Contacts between Many Bodies

B. Muth[1], M.-K. Müller[2], P. Eberhard[1], and S. Luding[2]

[1]Institute B of Mechanics, University of Stuttgart,
Pfaffenwaldring 9, 70569 Stuttgart, Germany,
[muth,eberhard]@mechb.uni-stuttgart.de

[2]DelftChemTech, TU Delft,
Julianalaan 136, 2628 BL Delft, The Netherlands,
[s.luding,m.k.mueller]@tnw.tudelft.nl

Abstract

In this paper the calculation and administration for motion and contacts of systems is investigated that are consisting of many colliding bodies. Both, two dimensional (2D) and three dimensional (3D) cases are investigated. In order to reduce the high calculation time that is usually spent on collision detection, sophisticated sorting algorithms for the neighborhood search are required. Here, these algorithms are compared with respect to their efficiency, for systems consisting of spherical 3D bodies of different sizes. The very time consuming collision detection then only needs to be done for neighboring bodies. The collision detection for polygonal bodies as well as the determination of the contact geometry is alluded. In order to simulate a very high number of contacting bodies, methods from Molecular Dynamics are used. By means of constitutive equations, the contact force can be determined, as addressed for polygonal bodies.

key words: contacts, neighborhood search, collision detection, molecular dynamics, polygonal bodies

1 Introduction

For the determination of the dynamical behavior of systems consisting of many objects, particles or atoms, several fully developed approaches exist. The main differences are the assumptions about the particle shapes and their behavior on collisions. Here, we examine bodies that can be treated as perfectly rigid objects, as non-deformable objects with (small) overlaps at the contact zones, or as deformable bodies with a peculiar contact dynamics.

Very efficient methods were developed for molecular dynamics (MD) simulations, and are, for example, applied to the dynamic and static behavior of granular matter [Kishino01, PöschelLuding01, Lätzel et al.01, Kishino01]. However, also basic systems such as gases, fluids, molecules or charge carriers can be investigated [AllenTildesley89, Luding98b, PöschelLuding01]. Here it can be dealt with motions and contacts of many thousands of particles. Besides body forces like the gravitational forces, one typically has contact forces resulting from the boundaries of the system and from other particles within the system. The formulation of the contact forces between the different bodies is based on simple models in order to keep the calculation times within a feasible range. Here, usually very small penetrations between the otherwise non-deformable particles are accepted, compare [AllenTildesley89, Luding98a, Rapaport95, Lätzel et al.01].
The normal contact forces act in direction opposite to the occurring penetrations and are usually modeled as a spring dashpot element that combines elastic and viscous response on the contact level. Then the spring force is proportional to the penetration of the particles, see [Luding98a, PetersDžiugys02], and it corresponds to a penalty force.
For a system consisting of n particles, the required calculation operations for the collision detection will be of order $O(n^2)$, causing huge computational effort. However, for systems with short-range forces, only particles in their respective neighborhood can interact, so that a tremendous reduction of computational effort down to the order $O(n)$ can be achieved [AllenTildesley89] by utilizing this fact.
Here, we want to combine the methods from MD, with their efficiency and their possibility to simulate a very high number of bodies, with the advantages of MBS, where non-convex body shapes are possible, whereas in MD mostly convex polygons were used, [Addetta et al.02, Matuttis et al.00, Schinner99]. In Section 2, different approaches to save computational effort as mentioned above are introduced and discussed. By means of these studies, neighboring body pairs are found efficiently. These body pairs are then checked for collision, and in the case of collision, the contact forces have to be determined. This is done for convex and non-convex polygonal bodies, in Section 3. In Section 4 these methods are compared using several test examples and, finally, in Section 5, the results are discussed with respect to possible applications.

2 Neighbor Search Methods

The following three briefly presented methods can be used in order to find neighboring bodies of a particle efficiently. Two of these methods identify the neighboring particles of a body by defining splitted regions of the system and considering all particles within the same region as neighbors. The third method is based on a different approach, where a bounding box is placed around each body. Every body whose bounding box is colliding with the bounding box of another particle is considered to be a neighbor of this particle [Schinner99]. For each method the neighboring particles are stored in a neighbor data-structure after the pre-sorting has been finished. The collision detection and force calculation then only needs to be done for these neighboring bodies.

2.1 Verlet-Neighbor List

The first method is called the Verlet-Neighbor List (VL) [AllenTildesley89]. An imaginary sphere is drawn around each particle of the system, that is larger than the radius of the particle, compare [AllenTildesley89, Muth01]. Particles within these enclosing spheres are considered as neighbors of the particular body and are stored in a list, [Vu-Quoc et al.00]. The optimal extension of the zone around the bodies depends on the velocity of the particles and on the density of the whole system.
In order to create the neighbor lists, for each body all particles of higher numbers than the body itself have to be tested, whether they lie inside the test sphere or not. Particles of lower numbers have not to be tested as no pair needs to be checked twice.
Creating these neighbor lists therefore requires $n(n-1)/2$ calculation steps, which means the number of necessary arithmetic operations is of order $O(n^2)$. However, these updates of the lists do not have to be done for every time step. The necessary update frequency depends on the density of the system, the velocity of the particles, and on the size of the spheres. This update frequency, as well as the radius of the spheres around the particles is a parameter that can be tuned. Both parameters are interdependent, because the value for the radius is inversely related to the rate at which the list must be rebuilt, see [Rapaport95]. The smaller the zone around the particles, the more often the reconstruction of the lists needs to be done. The larger it is, the more particles belong to the neighborhood requiring more contact calculation time. The real collision detection and force calculation, that requires quite some effort, now only has to be done for the particle pairs which are stored in the lists.

2.2 Linked Cell Method

An alternative approach that is often used in order to determine the neighbors of a body is the Linked Cell (LC) method, where the system is divided into a regular lattice of, e.g. for cubic systems, $m \times m \times m$ cells (3D) or $m \times m$ cells (2D), compare [AllenTildesley89]. Non-cubic systems can also be dealt with and the cell shape can be chosen as to fit the system. The optimal size of the cells, as the size of the Verlet spheres in the VL, depends on the velocity of the particles and on the density of the system, but the cell sizes all need to exceed at least the size of a particle in length. The major difference between LC and VL is, that for LC the cells are not sticked to particles and, therefore, they are not moving along with the particles. Then, if particles are temporarily assigned to special cells on the basis of their current positions, it is obvious that interactions are only possible between those in the same or in directly adjacent cells, see [Rapaport95]. This means for a 2D system, that only particles within nine different cells for 2D and 27 cells for 3D may be neighbors. Again, as in the previous description for VL, particle pairs are only checked once. Thus, not all nine cells have to be tested, but only the center cell plus half of the neighboring cells. That means one plus four (2D), Fig. 1 (left), and one plus thirteen cells (3D), Fig. 1 (right), have to be checked. Cast into a formula, we have $(3^d + 1)/2$ cells to examine, where $d = 2$ for a 2D system and $d = 3$ for a 3D system.

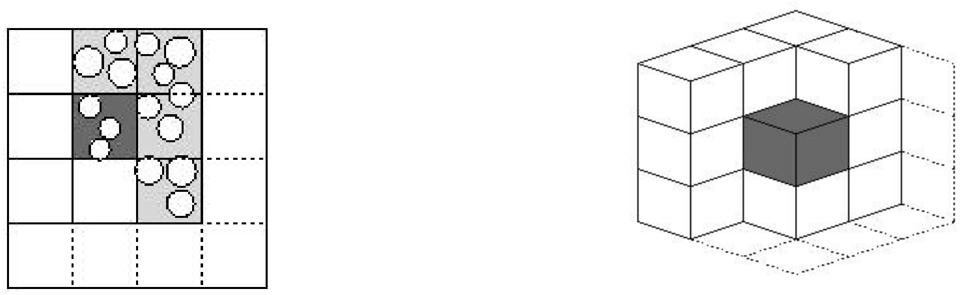
Figure 1: Neighboring cells that need to be investigated are shaded grey in 2D (left) and are shown as solid cubes in 3D (right).

2.3 Linked Linear List

The third possibility described in this comparison is the Linked Linear List (LLL) [Schinner99]. This method is quite different to the other approaches. In a first step, bounding boxes are laid around each particle, Fig. 2, that are sized in such a way, that each particle fits exactly in its box. The edges of each bounding box are aligned parallel to the system axes.
In a next step the bounding boxes are projected separately onto the system axes, see Fig. 2. In the following, only the order of the beginnings 'b' and endings 'e' of the projections of the bounding boxes along the axes is of interest. Such a stored sequence has a length, which corresponds to twice the number of particles in the system. If there is the beginning, ending, or both, of another particle in between the beginning and ending of a particular body, then there will be an overlap of the projections of the bounding boxes of both particles along this axis. A collision of two bounding boxes exists for an overlap of these projections along each axis. Each particle pair, whose bounding boxes are overlapping, is considered as a neighboring body pair, that has to be stored in a linear list.
Although the sequences have to be updated for each time step, the necessary calculation times can be reduced to an amount proportional to the total number of

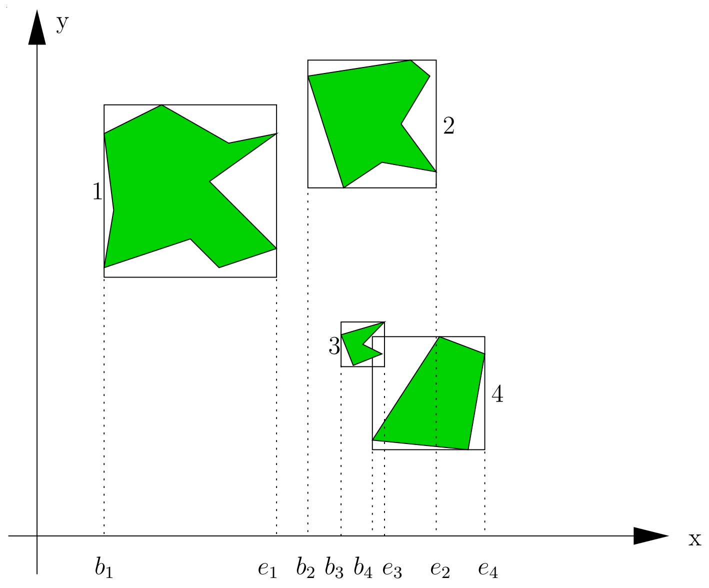

Figure 2: Bounding boxes around each particle.

particles in the system, as there has to be done only an update of the old sequence for each new time step. That corresponds to sorting an already nearly sorted list. This update can simply be done by going through these sequences linearly and checking for new changes in the order. While seeking for new colliding bounding boxes by looking for permutations, four different cases have to be distinguished, compare [Schinner99].

If two beginnings or two endings have to be changed, nothing has to be done in the generated linear lists, as the collision status between any particle will not change. If a collision along an axis has to be removed, or if there is a new collision between two particles along an axis, the collision information along the other axes is essential. One has to know whether there is a new or old collision along all axes and, therefore, between the bounding boxes or, whether there is no overlap any more between two so far colliding bounding boxes at least in one direction. For our 2D example going through the list along the y-axis, see Fig. 2, leads to the potential collision between particles (3/4), and (1/2). As the location of particle 3 along the x-axis is from position four to six, whereas the beginning of particle 4 has the position five, there is also an overlap of bounding boxes 3 and 4 along the x-axis and, therefore, a collision of the bounding boxes of particles 3 and 4. Particles 3 and 4 are now considered to be neighbors, that have to be checked for collision. Therefore, a linked list is created in the form of a sparse matrix where the colliding bounding boxes are stored. Collision pair (3/4) is stored at position $3, 4$ of the matrix, see Fig. 3, that shows particle pair storage for an arbitrary configuration.

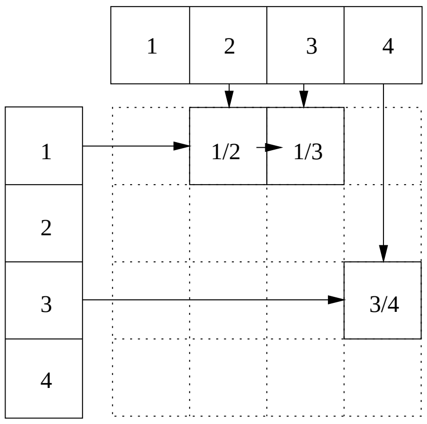

Figure 3: Storage of colliding bounding box pairs, if e.g. boxes (1/2), (1/3) and (3/4) collide.

2.4 Discussion

The two techniques described in Section 2.1 and Section 2.2 have in common that both identify neighboring particles by considering bodies inside certain zones as neighbors. For both methods these zones have to be at least larger than the particles themselves. Out of that, two problems can arise. Firstly, if the particles within the system are polydisperse, that means their sizes differ, then the size of the grid (LC) or circles (VL) has to conform to the largest particle existing in the system. Hence, for highly polydisperse mixtures, the smaller particles may increase the number n_c of particles within one cell, which might even, in the worst case, be close to n, see [Schinner99].
As the neighborhood zones around a particle are larger than the particles, the neighbor data-structure does not have to be updated in each time step. The size of the zones and the necessary update frequency are interdependent and not quite easy to guess. Therefore, another problem of both methods is the ascertainment of optimal values for both, the update frequency for the lists and the size of the zones. If the cells are chosen smaller than the particle size, a successful contact detection can be impossible in the way described above. Besides that, if the cells are either very small or very large, the contact detection is inefficient because updates are too frequently necessary or too many particles are in the neighborhood, respectively. Therefore, the choice of these values is very important, and it may take a lot of personal time getting experience with the investigated system.

3 Treatment of different bodies

In the following section, the collision detection for different polygonal bodies in two dimensions will be determined. Convex polygons may be calculated as well as non-convex ones. In order to be able to apply the forces and moments to the polygons, it is necessary to investigate the contact geometry, Section 3.2. The contact force calculation and the moments that act on the bodies are described in Section 3.3.

3.1 Collision Detection

Once neighboring body pairs have been detected, these pairs have to be checked for collision. For round particles this is quite easy, since there only the distance of the centers of masses has to be compared with the sum of the radii of both bodies. In the case of polygonal particles, there may occur several collisions between two bodies and, therefore, the situation is much more complicated.
Only the nodes of each polygon are checked and it has to be distinguished between the situation where a corner of the investigated body, e.g. body i, is entering another one, e.g. body j, and the situation that the observed body i is being entered by body j. Of course it is possible, that body i is entering body j and at the same time is being entered by body j. In the following we want to call the tested particle master body and its neighbors slave bodies. In order to know, which corner of which body is colliding with another body, every corner of each body has to be checked independently. This is done by means of a 'Point in Polygon' algorithm. Using such a method it is possible to check, whether an arbitrary point P (here the considered corner of the tested polygon) is inside a body or not. One of these algorithms is called the 'Ray Crossing Method' [O'Rourke93].

3.2 Contact Geometry

Let us consider that there is a vertex of one body inside another body. In order to apply the correspondent contact force and the moment that is acting on the bodies, we need to detect the contact geometry. For spherical bodies this is quite simple again. The normal contact force is directed from the center of mass of one body to the center of mass of the other body and thus does not result in a moment applied to the bodies.
This is different for polygonal bodies. Here, the normal contact force may act at any position of the body and then certainly can result in a moment around the center of mass for both bodies, see Fig. 4.
For this reason, the contact geometry has to be detected. There the position of the applied force is determined. This can be done according to the node-to-segment approach originally used for discretised contact mechanics [Wriggers02]. Here, a slave point $\mathbf{x}_s$ comes into contact with a master segment, given by the two points $\mathbf{x}_{m1}$ and $\mathbf{x}_{m2}$, Fig. 5. The variable ξ determines the position of the projection of $\mathbf{x}_s$ on the edge between $\mathbf{x}_{m1}$ and $\mathbf{x}_{m2}$, and its value is $\xi = 0$ for $\mathbf{x}_{m1}$ and $\xi = 1$ for $\mathbf{x}_{m2}$. The value of ξ has to be determined, which corresponds to the closest point projection $\mathbf{x}_{s\,proj}$.

Therefore, the projection $\mathbf{x}_{s\,proj}$ of point $\mathbf{x}_s$ onto the master surface has to be done, [Pfister99, Wriggers02]. The tangential vector of the master segment can be detected as

$$\mathbf{t}_m = \frac{\mathbf{x}_{m2} - \mathbf{x}_{m1}}{\|\mathbf{x}_{m2} - \mathbf{x}_{m1}\|}. \tag{1}$$

The projection then leads to the scalar product

$$\xi_{proj} = \mathbf{t}_m \cdot \frac{(\mathbf{x}_s - \mathbf{x}_{m1})}{\|\mathbf{x}_{m2} - \mathbf{x}_{m1}\|}. \tag{2}$$

The direction of the normal contact force can be derived as

$$\mathbf{n}_m = \mathbf{e}_z \times \mathbf{t}_m \;=\; \frac{(\mathbf{x}_s - \mathbf{x}_{s\,proj})}{\|\mathbf{x}_s - \mathbf{x}_{s\,proj}\|}, \tag{3}$$

where $\mathbf{e}_z$ is the coordinate axis, about which the rotation is carried out. The gap g, the distance from the slave point to the master segment, that is needed for the

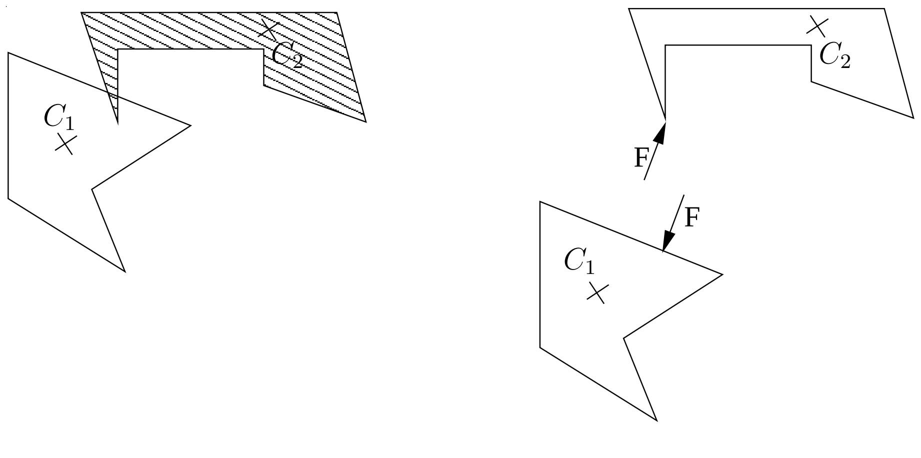

Figure 4: Two particles in contact with their resulting normal forces.

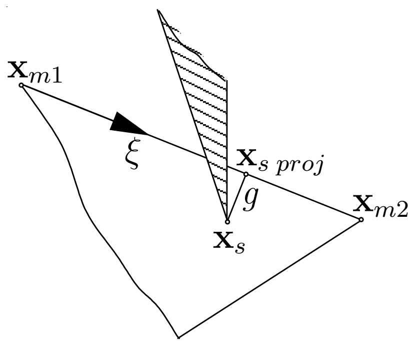

Figure 5: Node-To-Segment element consisting of one slave node $\mathbf{x}_s$ and two master nodes $\mathbf{x}_{m1}$ and $\mathbf{x}_{m2}$.

contact force calculation, can then be computed as

$$g = (\mathbf{x}_s - \mathbf{x}_{m1}) \cdot \mathbf{n}_m \ = \ \|\mathbf{x}_s - \mathbf{x}_{s\,proj}\|. \tag{4}$$

3.3 Contact Force Calculation

As soon as the contact geometry is detected and the information of the contact kinematics is stored, the contact force calculation for all bodies that are in contact can be performed.
The used contact force model is based on a penalty approach with linear spring dashpot elements. Therefore, the balance of linear momentum for a body i that is in contact with other bodies j is

$$m_i \frac{d^2}{dt^2}\mathbf{r}_i = \underbrace{\sum_j \left(k g_{ij} + v_{rel\,ij} d\right) \mathbf{n}_{ij}}_{\mathbf{f}_i}, \tag{5}$$

with the mass of the body m_i, the positions of the body $\mathbf{r}_i$ and the force $\mathbf{f}_i$ that is applied to it. Here, k equals the spring constant, whereas d equals the damping coefficient due to the dissipative force. The gap g_{ij} for polygonal bodies is calculated as described in Section 3.2 with the normal direction at the contact, $\mathbf{n}_{ij}$, [Luding98a, Tzaferopoulos95]. The forces are only applied to the particles if they are in contact. Frictional forces as well as adhesive contact forces are not considered here.
As soon as the normal contact forces that are applied to each body are detected, the moments acting about the center of gravity may be determined. Since mostly the coordinates for the master edges are given in the body-fixed coordinate system in the center of gravity, the moment about body i is

$$\mathbf{M}_i = (\mathbf{x}_{m1} + \xi_{proj}\mathbf{t}_m) \times \mathbf{f}_i. \tag{6}$$

The detected moments are then adopted to the balance of angular momentum. These equations are solved by means of the explicit Verlet integrator. The new positions of the particles are computed from the actual positions and the positions for an old time step, that means without knowledge of the velocities of the particles.

4 Comparisons and Results

The three techniques introduced shall in the following be compared with respect to the simulation times for different test series. Goal of these series is, to keep some system traits as constant as possible and to change only some well defined influencing factors.

4.1 Spatial Monodisperse Increasing Systems

In a series of 3D systems examples with a different number of particles are studied. An increasing number of spherical particles of equal size, $R = 5.0 \times 10^{-4}$ m, and a proportionally growing space around the particles is investigated in this section. The stiffness of the particles is $k = 10^5$ N/m, the damping coefficient is neglected; the density of the particles is $\rho = 10^{10}$ kg/m^3, and the time step for the integration is chosen as $\Delta t = 10^{-5}$ s. Here, systems of 512, 2197, 5832, 12167, 21952, 35937, and 59320 particles are investigated, where the linked cell size was kept equal $l_c = 2 \times 10^{-3}$ m for each system. The behavior of the computation time with respect to the number of particles within the particular system is shown in Fig. 6 as log-log plot.

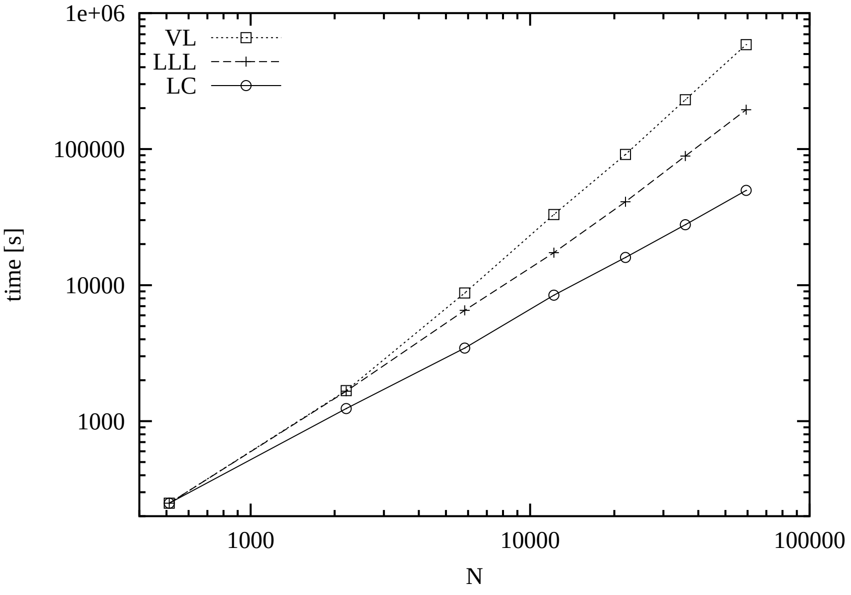

Figure 6: Comparison results for an increasing 3D example.

We normalized the results in such a way that for the smallest investigated number of particles (where the influence of the used method is mainly negligible) a contact scaling factor has been computed. Therefore, the curves in the figure have the same starting point.
The more the number of particles rises, (and with it the number of contacts during the simulation) the more the performance for VL escalates in comparison to LLL and LC. Here LC has the best performance: a roughly similar curve for the behavior of VL is $y \approx (0.027\,x)^{1.8}$, the polynomial for LLL is $y \approx (0.12\,x)^{1.36}$ and the polynomial for LC is $y \approx (0.2\,x)^{1.15}$. Therefore, it can be said, that the behavior of LLL and LC both remain close to linear in contrast to the behavior for VL, which is almost quadratic.

4.2 Spatial Polydisperse Fracturing Systems

In another 3D comparison the system size is kept constant. In contrast to the previous example, the number of particles is increased, but no change of either the system or cell-size is undertaken, $l_c = 0.033$ m. Chosen parameters for this system are the stiffness $k = 4\times10^6$ N/m, damping coefficient $d = 0$ Ns/m, density of the bodies $\rho = 7000$ kg/m^3, and the time step for the integration $\Delta t = 4\times10^{-7}$ s. The systems can be seen as a series of fracture of some of the particles, where neither the volume enclosed in the system boundaries nor the volumetric content of the system is changed. That means, that the volume fraction and the density of the system are unchanged, while the number of particles within the system is increased. In the first system 1000 particles are situated, with equal radii $R = 0.01$ m. Approximately half of these particles are now successively fractured in the next systems. There are about 500 particles of radius R, but approximately eight times 500 particles of radius $R/2$ and thus an eighth of the original particle volume. The systems therefore contain

1000 particles (see Fig. 7 on the left),
4451 particles, about 4000 smaller bodies of $r = \frac{R}{2}$,
14676 particles, about 14000 particles of $r = \frac{R}{3}$,
32374 particles, about 31000 particles of $r = \frac{R}{4}$, and
65480 particles, about 65000 particles of $r = \frac{R}{5}$ (see Fig. 7 on the right).

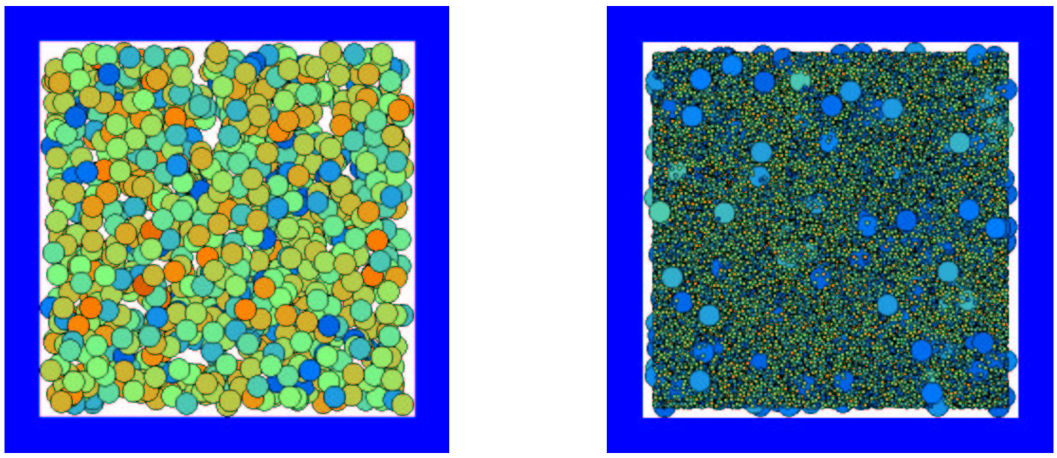

Figure 7: A monodisperse system and a very polydisperse system with $r = R/5$ of exactly the same volume and density (volume-fraction $\nu = 0.12$).

Here, R is the large original radius of all particles of the first system, and r the smaller radius of the many small particles in the other systems. For this system the computation time needed per particle is presented in Fig. 8 over the ratio of the radii that is equivalent to the polydispersity of the system.

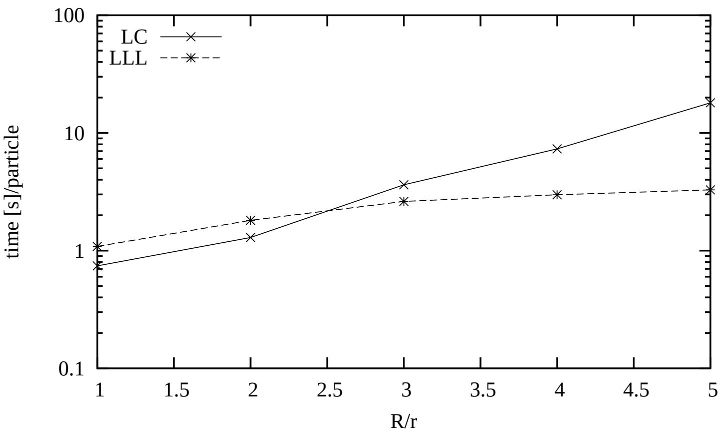

Figure 8: Computation time for a fracturing system of constant density.

It can be seen that the gradient of the curve over the polydispersity of the LC calculation is a lot higher than for LLL, and the curves are intersecting at about $R/r = 2.5$. This means that the LC method has advantages for quite monodisperse systems, while the LLL shows its advantages for polydisperse systems.

4.3 Simulation Results for 2D Polygonal Bodies

In the following section, the compared methods shall be applied to a polygonal system. For such systems the collision detection is usually very time consuming, as described in Section, 3.1. Here, we applied the LLL method to a quite polydisperse non-convex system. Then, only neighboring bodies need to be checked for collision. During the collision check the already created bounding box can be used once more by accomplishing a pre-test for each vertex of a body pair. In the pre-test all vertices are tested, whether they are inside the bounding box of the other body. Only for such vertices the time consuming collision detection has to be carried out.

Here, we want to present briefly a simulation result of such a 2D non-convex polygonal system. A series from an animation for such a system is shown in Fig 9. Some non-convex bodies, for example micro chips, are gliding down a wall arrangement in a production mechanism.

5 Conclusions

In this paper three different methods have been introduced in order to reduce the calculation time for collision detection for systems consisting of many particles with contact interactions only. The basic idea of these techniques is the fact, that usually there are lots of particles in a system, which cannot be in touch, as they are too far apart. The comparison has been done for 3D systems consisting

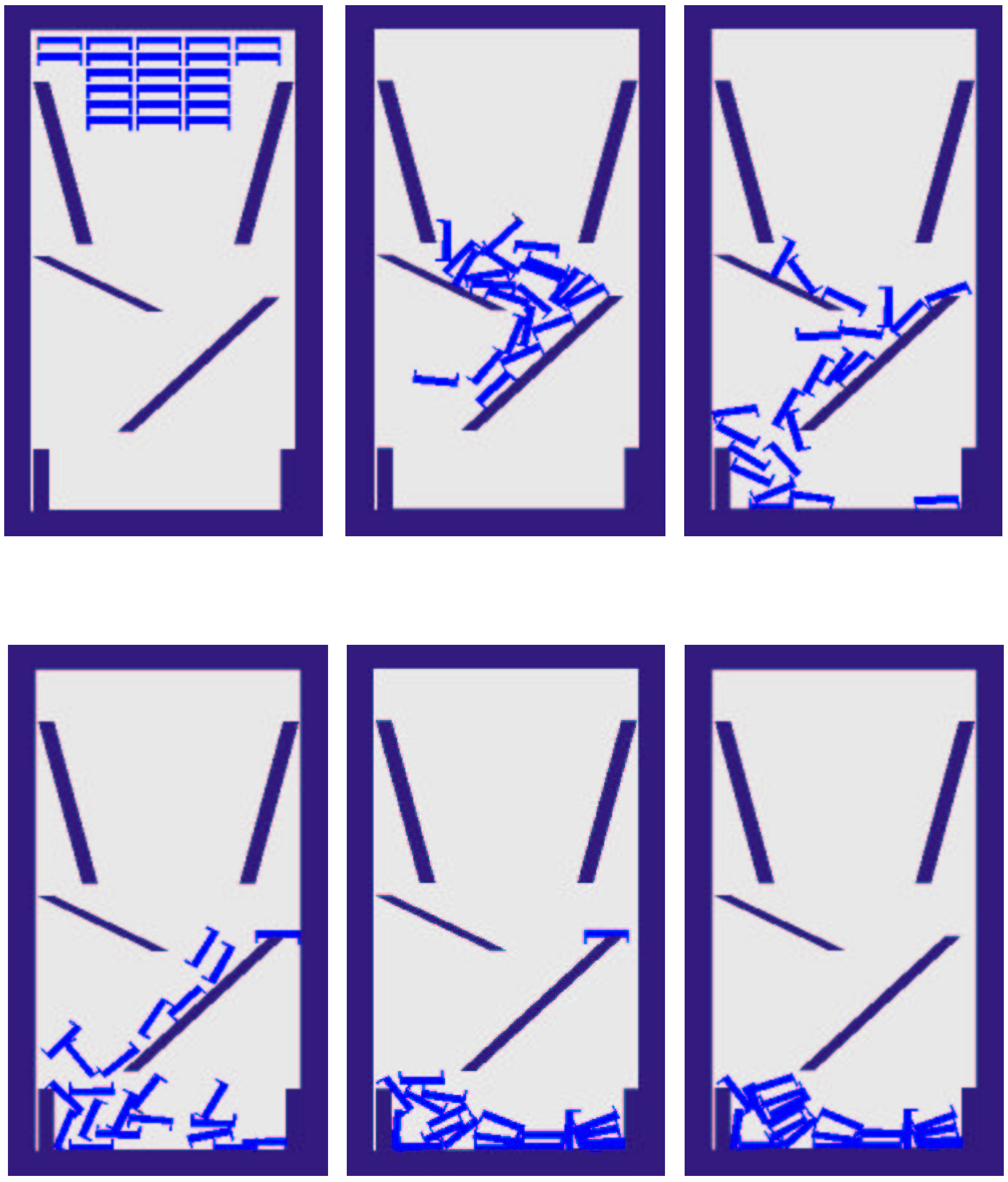

Figure 9: Non-convex polygonal bodies for $t = 0$s, $t = 0.05$s, $t = 0.1$s, $t = 0.13$s, $t = 0.22$s, and $t = 0.3$s.

of spherical bodies, but the contact detection and contact force calculation has also been implemented for convex and non-convex polygons. For such a system exemplary scenes from an animation have been shown.
It has been shown, that for rather small systems the traditional VL is a quite good technique, its efficiency being reasonable as it is very easy to implement.

For larger systems the LC and LLL methods become more and more efficient compared to the VL. The LC method shows very good performance with a lower increase of computing time for monodisperse systems, as compared to the LLL method. A practical problem using LC is the optimal choice of the linked cell size. For too small sizes and for too large sizes the computation becomes inefficient. Therefore, a new optimal cell size has to be found for each new system.
As the linked cell size is dependent on the largest particle in the system, and as the calculation time is dependent on the linked cell size, the time response for LLL becomes much better as compared to the LC for polydisperse partial fracturing systems of constant volume and density with increasing numbers of particles.
The compared methods have been applied to a non-convex system consisting of polygonal bodies. For such systems, the collision detection is very time consuming. Great advantages regarding simulation time can be made by applying the neighbor-search methods to such systems.

References

[AllenTildesley89] **Allen, M.P., Tildesley, D.J.**, 1989, *Computer Simulations of Liquids*, Clarendon Press, Oxford.

[Addetta et al.02] **D'Addetta, G.A., Kun, F., Ramm, E.**, 2002, On the Application of a Discrete Model to the Fracture Process of Cohesive Granular Materials, *Granular Matter* 4 (2), 77-90.

[Kishino01] **Kishino, Y.**, 2001, (ed.), *Powders & Grains*, Balkema, Rotterdam.

[Lätzel et al.01] **Lätzel, M., Luding, S., and Herrmann, H.J.**, (ed.), 2001, Continuous and Discontinuous Modelling of Cohesive Frictional Materials, *Lecture Notes in Physics* 568, Springer, Berlin.

[Luding98a] **Luding, S.**, 1998, Collisions & Contacts between Two Particles, in: H. J. Herrmann, J.-P. Hovi, and S. Luding, (ed.), *Physics of Dry Granular Media*, NATO ASI Series E350, Kluwer Academic Publishers, Dordrecht, 285.

[Luding98b] **Luding, S.**, 1998, *Die Physik trockener granularer Medien*, Logos Verlag, Berlin.

[Matuttis et al.00] **Matuttis, H.-G., Luding, S., and Herrmann, H.J.**, 2000, Discrete Element Methods for the Simulation of Dense Packings and Heaps made of Spherical and Non-Spherical Particles, *Powder Technology*, 109, 278-292.

[Muth01] **Muth, B.**, 2001, Simulation von Kontaktvorgängen einfacher Körper mit Methoden der Molekulardynamik, Diploma Thesis, University of Stuttgart.

[O'Rourke93] **O'Rourke, J.**, 1993, *Computational Geometry in C*, Cambridge University Press, Cambridge.

[PetersDžiugys02] **Peters, B., Džiugys, A.**, 2002, Numerical Simulation of the Motion of Granular Material Using Object-Oriented Techniques, *Comp. Methods Appl. Mech. Engrg.*, 191, 1983–2007.

[Pfister99] **Pfister, J.**, 1999, Implementierung von Reibkontakten für nichtlineare Finite Elemente Berechnungen, Diploma Thesis, University of Stuttgart.

[PöschelLuding01] **Pöschel, T., Luding, S.**, (ed.), 2001, Granular Gases, *Lecture Notes in Physics*, 564, Springer, Berlin.

[Rapaport95] **Rapaport, D.C.**, 1995, *The Art of Molecular Dynamics Simulation*, Cambridge University Press, Cambridge.

[Schiehlen86] **Schiehlen, W.**, 1986, *Technische Dynamik*, B.G. Teubner, Stuttgart.

[Schinner99] **Schinner, A.**, 1999, Fast Algorithms for the Simulations of Polygonal Particles, *Granular Matter*, 2(1), 35–43.

[Tzaferopoulos95] **Tzaferopoulos, M.A.**, 1995, On the Numerical Modeling of Convex Particle Assemblies with Friction, *Comp. Methods Appl. Mech. Engrg.*, 127, 371–386.

[Vu-Quoc et al.00] **Vu-Quoc, L., Zhang, X., and Walton, O.R.**, 2000, A 3-D Discrete-Elemente Method for Dry Granular Flows of Ellipsoidal Particles, *Comp. Methods Appl. Mech. Engrg.*, 187, 483–528.

[Wriggers02] **Wriggers, P.**, 2002, *Computational Contact Mechanics*, J. Wiley & Sons, Chichester.