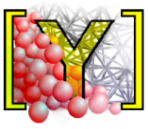

Yade DEM Formulation

Václav Šmilauer, Bruno Chareyre

February 17,2011
(1st edition - from release bzr2718)

Editor

Václav Šmilauer
CVUT Prague - lab. 3SR Grenoble University

Authors

Václav Šmilauer
CVUT Prague - lab. 3SR Grenoble University

Bruno Chareyre
Grenoble INP, UJF, CNRS, lab. 3SR

Citing this document

Please use the following reference, as explained at http://yade-dem/doc/citing.html:

V. Šmilauer, B. Chareyre (2010), **Yade dem formulation**. In *Yade Documentation* (V. Šmilauer, ed.), The Yade Project , 1st ed. (http://yade-dem.org/doc/formulation.html)

Abstract

In this chapter, we mathematically describe general features of explicit DEM simulations, with some reference to Yade implementation of these algorithms. They are given roughly in the order as they appear in the simulation loop: collisions detection, creating new interactions and determining their properties, kinematics representation, contact law, forces applied on particles.
Keywords: Yade, discrete element method, finite differences, collisions, contact representation.

Contents

Chapter 1

DEM Background

In this chapter, we mathematically describe general features of explicit DEM simulations, with some reference to Yade implementation of these algorithms. They are given roughly in the order as they appear in simulation; first, two particles might establish a new interaction, which consists in

1. detecting collision between particles;
2. creating new interaction and determining its properties (such as stiffness); they are either precomputed or derived from properties of both particles;

Then, for already existing interactions, the following is performed:

1. strain evaluation;
2. stress computation based on strains;
3. force application to particles in interaction.

This simplified description serves only to give meaning to the ordering of sections within this chapter. A more detailed description of this *simulation loop* is given later.

1.1 Collision detection

1.1.1 Generalities

Exact computation of collision configuration between two particles can be relatively expensive (for instance between Sphere and Facet). Taking a general pair of bodies $\mathtt{i}$ and $\mathtt{j}$ and their "exact" (In the sense of precision admissible by numerical implementation.) spatial predicates (called Shape in Yade) represented by point sets $P_\mathtt{i}$, $P_\mathtt{j}$ the detection generally proceeds in 2 passes:

1. fast collision detection using approximate predicate $\tilde{P}_\mathtt{i}$ and $\tilde{P}_\mathtt{j}$; they are pre-constructed in such a way as to abstract away individual features of $P_\mathtt{i}$ and $P_\mathtt{j}$ and satisfy the condition

$$\forall \mathbf{x} \in \mathbb{R}^3 : x \in P_\mathtt{i} \Rightarrow x \in \tilde{P}_\mathtt{i} \tag{1.1}$$

   (likewise for $P_\mathtt{j}$). The approximate predicate is called "bounding volume" (Bound in Yade) since it bounds any particle's volume from outside (by virtue of the implication). It follows that $(P_\mathtt{i} \cap P_\mathtt{j}) \neq \emptyset \Rightarrow (\tilde{P}_\mathtt{i} \cap \tilde{P}_\mathtt{j}) \neq \emptyset$ and, by applying *modus tollens*,

$$\left(\tilde{P}_\mathtt{i} \cap \tilde{P}_\mathtt{j}\right) = \emptyset \Rightarrow \left(P_\mathtt{i} \cap P_\mathtt{j}\right) = \emptyset \tag{1.2}$$

   which is a candidate exclusion rule in the proper sense.

2. By filtering away impossible collisions in (1.2), a more expensive, exact collision detection algorithms can be run on possible interactions, filtering out remaining spurious couples $(\tilde{P}_i \cap \tilde{P}_j) \neq \emptyset \wedge (P_i \cap P_j) = \emptyset$. These algorithms operate on P_i and P_j and have to be able to handle all possible combinations of shape types.

It is only the first step we are concerned with here.

1.1.2 Algorithms

Collision evaluation algorithms have been the subject of extensive research in fields such as robotics, computer graphics and simulations. They can be roughly divided in two groups:

Hierarchical algorithms which recursively subdivide space and restrict the number of approximate checks in the first pass, knowing that lower-level bounding volumes can intersect only if they are part of the same higher-level bounding volume. Hierarchy elements are bounding volumes of different kinds: octrees [Jung1997], bounding spheres [Hubbard1996], k-DOP's [Klosowski1998].

Flat algorithms work directly with bounding volumes without grouping them in hierarchies first; let us only mention two kinds commonly used in particle simulations:

> **Sweep and prune** algorithm operates on axis-aligned bounding boxes, which overlap if and only if they overlap along all axes. These algorithms have roughly $\mathcal{O}(n \log n)$ complexity, where n is number of particles as long as they exploit temporal coherence of the simulation.
>
> **Grid algorithms** represent continuous R^3 space by a finite set of regularly spaced points, leading to very fast neighbor search; they can reach the $\mathcal{O}(n)$ complexity [Munjiza1998] and recent research suggests ways to overcome one of the major drawbacks of this method, which is the necessity to adjust grid cell size to the largest particle in the simulation ([Munjiza2006], the "multistep" extension).

Temporal coherence expresses the fact that motion of particles in simulation is not arbitrary but governed by physical laws. This knowledge can be exploited to optimize performance.

Numerical stability of integrating motion equations dictates an upper limit on Δt (sect. *sect-formulation-dt*) and, by consequence, on displacement of particles during one step. This consideration is taken into account in [Munjiza2006], implying that any particle may not move further than to a neighboring grid cell during one step allowing the $\mathcal{O}(n)$ complexity; it is also explored in the periodic variant of the sweep and prune algorithm described below.

On a finer level, it is common to enlarge $\tilde{P}_i$ predicates in such a way that they satisfy the (1.1) condition during *several* timesteps; the first collision detection pass might then be run with stride, speeding up the simulation considerably. The original publication of this optimization by Verlet [Verlet1967] used enlarged list of neighbors, giving this technique the name *Verlet list*. In general cases, however, where neighbor lists are not necessarily used, the term *Verlet distance* is employed.

1.1.3 Sweep and prune

Let us describe in detail the sweep and prune algorithm used for collision detection in Yade (class InsertionSortCollider). Axis-aligned bounding boxes (Aabb) are used as $\tilde{P}_i$; each Aabb is given by lower and upper corner $\in R^3$ (in the following, $\tilde{P}_i^{x0}$, $\tilde{P}_i^{x1}$ are minimum/maximum coordinates of $\tilde{P}_i$ along the x-axis and so on). Construction of Aabb from various particle Shape's (such as Sphere, Facet, Wall) is straightforward, handled by appropriate classes deriving form BoundFunctor (Bo1_Sphere_Aabb, Bo1_Facet_Aabb, ...).

Presence of overlap of two Aabb's can be determined from conjunction of separate overlaps of intervals along each axis (fig-sweep-and-prune):

$$\left(\tilde{P}_i \cap \tilde{P}_j\right) \neq \emptyset \Leftrightarrow \bigwedge_{w \in \{x,y,z\}} \left[\left(\left(\tilde{P}_i^{w0}, \tilde{P}_i^{w1}\right) \cap \left(\tilde{P}_j^{w0}, \tilde{P}_j^{w1}\right)\right) \neq \emptyset\right]$$

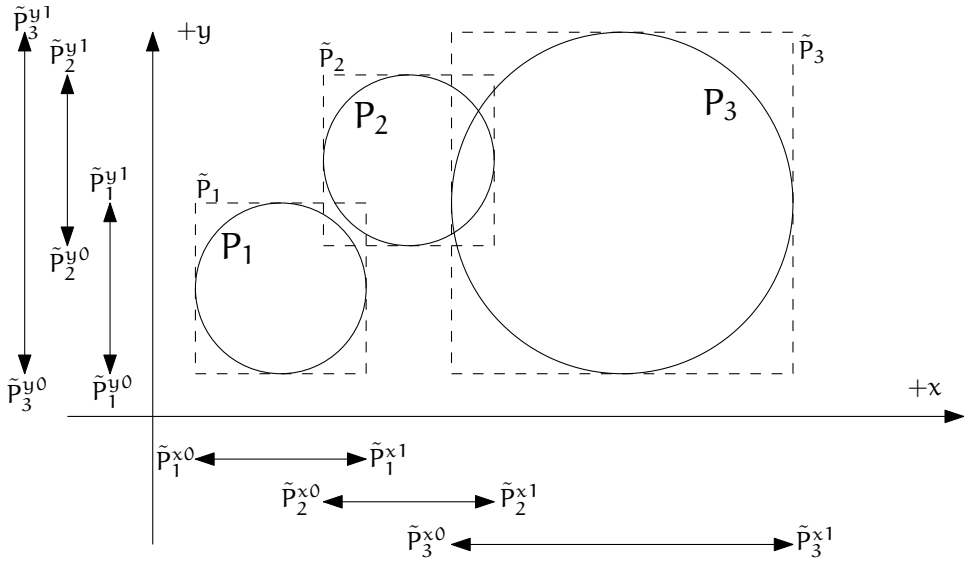

Figure 1.1: Sweep and prune algorithm (shown in 2D), where Aabb of each sphere is represented by minimum and maximum value along each axis. Spatial overlap of Aabb's is present if they overlap along all axes. In this case, $\tilde{P}_1 \cap \tilde{P}_2 \neq \emptyset$ (but note that $P_1 \cap P_2 = \emptyset$) and $\tilde{P}_2 \cap \tilde{P}_3 \neq \emptyset$.}

where (a, b) denotes interval in $\mathbb{R}$.

The collider keeps 3 separate lists (arrays) L_w for each axis $w \in \{x, y, z\}$

$$L_w = \bigcup_i \left\{ \tilde{P}_i^{w0}, \tilde{P}_i^{w1} \right\}$$

where i traverses all particles. L_w arrays (sorted sets) contain respective coordinates of minimum and maximum corners for each Aabb (we call these coordinates *bound* in the following); besides bound, each of list elements further carries `id` referring to particle it belongs to, and a flag whether it is lower or upper bound.

In the initial step, all lists are sorted (using quicksort, average $\mathcal{O}(n \log n)$) and one axis is used to create initial interactions: the range between lower and upper bound for each body is traversed, while bounds in-between indicate potential Aabb overlaps which must be checked on the remaining axes as well.

At each successive step, lists are already pre-sorted. Inversions occur where a particle's coordinate has just crossed another particle's coordinate; this number is limited by numerical stability of simulation and its physical meaning (giving spatio-temporal coherence to the algorithm). The insertion sort algorithm swaps neighboring elements if they are inverted, and has complexity between bigO{n} and bigO{n^2}, for pre-sorted and unsorted lists respectively. For our purposes, we need only to handle inversions, which by nature of the sort algorithm are detected inside the sort loop. An inversion might signify:

- overlap along the current axis, if an upper bound inverts (swaps) with a lower bound (i.e. that the upper bound with a higher coordinate was out of order in coming before the lower bound with a lower coordinate). Overlap along the other 2 axes is checked and if there is overlap along all axes, a new potential interaction is created.
- End of overlap along the current axis, if lower bound inverts (swaps) with an upper bound. If there is only potential interaction between the two particles in question, it is deleted.
- Nothing if both bounds are upper or both lower.

Aperiodic insertion sort

Let us show the sort algorithm on a sample sequence of numbers:

$$\| \quad 3 \quad 7 \quad 2 \quad 4 \quad \|$$

Elements are traversed from left to right; each of them keeps inverting (swapping) with neighbors to the left, moving left itself, until any of the following conditions is satisfied:

($\leq$)	the sorting order with the left neighbor is correct, or
($\|\|$)	the element is at the beginning of the sequence.

We start at the leftmost element (the current element is marked $\boxed{\mathtt{i}}$)

$$\| \quad \boxed{3} \quad 7 \quad 2 \quad 4 \quad \|.$$

It obviously immediately satisfies ($\|\|$), and we move to the next element:

$$\| \quad 3 \overset{\leq}{\leftarrow} \boxed{7} \quad 2 \quad 4 \quad \|.$$

Condition ($\leq$) holds, therefore we move to the right. The $\boxed{2}$ is not in order (violating ($\leq$)) and two inversions take place; after that, ($\|\|$) holds:

$$\begin{aligned}
&\| \quad 3 \quad 7 \overset{\not\leq}{\leftarrow} \boxed{2} \quad 4 \quad \|, \\
&\| \quad 3 \overset{\not\leq}{\leftarrow} \boxed{2} \quad 7 \quad 4 \quad \|, \\
&\| \quad \boxed{2} \quad 3 \quad 7 \quad 4 \quad \|.
\end{aligned}$$

The last element $\boxed{4}$ first violates ($\leq$), but satisfies it after one inversion

$$\begin{aligned}
&\| \quad 2 \quad 3 \quad 7 \overset{\not\leq}{\leftarrow} \boxed{4} \quad \|, \\
&\| \quad 2 \quad 3 \overset{\leq}{\leftarrow} \boxed{4} \quad 7 \quad \|.
\end{aligned}$$

All elements having been traversed, the sequence is now sorted.

It is obvious that if the initial sequence were sorted, elements only would have to be traversed without any inversion to handle (that happens in $\mathcal{O}(\mathtt{n})$ time).

For each inversion during the sort in simulation, the function that investigates change in Aabb overlap is invoked, creating or deleting interactions.

The periodic variant of the sort algorithm is described in *sect-periodic-insertion-sort*, along with other periodic-boundary related topics.

Optimization with Verlet distances

As noted above, [Verlet1967] explored the possibility of running the collision detection only sparsely by enlarging predicates $\tilde{\mathtt{P}}_\mathtt{i}$.

In Yade, this is achieved by enlarging Aabb of particles by fixed relative length in all dimensions $\Delta\mathtt{L}$ (InsertionSortCollider.sweepLength). Suppose the collider run last time at step $\mathtt{m}$ and the current step is

n. NewtonIntegrator tracks maximum distance traversed by particles (via maximum velocity magnitudes $v^{\circ}_{max} = \max|\dot{u}^{\circ}_i|$ in each step, with the initial cummulative distance $L_{max} = 0$,

$$L^{\circ}_{max} = L^{-}_{max} + v^{\circ}_{max}\Delta t^{\circ} \tag{1.3}$$

triggering the collider re-run as soon as

$$L^{\circ}_{max} > \Delta L. \tag{1.4}$$

The disadvantage of this approach is that even one fast particle determines v°_{max}.

A solution is to track maxima per particle groups. The possibility of tracking each particle separately (that is what ESyS-Particle does) seemed to us too fine-grained. Instead, we assign particles to b_n (InsertionSortCollider.nBins) *velocity bins* based on their current velocity magnitude. The bins' limit values are geometrical with the coefficient $b_c > 1$ (InsertionSortCollider.binCoeff), the maximum velocity being the current global velocity maximum v°_{max} (with some constraints on its change rate, to avoid large oscillations); for bin $i \in \{0, \dots, b_n\}$ and particle j:

$$v^{\circ}_{max} b_c^{-(i+1)} \leq |\dot{u}^{\circ}_j| < v_{max} b_c^{-i}.$$

(note that in this case, superscripts of b_c mean exponentiation). Equations (1.3)–(1.4) are used for each bin separately; however, when (1.4) is satisfied, full collider re-run is necessary and all bins' distances are reset.

Particles in high-speed oscillatory motion could be put into a slow bin if they happen to be at the point where their instantaneous speed is low, causing the necessity of early collider re-run. This is avoided by allowing particles to only go slower by one bin rather than several at once.

Results of using Verlet distance depend highly on the nature of simulation and choice of parameters InsertionSortCollider.nBins and InsertionSortColldier.binCoeff. The binning algorithm was specifically designed for simulating local fracture of larger concrete specimen; in that way, only particles in the fracturing zone, with greater velocities, had the Aabb's enlarged, without affecting quasi-still particles outside of this zone. In such cases, up to 50% overall computation time savings were observed, collider being run every 100 steps in average.

1.2 Creating interaction between particles

Collision detection described above is only approximate. Exact collision detection depends on the geometry of individual particles and is handled separately. In Yade terminology, the Collider creates only *potential* interactions; potential interactions are evaluated exactly using specialized algorithms for collision of two spheres or other combinations. Exact collision detection must be run at every timestep since it is at every step that particles can change their mutual position (the collider is only run sometimes if the Verlet distance optimization is in use). Some exact collision detection algorithms are described in *Strain evaluation*; in Yade, they are implemented in classes deriving from IGeomFunctor (prefixed with `Ig2`).

Besides detection of geometrical overlap (which corresponds to IGeom in Yade), there are also non-geometrical properties of the interaction to be determined (IPhys). In Yade, they are computed for every new interaction by calling a functor deriving from IPhysFunctor (prefixed with `Ip2`) which accepts the given combination of Material types of both particles.

1.2.1 Stiffnesses

Basic DEM interaction defines two stiffnesses: normal stiffness K_N and shear (tangent) stiffness K_T. It is desirable that K_N be related to fictitious Young's modulus of the particles' material, while K_T is typically determined as a given fraction of computed K_N. The K_T/K_N ratio determines macroscopic

Poisson's ratio of the arrangement, which can be shown by dimensional analysis: elastic continuum has two parameters (E and ν) and basic DEM model also has 2 parameters with the same dimensions K_N and K_T/K_N; macroscopic Poisson's ratio is therefore determined solely by K_T/K_N and macroscopic Young's modulus is then proportional to K_N and affected by K_T/K_N.

Naturally, such analysis is highly simplifying and does not account for particle radius distribution, packing configuration and other possible parameters such as the interaction radius introduced later.

Normal stiffness

The algorithm commonly used in Yade computes normal interaction stiffness as stiffness of two springs in serial configuration with lengths equal to the sphere radii (fig-spheres-contact-stiffness).

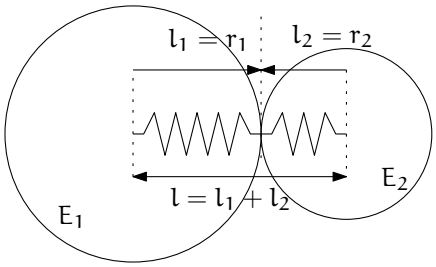

Figure 1.2: Series of 2 springs representing normal stiffness of contact between 2 spheres.

Let us define distance $l = l_1 + l_2$, where l_i are distances between contact point and sphere centers, which are initially (roughly speaking) equal to sphere radii. Change of distance between the spehre centers Δl is distributed onto deformations of both spheres $\Delta l = \Delta l_1 + \Delta l_2$ proportionally to their compliances. Displacement change Δl_i generates force $F_i = K_i \Delta l_i$, where K_i assures proportionality and has physical meaning and dimension of stiffness; K_i is related to the sphere material modulus E_i and some length $\tilde{l}_i$ proportional to r_i.

$$
\begin{aligned}
\Delta l &= \Delta l_1 + \Delta l_2 \\
K_i &= E_i \tilde{l}_i \\
K_N \Delta l &= F = F_1 = F_2 \\
K_N \left(\Delta l_1 + \Delta l_2\right) &= F \\
K_N \left(\frac{F}{K_1} + \frac{F}{K_2}\right) &= F \\
K_1^{-1} + K_2^{-1} &= K_N^{-1} \\
K_N &= \frac{K_1 K_2}{K_1 + K_2} \\
K_N &= \frac{E_1 \tilde{l}_1 E_2 \tilde{l}_2}{E_1 \tilde{l}_1 + E_2 \tilde{l}_2}
\end{aligned}
$$

The most used class computing interaction properties Ip2_FrictMat_FrictMat_FrictPhys uses $\tilde{l}_i = 2r_i$.

Some formulations define an equivalent cross-section A_{eq}, which in that case appears in the $\tilde{l}_i$ term as $K_i = E_i \tilde{l}_i = E_i \frac{A_{eq}}{l_i}$. Such is the case for the concrete model (Ip2_CpmMat_CpmMat_CpmPhys), where $A_{eq} = \min(r_1, r_2)$.

For reasons given above, no pretense about equality of particle-level E_i and macroscopic modulus E should be made. Some formulations, such as [Hentz2003], introduce parameters to match them numerically. This is not appropriate, in our opinion, since it binds those values to particular features of the sphere arrangement that was used for calibration.

1.2.2 Other parameters

Non-elastic parameters differ for various material models. Usually, though, they are averaged from the particles' material properties, if it makes sense. For instance, Ip2_CpmMat_CpmMat_CpmPhys averages most quantities, while Ip2_FrictMat_FrictMat_FrictPhys computes internal friction angle as $\varphi = \min(\varphi_1, \varphi_2)$ to avoid friction with bodies that are frictionless.

1.3 Strain evaluation

In the general case, mutual configuration of two particles has 6 degrees of freedom (DoFs) just like a beam in 3D space: both particles have 6 DoFs each, but the interaction itself is free to move and rotate in space (with both spheres) having 6 DoFs itself; then $12 - 6 = 6$. They are shown at fig-spheres-dofs.

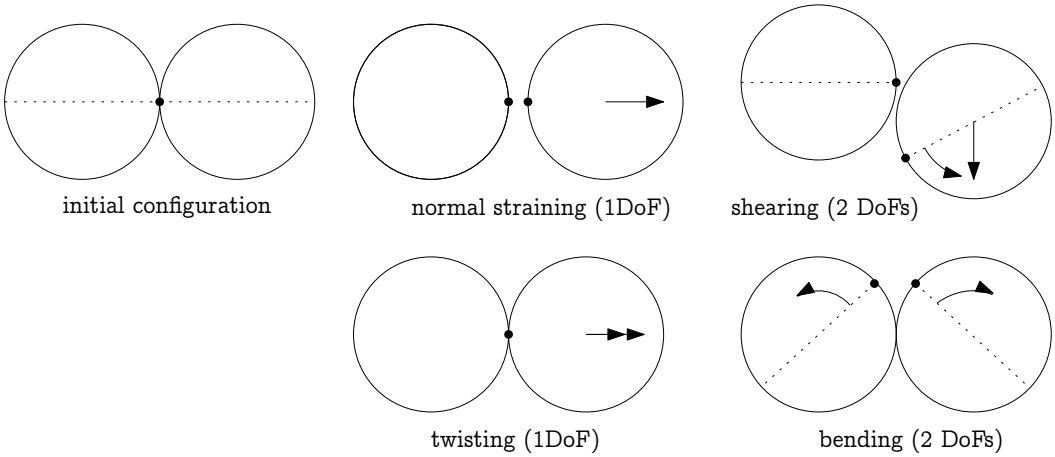

Figure 1.3: Degrees of freedom of configuration of two spheres. Normal strain appears if there is a difference of linear velocity along the interaction axis ($\mathbf{n}$); shearing originates from the difference of linear velocities perpendicular to $\mathbf{n}$ *and* from the part of $\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2$ perpendicular to $\mathbf{n}$; twisting is caused by the part of $\boldsymbol{\omega}_1 - \boldsymbol{\omega}_2$ parallel with $\mathbf{n}$; bending comes from the part of $\boldsymbol{\omega}_1 - \boldsymbol{\omega}_2$ perpendicular to $\mathbf{n}$.

We will only describe normal and shear components of strain in the following, leaving torsion and bending aside. The reason is that most constitutive laws for contacts do not use the latter two.

1.3.1 Normal strain

Constants

Let us consider two spheres with *initial* centers $\bar{\mathbf{C}}_1$, $\bar{\mathbf{C}}_2$ and radii r_1, r_2 that enter into contact. The order of spheres within the contact is arbitrary and has no influence on the behavior. Then we define lengths

$$d_0 = |\bar{\mathbf{C}}_2 - \bar{\mathbf{C}}_1|$$
$$d_1 = r_1 + \frac{d_0 - r_1 - r_2}{2}, \qquad d_2 = d_0 - d_1.$$

These quantities are *constant* throughout the life of the interaction and are computed only once when the interaction is established. The distance d_0 is the *reference distance* and is used for the conversion of absolute displacements to dimensionless strain, for instance. It is also the distance where (for usual contact laws) there is neither repulsive nor attractive force between the spheres, whence the name *equilibrium distance*.

Distances d_1 and d_2 define reduced (or expanded) radii of spheres; geometrical radii r_1 and r_2 are used only for collision detection and may not be the same as d_1 and d_2, as shown in fig. fig-sphere-sphere.

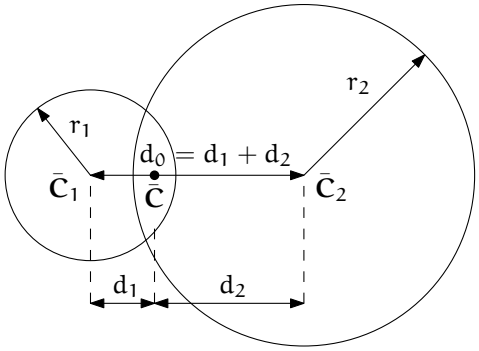

Figure 1.4: Geometry of the initial contact of 2 spheres; this case pictures spheres which already overlap when the contact is created (which can be the case at the beginning of a simulation) for the sake of generality. The initial contact point $\bar{\mathbf{C}}$ is in the middle of the overlap zone.

This difference is exploited in cases where the average number of contacts between spheres should be increased, e.g. to influence the response in compression or to stabilize the packing. In such case, interactions will be created also for spheres that do not geometrically overlap based on the *interaction radius* R_I, a dimensionless parameter determining „non-locality“ of contact detection. For $R_I = 1$, only spheres that touch are considered in contact; the general condition reads

$$d_0 \leq R_I(r_1 + r_2). \tag{1.5}$$

The value of R_I directly influences the average number of interactions per sphere (percolation), which for some models is necessary in order to achieve realistic results. In such cases, Aabb (or $\tilde{P}_i$ predicates in general) must be enlarged accordingly (Bo1_Sphere_Aabb.aabbEnlargeFactor).

Contact cross-section

Some constitutive laws are formulated with strains and stresses (Law2_Dem3DofGeom_CpmPhys_Cpm, the concrete model described later, for instance); in that case, equivalent cross-section of the contact must be introduced for the sake of dimensionality. The exact definition is rather arbitrary; the CPM model (Ip2_CpmMat_CpmMat_CpmPhys) uses the relation

$$A_{eq} = \pi \min(r_1, r_2)^2 \tag{1.6}$$

which will be used to convert stresses to forces, if the constitutive law used is formulated in terms of stresses and strains. Note that other values than π can be used; it will merely scale macroscopic packing stiffness; it is only for the intuitive notion of a truss-like element between the particle centers that we choose A_{eq} representing the circle area. Besides that, another function than $\min(r_1, r_2)$ can be used, although the result should depend linearly on r_1 and r_2 so that the equation gives consistent results if the particle dimensions are scaled.

Variables

The following state variables are updated as spheres undergo motion during the simulation (as $\mathbf{C}_1^\circ$ and $\mathbf{C}_2^\circ$ change):

$$\mathbf{n}^\circ = \frac{\mathbf{C}_2^\circ - \mathbf{C}_1^\circ}{|\mathbf{C}_2^\circ - \mathbf{C}_1^\circ|} \equiv \widehat{\mathbf{C}_2^\circ - \mathbf{C}_1^\circ} \tag{1.7}$$

and

$$\mathbf{C}^\circ = \mathbf{C}_1^\circ + \left(d_1 - \frac{d_0 - |\mathbf{C}_2^\circ - \mathbf{C}_1^\circ|}{2} \right) \mathbf{n}. \tag{1.8}$$

The contact point $\mathbf{C}^\circ$ is always in the middle of the spheres' overlap zone (even if the overlap is negative, when it is in the middle of the empty space between the spheres). The *contact plane* is always perpendicular to the contact plane normal $\mathbf{n}^\circ$ and passes through $\mathbf{C}^\circ$.

Normal displacement and strain can be defined as

$$u_N = |\mathbf{C}_2^\circ - \mathbf{C}_1^\circ| - d_0,$$
$$\varepsilon_N = \frac{u_N}{d_0} = \frac{|\mathbf{C}_2^\circ - \mathbf{C}_1^\circ|}{d_0} - 1.$$

Since u_N is always aligned with $\mathbf{n}$, it can be stored as a scalar value multiplied by $\mathbf{n}$ if necessary.

For massively compressive simulations, it might be beneficial to use the logarithmic strain, such that the strain tends to $-\infty$ (rather than -1) as centers of both spheres approach. Otherwise, repulsive force would remain finite and the spheres could penetrate through each other. Therefore, we can adjust the definition of normal strain as follows:

$$\varepsilon_N = \begin{cases} \log\left(\frac{|\mathbf{C}_2^\circ - \mathbf{C}_1^\circ|}{d_0}\right) & \text{if } |\mathbf{C}_2^\circ - \mathbf{C}_1^\circ| < d_0 \\ \frac{|\mathbf{C}_2^\circ - \mathbf{C}_1^\circ|}{d_0} - 1 & \text{otherwise.} \end{cases}$$

Such definition, however, has the disadvantage of effectively increasing rigidity (up to infinity) of contacts, requiring Δt to be adjusted, lest the simulation becomes unstable. Such dynamic adjustment is possible using a stiffness-based time-stepper (GlobalStiffnessTimeStepper in Yade).

1.3.2 Shear strain

In order to keep $\mathbf{u}_T$ consistent (e.g. that $\mathbf{u}_T$ must be constant if two spheres retain mutually constant configuration but move arbitrarily in space), then either $\mathbf{u}_T$ must track spheres' spatial motion or must (somehow) rely on sphere-local data exclusively.

These two possibilities lead to two algorithms of computing shear strains. They should give the same results (disregarding numerical imprecision), but there is a trade-off between computational cost of the incremental method and robustness of the total one.

Geometrical meaning of shear strain is shown in fig-shear-2d.

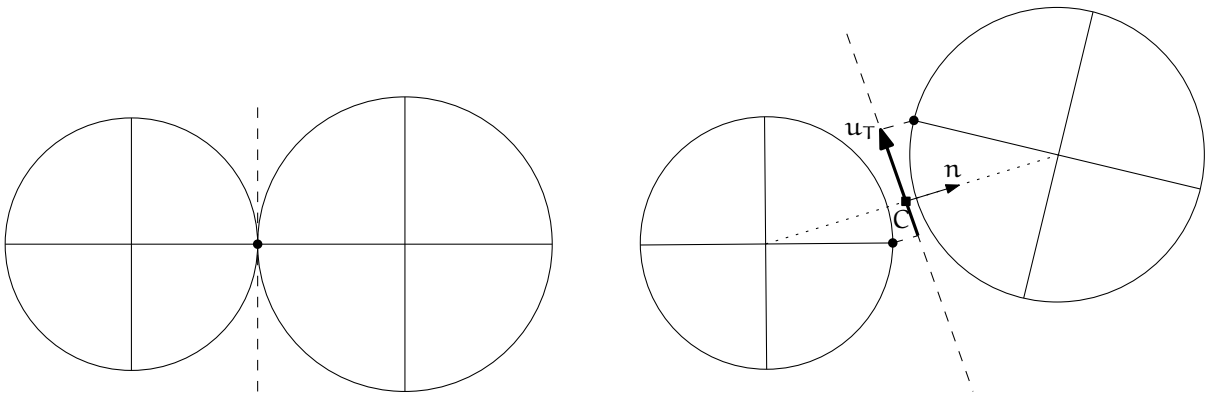

Figure 1.5: Evolution of shear displacement $\mathbf{u}_T$ due to mutual motion of spheres, both linear and rotational. Left configuration is the initial contact, right configuration is after displacement and rotation of one particle.

Incremental algorithm

The incremental algorithm is widely used in DEM codes and is described frequently ([Luding2008], [Alonso2004]). Yade implements this algorithm in the ScGeom class. At each step, shear displacement $\mathbf{u}_T$ is updated; the update increment can be decomposed in 2 parts: motion of the interaction (i.e. $\mathbf{C}$ and $\mathbf{n}$) in global space and mutual motion of spheres.

1. Contact moves dues to changes of the spheres' positions $\mathbf{C}_1$ and $\mathbf{C}_2$, which updates current $\mathbf{C}^\circ$ and $\mathbf{n}^\circ$ as per (1.8) and (1.7). $\mathbf{u}_T^-$ is perpendicular to the contact plane at the previous step $\mathbf{n}^-$ and must be updated so that $\mathbf{u}_T^- + (\Delta\mathbf{u}_T) = \mathbf{u}_T^\circ \perp \mathbf{n}^\circ$; this is done by perpendicular projection to the plane first (which might decrease $|\mathbf{u}_T|$) and adding what corresponds to spatial rotation of the interaction instead:

$$\begin{aligned}(\Delta\mathbf{u}_T)_1 &= -\mathbf{u}_T^- \times (\mathbf{n}^- \times \mathbf{n}^\circ)\\ (\Delta\mathbf{u}_T)_2 &= -\mathbf{u}_T^- \times \left(\frac{\Delta t}{2}\mathbf{n}^\circ \cdot (\boldsymbol{\omega}_1^{\ominus} + \boldsymbol{\omega}_2^{\ominus})\right)\mathbf{n}^\circ\end{aligned}$$

2. Mutual movement of spheres, using only its part perpendicular to $\mathbf{n}^\circ$; $\boldsymbol{\nu}_{12}$ denotes mutual velocity of spheres at the contact point:

$$\begin{aligned}\boldsymbol{\nu}_{12} &= \left(\boldsymbol{\nu}_2^{\ominus} + \boldsymbol{\omega}_2^- \times (-d_2\mathbf{n}^\circ)\right) - \left(\boldsymbol{\nu}_1^{\ominus} + \boldsymbol{\omega}_1^{\ominus} \times (d_1\mathbf{n}^\circ)\right)\\ \boldsymbol{\nu}_{12}^\perp &= \boldsymbol{\nu}_{12} - (\mathbf{n}^\circ \cdot \boldsymbol{\nu}_{12})\mathbf{n}^\circ\\ (\Delta\mathbf{u}_T)_3 &= -\Delta t\boldsymbol{\nu}_{12}^\perp\end{aligned}$$

Finally, we compute

$$\mathbf{u}_T^\circ = \mathbf{u}_T^- + (\Delta\mathbf{u}_T)_1 + (\Delta\mathbf{u}_T)_2 + (\Delta\mathbf{u}_T)_3.$$

Total algorithm

The following algorithm, aiming at stabilization of response even with large rotation speeds or Δt approaching stability limit, was designed in [Smilauer2010b]. (A similar algorithm based on total formulation, which covers additionally bending and torsion, was proposed in [Wang2009].) It is based on tracking original contact points (with zero shear) in the particle-local frame.

In this section, variable symbols implicitly denote their current values unless explicitly stated otherwise.

Shear strain may have two sources: mutual rotation of spheres or transversal displacement of one sphere with respect to the other. Shear strain does not change if both spheres move or rotate but are not in linear or angular motion mutually. To accurately and reliably model this situation, for every new contact the initial contact point $\bar{\mathbf{C}}$ is mapped into local sphere coordinates ($\mathbf{p}_{01}$, $\mathbf{p}_{02}$). As we want to determine the distance between both points (i.e. how long the trajectory in on both spheres' surfaces together), the shortest path from current $\mathbf{C}$ to the initial locally mapped point on the sphere's surface is „unrolled" to the contact plane ($\mathbf{p}'_{01}$, $\mathbf{p}'_{02}$); then we can measure their linear distance $\mathbf{u}_T$ and define shear strain $\boldsymbol{\varepsilon}_T = \mathbf{u}_T/d_0$ (fig. fig-shear-displacement).

More formally, taking $\bar{\mathbf{C}}_i$, $\bar{q}_i$ for the sphere initial positions and orientations (as quaterions) in global coordinates, the initial sphere-local contact point *orientation* (relative to sphere-local axis $\hat{x}$) is remembered:

$$\begin{aligned}\bar{\mathbf{n}} &= \widehat{\mathbf{C}_1 - \mathbf{C}_2},\\ \bar{q}_{01} &= \mathrm{Align}(\hat{x}, \bar{q}_1^* \bar{\mathbf{n}} \bar{q}_1^{**}),\\ \bar{q}_{02} &= \mathrm{Align}(\hat{x}, \bar{q}_2^* (-\bar{\mathbf{n}}) \bar{q}_2^{**}).\end{aligned}$$

After some spheres motion, the original point can be "unrolled" to the current contact plane:

$$\begin{aligned}q &= \mathrm{Align}(\mathbf{n}, q_1 \bar{q}_{01} \hat{x} (q_1 \bar{q}_{01})^*) \quad \text{(auxiliary)}\\ \mathbf{p}'_{01} &= q_\vartheta d_1 (q_\mathbf{u} \times \mathbf{n})\end{aligned}$$

where $q_\mathbf{u}$, q_ϑ are axis and angle components of q and p'_{01} is the unrolled point. Similarly,

$$\begin{aligned}q &= \mathrm{Align}(\mathbf{n}, q_2 \bar{q}_{02} \hat{x} (q_2 \bar{q}_{02})^*)\\ \mathbf{p}'_{02} &= q_\vartheta d_1 (q_\mathbf{u} \times (-\mathbf{n})).\end{aligned}$$

Shear displacement and strain are then computed easily:

$$\mathbf{u}_T = \mathbf{p}'_{02} - \mathbf{p}'_{01}$$
$$\boldsymbol{\varepsilon}_T = \frac{\mathbf{u}_T}{d_0}$$

When using material law with plasticity in shear, it may be necessary to limit maximum shear strain, in which case the mapped points are moved closer together to the requested distance (without changing $\hat{\mathbf{u}}_T$). This allows us to remember the previous strain direction and also avoids summation of increments of plastic strain at every step (fig-shear-slip).

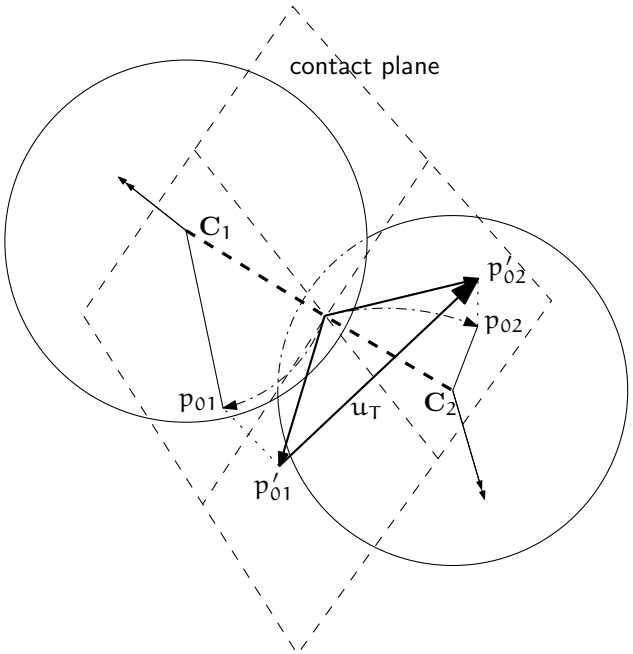

Figure 1.6: Shear displacement computation for two spheres in relative motion.

This algorithm is straightforwardly modified to facet-sphere interactions. In Yade, it is implemented by Dem3DofGeom and related classes.

1.4 Stress evaluation (example)

Once strain on a contact is computed, it can be used to compute stresses/forces acting on both spheres.

The constitutive law presented here is the most usual DEM formulation, originally proposed by Cundall. While the strain evaluation will be similar to algorithms described in the previous section regardless of stress evaluation, stress evaluation itself depends on the nature of the material being modeled. The constitutive law presented here is the most simple non-cohesive elastic case with dry friction, which Yade implements in Law2_Dem3DofGeom_FrictPhys_Basic (all constitutive laws derive from base class LawFunctor).

In DEM generally, some constitutive laws are expressed using strains and stresses while others prefer displacement/force formulation. The law described here falls in the latter category.

When new contact is established (discussed in *sect-simulation-loop*) it has its properties (IPhys) computed from Materials associated with both particles. In the simple case of frictional material FrictMat, Ip2_FrictMat_FrictMat_FrictPhys creates a new FrictPhys instance, which defines normal stiffness K_N, shear stiffness K_T and friction angle φ.

At each step, given normal and shear displacements u_N, $\mathbf{u}_T$, normal and shear forces are computed (if

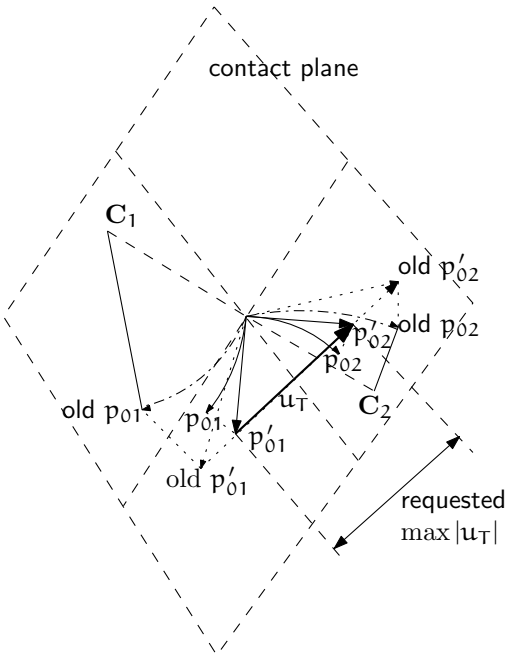

Figure 1.7: Shear plastic slip for two spheres.

$u_N > 0$, the contact is deleted without generating any forces):

$$\begin{aligned} F_N &= K_N u_N n, \\ F_T^t &= K_T u_T \end{aligned}$$

where F_N is normal force and F_T is trial shear force. A simple non-associated stress return algorithm is applied to compute final shear force

$$F_T = \begin{cases} F_T^t \frac{|F_N| \tan \varphi}{F_T^t} & \text{if } |F_T| > |F_N| \tan \varphi, \\ F_T^t & \text{otherwise.} \end{cases}$$

Summary force $F = F_N + F_T$ is then applied to both particles – each particle accumulates forces and torques acting on it in the course of each step. Because the force computed acts at contact point C, which is difference from spheres' centers, torque generated by F must also be considered.

$$\begin{aligned} F_1 &+= F & F_2 &+= -F \\ T_1 &+= d_1(-n) \times F & T_2 &+= d_2 n \times F. \end{aligned}$$

1.5 Motion integration

Each particle accumulates generalized forces (forces and torques) from the contacts in which it participates. These generalized forces are then used to integrate motion equations for each particle separately; therefore, we omit i indices denoting the i-th particle in this section.

The customary leapfrog scheme (also known as the Verlet scheme) is used, with some adjustments for rotation of non-spherical particles, as explained below. The "leapfrog" name comes from the fact that even derivatives of position/orientation are known at on-step points, whereas odd derivatives are known at mid-step points. Let us recall that we use a^-, a°, a^+ for on-step values of a at $t - \Delta t$, t and $t + \Delta t$ respectively; and $a^\ominus$, $a^\oplus$ for mid-step values of a at $t - \Delta t/2$, $t + \Delta t/2$.

Described integration algorithms are implemented in the NewtonIntegrator class in Yade.

1.5.1 Position

Integrating motion consists in using current acceleration $\ddot{\mathbf{u}}^{\circ}$ on a particle to update its position from the current value $\mathbf{u}^{\circ}$ to its value at the next timestep $\mathbf{u}^{+}$. Computation of acceleration, knowing current forces $\mathbf{F}$ acting on the particle in question and its mass m, is simply

$$\ddot{\mathbf{u}}^{\circ} = \mathbf{F}/m.$$

Using the 2nd order finite difference with step Δt, we obtain

$$\ddot{\mathbf{u}}^{\circ} \cong \frac{\mathbf{u}^{-} - 2\mathbf{u}^{\circ} + \mathbf{u}^{+}}{\Delta t^2}$$

from which we express

$$\begin{aligned}\mathbf{u}^{+} &= 2\mathbf{u}^{\circ} - \mathbf{u}^{-} + \ddot{\mathbf{u}}^{\circ}\Delta t^2 = \\ &= \mathbf{u}^{\circ} + \Delta t \underbrace{\left(\frac{\mathbf{u}^{\circ} - \mathbf{u}^{-}}{\Delta t} + \ddot{\mathbf{u}}^{\circ}\Delta t\right)}_{(\dagger)}.\end{aligned}$$

Typically, $\mathbf{u}^{-}$ is already not known (only $\mathbf{u}^{\circ}$ is); we notice, however, that

$$\dot{\mathbf{u}}^{\ominus} \simeq \frac{\mathbf{u}^{\circ} - \mathbf{u}^{-}}{\Delta t},$$

i.e. the mean velocity during the previous step, which is known. Plugging this approximate into the $(\dagger)$ term, we also notice that mean velocity during the current step can be approximated as

$$\dot{\mathbf{u}}^{\oplus} \simeq \dot{\mathbf{u}}^{\ominus} + \ddot{\mathbf{u}}^{\circ}\Delta t,$$

which is $(\dagger)$; we arrive finally at

$$\mathbf{u}^{+} = \mathbf{u}^{\circ} + \Delta t\left(\dot{\mathbf{u}}^{\ominus} + \ddot{\mathbf{u}}^{\circ}\Delta t\right).$$

The algorithm can then be written down by first computing current mean velocity $\dot{\mathbf{u}}^{\oplus}$ which we need to store for the next step (just as we use its old value $\dot{\mathbf{u}}^{\ominus}$ now), then computing the position for the next time step $\mathbf{u}^{+}$:

$$\begin{aligned}\dot{\mathbf{u}}^{\oplus} &= \dot{\mathbf{u}}^{\ominus} + \ddot{\mathbf{u}}^{\circ}\Delta t \\ \mathbf{u}^{+} &= \mathbf{u}^{\circ} + \dot{\mathbf{u}}^{\oplus}\Delta t.\end{aligned}$$

Positions are known at times $i\Delta t$ (if Δt is constant) while velocities are known at $i\Delta t + \frac{\Delta t}{2}$. The facet that they interleave (jump over each other) in such way gave rise to the colloquial name "leapfrog" scheme.

1.5.2 Orientation (spherical)

Updating particle orientation q° proceeds in an analogous way to position update. First, we compute current angular acceleration $\dot{\boldsymbol{\omega}}^{\circ}$ from known current torque $\mathbf{T}$. For spherical particles where the inertia tensor is diagonal in any orientation (therefore also in current global orientation), satisfying $\mathbf{I}_{11} = \mathbf{I}_{22} = \mathbf{I}_{33}$, we can write

$$\dot{\boldsymbol{\omega}}^{\circ}_{i} = \mathbf{T}_{i}/\mathbf{I}_{11},$$

We use the same approximation scheme, obtaining an equation analogous to (**??**)

$$\boldsymbol{\omega}^{\oplus} = \boldsymbol{\omega}^{\ominus} + \Delta t \dot{\boldsymbol{\omega}}^{\circ}.$$

The quaternion Δq representing rotation vector $\boldsymbol{\omega}^{\oplus}\Delta t$ is constructed, i.e. such that

$$(\Delta q)_{\vartheta} = |\boldsymbol{\omega}^{\oplus}|,$$
$$(\Delta q)_{\mathbf{u}} = \widehat{\boldsymbol{\omega}^{\oplus}}$$

Finally, we compute the next orientation q^{+} by rotation composition

$$q^{+} = \Delta q q^{\circ}.$$

1.5.3 Orientation (aspherical)

Integrating rotation of aspherical particles is considerably more complicated than their position, as their local reference frame is not inertial. Rotation of rigid body in the local frame, where inertia matrix $\mathbf{I}$ is diagonal, is described in the continuous form by Euler's equations ($i \in \{1,2,3\}$ and i, j, k are subsequent indices):

$$\mathbf{T}_i = \mathbf{I}_{ii}\dot{\boldsymbol{\omega}}_i + (\mathbf{I}_{kk} - \mathbf{I}_{jj})\boldsymbol{\omega}_j\boldsymbol{\omega}_k.$$

Due to the presence of the current values of both $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$, they cannot be solved using the standard leapfrog algorithm (that was the case for translational motion and also for the spherical bodies' rotation where this equation reduced to $\mathbf{T} = \mathbf{I}\dot{\boldsymbol{\omega}}$).

The algorithm presented here is described by [Allen1989] (pg. 84–89) and was designed by Fincham for molecular dynamics problems; it is based on extending the leapfrog algorithm by mid-step/on-step estimators of quantities known at on-step/mid-step points in the basic formulation. Although it has received criticism and more precise algorithms are known ([Omelyan1999], [Neto2006], [Johnson2008]), this one is currently implemented in Yade for its relative simplicity.

Each body has its local coordinate system based on the principal axes of inertia for that body. We use $\widetilde{\bullet}$ to denote vectors in local coordinates. The orientation of the local system is given by the current particle's orientation q° as a quaternion; this quaternion can be expressed as the (current) rotation matrix $\boldsymbol{A}$. Therefore, every vector $\mathbf{a}$ is transformed as $\widetilde{\mathbf{a}} = q\mathbf{a}q^{*} = \boldsymbol{A}\mathbf{a}$. Since $\boldsymbol{A}$ is a rotation (orthogonal) matrix, the inverse rotation $\boldsymbol{A}^{-1} = \boldsymbol{A}^{T}$.

For given particle in question, we know

- $\widetilde{\mathbf{I}}^{\circ}$ (constant) inertia matrix; diagonal, since in local, principal coordinates,
- $\mathbf{T}^{\circ}$ external torque,
- q° current orientation (and its equivalent rotation matrix $\boldsymbol{A}$),
- $\boldsymbol{\omega}^{\ominus}$ mid-step angular velocity,
- $\mathbf{L}^{\ominus}$ mid-step angular momentum; this is an auxiliary variable that must be tracked in addition for use in this algorithm. It will be zero in the initial step.

Our goal is to compute new values of the latter three, that is $\mathbf{L}^{\oplus}$, q^{+}, $\boldsymbol{\omega}^{\oplus}$. We first estimate current angular momentum and compute current local angular velocity:

$$\mathbf{L}^{\circ} = \mathbf{L}^{\ominus} + \mathbf{T}^{\circ}\frac{\Delta t}{2}, \qquad \widetilde{\mathbf{L}}^{\circ} = \boldsymbol{A}\mathbf{L}^{\circ},$$
$$\mathbf{L}^{\oplus} = \mathbf{L}^{\ominus} + \mathbf{T}^{\circ}\Delta t, \qquad \widetilde{\mathbf{L}}^{\oplus} = \boldsymbol{A}\mathbf{L}^{\oplus},$$
$$\widetilde{\boldsymbol{\omega}}^{\circ} = \widetilde{\mathbf{I}}^{\circ -1}\widetilde{\mathbf{L}}^{\circ},$$
$$\widetilde{\boldsymbol{\omega}}^{\oplus} = \widetilde{\mathbf{I}}^{\circ -1}\widetilde{\mathbf{L}}^{\oplus}.$$

Then we compute $\dot{q}^\circ$, using q° and $\widetilde{\omega}^\circ$:

$$\begin{pmatrix}\dot{q}^\circ_w\\ \dot{q}^\circ_x\\ \dot{q}^\circ_y\\ \dot{q}^\circ_z\end{pmatrix} = \frac{1}{2}\begin{pmatrix}q^\circ_w & -q^\circ_x & -q^\circ_y & -q^\circ_z\\ q^\circ_x & q^\circ_w & -q^\circ_z & q^\circ_y\\ q^\circ_y & q^\circ_z & q^\circ_w & -q^\circ_x\\ q^\circ_z & -q^\circ_y & q^\circ_x & q^\circ_w\end{pmatrix}\begin{pmatrix}0\\ \widetilde{\omega}^\circ_x\\ \widetilde{\omega}^\circ_y\\ \widetilde{\omega}^\circ_z\end{pmatrix},$$
$$q^\oplus = q^\circ + \dot{q}^\circ\frac{\Delta t}{2}.$$

We evaluate $\dot{q}^\oplus$ from $q^\oplus$ and $\widetilde{\omega}^\oplus$ in the same way as in (**??**) but shifted by $\Delta t/2$ ahead. Then we can finally compute the desired values

$$q^+ = q^\circ + \dot{q}^\oplus\Delta t,$$
$$\boldsymbol{\omega}^\oplus = \mathbf{A}^{-1}\widetilde{\boldsymbol{\omega}}^\oplus$$

1.5.4 Clumps (rigid aggregates)

DEM simulations frequently make use of rigid aggregates of particles to model complex shapes [Price2007] called *clumps*, typically composed of many spheres. Dynamic properties of clumps are computed from the properties of its members: the clump's mass m_c is summed over members, the inertia tensor $\mathbf{I}_c$ with respect to the clump's centroid is computed using the parallel axes theorem; local axes are oriented such that they are principal and inertia tensor is diagonal and clump's orientation is changed to compensate rotation of the local system, as to not change the clump members' positions in global space. Initial positions and orientations of all clump members in local coordinate system are stored.

In Yade (class Clump), clump members behave as stand-alone particles during simulation for purposes of collision detection and contact resolution, except that they have no contacts created among themselves within one clump. It is at the stage of motion integration that they are treated specially. Instead of integrating each of them separately, forces/torques on those particles $\mathbf{F}_i$, $\mathbf{T}_i$ are converted to forces/torques on the clump itself. Let us denote r_i relative position of each particle with regards to clump's centroid, in global orientation. Then summary force and torque on the clump are

$$\mathbf{F}_c = \sum F_i,$$
$$\mathbf{T}_c = \sum r_i \times F_i + T_i.$$

Motion of the clump is then integrated, using aspherical rotation integration. Afterwards, clump members are displaced in global space, to keep their initial positions and orientations in the clump's local coordinate system. In such a way, relative positions of clump members are always the same, resulting in the behavior of a rigid aggregate.

1.5.5 Numerical damping

In simulations of quasi-static phenomena, it it desirable to dissipate kinetic energy of particles. Since most constitutive laws (including Law_ScGeom_FrictPhys_Basic shown above, *sect-formulation-stress-cundall*) do not include velocity-based damping (such as one in [Addetta2001]), it is possible to use artificial numerical damping. The formulation is described in [Pfc3dManual30], although our version is slightly adapted. The basic idea is to decrease forces which increase the particle velocities and vice versa by $(\Delta F)_d$, comparing the current acceleration sense and particle velocity sense. This is done by component, which makes the damping scheme clearly non-physical, as it is not invariant with respect to coordinate system rotation; on the other hand, it is very easy to compute. Cundall proposed the form (we omit particle indices i since it applies to all of them separately):

$$\frac{(\Delta\mathbf{F})_{dw}}{\mathbf{F}_w} = -\lambda_d\,\mathrm{sgn}(\mathbf{F}_w\dot{\mathbf{u}}^\ominus_w), \quad w\in\{x,y,z\}$$

where λ_d is the damping coefficient. This formulation has several advantages [Hentz2003]:

- it acts on forces (accelerations), not constraining uniform motion;
- it is independent of eigenfrequencies of particles, they will be all damped equally;
- it needs only the dimensionless parameter λ_d which does not have to be scaled.

In Yade, we use the adapted form

$$\frac{(\Delta F)_{dw}}{F_w} = -\lambda_d \operatorname{sgn} F_w \underbrace{\left(\dot{u}_w^{\ominus} + \frac{\ddot{u}_w^{\circ}\Delta t}{2}\right)}_{\simeq \dot{u}_w^{\circ}}, \tag{1.9}$$

where we replaced the previous mid-step velocity $\dot{u}^{\ominus}$ by its on-step estimate in parentheses. This is to avoid locked-in forces that appear if the velocity changes its sign due to force application at each step, i.e. when the particle in question oscillates around the position of equilibrium with $2\Delta t$ period.

In Yade, damping (1.9) is implemented in the NewtonIntegrator engine; the damping coefficient λ_d is NewtonIntegrator.damping.

1.5.6 Stability considerations

Critical timestep

In order to ensure stability for the explicit integration sceheme, an upper limit is imposed on Δt:

$$\Delta t_{cr} = \frac{2}{\omega_{max}} \tag{1.10}$$

where ω_{max} is the highest eigenfrequency within the system.

Single mass-spring system

Single 1D mass-spring system with mass m and stiffness K is governed by the equation

$$m\ddot{x} = -Kx$$

where x is displacement from the mean (equilibrium) position. The solution of harmonic oscillation is $x(t) = A\cos(\omega t + \varphi)$ where phase φ and amplitude A are determined by initial conditions. The angular frequency

$$\omega^{(1)} = \sqrt{\frac{K}{m}} \tag{1.11}$$

does not depend on initial conditions. Since there is one single mass, $\omega_{max}^{(1)} = \omega^{(1)}$. Plugging (1.11) into (1.10), we obtain

$$\Delta t_{cr}^{(1)} = 2/\omega_{max}^{(1)} = 2\sqrt{m/K}$$

for a single oscillator.

General mass-spring system

In a general mass-spring system, the highest frequency occurs if two connected masses m_i, m_j are in opposite motion; let us suppose they have equal velocities (which is conservative) and they are connected by a spring with stiffness K_i: displacement Δx_i of m_i will be accompained by $\Delta x_j = -\Delta x_i$ of m_j, giving

$\Delta F_i = -K_i(\Delta x_i - (-\Delta x_i)) = -2K_i\Delta x_i$. That results in apparent stiffness $K_i^{(2)} = 2K_i$, giving maximum angular frequency of the whole system

$$\omega_{max} = \max_i \sqrt{K_i^{(2)}/m_i}.$$

The overall critical timestep is then

$$\Delta t_{cr} = \frac{2}{\omega_{max}} = \min_i 2\sqrt{\frac{m_i}{K_i^{(2)}}} = \min_i 2\sqrt{\frac{m_i}{2K_i}} = \min_i \sqrt{2}\sqrt{\frac{m_i}{K_i}}. \tag{1.12}$$

This equation can be used for all 6 degrees of freedom (DOF) in translation and rotation, by considering generalized mass and stiffness matrices M and K, and replacing fractions $\frac{m_i}{K_i}$ by eigen values of $K.M^{-1}$. The critical timestep is then associated to the eigen mode with highest frequency :

$$\Delta t_{cr} = \min \Delta t_{crk}, \quad k \in \{1, ..., 6\}. \tag{1.13}$$

DEM simulations

In DEM simulations, per-particle stiffness $\mathbf{K}_{ij}$ is determined from the stiffnesses of contacts in which it participates [Chareyre2005]. Suppose each contact has normal stiffness K_{Nk}, shear stiffness $K_{Tk} = \xi K_{Nk}$ and is oriented by normal $\mathbf{n}_k$. A translational stiffness matrix $\mathbf{K}_{ij}$ can be defined as the sum of contributions of all contacts in which it participates (indices k), as

$$\mathbf{K}_{ij} = \sum_k (K_{Nk} - K_{Tk})\mathbf{n}_i\mathbf{n}_j + K_{Tk} = \sum_j K_{Nk}\left((1-\xi)\mathbf{n}_i\mathbf{n}_j + \xi\right) \tag{1.14}$$

with i and $j \in \{x, y, z\}$. Equations (1.13) and (1.14) determine Δt_{cr} in a simulation. A similar approach generalized to all 6 DOFs is implemented by the GlobalStiffnessTimeStepper engine in Yade. The derivation of generalized stiffness including rotational terms is very similar but not developped here, for simplicity. For full reference, see "PFC3D - Theoretical Background".

Note that for computation efficiency reasons, eigenvalues of the stiffness matrices are not computed. They are only approximated assuming than DOF's are uncoupled, and using diagonal terms of $K.M^{-1}$. They give good approximates in typical mechanical systems.

There is one important condition that $\omega_{max} > 0$: if there are no contacts between particles and $\omega_{max} = 0$, we would obtain value $\Delta t_{cr} = \infty$. While formally correct, this value is numerically erroneous: we were silently supposing that stiffness remains constant during each timestep, which is not true if contacts are created as particles collide. In case of no contact, therefore, stiffness must be pre-estimated based on future interactions, as shown in the next section.

Estimation of Δt_{cr} by wave propagation speed

Estimating timestep in absence of interactions is based on the connection between interaction stiffnesses and the particle's properties. Note that in this section, symbols E and ρ refer exceptionally to Young's modulus and density of *particles*, not of macroscopic arrangement.

In Yade, particles have associated Material which defines density ρ (Material.density), and also may define (in ElastMat and derived classes) particle's "Young's modulus" E (ElastMat.young). ρ is used when particle's mass m is initially computed from its ρ, while E is taken in account when creating new interaction between particles, affecting stiffness K_N. Knowing m and K_N, we can estimate (1.14) for each particle; we obviously neglect

- number of interactions per particle N_i; for a "reasonable" radius distribution, however, there is a geometrically imposed upper limit (6 for a packing of spheres with equal radii, for instance);
- the exact relationship the between particles' rigidities E_i, E_j, supposing only that K_N is somehow proportional to them.

By defining E and ρ, particles have continuum-like quantities. Explicit integration schemes for continuum equations impose a critical timestep based on sonic speed $\sqrt{E/\rho}$; the elastic wave must not propagate farther than the minimum distance of integration points l_{min} during one step. Since E, ρ are parameters of the elastic continuum and l_{min} is fixed beforehand, we obtain

$$\Delta t_{cr}^{(c)} = l_{min}\sqrt{\frac{\rho}{E}}.$$

For our purposes, we define E and ρ for each particle separately; l_{min} can be replaced by the sphere's radius R_i; technically, $l_{min} = 2R_i$ could be used, but because of possible interactions of spheres and facets (which have zero thickness), we consider $l_{min} = R_i$ instead. Then

$$\Delta t_{cr}^{(p)} = \min_i R_i\sqrt{\frac{\rho_i}{E_i}}.$$

This algorithm is implemented in the utils.PWaveTimeStep function.

Let us compare this result to (1.12); this necessitates making several simplifying hypotheses:

- all particles are spherical and have the same radius R;
- the sphere's material has the same E and ρ
- the average number of contacts per sphere is N;
- the contacts have sufficiently uniform spatial distribution around each particle;
- the $\xi = K_N/K_T$ ratio is constant for all interactions;
- contact stiffness K_N is computed from E using a formula of the form

$$K_N = E\pi' R', \tag{1.15}$$

  where π' is some constant depending on the algorithm in usefootnote{For example, $\pi' = \pi/2$ in the concrete particle model (Ip2_CpmMat_CpmMat_CpmPhys), while $\pi' = 2$ in the classical DEM model (Ip2_FrictMat_FrictMat_FrictPhys) as implemented in Yade.} and R' is half-distance between spheres in contact, equal to R for the case of interaction radius $R_I = 1$. If $R_I = 1$ (and $R' \equiv R$ by consequence), all interactions will have the same stiffness K_N. In other cases, we will consider K_N as the average stiffness computed from average R' (see below).

As all particles have the same parameters, we drop the i index in the following formulas.

We try to express the average per-particle stiffness from (1.14). It is a sum over all interactions where K_N and ξ are scalars that will not rotate with interaction, while $\mathbf{n}_w$ is w-th component of unit interaction normal $\mathbf{n}$. Since we supposed uniform spatial distribution, we can replace $\mathbf{n}_w^2$ by its average value $\overline{\mathbf{n}}_w^2$. Recognizing components of $\mathbf{n}$ as direction cosines, the average values of $\mathbf{n}_w^2$ is $1/3$. %we find the average value by integrating over all possible orientations, which are uniformly distributed in space:

Moreover, since all directions are equal, we can write the per-body stiffness as $K = \mathbf{K}_w$ for all $w \in \{x, y, z\}$. We obtain

$$K = \sum K_N\left((1-\xi)\frac{1}{3} + \xi\right) = \sum K_N\frac{1-2\xi}{3}$$

and can put constant terms (everything) in front of the summation. $\sum 1$ equals the number of contacts per sphere, i.e. N. Arriving at

$$K = NK_N\frac{1-2\xi}{3},$$

we substitute K into (1.12) using (1.15):

$$\Delta t_{cr} = \sqrt{2}\sqrt{\frac{m}{K}} = \sqrt{2}\sqrt{\frac{\frac{4}{3}\pi R^3 \rho}{NE\pi' R\frac{1-2\xi}{3}}} = \underbrace{R\sqrt{\frac{\rho}{E}}}_{\Delta t_{cr}^{(p)}} 2\sqrt{\frac{\pi/\pi'}{N(1-2\xi)}}.$$

The ratio of timestep $\Delta t_{cr}^{(p)}$ predicted by the p-wave velocity and numerically stable timestep Δt_{cr} is the inverse value of the last (dimensionless) term:

$$\frac{\Delta t_{cr}^{(p)}}{\Delta t_{cr}} = 2\sqrt{\frac{N(1+\xi)}{\pi/\pi'}}.$$

Actual values of this ratio depend on characteristics of packing N, $K_N/K_T = \xi$ ratio and the way of computing contact stiffness from particle rigidity. Let us show it for two models in Yade:

Concrete particle model computes contact stiffness from the equivalent area A_{eq} first (1.6),

$$A_{eq} = \pi R^2 K_N \qquad = \frac{A_{eq} E}{d_0}.$$

d_0 is the initial contact length, which will be, for interaction radius (1.5) $R_I > 1$, in average larger than $2R$. For $R_I = 1.5$ (sect.~ref{sect-calibration-elastic-properties}), we can roughly estimate $\overline{d}_0 = 1.25 \cdot 2R = \frac{5}{2}R$, getting

$$K_N = E\left(\frac{2}{5}\pi\right) R$$

where $\frac{2}{5}\pi = \pi'$ by comparison with (1.15).

Interaction radius $R_I = 1.5$ leads to average $N \approx 12$ interactions per sphere for dense packing of spheres with the same radius R. $\xi = 0.2$ is calibrated (sect.~ref{sect-calibration-elastic-properties}) to match the desired macroscopic Poisson's ratio $\nu = 0.2$.

Finally, we obtain the ratio

$$\frac{\Delta t_{cr}^{(p)}}{\Delta t_{cr}} = 2\sqrt{\frac{12(1-2\cdot 0.2)}{\frac{\pi}{(2/5)\pi}}} = 3.39,$$

showing significant overestimation by the p-wave algorithm.

Non-cohesive dry friction model is the basic model proposed by Cundall explained in ref{sect-formulation-stress-cundall}. Supposing almost-constant sphere radius R and rather dense packing, each sphere will have $N = 6$ interactions on average (that corresponds to maximally dense packing of spheres with a constant radius). If we use the Ip2_FrictMat_FrictMat_FrictPhys class, we have $\pi' = 2$, as $K_N = E2R$; we again use $\xi = 0.2$ (for lack of a more significant value). In this case, we obtain the result

$$\frac{\Delta t_{cr}^{(p)}}{\Delta t_{cr}} = 2\sqrt{\frac{6(1-2\cdot 0.2)}{\pi/2}} = 3.02$$

which again overestimates the numerical critical timestep.

To conclude, p-wave timestep gives estimate proportional to the real Δt_{cr}, but in the cases shown, the value of about $\Delta t = 0.3\Delta t_{cr}^{(p)}$ should be used to guarantee stable simulation.

Non-elastic Δt constraints

Let us note at this place that not only Δt_{cr} assuring numerical stability of motion integration is a constraint. In systems where particles move at relatively high velocities, position change during one timestep can lead to non-elastic irreversible effects such as damage. The Δt needed for reasonable result can be lower Δt_{cr}. We have no rigorously derived rules for such cases.

1.6 Periodic boundary conditions

While most DEM simulations happen in R^3 space, it is frequently useful to avoid boundary effects by using periodic space instead. In order to satisfy periodicity conditions, periodic space is created by repetition of parallelepiped-shaped cell. In Yade, periodic space is implemented in the Cell class. The geometry of the cell in the reference coordinates system is defined by three edges of the parallepiped. The corresponding base vectors are stored in the columns of matrix $\mathbf{H}$ (Cell.hSize).

The initial $\mathbf{H}$ can be explicitely defined as a 3x3 matrix at the begining of the simulation. There are no restricitions on the possible shapes: any parallelepiped is accepted as the initial cell. If the base vectors are axis-aligned, defining only their sizes can be more convenient than defining the full $\mathbf{H}$ matrix; in that case it is enough to define the norms of columns in $\mathbf{H}$ (see Cell.size).

After the definition of the initial cell's geometry, $\mathbf{H}$ should generally not be modified by direct assignment. Instead, its deformation rate will be defined via the velocity gradient Cell.velGrad described below. It is the only variable that let the period deformation be correctly accounted for in constitutive laws and Newton integrator (NewtonIntegrator).

1.6.1 Deformations handling

The deformation of the cell over time is defined via a matrix representing the gradient of an homogeneous velocity field $\nabla \boldsymbol{v}$ (Cell.velGrad). This gradient represents arbitrary combinations of rotations and stretches. It can be imposed externaly or updated by boundary controllers (see PeriTriaxController or Peri3dController) in order to reach target strain values or to maintain some prescribed stress.

The velocity gradient is integrated automatically over time, and the cumulated transformation is reflected in the transformation matrix $\mathbf{F}$ (Cell.trsf) and the current shape of the cell $\mathbf{H}$. The per-step transformation update reads (it is similar for $\mathbf{H}$), with I the identity matrix:

$$\mathbf{F}^{+} = (\mathrm{I} + \nabla \boldsymbol{v} \Delta t)\mathbf{F}^{\circ}.$$

$\mathbf{F}$ can be set back to identity at any point in simulations, in order to define the current state as reference for strains definition in boundary controllers. It will have no effect on $\mathbf{H}$.

Along with the automatic integration of cell transformation, there is an option to homothetically displace all particles so that $\nabla \boldsymbol{v}$ is applied over the whole simulation (enabled via Cell.homoDeform). This avoids all boundary effects coming from change of the velocity gradient.

1.6.2 Collision detection in periodic cell

In usual implementations, particle positions are forced to be inside the cell by wrapping their positions if they get over the boundary (so that they appear on the other side). As we wanted to avoid abrupt changes of position (it would make particle's velocity inconsistent with step displacement change), a different method was chosen.

Approximate collision detection

Pass 1 collision detection (based on sweep and prune algorithm, sect.~ref{sect-sweep-and-prune}) operates on axis-aligned bounding boxes (Aabb) of particles. During the collision detection phase, bounds of

all Aabb's are wrapped inside the cell in the first step. At subsequent runs, every bound remembers by how many cells it was initially shifted from coordinate given by the Aabb and uses this offset repeatedly as it is being updated from Aabb during particle's motion. Bounds are sorted using the periodic insertion sort algorithm (sect.~ref{sect-periodic-insertion-sort}), which tracks periodic cell boundary ‖.

Upon inversion of two Aabb's, their collision along all three axes is checked, wrapping real coordinates inside the cell for that purpose.

This algorithm detects collisions as if all particles were inside the cell but without the need of constructing "ghost particles" (to represent periodic image of a particle which enters the cell from the other side) or changing the particle's positions.

It is required by the implementation (and partly by the algorithm itself) that particles do not span more than half of the current cell size along any axis; the reason is that otherwise two (or more) contacts between both particles could appear, on each side. Since Yade identifies contacts by Body.id of both bodies, they would not be distinguishable.

In presence of shear, the sweep-and-prune collider could not sort bounds independently along three axes: collision along x axis depends on the mutual position of particles on the y axis. Therefore, bounding boxes *are expressed in transformed coordinates* which are perpendicular in the sense of collision detection. This requires some extra computation: Aabb of sphere in transformed coordinates will no longer be cube, but cuboid, as the sphere itself will appear as ellipsoid after transformation. Inversely, the sphere in simulation space will have a parallelepiped bounding "box", which is cuboid around the ellipsoid in transformed axes (the Aabb has axes aligned with transformed cell basis). This is shown in fig. fig-cell-shear-aabb.

The restriction of a single particle not spanning more than half of the transformed axis becomes stringent as Aabb is enlarged due to shear. Considering Aabb of a sphere with radius r in the cell where $x' \equiv x$, $z' \equiv z$, but $\angle(y, y') = \varphi$, the x-span of the Aabb will be multiplied by $1/\cos\varphi$. For the infinite shear $\varphi \to \pi/2$, which can be desirable to simulate, we have $1/\cos\varphi \to \infty$. Fortunately, this limitation can be easily circumvented by realizing the quasi-identity of all periodic cells which, if repeated in space, create the same grid with their corners: the periodic cell can be flipped, keeping all particle interactions intact, as shown in fig. fig-cell-flip. It only necessitates adjusting the Interaction.cellDist of interactions and re-initialization of the collider (`Collider::invalidatePersistentData`). Cell flipping is implemented in the utils.flipCell function.

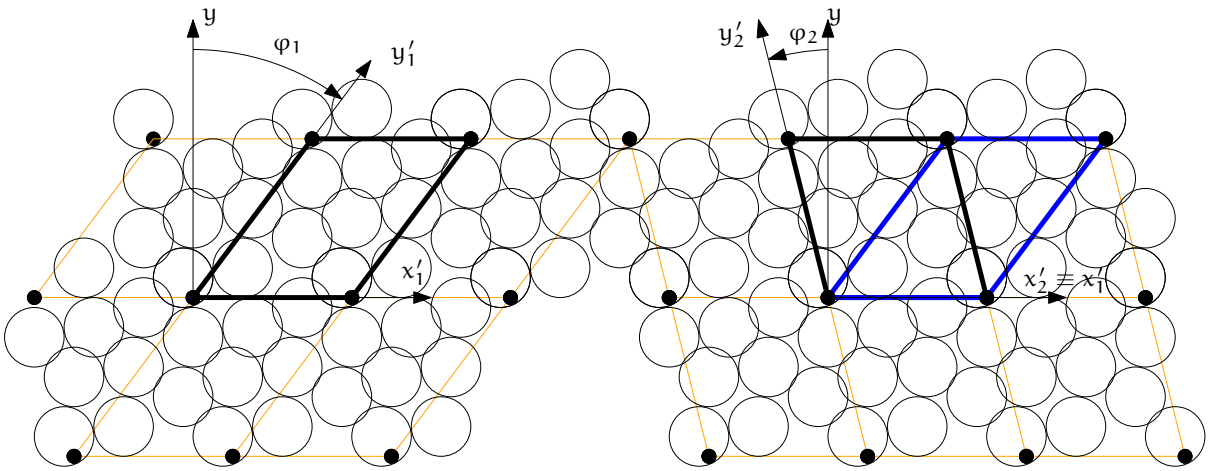

Figure 1.8: Flipping cell (utils.flipCell) to avoid infinite stretch of the bounding boxes' spans with growing φ. Cell flip does not affect interactions from the point of view of the simulation. The periodic arrangement on the left is the same as the one on the right, only the cell is situated differently between identical grid points of repetition; at the same time $|\varphi_2| < |\varphi_1|$ and sphere bounding box's x-span stretched by $1/\cos\varphi$ becomes smaller. Flipping can be repeated, making effective infinite shear possible.

This algorithm is implemented in InsertionSortCollider and is used whenever simulation is periodic (Omega.isPeriodic); individual BoundFunctor's are responsible for computing sheared Aabb's; currently it is implemented for spheres and facets (in Bo1_Sphere_Aabb and Bo1_Facet_Aabb respectively).

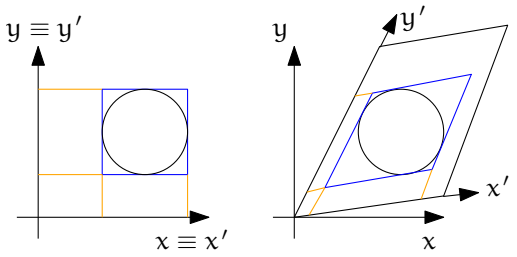

Figure 1.9: Constructing axis-aligned bounding box (Aabb) of a sphere in simulation space coordinates (without periodic cell – left) and transformed cell coordinates (right), where collision detection axes x', y' are not identical with simulation space axes x, y. Bounds' projection to axes is shown by orange lines.

Exact collision detection

When the collider detects approximate contact (on the Aabb level) and the contact does not yet exist, it creates *potential* contact, which is subsequently checked by exact collision algorithms (depending on the combination of Shapes). Since particles can interact over many periodic cells (recall we never change their positions in simulation space), the collider embeds the relative cell coordinate of particles in the interaction itself (Interaction.cellDist) as an *integer* vector $\mathbf{c}$. Multiplying current cell size $\mathbf{Ts}$ by $\mathbf{c}$ component-wise, we obtain particle offset $\Delta\mathbf{x}$ in aperiodic $\mathbb{R}^3$; this value is passed (from InteractionLoop) to the functor computing exact collision (IGeomFunctor), which adds it to the position of the particle Interaction.id2.

By storing the integral offset $\mathbf{c}$, $\Delta\mathbf{x}$ automatically updates as cell parameters change.

Periodic insertion sort algorithm

The extension of sweep and prune algorithm (described in *Sweep and prune*) to periodic boundary conditions is non-trivial. Its cornerstone is a periodic variant of the insertion sort algorithm, which involves keeping track of the "period" of each boundary; e.g. taking period $\langle 0, 10)$, then $8_1 \equiv -2_2 < 2_2$ (subscript indicating period). Doing so efficiently (without shuffling data in memory around as bound wraps from one period to another) requires moving period boundary rather than bounds themselves and making the comparison work transparently at the edge of the container.

This algorithm was also extended to handle non-orthogonal periodic Cell boundaries by working in transformed rather than Cartesian coordinates; this modifies computation of Aabb from Cartesian coordinates in which bodies are positioned (treated in detail in *Approximate collision detection*).

The sort algorithm is tracking Aabb extrema along all axes. At the collider's initialization, each value is assigned an integral period, i.e. its distance from the cell's interior expressed in the cell's dimension along its respective axis, and is wrapped to a value inside the cell. We put the period number in subscript.

Let us give an example of coordinate sequence along x axis (in a real case, the number of elements would be even, as there is maximum and minimum value couple for each particle; this demonstration only shows the sorting algorithm, however.)

$$4_1 \qquad 12_2 \quad \| \quad -1_2 \qquad -2_4 \qquad 5_0$$

with cell x-size $s_x = 10$. The 4_1 value then means that the real coordinate x_i of this extremum is $x_i + 1 \cdot 10 = 4$, i.e. $x_i = -4$. The $\|$ symbol denotes the periodic cell boundary.

Sorting starts from the first element in the cell, i.e. right of $\|$, and inverts elements as in the aperiodic variant. The rules are, however, more complicated due to the presence of the boundary $\|$:

$(\leq)$	stop inverting if neighbors are ordered;
$(\|\bullet)$	current element left of $\|$ is below 0 (lower period boundary); in this case, decrement element's period, decrease its coordinate by s_x and move $\|$ right;
$(\bullet\|)$	current element right of $\|$ is above s_x (upper period boundary); increment element's period, increase its coordinate by s_x and move $\|$ left;
$(\nleftrightarrow)$	inversion across $\|$ must subtract s_x from the left coordinate during comparison. If the elements are not in order, they are swapped, but they must have their periods changed as they traverse $\|$. Apply $(\|\circ)$ if necessary;
$(\|\circ)$	if after $(\nleftrightarrow)$ the element that is now right of $\|$ has $x_i < s_x$, decrease its coordinate by s_x and decrement its period. Do not move $\|$.

In the first step, $(\|\bullet)$ is applied, and inversion with 12_2 happens; then we stop because of $(\leq)$:

$$4_1 \quad 12_2 \quad \| \quad \boxed{-1_2} \quad -2_4 \quad 5_0,$$

$$4_1 \quad 12_2 \quad \underset{\nleq}{\leftarrow} \boxed{9_1} \quad \| \quad -2_4 \quad 5_0,$$

$$4_1 \quad \underset{\leq}{\leftarrow} \boxed{9_1} \quad 12_2 \quad \| \quad -2_4 \quad 5_0.$$

We move to next element $\boxed{-2_4}$; first, we apply $(\|\bullet)$, then invert until $(\leq)$:

$$4_1 \quad 9_1 \quad 12_2 \quad \| \quad \boxed{-2_4} \quad 5_0,$$

$$4_1 \quad 9_1 \quad 12_2 \quad \underset{\nleq}{\leftarrow} \boxed{8_3} \quad \| \quad 5_0,$$

$$4_1 \quad 9_1 \quad \underset{\nleq}{\leftarrow} \boxed{8_3} \quad 12_2 \quad \| \quad 5_0,$$

$$4_1 \quad \underset{\leq}{\leftarrow} \boxed{8_3} \quad 9_1 \quad 12_2 \quad \| \quad 5_0.$$

The next element is $\boxed{5_0}$; we satisfy $(\nleftrightarrow)$, therefore instead of comparing $12_2 > 5_0$, we must do $(12_2 - s_x) = 2_3 \leq 5$; we adjust periods when swapping over $\|$ and apply $(\|\circ)$, turning 12_2 into 2_3; then we keep inverting, until $(\leq)$:

$$4_1 \quad 8_3 \quad 9_1 \quad 12_2 \quad \underset{\nleq}{\leftarrow} \| \quad \boxed{5_0},$$

$$4_1 \quad 8_3 \quad 9_1 \quad \underset{\nleq}{\leftarrow} \boxed{5_{-1}} \quad \| \quad 2_3,$$

$$4_1 \quad 8_3 \quad \underset{\nleq}{\leftarrow} \boxed{5_{-1}} \quad 9_1 \quad \| \quad 2_3,$$

$$4_1 \quad \underset{\leq}{\leftarrow} \boxed{5_{-1}} \quad 8_3 \quad 9_1 \quad \| \quad 2_3.$$

We move (wrapping around) to $\boxed{4_1}$, which is ordered:

$$\boxed{4_1} \quad 5_{-1} \quad 8_3 \quad 9_1 \quad \| \quad 2_3 \qquad (\geq)$$

and so is the last element

$$4_1 \quad \underset{\leq}{\leftarrow} \boxed{5_{-1}} \quad 8_3 \quad 9_1 \quad \| \quad 2_3.$$

1.7 Computational aspects

1.7.1 Cost

The DEM computation using an explicit integration scheme demands a relatively high number of steps during simulation, compared to implicit scehemes. The total computation time Z of simulation spanning T seconds (of simulated time), containing N particles in volume V depends on:

- linearly, the number of steps $i = T/(s_t \Delta t_{cr})$, where s_t is timestep safety factor; Δt_{cr} can be estimated by p-wave velocity using E and ρ (sect.~ref{sect-dt-pwave}) as $\Delta t_{cr}^{(p)} = r\sqrt{\frac{\rho}{E}}$. Therefore

$$i = \frac{T}{s_t r}\sqrt{\frac{E}{\rho}}.$$

- the number of particles N; for fixed value of simulated domain volume V and particle radius r

$$N = p\frac{V}{\frac{4}{3}\pi r^3},$$

  where p is packing porosity, roughly $\frac{1}{2}$ for dense irregular packings of spheres of similar radius.

  The dependency is not strictly linear (which would be the best case), as some algorithms do not scale linearly; a case in point is the sweep and prune collision detection algorithm introduced in **:ref:‘sect-sweep-and-prune‘_**, with scaling roughly $\mathcal{O}(N \log N)$.

  The number of interactions scales with N, as long as packing characteristics are the same.

- the number of computational cores n_{cpu}; in the ideal case, the dependency would be inverse-linear were all algorithms parallelized (in Yade, collision detection is not).

Let us suppose linear scaling. Additionally, let us suppose that the material to be simulated (E, ρ) and the simulation setup (V, T) are given in advance. Finally, dimensionless constants s_t, p and n_{cpu} will have a fixed value. This leaves us with one last degree of freedom, r. We may write

$$Z \propto iN\frac{1}{n_{cpu}} = \frac{T}{s_t r}\sqrt{\frac{E}{\rho}}p\frac{V}{\frac{4}{3}\pi r^3}\frac{1}{n_{cpu}} \propto \frac{1}{r}\frac{1}{r^3} = \frac{1}{r^4}.$$

This (rather trivial) result is essential to realize DEM scaling; if we want to have finer results, refining the “mesh” by halving r, the computation time will grow $2^4 = 16$ times.

For very crude estimates, one can use a known simulation to obtain a machine “constant”

$$\mu = \frac{Z}{Ni}$$

with the meaning of time per particle and per timestep (in the order of 10^{-6} s for current machines). μ will be only useful if simulation characteristics are similar and non-linearities in scaling do not have major influence, i.e. N should be in the same order of magnitude as in the reference case.

1.7.2 Result indeterminism

It is naturally expected that running the same simulation several times will give exactly the same results: although the computation is done with finite precision, round-off errors would be deterministically the same at every run. While this is true for *single-threaded* computation where exact order of all operations is given by the simulation itself, it is not true anymore in *multi-threaded* computation which is described in detail in later sections.

The straight-forward manner of parallel processing in explicit DEM is given by the possibility of treating interactions in arbitrary order. Strain and stress is evaluated for each interaction independently, but forces from interactions have to be summed up. If summation order is also arbitrary (in Yade, forces are accumulated for each thread in the order interactions are processed, then summed together), then the results can be slightly different. For instance

```
(1/10.)+(1/13.)+(1/17.)=0.23574660633484162
(1/17.)+(1/13.)+(1/10.)=0.23574660633484165
```

As forces generated by interactions are assigned to bodies in quasi-random order, summary force F_i on the body can be different between single-threaded and multi-threaded computations, but also between different runs of multi-threaded computation with exactly the same parameters. Exact thread scheduling by the kernel is not predictable since it depends on asynchronous events (hardware interrupts) and other unrelated tasks running on the system; and it is thread scheduling that ultimately determines summation order of force contributions from interactions.

Numerical damping influence

The effect of summation order can be significantly amplified by the usage of a *discontinuous* damping function in NewtonIntegrator given in (1.9) as

$$\frac{(\Delta\mathbf{F})_{dw}}{\mathbf{F}_w} = -\lambda_d \operatorname{sgn}\mathbf{F}_w \left(\dot{\mathbf{u}}_w^{\ominus} + \frac{\ddot{\mathbf{u}}_w^{\circ}\Delta t}{2}\right).$$

If the sgn argument is close to zero then the least significant finite precision artifact can determine whether the equation (relative increment of $\mathbf{F}_w$) is $+\lambda_d$ or $-\lambda_d$. Given commonly used values of $\lambda_d = 0.4$, it means that such artifact propagates from least significant place to the most significant one at once.

Bibliography

[Camborde2000a] F. Camborde, C. Mariotti, F.V. Donzé (2000), **Numerical study of rock and concrete behaviour by discrete element modelling**. *Computers and Geotechnics* (27), pages 225–247.

[Chen2007a] Feng Chen, Eric. C. Drumm, Georges Guiochon (2007), **Prediction/verification of particle motion in one dimension with the discrete-element method**. *International Journal of Geomechanics, ASCE* (7), pages 344–352. DOI 10.1061/(ASCE)1532-3641(2007)7:5(344)

[Dang2010] H. K. Dang, M. A. Meguid (2010), **Algorithm to generate a discrete element specimen with predefined properties**. *International Journal of Geomechanics* (10), pages 85-91. DOI 10.1061/(ASCE)GM.1943-5622.0000028

[Donze1994a] F.V. Donzé, P. Mora, S.A. Magnier (1994), **Numerical simulation of faults and shear zones**. *Geophys. J. Int.* (116), pages 46–52.

[Donze1995a] F.V. Donzé, S.A. Magnier (1995), **Formulation of a three-dimensional numerical model of brittle behavior**. *Geophys. J. Int.* (122), pages 790–802.

[Donze1999a] F.V. Donzé, S.A. Magnier, L. Daudeville, C. Mariotti, L. Davenne (1999), **Study of the behavior of concrete at high strain rate compressions by a discrete element method**. *ASCE J. of Eng. Mech* (125), pages 1154–1163. DOI 10.1016/S0266-352X(00)00013-6

[Donze2004a] F.V. Donzé, P. Bernasconi (2004), **Simulation of the blasting patterns in shaft sinking using a discrete element method**. *Electronic Journal of Geotechnical Engineering* (9), pages 1–44.

[Duriez2010] J. Duriez, F.Darve, F.-V.Donze (2010), **A discrete modeling-based constitutive relation for infilled rock joints**. *International Journal of Rock Mechanics & Mining Sciences*. DOI 10.1016/j.ijrmms.2010.09.008 (in press)

[Harthong2009] Harthong, B., Jerier, J. F., Dorémus, P., Imbault, D., Donzé, F. V. (2009), **Modeling of high-density compaction of granular materials by the discrete element method**. *International Journal of Solids and Structures* (46), pages 3357–3364. DOI 10.1016/j.ijsolstr.2009.05.008

[Hassan2010] A. Hassan, B. Chareyre, F. Darve, J. Meyssonier, F. Flin (2010 (submitted)), **Microtomography-based discrete element modelling of creep in snow**. *Granular Matter*.

[Hentz2004a] S. Hentz, F.V. Donzé, L.Daudeville (2004), **Discrete element modelling of concrete submitted to dynamic loading at high strain rates**. *Computers and Structures* (82), pages 2509–2524. DOI 10.1016/j.compstruc.2004.05.016

[Hentz2004b] S. Hentz, L. Daudeville, F.V. Donzé (2004), **Identification and validation of a discrete element model for concrete**. *ASCE Journal of Engineering Mechanics* (130), pages 709–719. DOI 10.1061/(ASCE)0733-9399(2004)130:6(709)

[Hentz2005a] S. Hentz, F.V. Donzé, L.Daudeville (2005), **Discrete elements modeling of a reinforced concrete structure submitted to a rock impact**. *Italian Geotechnical Journal* (XXXIX), pages 83–94.

[Jerier2009] Jerier, Jean-François, Imbault, Didier, Donzé, Fréederic-Victor, Doremus, Pierre (2009), **A geometric algorithm based on tetrahedral meshes to generate a dense polydisperse sphere packing**. *Granular Matter* (11). DOI 10.1007/s10035-008-0116-0

[Jerier2010] Jerier, Jean-François, Richefeu, Vincent, Imbault, Didier, Donzé, Fréderic-Victor (2010), **Packing spherical discrete elements for large scale simulations**. *Computer Methods in Applied Mechanics and Engineering*. DOI 10.1016/j.cma.2010.01.016

[Jerier2010b] J.-F. Jerier, B. Hathong, V. Richefeu, B. Chareyre, D. Imbault, F.-V. Donze, P. Doremus (2010), **Study of cold powder compaction by using the discrete element method**. *Powder Technology* (In Press). DOI DOI: 10.1016/j.powtec.2010.08.056

[Kozicki2005a] J. Kozicki (2005), **Discrete lattice model used to describe the fracture process of concrete**. *Discrete Element Group for Risk Mitigation Annual Report 1, Grenoble University of Joseph Fourier, France*, pages 95–101. (fulltext)

[Kozicki2006a] J. Kozicki, J. Tejchman (2006), **2d lattice model for fracture in brittle materials**. *Archives of Hydro-Engineering and Environmental Mechanics* (53), pages 71–88. (fulltext)

[Kozicki2007a] J. Kozicki, J. Tejchman (2007), **Effect of aggregate structure on fracture process in concrete using 2d lattice model"**. *Archives of Mechanics* (59), pages 365–384. (fulltext)

[Kozicki2008] J. Kozicki, F.V. Donzé (2008), **A new open-source software developed for numerical simulations using discrete modeling methods**. *Computer Methods in Applied Mechanics and Engineering* (197), pages 4429–4443. DOI 10.1016/j.cma.2008.05.023 (fulltext)

[Kozicki2009] J. Kozicki, F.V. Donzé (2009), **Yade-open dem: an open-source software using a discrete element method to simulate granular material**. *Engineering Computations* (26), pages 786–805. DOI 10.1108/02644400910985170 (fulltext)

[Magnier1998a] S.A. Magnier, F.V. Donzé (1998), **Numerical simulation of impacts using a discrete element method**. *Mech. Cohes.-frict. Mater.* (3), pages 257–276. DOI 10.1002/(SICI)1099-1484(199807)3:3<257::AID-CFM50>3.0.CO;2-Z

[Nicot2007a] Nicot, F., L. Sibille, F.V. Donzé, F. Darve (2007), **From microscopic to macroscopic second-order work in granular assemblies**. *Int. J. Mech. Mater.* (39), pages 664–684. DOI 10.1016/j.mechmat.2006.10.003

[Scholtes2009a] Scholtès, L., Chareyre, B., Nicot, F., Darve, F. (2009), **Micromechanics of granular materials with capillary effects**. *International Journal of Engineering Science* (47), pages 64–75. DOI 10.1016/j.ijengsci.2008.07.002

[Scholtes2009b] Scholtès, L., Hicher, P.-Y., Chareyre, B., Nicot, F., Darve, F. (2009), **On the capillary stress tensor in wet granular materials**. *International Journal for Numerical and Analytical Methods in Geomechanics* (33), pages 1289–1313. DOI 10.1002/nag.767

[Scholtes2009c] Scholtès, L., Chareyre, B., Nicot, F., Darve, F. (2009), **Discrete modelling of capillary mechanisms in multi-phase granular media**. *Computer Modeling in Engineering and Sciences* (52), pages 297–318. DOI ‘ <http://dx.doi.org/>‘_

[Sibille2007a] L. Sibille, F. Nicot, F.V. Donzé, F. Darve (2007), **Material instability in granular assemblies from fundamentally different models**. *International Journal For Numerical and Analytical Methods in Geomechanics* (31), pages 457–481. DOI 10.1002/nag.591

[Sibille2008a] L. Sibille, F.-V. Donzé, F. Nicot, B. Chareyre, F. Darve (2008), **From bifurcation to failure in a granular material: a dem analysis**. *Acta Geotechnica* (3), pages 15–24. DOI 10.1007/s11440-007-0035-y

[Smilauer2006] Václav Šmilauer (2006), **The splendors and miseries of yade design**. *Annual Report of Discrete Element Group for Hazard Mitigation.* (fulltext)

[Catalano2008a] E. Catalano (2008), **Infiltration effects on a partially saturated slope - an application of the discrete element method and its implementation in the open-source software yade**. Master thesis at *UJF-Grenoble.* (fulltext)

[Duriez2009a] J. Duriez (2009), **Stabilité des massifs rocheux : une approche mécanique**. PhD thesis at *Institut polytechnique de Grenoble.* (fulltext)

[Kozicki2007b] J. Kozicki (2007), **Application of discrete models to describe the fracture process in brittle materials**. PhD thesis at *Gdansk University of Technology.* (fulltext)

[Scholtes2009d] Luc Scholtès (2009), **modélisation micromécanique des milieux granulaires partiellement saturés**. PhD thesis at *Institut National Polytechnique de Grenoble.* (fulltext)

[Smilauer2010b] Václav Šmilauer (2010), **Cohesive particle model using the discrete element method on the yade platform**. PhD thesis at *Czech Technical University in Prague, Faculty of Civil Engineering & Université Grenoble I – Joseph Fourier, École doctorale I-MEP2*. (fulltext) (LaTeX sources)

[Smilauer2010c] Václav Šmilauer (2010), **Doctoral thesis statement**. *(PhD thesis summary)*. (fulltext) (LaTeX sources)

[Chareyre2009] Chareyre B., Scholtès L. (2009), **Micro-statics and micro-kinematics of capillary phenomena in dense granular materials**. In *Powders and Grains 2009 (Golden, USA)*.

[Chen2008a] Chen, F., Drumm, E.C., Guiochon, G., Suzuki, K (2008), **Discrete element simulation of 1d upward seepage flow with particle-fluid interaction using coupled open source software**. In *Proceedings of The 12th International Conference of the International Association for Computer Methods and Advances in Geomechanics (IACMAG) Goa, India*.

[Chen2009] Chen, F., Drumm, E.C., Guiochon, G. (2009), **3d dem analysis of graded rock fill sinkhole repair: particle size effects on the probability of stability**. In *Transportation Research Board Conference (Washington DC)*.

[Dang2008a] Dang, H.K., Mohamed, M.A. (2008), **An algorithm to generate a specimen for discrete element simulations with a predefined grain size distribution.**. In *61th Canadian Geotechnical Conference, Edmonton, Alberta*.

[Dang2008b] Dang, H.K., Mohamed, M.A. (2008), **3d simulation of the trap door problem using the discrete element method.**. In *61th Canadian Geotechnical Conference, Edmonton, Alberta*.

[Gillibert2009] Gillibert L., Flin F., Rolland du Roscoat S., Chareyre B., Philip A., Lesaffre B., Meyssonier J. (2009), **Curvature-driven grain segmentation: application to snow images from x-ray microtomography**. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Miami, USA)*.

[Hicher2009] Hicher P.-Y., Scholtès L., Chareyre B., Nicot F., Darve F. (2008), **On the capillary stress tensor in wet granular materials**. In *Inaugural International Conference of the Engineering Mechanics Institute (EM08) - (Minneapolis, USA)*.

[Kozicki2003a] J. Kozicki, J. Tejchman (2003), **Discrete methods to describe the behaviour of quasi-brittle and granular materials**. In *16th Engineering Mechanics Conference, University of Washington, Seattle, CD–ROM*.

[Kozicki2003c] J. Kozicki, J. Tejchman (2003), **Lattice method to describe the behaviour of quasi-brittle materials**. In *CURE Workshop, Effective use of building materials, Sopot*.

[Kozicki2004a] J. Kozicki, J. Tejchman (2004), **Study of fracture process in concrete using a discrete lattice model**. In *CURE Workshop, Simulations in Urban Engineering, Gdansk*.

[Kozicki2005b] J. Kozicki, J. Tejchman (2005), **Simulations of fracture in concrete elements using a discrete lattice model**. In *Proc. Conf. Computer Methods in Mechanics (CMM 2005), Czestochowa, Poland*.

[Kozicki2005c] J. Kozicki, J. Tejchman (2005), **Simulation of the crack propagation in concrete with a discrete lattice model**. In *Proc. Conf. Analytical Models and New Concepts in Concrete and Masonry Structures (AMCM 2005), Gliwice, Poland*.

[Kozicki2006b] J. Kozicki, J. Tejchman (2006), **Modelling of fracture process in brittle materials using a lattice model**. In *Computational Modelling of Concrete Structures, EURO-C (eds.: G. Meschke, R. de Borst, H. Mang and N. Bicanic), Taylor anf Francis*.

[Kozicki2006c] J. Kozicki, J. Tejchman (2006), **Lattice type fracture model for brittle materials**. In *35th Solid Mechanics Conference (SOLMECH 2006), Krakow*.

[Kozicki2007c] J. Kozicki, J. Tejchman (2007), **Simulations of fracture processes in concrete using a 3d lattice model**. In *Int. Conf. on Computational Fracture and Failure of Materials and Structures (CFRAC 2007), Nantes*. (fulltext)

[Kozicki2007d] J. Kozicki, J. Tejchman (2007), **Effect of aggregate density on fracture process in concrete using 2d discrete lattice model**. In *Proc. Conf. Computer Methods in Mechanics (CMM 2007), Lodz-Spala*.

[Kozicki2007e] J. Kozicki, J. Tejchman (2007), **Modelling of a direct shear test in granular bodies with a continuum and a discrete approach**. In *Proc. Conf. Computer Methods in Mechanics (CMM 2007), Lodz-Spala*.

[Kozicki2007f] J. Kozicki, J. Tejchman (2007), **Investigations of size effect in tensile fracture of concrete using a lattice model**. In *Proc. Conf. Modelling of Heterogeneous Materials with Applications in Construction and Biomedical Engineering (MHM 2007), Prague*.

[Scholtes2007a] L. Scholtès, B. Chareyre, F. Nicot, F. Darve (2007), **Micromechanical modelling of unsaturated granular media**. In *Proceedings ECCOMAS-MHM07, Prague*.

[Scholtes2008a] L. Scholtès, B. Chareyre, F. Nicot, F. Darve (2008), **Capillary effects modelling in unsaturated granular materials**. In *8th World Congress on Computational Mechanics - 5th European Congress on Computational Methods in Applied Sciences and Engineering, Venice*.

[Scholtes2008b] L. Scholtès, P.-Y. Hicher, F.Nicot, B. Chareyre, F. Darve (2008), **On the capillary stress tensor in unsaturated granular materials**. In *EM08: Inaugural International Conference of the Engineering Mechanics Institute, Minneapolis*.

[Scholtes2009e] Scholtes L, Chareyre B, Darve F (2009), **Micromechanics of partialy saturated granular material**. In *Int. Conf. on Particle Based Methods, ECCOMAS-Particles*.

[Shiu2007a] W. Shiu, F.V. Donze, L. Daudeville (2007), **Discrete element modelling of missile impacts on a reinforced concrete target**. In *Int. Conf. on Computational Fracture and Failure of Materials and Structures (CFRAC 2007), Nantes*.

[Smilauer2007a] V. Šmilauer (2007), **Discrete and hybrid models: applications to concrete damage**. In *Unpublished*. (fulltext)

[Smilauer2008] Václav Šmilauer (2008), **Commanding c++ with python**. In *ALERT Doctoral school talk*. (fulltext)

[Smilauer2010a] Václav Šmilauer (2010), **Yade: past, present, future**. In *Internal seminary in Laboratoire 3S-R, Grenoble*. (fulltext) (LaTeX sources)

[Stransky2010] Jan Stránský, Milan Jirásek, Václav Šmilauer (2010), **Macroscopic elastic properties of particle models**. In *Proceedings of the International Conference on Modelling and Simulation 2010, Prague*. (fulltext)

[yade:background] V. Šmilauer, B. Chareyre (2010), **Yade dem formulation**. In *Yade Documentation* (V. Šmilauer, ed.), The Yade Project , 1st ed. (http://yade-dem.org/doc/formulation.html)

[yade:doc] V. Šmilauer, E. Catalano, B. Chareyre, S. Dorofeenko, J. Duriez, A. Gladky, J. Kozicki, C. Modenese, L. Scholtès, L. Sibille, J. Stránský, K. Thoeni (2010), **Yade Documentation**. The Yade Project. (http://yade-dem.org/doc/)

[yade:manual] V. Šmilauer, A. Gladky, J. Kozicki, C. Modenese, J. Stránský (2010), **Yade, using and programming**. In *Yade Documentation* (V. Šmilauer, ed.), The Yade Project , 1st ed. (fulltext) (http://yade-dem.org/doc/)

[yade:project] **Yade: open source discrete element method** (http://yade-dem.org)

[yade:reference] V. Šmilauer, E. Catalano, B. Chareyre, S. Dorofeenko, J. Duriez, A. Gladky, J. Kozicki, C. Modenese, L. Scholtès, L. Sibille, J. Stránský, K. Thoeni (2010), **Yade Reference Documentation**. In *Yade Documentation* (V. Šmilauer, ed.), The Yade Project , 1st ed. (http://yade-dem.org/doc/)

[Addetta2001] G.~A. D'Addetta, F. Kun, E. Ramm, H.~J. Herrmann (2001), **From solids to granulates - Discrete element simulations of fracture and fragmentation processes in geomaterials.**. In *Continuous and Discontinuous Modelling of Cohesive-Frictional Materials*. (fulltext)

[Allen1989] M. P. Allen, D. J. Tildesley (1989), **Computer simulation of liquids**. Clarendon Press.

[Alonso2004] F. Alonso-Marroqu? R. Garc?Rojo, H. J. Herrmann (2004), **Micro-mechanical investigation of the granular ratcheting**. In *Cyclic Behaviour of Soils and Liquefaction Phenomena*. (fulltext)

[Bertrand2005] D. Bertrand, F. Nicot, P. Gotteland, S. Lambert (2005), **Modelling a geo-composite cell using discrete analysis**. *Computers and Geotechnics* (32), pages 564–577.

[Bertrand2008] D. Bertrand, F. Nicot, P. Gotteland, S. Lambert (2008), **Discrete element method (dem) numerical modeling of double-twisted hexagonal mesh**. *Canadian Geotechnical Journal* (45), pages 1104–1117.

[Chareyre2002a] B. Chareyre, L. Briancon, P. Villard (2002), **Theoretical versus experimental modeling of the anchorage capacity of geotextiles in trenches.**. *Geosynthet. Int.* (9), pages 97–123.

[Chareyre2002b] B. Chareyre, P. Villard (2002), **Discrete element modeling of curved geosynthetic anchorages with known macro-properties.**. In *Proc., First Int. PFC Symposium, Gelsenkirchen, Germany*.

[Chareyre2003] Bruno Chareyre (2003), **Mod?sation du comportement d'ouvrages composites sol-g?ynth?que par ?ments discrets - application aux tranch? d'ancrage en t? de talus.**. PhD thesis at *Grenoble University*. (fulltext)

[Chareyre2005] Bruno Chareyre, Pascal Villard (2005), **Dynamic spar elements and discrete element methods in two dimensions for the modeling of soil-inclusion problems**. *Journal of Engineering Mechanics* (131), pages 689–698. DOI 10.1061/(ASCE)0733-9399(2005)131:7(689) (fulltext)

[CundallStrack1979] P.A. Cundall, O.D.L. Strack (1979), **A discrete numerical model for granular assemblies**. *Geotechnique* (), pages 47–65. DOI 10.1680/geot.1979.29.1.47

[DeghmReport2006] F. V. Donz?ed.), **Annual report 2006** (2006). *Discrete Element Group for Hazard Mitigation*. Universit?oseph Fourier, Grenoble (fulltext)

[Duriez2010] J. Duriez, F.Darve, F.-V.Donze (2010), **A discrete modeling-based constitutive relation for infilled rock joints**. *International Journal of Rock Mechanics & Mining Sciences*. (in press)

[GarciaRojo2004] R. Garc?Rojo, S. McNamara, H. J. Herrmann (2004), **Discrete element methods for the micro-mechanical investigation of granular ratcheting**. In *Proceedings ECCOMAS 2004*. (fulltext)

[Hentz2003] S?astien Hentz (2003), **Mod?sation d'une structure en b?n arm?oumise ?n choc par la m?ode des el?nts discrets**. PhD thesis at *Universit?renoble 1 – Joseph Fourier*.

[Hubbard1996] Philip M. Hubbard (1996), **Approximating polyhedra with spheres for time-critical collision detection**. *ACM Trans. Graph.* (15), pages 179–210. DOI 10.1145/231731.231732

[Johnson2008] Scott M. Johnson, John R. Williams, Benjamin K. Cook (2008), **Quaternion-based rigid body rotation integration algorithms for use in particle methods**. *International Journal for Numerical Methods in Engineering* (74), pages 1303–1313. DOI 10.1002/nme.2210

[Jung1997] Derek Jung, Kamal K. Gupta (1997), **Octree-based hierarchical distance maps for collision detection**. *Journal of Robotic Systems* (14), pages 789–806. DOI 10.1002/(SICI)1097-4563(199711)14:11<789::AID-ROB3>3.0.CO;2-Q

[Klosowski1998] James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, Karel Zikan (1998), **Efficient collision detection using bounding volume hierarchies of k-dops**. *IEEE Transactions on Visualization and Computer Graphics* (4), pages 21–36. (fulltext)

[Kuhl2001] E. Kuhl, G. A. D'Addetta, M. Leukart, E. Ramm (2001), **Microplane modelling and particle modelling of cohesive-frictional materials**. In *Continuous and Discontinuous Modelling of Cohesive-Frictional Materials*. DOI 10.1007/3-540-44424-6_3 (fulltext)

[Lu1998] Ya Yan Lu (1998), **Computing the logarithm of a symmetric positive definite matrix**. *Appl. Numer. Math* (26), pages 483–496. DOI 10.1016/S0168-9274(97)00103-7 (fulltext)

[Luding2008] Stefan Luding (2008), **Introduction to discrete element methods**. In *European Journal of Environmental and Civil Engineering*.

[McNamara2008] S. McNamara, R. Garc?Rojo, H. J. Herrmann (2008), **Microscopic origin of granular ratcheting**. *Physical Review E* (77). DOI 11.1103/PhysRevE.77.031304

[Munjiza1998] A. Munjiza, K. R. F. Andrews (1998), **Nbs contact detection algorithm for bodies of similar size**. *International Journal for Numerical Methods in Engineering* (43), pages 131–149. DOI 10.1002/(SICI)1097-0207(19980915)43:1<131::AID-NME447>3.0.CO;2-S

[Munjiza2006] A. Munjiza, E. Rougier, N. W. M. John (2006), **Mr linear contact detection algorithm**. *International Journal for Numerical Methods in Engineering* (66), pages 46–71. DOI 10.1002/nme.1538

[Neto2006] Natale Neto, Luca Bellucci (2006), **A new algorithm for rigid body molecular dynamics**. *Chemical Physics* (328), pages 259–268. DOI 10.1016/j.chemphys.2006.07.009

[Omelyan1999] Igor P. Omelyan (1999), **A new leapfrog integrator of rotational motion. the revised angular-momentum approach**. *Molecular Simulation* (22). DOI 10.1080/08927029908022097 (fulltext)

[Pfc3dManual30] ICG (2003), **Pfc3d (particle flow code in 3d) theory and background manual, version 3.0**. Itasca Consulting Group.

[Pournin2001] L. Pournin, Th. M. Liebling, A. Mocellin (2001), **Molecular-dynamics force models for better control of energy dissipation in numerical simulations of dense granular media**. *Phys. Rev. E* (65), pages 011302. DOI 10.1103/PhysRevE.65.011302

[Price2007] Mathew Price, Vasile Murariu, Garry Morrison (2007), **Sphere clump generation and trajectory comparison for real particles**. In *Proceedings of Discrete Element Modelling 2007*. (fulltext)

[Satake1982] M. Satake (1982), **Fabric tensor in granular materials.**. In *Proc., IUTAM Symp. on Deformation and Failure of Granular materials, Delft, The Netherlands*.

[Thornton1991] Colin Thornton, K. K. Yin (1991), **Impact of elastic spheres with and without adhesion**. *Powder technology* (65), pages 153–166. DOI 10.1016/0032-5910(91)80178-L

[Thornton2000] Colin Thornton (2000), **Numerical simulations of deviatoric shear deformation of granular media**. *G?echnique* (50), pages 43–53. DOI 10.1680/geot.2000.50.1.43

[Verlet1967] Loup Verlet (1967), **Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules**. *Phys. Rev.* (159), pages 98. DOI 10.1103/PhysRev.159.98

[Villard2004a] P. Villard, B. Chareyre (2004), **Design methods for geosynthetic anchor trenches on the basis of true scale experiments and discrete element modelling**. *Canadian Geotechnical Journal* (41), pages 1193–1205.

[Wang2009] Yucang Wang (2009), **A new algorithm to model the dynamics of 3-d bonded rigid bodies with rotations**. *Acta Geotechnica* (4), pages 117–127. DOI 10.1007/s11440-008-0072-1 (fulltext)

[cgal] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, Mariette Yvinec (2002), **Triangulations in cgal**. *Computational Geometry: Theory and Applications* (22), pages 5–19.