Effective Algorithm for Detection of a Collision between Spherical Particles

Jacek S. Leszczynski and Mariusz Ciesielski

Czestochowa University of Technology, Institute of Mathematics & Computer Science, ul. Dabrowskiego 73, 42-200 Czestochowa, Poland
{jale,cmariusz}@k2.pcz.czest.pl

Abstract. In this work we present a novel algorithm which detects contacts between spherical particles in 2D and 3D. We estimate efficiency of this algorithm throughout analysis of the execution time. We also compare our results with the Linked Cell Method.

1 Introduction

The dynamics of granular materials is characterised by particles which move under arbitrary extortion and interact with each other. Many properties of granular materials are still under investigations, especially convection, segregation, granular flows, ability to clusterisation, etc. Therefore computer simulations become an interesting tool which can develop physics and engineering groups. In discrete approaches, such as the molecular dynamics method and the event-driven method we need to detect particle collisions in order to add additional conditions which arise during the collision. Moreover collision detection has also many practical applications, e.g. in modelling of physical objects, in computer animations and in robotics.

The mechanism of collision detection involves time of calculations and mutual locations of contacting objects. Especially we need to detect the begin time of a contact which issues from the precise detection of two contacting points in the colliding objects. Each object is characterised by some shape and this is the main reason that to apply a convenient algorithm for the collision detection.

The general way of collision detection includes an information whichever the geometrical contact occurred. Above problem in the objective of study in papers [4,7,8] where different mechanisms of collision detection are investigated. The key aspect in the existing algorithms is how to detect the collisions in the fast way that to reduce the computational time.

In this paper we will focus on the two algorithms applied for the collision detection in 2D and 3D where particles have simple circular or spherical forms. The first algorithm called the Linked Cell Method [1,2,3,4,7,8] assumes space division into a regular lattice. Within the lattice one try to find particle collisions. We will present here a novel algorithm in order to reduce the computational time. The novel algorithm involves another way of collision detection in comparison

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3037, pp. 348–355, 2004.
© Springer-Verlag Berlin Heidelberg 2004

to the Linked Cell Method. In the next sections we will explain some details of the algorithm especially for spherical particles.

The spherical shape of the particles makes only the mathematical description easier but does not limit collision detection to be in any way less meaningful. The reader may find more informations in [4,7] concerning collision detection adapted to an arbitrary form of particle shapes. In other words, the detection of collision for arbitrary shapes of particles may decompose into the collision detection of particles having spherical shapes (the spherical shape is generated on the particle having the arbitrary shape) and within the spherical shapes one try to find collision for arbitrary shapes of the particles.

In this paper we will focus on the molecular dynamics method [5] where during motion of particles the contacts may eventually take place. This method takes into consideration an expression for the repulsive force acting between a pair of contacting particles. Within the molecular dynamics particles virtually overlap when a collision occurs. The overlap reflects the quantity important deformations of particle surfaces. Before the application of the repulsive force in motion equation we need to detect a contact between two particles. Therefore we introduce particles numbers by the index $i = 1, \ldots, np$ where np is assumed as the total number of considered particles. We also introduce the natural function $j(i) \neq i$, of a particle i in order to find the particle index in a set of particles. According to [5] we define the overlap of two spherical particles being in a contact as

$$\left\|\zeta_{j(i)}\right\| = r_{j(i)} + r_i - \left\|\mathbf{x}_{j(i)} - \mathbf{x}_i\right\| \tag{1}$$

where r_i, $r_{j(i)}$ are particle radiuses, $\left\|\cdot\right\|$ is the norm representing relative distance between the mass centre of particles. When a contact happens over time of calculations we need to consider $\left\|\zeta_{j(i)}\right\| = 0$. On the other hand, when $\left\|\zeta_{j(i)}\right\| \geq 0$ we detect also a contact between two particles.

Considering a system of particles np we take into account many arithmetic operations in order to check particle contacts. However, in order to check all possible contacts between a pair of particles one has to calculate $np(np-1)/2$ arithmetic operations. This is very simple algorithm which is time consuming, especially when total number of considered particles np is large. Unfortunately, above algorithm is practically unacceptable. In computer simulations one uses algorithms which analyse lists of neighbouring particles or the lists of particle groups.

In this paper we will consider the Linked Cell Method as well as our novel algorithm for the collision detection. We shall present simulation results taking into account efficiencies of two methods.

2 Algorithms for Detection of Collisions

2.1 Linked Cell Method

The Linked Cell Method is very efficient [1,7,8] in practical simulations where the total number of particles is large. Within this method one divides the considered

area of space into a regular lattice where n_{cell} represents the total number of cells. For 2D problems we have $n_{cell} = m_1 \cdot m_2$ but for 3D problems we consider $n_{cell} = m_1 \cdot m_2 \cdot m_3$ where m_k is the division number of space in the direction k. Moreover we can calculate dimensions of an individual cell as $a_k = L_k/m_k$ where L_k are global dimensions of the considered space. We also take into consideration a relationship between the cell dimensions and particle dimensions as

$$\min_k(a_k) \geq d_{\max} \tag{2}$$

where $d_{\max}$ is maximal diameter of a particle in a set of considered particles np. The averaged particle number in each cell is $p_{avg} = np/n_{cell}$.

Implementation of the cell structures algorithm discuses in details in [1,8]. The first step of this implementation includes searching and sorting procedures of particle positions in order to find place in the appropriate cell. In the procedures the mass centre of particle defines the particle position in the cell. We call this step of calculations as grouping of particles into the cells structures. In this step two matrices are used. The first header matrix (which has the total number of elements n_{cell}) and the one-dimensional linked-list matrix (the dimension of this matrix is np) are necessary for storage of particle indexes in the appropriate cells. This way of storage is very convenient and optimises computer memory. Even we predict time t_{cell} in which one particle finds a place in the cell, the total time associated with all considered particles occuping place in cells is $np \cdot t_{cell}$. The second step of the Linked Cell Method involves detection of contacts for particles occuping space in neighbouring cells connecting with the considering cell. In this step we check all possible contacts in a simple way - every particle with every other ones. The number of possible contacts checken between several pairs of particles is determined as $4.5 \cdot np \cdot p_{avg}$ in 2D and $13.5 \cdot np \cdot p_{avg}$ in 3D - instead of the brute-force approach where we have $0.5 \cdot np \cdot (np - 1)$. Assuming time t_{test} used to check one pair of contacting particles we calculate time necessary for execution of the algorithm as $t_{sim} = np \cdot t_{cell} + 4.5 \cdot np \cdot p_{avg} \cdot t_{test}$ in 2D or $t_{sim} = np \cdot t_{cell} + 13.5 \cdot np \cdot p_{avg} \cdot t_{test}$ in 3D. The shortest value of execution time t_{sim} issues from the smallest value of the mean number of particles occuping space in one cell p_{avg}. However in prediction of the parameter p_{avg} we need to take into account expression (2) which has direct influence on the total number of cells n_{cell}.

2.2 New Method which Detects Particle Contacts

In this section we will propose a new algorithm which indicates particle contacts. We assume that particle shape has the spherical form. Fig. 1 shows same details of the algorithm with useful notations. For the contact calculations we need to input the following data: the total number of particles np, particle radiuses r_i and particle positions $\mathbf{x}_i$ of the mass centre, for $i = 1, \ldots, np$. First step of the algorithm basis on calculations of distances l_i between the begin of system coordinates $\mathbf{x}_0$ and the point of a particle which lies on the sphere at the particle. In solving this problem generally, the following algorithm is presented:

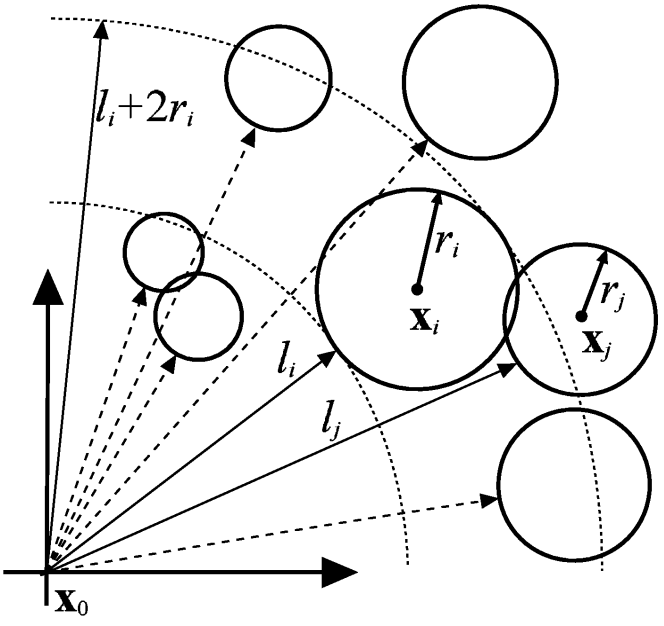

Fig. 1. Scheme illustrates our algorithm with useful notations.

Algorithm 1

Step 1. Introductory calculations:
- right choose of point $\mathbf{x}_0$ which establishes the begin of system coordinates (It means that all particle positions should have the same sign, positive or negative, in the system of coordinates.),
- calculations of the distances l_i in the form

$$l_i = \left\|\mathbf{x}_i - \mathbf{x}_0\right\| - r_i. \tag{3}$$

Step 2. The distances l_i are sorted by an arbitrary sorting algorithm, e.g. QuickSort [6]. (As a direct result of this sort we obtain a matrix nl in which several particle indexes are included.),

Step 3. Searching and detection of contacts:

```
for i = 1 to np − 1
  for j = i + 1 to np
    dist = l_{nl_j} − l_{nl_i}
    if 2 · r_{nl_i} < dist then
      break
    else
      ‖ζ‖ = r_{nl_i} + r_{nl_j} − ‖x_{nl_j} − x_{nl_i}‖ //overlap caclulations
      if ‖ζ‖ ≥ 0 then
```

detect_a_contact(nl_i, nl_j) // contact between particles nl_i and nl_j
endif
endif
end
end.

Presented algorithm requires to reserve memory in order to store two one-dimensional matrices where the each dimension equals to np. In this case we have a matrix of indexes nl and a matrix with the particle distances l. We estimate time t_l in which the one distance l_i is calculated and time t_{sort} necessary for the sorting procedure, and time t_{test} is a time where a pair of particle distances is checken. In t_{test} we have to calculate $\left\|\zeta_{j(i)}\right\|$ by formula (1). This is the arithmetical operation which is time consuming. Sumarising all times we obtain time t_{sim} which reflects same cost of the algorithm $t_{sim} = np \cdot t_l + t_{sort} + nt \cdot t_{test}$. In this expression nt indicates particle pairs which are eventually in the contact. The value nt, being depended on values r_i and L_k, decreases when particle concentration decreases in the considered volume. This happen when we consider the large volume and small diameters of particles.

The algorithm has the following advantages:

- it uses small amount of computer memory in order to storage data,
- it checks contacts locally being dependent on choose of the begin point $\mathbf{x}_0$ of system coordinates,
- structure of the computational code is very simple,
- it is independent on space dimension. (It means that we find contacts in the same way for both 2D and 3D problems. Note that at the preparation stage the norm of relative distance between the mass centre of particles differs in 2D in comparison to 3D.)

However, we need to take into account that efficiency of the algorithm represented by t_{sim}, decreases for dense packing of particles. This is disadvantage at the algorithm.

3 Simulation Results

On the base of previous section we perform computer simulations for detection of particle contacts. In this section we will try to compare results of simulation obtained form the Linked Cell Method and of course from our algorithm. The basic indicator for such a comparison is time necessary for detection of all contacts t_{sim} which is registered for above algorithms operating in the same initial date.

To prepare the initial data we generate particle diameter and particle position randomly. Nevertheless we assume a range of variation of the particle diameter. Therefore we prepare three the following test:

$$\begin{array}{ll} A. & d \in < 0.05, 0.1 >, \\ B. & d \in < 0.02, 0.05 >, \\ C. & d \in < 0.005, 0.02 > . \end{array} \tag{4}$$

With regard to particle location we assume a rectangular box in 2D and a cubic box in 3D where the box dimension is nx. We generated randomly particle positions $\mathbf{x}_i$ within the box. The parameter nx is assumed to be $np/nx^2 = 100$ for 2D problems and $np/nx^3 = 1000$ for 3D problems. For presented initial data we calculate a ratio which is the number of contacts eventually happen to the number of particles occuping space in the box. Taking into account three assumptions presented by formula (4) we obtain several results of the ratio as $A - 87\%$, $B - 19\%$, $C - 2\%$ for 2D and $A - 90\%$, $B - 10\%$, $C - 4\%$ for 3D.

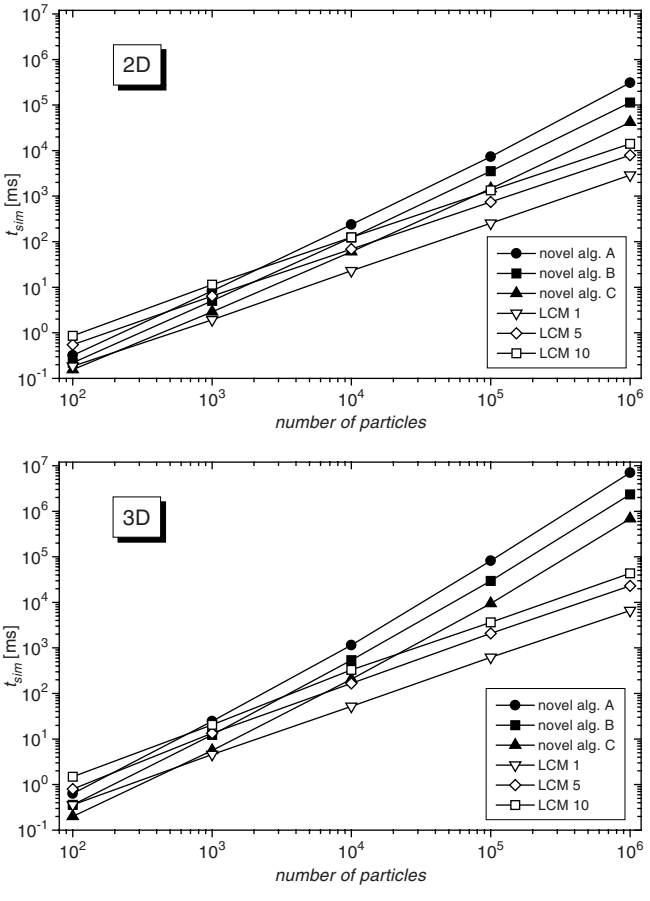

Fig. 2. Simulation results of the execution time t_{sim} over the number of considered particles for both 2D and 3D.

Fig. 2 shows the execution time t_{sim} consumed by the algorithms over the number of considered particles in the box. We taken into account this time

Table 1. Comparison of the execution time for 10000 particles and $d \in \langle 0.02, 1.0 \rangle$

	novel alg.	LCM100	LCM204	LCM625	LCM1111
t_{sim} $[s]$	1.627	1.141	2.296	8.828	12.332

Table 2. Comparison of the execution time for 100 particles and $d \in \langle 0.1, 2.0 \rangle$

	novel alg.	LCM11	LCM25	LCM100
t_{sim} $[s]$	0.37	1.01	1.4	1.59

Table 3. Comparison of the execution time for 100 particles with dimensions $d \in \langle 0.1, 0.3 \rangle$ and one big particle $d = 5.64$

	novel alg.	LCM101
t_{sim} $[s]$	0.197	1.38

calculated for both 2D and 3D problems. Open symbol connected with a line represent results obtained by our algorithm. For the Linked Cell Method we performed simulations being dependent on the averaged number of particles p_{avg} in one cell. In this case we calculated the number of lattice cells n_{cell} that to obtain the averaged number of particles in one cell as 1, 5 or 10 particles per one cell. Fig. 2 shows above results are representing by indicators LCM1, LCM5 and LCM10. We can observe that the number of particles occupying space in one cell is significant for the execution time in LCM. Nevertheless, we cannot observe this in our algorithm. The number of particles occupying space in one cell can increase in calculations when we consider a situation for particles differing more by their dimensions. This could happen for one big particle and a lot of small particles. To show this disadvantage in LCM we perform the following tests in 2D. Table 1 shows the execution time for 10000 particles differing by sizes $d \in \langle 0.02, 1.0 \rangle$. When the number of particles increases in one cell (LCM100, LCM204, LCM625, LCM1111) the execution time also increases in comparison to our algorithm, where this time does not change. Table 2 presents similar case to the previous one but particle dispersions is larger. It should be noted that greater dispersion in particle sizes influences significally to extension of the execution time calculated by LCM. The interesting case is also presented by Table 3. In this case we considered one cell with 100 small particles differing by sizes in the range of $d \in \langle 0.1, 0.3 \rangle$ and one big particle with dimension $d = 5.64$. We can observe that the execution time increases up to $1.38\,s$ for LCM in comparison to our algorithm where $t_{sim} = 0.197\,s$.

4 Conclusions

When we consider results presented by Fig.2, we can notice that the averaged number of particles p_{avg} in the Linked Cell Method has direct influence on the

execution time t_{sim}. For this method we decrease the value of the parameter p_{avg} the execution time t_{sim} is decreased too. For our algorithm we confirm that the execution time t_{sim} decreases when the averaged diameter of a particle in the particle distribution decreases too. Moreover in our algorithm we do not observe any influence of particle dispersions in the particle distribution on the execution time t_{sim}. However, some influence of this dispersion are noted in the Linked Cell Method. In direct comparison of the execution time being a result of above methods we state that the Linked Cell Method presents better results for dense number of particles in considered space. However the particles should not have large dispersion of diameters in the particle distribution. In opposite to previous method, our algorithm is more suitable and flexible (it generates smaller values of the execution time in comparison to the Linked Cell Method) when particle concentration in the considered volume is small and the dispersion of particle diameters in the distribution function may differ much more than in the Linked Cell Method.

Acknowledgment. This work was supported by the State Committee for Scientific Research (KBN) under the grant 4 T10B 049 25.

References

1. Allen M.P. and Tildesly D.J.: Computer Simulation of Liquids, Clarendon, Oxford, (1987)
2. Hockney R.W., Eastwood J.W.: Computer Simulation Using Particles, McGraw-Hill, New York (1981)
3. Iwai T., Hong C.W., Greil P.: Fast particle pair detection algorithms for particle simulations, International Journal of Modern Physics C **10** (1999), pp. 823-837
4. Muth B., Müller M.-K., Eberhard P., and Luding S.: Collision detection and administration for many colliding bodies, submitted (2002)
5. Pourin L., Liebling Th.M.: Molecular-dynamics force models for better control of energy dissipation in numerical simulations of dense granular media, Physical Review E, **65** (2001) 011032 pp. 1-7
6. Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P.: Numerical Recipes in C/Fortran, Cambridge University Press, (1992)
7. Schinner A.: Fast algorithms for the simulation of polygonal particles, Granular Matter **2** (1999) 1, pp. 35-43
8. Tijskens E., Ramon H., de Baerdemaekerm J.: Discrete element modelling for process simulation in agriculture, Journal of Sound and Vibration **266** (2003), pp. 493-514