simulator at time $t$
Output: state of the simulator at time $t + \Delta t$
ParticleArray.for_each(predictor);
if neighbor update condition is satisfied then
    NeighborTable.update(ParticleArray);
    calculate Verlet list;
    insert and delete interaction;
    update contact lists;
end
ParticleArray.for_each(update vertices);
ParticleArray.for_each(set Force to zero);
ParticleArray.for_each(set Torque to zero);
InteractionStorage.for_each(update contacts);
InteractionStorage.for_each(calculate Forces);
ParticleArray.applyGravity(gravity);
TagsList.for_each(set Force);
TagsList.for_each(set Torque);
TagsList.for_each(set Velocity);
TagsList.for_each(set Angular Velocity);
ParticleArray.for_each(corrector);
```

Algorithm 2: One time step in the Simulator.

3 Non-dissipative granular dynamic simulations

As a first example we present a model for many body conservative systems. We introduce a simple contact force which includes only elastic repulsion between spheropolygons. We first prove that the system is conservative, i.e. the energy of the system does not change with time. Then we use many-body simulations with different particle shape to evaluate the discretization error in the energy conservation.

3.1 Energy balance

The elastic vertex-edge interaction between the particles is obtained by using the following force in (3) and (5):

$$\vec{F}(V, E) = k\delta(V, E)\vec{N}, \tag{8}$$

where k is the stiffness constant, $\delta(V, E)$ is given by (2) and $\vec{N}$ is the unit normal vector:

$$\vec{N} = \frac{\vec{Y} - \vec{X}}{||\vec{Y} - \vec{X}||} \tag{9}$$

Here $\vec{X}$ is the position of the vertex V and $\vec{Y}$ is its closest point on the edge E.

The question which now arises is whether the vertex-edge interaction in (8) leads to a conservative system. Let us multiply the first equation in (6) by $\dot{\vec{r}}$ and the second one by $\dot{\varphi}$. Next we sum both equations and then sum over all particles. After some algebra we get the energy conservation equation:

$$E_T = \sum_{ij} E^e_{ij} + \sum_i \left(\frac{1}{2} m_i v_i^2 + \frac{1}{2} I_i \omega_i^2\right) = cte. \tag{10}$$

The first term of this equation corresponds to the potential elastic energy at the contacts:

$$E^e_{ij} = \frac{1}{2}k\left(\sum_{V_i E_j} \delta(V_i, E_j) + \sum_{V_j E_i} \delta(V_j, E_i)\right). \qquad (11)$$

The other terms of (10) are the linear and rotational kinetic energy of the particles. Therefore we have proven that the elastic force in (8) belongs to the potential energy defined by (11), which proves that our model is conservative. The simplicity of this force contrasts to the Pöschel's model for interacting triangles [21], where the forces and torques associated to his potential energy lead to a much more expensive calculation.

3.2 Discretization error

Other important aspect of this dynamics simulation is the accuracy of the numerical solution. The numerical error in the energy calculation is evaluated by performing a series of simulations with many non-spherical particles interaction via the elastic force given by (8). Each test consists of 400 particles confined by four fixed rectangular walls with a volume fraction of $\Phi = 0.186$. Each sample consists of particles with a specific spheropolygon-like shape as shown Fig. 5: disks (point + disks) rice (line + disk), peanuts (polyline + disk), and pebbles (triangle + disk). The dimensionless parameters of the simulations are a constant stiffness $k = 1$, density $\sigma = 1$, time step $\Delta t = 0.05$ and Verlet distance $\delta = 0.11$.

Initially, each particle has zero angular velocity and a linear velocity of 1 with random orientation. Due to collisions, the linear momentum of each particle changes and part of it is transferred to angular momentum. Fig. 6 shows the potential (a) kinetic (b) and total (c) energy of the system. Elastic energy has a negligible contribution to the energy budget, as it differs from zero only for short times during collisions. Energy conservation is numerically verified within a relative error of 0.01%. The energy fluctuations are produced

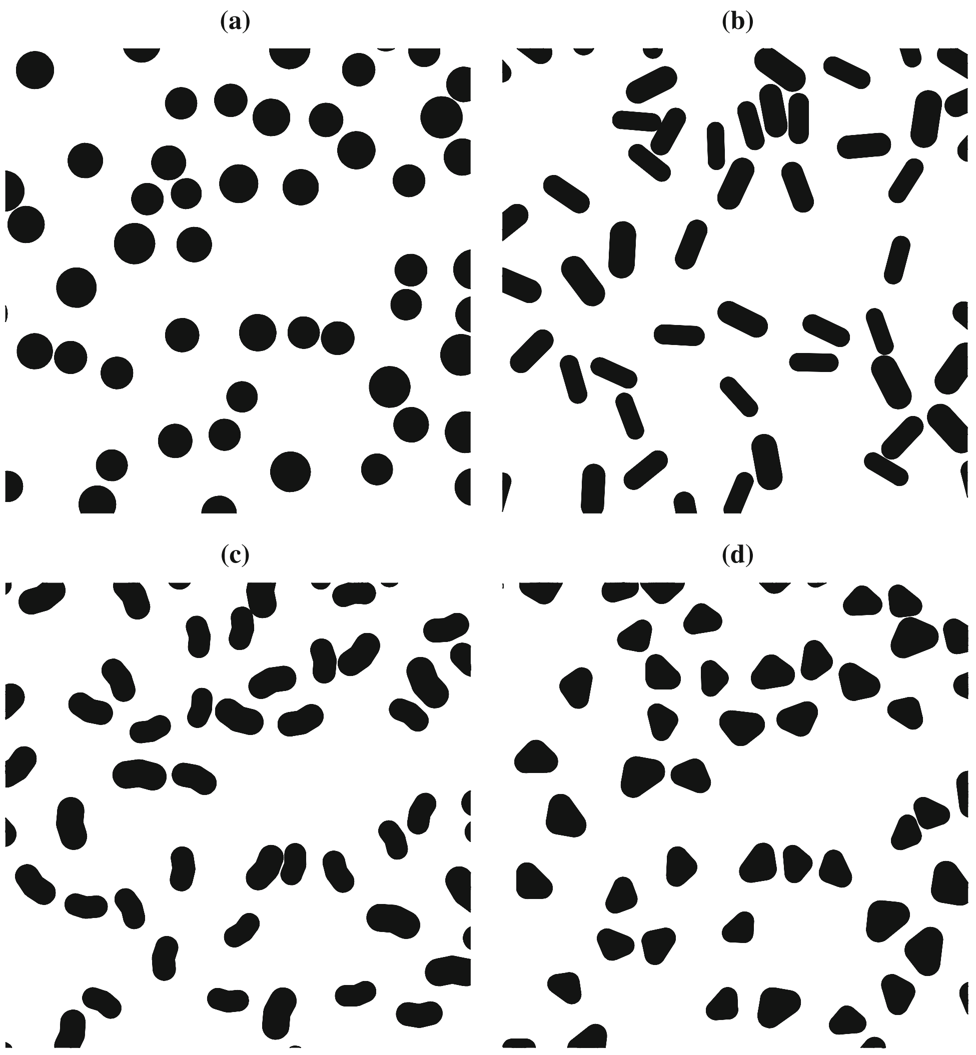

Fig. 5 Systems obtained from Minkowski sum approach: **a**)disks (Vertex + disk). **b** rice (segment + disk), **c** peanuts (Polyline + disk) and **d** pebbles (triangle + disk). The particles are generated with a uniform distribution of areas between A_{min} and A_{max}. The polydispersity factor is chosen as $(A_{max} - A_{min})/(A_{max} + A_{min}) = 0.1$

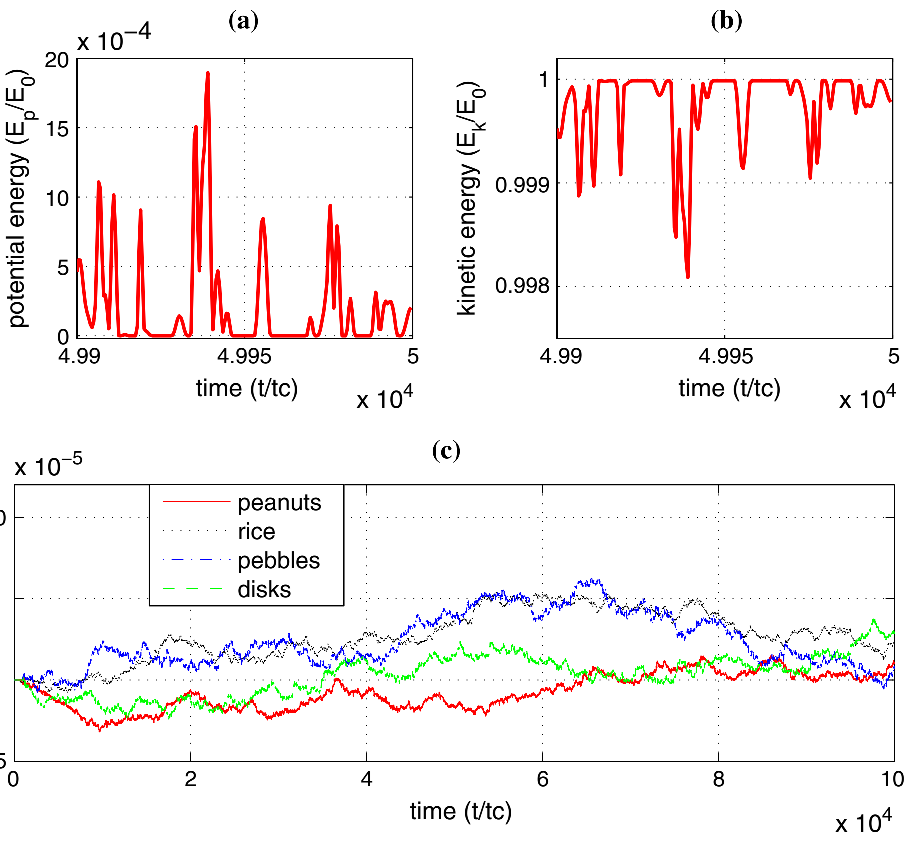

Fig. 6 Time evolution of the total kinetic and potential energy in the non-dissipative system. As is expected from the energy conservation, the total energy keeps constant, except numerical error which are lower than 0.01% throughout all the simulations

by time discretization. We shall note also that energy has a trend to grow slowly in all samples, which seems to be inherent from most of the integration methods.

4 Dissipative granular dynamics simulation

In this section we present numerical results with dissipative granular materials. These systems are simulated by using the following force in (3) and (5):

$$\vec{F}(V,E) = k_n\delta_n\vec{N} + k_t\delta_t\vec{T} \tag{12}$$

Where $\vec{N}$ is the normal unit vector given by (9). The tangential vector $\vec{T}$ is taken perpendicular to $\vec{N}$. The overlapping length δ_n is defined in (2). The elastic displacement δ_t takes into account frictional forces, and it satisfies the sliding condition $|F_t| < \mu F_n$, where μ is the coefficient of friction [28]. The parameters of the simulations are the normal stiffness $k = 1$, the tangential stiffness $k_t = 0.1k_n$, friction coefficient $\mu = 0.5$, density $\sigma = 1$, time step $\Delta t = 0.05$ and the Verlet distance $\alpha = 0.1$. Viscosity forces proportional to relative velocity of the contacts are also included to allows relaxation of the system.

We start with a loose packing of particles confined in a rectangular space by four walls. First we set the friction coefficient to zero. Then we confine the sample by applying a constant pressure (force per unit of length) on the four walls. The resulting packings for disks, rice, peanuts and pebbles are shown in Fig. 7. The densest packings are reached with rice and pebbles (packing fractions around 0.88). peanuts have an intermediate packing fraction of 0.87. Whereas the disk exhibits the lowest packing fraction of 0.80. The main reason of the high packing fraction of rice is that the particles tend to align, leading to edge-edge contacts with no space between them. Peanuts tend to align too, but they reach less packing fraction due to the non-convexity of the particles.

We can now test the stress-strain response of the packings by performing biaxial tests simulations. First the friction coefficient is set to the original value and the sample is compressed until it reached a pressure value $p_0 = 0.001k_n$. Then the lateral walls are subjected to constant pressure by applying an horizontal force $F_{left} = p_oH$, where H is the height of the sample. The horizontal walls are moved towards each other with constant velocity. This velocity is chosen small enough to avoid time-dependency effects. The axial strain is defined as $\epsilon = \Delta H/H_0$, H_0 being the initial height of the sample. the stress at the top wall is given by $\sigma_{yy} = F_{top}/W$, where F_{top} is the force imposed on the top wall by the particles, and W is the width of the sample. The void ratio is calculated as $\nu = (A - A_p)/A_P$, where $A = HW$ is the area of the sample and A_p is the total area of the particles.

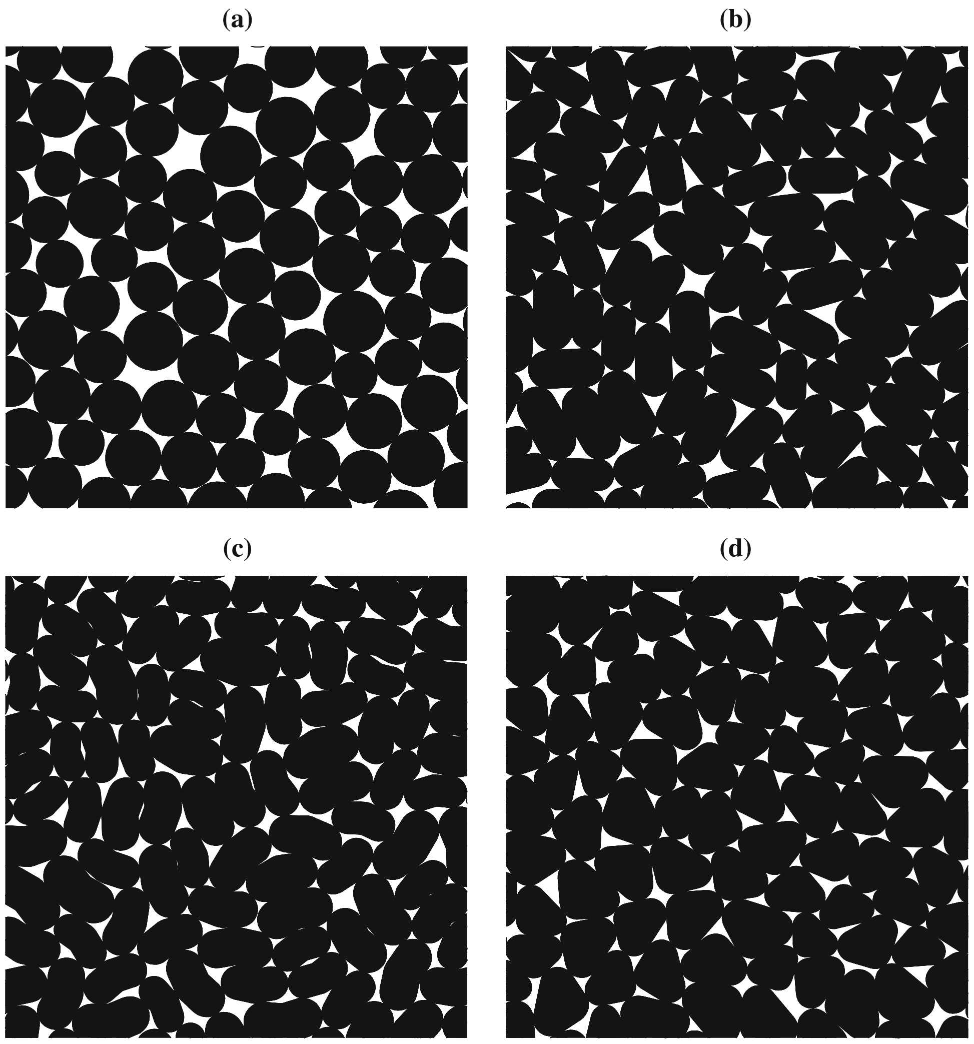

Fig. 7 Granular packing obtained with disks, rice, peanuts and pebbles

Figure 8 shows the evolution of the stress and the void ratio during the biaxial test. At the beginning of the simulation, the stiffness (i.e. the slope of the stress-strain curve) decreases as the strain increases. The void ratio first decreases until it reaches a minimum. Then the void ratio starts to increase and it reaches the inflexion point when the stress peaks. After this peak the void ratio keeps increasing and it reaches a saturation level. These are key aspects of the response of two-dimensional analogue granular material [29]. For large deformations the stress and the void ratio reach a constant value, which corresponds to the critical-state theory of soil mechanics [30]. Note that the limit stress for non-circular particles is much larger than that one of circular particles, reflecting an important effect of particle shape.

Another interesting aspect of the simulations is the emergence of stress fluctuations. These fluctuations are given by stress drops accompanied by a sudden dilation of the sample. Stress drops are pronounced and infrequent for non-circular particles, whereas they are less pronounced and more frequent for circular particles. There are some experimental evidence on stress fluctuations in torsional test with sand [31]. However, the stress drops are only visible in coarser materials such as beds of glass spheres [32]. Recent numerical simulations with circular [33,34] and polygonal particles [4] shows that the stress drops involves a small number of particles along few force chains. This may be the reason of why stress drops are not observed in fine materials. A sample of sand may contain million of particles. Therefore each stress drop involves few hundreds of particles along few chains, which may be difficult to observe in experimental tests. This is different for samples containing glass spheres, which contains only few thousands of particles. Therefore the collapse of a single force chain on the packing of glass spheres must have a relevant effect in the whole sample. The stress fluctuations depend also on particle size distribution. In our simulations the particles are about the same size, which leads to important stress drops. This may not be necessary true for polidisperse media. Indeed, previous simulation on lattices with breakable bond show that stress drops decrease as the disorder increases [35]. We therefore

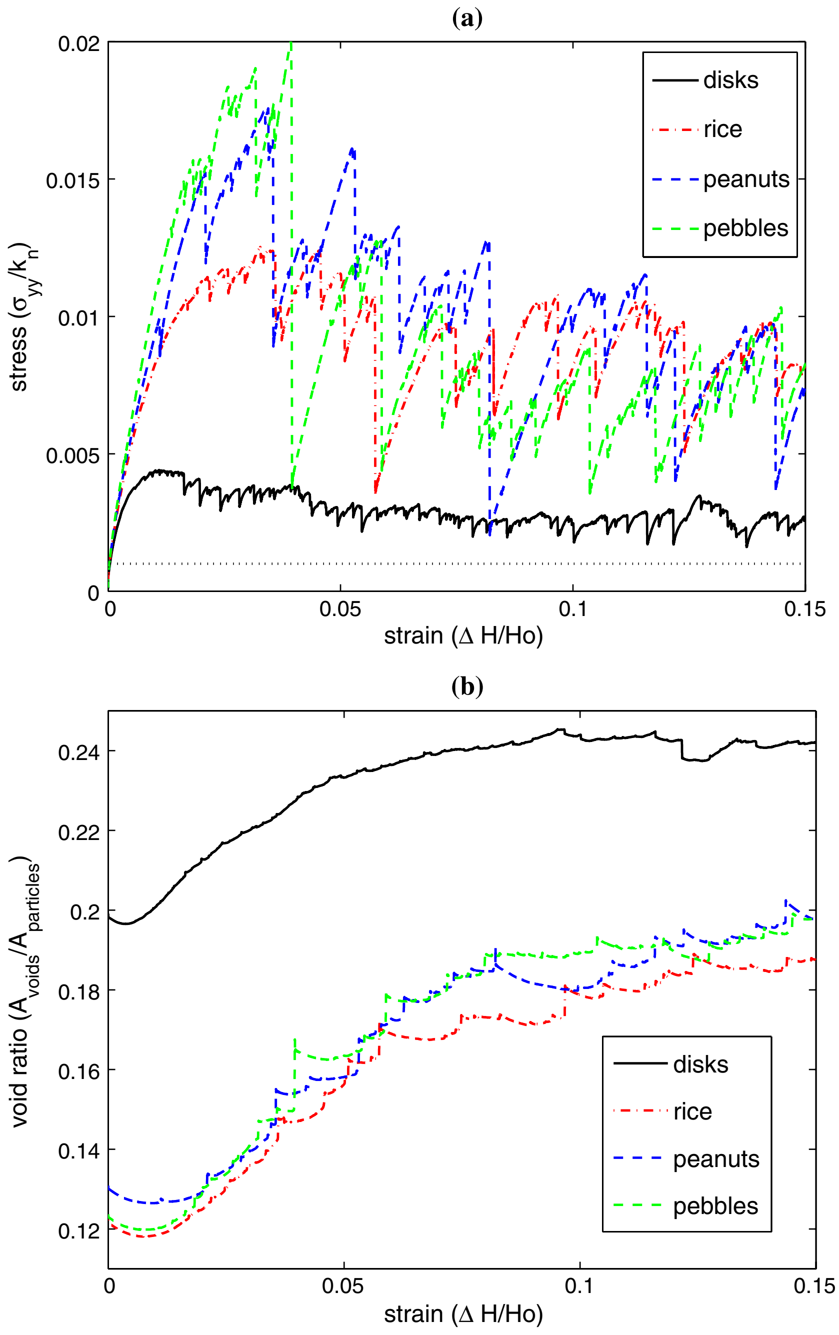

Fig. 8 Stress versus strain (**a**) and void ration versus strain (**b**) for different particle shapes, in biaxial test simulations.The dotted line in (**a**) represents p/k_n, where p is the applied pressure on the lateral walls

conclude that the dynamics of the stress drops depends both on particle shape and the size distribution of the granular assembly.

The mechanism of collapse of force chains is still not clear too. The buckling of force chains has been proposed [34]. However, our simulations with rice particles lead to stress drops without buckling of force chains [36]. Further numerical simulations should be done to understand better the mechanism of stress collapse and its relevance on the overall response of real granular materials

5 Performance

In this section we compare the efficiency of the many-body simulations of systems consisting on circular particles, spheropolygons and particles consisting on clumps of spheres. Each spheropolygon corresponds to a particle with complex shape, namely a Minkowski cow [24], and it consists on 62 vertices. The clump of spheres represents the same complex shaped particle, (a clumpy cow) and it needs 726 particles. The macroparticles are simulated by summing up

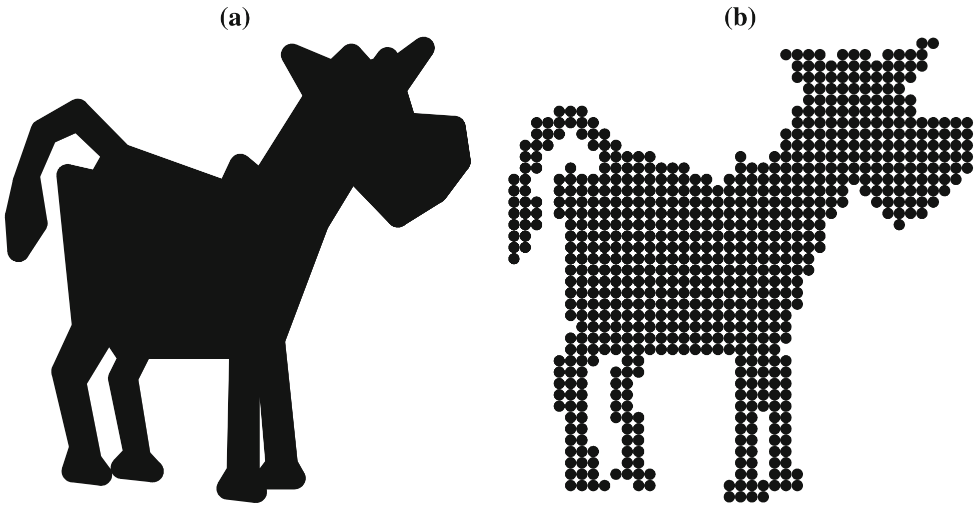

Fig. 9 **a** Minkowski cow obtained by sweeping a disk around a polygon of 62 edges. The disk has radius 10 cm and the length of the polygon is 3 m. **b** The cow as represented by a clump of disks using 726 disks

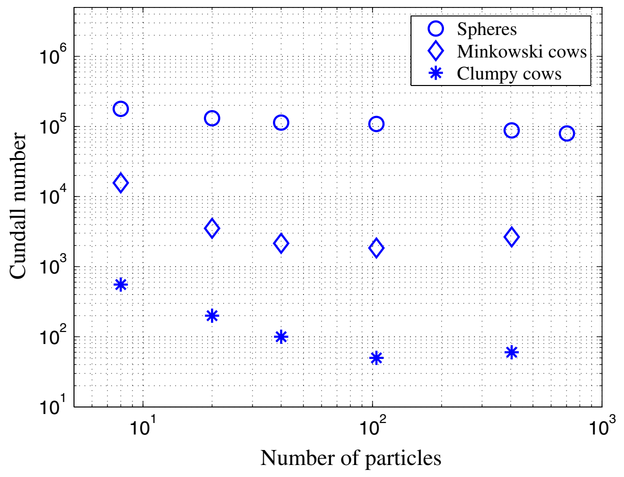

Fig. 10 Cundall number versus the number of particles, in simulations with disks, spheropolygons and clumpy particles

the contact forces between the constituting disks, and updating all the disks of each particle according with its current position (Fig. 9). The performance of the simulations is estimated by running different processes in a Pentium 4, 3.0GHz, and calculating the *Cundall number* [37] for each one of them. This number is the amount of particle time steps executed by the processor in one second, which is calculated as $c = N_T N / T_{CPU}$, where N_T is the number of time steps, N is the number of particles and T_{CPU} is the CPU time of the simulation. While the Cundall number provides a method to compare efficiency of various DEM algorithms, it has the disadvantage that it depends on the processor and the compiler.

Figure 10 shows the Cundall number versus the number of particles for the three cases. When the number of particles is between $N = 100$ and $N = 1000$ the Cundall number is approximately constant. This constant is around 100, 000 for disks, 2, 000 for spheropolygons, and 50 for clumpy particles. Therefore the simulations with spheropolygons, although slower than simulations with disks, are much faster than simulations with clumpy particles. This is because each time step needs to update the position of 62 vertices of the spheropolygons whereas it needs to update the position of the 726 disks in the case of the clumpy particles. Therefore simulations with spheropolygons are more efficient than those ones with clumps of disks, because the former ones require less elements to represent the particle shape.

6 Concluding remarks

We have presented a method based on the Minkowski sum approach to simulate conservative and dissipative interactions between 2D complex shaped particles. The method allows simulations of arbitrarily shaped particles and multiple contact interactions between each pair of particles. The implementation of the method follows the philosophy of Object Oriented Programming. Based on the C++ STL, we introduced the concept of encapsulated containers, which are used to handle particles, interactions and physical constraints. Benchmark tests proved high accuracy in the energy conservation for simulations of non-dissipative systems, and the emergence of the critical state with stick-slip fluctuations for dissipative granular systems under shear deformation.

There is a fundamental difference in the modelling of the interaction with spheropolygons with respect to the interaction of polygons. In the first case, forces are calculated in terms of vertex-edge distances. In the latter case, forces are calculated in terms of the overlap between polygons. There are two main reasons to use spheropolygons instead of polygons: (1) The elastic force used in spheropolygons comes from a potential, so that this model provides an equation for energy balance. Therefore spheropolygons provide a solution of the long standing problem of energy balance of simulations with polygons, where elastic forces, when calculating in terms of overlap, proves to be non-conservative; (2) The calculation of the vertex-edge distances involves less floating point operations than calculations of overlap areas, which makes the simulations with spheropolygons more efficient than with polygons.

We also proved that the particle shape representation using the Minkowski sum approach is more efficient than the clump-of-disks method. This is because the amount of data required to represent the particle shapes using Minkowski sum is lower than that used in the clump-of-circles method. Benchmark tests show that the simulation with spheropolygons is 40 times faster that simulations with clumps of disks.

To achieve efficient simulations we presented a technique to identify neighborhood, which combines the Linked Cell Method (LCM) with an extension of the Verlet List method for non-spherical particles. In order to further speed up the simulations we introduced a list of potential vertex-edge contacts for each pair of neighboring particles, and an *update condition*, which decides whether the neighbor list needs to be updated. The algorithm presents an $O(N)$ complexity, where N is the number of particles.

Future work will require a comparison of the efficiency of our method with other algorithms with linear complexity, such as the Linked Linear List (LLL) [38,39] and the triangulation method [13,14]. In the Linked Linear List algorithm each particle is enclosed by a rectangular box. Then two particles are neighbor when their boxes overlap. LLL and LCM are recommended for systems with large number of particles, but LLL are more efficient than LCM for polydisperse packings [39]. In the method of triangulation, each particle is covered by spheres. Then a triangulated mesh is constructed with the centers of those spheres. The elements of the mesh are used to determine the neighborhood between particles. An important advantage of the triangulated meshes is that they can be used for the discretization of continuum equations embedded to DEM, providing methods to simulate multiface flows and to couple particles with fluids.

The extension of our interaction model for 3D spheropolyhedrons is straightforward: Instead of vertex-edge interaction in 2D, we need to include all vertex–face and edge–edge interactions.

$$\vec{F}_{ij} = \sum_{E_i,E_j} \vec{F}(E_i, E_j) + \sum_{V_i,F_j} \vec{F}(V_i, F_j) + \sum_{V_j,F_i} \vec{F}(V_j, F_i). \qquad (13)$$

Where E, V and F account for edges, vertex and faces. The rotational dynamics of the spheropolyhedrons can be solved by the numerical solution of the Euler equations using quaternions, which can be found in the literature [40]

The motivation for modelling complex shaped particles in 3D using spheropolyhedrons is to overcome the difficulties of simulating interacting polyhedrons, where the calculation of forces in terms of overlap is far from trivial. The method can also be used for irregular spheropolyhedrons, which can provide a wider range of particle shape representations, including non-convex particles and particles with tunable grain roundness. These features can be included with polyhedrons, but with an exhaustive number of features, and hence with a huge computational cost.

Acknowledgments We thank Syed Imran for technical support; R. Cruz-Hidalgo and K. Steube for review of an early version of the manuscript; A.J. Hale and M. L. Kettle for writing corrections; and S. Latham, E. Heesen, P. Mora, W. Hancock, P. Clearly, S. Luding, and S. McNamara for discussions. This work is supported by the Australian Research Council and the University of Queensland Early Carrier Research Grant.

References

1. Carlson, H., McCammon, A.: Accommodating protein flexibility in computational drug design. Mol. Pharmacol. **57**(2), 213–218 (2000)
2. Darve, F., Laouafa, F.: Instabilities in granular materials and application to landslides. Mech. Cohesive-Frictional Mater. **5**, 627–652 (2000)
3. Pöschel, T., Luding, S.: Granular Gases. Springer, Berlin (2000)
4. Alonso-Marroquin, F., Vardoulakis, I., Herrmann, H.J., Weatherley, D., Mora, P.: Effect of rolling on dissipation in fault gouges. Phys. Rev. E **74**, 031306 (2006)
5. Ting, J., Khwaja, M., Meachum, L., Rowell, J.: An ellipse-based discrete element model for granular materials. Int. J. Numer. Anal. Methods Geomech. **17**(9), 603–623 (1993)
6. Lin, X., Ng, T.: A three-dimensional discrete element model using arrays of ellipsoids. Geotechnique **47**(2), 319–329 (1997)
7. Mustoe, G., Miyata, M.: Material flow analyses of noncircular-shaped granular media using discrete element methods. J. Eng. Mech. **127**(10), 1017–1026 (2001)
8. Cheng, Y., Nakata, Y., Bolton, M.: Discrete element simulation of crushable soil. Geotechnique **53**(7), 633–641 (2003)
9. McDowell, G.R., Bolton, M.D., Robertson, D.: The fractal crushing of granular materials. J. Mech. Phys. Solids **44**(12), 2079–2102 (1996)
10. Alonso-Marroquin, F., Herrmann, H.J.: Calculation of the incremental stress-strain relation of a polygonal packing. Phys. Rev. E **66**, 021301 (2002)
11. Matuttis, H.-G., Luding, S., Herrmann, H.J.: Discrete element methods for the simulation of dense packings and heaps made of spherical and non-spherical particles. Powder Technol. **109**, 278–292 (2000)
12. Mirghasemi, A.A., Rothenburg, L., Matyas, E.L.: Influence of particle shape on engineering properties of assemblies of two-dimensional polygon-shaped particles. Geotechnique **52**(3), 209–217 (2002)
13. Müller, D., Liebling, T.M.: Detection of collisions of polygons by using a triangulation. In: Contact Mechanics, Proceedings of the 2nd Contact Mechanics International Symposium, pp. 369–372. Carry-le-Rouet, France (1994)
14. Müller, D.: Techniqes informatiques efficaces pour la simulation de milieux granulaires pare des méthodes d'éléments distinct, thesis no. 1545, Ph.D. thesis, École Polytechnique Fédérale de Lausanne (1996)
15. Mirtich, B.: V-clip: fast and robust polyhedral collision detection. ACM Trans. Graph. (TOG) **15**(3), 177–208 (1998)

16. Cundall, P.: Formulation of a three-dimensional distinct element model–Part I. A scheme to detect and represent contacts in a system composed of many polyhedral blocks. Int. J. Rock Mech. Min. Sci. Geomech. Abstr **25**(3), 107–116 (1988)
17. McNamara, S., Luding, S.: Energy flows in vibrated granular media. Phys. Rev. E **58**, 813–822 (1998)
18. Jean, M.: The non-smooth contact dynamics method. Comput. Methods Appl. Mech. Eng. **177**(3–4), 235–257 (1999)
19. McNamara, S., García-Rojo, R., Herrmann, H.: Indeterminacy and the onset of motion in a simple granular packing. Phys. Rev. E **72**(2), 21304 (2005)
20. Alonso-Marroquin, F., Luding, S., Herrmann, H., Vardoulakis, I.: Role of the anisotropy in the elastoplastic response of a polygonal packing. Phys. Rev. E **51**, 051304 (2005)
21. Poeschel, T., Schwager, T.: Computational Granular Dynamics. Springer, Berlin (2004)
22. Pournin, L., Liebling, T.: A generalization of distinct element method to tridimentional particles with complex shapes. In: Powders & Grains 2005, pp. 1375–1478. Leiden, Balkema (2005)
23. Pournin, L.: On the behavior of spherical and non-spherical grain assemblies, its modeling and numerical simulation. Ph.D. thesis, École Polytechnique Fédérale de Lausanne (2005)
24. Alonso-Marroquin, F.: Spheropolygons: a new method to simulate conservative and dissipative interactions between 2d complex-shaped rigid bodies. Europhys. Lett. **83**(1), 14001 (2008)
25. Pournin, L., Weber, M., Tsukahara, M., Ferrez, J.-A., Ramaioli, M., Liebling, T.M.: Three-dimensional distinct element simulation of spherocylinder crystallization. Granul. Matter **7**(2–3), 119–126 (2005)
26. Franklin, R.: How do I find if a point lies within a polygon. In: comp.graphics.algorithms FAQ, www.faqs.org/faqs/graphics/algorithms-faq/, p. Subject 2.03 (1994)
27. Josuttis, N.: The C++ Standard Library: A Tutorial and Reference. Addison-Wesley Professional, USA (1999)
28. García-Rojo, R., Alonso-Marroquín, F., Herrmann, H.: Characterization of the material response in granular ratcheting. Phys. Rev. E **72**(4), 41302 (2005)
29. Sibille, L., Froiio, F.: A numerical photogrammetry technique for measuring microscale kinematics and fabric in Schneebeli materials. Granul. Matter **9**(3), 183–193 (2007)
30. Wood, D.M.: Soil Behaviour and Critical State Soil Mechanics. ISBN: 0-521-33782-8, Cambridge (1990)
31. Vardoulakis, I., Georgopoulos, I.O.: The stress-dilatancy hypothesis revisited: shear-banding related instabilities. Soils & Foundations **45**, 61–76 (2005)
32. Adjemian, F., Evesque, P., Jia, X.: Ultrasonic experiment coupled with triaxial test for micro-seismicity detection in granular media. In: García-Rojo, R., Herrmann, H., McNamara, S. (eds.) Powders and Grains 2005, pp. 281–285. Leiden, Balkema (2005)
33. Pena, A., Lizcano, A., Alonso-Marroquin, F., Herrmann, H.J.: Biaxial test simulations using a packing of polygonal particles. Int. J. Numer. Anal. Meth. Geomech. **32**(2), 143 (2008)
34. Tordesillas, A.: Force chain buckling, unjamming transitions and shear banding in dense granular assemblies. Philos. Mag. **87**(32), 4987–5016 (2007)
35. Herrmann, H., Hansen, A., Roux, S.: Fracture of disordered, elastic lattices in two dimensions. Phys. Rev. B **39**(1), 637–648 (1989)
36. Tordesillas, A., Alonso-Marroquin, F.: On the connection between aspect ratio and rolling resistance (in preparation)
37. Cleary, P.W., Sinnott, M.D., Morrison, R.D.: DEM prediction of particle flows in grinding processes. Int. J. Numer. Meth. Fluids **58**, 319–353 (2008)
38. Muth, B., Eberhard, P., Luding, S.: Collisions between particles of complex shape. In: Powders & Grains 2005, pp. 1379–1383. Leiden, Balkema (2005)
39. Muth, B., Muller, M., Eberhard, P., Luding, S.: Collision detection and administration methods for many particles with different sizes. In: DEM07 Proceedings CD, pp. 1–18. www2.msm.ctw.utwente.nl/sluding/PAPERS/dem07.pdf
40. Wang, Y., Abe, S., Latham, S., Mora, P.: Implementation of particle-scale rotation in the 3-D lattice solid model. Pure Appl. Geophys. **163**(9), 1769–1785 (2006)