Particuology 12 (2014) 113–121

Contents lists available at ScienceDirect

Particuology

journal homepage: www.elsevier.com/locate/partic

Accelerating CFD–DEM simulation of processes with wide particle size distributions

Tamir Brosh[a,b,*], Haim Kalman[a], Avi Levy[a]

[a] Pearlstone Centre for Aeronautical Engineering Studies, Department of Mechanical Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

[b] School of Mechanical and Systems Engineering, Stephenson Building, Claremont Road, Newcastle University, Newcastle upon Tyne NE1 7RU, United Kingdom

ARTICLE INFO

Article history:
Received 17 December 2012
Received in revised form 14 March 2013
Accepted 27 April 2013

Keywords:
Computational fluid dynamics
Discrete element model
Search algorithm
Increased time-step

ABSTRACT

Size-reduction systems have been extensively used in industry for many years. Nevertheless, reliable engineering tools to be used to predict the comminution of particles are scarce. Computational fluid dynamics (CFD)–discrete element model (DEM) numerical simulation may be used to predict such a complex phenomenon and therefore establish a proper design and optimization model for comminution systems. They may also be used to predict attrition in systems where particle attrition is significant. Therefore, empirical comminution functions (which are applicable for any attrition/comminution process), such as: strength distribution, selection, equivalence, breakage, and fatigue, have been integrated into the three-dimensional CFD–DEM simulation tool. The main drawback of such a design tool is the long computational time required owing to the large number of particles and the minute time-step required to maintain a steady solution while simulating the flow of particulate materials with very fine particles.

The present study developed several methods to accelerate CFD–DEM simulations: reducing the number of operations carried out at the single-particle level, constructing a DEM grid detached from the CFD grid enabling a no binary search, generating a sub-grid within the DEM grid to enable a no binary search for fine particles, and increasing the computational time-step and eliminating the finest particles in the simulation while still tracking their contribution to the process.

The total speedup of the simulation process without the elimination of the finest particles was a factor of about 17. The elimination of the finest particles gave additional speedup of a factor of at least 18. Therefore, the simulation of a grinding process can run at least 300 times faster than the conventional method in which a standard no binary search is employed and the smallest particles are tracked.

1. Introduction

In computational fluid dynamics (CFD)–discrete element model (DEM) simulations, particle motion is dictated by particle–fluid interactions, particle–wall interactions, and particle–particle interactions. Hence, the calculation of the interactions between particles is expensive in terms of calculation time. The simulation can run faster when using a more efficient search algorithm and when the number of operations carried out in each time-step is reduced. Additionally, to maintain a stable numerical scheme, the time-step must be small enough to allow particle–particle interaction and particle–wall interaction to occur over the duration of several time-steps. This limit of the time-step is related to the size of the particles—the smaller the particles, the smaller the time-step. Grinding simulations thus require very small time-steps owing to the size of the fragments and the large number of particles resulting from constant feeding and grinding.

To make simulation more feasible, the present work developed an efficient search algorithm, a method to increase the maximal allowed time-step, and a method of working around the generation of a large number of fine particles.

2. Contact search algorithm

The simplest search is a standard search with order $O(N^2)$, in which each particle checks the distance between itself and all other

Abbreviations: CFD, Computational fluid dynamics; DEM, Discrete element model; NBS, No binary search.

* Corresponding author at: School of Mechanical and Systems Engineering, Stephenson Building, Claremont Road, Newcastle University, Newcastle upon Tyne NE1 7RU, United Kingdom. Tel.: +44 01912223081.

E-mail addresses: broshta@post.bgu.ac.il, Tamir.brosh@ncl.ac.uk (T. Brosh).

1674-2001/$ – see front matter © 2013 Chinese Society of Particuology and Institute of Process Engineering, Chinese Academy of Sciences. Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.partic.2013.04.008

Nomenclature

d	Diameter (m)
e	Coefficient of restitution
E	Modulus of elasticity (Pa)
k	Particle stiffness ($N/m^{-1.5}$)
m	Mass (kg)
r	Particle radius (m)
V_{impact}	Impact velocity (relative normal velocity) (m/s)

Greek letters

γ	Constant correcting the damping coefficient, $\gamma = \sqrt{\frac{\ln^2 e}{\pi^2+\ln^2 e}}$
δ	Displacement (overlap) (m)
Δt	Time-step (s)
η	Particle damping coefficient, $\eta = \gamma\sqrt{km}$ ($N\,s/m^{1.25}$)
ν	Poisson's ratio
ρ	Density (kg/m^3)

Subscripts

1, 2	Indexes
max	Maximal
n	Normal
p	Particle
t	Tangent
w	Wall

particles in the simulation and two particles are deemed to be in contact if their separation is less than the sum of their radii. This is an inefficient process as we have to scan $N(N-1)/2$ particles in each time-step.

One way to reduce the search time is to employ "bookkeeping". Each particle keeps a list of its neighboring particles and there is thus no need to scan the entire domain; only particles in the vicinity are considered. This method, called Verlet list, is efficient in cases where particles move with similar velocity (Pöschel & Schwager, 2005). However, the method is far less effective when the system is composed of many particles having different sizes and velocities, in which case the particles are rapidly removed from the Verlet lists as a standard search with order $O(N^2)$ has to be employed as frequently as the Verlet lists are reconstructed.

Another way to reduce the search time is to divide the computational domain into cells (or smaller domains) and assign each particle to the cell that corresponds to the particle's position. Therefore, no computer resources are wasted in searching for contact between particles that have no chance of interaction. An example is the octree search (Lohner, 1988) in which the computational domain is split into cells, and only cells that hold a large number of particles are split into smaller cells recursively. In this way, cells of different size are created with a desired/designated number of particles in each cell. This method will have a search order of $O(N\log_8 N)$ for the three-dimensional case. The octree search is efficient for the calculation of particulate systems, for there is no need to reconstruct the octree frequently (Boyalakuntla, 2003). Unfortunately, in the case of milling simulation, the method is less efficient as a large number of fine particles are moving much faster than the few larger particles, and massive reconstruction is needed.

The above methods are unstructured methods; that is, the lists or cells are constructed depending on the dynamics of the system. This makes them less efficient when the system is very dynamic, owing to frequent cell refinement or memory allocation. The no binary search (NBS) proposed by Munjiza and Andrews (1998) is a structured method that demands more memory but in return has a search order of O(N). In this method, the computational domain is divided into cells of equal size. The particles that are located within the cells are linked to them. Interaction is calculated only between particles within the same cell and between particles in neighboring cells. Obviously, the calculation is less efficient when a cell holds more particles, and more memory is required when each cell is smaller. One way of reducing the memory requirement and simulation time is to remove unnecessary cells (empty cells) from the simulation. This approach is followed in the present work and was recently presented by Ogarko & Luding (2010). The cell cannot be smaller than the largest particles in the system as this may lead to erroneous contact detection. The search is particularly suitable for particles of similar size. This is not the case in milling simulations, where one cell can hold a single product particle but hundreds of fine particles.

The NBS is the most suitable search for the present work as it is the fastest and does not require frequent cell/list generation. The advantages of the method and improvements made to the method in the present work are presented in the following section.

2.1. DEM cell

As discussed above, to employ the NBS algorithm, a set of uniform cells (DEM cells) is generated by mapping the computational domain into a Cartesian DEM grid. The DEM grid is generated by dividing the width, length, and height of the computational domain into cubic uniform cells. Particles are assigned to a DEM cell according to their position by dividing the projected distance on each axis between the particle and the first node of the grid to the size of the DEM cell. This gives the indices that correspond to the DEM cell position in the three-dimensional array. Additionally, it is easy to add new empty DEM cells to the list.

Some improvements were made to the search algorithm. The first and obvious improvement is that once contact is detected, the force acting between particles is stored by both particles. This means that once a particle has completed the calculation of the forces acting on it, it can move to its new position. There is no need to repeat the search for the second particle as the force has already been calculated for that interaction. This means that each particle in each cell has to check for interaction with particles in only 13 (3^2 above + 3 to the side + 1 in front = 13) of the 26 neighboring cells (reducing the search time almost by half). Obviously, if there is more than one particle in the cell, the particles of the same cell have to look for interactions among themselves. This algorithm can be used only if the DEM cell is larger than the largest particle in the simulation (i.e., the DEM cell edge is larger than the diameter of the largest particles). The above limitation makes it possible to have several particles in one DEM cell. To reduce the overhead of calculating the interaction between particles at the edge of the DEM cell, another improvement of the search algorithm is the introduction of a condition checking the need for a particle to look for interactions with neighboring cells. That is, only if a particle is close enough to the face/edge/node of a neighboring cell does the algorithm sweep the cell for interactions. This condition can be verified quickly by storing the maximal and minimal coordinates of the DEM cell as part of the DEM cell structure and subtracting the distance between the particle center and the maximal coordinates of the DEM cell (separately in each principal direction). The search procedure is illustrated in Fig. 1 (in two dimensions). Particles marked in solid black are particles for which all forces acting on them have been calculated and the particles have moved to their new position (in the current time-step). The dashed particle is the particle for which the search is being conducted (active particle) and the dotted

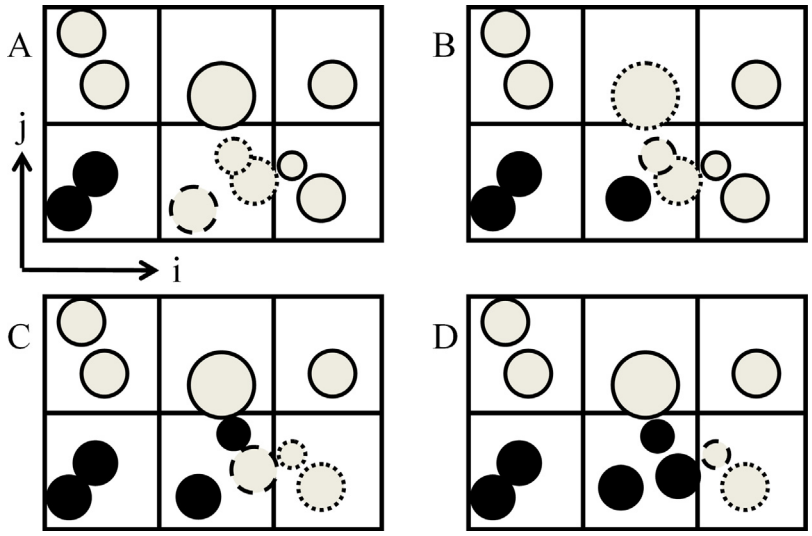

Fig. 1. NBS illustration in stages. Positions of solid black particles have been updated, the dashed particle is the active particle, and the algorithm is searching for interaction with the dotted particles.

particles are the particles that the algorithm is scanning for interaction (target particles).

At the top left (A) in the figure, the active particle is not close to any of the edges that may be scanned. The active particle scans only target particles within its own DEM cell. Once the active particle completes all the calculations needed, it moves to its new position (and is denoted in solid black) and the next particle in the DEM cell is called. At the top right (B), the new active particle is close to the top edge of the DEM cell, and target particles in the DEM cell above are thus scanned as well as particles within the active particle's own DEM cell. Again, the active particle moves to its new position once the calculations are completed. At the bottom left (C), the active particle is the last particle in the DEM cell and it thus has no particles in its own cell to scan; however, it is close to the edge on the right, and there are thus target particles in the neighboring DEM cell. At the bottom right (D), after all particles in the DEM cells have moved to their new positions, the following DEM cell is called and the search continues for the next active particle in the new DEM cell.

In the case of uniformly dispersed particles, each DEM cell should hold a small number of particles. However, a problem may occur if a uniform cell size is used in the case of particles dispersed densely in some regions and sparsely in other regions. A grid composed of cells that are too large will result in a large computational effort in dense regions and a grid composed of DEM cells that are too small will have a high computational cost because of the many empty cells. To resolve this problem, a hashed DEM grid was developed for the simulation (Kelager, 2006; Ogarko & Luding, 2012; Perkins & Williams, 2001). Besides being used extensively in smoothed particle hydrodynamics simulations, this method is suitable for our purpose (Perkins & Williams, 2001). In this method, the DEM grid is generated over the entire domain. Particles are linked to each DEM cell, and a list or table of all the DEM cells holding at least one particle is created (a hash table, or list in our case). The hash list is updated only if a particle moves into an empty DEM cell or the last particle exits a DEM cell. When using this method, a search from the order of $O(N_{particles})$ is expected for any system. If the hashed list is not used, the additional computational effort will make the simulation run more slowly; e.g., in the case of a heavily dispersed system where there are more DEM cells than particles, the search efficiency will be of the order of $O(N_{DEM\ cell})$.

2.2. Sub-DEM grid

As mentioned earlier, simulating processes with a wide particle size distribution (e.g., the particle size distribution in milling) are problematic (Particles are fed at sizes in the range of 10^{-4} m and are ground down to the size of 10^{-6} m). Considering a simple breakage scenario of a particle that breaks into fragments one-fifth of its size, suddenly 125 fragments are added to the simulation and each particle in the vicinity of those fragments carries the burden of 125 more calculations. Adding to the computational burden, the search for interaction between the new particles is also massive; i.e., $125 \times 124/2 = 7750$ possible interactions are added to the calculation. We put aside the interaction between fragments upon brakeage; this interaction is unavoidable and is discussed in detail in our previous work on DEM comminution simulations (Brosh, Kalman, & Levy, 2011). Once fragments are detached, the calculation effort is enormous until the fragments drift apart. To resolve this while still using the relatively simple DEM grid with an NBS, the DEM cells are split into sub-DEM cells, and for fine particles, the NBS is performed only within the sub-DEM cell. Obviously, large particles still have to sweep all fine particles within the sub-DEM cell of the DEM cell but that calculation is not as intensive as searching between the many fine particles.

To illustrate the significance of this sub-cell generation, a domain with a wide particle size distribution is illustrated in Fig. 2; a standard NBS is used at the left (A) whereas the DEM cell is divided into sub-cells at the right (B). In Fig. 2(A), it is evident that each particle in each DEM cell searches for interactions inefficiently as each DEM cell holds a large number of particles. However, the cell size cannot be decreased as the largest particle in the simulation limits its minimal size. In Fig. 2(B), the DEM cell is divided into 3^2 sub-cells. It is clear that the calculation cost is far less since there are fewer particles in each sub-cell. For jet-milling simulation, it was found that the DEM cell should be divided into 5^3–6^3 sub-DEM cells to achieve the greatest acceleration of the algorithm.

Fig. 3 presents the normalized simulation time, which is the time it takes to run 100 time-steps divided by the minimal time it takes to run 100 time-steps for the same number of particles. It is clear that good speedup is achieved by dividing the DEM cell into 5^3–6^3 sub-cells. The figure shows that the simulation can run about four times as fast employing the sub-DEM grid. The change in the normalized time for the different numbers of particles was due to the

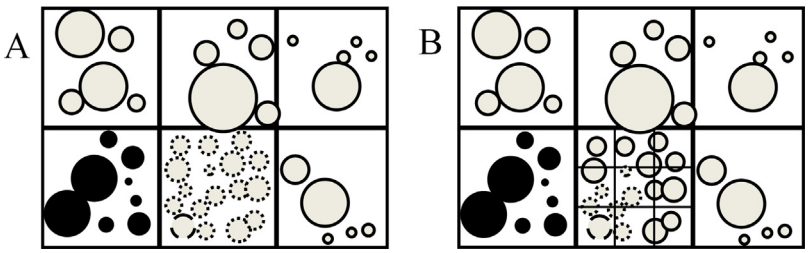

Fig. 2. Illustration of NBS without a sub-grid (A) and with a sub-grid (B) division into 3^2 sub-cells.

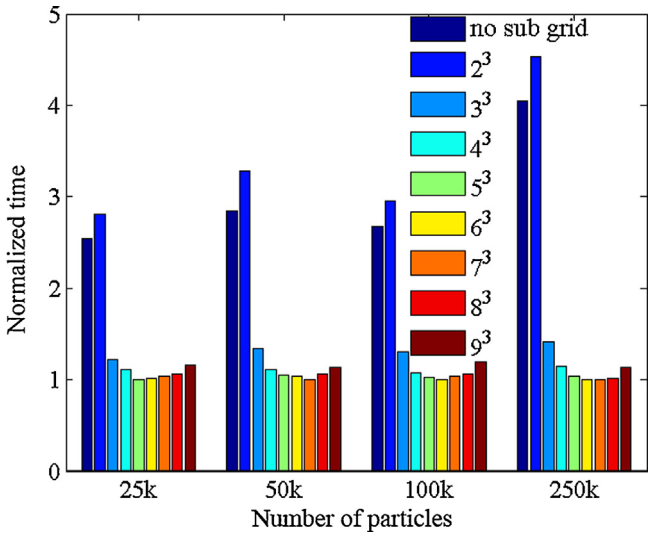

Fig. 3. Normalized simulation time under different sub-grid divisions for different numbers of particles.

size distribution of the particles within the selected section. Fig. 4 presents the size distribution of the particles within each section. The section containing 250k particles has the largest ratio of fine particles followed by the section containing 50k particles, and then the sections having 100k and 25k particles (i.e., the latter two sections have similar distributions). This agrees well with the speedup presented in Fig. 3, where the maximal speedup is obtained for 250k and then 50k particles, followed by 100k and 25k particles. Hence, the speedup is more efficient as the number of fine particles

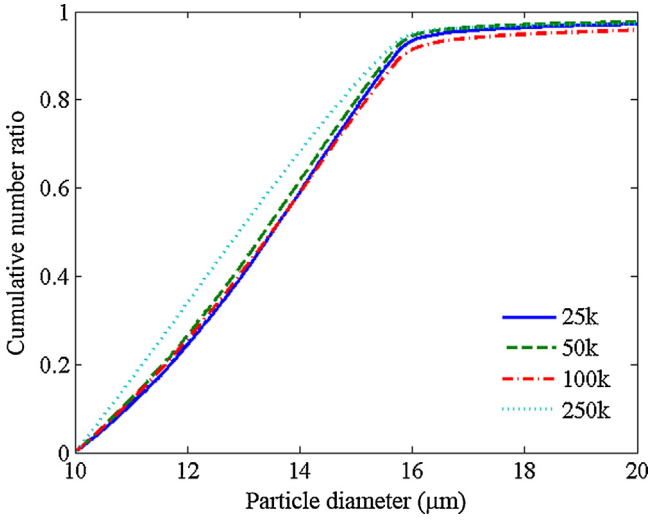

Fig. 4. Number based size under distribution in a random volume of the jet-mill.

increases. Recently, Ogarko and Luding (2012) presented a similar algorithm with two (or more) completely detached grids. In their work, they reported a much higher speedup; this was mostly due to the highly dense system that they investigated.

3. Reduced computational effort at the single-particle level

The above sections discussed the means of reducing the computational effort using an efficient search algorithm. However, as efficient as this algorithm may be, the simulation still has an enormous calculation cost as there are many particles in the system. Therefore, steps must also be taken to reduce the computational effort at the level of the single particle. Significant improvement can be achieved by reducing the calculation cost for each particle at each time-step and using as large a time-step as possible.

3.1. Particle–fluid interaction

The first and most common way to reduce the simulation time in DEM–CFD simulations is to let the CFD time-step be larger than the DEM time-step. This is a common practice in CFD–DEM simulations. However, extensive communication takes place while transferring flow properties between the CFD and DEM solvers. Therefore, several ways to reduce the communication time required in considering fluid–particle interactions were examined in this study. One way of reducing the calculation cost per particle is to use the mean fluid properties within the DEM cells discussed above. By holding the mean fluid properties at the DEM cell level, the particles have no need to interact with the CFD solver during each DEM time-step; instead, the interaction between the DEM and CFD is calculated at the beginning and the end of the DEM part of the simulation (i.e., once every CFD time-step), while the particles are retrieving the fluid properties from the DEM cell. The above procedure saves much computational time since the communication between the CFD solver and DEM solver is very time consuming.

Another way to reduce the calculation time for particle–fluid interactions is to calculate the drag and lift forces acting on the particle only if there is a significant change in the relative velocity between the particle and fluid. Owing to the small time-step needed for stable particle–wall and particle–particle interaction, stability is easily maintained for the particle–fluid interaction. Therefore, instead of using the formulation for calculating the drag and lift (see Brosh, Batat, Kalman, Levy, & Brown, 2008 for the formulation used in this work or the review by Zhu, Zhou, Yang, & Yu, 2007), each particle stores the drag and lift forces acting on it together with its velocity relative to the fluid, and for as long as the particle relative velocity does not change by more than a certain threshold (in any direction), the particle uses the same drag and lift. The algorithm for deciding whether to use the same drag or to calculate a new drag force is shown in Fig. 5. A comparison was made between a model in which the threshold was 5% and the model that calculates drag and lift in every DEM time-step. The model was validated for 0.1–10-mm salt particles accelerating to

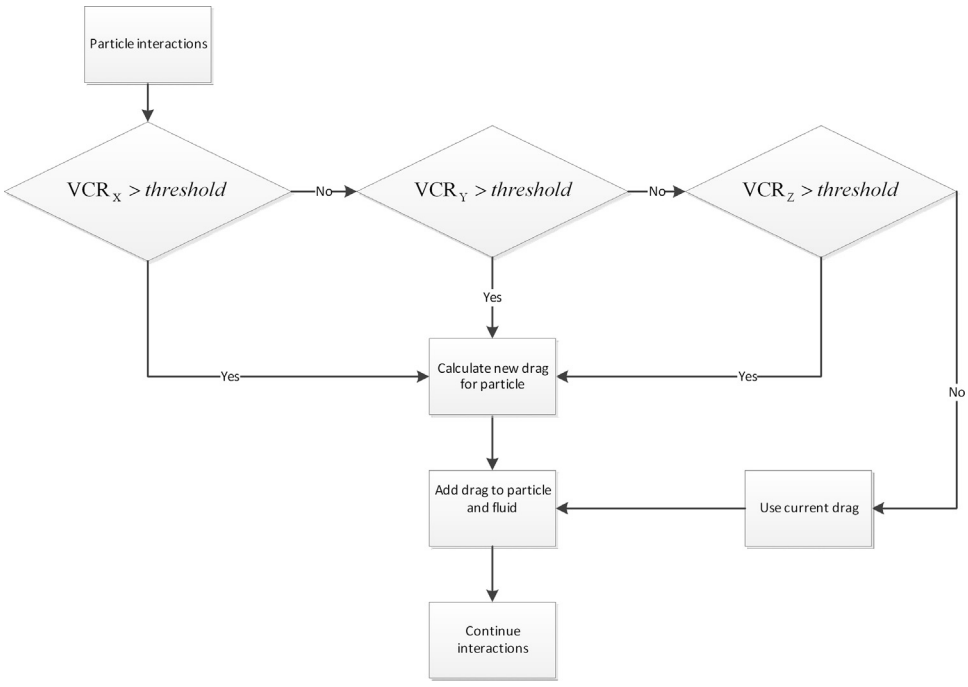

Fig. 5. The condition for recalculation of the drag and lift forces depending on the change in the relative velocity between the fluid and particle along each axis (X, Y, Z), where VCR_i is the velocity change ratio in each direction and defined as: $VCR_i = \left| \frac{V_{i,current} - V_{i,stored}}{V_{i,stored}} \right|$.

their terminal velocity. Under the above conditions, both models deviated by a maximum of 3% from the analytically expected terminal velocity, and it was found that calculation using the model with a threshold value of 5% was about 6% faster than the conventional calculation.

3.2. Particle–particle interaction

In DEM simulations, it is customary to calculate the equivalent collision parameters for particles of different size as a variation on series connection of the particle properties (Mishra & Murty, 2001; Teng, Wang, Zhang, & Gogos, 2011); e.g.,

$$E^* = \frac{1}{(1-\nu_1^2)/E_1 + (1-\nu_2^2)/E_2} \quad \text{or} \quad r^* = \frac{1}{1/r_1 + 1/r_2}, \tag{1}$$

where E, ν, and r are respectively the modulus of elasticity, Poisson's ratio, and radius of the particles. When using this approach, the stiffness and damping coefficients and the equivalent mass of both particles have to be recalculated for each collision and time-step. To reduce this excess calculation, we assume that the particle properties (i.e., spring and damper properties) are particle properties, not collision properties, therefore need to be calculated only once (upon generation of the particle). When collision commences, the individual spring and damper properties are summed in series for the normal part of the collision and in parallel for the tangential part. Therefore, for the normal part of the collision, one has

$$k_{np} = \frac{2\sqrt{2r}E_p}{3(1-\nu_p^2)} \rightarrow k_n = \left(\frac{1}{k_{npi}} + \frac{1}{k_{npj}} \right)^{-1},$$
$$\eta_{np} = 4\gamma_p \sqrt{k_{np} m_p} \rightarrow \eta_n = \left(\frac{1}{\eta_{npi}} + \frac{1}{\eta_{npj}} \right)^{-1}, \tag{2}$$

and for the tangential part of the collision, one has

$$k_{tp} = \frac{\sqrt{2r}E_p}{2(2-\nu_p)(1+\nu_p)} \delta_n^{1/2} \rightarrow k_t = k_{tpi} + k_{tpj},$$
$$\eta_{tp} = \gamma_p \sqrt{k_{tp} m_p} \rightarrow \eta_t = \eta_{tpi} + \eta_{tpj}, \tag{3}$$

where k, ν, η, and m are respectively the stiffness, damping coefficient, a constant correcting the damping coefficient, and mass of the particles.

This summation is presented in Fig. 6. It should be noted that the parameters of the spring and damper have the same values as the customarily calculated equivalent collision parameters when two particles having the same properties collide.

Fig. 7 shows the stiffness calculated using the mean properties and the series summation for particles having a diameter of 10–100 μm, colliding with a 30-μm particle. (Particle properties are given in Table 1.) When both particles are 30 μm in diameter

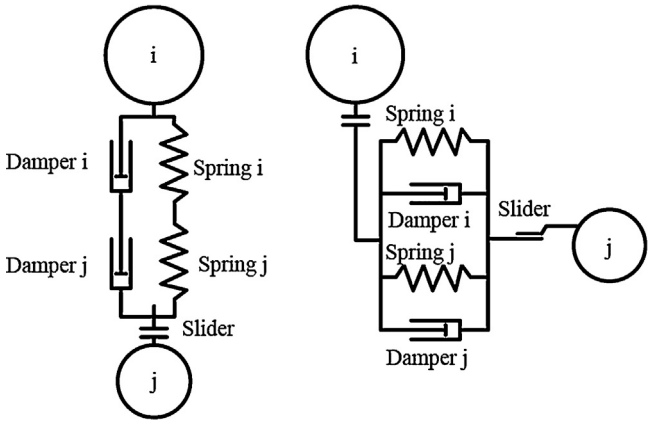

Fig. 6. Normal (left) and tangential (right) spring summation.

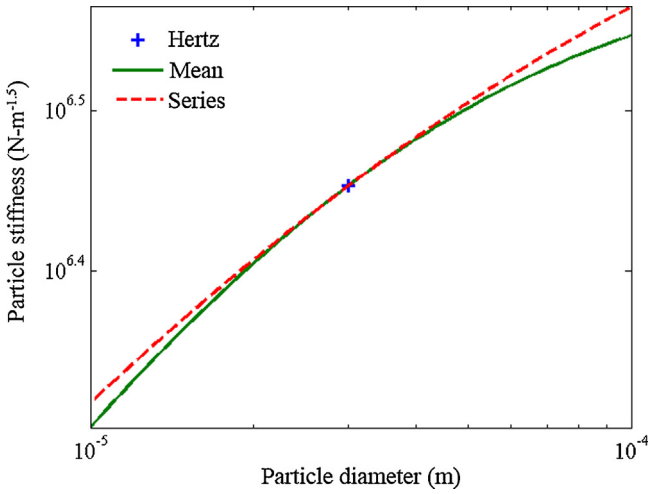

Fig. 7. Comparison of particle stiffness for a 30-μm particle under impact with particles having diameters of 10–100 μm.

Table 1
Particle properties for stiffness and damper comparison.

Poisson's ratio	0.3
Young's modulus (GPa)	1
Particle density (kg/m^3)	2000
Coefficient of restitution	0.5
Particle diameter (μm)	10–100

(Hertzian contact model, marked with the cross in Fig. 7), it is clear that both cases accurately represent the Hertz model.

However, the mean-property model does not predict the same damper coefficient as would be expected for particles of the same size. This can be seen in Fig. 8, in which the mean-property model for the damper (Raji & Favier, 2004) under-predicts the damper coefficient and may lead to false rebound velocities. The developed series model for calculating the normal forces during the collision is thus computationally friendly and appears to represent the collision more accurately.

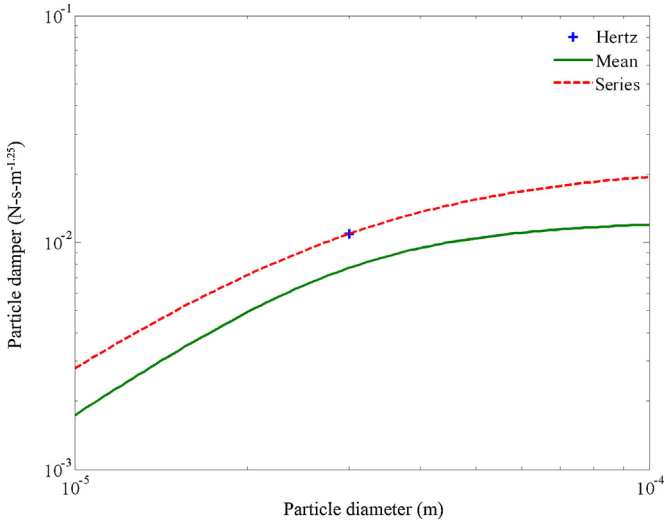

Fig. 8. Comparison of the particle damper for a 30-μm particle colliding with particles having diameters of 10–100 μm.

4. Reduced stiffness

To maintain a stable solution, the computational time-step is reduced as the particles become smaller. The condition for stability was deduced by Tsuji, Tanaka, and Ishida (1992):

$$\Delta t_{\max} \propto \sqrt{\frac{m_p}{k_n}}. \tag{4}$$

Since the stiffness is higher for particle–wall collision, it can be presented as a function of the properties of the particle and wall:

$$\Delta t_{\max} \propto \sqrt{\frac{m_p}{k_n}} \propto \sqrt{m_p \Big/ \frac{\sqrt{8d_p}}{\frac{3(1-\nu^2)}{E_p} + \frac{3(1-\nu_w^2)}{E_w}}} \propto d_p^{1.25}. \tag{5}$$

It is seen that, under the assumption of constant particle properties, the maximum allowed time-step is proportional to the diameter of the smallest particle in the simulation to the power of 1.25. When enabling particle breakage during the simulation, the number of particles in the simulation naturally increases and the size of the smallest particle in the simulation decreases. To reduce the actual simulation time and computational effort, when the number of particles in the simulation is controlled by the investigated operating conditions, the computational time-step should be set as large as possible. Kruggel-Emden, Stepanek, and Munjiza (2010) extended the work of Kruggel-Emden, Simsek, Rickelt, Wirtz, and Scherer (2007) and suggested reducing the simulation time by introducing several new stiffness models. The models presented by Kruggel-Emden, Stepanek, and Munjiza (2010) provided stability while working in larger time-steps; however, the stiffness models have new parameters that should be validated prior to the simulation.

In the present work, we propose a simple model that artificially reduces the stiffness of the smallest particles (which have the smallest resistance to drag). By artificially reducing the stiffness, the maximum allowed time-step is increased without greatly affecting the behavior of the system. This modification will distort slightly the duration of the collision time and the penetration of a particle into the wall or other particle. However, this modification is reasonable for as long as the flow is drag driven and not collision driven.

The new parameter for the particles' decreased modulus of elasticity can be calculated from Eq. (5) as

$$E_p = \min\left(\frac{\pi^3 d_p^{2.5}\rho_p E_w(1-\nu_p^2)}{5.657E_w\Delta t_{DEM}^2 - \pi^3 d_p^{2.5}\rho_p(1-\nu_w^2)}, E_{p,real}\right), \tag{6}$$

or that in terms of decreased normal stiffness can be calculated from Eq. (4) as

$$k_{n\,modified} = k_n \min\left(\left(\frac{d_p}{d_{threshold}}\right)^{2.5}, 1\right). \tag{7}$$

This manipulation can be effective; however, since we do not want to allow particles to penetrate too far into the wall, it was decided to limit the penetration to 15% of the displacement–diameter ratio. Fig. 9 shows that for a salt particle with typical impact velocity of 40 m/s and coefficient of restitution of 0.5 (for the sake of illustration), the threshold diameter (d_{th}) limit is 25 μm, which results in a speedup factor (Δt_{ratio}) of about 3. The figure was generated by storing the velocity and displacement of each particle from the beginning of the collision until it is no longer it touches with the wall. Fig. 9 also shows that the rebound velocity is slightly over-predicted, but this over-prediction is within the acceptable range.

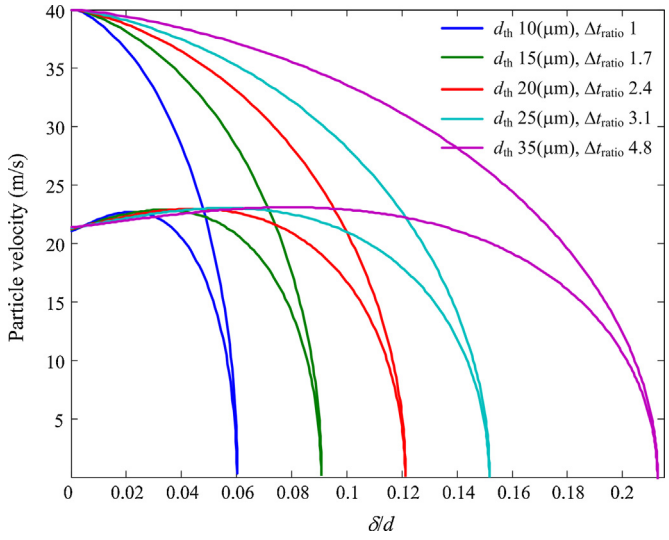

Fig. 9. Displacement–diameter ratio at an impact velocity of 40 m/s.

As a precaution in the event of too great penetration, the particle's normal velocity is set to the expected rebound velocity. This is done by multiplying the velocity at the point of the too great penetration by the coefficient of restitution and reversing its direction:

$$V_n^{t+\Delta t} = -eV_n^t, \tag{8}$$

where e is the coefficient of restitution. This may result in a rebound velocity that is too low if the particles are too soft. However, this situation should not occur under normal running conditions. Fig. 10 shows that in the unlikely event of penetration that is too great, the rebound velocity does not change much owing to the above correction.

The maximal displacement-to-diameter ratio (δ_{max}/d) is an important parameter when stiffness is manipulated in the simulation to avoid too large penetration into the wall. To obtain the formulation for the maximal displacement, a complicated nonlinear differential equation has to be solved:

$$m\ddot{x} = kx^{3/2} - \eta x^{1/4}\dot{x}. \tag{9}$$

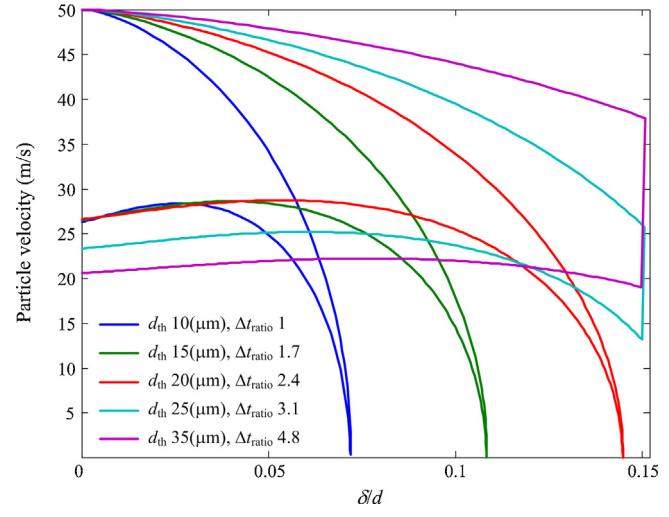

Fig. 10. Displacement–diameter ratio at an impact velocity of 50 m/s with limits of penetration.

Table 2
Particle and wall properties for maximal displacement simulations.

Property	Range
Particle diameter (μm)	1–10000
Particle Young's modulus (Pa)	10^7–10^{11}
Particle Poisson's ratio	0.34
Particle density (kg/m^3)	500–5000
Particle-wall coefficient of restitution	0.4–1
Wall Young's modulus (Pa)	10^9–10^{11}
Wall Poisson's ratio	0.3
Particle-wall impact velocity (m/s)	1–100
Time step	1/10 of the maximum allowed time step

Therefore, we initially solve the simple case of a nonlinear spring without a dashpot:

$$m\ddot{x} = kx^{3/2} \tag{10}$$

By eliminating the time dependence, we obtain a simple differential equation:

$$mV\frac{dV}{dx} = kx^{3/2} \tag{11}$$

Since the boundary conditions are known and we are interested only in the maximal displacement, all we have to do is to integrate both sides of Eq. (11) from $x = 0$ to the maximal displacement and from $V = V_{impact}$ to $V = 0$, resulting in the expression

$$m\int_{V_{impact}}^{0} VdV = k\int_0^{\delta_{max}} x^{3/2}dx \rightarrow \delta_{max} = \left(\frac{5m}{4k}V_{impact}^2\right)^{2/5} \tag{12}$$

After derivation of the analytical solution for the maximal displacement, several thousands of numerical experiments were conducted to obtain an empirical expression for the maximal displacement with the dashpot. Since we know the maximal displacement is a maximum for a coefficient of restitution $e = 1$, we look for expressions that will reduce the maximal displacement as the coefficient of restitution decreases. We have obtained a very strong correlation as

$$\delta_{max} = (0.73 + 0.27(e - \gamma))\left(\frac{5m}{4k}V_{impact}^2\right)^{2/5} \tag{13}$$

The numerical simulations were conducted with various properties of the particle and wall as listed in Table 2. Fig. 11 clearly

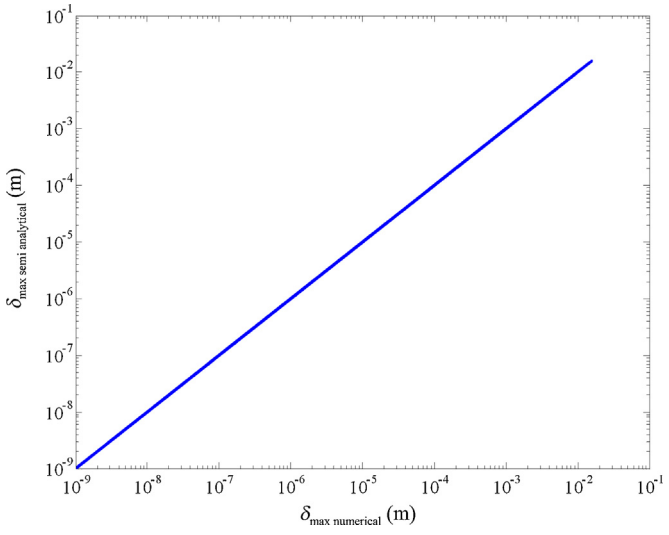

Fig. 11. Numerical vs. semi-analytical maximal displacement.

shows that the empirical correlation sits perfectly with the simulation results. The small difference (<0.3% for $\Delta t = (1/10)\Delta t_{max}$) between the prediction and numerical results is most likely due to the numerical accuracy of the integration scheme, as it decreases for smaller time-step ratios.

5. Reducing computational effort by eliminating very fine particles

When particles break, a considerable number of ultra-fine particles are generated. The time-step that corresponds to these particles is very small and makes the calculation time unreasonable. Additionally, the computational effort due to the large number of these particles makes the calculation of their motion unfeasible. Keeping in mind that drag acting on these particles is far stronger than any other force acting on the particles (with the exception of attracting forces), we decided to treat these ultra-fine particles as dust particles. Dust particles are not simulated upon creation; instead, we assume that they exit the mill immediately and are stored in a table. In the table, the dust is sorted into several size fractions, enabling the analysis of the actual size distribution at the outlet. It is worth noting that this procedure makes the simulation feasible, but slightly reduces the ability of the simulation to predict agglomeration and caking. Nevertheless, running the simulation without this procedure results in a simulation time that is unrealistically long.

Several threshold diameters (where fragments smaller than the threshold diameter are treated as dust) were tested. The resulting size distributions of both dust particles and physical particles at the outlet of a jet mill were compared with the experimental predictions in the same manner discussed by Brosh, Levy, Kalman, Ricard, and Peyron (2010). The threshold diameter for our milling simulation was found to be 10 μm. Simulations with threshold diameters of 25–50 μm over-predicted the mean size distribution of the product because they failed to account for successive breakage, while simulations with threshold diameters of 5–10 μm were in reasonable agreement with the experimental data. However, since simulations with a threshold diameter of 5 μm took considerably longer, the threshold diameter was set to 10 μm throughout most of our simulation.

It is important to note that when holding data for dust particles in the range of 1–10 μm as in the case of our simulations, the 10-μm threshold diameter makes the simulation about 18 times faster (in terms of time-steps, and much faster when considering the number of particles). This is very efficient when the mean diameter of the product is in the range of 5–10 μm. When a finer product is desired, the threshold diameter should be set to a lower value to consider the breakage of finer particles, and for a coarser product, a larger threshold diameter may be used to reduce the simulation time.

6. Conclusions

This work presented methods for speeding up simulation of processes with a wide particle size distribution. A DEM grid was constructed to speed up the simulation. The DEM cells within the DEM grid are linked and hashed, making the simulation run much faster as the empty cells are skipped over in the simulation. Using the DEM cells, the CFD and DEM solvers could communicate in a fast and easy manner. It was found that by carrying out an NBS in the hashed DEM cell for particles of similar size, near linear search behavior is expected and the contact search algorithm is much faster.

When comminution functions were used, the wide size distribution and the large number of particles slowed the simulation rapidly. To reduce the numerical burden due to the wide size distribution, a sub-DEM grid was added to the search algorithm. The sub-DEM grid was found to be most efficient at one-fifth the scale of the 0.5-mm DEM cell, in which case the simulations were accelerated by a factor of about 3–5.

Owing to the limitations of the DEM when working with small particles, a new approach was developed to improve stability. In this approach, the stiffness of the smallest particles in the simulations was reduced, making the particles softer and therefore allowing the maximal time-step to be increased. To find the limitations of this method (i.e., maximal penetration of 15%), an expression for the maximal penetration when using the Hertz–Mindlin contact model was developed. The new expression depends only on particle properties and enables the prediction of the maximal force acting on the particle. Using this expression, a new particle–particle selection function was developed.

The reduced-stiffness approach allowed the simulation to run about three times as fast. However, this was not sufficient, and additional speedup was achieved by eliminating the finest particles in the simulations. This was done by choosing a threshold diameter from which downwards particles are considered as dust and are no longer simulated. Obviously, these dust particles are of interest in grinding simulations, and the size distribution of the dust is thus stored in the simulation for further analysis. Unfortunately, the dust particles are the particles most likely to agglomerate or cake; however, running the simulations with a large number of smaller particles was found to be too time consuming. Nevertheless, the simulation was found to produce reasonable results.

References

Brosh, T., Batat, Y., Kalman, H., Levy, A., & Brown, A. B. (2008). Particle motion and classification in a jet mill. *Bulk Solids & Powder–Science & Technology*, *3*(2), 83–88.

Brosh, T., Kalman, H., & Levy, A. (2011). Fragments spawning and interaction models for DEM breakage simulation. *Granular Matter*, *13*(6), 765–776.

Brosh, T., Levy, A., Kalman, H., Ricard, F., & Peyron, I. (2010 April). Implementation of comminution functions in jet-mill DEM simulation. In *In World Congress Particle Technology 6 (WCPT6)* Nuremberg, Germany.

Boyalakuntla, D. S. (2003). *Simulation of granular and gas–solid flows using discrete element method*. USA: Doctoral dissertation, Carnegie Mellon University.

Kelager, M. (2006). *Lagrangian fluid dynamics using smoothed particle hydrodynamics*. Denmark: Master's thesis, University of Copenhagen.

Kruggel-Emden, H., Simsek, E., Rickelt, S., Wirtz, S., & Scherer, V. (2007). Review and extension of normal force models for the discrete element method. *Powder Technology*, *171*(3), 157–173.

Kruggel-Emden, H., Stepanek, F., & Munjiza, A. (2010). A study on adjusted contact force laws for accelerated large scale discrete element simulations. *Particuology*, *8*(2), 161–175.

Lohner, R. (1988). Some useful data structures for the generation of unstructured grids. *Communications in Applied Numerical Methods*, *4*, 123–135.

Mishra, B. K., & Murty, C. V. R. (2001). On the determination of contact parameters for realistic DEM simulations of ball mills. *Powder Technology*, *115*(3), 290–297.

Munjiza, A., & Andrews, K. R. F. (1998). NBS contact detection algorithm for bodies of similar size. *International Journal for Numerical Methods in Engineering*, *43*(1), 131–149.

Ogarko, V., & Luding, S. (2010 April). Data structures and algorithms for contact detection in numerical simulation of discrete particle systems. In *In World Congress on Particle Technology 6 (WCPT 6)* Nuremberg, Germany.

Ogarko, V., & Luding, S. (2012). A fast multilevel algorithm for contact detection of arbitrarily polydisperse objects. *Computer Physics Communications*, *183*(4), 931–936.

Perkins, E., & Williams, J. R. (2001). A fast contact detection algorithm insensitive to object sizes. *Engineering Computations*, *18*(1/2), 48–62.

Pöschel, T., & Schwager, T. (2005). *Computational granular dynamics: Models and algorithms*. Berlin: Springer.

Raji, A. O., & Favier, J. F. (2004). Model for the deformation in agricultural and food particulate materials under bulk compressive loading using discrete element method. I: Theory, model development and validation. *Journal of Food Engineering*, *64*(3), 359–371.

Teng, S., Wang, P., Zhang, Q., & Gogos, C. (2011). Analysis of fluid energy mill by gas-solid two-phase flow simulation. *Powder Technology*, *208*(3), 684–693.

Tsuji, Y., Tanaka, T., & Ishida, T. (1992). Lagrangian numerical simulation of plug flow of cohesionless particles in a horizontal pipe. *Powder Technology*, *71*(3), 239–250.

Zhu, H. P., Zhou, Z. Y., Yang, R. Y., & Yu, A. B. (2007). Discrete particle simulation of particulate systems: Theoretical developments. *Chemical Engineering Science*, *62*(13), 3378–3396.