Towards one billion particle systems

A. Munjiza*, E. Rougier, N.W.M. John

Department of Engineering, Queen Mary, University of London, London E1 4NS, UK

Abstract

Two new linear contact detection algorithms are developed. The first algorithm is suitable for particles of near spherical shape and near constant size. The second algorithm is suitable for particles of near spherical shape greatly differing in size. Both algorithms are insensitive to packing density in terms of CPU and random access memory (RAM) requirements. The data structure is very simple and RAM requirements are small enabling theoretically billions of particles to be addressed on a desktop machine.

Keywords: Discrete elements; Particulates; Contact

1. Introduction

Large discrete element systems usually comprise millions of discrete elements [1,2,3]. However, there are systems where the physics of the problem requires billions of discrete elements [4]. In addition, although very often discrete elements may be of very simple shape, such as spherical discrete elements, most often they vary in size considerably with a variation of 1 to 100 not being unusual.

In other words, to move from a system with one million particles to one with one billion particles, contact detection [5,6,7,8] must be handled efficiently by minimizing the RAM requirements. Also, the size distribution of the particles should be taken into account in the contact detection algorithm while the CPU time is minimized and efficiency of the algorithm is made independent of spatial distribution of discrete elements. One could argue that nobinary search (NBS) and C-GRID algorithms [9,10] fulfil all of these criteria, with C-GRID algorithm also taking into account the shape of discrete elements. Unfortunatelly, this is not the case. In our quest for ever faster, simpler, more robust solutions, almost problem-specific solutions are needed in order to arrive at the most efficient data structure for a given problem. Thus, in this paper a robust solution for spherical particles of the same size is proposed in the form of a Munjiza–Rougier (MR) contact detection algorithm and this is combined with a robust solution for spherical particles of different size – multistep Munjiza–Rougier (MMR) contact detection algorithm. A short summary of both algorithms is provided in this paper, while the detailed description is published in a journal paper.

2. MR contact detection algorithm

The MR contact detection algorithm comprises of three parts:

(1) mapping of discrete elements onto identical cells;
(2) sorting of discrete elements according to the cell to which they are mapped;
(3) searching the sorted list of discrete elements for contacts.

The MR contact detection algorithm is based on the assumption that all the discrete elements in the system can be approximated by a sphere of diameter, d, which is obtained from the size of the largest discrete element in the system. The entire space is subdivided into identical cubical cells of size d with each cell being identified by a triplet of integer numbers (i_x, i_y, i_z). The mapping of the set of spherical bounding boxes onto the set of cells is defined in such a way that each discrete element is assigned to one and only one cell. Representation of the mapping is achieved through using a double connected closed linked list of bounding boxes. In order for the list to represent the spatial distribution of the bounding boxes, the list is sorted according to a criterion that has spatial meaning. This is done using a novel MR-linear sort algorithm. The MR-linear sort algorithm is based

* Corresponding author. E-mail: a.munjiza@gmul.ac.uk

Computational Fluid and Solid Mechanics 2005
K.J. Bathe (Editor)

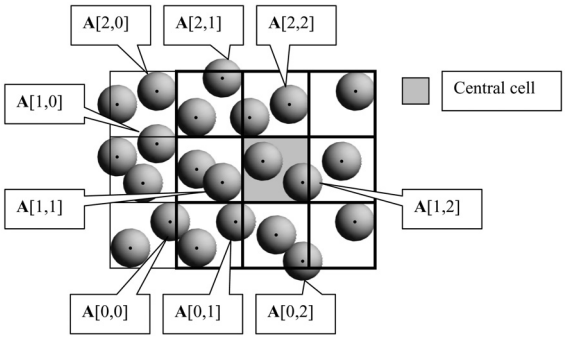

Fig 1. Relationship between matrix **A** and bounding boxes in neighbouring cells.

on the assumptions that no discrete element can move more than the size of a single cell in a given time step. This is a reasonable assumption as it is impossible for a given discrete element to move such a great distance within a single time step. The MR-linear sort algorithm takes advantage of the fact that the list is nearly sorted. The list is parsed starting from the list head (LH), which has integerized coordinates (0, 0, 0), and each bounding box is checked against its previous one. If the bounding box is greater than its previous one, then no change is needed. The parsing of the list continues until a bounding box that is smaller than its previous one is found. When this happens, the current bounding box needs to be relocated into the appropriate place in the list. To be able to easily move such a bounding box, a 3 by 3 matrix, **A,** of pointers to the bounding boxes immediately before each of the neighbouring cells of the current bounding box is used, i.e. the biggest bounding box in the list that is smaller than the neighbouring cell under consideration, see Fig. 1. The list is parsed only once and all the pointers are advanced forward only. From this it follows that the theoretical CPU time for the MR-linear sort algorithm is given by:

$$T \propto N \tag{1}$$

Detection of contact is done by checking all the discrete elements mapped to a particular cell against all the discrete elements in neighbouring cells, because only discrete elements from neighbouring cells can touch each other. To avoid repetition in the contact detection process, a contact mask is used for each central cell. The contact mask is divided into two rows (in 2D). Since the bounding boxes in the list are ordered, for each central cell and each row there is a beginning target bounding box and an ending target bounding box. In 3D space, the situation is similar, except that in this case instead of two rows in the contact mask there are five rows. The solution to the contact detection problem is therefore reduced to parsing the bounding boxes between beginning and ending boxes. As the list is parsed only once and all the pointers are advanced only forward, it follows that the theoretical CPU time for the MR-linear search algorithm is given by:

$$T \propto N \tag{2}$$

3. Multistep linear search algorithm (MMR)

The major drawback of the MR algorithm is the fact that all discrete elements are represented with bounding boxes of the same size. In addition, the diameter of the bounding box is such that the largest discrete element comprising the system is contained within a bounding box. The result of this is that the contacts are over-reported, i.e. MR algorithm reports a large number of contacts between small particles although these particles may be relatively far from each other and, therefore, not in contact. Due to this the performance of the algorithm deteriates with the square of the ratio between the size of the largest and smallest particle. The basic idea of the multistep-MR (MMR) algorithm is relatively simple and involves dividing particles into groups according to size:

Group 0: Particles represented by bounding box of size $d_0 = d$

Group 1: Particles represented by bounding box of size $d_1 = d/\alpha$; $\alpha > 1$

Group 2: Particles represented by bounding box of size $d_2 = d/\alpha^2$; $\alpha > 1$

Group 3: Particles represented by bounding box of size $d_3 = d/\alpha^3$; $\alpha > 1$

...

Group n: Particles represented by bounding box of size $d_n = d/\alpha^n$; $\alpha > 1$

(3)

MMR is implemented in n steps:

STEP 0: Initially all the particles are represented by the bounding box associated with group 0. All the bounding boxes are mapped onto the cells. It is worth mentioning that the size of the cells is equal to the size of the bounding box, i.e. d_0. Contact detection is performed using MR procedures in a usual way. However, separate lists are assembled for particles from group 0 and all other particles and contact search is performed for contact between particles from group 0 and particles from all other groups.

STEP 1: Particles of group 0 are first removed from the system. All the remaining particles are then represented by the bounding box of size d_1 and contact

detection is performed in a usual way, except that a separate list for particles of group 1 is kept and a search only performed for contacts between particles from group 1 and all other remaining particles.

STEP *i*: Analogy to step 1 is employed.

STEP *n*: During execution of step *n* only particles from group *n* remain in the system. The key characteristic of the MMR algorithm described above is that it uses MR algorithm to detect contacts between particles of different sizes. The ratio between the smaller and the largest particle depends on the total number of steps *n* and parameter α and is given by

$$\frac{d_{\max}}{d_{\min}} = \alpha^n \text{ for say } \alpha = 2 \text{ in 10 steps } \frac{d_{\max}}{d_{\min}} = 2^{10} \text{ is obtained.} \tag{4}$$

Random access memory (RAM) requirements of MMR are identical to RAM requirements of MR contact detection algorithm:

in 2D: M = 2*N* integer numbers
(two pointers per discrete element); (5)

in 3D: M = 2*N* integer numbers
(two pointers per discrete element);

in *n*-dimensional space:
M = 2*N* integer numbers
(two pointers per discrete element);

An important property of both MMR contact detection algorithm is that neither memory nor CPU requirements are a function of spatial distribution of discrete elements, thus the algorithm performs well for both loose and dense packs.

4. Numerical experiments

In order to demonstrate the above properties of MR and MMR-linear contact detection algorithms a set of numerical experiments on systems consisting of *N* discrete elements all of either constant diameter *d* (Fig. 2) or varying diameter that changes from 1 to 1/32 according to a uniform (linear) size distribution is performed. The particles are placed randomly inside a cube-shaped space in such a way that a relatively dense pack is obtained without particles overlapping each other.

The cumulative CPU times for all ten contact detections as a function of the total number of discrete elements comprising the problem are recorded. First, the problem is run with all particles being of the same diameter $d = 1$. The CPU time as function of number of particles is shown in Fig. 3. The results shown are obtained by changing *N* from $N = 8000$ to $N = 5\,000\,211$. It is worth noting that the total CPU time is proportional to *N*. This is because both search and sort algorithms have linear CPU time requirements.

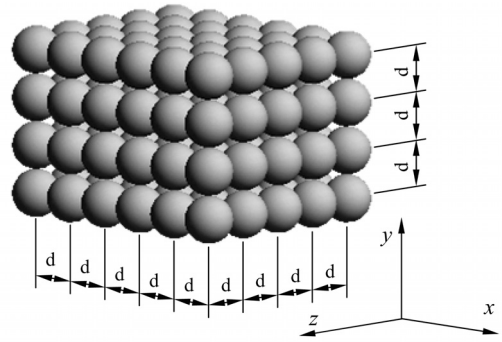

Fig. 2. Initial pack configuration.

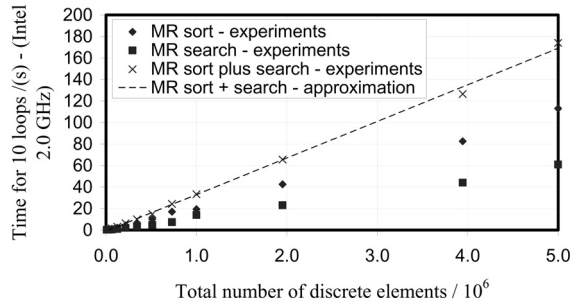

Fig. 3. CPU time as a function of the number of particles. MR contact detection comprising of MR sort followed by MR search.

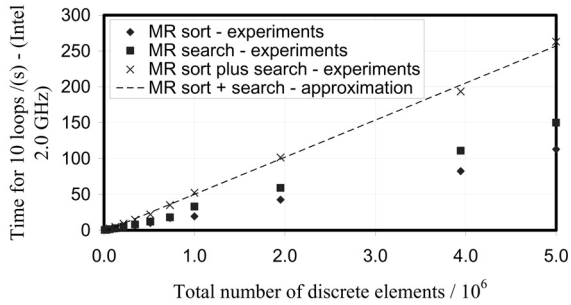

Fig. 4. CPU time as a function of the number of particles – MMR contact detection algorithm, uniform size distribution.

The same problem is repeated with particles being of diameter varying from $d = 1$ to $d = 1/32$ and obeying uniform size distribution. For each problem, contact detection is repeated ten times and the cumulative CPU times for all ten contact detections as a function of the total number of discrete elements comprising the problem are recorded. The CPU time as a function of number of particles is shown in Fig. 4.

5. Conclusion

The MR-linear contact detection algorithm presented in this paper belongs to the category of so-called linear contact detection algorithms. These include NBS and C-GRID algorithms. Unlike the NBS algorithm, in the MR-linear contact detection algorithm the data structure is constantly available. In the NBS algorithm the data structure is rebuilt each time step. With the MR-linear contact detection algorithm the data structure is only modified each time step. Thus, the MR-linear contact detection algorithm has all the advantages of a binary tree based algorithm. In addition, it has much better performance, as demonstrated by the examples shown in this paper. Total detection time for the algorithm does not depend on packing density, a result confirmed by both theoretical investigations and numerical experiments. On the other hand, the relationship between the total detection time and the total number of discrete elements comprising the problem is linear, again a result confirmed by both theoretical investigations and numerical experiments. Memory requirements of the algorithm are insignificant and do not change at all with change in packing density.

The multi-step MR contact detection algorithm presented in this paper is yet another example of so-called linear contact detection algorithms. Unlike the MR contact detection algorithm, which is suitable for systems comprising particles of similar size, the multistep algorithm presented in this paper is suitable for systems comprising particles of greatly different sizes. The RAM requirements are independent of size distribution of the pack and are the same as those for the MR algorithm. However, CPU requirements depend on the number of steps and size distribution. The upper limit for CPU time of the n-step algorithm (the worst case scenario) is n-multiple of CPU time for the single size MR algorithm.

References

[1] Chung SH, Mustoe GGW. Effects of particle shape and size distribution on stemming performance in blasting. Proc 3rd International Conf. on Discrete Element Methods, Santa Fé, NM, USA, 2002.

[2] Owen DRJ, Feng YT, Cottrel MG, Yu J. Discrete/finite element modelling of industrial applications with multi-fracturing and particulate phenomena, Proc. 3rd International Conf on Discrete Element Methods, Santa Fé, NM, USA, 2002.

[3] Williams JR, Mustoe GGW. Proc 2nd US Conf on Discrete Element Methods, MIT, MA, 1993.

[4] Munjiza A. The Combined Finite-Discrete Element Method, Chichester: Wiley, 2004.

[5] Preece DS, Chung SH. An algorithm for improving 2-D and 3-D spherical element behavior during formation of muck piles resulting from rock blasting. Proc 3rd International Conf. on Discrete Element Methods, Santa Fé, NM, USA, 2002.

[6] O'Connor R, Gill J, Williams JR. A linear complexity contact detection algorithm for multi-body simulation, Proc 2nd US Conf on Discrete Element Methods, MIT, MA, 1993.

[7] Oldenburg M, Nilsson L. The position code algorithm for contact searching. Int J Numer Meth Eng 1994;37:359–386.

[8] Munjiza A, Andrews KRF. NBS contact detection algorithm for bodies of similar size. Int J Num Methods Eng 1998;43:131–149.

[9] Perkins E, Williams JR. Generalized spatial binning of bodies of different sizes. Proc 3rd International Conf on Discrete Element Methods, Santa Fé, NM, USA, 2002.

[10] Bonet J, Peraire J. An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. Int J Numer Meth Eng 1991;31:1–17.