Granular Matter (2013) 15:85–93
DOI 10.1007/s10035-012-0381-9

ORIGINAL PAPER

Polyhedral particles for the discrete element method

Geometry representation, contact detection and particle generation

Benjamin Nassauer · Thomas Liedke · Meinhard Kuna

Received: 10 July 2012 / Published online: 1 November 2012

Abstract The geometry of convex polyhedra is described by a set of half spaces. This geometry representation is used in the discrete element method to model polyhedral particles. An algorithm for contact detection and the calculation of the interaction forces for these particles is presented. Finally the presented model is exemplified by simulating the particle flow through a hopper.

Keywords DEM · Convex polyhedral particles · Half space representation · LIGGGHTS

1 Introduction

The discrete element method (DEM) is a powerful numerical tool for simulating the mechanical behaviour of systems with a large number of particles. It is successfully used in a wide range of applications like granular flow, powder mechanics or rock mechanics. The focus of the method is on the motion and the interaction of particles, so the real physical particles, ranging from micro-scale powders up to large rock formations, are represented as rigid geometric bodies. Here, the usage of spherical particles is quite common. Whereas simulations with spherical particles can include millions of particles, using non-spherical particles is still not an easy task. But because the particle shape has a strong influence on the mechanical behaviour there is a strong need to do so. Whereas for spherical particles the geometry is described by the radius and the interaction forces can easily be calculate by contact laws like Hertzian contact, for non-spherical particles the geometry representation and calculation of contact forces is much more complex. One way to handle non-spherical particles is the use of super-quadrics [1] or superellipsoids [2]. This approach has the advantage to describe the geometry of the particle by a single equation. Nevertheless the range of geometries that can be modelled is limited. Another approach is the glued-sphere method [3,4]. In this method the geometry of the particle is represented by several spheres which are glued together as a rigid body. This method has the disadvantage, that for a good representation of complex geometries large numbers of spheres are needed. It also is not possible to model sharp edges by this method. However for simulating abrasive processes of particles on workpieces like grinding, polishing, wear, indentation fracture or crushing, the modeling of sharp edges is indispensable. A detailed description of different geometry representations can be found in [5]. For polyhedral particles several approaches can be found in the literature, e.g. [6–9]. The algorithms used in these approaches require complex geometrical and algebraic operations for contact detection, some of which involve even iterative procedures (e.g. [9]) or distinction of special cases (e.g. [8]). Moreover, a complete determination of the geometry of the overlapping region is not provided in any of these methods. Therefore only rather simple contact force models can be used, based on the interpenetration depth between the particles.

In the present paper an approach for modelling convex polyhedra by a set of half spaces is presented. The convex polyhedra are a good representation of the geometry of a variety of granulate materials [10]. On the basis of this geometry representation a fast and straight forward algorithm for contact detection is developed. This contact detection method has the advantage of not only finding the contact very fast, but also providing a complete description of the shape of the overlapping region. This description is than used to calculate the interaction forces.

B. Nassauer (✉) · T. Liedke · M. Kuna
Institute for Mechanics and Fluiddynamics, TU Bergakademie Freiberg, 09596 Freiberg, Germany
e-mail: Benjamin.Nassauer@imfd.tu-freiberg.de

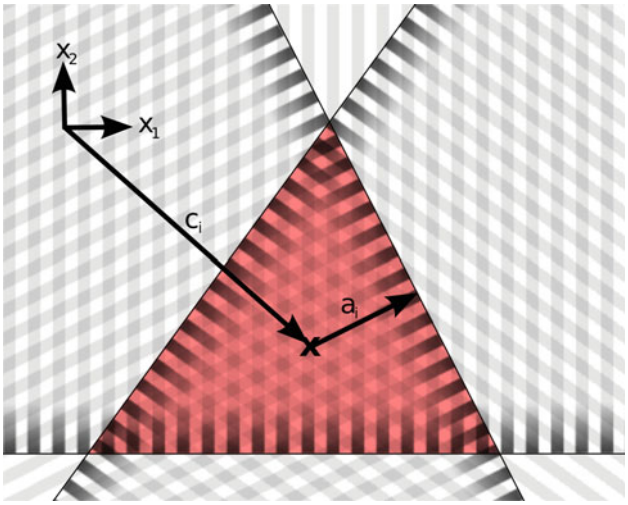

Fig. 1 Representation of a convex particle (*red*) at position c_i by half-spaces a_i (2D simplification) (color figure online)

2 Geometry representation

Lee et al. [11] have suggested a method to represent convex particles by a set of half spaces. The particle is defined as the space that belongs to all half spaces, as illustrated in Fig. 1 schematically for two dimensions. In the following part index notation and Einstein's summation convention will be used, i.e. the subscript index denotes the spatial direction. This doesn't include superscript indices, which are only used to specify the variables in this paper.

Each half space is described by an equation of the form

$$(x_i - c_i - a_i)a_i \leq 0 \tag{1}$$

where c_i is the center of mass of the particle, a_i is a vector from the center of mass to the boundary face in normal direction and x_i is an arbitrary point inside the halfspace. In a local coordinate system located at the center of mass of the particle Eq. (1) reduces to

$$(x_i - a_i)a_i \leq 0. \tag{2}$$

All further geometrical information can be calculated from this database as described in the following.

The first step is to find possible corner points of the polyhedron. This is done by calculating the intersection points of all combinations of three boundary faces out of all faces. We consider the set of half spaces to be given in a coordinate system $\tilde{x}_i$, with the origin inside the particle. With the vectors of the three faces $\tilde{a}_i^j$, where the superscript j indicates the number of the used plane, this leads to the system of equations

$$\begin{aligned}(\tilde{x}_i - \tilde{a}_i^1)\tilde{a}_i^1 &= 0\\ (\tilde{x}_i - \tilde{a}_i^2)\tilde{a}_i^2 &= 0\\ (\tilde{x}_i - \tilde{a}_i^3)\tilde{a}_i^3 &= 0,\end{aligned} \tag{3}$$

which can be arranged to

$$\begin{bmatrix}\tilde{a}_1^1 & \tilde{a}_2^1 & \tilde{a}_3^1\\ \tilde{a}_1^2 & \tilde{a}_2^2 & \tilde{a}_3^2\\ \tilde{a}_1^3 & \tilde{a}_2^3 & \tilde{a}_3^3\end{bmatrix}\begin{pmatrix}\tilde{x}_1\\ \tilde{x}_2\\ \tilde{x}_3\end{pmatrix} = \begin{pmatrix}\tilde{a}_i^1\tilde{a}_i^1\\ \tilde{a}_i^2\tilde{a}_i^2\\ \tilde{a}_i^3\tilde{a}_i^3\end{pmatrix}. \tag{4}$$

The expressions in the right hand side vector are scalar products. This Eq. (4) is solved for all face combinations.

In the second step the points have to be checked whether they belong to all half spaces. Only in this case they are corner points of the polyhedron.

To calculate geometrical characteristics like volume, mass or moments of inertia the faces have to be triangulated using the afore calculated corner points. Each triangle forms a tetrahedron with the center of mass and thus the calculation becomes trivial. In order to perform the triangulation of a face, the points $\tilde{p}_i^j$ of the face have to be ordered in the following way:

- First it is checked, whether the number of points $\tilde{p}_i^j$ of the face is greater than three. Otherwise no ordering will be performed.
- The first point in the unordered list is defined as the second point in the ordered list $\tilde{p}_i^{2o} = \tilde{p}_i^1$.
- Then the points $\tilde{p}_i^\alpha$ and $\tilde{p}_i^\beta$ are found that maximize the angle ϕ between $\tilde{p}_i^{2o}$, $\tilde{p}_i^\alpha$ and $\tilde{p}_i^\beta$. This means

$$\cos\phi = \frac{\alpha_i\beta_i}{|\alpha_i||\beta_i|} \tag{5}$$

with

$$\begin{aligned}\alpha_i &= \tilde{p}_i^\alpha - \tilde{p}_i^{2o}\\ \beta_i &= \tilde{p}_i^\beta - \tilde{p}_i^{2o}\end{aligned} \tag{6}$$

has to be minimized.
- With this the first and third ordered point are defined as $\tilde{p}_i^{1o} = \tilde{p}_i^\alpha$ and $\tilde{p}_i^{3o} = \tilde{p}_i^\beta$.
- For the remaining points, when n points are already sorted, the point $\tilde{p}_i^j$ has to be found that minimizes Eq. (5) when α_i and β_i are defined as

$$\begin{aligned}\alpha_i &= \tilde{p}_i^{(n-1)o} - \tilde{p}_i^{no}\\ \beta_i &= \tilde{p}_i^j - \tilde{p}_i^{no}.\end{aligned} \tag{7}$$

Then the volume V, the surface A and the center of mass $\tilde{c}_i$ of the polyhedron can be calculated as

$$V = \sum_{k=1}^{m} V^k = \sum_{k=1}^{m} \frac{1}{6}\left|\left(\epsilon_{ikl}\tilde{p}_k^{k1}\tilde{p}_l^{k2}\right)\tilde{p}_i^{k3}\right| \tag{8}$$

$$A = \sum_{k=1}^{m} \frac{1}{2}\left|\epsilon_{ikl}\left(\tilde{p}_k^{k3} - \tilde{p}_k^{k2}\right)\left(\tilde{p}_l^{k1} - \tilde{p}_l^{k2}\right)\right| \tag{9}$$

$$\tilde{c}_i = \frac{1}{V}\sum_{k=1}^{m} V^k \tilde{c}_i^k, \tag{10}$$

where m is the number of tetrahedrons respective triangles. $\tilde{p}_i^{k1}$, $\tilde{p}_i^{k2}$ and $\tilde{p}_i^{k3}$ are the corner points of the tetrahedron respective triangle k. ϵ_{ikl} is the Levi-Cevita symbol and $\epsilon_{ijkl}a_k b_l$ denotes the cross product of the vectors a_k and b_l. V^k and $\tilde{c}_i^k$ are defined as

$$V^k = \frac{1}{6}\left|\left(\epsilon_{ikl}\tilde{p}_k^{k1}\tilde{p}_l^{k2}\right)\tilde{p}_i^{k3}\right| \tag{11}$$

$$\tilde{c}_i^k = \frac{1}{4}\left(\tilde{p}_i^{k1} + \tilde{p}_i^{k2} + \tilde{p}_i^{k3}\right). \tag{12}$$

2.1 Calculation of moments of inertia

The moments of inertia of a polyhedron can be calculated by the following algorithm, which is based on [12]. Before the moments of inertia are calculated, the points and planes are transferred into a new coordinate system $\overline{x}_i$ with the origin in the center of mass of the particle.

$$\overline{p}_i = \tilde{p}_i - \tilde{c}_i \tag{13}$$

$$\overline{a}_i = \tilde{a}_i\left(1 - \frac{\tilde{a}_i\tilde{c}_i}{\tilde{a}_i\tilde{a}_i}\right) \tag{14}$$

The moments of inertia of a body with constant density ρ are given with respect to the coordinate system $\overline{x}_i$ as

$$\overline{I}_{ij} = \rho\int_V r^2\delta_{ij} - \overline{x}_i\overline{x}_j \mathrm{d}V, \tag{15}$$

where δ_{ij} is the Kronecker-delta and r^2 is defined as

$$r^2 = \overline{x}_1^2 + \overline{x}_2^2 + \overline{x}_3^2. \tag{16}$$

A tensor function F_{ijh} is introduced such that

$$r^2\delta_{ij} - \overline{x}_i\overline{x}_j = f_{ij}(\overline{x}_n) = \frac{\partial}{\partial\overline{x}_h}F_{ijh}(\overline{x}_n). \tag{17}$$

This enables to transform Eq. (15) by Gauss's theorem into

$$\overline{I}_{ij} = \rho\int_A \overline{n}_h F_{ijh}\mathrm{d}A, \tag{18}$$

where $\overline{n}_i$ is the surface normal. The surface integral is split up by the triangulation.

$$\overline{I}_{ij} = \rho\sum_{k=1}^{m}\int_{A^k} \overline{n}_h^k F_{ijh}^k(\overline{x}_h)\mathrm{d}A, \tag{19}$$

From Eq. (17) the function $F_{ijh}(\overline{x}_h)$ is derived as

$$\begin{aligned}
F_{11h}(\overline{x}_i) &= \left(0\ \tfrac{\overline{x}_2^3}{3}\ \tfrac{\overline{x}_3^3}{3}\right)\\
F_{22h}(\overline{x}_i) &= \left(\tfrac{\overline{x}_1^3}{3}\ 0\ \tfrac{\overline{x}_3^3}{3}\right)\\
F_{33h}(\overline{x}_i) &= \left(\tfrac{\overline{x}_1^3}{3}\ \tfrac{\overline{x}_2^3}{3}\ 0\right)\\
F_{12h}(\overline{x}_i) &= F_{21h}(\overline{x}_i) = \left(\tfrac{\overline{x}_1^2\overline{x}_2}{2}\ 0\ 0\right)\\
F_{13h}(\overline{x}_i) &= F_{31h}(\overline{x}_i) = \left(0\ 0\ \tfrac{\overline{x}_3^2\overline{x}_1}{2}\right)\\
F_{23h}(\overline{x}_i) &= F_{32h}(\overline{x}_i) = \left(0\ \tfrac{\overline{x}_2^2\overline{x}_3}{2}\ 0\right).
\end{aligned} \tag{20}$$

Considering the corners of the triangle $\overline{p}_i^1$, $\overline{p}_i^2$ and $\overline{p}_i^3$, the vectors s_i and t_i are defined as

$$\begin{aligned}
s_i &= \overline{p}_i^2 - \overline{p}_i^1\\
t_i &= \overline{p}_i^3 - \overline{p}_i^1.
\end{aligned} \tag{21}$$

Then the surface A^k can be parametrized as

$$A^k(u,v) = \overline{x}_i(u,v) = \overline{p}_i^1 + us_i + vt_i. \tag{22}$$

The infinitesimal surface $\mathrm{d}A$ can be transformed to

$$\mathrm{d}A = \left|\frac{\partial A^k}{\partial u}\times\frac{\partial A^k}{\partial v}\right|\mathrm{d}u\mathrm{d}v = |s_i\times t_i|\mathrm{d}u\mathrm{d}v. \tag{23}$$

Finally the mass moment of inertia of the whole polyhedron can be calculated using Eq. (15) which now reads

$$\overline{I}_{ij} = \rho\sum_{k=1}^{m}\int_0^1\int_0^{1-v}\overline{n}_h^k F_{ijh}^k(\overline{x}_i(u,v))|s_i\times t_i|\mathrm{d}u\mathrm{d}v. \tag{24}$$

Alternative ways of calculating the inertia are described in the literature, e.g. [13].

2.2 Coordinate transformation

The corner points $\overline{p}_i$ and plane vectors $\overline{a}_i$ have to be transformed into the local coordinate system of the particle, which is defined by the center of mass c_i and the principal axes of inertia. The translation of the coordinate system into the center of mass has already been performed before the calculation of the moments of inertia. Now the coordinate system $\overline{x}_i$ will be rotated into the coordinate system x_i which is oriented in the direction of the principal axes of inertia.

Given the eigenvectors e_i^j of the inertia tensor, a rotation matrix is defined as

$$r_{ij} = \begin{bmatrix}
\frac{e_1^1}{|e_i^1|} & \frac{e_1^2}{|e_i^2|} & \frac{e_1^3}{|e_i^3|}\\
\frac{e_2^1}{|e_i^1|} & \frac{e_2^2}{|e_i^2|} & \frac{e_2^3}{|e_i^3|}\\
\frac{e_3^1}{|e_i^1|} & \frac{e_3^2}{|e_i^2|} & \frac{e_3^3}{|e_i^3|}
\end{bmatrix}. \tag{25}$$

Then the points and planes are rotated.

$$p_i = r_{ij}\overline{p}_j \tag{26}$$

$$a_i = r_{ij}\overline{a}_j. \tag{27}$$

It would be sufficient to store the half spaces of each particle, as all further geometric information can be derived from them every time they are needed. Nevertheless, all other afore mentioned information are stored too in order to save calculation time.

2.3 Representation of global orientation by quaternions

As mentioned before two coordinate systems are used for the calculation. The first coordinate system is a global coordinate system, fixed in space. Besides the global coordinate system every particle has its own local body-fixed coordinate system which is located in the center of mass and is in the direction of the principal axes of inertia. During simulation several variables have to be transferred from one coordinate system to the other. The position of the local coordinate system in the global coordinate system is defined by the global coordinates of the center of mass of the particle. The orientation of the local coordinate system with respect to the global one can be expressed by quaternions.

Quaternions are four dimensional vectors. If they are used to describe rotations the first number describes the angle of the rotation and the other three numbers describe the rotation axes. The normalised quaternion $q_i = (q_1, q_2, q_3, q_4)^T$ for a rotation around an axes ω_i (vector of unity) by an angle α is obtained by

$$q_i = \left(\cos\frac{\alpha}{2}, \omega_1 \sin\frac{\alpha}{2}, \omega_2 \sin\frac{\alpha}{2}, \omega_3 \sin\frac{\alpha}{2}\right)^T. \tag{28}$$

A rotation matrix R_{ij} can be derived from the quaternions as

$$R_{ij} = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2q_2q_3 - 2q_1q_4 & 2q_1q_3 + 2q_2q_4 \\ 2q_2q_3 + 2q_1q_4 & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2q_3q_4 - 2q_1q_2 \\ 2q_2q_4 - 2q_1q_3 & 2q_1q_2 + 2q_3q_4 & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix}. \tag{29}$$

Then the coordinates x_i^g of a point in the global system can be transformed into the local coordinates x_i^l by

$$x_i^l = R_{ji}(x_j^g - c_j), \tag{30}$$

and the other way round by

$$x_i^g = c_i + R_{ij}x_j^l. \tag{31}$$

For vectors v_i like the force or velocity, which only describe a direction and are not fixed to a point in space, the translation of the coordinate systems can be omitted. So Eqs. (30) and (31) reduce to a pure rotation

$$v_i^l = R_{ji}v_j^g \tag{32}$$

and

$$v_i^g = R_{ij}v_j^l. \tag{33}$$

3 Contact detection

For contact detection the overlapping region is defined as the region belonging to all half spaces of the two polyhedra in contact, [11]. The same algorithm as for the description of the particles can be applied. This way an exact geometric representation of the overlapping region is obtained, which can be used for the calculation of the interaction forces. A separate treatment of the different contact types, like corner to face, edge to edge or face to face, is not necessary. In the following the contact algorithm is described in detail.

- To each particle j a radius r^j is assigned, which defines a sphere completely surrounding the polyhedron. In a first step of the contact algorithm it is checked whether these spheres are in contact. This means the inequality

  $$|c_i^2 - c_i^1| < r^1 + r^2 \tag{34}$$

  has to be fulfilled, where c_i^j are the centers of the particles. Otherwise the contact detection is terminated.
- In order to speed up the algorithm all corner points of one particle are checked whether they are close enough to the other particle for being in contact. This is performed by the following algorithm, which is visualized in Fig. 2:

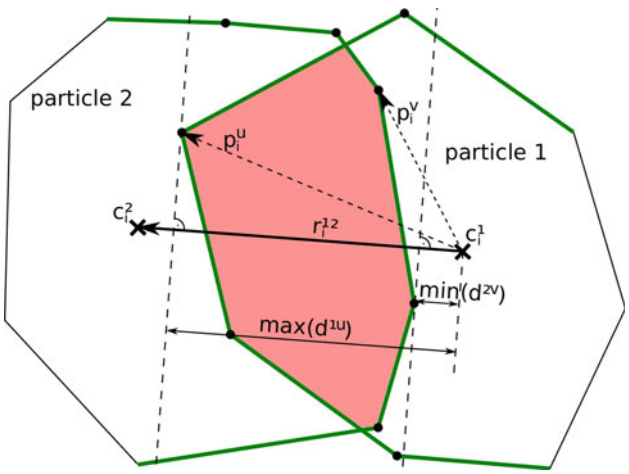

Fig. 2 An area (*dotted lines*) is defined which encloses the overlapping region (*red*). All points (*black*) and planes (*green*) of the polyhedra are taken into account for contact calculation. All other points and planes are excluded from the calculation (2D simplification) (color figure online)

- First the vector between the centers of the particles r_i^{12} is calculated.

$$r_i^{12} = c_i^2 - c_i^1 \tag{35}$$

- The distance d^{1u} of a corner u of particle 1 to the center of particle 1 in direction of r_i^{12} is

$$d^{1u} = \left| \frac{r_i^{12} p_i^u}{r_i^{12} r_i^{12}} \right|, \tag{36}$$

where p_i^u is the vector from the center of particle 1 to the corner u.

- In the same way the distance d^{2v} of a corner v of particle 2 to the center of particle 1 in direction of r_i^{12} reads

$$d^{2v} = \left| \frac{r_i^{12} p_i^v}{r_i^{12} r_i^{12}} \right|, \tag{37}$$

where p_i^v is the vector from the center of particle 1 to the corner v.

- All corners of particle 1 that fulfill the inequality

$$d^{1u} > min(d^{2v}) \tag{38}$$

are considered valid.

- All corners of particle 2 that fulfill the inequality

$$d^{2v} < max(d^{1u}) \tag{39}$$

are considered valid.

- If no valid corners are found the contact detection is terminated.

– All boundary planes of the particles that contain at least one valid corner are considered as valid for contact detection.
– New points are found by the intersection of two planes of one particle and one plane of the other particle. All possible combinations of valid planes have to be considered. The system of equations for one combination reads

$$\begin{bmatrix} a_1^{1g} & a_2^{1g} & a_3^{1g} \\ a_1^{2g} & a_2^{2g} & a_3^{2g} \\ a_1^{3g} & a_2^{3g} & a_3^{3g} \end{bmatrix} \begin{pmatrix} x_1^g \\ x_2^g \\ x_3^g \end{pmatrix} = \begin{pmatrix} \left(c_i^1 + a_i^{1g}\right) a_i^{1g} \\ \left(c_i^2 + a_i^{2g}\right) a_i^{2g} \\ \left(c_i^3 + a_i^{3g}\right) a_i^{3g} \end{pmatrix}, \tag{40}$$

where a_i^{jg} are the vectors of the faces in global coordinates and c_i^j is the center of the particle the face belongs to.
– The points belonging to the overlapping region have to be found.

- All corners of particle 1 are checked whether they are inside particle 2.
- All corners of particle 2 are checked whether they are inside particle 1.
- All new points are checked whether they are inside particle 1 and particle 2.

– All planes that include at least 3 points of the overlapping region are part of the region.
– Volume and center of mass of the overlapping region can be calculate as described above for the particles. For this purpose a new coordinate system is introduced that results from translating the coordinate system of the first particle to a point c_i^o inside the overlapping region, that is defined by

$$c_i^o = \frac{1}{n} \sum_{w=1}^{n} p_i^{ow}, \tag{41}$$

where p_i^{ow} are the corners of the overlapping domain and n is the number of valid corners.

4 Interaction forces

In order to calculate contact forces one has to specify the magnitude, the application point and the direction of the force [14]. The interpenetration of two particles can be correlated to the volume V of the overlapping region. The force application point is simply located at the center of mass of the overlapping region. To calculate the force direction n_i^f, the surface normal n_i^A of the overlapping region is integrated over the part of the surface that belongs to one of the particles, see Fig. 3.

$$n_i^f = \frac{\int_A n_i^A ds}{|\int_A n_i^A ds|} = \frac{1}{\sum_j A^j} \sum_j A^j n_i^{Aj} \tag{42}$$

Additionally, the relative velocity v_i^{rel} of the particles at the load application point is considered, which is defined by the velocity v_i and the rotation of the particles. The relative velocity is divided in a normal v_i^{reln} and tangential v_i^{relt} component, where v_i^{reln} is parallel to n_i^f. Based on this the normal force f_i^n and tangential force f_i^t can be calculated as

$$f_i^n = \underbrace{k^n V n_i^f}_{elastic\ force} \underbrace{-k^{nd} v_i^{reln}}_{viscous\ damping} \tag{43}$$

$$f_i^t = \underbrace{\mu |f_i^n| \frac{v_i^{relt}}{|v_i^{relt}|} \left(1 - \frac{0.1}{0.1 + |v_i^{relt}|^2}\right)}_{friction\ force} \underbrace{-k^{td} v_i^{relt}}_{viscous\ damping}, \tag{44}$$

with k^n, k^{nd}, μ and k^{td} being the constants for the elastic force, viscous damping in normal direction, friction and viscous damping in tangential direction, respectively. The term in parenthesis of the friction force just has the function to

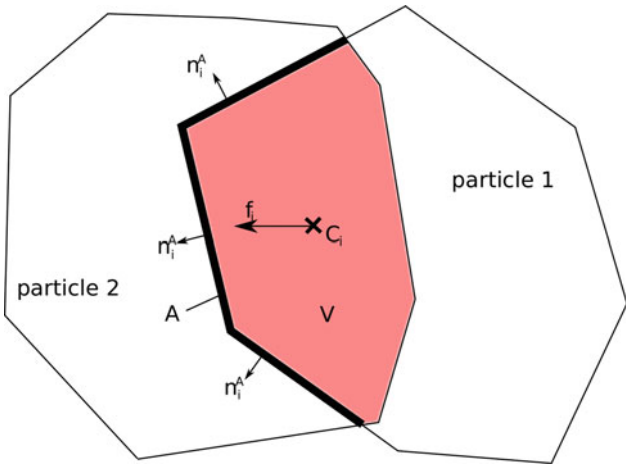

Fig. 3 The overlapping region (*red*) is used as the database for the calculation of the contact forces. The force application point is defined by the center of mass c_i, the magnitude is definde by the volume V and the direction is defined by integrating the surface normal n_i^A over the surface belonging to particle 1 (*bold black line*) (2D simplification) (color figure online)

avoid a jump in the friction force at zero velocity. Other formulations are possible. The total force is

$$f_i = f_i^n + f_i^t. \tag{45}$$

If l_i is a vector from the center of the particle to the force application point, then the torque M_i is

$$M_i = \epsilon_{ijk} l_j f_k. \tag{46}$$

It has to be mentioned, that the assumed equations for calculating the forces are just a first simple suggestion, which can be improved in future work.

5 Random particle generation

In order to get realistic randomly shaped particles a primary solid, usually a cube, is trimmed by random cut planes for several times. The creation process consists of three steps, first the orientation of the cut plane is defined, second the distance of a plane which touches the outermost corner is calculated and third the cut plane is defined by scaling the distance vector by a random value (see Fig. 4). The procedure is based on the approach of [10], and will be described in more detail in the following.

First a normalized, randomly oriented vector n_i^c with origin in the center of the particle is created. The orientation of this vector is defined by two angles ψ and ϕ in a spherical coordinate system. The values of ψ and ϕ are chosen by chance. The probability distribution functions for these angles can be used to influence the shape of the resulting particle. In order to make all orientations of the vector n_i^c equiprobable, a uniform probability distribution function

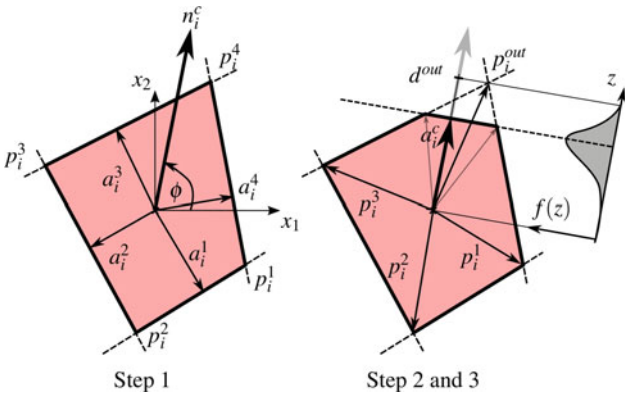

Fig. 4 Particle generation scheme (2D simplification). A randomly oriented vector n_i^c is used to generate a new half space plane a_i^c to trim the original polyhedron

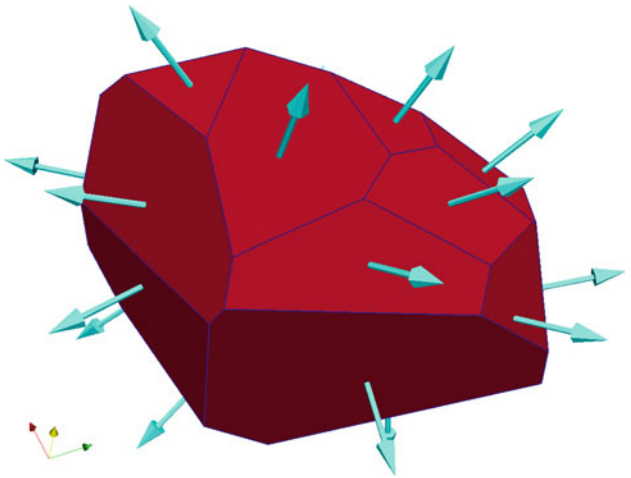

Fig. 5 Example particle generated by trimming a primary solid with randomly generated cut planes. The *arrows* indicate the plane normal directions

$f(\psi)$ is applied to ψ between $[0, 2\pi]$ and a probability distribution function $f(\phi) = \cos(\phi)$ is used for ϕ in the range $[-\pi/2, \pi/2]$.

Second, the distance d^{out} of a plane with normal direction n_i^c, which touches the outer corner p_i^{out} has to be determined. This is done by calculating the orthogonal projected length of each corner vector p_i on the plane direction vector n_i^c and taking the maximum value of the resulting distances.

$$d^{out} = \max(n_i^k p_i^k), \quad k = 1, 2 \ldots n_{vertices} \tag{47}$$

Third, the normal vector a_i^c of the new cut plane is defined by scaling the original normal vector n_i^c to a random length between zero and d^{out}. For this purpose a third probability function $f(z)$, usually a log-normal distribution, is used (see Fig. 4).

The shape of the resulting particles depends on the chosen probability functions for the orientation and the length of the plane vector. Furthermore, the 'roundness' of the edges

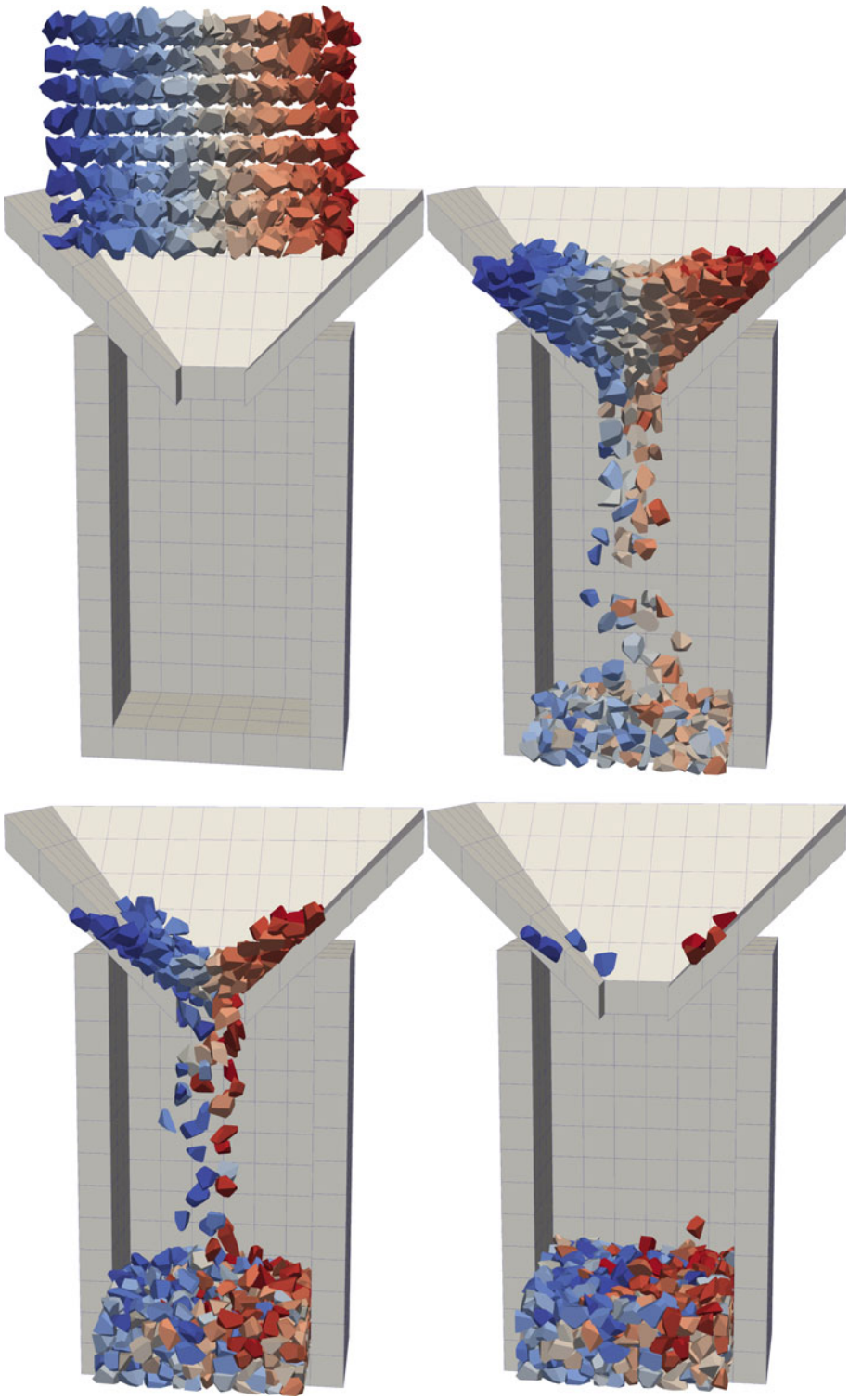

Fig. 6 800 Particles (coloured by id) falling into a hopper (*grey*, only half of the hopper shown) at different time steps

and corners can be influenced by changing the probability function for the cut position $f(z)$ during the creation process. Shifting the maxima of $f(z)$ with each new created plane towards the outer corner leads to a more rounded shape. This process is intended to be close to the creation process of real particles, where contact forces during crushing processes lead more often to fracture of small, external corners. Figure 5 shows an example particle.

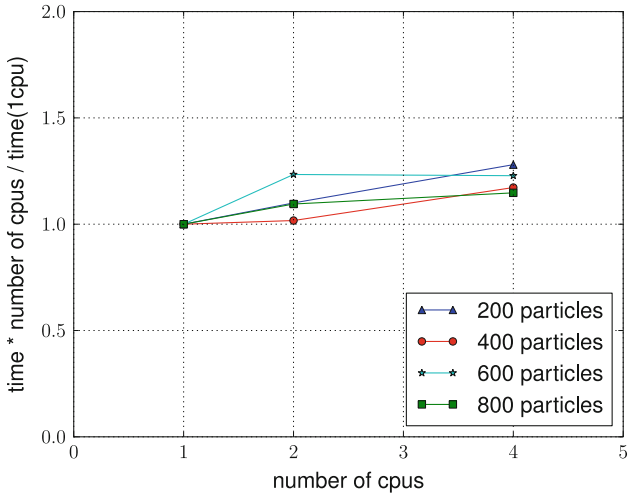

Fig. 7 Performance for increasing number of processors

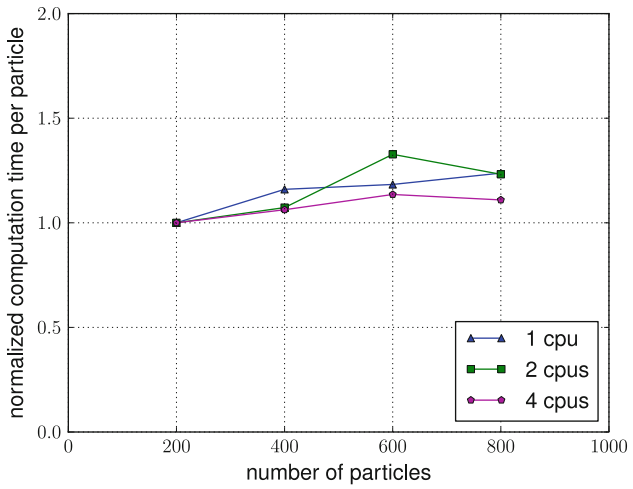

Fig. 8 Performance for increasing number of particles

6 Example

The presented method is implemented in the open source DEM code LIGGGHTS [15] and used to simulate the flow of particles through a hopper into a silo. The walls are modelled with polyhedral particles which are fixed at their position but included in the calculation of interaction forces. In several simulations the hopper is filled with 200, 400 and 800 particles. The simulations where performed on a desktop PC with Intel quad core 2 processor. The calculations where repeated with a single processor as well as in parallel with 2 and 4 processors.

Figure 6 shows the results for the hopper flow with 800 moving particles at different time steps. Figures 7 and 8 show the dependency of the computing time on the number of particles and the number of processors. For the considered particle numbers it can be seen, that the computing time increases almost proportional with the number of particles, which is important when running simulations with higher numbers of particles. On the other hand the computing time decreases almost inversely proportional with the number of processors. This shows that the parallelization is very efficient.

7 Conclusion

Convex polyhedral particles are able to give a good approximation of the shape of various particle types in granular matters. The geometry of these polyhedra was described by a set of half spaces. Based on this geometry description a fast algorithm for the contact detection and the calculation of the interaction forces was developed. This algorithm mainly consists of calculating that polyhedral domain, which is obtained by interpenetration of two particles in contact and is calculated by all half spaces defining the two polyhedra. All contact types can be recognized by the presented algorithm without distinction of special cases. A complete description of the overlapping domain is obtained, which is a good basis for the calculation of interaction forces. Several steps are included in the algorithm to speed up the calculation as well. The model was implemented in the DEM code LIGGGHTS and used to simulate the particle flow through a hopper. The test showed, that the calculation time per particle is almost constant, if the number of particles is increased. Parallel computing efficiency is very good, too.

The presented techniques are not limited to Discrete Element Method, but can also be used in other numerical methods. There exist numerous particle based numerical methods where the geometry representation, contact detection and calculation of overlapping region plays an important role. These methods include simulations of polyhedral packing [16,17] or Finite-Discrete Element Method [18,19].

References

1. Cleary, P.W.: Industrial particle flow modelling using discrete element method. Eng. Comput. **26**(6), 698–743 (2009)
2. Lillie, C., Wriggers, P.: Three-dimensional modelling of discrete particles by superellipsoids. Proc. Appl. Math. Mech. **6**, 101–102 (2006)
3. Bierwisch, C., Kübler, R., Kleer, G., Moseler, M.: Modelling of contact regimes in wire sawing with dissipative particle dynamics. Phil. Trans. R. Soc. A **369**, 2422–2430 (2011)
4. Kodam, M., Bharadwaj, R., Curtis, J., Hancock, B., Wassgren, C.: Force model considerations for glued-sphere discrete element method simulations. Chem. Eng. Sci. **64**, 3466–3475 (2009)
5. Houge, C.: Shape representation and contact detection for discrete element simulation of arbitrary geometries. Eng. Comput. **15**(3), 374–390 (1998)
6. Zhao, D., Nezami, E.G., Hashash, Y.M.A., Ghaboussi, J.: Three-dimensional discrete element simulation for granular materials.

Eng. Comput. Int. J. Comput. Aided Eng. Softw. **23**(7), 749–770 (2006)
7. Eberhard, P., Muth, B.: Dynamics of poured polyhedra of different shape. Lect. Notes Appl. Comput. Mech. **28**, 245–269 (2006)
8. Ghaboussi, J., Barbosa, R.: Three-dimensional discrete element method for granular materials. Int. J. Numer. Anal. Methods Geomech. **14**, 451–472 (1990)
9. Cundall, P.A.: 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)
10. De Pellegrin, D.V., Stachowiak, G.W.: Simulation of three-dimensional abrasive particles. Wear **258**, 208–216 (2005)
11. Lee, Y., Fang, C., Tsou, Y.-R., Lu, L.-S., Yang, C.-T.: A packing algorithm for three-dimensional convex particles. Granul. Matter **11**, 307–315 (2009)
12. Eberly, D.: Polyhedral Mass Properties (Revisited). http://www.geometrictools.com/Documentation/Documentation.html (last view may 9 2012)
13. Tonon, F.: Explicit exact formulas for the 3-D tetrahedron inertia tensor in terms of its vertex coordinates. J. Math. Stat. **1**(1), 8–11 (2004)
14. Feng, Y.T., Han, K., Owen, D.R.J.: Energy-conserving contact interaction models for arbitrarily shaped discrete elements. Comput. Methods Appl. Mech. Eng. **205–208**, 169–177 (2012)
15. Kloss, C., Goniva, C.: LIGGGHTS A new open source discrete element simulation software. In: Proceedings of The Fifth International Conference on Discrete Element Methods, London, UK, 25–26 August ISBN 978-0-9551179-8-5 (2010)
16. Damasceno, P.F., Engel, M., Glotzer, S.C.: Crystalline assemblies and densest packings of a family of truncated tetrahedra and the role of directional entropic forces. ACS Nano **6**(1), 609–614 (2012)
17. Torquato, S., Jiao, Y.: Dense packings of the platonic and archimedean solids. Nature **460**, 876–879 (2009)
18. Munjiza, A., Andrews, K.R.F.: Penalty function method for combined finitediscrete element systems comprising large number of separate bodies. Int. J. Numer. Methods Eng. **49**, 1377–1396 (2000)
19. Owen, D.R.J., et al.: The modelling of multi-fracturing solids and particulate media. Int. J. Numer. Methods Eng. **60**, 317–339 (2004)