Collision Detection and Administration Methods for Many Particles with Different Sizes

B. Muth[1], M.-K. Müller[2], P. Eberhard[1], and S. Luding[2,3]

[1] Institute of Engineering and Computational Mechanics,
University of Stuttgart,
Beate.Muth.76@web.de, eberhard@itm.uni-stuttgart.de

[2] Particle Technology, NSM, DelftChemTech, TUDelft,
Julianalaan 136, 2628 BL Delft, Netherlands,
m.k.mueller@tnw.tudelft.nl, s.luding@tudelft.nl

[3] MSM, CTW, University of Twente,
Postbus 217, 7500 AE Enschede, Netherlands

Abstract

This paper deals with the calculation of the motion and the administration of the contacts for systems with many colliding bodies of round shape and possibly large size-differences. Both two dimensional (2D) and three dimensional (3D) cases are investigated, while the efficiency of the employed algorithms is compared. For the integration of the equations of motion, standard methods are used, but to reduce the effort for collision detection, more sophisticated administration algorithms for the neighborhood search are prosented. Especially for large systems with many particles and a wide, polydisperse size distribution, this is a challenge. Three methods, the Verlet-Neighbor List (VL), the Linked Cell (LC) method, and the Linked Linear List (LLL), are discussed and compared for 2D and 3D. Only LLL performs well for strongly different particle sizes.

keywords: contact detection; neighborhood search; molecular dynamics (MD); discrete element method (DEM); polydisperse size-distribution

1 Introduction

In order to determine 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 spherical particles, that can be treated as (*i*) perfectly rigid objects, as (*ii*) non-deformable objects with (small) overlaps at the contacts, or (*iii*) as deformable bodies with a peculiar contact dynamics.

1.1 A Methods overview

Systems consisting of bodies with negligible deformations, i.e., case (*i*), can be described by means of the so-called multibody system method (MBS) [12, 17], where mass point systems may be regarded as a special case of the MBS. Deformable bodies, in case (*ii*), can be represented by a collection of non-deformable particles connected by springs if they overlap, and not interacting otherwise [14]. Their motion is computed, using a molecular dynamics (MD) method [1], also called the discrete element method (DEM) when applied to particle systems. (The difference between MD and DEM are the interaction forces between the particles, otherwise both methods are similar in spirit.) For completeness, we remark that the so-called event-driven (ED) molecular dynamics [4, 8, 15] also assumes hard (perfectly rigid) spheres and thus can be seen as a special case of MD/DEM closer to MBS. For truly flexible, deformable bodies, i.e., case (*iii*), usually the Finite-Element-Method (FEM) or the Boundary-Element-Method (BEM) are used, see [2, 13].

Each of the three methods has its own advantages and disadvantages. While the MBS is, in general, characterized by relatively short computation times due to a small number of degrees of freedom, deformations cannot be handled. On the other hand, systems investigated using FEM have a large number of degrees of freedom that yield a rather extensive number of equations of motion, while deformations are properly taken into account. The MD/DEM methods can be seen as a compromise in so far that the number of degrees of freedom is kept small by assuming simple, overlap dependent contact force laws that rely on certain assumptions, e.g., neglecting the eigen-modes of a particle.

An expansion of the MBS method for elastic bodies is presented in [9]. Hybrid MBS/FEM contact calculations are presented in [2], where colliding bodies are examined by the FEM approach in order to incorporate deformations, while all the other bodies of the system are regarded as rigid. This hybrid approach makes use of the advantages of both methods, but also their drawbacks, i.e., the number of contacting particles is quite limited due to the detailed FEM modeling involved for each particle.

Alternatively, efficient algorithms were developed for molecular dynamics and discrete particle systems like granular matter [3, 15, 19] so that molecular (gases, fluids, solids) and particulate (powder, sand) systems can be investigated [1, 7, 15]. The motion and the contacts of many thousands of particles interacting with each other and the boundaries of the system, and also involving body forces like gravity, can be modeled.

1.2 The Force Model

The formulation of the interaction forces between the different bodies is based on models as simple as possible, in order to keep the calculation time feasible. Usually, very small penetration of otherwise non-deformable particles are accepted, compare [1, 5, 16, 19], and are used as the basis of the force calculation.

For a given pair of particles, a *normal contact force* acts in the direction of the center-to-center connection line – opposite to the penetration and thus repulsive. The usual model is a spring-dashpot element that combines elastic and viscous response on the contact level. The spring force is proportional to the penetration/overlap of the particles, see [5, 11], also called a "penalty" force.

Besides the simplest linear model, also various non-linear models are available [5]. Apart from the repulsive, and viscous contact forces, also attractive forces may occur due to adhesion/cohesion, or electro-static interactions. The latter are typically long-ranged and are not subject of this study, where we focus on short range, contact interactions only. For realistic materials, particle deformation can also involve plastic deformation and friction as alternative sources for dissipation; we refer to [6] for more details and restrict ourselves to the most simple, linear contact forces in the following.

For a system consisting of n particles with arbitrary interactions, the required calculation operations for the force computation will be of the order $O(n^2)$, causing huge computational effort. However, for the systems with short-range contact 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 [1]. In this study, we consider n spherical bodies with radii r_i, consecutively numbered from $i = 1, \ldots, n$. The used contact force model is based on linear spring-dashpot elements, so that the equations of motion are

$$m_i \frac{d^2}{dt^2}\mathbf{r}_i = \sum_j \underbrace{(k\delta_{ij} + d\dot{\delta}_{ij})\mathbf{n}_{ij}}_{\mathbf{f}_{ij}} \ , \tag{1}$$

with the masses of the bodies m_i, the positions $\mathbf{r}_i$, and the forces $\mathbf{f}_{ij}$ acting between i and j, while the sum runs over all particles j in the neighborhood of particle i. Here, k is a spring constant, while d is the damping coefficient for the dissipative force. The overlap between two particles i and j is

$$\delta_{ij} = r_i + r_j - (\mathbf{r}_i - \mathbf{r}_j) \cdot \mathbf{n}_{ij} \ , \tag{2}$$

with the normal vector between both particles, $\mathbf{n}_{ij}$, parallel to the line connecting their centers, pointing from j to i. The contact forces are applied only for the case $\delta_{ij} > 0$. Frictional forces as well as adhesive contact forces are not considered here. The equations of motion are solved by means of the explicit Verlet integration scheme. The new positions of the particles are computed based on the information about the actual positions and the positions during the previous time step, that

means without knowledge of the velocities of the particles. The velocities can be approximated from the positions at two different time-steps. The algorithm is second order in accuray, if no velocity dependent forces are involved.

1.3 Overview

In section 2, different approaches to save computational effort are introduced and discussed. In section 3 these methods are compared using several test examples and finally, in section 4, the results are summarized and discussed with respect to possible applications.

2 Neighbor Search Methods

Three methods, as mentioned in the abstract, to find neighboring bodies of a particle efficiently are presented in this section. Two of them, VL and LC, identify the neighboring particles of a body by regarding special regions of the system and considering all particles within the same region as neighbors. As both methods operate with similar ideas, they are also sometimes used simultaneously [16]. The third method, LLL, is based on a different approach. Around each body a bounding box is placed, which is proportional to the particle size; every body whose bounding box is colliding with the bounding box of another particle is considered to be a neighbor of this particle and thus a potential contact partner [18].

For each method the neighboring particles are stored in a neighbor data-structure (NDS) after some pre-sorting. The collision detection needs to be done only for neighboring and potentially colliding bodies. Hence, the necessary calculation operations for collision detection can be reduced down to an order proportional to the number of particles in the system, i.e. $O(n)$, because the update of the NDS can be optimized. In the following we will examine whether this expected performance also holds for LLL – but first the three methods are introduced.

2.1 Verlet-Neighbor List (VL)

The first method presented, the Verlet-Neighbor List (VL), is quite similar to Verlet's originally proposed method. As shown in Fig. 1 an imaginary sphere is drawn around each particle, here with a radius of five times the maximum radius, compare [1]. Particles within these enclosing spheres are considered as neighbors of the body in the center of the sphere. The optimal size of the zone around the bodies depends on the velocity of the particles and on the density of the system.

For each particle a list is generated, where all neighboring bodies are stored [20]. In order to compile these 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 are not tested since no pair needs has to be checked twice.

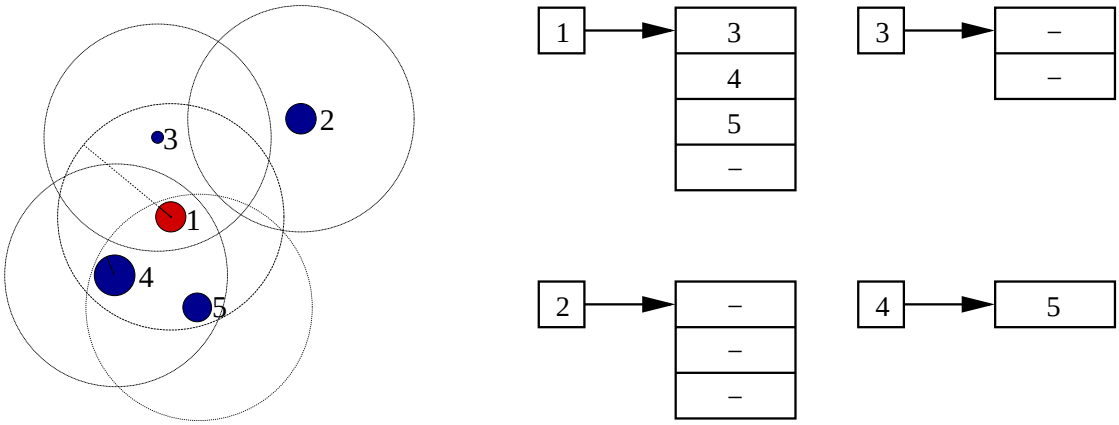

Figure 1: Verlet circles (2D) or spheres (3D) and particle storage in lists.

Creating the neighbor lists requires $n(n-1)/2$ calculations, which means the number of necessary operations is still of order $O(n^2)$. However, the lists do not have to be updated for every time step. The update frequency depends on the density of the system, the velocity of the particles, and on the size of the spheres. For relatively dense systems, an update of only each 100th time step can be enough. Like the value for the radius of the spheres around the particles, also the update frequency is a parameter that can be tuned and, in principle, has to be checked for every new simulation. Both parameters are interdependent, because the value for the radius is inversely related to the rate at which the list must be rebuilt, see [16]. 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 additional effort, especially for more advance interaction models, now only has to be done for the particle pairs which are stored in the lists. That leads for the example of Fig. 1 to just four pairs (and force calculations) instead of 10, as originally required.

2.2 Linked Cell Method (LC)

An alternative approach that is often used to indentify the neighbors of a body is the Linked Cell (LC) method, where the system is divided into a lattice; for cubic systems, this can be $m \times m \times m$ cells (3D) [1]. The optimal size of the cells, as before the size of the Verlet spheres in the VL method, depends on the velocity of the particles and on the density of the system. As a rule, the cell sizes all need to exceed at least the size of the largest particle. The major difference between

LC and VL is, that for LC the cells are not attached to particles and thus are not moving. The size of the cells can be selected such that one cell contains about three particles, see Fig. 2. Then, if particles are temporarily assigned to a certain cell 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 [16]. This means for a 2D system, that only particles within nine different cells for 2D and 27 cells for 3D may contain neighbors. Since particle pairs have to be checked only once, not all cells have to be tested, but only the center cell plus half of the neighbor cells. That means 1+4=5 cells in 2D, Fig. 3, or 1+13=14 cells in 3D, see Fig. 3. Cast into a formula, one has $(3^d + 1)/2$ cells to examine for possible neighbors.

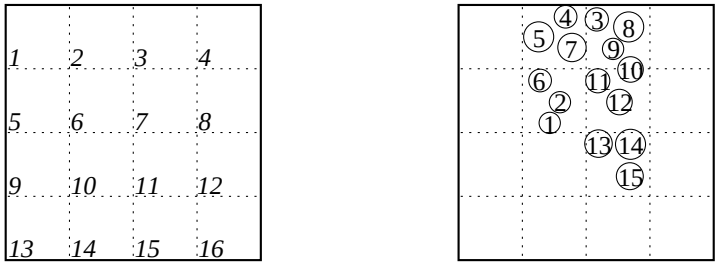

Figure 2: Linked cells numbering and particle numbering of a 2D system.

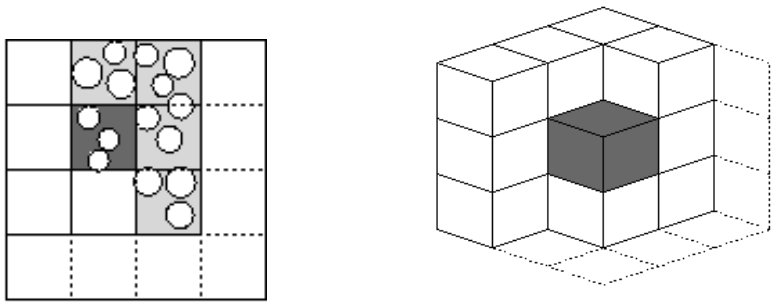

Figure 3: (Left) Cells that need to be investigated (grey) for the neighbor list of bodies in cell 6 (dark grey) for the example in Fig. 2. (Right) Neighboring cells to be investigated for a particle in the dark grey cell in 3D.

For a 2D system, see Fig. 2, the neighbor data-structure (NDS) for body 1 contains the particles from 2 up to 15. For particle 2 all particles of the neighboring cells except number 1 are neighbors and for particle 6 only particles of the four neighboring cells are neighbors. This is due to the fact that also within a cell the same strategy is applied in order to avoid multiple checks of the same particle pair.

In a d-dimensional system, on average, one has $n_c = n/m^d$ particles in each cell, with n, the number of bodies in the whole system and m, the number of

cells in each direction. Therefore, as an estimate, only

$$p \approx \left[\left(3^d - 1\right) \frac{{n_c}^2}{2} + \frac{n_c(n_c - 1)}{2} \right] m^d = \left[3^d \frac{{n_c}^2}{2} - \frac{n_c}{2} \right] m^d \tag{3}$$

distances (contact conditions) need to be examined (for the majority of cells that are not situated along the edge of the system). The first term takes neighbors outside into account, while the second those inside the cell of the particle of interest. The factors 1/2 are due to the fact that every pair shall be checked only once. Since the bracket in Eq. (3) equals the approximate number of necessary operations for one cell, it has to be multiplied by the number of cells in the system, m^d. An estimate for the average number of pair checks was given already in [1] as: $p \approx 4.5 n n_c$, for 2D, almost identical to Eq. (3) for large n_c. For each particle, in half of the nine cells approximately n_c particles will be neighbors. This contrasts with $p = \frac{1}{2} n(n-1)$, which is of order $O(n^2)$ if no neighborhood search method is used. For systems with $m \leq 3$, there is no benefit in using any NDS, but usually the systems with many particles possess much more cells.

2.3 Linked Linear List (LLL)

The third possibility (quite different from the other approaches) to keep track of neighbors for large systems, is the Linked Linear List (LLL) [18]. In a first step, bounding boxes are arranged around each particle, see Fig. 4, sized such that each particle fits exactly in its box, and the edges are aligned parallel to the system axes.

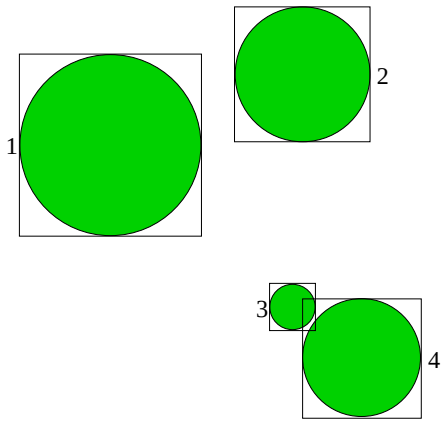

Figure 4: Bounding boxes around each particle.

Linear List Generation

In a second step, the bounding boxes are projected separately onto the system axes. Such a projection onto the x-axis for the situation in Fig. 4 is shown in

Fig. 5. In the following, only the order of the beginnings 'b' and endings 'e' of the projections of the bounding boxes along the axes are of interest. For this reason the sequences are stored in linear lists.

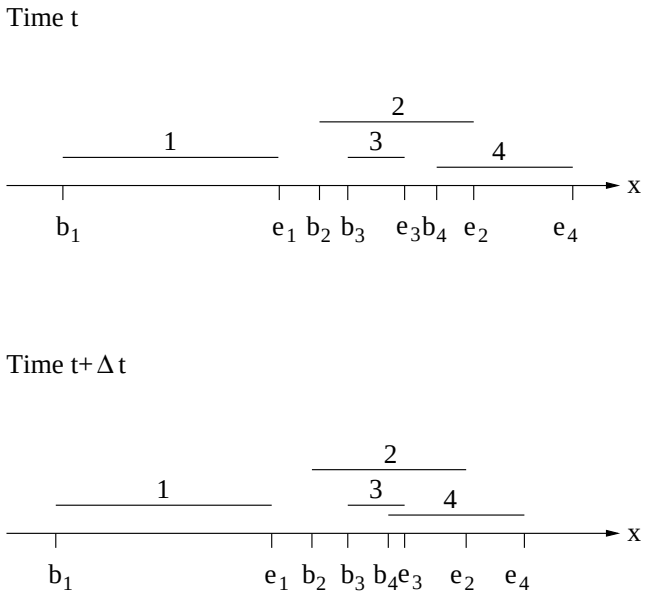

Figure 5: Particles projected on the x-axis for two different times, [18].

For a 3D system, three different projections are necessary and, therefore, three lists will be generated. Each of them has length $2n$, corresponding to twice the number of particles in the system. If there is either beginning, ending, or both, of another particle in between the beginning and ending of a certain body, then there will be an overlap of the projections of their bounding boxes along this axis. A collision of two bounding boxes exists for an overlap of these projections along all three axes.

Update of the Linear Lists

Checking whether there is some part of a projection in between the beginning and ending of another projection for each particle along each axis still takes a lot of time. But, although these lists have to be updated for each time step, the necessary calculation times can be reduced to an amount proportional to the total number of particles in the system, as there has to be done only an update of the old list for each new time step. This corresponds to sorting an already nearly sorted list: The update can simply be done by going through the lists sequentially and checking for changes in the order that are usually permutations, compare for example Fig. 5, where e_3 and b_4 are exchanged. If the order of the beginnings and endings does not have to be changed, the collision status of the particles also will remain unchanged. While seeking for new colliding bounding

boxes by looking for permutations in the lists, four different cases have to be considered, compare [18].

1. Two beginnings are changed, which means the bounding boxes have been overlapping and continue to overlap.

2. Two endings are changed, which also means the bounding boxes have been overlapping and continue to overlap.

3. A beginning and a proximate ending are changed, which means a so far occurring overlap has to be removed.

4. Both ending and following beginning of another particle are exchanged, which means a previously non-existing overlap has to be taken into account.

For the first two cases, except for the exchange, nothing has to be done in the 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 an old collision along all axes. If there is no overlap any more between two so far colliding bounding boxes in at least one direction, the collision has to be removed. Therefore, two (three) more columns in 2D (3D) are added to the lists that store the information of the positions of beginnings and endings along the axes, see Fig. 6.

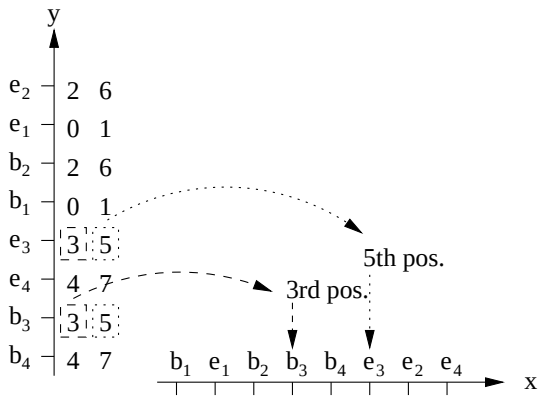

Figure 6: Lists containing also the position information along the other axes.

For our 2D example going through the list along the y-axis, see Fig. 6, 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 three to five, whereas the beginning of particle 4 has the position four, there is also an overlap of bounding boxes 3 and 4 along the x-axis and, therefore, a real 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 overlap. 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. 7.

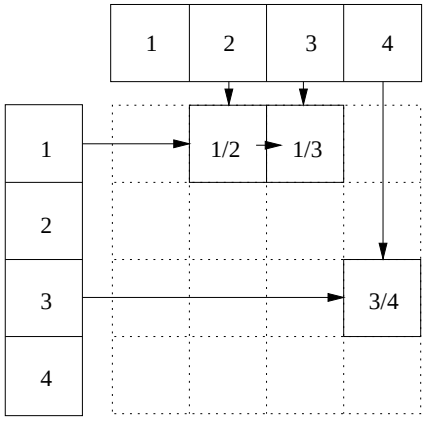

Figure 7: Storage of colliding bounding box pairs, if e.g. boxes (1/2), (1/3) and (3/4) collide.

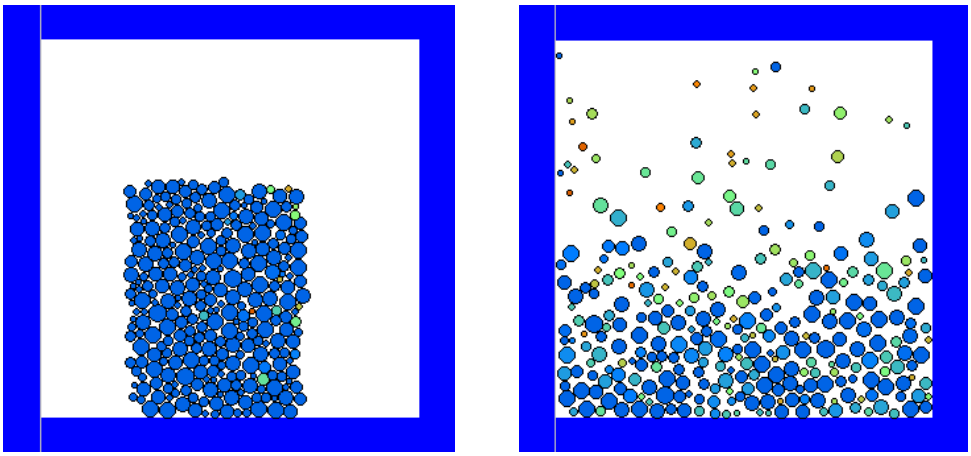

Figure 8: System consisting of 300 particles for two different points of time. In the initial situation (Left) all particles float with zero velocity, while the later situation (Right), after a simulation time of 2 sec, the particles are accelerated by gravity and have collided and interacted. This simulation is without dissipation and friction.

Figure 8 shows an example of a 2D system and Fig. 9 the matrix structure that is obtained. Since the particle numbers are initially sequential, the entries in the matrix of Fig. 9 (Left) are close to the diagonal, showing that only neighboring bodies have colliding bounding boxes. Note that this is only true for the special

numbering, that we use at the beginning of our simulations. Nevertheless, the matrix is quite sparse always, see Fig. 9 (Right).

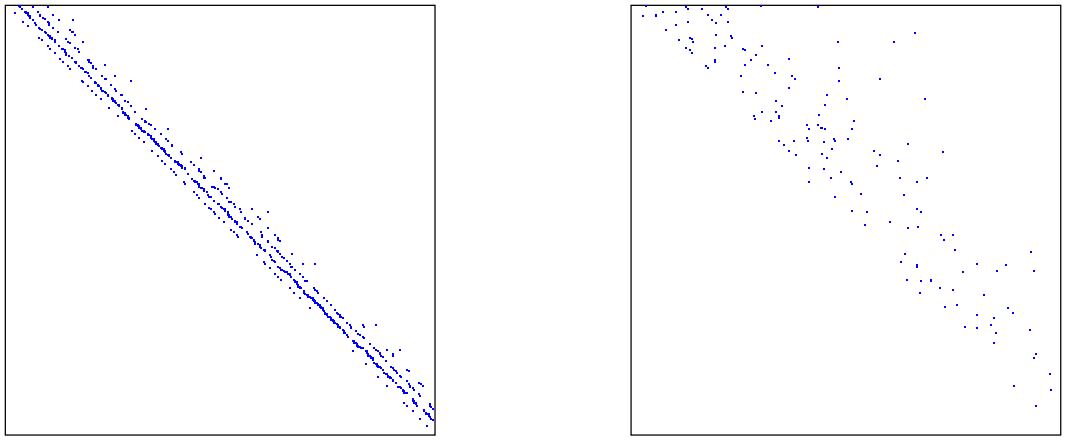

Figure 9: Matrix structure for the two snapshots from Fig. 8, where the dots are the potential contacts, i.e., box-collisions.

As collisions between the bounding boxes of two particles are only treated once, in Fig. 9, an upper triangular matrix is shown, where the diagonal of the matrix is empty.

2.4 Discussion

The two techniques VL and LC, described in secs. 2.1 and 2.2, have in common that both identify neighboring particles by considering bodies inside certain volumes as neighbors. For both methods these zones have to be at least somewhat larger than the particles themselves. From this, two problems can arise. Firstly, if the particles within the system are polydisperse, i.e., their sizes differ, 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 be close to n in the worst case [18].

As the neighborhood zones around a particle are larger than the particles, the NDS 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 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 Results

The three techniques introduced shall in the following be compared with respect to the simulation times needed as function of n. Differnt tests are used and the results are scaled, in order to keep the influence of different computers, of different programming languages and compilers, and from different programming styles, as small as possible. (The program for the LC was programmed in C++ while the VL and LLL were programmed in C and different hardware and operatinig systems were used.) The goal is not an absolute comparison, but the understanding of the n-dependence of the three algorithms.

3.1 Comparison of a Planar Polydisperse Example

In the first 2D test series, examples with different numbers of particles are studied. The system is polydisperse, i.e. the diameters of all particles are different. The sizes were randomly drawn from a homogeneous distribution in the interval $[R_0(1-w_0), R_0(1+w_0)]$, with mean radius $R_0 = 10^{-3}$ m and width $w_0 = 0.5$. Other system parameters are the stiffness for the calculation of the (penalty) repulsive force $k = 10^7$ N/m, the dissipative constant $d = 0$ Ns/m, the density of the particles $\rho = 7000$ kg/m^2, and the time step for the integration of $dt = 10^{-7}$s. The attempt was to keep the density of the system constant, while simultaneously the number of particles was duplicated stepwise and the system size was enlarged accordingly, see Fig. 10. The simulation consisted of the particles falling under gravity and spreading homogeneously over the system.

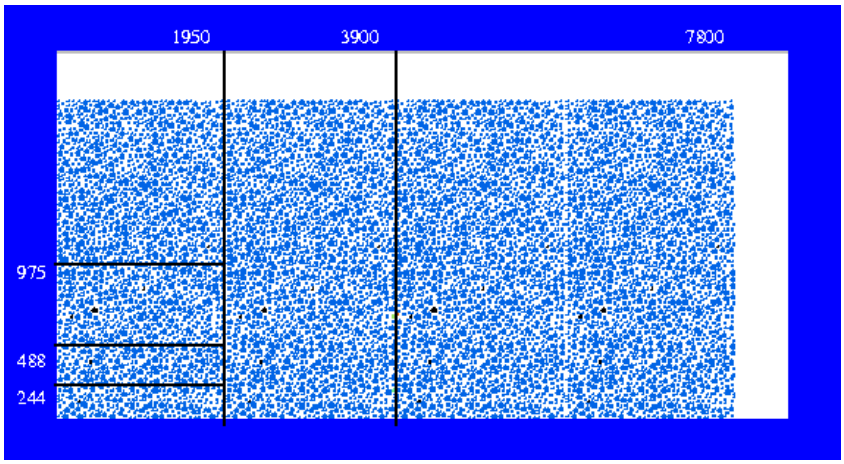

Figure 10: Example of a dense system with 7800 particles. The lines show the arrangement of the particles for all six system sizes.

Keeping the system density for each example exactly constant was not possible, since the container size divided by the linked cell size always has to be integer and the linked cell size for the whole series was kept constant in order to allow

for a comparison, i.e., $l_c = 7.0 \times 10^{-3}$ m. The first series consists of 244, 488, 975, 1950, 3900, and 7800 particles. The computation times with respect to the increasing number of particles are shown in Fig. 11.

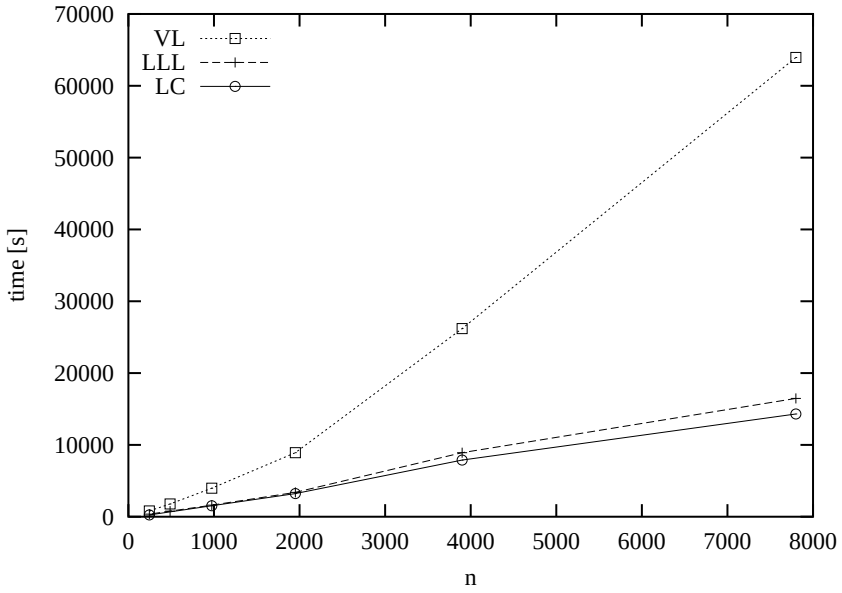

Figure 11: Comparison of the results for the 2D example.

From this picture it is quite clear, that the curve for the Verlet method is the steepest, increasing non-linearly with n, whereas both other curves display very similar, almost linear behavior. All three curves show a jump between 1950 and 3900 particles. However, the reason for this are changes in the density of the system, which varies from $\nu_{cont} \approx 0.670$ to 0.678.

3.2 Monodisperse Example in 3D

A series of 3D systems with an increasing number of particles of equal size, $R = 5.0 \times 10^{-4}$ m, and a proportionally growing space around the particles is investigated in this subsection. The stiffness of the particles was chosen as $k = 10^5$ N/m, the damping coefficient again was neglected, the density of the particles was $\rho = 10^{10}$ kg/m^3 (the units are arbitrary here, the choice was to deal with masses of order unity), and the time step for the integration was chosen as $dt = 10^{-5}$ s. Here, systems with 512, 2197, 5832, 12167, 21952, 35937, and 59320 particles are investigated, where again the linked cell size was kept constant, $l_c = 2.0 \times 10^{-3}$ m. The behavior of the computation time with respect to the number of particles within the particular system is shown in Fig. 12 as log-log plot.

With increasing number of particles (and with it the number of contacts during the simulation), the worse VL performs in comparison to LLL and LC. Here, LC has the best performance: the gradient is lower than for LLL. The

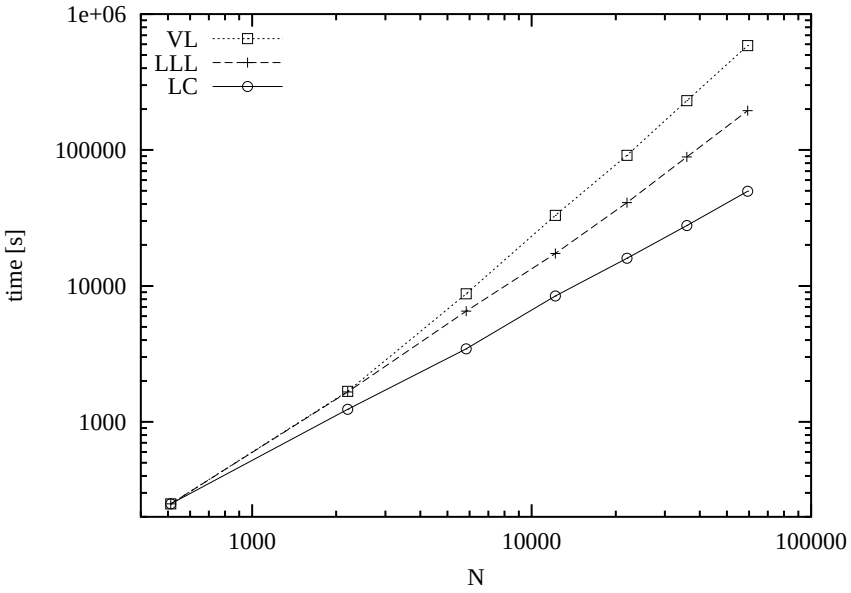

Figure 12: Comparison results for an increasing 3D example.

curves can be fitted by power-laws: for VL one has $y \approx (0.027\ x)^{1.8}$, for LLL one has $y \approx (0.12\ x)^{1.36}$ and for LC one obtains $y \approx (0.2\ x)^{1.15}$. The dependency of LLL and LC on n is close to linear in contrast to the behavior of VL, which behaves almost quadratic with n.

3.3 Polydisperse Example in 3D

In the last 3D comparison the system size is kept constant, but the number of particles is increased. Here, no change of either the system or cell-size is applied, keeping $l_c = 0.033$ m fixed. Chosen parameters for this system are the stiffness $k = 4.0 \times 10^6$ N/m, damping coefficient $d = 0$ Ns/m, density of the bodies $\rho = 7000$ kg/m^3, and the time step for the integration $dt = 4 \times 10^{-7}$ s. The systems can be seen as a series of fracture of some of the particles, where neither the volume enclosed in the system nor its mass content are changed. Thus, the volume fraction and the density of the system are unchanged, while the number of particles 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: There are then about 500 "old" particles of radius R, but approximately 8×500 particles of radius $R/2$, each with one eighth of the original particle volume. The systems therefore contain

1000 particles (see Fig. 13 on the left),

4451 particles, about 4000 smaller bodies of $r = R/2$,

14676 particles, about 14000 particles of $r = R/3$,

32374 particles, about 31000 particles of $r = R/4$, and

65480 particles, about 65000 particles of $r = R/5$ (see Fig. 13 on the right).

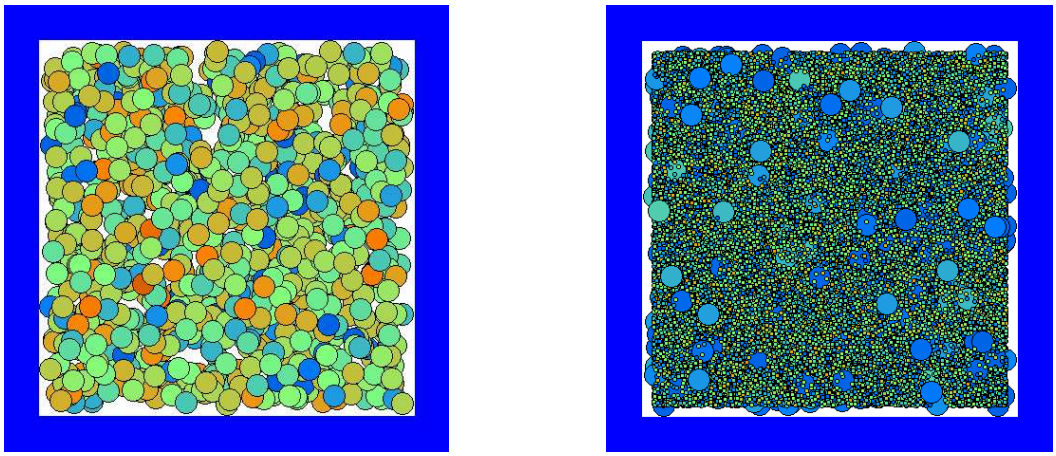

Figure 13: A monodisperse system and a very polydisperse system with $r = R/5$ of exactly the same volume and density (volume-fraction $\nu = 0.12$).

Here, R and r are the original radius of all particles and the size of the fractured particles, respectively. In Fig. 13 there are no overlaps between the bodies: an apparent overlap is due to the positions at different depths. For this system the computation time needed per particle is presented in Fig. 14 over the ratio of the radii that is equivalent to the polydispersity of the system.

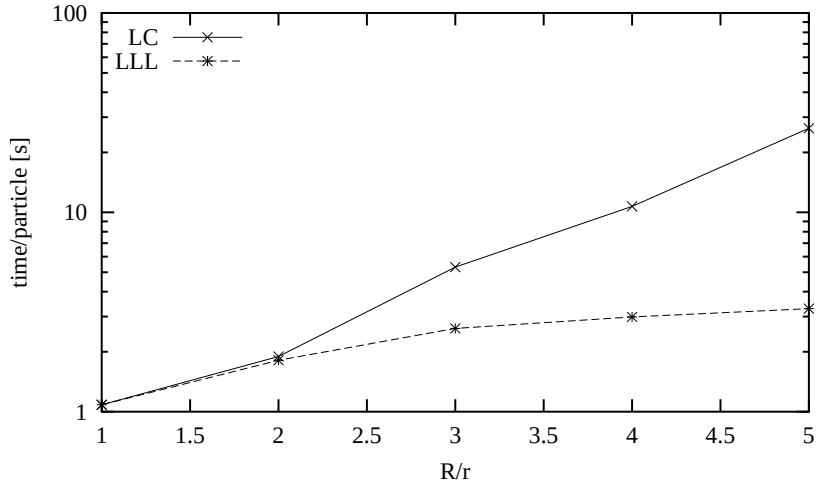

Figure 14: Computation time for a polydisperse system of constant density.

It can be seen that the gradient of the curve over the polydispersity of the LC calculation is a lot higher than for LLL. Before scaling, so that both curves

have the same starting reference point, LC was faster than LLL, and the curves were 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 Conclusions

In this paper three different methods have been introduced in order to reduce the calculation times 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 distant. The presented methods save a lot of time by excluding such particles from a detailed and time consuming contact examination and evaluation.

For rather small systems, the traditional and simple 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. The LC method shows very good performance with a lower increase of computing time for monodisperse systems, as compared to the LLL method. A problem using LC might be the optimal choice of the linked cell size. For too small sizes and for too large sizes the results become inefficient. An optimal cell size has to be found for each different system, since there is no general rule to our knowledge.

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 due to increasing n_c with increasing cell size, LC becomes very inefficient for wide size distributions. The time spent using LLL behaves much better, as compared to the LC, for polydisperse (partial fracturing) systems of constant volume and density but increasing numbers of particles.

Therefore, there is no unique "winner" of our comparison and for computations it is required to know the methods with their strengths and weaknesses in order to choose the most appropriate one.

In the future, it might be interesting to use LLL for polydisperse systems with even wider size distributions, [10]. Another interesting task will be the contact detection and force calculation of polygonal/polyhedral particles and the neighborhood search for strongly anisotropic particles. Finally, since the collision detection for concave particles is even more complicated and time consuming, the choice of the most appropriate neighborhood search algorithm will remain an issue of interest and requires further research.

References

[1] M. P. Allen and D. J. Tildesley. Computer Simulations of Liquids (Clarendon Press, Oxford, 1989).

[2] P. Eberhard. Kontaktuntersuchungen durch hybride Mehrkörpersysteme / Finite Elemente Simulationen, in German (Shaker, Aachen, 2000).

[3] Y. Kishino, Editor. Powders & Grains (Balkema, Rotterdam, 2001).

[4] B. Lubachevsky. How to Simulate Billiards and Similar Systems, Journal of Computational Physics 94 (1991) 255–283.

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

[6] S. Luding. Cohesive Frictional Powders: Contact Models for Tension. Granular Matter, (2007), in press.

[7] S. Luding. Die Physik trockener granularer Medien, in German (Logos Verlag, Berlin, 1998).

[8] S. Luding, M. Huthmann, S. McNamara, and A. Zippelius. Homogeneous Cooling of Rough Dissipative Particles: Theory and Simulations. Physical Review E 58 (1998) 3416–3425.

[9] F. Melzer. Symbolisch-numerische Modellierung elastischer Mehrkörpersysteme mit Anwendung auf rechnerische Lebensdauervorhersagen, Fortschritt-Berichte VDI, Reihe **20**, Nr. 139 (VDI Verlag, Düsseldorf, 1994).

[10] B. Muth, G. Of, P. Eberhard, and O. Steinbach. Collision Detection for Complicated Polyhedra Using the Fast Multipole Method or Ray Crossing, Archive of Applied Mechanics 77(7) (2007) 503–521.

[11] B. Peters and A. Džiugys. Numerical Simulation of the Motion of Granular Material Using Object-Oriented Techniques, Computational Methods in Applied Mechanics and Engineering 191 (2002) 1983–2007.

[12] F. Pfeiffer. Multibody Dynamics with Unilateral Contacts (Wiley, New York, 1996).

[13] J. Pfister and P. Eberhard. Frictional Contact of Flexible and Rigid Bodies, Granular Matter 4(1) (2002) 25–36.

[14] T. Pöschel and V. Buchholtz. Static Friction Phenomena in Granular Materials: Coulomb Law vs. Particle Geometry, Physical Review Letters 71(24) (1993) 3963.

[15] T. Pöschel and S. Luding, Editors. Granular Gases, Lecture Notes in Physics 564 (Springer, Berlin, 2001).

[16] D. C. Rapaport. The Art of Molecular Dynamics Simulation (Cambridge University Press, Cambridge, 1995).

[17] W. Schiehlen, and P. Eberhard. Technische Dynamik, in German (B.G. Teubner, Wiesbaden, 2004).

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

[19] P. A. Vermeer, S. Diebels, W. Ehlers, H. J. Herrmann, S. Luding, and E. Ramm, Editors. Continuous and Discontinuous Modelling of Cohesive Frictional Materials, Lecture Notes in Physics 568 (Springer, Berlin, 2001).

[20] L. Vu-Quoc, X. Zhang, and O.R. Walton. A 3-D Discrete-Elemente Method for Dry Granular Flows of Ellipsoidal Particles, Computational Methods in Applied Mechanics and Engineering 187 (2000) 483–528.