Original scientific paper
Accepted 15. 07. 2002.

DAMIR LAZAREVIĆ
JOSIP DVORNIK
KREŠIMIR FRESL

Contact Detection Algorithm for Discrete Element Analysis

Algoritam određivanja kontakata za metodu diskretnih elemenata

SAŽETAK

U radu je opisana inačica na mreži utemeljenog algoritma za nalaženje parova čestica između kojih se može pojaviti međudjelovanje kratkoga dometa, posebno prilagođena numeričkim analizama velikih sustava čestica podjednakih promjera, ravnomjerno raspodijeljenih u prostoru. Algoritam je primijenjen u modeliranju punjenja i pražnjenja silosa s granularnim materijalom.

Ključne riječi: metoda diskretnih elemenata, određivanje kontakata, sortiranje, pretraživanje

Contact Detection Algorithm for Discrete Element Analysis

ABSTRACT

The grid based variation of the contact detection algorithm, with corresponding data structures, is described. This variation is tuned for numerical analysis of large, homogenously distributed assemblages of particles of fairly equal diameters. As an example, algorithm is used in the modelling of granular material silo filling and discharge.

Key words: discrete element method, contact detection, sorting, searching

MSC 2000: 68P10, 70E55, 70F35

1 Introduction

Silos are industrial structures which experience a significant percentage of damage and collapse in comparison with other engineering structures: over 1000 silos, bins and hoppers fail in North America each year [7]. The main reason for such a state lies in the fact that a satisfactory theory about the motion of granular materials in silos has not yet been fully developed [5, 6, 11].

More or less validated differential equations (versions of Janssen–Koenen equation) and their exact or approximate solutions exist for filling stages and content at rest [13]. As the moving part of the total mass is small (only cap of the contents moves in form of avalanches) and the arching is negligible, these states do not cause significant dynamic effects; principles of continuum mechanics excluding inertial forces are therefore acceptable.

On the other hand, in the course of the discharge stages the usual state of the content is that of a nonuniform, relatively slow flow of material, characterized by arching and a large number of collisions between particles, and, therefore, by high dissipation of energy which leads to potential instabilities in solving equations derived from thermodynamical or hydrodynamical analogies (these analogies are more appropriate for the rapid flow of the content, but, unfortunately, prerequisites for its occurrence are rare in the regimes of silo usage) [4].

Recently, new computational methods, usually called *discrete* or *distinct element methods*, were developed with the aim to more closely model the behaviour of multibody (N–body) or particle assemblages [2, 18, 19].

These discrete numerical approaches comprise three main parts: (i) interaction model, (ii) determination of the interacting bodies, and (iii) numerical integration of the governing equations.

With respect to the interactions between bodies, discrete systems can be broadly classified into three groups [17]:

1. systems governed by long range forces, e. g. gravitational systems, where coupling is all–to–all,
2. systems in which interactions are medium range, e. g. molecular systems, and, finally,

3. systems with short range, mostly impact and contact, interactions, as the one with which we are concerned here, where each particle is usually coupled with dozen particles or thereabout.

The shapes of the particles can be approximated using various geometric solids [17], but the complexity of the form severely influences the computational time needed to determine the geometric details of the contact. Therefore, we opted for the simplest shape, the ball. However, to avoid crystalization, balls are given varying radii

$$r_i = r_{\min} + (r_{\max} - r_{\min})\,\mathtt{rnd}(), \tag{1}$$

where $r_{\max}$ and $r_{\min}$ are predefined extreme radii and $\mathtt{rnd}()$ is the random number generator with a uniform distribution on the unit segment. (If more complex shapes are needed, they can be realized by connecting two or more balls with some overlap. Similar idea was used to model the silo wall, section 4.)

The number of balls currently in the system will be denoted by $n(t)$.

If friction is omitted, rotational degrees of freedom need not be taken into account. Equations of motion of the centroid of the ith particle are then

$$\ddot{\mathbf{u}}_i(t) = \mathbf{M}_i^{-1}\mathbf{f}_i(t) + \mathbf{g}, \tag{2}$$

where $\ddot{\mathbf{u}}_i(t)$ is the acceleration of the centroid, $\mathbf{M}_i$ the diagonal mass matrix and $\mathbf{g}$ is the acceleration vector due to the gravity. The total applied force vector $\mathbf{f}_i(t)$ on the centroid of the particle i, interacting with the $k_i(t)$ particles, is given by

$$\mathbf{f}_i(t) = \sum_{\substack{j=1 \\ j\neq i}}^{k_i(t)} f_{i,j}(t)\,\mathbf{n}_{i,j}(t). \tag{3}$$

The short range interaction force $f_{i,j}(t)$ between particles i and j is modelled by the linear spring and the viscous damper in parallel (the so-called viscoelastic Kelvin or Voigt body) if the balls overlap and by the linear spring (Hookean body) if they are within the reach of cohesion and move apart. Maximum overlap or minimum distance between two balls is given by

$$\delta_{i,j}(t) = r_i + r_j - |\mathbf{u}_i(t) - \mathbf{u}_j(t)|, \tag{4}$$

where $\mathbf{u}_i(t)$ and $\mathbf{u}_j(t)$ are position vectors of the balls' centers. Clearly, the overlap $\delta_{i,j}(t) > 0$ is the numerical/geometrical counterpart of the squeezing of the balls during contact. The unit vector $\mathbf{n}_{i,j}(t)$ on the line joining centers of the balls is defined by

$$\mathbf{n}_{i,j}(t) = \frac{\mathbf{u}_{i,j}(t)}{|\mathbf{u}_{i,j}(t)|} = \frac{\mathbf{u}_i(t) - \mathbf{u}_j(t)}{|\mathbf{u}_i(t) - \mathbf{u}_j(t)|}. \tag{5}$$

System of equations (2) for $i = 1,\ldots,n(t)$ is an approximate description of the large displacements and strains problem. Although material linearity is assumed, the geometric nonlinearity still remains. Because of the frequent collisions, the paths, velocities and accelerations are not smooth functions. Not only the magnitudes, but also the types of the interaction forces between particles depend on the particles' positions and velocities and therefore change intensively in time. Described nonlinear problem has no analytical solution and some step by step technique should be used to numerically integrate equations of motion (our approach, a variation of the predictor–corrector method, is described in [9]).

What is more, neighbours of the ith particle, needed to perform the summation in (3), are not known in advance, but, as the particle system is in permanent motion, must be determined in each time step. Contact detection algorithm which facilitate efficient determination of the interacting particles will be more fully described in the sequel.

2 On spatial sorting and searching

The neighbour is defined here as a particle which is close enough to the observed particle so that any of the aforementioned short range interactions can be 'activated'. Determination of the interacting particles is called *contact detection*. More generally, contact detection is a determination of contact or overlap among members of a set of n geometric entities in an m–dimensional (Euclidean) space. Thus, it is a fundamental operation in a wide variety of diverse computation areas such as computational geometry and computer graphics (including CAD), particle physics and astrophysics, cartography and medical imaging, robotics and computational mechanics... And in particular, in computational mechanics contact detection is not restricted to discrete element methods. Finite element modelling of discontinous contact and fracture phenomena, unstructured multilevel/multigrid solution procedures, mesh generation algorithms, adaptive remeshing and remeshing necessitated by large mesh distortions (even in applications to 'oldfashioned' continuum mechanics) all require some form of contact detection. Closely related algorithms are used in recently developed meshless methods to obtain nodal connectivity and cloud overlap information, too.

The straightforward algorithm to find interacting particle pairs is to simply test each particle against every other in a nested loops:

for $i \leftarrow 1$ to $n(t)-1$ **do**
  for $j \leftarrow i+1$ to $n(t)$ **do**
    test particle i against particle j
  end for
end for

Obviously, for a system containing $n(t)$ particles, the number of required tests is proportional to $n(t)^2$, denoted by $O(n(t)^2)$. This is a very time consuming process for systems with many discrete elements (say 10 000 or more).[1] It was estimated that even with the most sophisticated contact detection algorithms these operations can amount to almost 60% of the total calculation time for large, short range, dynamic discrete systems [17].

These more advanced algorithms usually consist of two (possibly overlapping) phases called *spatial* or *neighbour searching* and *contact* or *geometric resolution* [17]. Spatial searching is the identification of the potential neighbours, while contact resolution determines whether candidate pairs actually interact, i. e. distances between candidates or depths of their mutual penetrations are calculated and compared to threshold values.[2] As the number of potential neighbours is small, the computational cost of contact resolution depends almost only on the complexity of the geometric representation of particles.

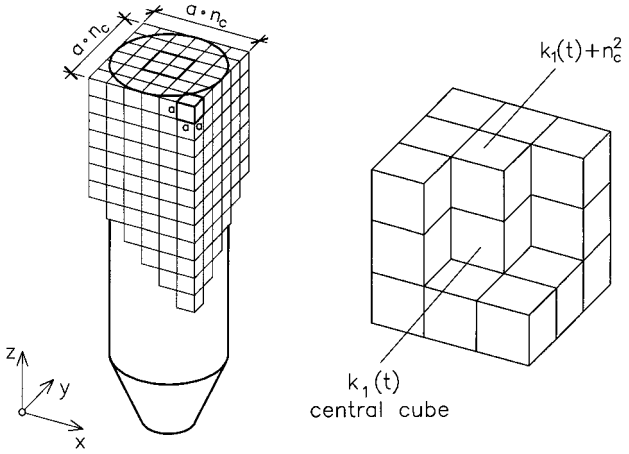
On the other hand, the cost of neighbour searching is dependent on the total number $n(t)$ of particles. Irregularly shaped particles are approximated with bounding boxes or bounding spheres, or even with equivalent spheres whose radius is obtained by taking the size of the largest particle in the system, e. g. [12].

Again, spatial searching is commonly performed in two steps. In the first step the complete set of particles is spatially ordered using some sorting algorithm and appropriate data structure is built. Then, in the second step, this sorted set is searched for potential neighbours. Spatial sorting and searching algorithms and corresponding data structures are mainly based on spatial decomposition. They can be roughly divided into two categories: region of interest (so-called search space) is either 'covered' with a grid, or partitioned in a hierarchical manner. Hierarchical decompositions, e. g. octrees and $3d$–trees [1, 3], are spatial generalizations of the well known one–dimensional binary search trees [8, 15]; average time complexity of neighbour searching is thus $O(n(t)\log n(t))$, although it can degrade to $O(n(t)^2)$ for highly unbalanced trees. Grid techniques, on the other hand, have time complexity $O(n(t))$, but they are much more sensitive to the uneven distribution of particles (clusters and empty space) and to the ball size variances, i. e. the ratio $r_{\max}/r_{\min}$ [12, 14].

3 Fixed cubes scheme

Silo content is densely packed and evenly distributed, except maybe in small areas under arches and vaults (and, of course, above the heap in filling phase). It is also reasonable to assume that particles have approximately equal sizes, i. e. that we can take for balls' radii $r_{\max}/r_{\min} \approx 1.05$. Therefore we developed a variation of the grid based spatial sorting and searching algorithm.

The main idea of the fixed cubes scheme is to cover the search space with cubes (figure 1a) and sort balls in them. Then, during the calculation of forces, contact resolution is made only through the contents of the cubes which intersect the observed ball, and not through the whole region of the silo model. The cube that contains the center of the observed ball will be called *the central cube*.



(a) (b)

Figure 1: Fixed cubes scheme: (a) covering the region of calculation, (b) central cube and its 26 neighbours (6 cubes are omitted for clearness).

If the size a of a cube is selected such that

$$a = 2r_{\max} + \Delta, \tag{6}$$

where $\Delta \geq 0$ is some small number, then all possible neighbours of the ball must be completely or partially contained in 26 cubes around the central cube ($3 \times 3 \times 3$ cubes subspace). These cubes will be called *surrounding cubes*.

[1] With respect to time, discrete systems can be pseudo–static, where relative position of particles do not change appreciably in time, or dynamic, where individual particles move significantly [12]. If the system is pseudo–static, the performance of a searching algorithm is not very important, because neighbours must be determined only once or occasionally, after several time steps.

[2] In our application, as the cohesion can be activated only if particles move apart, directions of motion and/or velocities should be determined, too.

It should be noted that the efficiency of this scheme requires careful selection of the size Δ. If Δ is too large, we have larger cubes, so that smaller number of surrounding cubes contain neighbours (i. e. more than 19 cubes can be immediately eliminated from the search, subsection 3.2), but there are more particles in each cube and contact resolution will be too long. On the other hand, assuming small time steps and small velocities, sorting procedure can be performed before the predictor phase only, but too small Δ (or $\Delta = 0$) may cause overlapping of the cubes by balls in the corrector phase, resulting in incorrect neighbour detection.

3.1 Central cube and its neighbours

According to the coordinates of the ball center $x_i(t)$, $y_i(t)$, $z_i(t)$, integer coordinates of the central cube are obtained from:

$$k_x(t) = \left\lceil \frac{x_i(t)}{a} \right\rceil, \quad k_y(t) = \left\lceil \frac{y_i(t)}{a} \right\rceil, \quad k_z(t) = \left\lceil \frac{z_i(t)}{a} \right\rceil, \quad (7)$$

where $\lceil \cdot \rceil$ means 'ceiling' of the given quotient [8]. Then the cube is assigned a unique index according to the formula

$$k_1(t) = n_c^2\big(k_z(t)-1\big) + n_c\big(k_y(t)-1\big) + k_x(t), \quad (8)$$

where n_c denotes the number of cubes in the global x and y directions (figure 1a). Now, depending on $k_1(t)$ and n_c it is easy to find indices of the remaining 26 cubes. For example, a cube above the central cube has an index given by $k_1(t) + n_c^2$ (figure 1b).

3.2 Elimination of 19 cubes

The condition (6), which can be rewritten as $a > 2r_{\max}$, gives three additional rules which arise from one another:

1. one ball cannot touch two opposite faces of the central cube at the same time,
2. ball can intersect up to three faces, three edges and contain one corner of the central cube,
3. ball can intersect up to 7 neighbouring cubes.

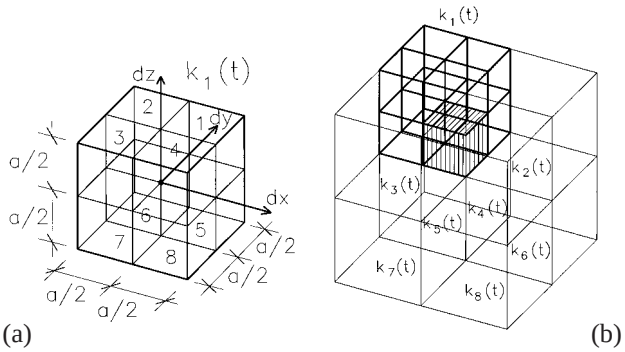
From the given statements it can be recognized that, depending on the position of the ball center in the local coordinate system (or octants/cells) of the central cube (figure 2a), one can eliminate 19 of 26 cubes. This is done by examining inequalities

$$d_x(t) \gtreqless 0, \qquad d_y(t) \gtreqless 0, \qquad d_z(t) \gtreqless 0, \quad (9)$$

where $d_x(t)$, $d_y(t)$ and $d_z(t)$ are local coordinates of the ball center given by

$$\begin{aligned} d_x(t) &= x_i(t) - a\left[k_x(t) - \frac{1}{2}\right], \\ d_y(t) &= y_i(t) - a\left[k_y(t) - \frac{1}{2}\right], \\ d_z(t) &= z_i(t) - a\left[k_z(t) - \frac{1}{2}\right]. \end{aligned} \quad (10)$$

For example, in the case of the ball center placed in the eighth cell (hatched in figure 2b), parts of the corresponding ball can be contained in up to seven cubes, denoted by $k_2(t), \ldots k_8(t)$, whose elements are jointed with the elements of the observed cell, or bound the observed cell, as presented in figure 2b. Similarly, other cells have their own seven neighbouring cubes.



(a) (b)

Figure 2: Example of elimination: (a) local coordinate system (cells) of the central cube, (b) the eighth cell – remaining cubes

3.3 Intersected cubes

Finally, it is now possible to determine which of the 7 candidates $k_i(t)$ intersect the observed ball. This is done by examining distances between the surface of the ball, i. e. sphere, and faces, edges and corner of the observed cell, which also belong to remaining 7 cubes.

First, distances between the center of the ball and the faces of the cell are calculated,

$$\begin{aligned} \delta_x(t) &= \frac{1}{2}a - |d_x(t)|, \\ \delta_y(t) &= \frac{1}{2}a - |d_y(t)|, \\ \delta_z(t) &= \frac{1}{2}a - |d_z(t)|, \end{aligned} \quad (11)$$

followed by an examination of distances between the sphere and faces,

$$\begin{aligned}\Delta_x(t) &= r_i - \delta_x(t) \gtreqless 0, \qquad k_2(t),\\ \Delta_y(t) &= r_i - \delta_y(t) \gtreqless 0, \qquad k_3(t),\\ \Delta_z(t) &= r_i - \delta_z(t) \gtreqless 0, \qquad k_5(t),\end{aligned} \tag{12}$$

edges,

$$\begin{aligned}\Delta_{x,y}(t) &= r_i - \sqrt{\delta_x^2(t)+\delta_y^2(t)} \gtreqless 0,\\ &\qquad k_2(t), k_3(t), k_4(t),\\ \Delta_{x,z}(t) &= r_i - \sqrt{\delta_x^2(t)+\delta_z^2(t)} \gtreqless 0,\\ &\qquad k_2(t), k_5(t), k_6(t),\\ \Delta_{y,z}(t) &= r_i - \sqrt{\delta_y^2(t)+\delta_z^2(t)} \gtreqless 0,\\ &\qquad k_3(t), k_5(t), k_7(t),\end{aligned} \tag{13}$$

and corner of that cell,

$$\begin{aligned}\Delta_{x,y,z}(t) &= r_i - \sqrt{\delta_x^2(t)+\delta_y^2(t)+\delta_z^2(t)} \gtreqless 0,\\ &\qquad k_2(t), k_3(t), k_4(t), k_5(t), k_6(t), k_7(t), k_8(t),\end{aligned} \tag{14}$$

as shown in the example of the eighth cell depicted in figure 3. For each test (12)–(14), satisfaction of the criterion > 0 means that the ball intersects listed cubes.

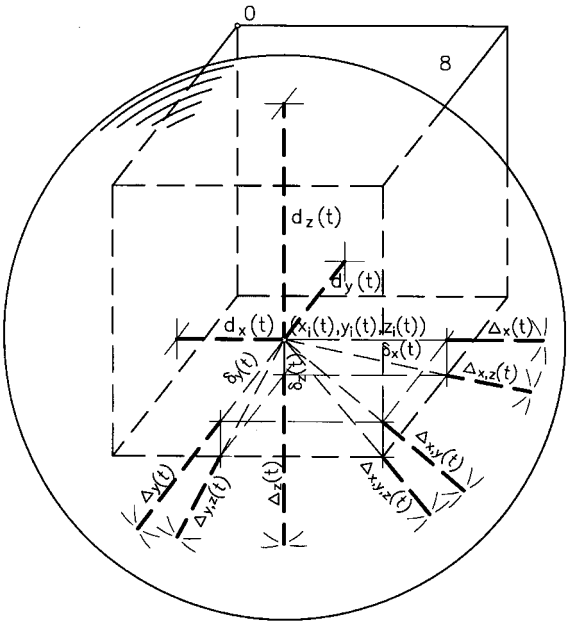


Figure 3: The center of the ball placed in the eighth cell

It should be pointed out that the ball, which do not intersect a face of the cell, can reach neither edges nor corners of that face. Therefore, (12)–(14) should be examined in the following order:

$$\text{faces} \rightarrow \text{edges} \rightarrow \text{corner}. \tag{15}$$

This logic (avoiding time argument t) can then be written as

$$\Delta_x \rightarrow \Delta_y \rightarrow \Delta_{x,y} \rightarrow \Delta_z \rightarrow \Delta_{x,z} \rightarrow \Delta_{y,z} \rightarrow \Delta_{x,y,z}. \tag{16}$$

Of course, during examinations, cube indices $k_i(t)$ must correspond to the chronology of examining distances given by (16). According to our convention (t is omitted), it holds that:

$$k_2 \rightarrow k_3 \rightarrow k_4 \rightarrow k_5 \rightarrow k_6 \rightarrow k_7 \rightarrow k_8. \tag{17}$$

As mentioned in the subsection 3.1, these indices are, depending on $k_1(t)$ and n_c, known a apriori and could be easily determined, as given by algorithm 1.

Algorithm 1: Forming the vector **k** of indices of cubes which could be intersected by the ball whose center is in the eighth cell.

1: **cell_8** $(n_c, k_x, k_y, k_z) \mapsto$ **k**
2: $k_1 \leftarrow n_c^2(k_z-1)+n_c(k_y-1)+k_x$ {central cube; (8)}
3: $k_2 \leftarrow k_1+1$ {$\Delta_x > 0$; 1st eq. in (12)}
4: $k_3 \leftarrow k_1-n_c$ {$\Delta_y > 0$; 2nd eq. in (12)}
5: $k_4 \leftarrow k_1-n_c+1$ {$\Delta_{x,y} > 0$; 1st eq. in (13)}
6: $k_5 \leftarrow k_1-n_c^2$ {$\Delta_z > 0$; 3rd eq. in (12)}
7: $k_6 \leftarrow k_1-n_c^2+1$ {$\Delta_{x,z} > 0$; 2nd eq. in (13)}
8: $k_7 \leftarrow k_1-n_c^2-n_c$ {$\Delta_{y,z} > 0$; 3rd eq. in (13)}
9: $k_8 \leftarrow k_1-n_c^2-n_c+1$ {$\Delta_{x,y,z} > 0$; (14)}

Cubes which bind other cells can be similary found and stored using appropriate procedure **cell_i**, $1 \le i \le 8$.

3.4 Special configurations and ambiguous cases

There exist some limiting, very rare, but in the case of a large number of balls possible positions, where indices of the central cube, the cell and the remaining cubes are geometrically ambiguous or undefined if no additional decisions are provided.

Undefined central cube If the center of the ball falls on the common face, at the edge or at the corner of some cubes, that is, if one or more expressions

$$\begin{aligned}x_i(t) \bmod a &= 0,\\ y_i(t) \bmod a &= 0,\\ z_i(t) \bmod a &= 0,\end{aligned} \tag{18}$$

are satisified, it is easy to verify that because of the 'ceiling' operations in (7), equation (8) chooses the cube with the highest integer coordinates (the highest index) as central.

Undefined cell If the center of the ball falls on the plane, on the axis, or at the origin of the local coordinate system of the central cube, that is one or more of the following equalities holds,

$$d_x(t) = 0, \qquad d_y(t) = 0, \qquad d_z(t) = 0 \tag{19}$$

(equalities in (9)), we decided to choose the same cell as in the > 0 case.

This is deemed reasonable because the procedure given by (11)–(14) will in the case of the positions of the center described in this and previous paragraph provide all remaining cubes irrespective of the central cube and cell index. In these positions, *one* possible cube is used as central, and *one* of its cells is adopted. It is irrelevant which of these cubes/cells is used.

It is interesting to note that described positions can provide indices of all remaining cubes immediately, without additional searching. For example, coordinates of the center placed at the common corner of the (eight) cubes must fulfill (18) and, with the index of the central cube given in previous paragraph, seven remaining cubes can directly be found using (8). Similarly, local coordinates of the ball center placed at the center of the central cube must fulfill (19), which means that the central cube is the only cube, because it contains the entire ball. Analogously, by recognizing single relations in (18) and (19), other specific positions of the ball center (on the face, at the edge, or on local axis) can be found and remaining cubes directly determined. However, as these special positions are very rare, additional tests needed to recognize them introduce unnecessary computational burden and is therefore omitted.

Undefined remaining cubes If the ball touches a face, an edge or a corner of the chosen cell, that is, if one or more equalities appear in (12)–(14), it is considered that the corresponding cubes also contain the ball, as in the > 0 case, because possible touching between ball and its neighbours in such a cube could activate the cohesion force.

3.5 A kind of a binary tree

It follows from the previous section that it is unnecessary to treat equalities in (9) and (12)–(14) independently. It is sufficient to add them to the $>$ case and always examine only *two* possibilities ($\geq$ and $<$) in given relations.

This fact, with (16), gives for every moving ball in the system search structure which resembles the known binary tree [8, 15]. The flow chart of the searching procedure for one ball is, according to (17), partially presented in figure 4.

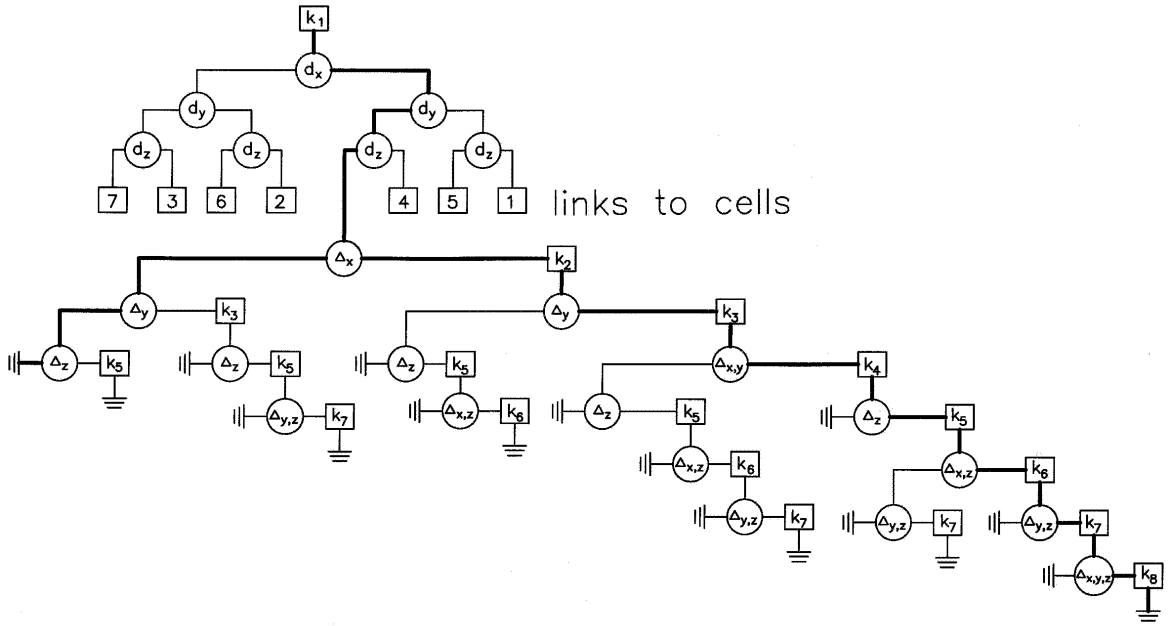


Figure 4: Shape and indexing of the search tree when the ball's center is placed in the eighth cell. Extreme search paths are marked.

The remaining seven subtrees denoted with the cell numbers rather than drawn are of the same shape but contain indices of the other, for certain cell neighbouring, cubes. The analysis of the tree shows that for extreme searching situations 6 examinations are needed if the ball is fully contained in the cube (this is a rare position and the shortest branch in the tree) and 10 examinations if the ball contains a corner of the cube (this is more frequent and the longest branch in the tree). These paths are marked in figure 4.

3.6 Programming strategy for sorting procedure

Memory organization of the procedures which will be described by pseudocode is now defined.[3]

The number of cubes $a_i(t)$ intersected by ball i and, *vice versa*, balls $b_k(t)$ by cube k, are saved as components of vectors $\mathbf{a}(t)$ and $\mathbf{b}(t)$, respectively. Indices i and k are saved as components of the matrix $\mathbf{B}(t)$ and $\mathbf{A}(t)$, that is $b_{k,b_k(t)}(t)$ and $a_{i,a_i(t)}(t)$. Thus, all temporary relations between cubes and balls in the system are stored. The filling procedure for adopted vectors and matrices is given by the algorithm 2.

Algorithm 2: Saving cube k that intersect ball i and vice versa.

1: **cube** $(i,k,\mathbf{a},\mathbf{b},\mathbf{A},\mathbf{B}) \mapsto [\mathbf{a},\mathbf{b},\mathbf{A},\mathbf{B}]$
2: $a_i \leftarrow a_i+1$ {add new cube}
3: $a_{i,a_i} \leftarrow k$ {store new cube index}
4: $b_k \leftarrow b_k+1$ {add new ball}
5: $b_{k,b_k} \leftarrow i$ {store new ball index}

Matrices $\mathbf{A}(t)$ and $\mathbf{B}(t)$ could also be represented as vectors using the linked allocation procedure [8]. This is not a problem for the matrix $\mathbf{A}(t)$ because balls are sorted in cubes successively, in increasing order, so that the matrix is filled from left to right and row by row. But, simultaneously, the matrix $\mathbf{B}(t)$ is filled almost randomly, so it is necessary to use one of the insertion techniques (for example bisection) for placing the ball index at the correct place of the equivalent vector. Of course, this storage scheme saves the amount of memory space because zeroes are not stored (though some additional vectors for addressing are needed), but also increases the amount of time taken for sorting phase. However, the silo contents has a relatively uniform density distribution, so that matrices $\mathbf{A}(t)$ and $\mathbf{B}(t)$ are always well populated and 'classical' array representation suffices.

The complete search algorithm is given by pseudocode 3 and the sorting procedure by pseudocode 4.

Algorithm 3: Searching for cubes that intersect given ball i.

```
 1: search (i,Δx,Δy,Δx,y,Δz,Δx,z,Δy,z,Δx,y,z,k,a,b,A,B)
    ↦ [a,b,A,B]
 2: [a,b,A,B] ← cube (i,k1,a,b,A,B) {cube k1}
 3: if Δx ≥ 0 then {cube k2}
 4:    [a,b,A,B] ← cube (i,k2,a,b,A,B)
 5:    if Δy ≥ 0 then {cube k3}
 6:       [a,b,A,B] ← cube (i,k3,a,b,A,B)
 7:       if Δx,y ≥ 0 then {cube k4}
 8:          [a,b,A,B] ← cube (i,k4,a,b,A,B)
 9:          if Δz ≥ 0 then {cube k5}
10:             [a,b,A,B] ← cube (i,k5,a,b,A,B)
11:             if Δx,z ≥ 0 then {cube k6}
12:                [a,b,A,B] ← cube (i,k6,a,b,A,B)
13:                if Δy,z ≥ 0 then {cube k7}
14:                   [a,b,A,B] ← cube (i,k7,a,b,A,B)
15:                   if Δx,y,z ≥ 0 then {cube k8}
16:                      [a,b,A,B] ← cube (i,k8,a,b,A,B)
17:             else {Δx,z < 0}
18:                if Δy,z ≥ 0 then {cube k7}
19:                   [a,b,A,B] ← cube (i,k7,a,b,A,B)
20:       else {Δx,y < 0}
21:          if Δz ≥ 0 then {cube k5}
22:             [a,b,A,B] ← cube (i,k5,a,b,A,B)
23:             if Δx,z ≥ 0 then {cube k6}
24:                [a,b,A,B] ← cube (i,k6,a,b,A,B)
25:             if Δy,z ≥ 0 then {cube k7}
26:                [a,b,A,B] ← cube (i,k7,a,b,A,B)
27:    else {Δy < 0}
28:       if Δz ≥ 0 then {cube k5}
29:          [a,b,A,B] ← cube (i,k5,a,b,A,B)
30:          if Δx,z ≥ 0 then {cube k6}
31:             [a,b,A,B] ← cube (i,k6,a,b,A,B)
32: else {Δx < 0}
33:    if Δy ≥ 0 then {cube k3}
34:       [a,b,A,B] ← cube (i,k3,a,b,A,B)
35:       if Δz ≥ 0 then {cube k5}
36:          [a,b,A,B] ← cube (i,k5,a,b,A,B)
37:          if Δy,z ≥ 0 then {cube k7}
38:             [a,b,A,B] ← cube (i,k7,a,b,A,B)
39:    else {Δy < 0}
40:       if Δz ≥ 0 then {cube k5}
41:          [a,b,A,B] ← cube (i,k5,a,b,A,B)
```

[3] Fortran 77 is chosen as implementation language. Therefore, given algorithms and corresponding data structures are in a sense not very 'contemporary' or 'fashionable'.

Algorithm 4: Sorting balls in cubes and vice versa.

1: **sort** $(n, n_w, n_c, a, \mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{r}) \mapsto [\mathbf{a}, \mathbf{b}, \mathbf{A}, \mathbf{B}]$
2: **for** $i \leftarrow n_w + 1$ to n **do** {sorting movable balls}
3: $k_x \leftarrow \lceil x_i/a \rceil$; $k_y \leftarrow \lceil y_i/a \rceil$; $k_z \leftarrow \lceil z_i/a \rceil$
{integer coordinates of the central cube; (7)}
4: $d_x \leftarrow x_i - a(k_x - 1/2)$; $d_y \leftarrow y_i - a(k_y - 1/2)$;
$d_z \leftarrow z_i - a(k_z - 1/2)$
{local coordinates of the center; (10)}
5: **if** $d_x \geq 0$ **then** {finding the cell; (9)}
6: **if** $d_y \geq 0$ **then**
7: **if** $d_z \geq 0$ **then** {cell 1}
8: $\mathbf{k} \leftarrow$ **cell_1** (n_c, k_x, k_y, k_z)
9: **else** {cell 5}
10: $\mathbf{k} \leftarrow$ **cell_5** (n_c, k_x, k_y, k_z)
11: **else** $\{d_y < 0\}$
12: **if** $d_z \geq 0$ **then** {cell 4}
13: $\mathbf{k} \leftarrow$ **cell_4** (n_c, k_x, k_y, k_z)
14: **else** {cell 8}
15: $\mathbf{k} \leftarrow$ **cell_8** (n_c, k_x, k_y, k_z)
16: **else** $\{d_x < 0\}$
17: **if** $d_y \geq 0$ **then**
18: **if** $d_z \geq 0$ **then** {cell 2}
19: $\mathbf{k} \leftarrow$ **cell_2** (n_c, k_x, k_y, k_z)
20: **else** {cell 6}
21: $\mathbf{k} \leftarrow$ **cell_6** (n_c, k_x, k_y, k_z)
22: **else** $\{d_y < 0\}$
23: **if** $d_z \geq 0$ **then** {cell 3}
24: $\mathbf{k} \leftarrow$ **cell_3** (n_c, k_x, k_y, k_z)
25: **else** {cell 7}
26: $\mathbf{k} \leftarrow$ **cell_7** (n_c, k_x, k_y, k_z)
27: $\delta_x \leftarrow a/2 - |d_x|$; $\delta_y \leftarrow a/2 - |d_y|$; $\delta_z \leftarrow a/2 - |d_z|$
{position of the centroid in the cell; (11)}
28: $\Delta_x \leftarrow r_i - \delta_x$; $\Delta_y \leftarrow r_i - \delta_y$; $\Delta_z \leftarrow r_i - \delta_z$
{distances between the sphere and faces of the cell; (12)}
29: $\Delta_{x,y} \leftarrow r_i - \sqrt{\delta_x^2 + \delta_y^2}$; $\Delta_{x,z} \leftarrow r_i - \sqrt{\delta_x^2 + \delta_z^2}$;
$\Delta_{y,z} \leftarrow r_i - \sqrt{\delta_y^2 + \delta_z^2}$
{distances between the sphere and edges of the cell; (13)}
30: $\Delta_{x,y,z} \leftarrow r_i - \sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$
{distance between sphere and corner of the cell; (14)}
31: $[\mathbf{a}, \mathbf{b}, \mathbf{A}, \mathbf{B}] \leftarrow$
search $(i, \Delta_x, \Delta_y, \Delta_{x,y}, \Delta_z, \Delta_{x,z}, \Delta_{y,z}, \Delta_{x,y,z}, \mathbf{k}, \mathbf{a}, \mathbf{b}, \mathbf{A}, \mathbf{B})$
{searching for intersected cubes}

The efficiency of given algorithms for the densest observed packing ($b_k(t) = 20$) and, theoretically, the sparsest packing ($b_k(t) = 1$) of the system with various numbers of balls, is shown on diagrams in figure 5. They show that sorting time is almost of order $O\big(n(t)\big)$ as expected [15, 12, 14] for a well distributed (not clustered) system as the one presented here.

3.7 Contact resolution and geometry

After spatial sorting is completed, contact resolution may be of the $O\big(\tilde{n}^2(t)\big)$ order, because of the small number of possible neighbours in cubes that intersect the observed ball (in our calculations $\tilde{n}(t) = b_k(t) \leq 20$).

Finally, it is quite simple to solve for geometry needed for determining interaction forces between balls i and j. As indicated in the introduction, two examinations to find if they overlap or possibly stick to each other are needed:

$$\delta_{i,j}(t) > 0, \tag{20}$$

$$-N^c_{\max;i,j}/k_{i,j} \leq \delta_{i,j}(t) \leq 0, \tag{21}$$

where distance $\delta_{i,j}(t)$ is given by (4), $N^c_{\max;i,j}$ is the maximum cohesion force and $k_{i,j}$ is the stiffness of the collision model. In the latter case, an additional testing is whether they are going apart or approaching one another along the line joining their centers:

$$d^n_{i,j}(t) = \dot{\mathbf{u}}_{i,j}(t) \cdot \mathbf{u}_{i,j}(t) < 0. \tag{22}$$

4 Boundary conditions

The model of the silo wall is made of fixed overlapping balls with randomized radii, again to prevent crystalization of balls representing silage material. This model also imitates friction due to roughness and geometric imperfections on the surface of the wall. Thus, in the absence of an expensive model of friction between balls (characterized by a friction coefficient), the aim was to simulate at least the geometric part of this phenomena. Boundary balls are generated with separate procedure **wall** $(d_h, h_h, d_c, h_c, k_w, c_w, n_w, \mathbf{k}, \mathbf{c}, \mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{r}, \dot{\mathbf{x}}, \dot{\mathbf{y}}, \dot{\mathbf{z}})$, not shown here. Values d_h, h_h and d_c, h_c are diameters and heights of conical and cylindrical parts of the silo and k_w, c_w are stiffness and viscosity of the wall model. The number of boundary wall balls is n_w. Their velocities are, of course, zero.

Two additional things must be mentioned here. First, it is sufficient to execute the sorting procedure and store the boundary balls in the appropriate cubes only once, in the beginning of the calculation, as they do not move. Thus, the sorting procedure must always be performed for the moving balls only, as indicated in the line 2 of the algorithm 4.

Second, the algorithm 2 requires that vectors $\mathbf{a}(t)$, $\mathbf{b}(t)$ and matrices $\mathbf{A}(t)$, $\mathbf{B}(t)$ should always be set to zero before

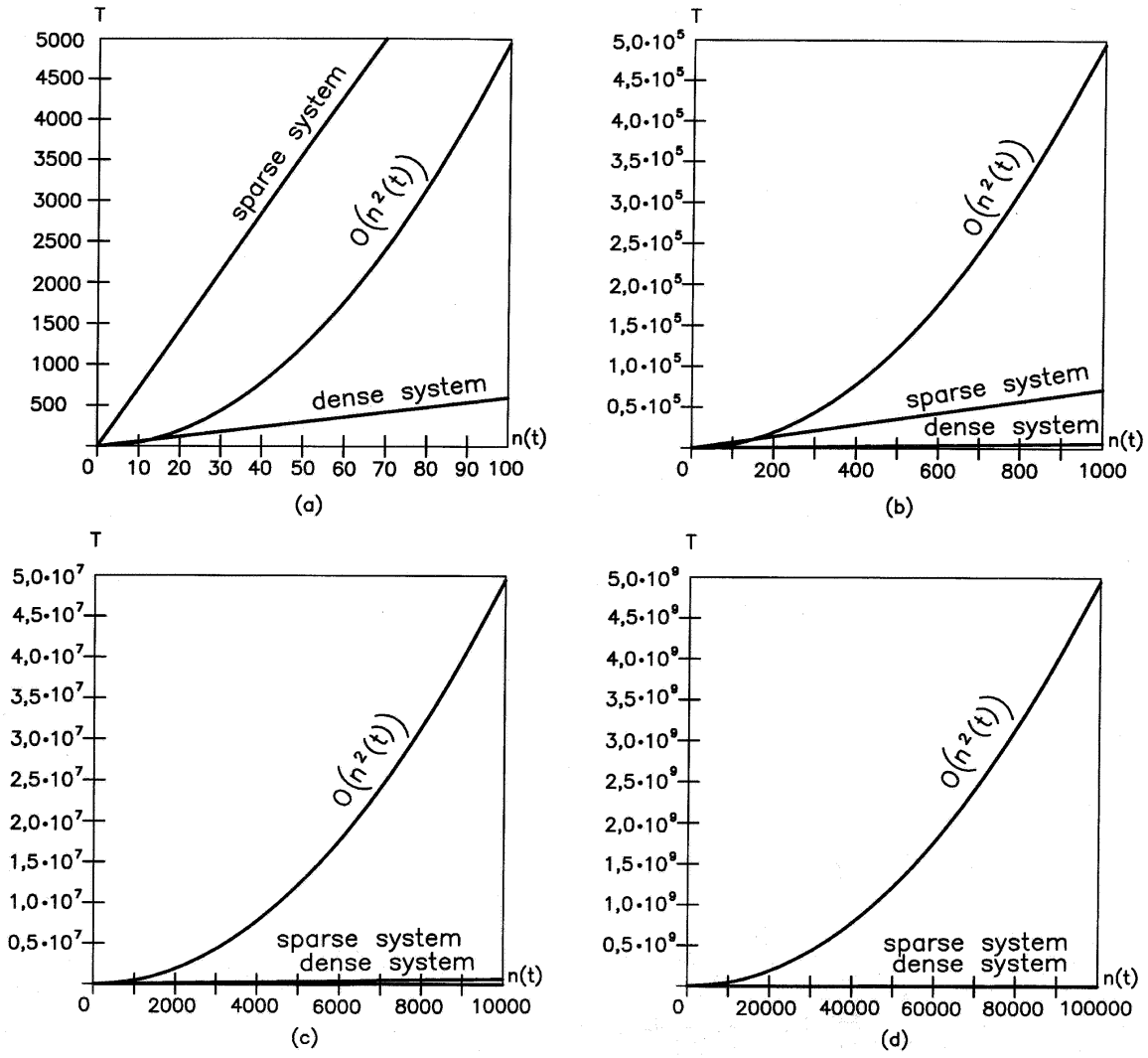


Figure 5: Efficiency of fixed cubes algorithm for sparse and dense systems in comparison with $O\left(n^2(t)\right)$ for various numbers of balls: (a) $n(t) \leq 100$, (b) $n(t) \leq 1000$, (c) $n(t) \leq 10000$, (d) $n(t) \leq 100000$.

the sorting procedure is executed. But from the previous comment it follows that this process should also be performed for moving balls only. Therefore indices of boundary balls must be saved and their cubes stored in $\mathbf{A}(t)$ and $\mathbf{B}(t)$. This is done by using the total number of boundary balls n_w (given by the procedure **wall**) and saving numbers of boundary balls in every cube **w** (given by first call of the procedure **sort** after **wall** is executed).

5 Calculation of interactions

Now, naïve nested loops from section 2 can be written as given in algorithm 5, where ball pairs are denoted by i and m instead of by i and j.

To avoid multiple interactions logical vector $\mathbf{s}(t)$ is used with additional testing to determine the cube which, according to (7) and (8), contains the midpoint of the line segment between centers:

$$x_s = \frac{1}{2}(x_i + x_m),\ y_s = \frac{1}{2}(y_i + y_m),\ z_s = \frac{1}{2}(z_i + z_m). \quad (23)$$

Once sorted, interactions between boundary and moving balls in the sense of the algorithm 5 are nothing special. Assuming zero velocities for boundary balls, all other constants of the collision model are obtained.

It should be mentioned that given vectors and matrices could be (in a more 'modern' implementation) allocated dynamically, because the system moves, so their sizes vary during calculation, i. e. $n(t) \geq 1$, $b_k(t) \geq 1$, $1 \leq a_i(t) \leq 8$.

Algorithm 5: Loops over sorted balls.

```
1: loops (n, n_w, n_c, s, a, b, A, B, x, y, z, r)
2: for i ← n_w + 1 to n do {movable balls}
3:   s_i ← true {solving ball i}
4:   for j ← 1 to a_i do {intersected cubes}
5:     k ← a_{i,j} {current cube index}
6:     for l ← 1 to b_k do {all balls in intersected cubes}
7:       m ← b_{k,l} {current ball index}
8:       if s_m ≡ false then {ball m is not solved}
9:         x_s ← (x_i + x_m)/2;  y_s ← (y_i + y_m)/2;
           z_s ← (z_i + z_m)/2
           {midpoint coordinates; (23)}
10:        k_x ← ⌈x_s/a⌉ ;  k_y ← ⌈y_s/a⌉ ;  k_z ← ⌈z_s/a⌉
           {integer coordinates of cube with the midpoint; (7)}
11:        k_s ← n_c^2(k_z − 1) + n_c(k_y − 1) + k_x {cube index; (8)}
12:        if k_s = k then {midpoint in the current cube}
13:          test particle i against particle m
```

The total number of cubes is fixed during the calculation due to the fixed cubes scheme used here, but it can be given as the input parameter.

6 Example

Described algorithms were implemented in Fortran 77 (version by Absoft) and incorporated in a computer program which simulates various regimes during silo exploatation. Also, for better visualization of results fast perspective routines using PGPLOT graphics library were programmed.

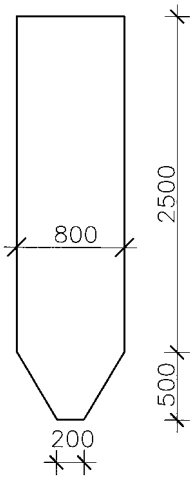
Some snapshots during silo filling and discharge are given in figures 6 and 7, respectively. For graphical presentation the silo model was cut with a plane through its vertical axis and only half of the model was rendered so that the insides of the silo can be seen. Input data of the presented example are given in table 1.

7 Conclusions

There is no universal spatial sorting and searching algorithm whose performance is (completely) independant of the characteristics of the analysed discrete system. Namely, discrete systems can be densely or sparsely packed, and, what is more important, particles can be evenly distributed or clustered. Furthermore, the range of bounding spheres' radii (i. e. whether spheres have equal, approximately equal or considerably differing radii) must be taken into account.

Table 1: Main data of the model.

$r_{\min}$	$0,215$ m
$r_{\max}$	$0,225$ m
a	$0,500$ m
Δ	$0,050$ m
$n_{\max}^{\text{contents}}$	22741
n_w	3821
γ	$1250\ \text{kg/m}^3$
k_c	10^7 N/m
k_w	10^9 N/m
c_c	10^8 Ns/m
c_w	10^7 Ns/m
$\Delta t_{\text{filling}}$	10^{-4} s
$\Delta t_{\text{discharge}}$	10^{-5} s



In particular, theoretical $O(n)$ performance of grid based algorithms cannot be attained if particles are clustered in few cells only, because there are many unused cells which nevertheless must be tested. Furthermore, the ball size variances lead to the so-called over–reporting problem as the size of the cells are determined by the largest particle in the system.

But silo content has homogenous spatial distribution while grains can be assumed to have fairly equal sizes and, therefore, prerequisites for the optimal behaviour of grid techniques are realised. Majority of computational time in discrete element simulations is spent in contact detection and it is, therefore, sensible to develop highly specialised algorithm tuned for discrete numerical modelling of silo exploatation.

References

[1] BONET, J.; PERAIRE, J.: *An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems*, International Journal for Numerical Methods in Engineering, **31** (1991), pp. 1–17.

[2] CUNDALL, P. A.; STRACK, O. D. L.: *A Distinct Element Model for Granular Assemblies*, Geotechnique, **29** (1979), pp. 47–65.

[3] FENG, Y. T.; OWEN, D. R. J.: *A Spatial Digital Tree Based Contact Detection Algorithm*, Proceedings of ICADD–4, Fourth International Conference on Analysis of Discontinuous Deformation (ed. Bićanić, N.), June 6th–8th, 2001, University of Glasgow, Scotland, UK, pp. 221–238.

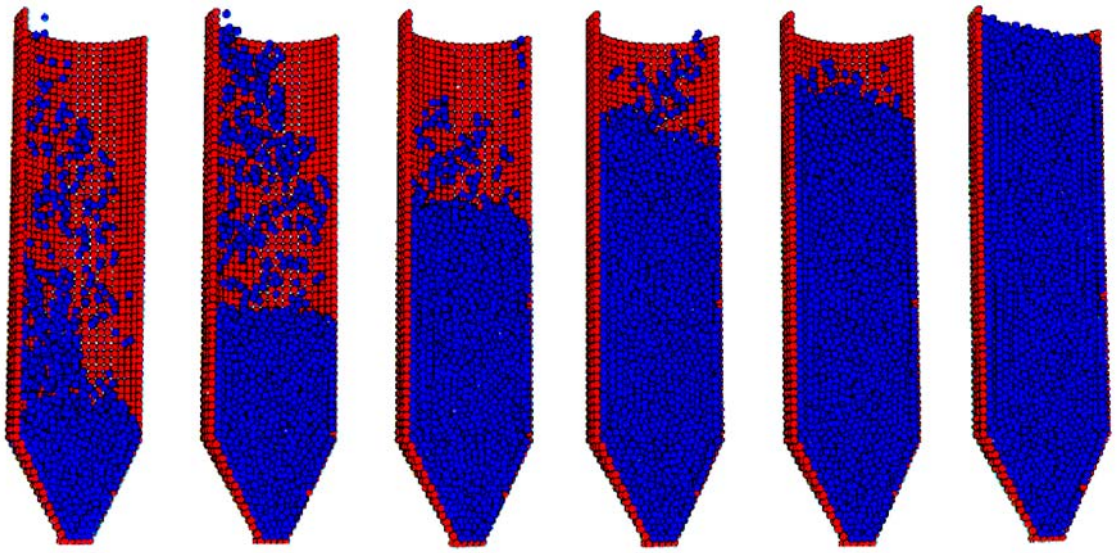


Figure 6: Silo filling.

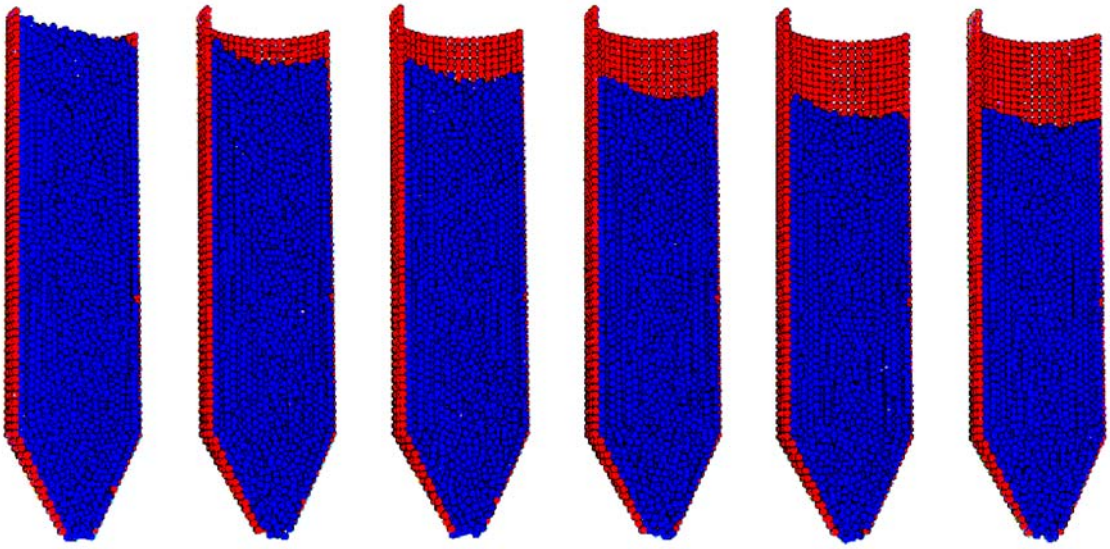


Figure 7: Silo discharge.

[4] JAEGER, H. M.: *Chicago Experiments on Convections, Compaction, and Compression*, Physics of Dry Granular Media (eds. Herrmann, H. J.; Hovi, J.–P.; Luding, S.), Kluwer Academic Publishers, Dordrecht, 1997, pp. 553–583.

[5] JAEGER, H. M.; NAGEL, S. R.: *Physics of the Granular State*, Science, **255** (1992), pp. 1523–1531.

[6] JAEGER, H. M.; NAGEL, S. R.; BEHRINGER, R. P.: *Granular Solids, Liquids and Gases*, Reviews of Modern Physics, **68** (1996), pp. 1259–1273.

[7] KNOWLTON, T. M.; CARSON, J. W.; KLINZING, G. E.; YANG, W-C.: *The Importance of Storage, Transfer, and Collection*, Chemical Engineering Progress, **90** (1994), pp. 44–54.

[8] KNUTH, D. E.: *The Art of Computer Programming. Volume 3: Sorting and Searching*, Addison–Wesley, Reading, Massachusetts, 1973.

[9] LAZAREVIĆ, D.; DVORNIK, K.; FRESL, K.: *Diskretno numeričko modeliranje opterećenja silosa*, Građevinar, **54** (2002) pp. 135–144.

[10] MARK, J.; HOLST, F. G.; ROTTER, J. M.; OOI, J. Y.; RONG, G. H.: *Numerical Modeling of Silo Filling. II: Discrete Element Analyses*, Journal of Engineering Mechanics, **125** (1999) pp. 104–110.

[11] MEHTA, A.; BARKER, G. C.: *The Dynamics of Sand*, Reports on Progress in Physics, 1994, pp. 383–416.

[12] MUNJIZA, A.; ANDREWS, K. R. F.: *NBS Contact Detection Algorithm for Bodies of Similar Size*, International Journal for Numerical Methods in Engineering, **43** (1998), pp. 131–149.

[13] NEDDERMAN, R. M.: *Statics and Kinematics of Granular Materials*, Cambridge University Press, Cambridge, New York, USA, 1992.

[14] PERKINS, E.; WILLIAMS, J.: *CGrid: Neighbor Searching for Many Body Simulation*, Proceedings of ICADD–4, Fourth International Conference on Analysis of Discontinuous Deformation (ed. Bićanić, N.), June 6th–8th, 2001, University of Glasgow, Scotland, UK, pp. 427–438.

[15] SEDGEWICK, R.: *Algorithms*, Addison–Wesley, Reading, Massachusetts, 1989.

[16] *Silos, Hoppers, Bins & Bunkers for Storing Bulk Materials*, The Best of Bulk Solids Handling. Selected Articles, Trans Tech Publications, Volume A/86, Clausthal–Zellerfeld, Germany, 1986.

[17] WILLIAMS, J. R., O'CONNOR, R.: *Discrete Element Simulation and the Contact Problem*, Archives of Computational Methods in Engineering, **6** (1999), pp. 279–304.

[18] WILLIAMS, J. R.; HOCKING, G.; MUSTOE, G. E.: *The Theoretical Basis of the Discrete Element Method*, NUMETA 85, Numerical Methods in Engineering: Theory and Applications, (eds. Middleton, J.; Pande, G. N.), Proceedings of the International Conference on Numerical Methods in Engineering: Theory and applications, Swansea, January 7th–11th, 1985, pp. 897–906.

[19] YSERENTANT, H.: *A New Class of Particle Methods*, Numerische Mathematik, **76** (1997), pp. 87–109.

Damir Lazarević
e-mail: damir@grad.hr

Josip Dvornik
e-mail: dvornik@grad.hr

Krešimir Fresl
e-mail: fresl@grad.hr

University of Zagreb
Faculty of Civil Engineering