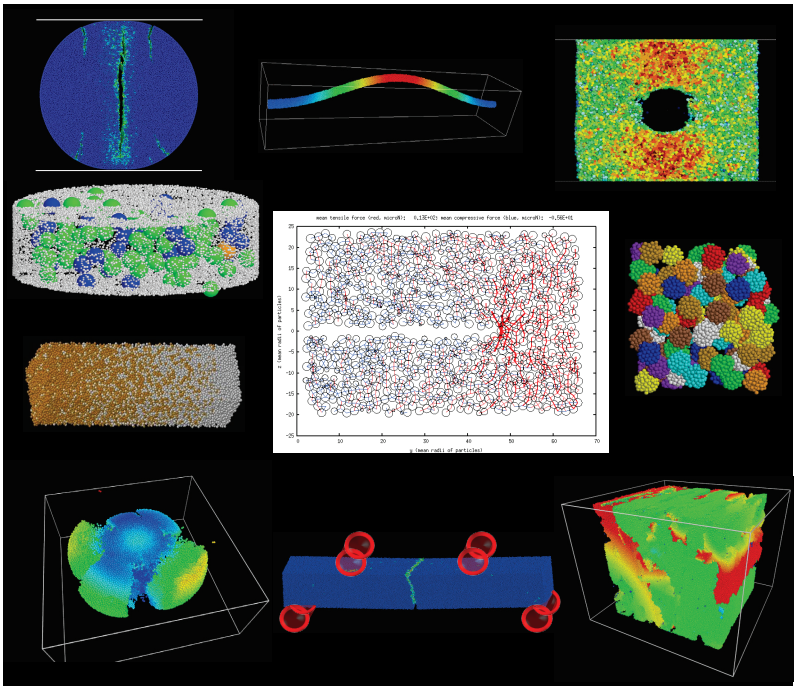
User guide for dp3D

September 2021



christophe.martin@grenoble-inp.fr

Contents

1 Introduction

1.1 Purpose and history of dp3D

dp3D is a simulation code that uses the Discrete Element Method (DEM) pioneered by Cundall and Strack in 1979 [1]. dp3D stands for "**d**iscrete **p**owder **3D**". The code is mainly intended to model materials that are used for engineering applications and which exhibit particulate features such as powders, or which behavior will trigger some discrete features (fracture in particular).

The code was initiated in 2000, during a one year sabbatical leave at the University of Kyoto with late Professor Shima. The very initial code development owns very much to the collaboration with Professor Shima. dp3D is now written with Fortran90[1]. The code has evolved very much in the last years thanks to the input of Prof. David Jauffrès, PhD students and post-docs working at SIMaP laboratory (Univ. Grenoble Alpes).

The main idea behind a DEM code is to model explicitly each particle via its interaction with its neighbors. In DEM, interactions are treated as dynamic processes, where the equilibrium position of particles is sought for at each time step. Interactions occur at contact points and are modeled as contact forces and moments. Particles are modeled in dp3D as spheres and may indent each other (soft contact approach), hence generating contact forces. The use of an explicit numerical scheme makes it possible to use non-linear contact laws and history-dependent laws that are often necessary to describe engineering materials. Examples of applications from dp3D simulations may be found at : dp3D animations[2].

1.2 Main features of dp3D

Two types of entities are physically represented in the dp3D code:

- **Particles** : spherical particles are defined in a coordinate file by their three coordinates and their radius. Particles have a finite mass and their position is dictated at each time step by Newton's second law ($\sum F = m\ddot{x}$).
- **Objects**: objects may interact with particles. An object may be a **plane**, a **cylinder** or a **sphere**. Contrarily to particles, the motion of objects is not dictated by Newton's second law but is solely dictated by the imposed strain conditions given by the user. In other word objects have an infinite mass.

The maximum number of particles is not limited per se in dp3D. Still, limits in available memory on the machine and reasonable CPU time should be taken into account when choosing particle number. Typically, less than 100 000 particles are most often reasonable and will lead to simulation time of the order of hours for small strain. Problems with several millions of particles have been successfully attempted with dp3D (running the parallel version) but they will test your patience (typically a week long calculation or more).

Apart from these particles and objects, the main ingredient of the code is the contact law that describes the mechanical interaction between two particles or between a particle and an object. The contact law defines the physical problem which is tackled in the simulation. The contact laws that are available in the code are:

- Elasticity with adhesion and decohesion following the JKR model [2] or the DMT model [3] (subroutines `elast_DMT.f90` and `elast_JKR.f90`) ;
- Plasticity with strain hardening [4] (subroutine `load_unload_plast.f90`);
- Elasticity, unloading and decohesion of a contact with a plastic history [5] (subroutine `load_unload_plast.f90`);
- Elasticity [6] and fracture of a bonded contact (subroutine `load_elast.f90`) with the possibility of taking into account the interactions due to other contacts [7];
- Viscoplasticity [4] (subroutine `load_unload_viscoplast.f90`);
- Sintering with grain-boundary diffusion limited mechanism [8, 9] (subroutine `load_sinter.f90`) or surface diffusion associated with grain growth (subroutine `coarsen_sinter.f90` [10];

[1] The code is protected by a copyright (IDDN.FR.001.150008.000.S.P.2008.000.31235)
[2] https://simap.grenoble-inp.fr/fr/equipes/animations-discrete-element-method

Several tangential contact laws are available. They may induce shear at the contact (intended for plastic type laws) or follow a simplified Hertz-Mindlin type model (intended for elastic type laws) with a simple Coulomb friction law. Viscous tangential contact law is also available when sintering is used. For bonded contacts, resisting moments are transferred. All these laws are described in the subroutines `tangent_*.f90` in the source code.

More details concerning the main contact laws used in `dp3D` may be found in the following references:

- Elasticity: [11] ;
- Plasticity: [12];
- Bonded contacts [13];
- Bonded contacts with interactions [7, 13, 14];
- Sintering : [10, 15, 16].

Once particles, objects, contacts laws (and the material properties which define them) are given, the `dp3D` calculation starts. The problem is discretized in time steps. The typical outline of a time step in the code is as follows:

- Displacement of boundaries to follow the macroscopic imposed conditions and/or displacement of the objects (if any);
- Optional affine displacement of particles to follow the macroscopic imposed conditions (as if particles followed exactly an homogeneous imposed strain);
- Contact detection from a potential contact list and calculation of the contact geometric parameters;
- Calculation of contact forces and moments from the indentation depth and/or relative velocity, relative rotations;
- Calculation of the total force and total moment applied on each particle;
- Calculation of the accelerations due to applied force and moments for each particle;
- Calculation of the new position and rotation of particles from Newton's second law with a velocity-Verlet algorithm.

Note that during a time step, velocities and accelerations are assumed constant.

Boundary conditions may be imposed by **objects** or by **periodic boundary conditions**. Under periodic boundary conditions (see Fig. 1), a particle that protrudes outside the periodic cell through a given face interacts with the particles of the opposite face. Similarly, when the centre of a particle lies outside the periodic cell, the particle is translated to the opposite face of the cell by a distance equal to the length between the two opposite faces. **Free surface conditions** may also be used.

1.3 Typical calculation with `dp3D`

In order to run a `dp3D` calculation, two files are needed. The first one is the *coordinate file*. It describes the particle coordinates and their radii, together with the objects that can be introduced in the simulation box. A list of bonds between particles i and j in the form ij $(i < j)$ may also be found at the end of this file. A detailed description of the *coordinate file* format is given in section 4. The coordinate file is not necessarily in the directory where `dp3D` is run. [1]

The second file needed to run `dp3D` describes the simulation conditions, it is named `input_dp3D`. This file must be in the directory where `dp3D` is run. This file is discussed in the application examples (section 2) and is detailed line by line in section 6.

[1] A *coordinate file* may be accompanied by a `file_histc` (see section 5.10) to communicate the contact history of a preceding calculation.. It may also be accompanied by a `file_clumps` (see section 5.11) when the `clumps` option is used.

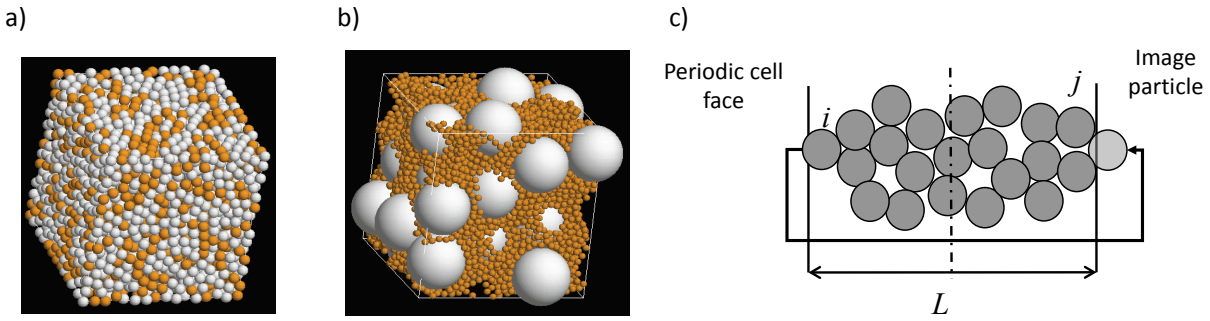


Figure 1: a) Rigid walls condition. b) Periodic conditions. c) Particle i that protrudes outside the periodic cell through the left face interacts with particle j (and others) of the opposite face as if an image particle were created on the right face.

To run the sequential version of `dp3D`, simply type: `dp3D` at the prompt. To run the parallel version, type: `dp3D -openMP`n
where n is the number of threads [1]. For the time being 4 to 8 threads may be considered as an optimum (see section 7 for a brief discussion on this topic).

A typical calculation procedure with `dp3D` is as follows:

1. Generation of a first coordinate file as a gas of particles with no initial contact. In this stage, you may choose the size (and size distribution) of particles, some specific features (such as aggregates or agglomerates of particles bonded together). This initial coordinate file is generated with the command `cdp3D -gas` using a `input_gas` file to describe the initial packing (sections 2.1, 2.2 and 3.4).

2. Densification of the gas of particles to obtain a jammed or nearly jammed assembly. At this stage interactions between particles are only elastic (section 2.2).

3. Compaction and/or sintering of the packing (sections 2.3, 2.16, 2.17, 2.19).

4. Testing of the packing to obtain its mechanical properties (elasticity, fracture, ...) (sections 2.9, 2.10, 2.11, 2.12).

At any stage during the calculation, it is possible to store the packing as a *coordinate file* and to reuse it for a different purpose. Hence at the end of the densification of the gas of particles, it is possible to reuse the *coordinate file* for various tasks.

[1] The command
`nohup dp3D -openMP4 > screen &`
is advised for long calculations that will continue running in the background after you log out.

1.4 Installation of dp3D

Untar the compressed file `dp3D.tar`, and choose the directories where you want to install the executables and the `dp3D_examples` where application examples are available. Some visualization tools come with the dp3D package (`rasmol`) to view the packings. You may prefer the more efficient tool `ovito` (http://www.ovito.org/), [17]. In that case, you need to install `ovito` yourself. If a stack memory problem arises at run-time, you may need to invoke `ulimit -s unlimited` to solve the issue.

If the package comes with source files, in the `src` directory, run the `makedp3D` shell file. First, it will ask if you want to use the default directory `/usr/local/dp3D/` as standard location for `dp3D` binaries. It will ask the fortran compiler to use [1], if you want to clean the object directory (recommended for a first time installation), and if you want to compile a parallel version of the code (via openMP).

Once the compilation has run correctly, the binary file will be, by default in `/usr/local/dp3D/` directory or in the directory that you have provided manually. This directory contains also the `dp3D` tools.

Additionally, a `dp3D_examples` directory is available. It is a set of subdirectories with example problems. They provide a good starting point for your simulations and are commented in detail in section 2. These examples come with an initial coordinate file and a `input_dp3D` file to set simulation conditions.

If you intend to develop the source code of `dp3D`, you will need to obtain the test_before_commit directory that stores benchmark tests. The source code is managed through the cvs tool. The command `dp3D -v` gives the executable version of `dp3D`.

1.5 Organization of this guide

In the present user's guide, the different stages will be presented with examples (section 2). These examples should help to set up your simulation problems. These examples are stored in the directory `dp3D_examples`, which comes with the installation package. In some of these exemples, we suggest short Exercise. We also introduce the main equations that are used in the contact laws in these exemples.

Section 3 presents other tools that come with `dp3D` and that are useful for preprocessing or postprocessing[2].

Section 4 goes into more details to describe the structure of the coordinate file that is needed to run `dp3D`.

Section 5 gives an overview of the output files generated by `dp3D` during the calculation.

Section 6 lists the key words available in the `input_dp3D` file with a short explanation.

Finally, Section 7 discusses briefly the use of dp3D on linux, especially with parallel options.

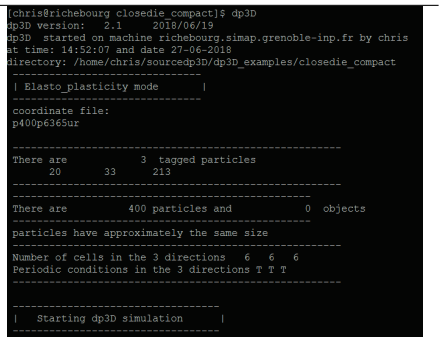
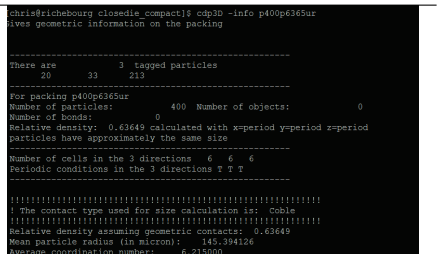
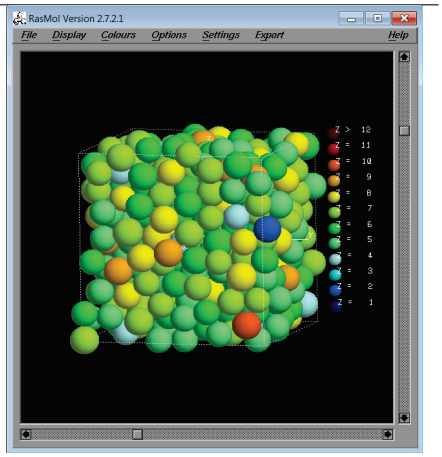
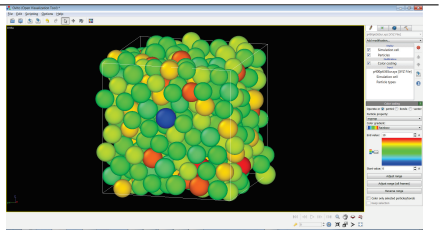
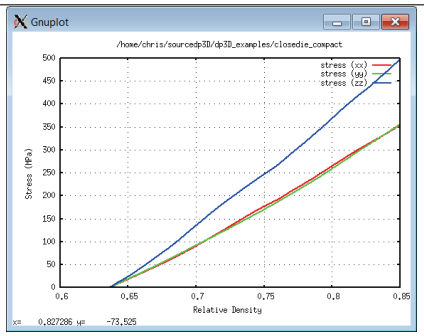
1.6 Citing dp3D

When publishing results obtained with the help of `dp3D`, please cite the paper that uses the corresponding contact law. Each example in the next section comes with a description of the contact law used with the reference given. In doubt, you may contact: Christophe.Martin@grenoble-inp.fr.

[1] only ifort and gfortran have been tested so far.

[2] For example, it is possible to view (or modify) the packing with the viewing tools `vdp3D` and `ovdp3D` or to generate plots from the files generated from the simulation using `ddp3D` or to modify the coordinate file using `cdp3D`.

1.7 Summary of the main commands in dp3D environment

command	example	description	illustration
dp3D	-openMP4	dp3D calculation	
cdp3D	-info coordfile	modifies or gives info on coordinate files	
vdp3D	-z coordfile	rasmol visualization of a coordinate file	
ovdp3D	-z coordfile	ovito visualization of a coordinate file	
ddp3D	-dens -sig	display graphically results from calculations	

2 dp3D by example

In this section, we comment the simulation examples that are given in the directory `dp3D_examples` which comes with the installation package. Each example comes with a `input_dp3D` file and with an initial coordinate file. Each `input_dp3D` is reproduced in the guide, following the description of the example.

`dp3D_examples` is a good starting point for initiating a calculation. You may want to copy the `input_dp3D` file that resembles the most your particular problem and make changes on this file as necessary. The important features for each example are highlighted. The examples that are described below use initial coordinate files with a small number of particles to ensure speedy calculation. You should consider using a larger number of particles.

Along the presentation of these examples, output files generated by `dp3D` will be described shortly. Also, we illustrate some of the most useful postprocessing tools that come with `dp3D`: `ddp3D`, `cdp3D` and `vdp3D`.

Note that the very initial stage for generating a gas of particles is not managed per se by `dp3D` but by `cdp3D -gas` command which use is described in section 2.1.

2.1 Generating a gas of particles: `cdp3D -gas`

- directory : `dp3D_examples/generate_gas_particles`

The command `cdp3D -gas`, together with the file `input_gas` (see example below) allows the generation of a gas of non-bonded particles or a gas of bonded particles (`motifs`) inside a simple box (parallelepipedic or cylindrical). Following is a brief description of the `input_gas` file entries for creating a gas of particles while section 3.4 will describe the creation of a gas of bonded clusters (`motifs`)[1].

An example of a typical file needed to generate a gas of particles is shown on the next page. The file must be named `input_gas` and must be in the directory where the command `cdp3D -gas` is run.

The `input_gas` file is organized very much like the `input_dp3D` file that will be used for `dp3D` calculations. In this first exemple, we limit ourselves to the packing of `particles` (no `motifs`). The box size ratios can be defined to obtain a rectangular simulation box. Note that you can create a z axis cylinder by stating `cyl_z` in the boundary conditions. Several types of size distribution are available (+−, normal and log normal). The deviation for the first two distributions is the standard deviation normalized by the mean particle size. For the log normal distribution, use the mean size and the standard deviation. The particles themselves can be stored in several classes with different sizes, size distributions, names and materials. The number of particles for a given class is defined in the keyword `number`. In this exemple, there are three classes with different sizes. The `seed` keyword allows different packings to be generated (but with the same macroscopic properties) by starting with a different random location for particles.

It is important to understand the simple scheme used in the gas generation. The particles defined via the `input_dp3D` file are randomly placed in the simulation box. If the particle has a contact with an already existing particle then another set of random coordinates is tried until the particle can be placed in the box[2]. It is clear that with such a scheme, large packing densities can not be attained (typically no more than 0.35 for a monomodal packing) and that placing first small particles and then large particles will be very difficult.

In the present exemple, the large particles are defined first (in class 1) and are located first in the simulation box. The small ones easily find a location in between the large ones and this is why a somewhat large packing density can be attained (0.40) .

[1] Note also that before generating a new whole coordinate file (new microstructure), it is worth considering that there exists a library of coordinate files that have been created by `dp3D` users in the `dp3D_library` directory (see section 3.2.)

[2] Except if the keyword `relative_overlap` has been set to a non-zero value. In that case, overlap are allowed between particles (`relative_overlap`>0.), or particles have a minimum distance between them (`relative_overlap`<0.).

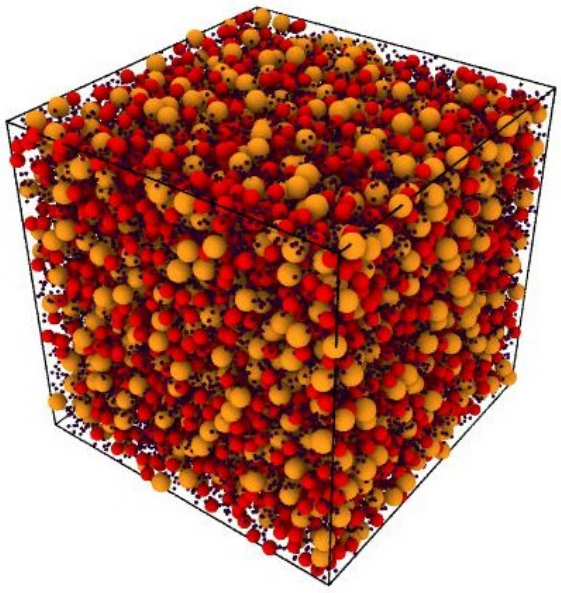


Figure 2: The gas of particles generated with the `input_gas` file.

Exercise:

- *Replace the final expected density* `packing_density=0.30` *by* `packing_density=0.80`. *Why is* `cdp3D -gas`
  unable to generate this packing ?
- *Replace the particle size of class 1* (1000 µm) *by 200 and the particle size of class 3* (200 µm) *by 1000. What is the outcome on the* `cdp3D -gas` *command ? Why is the generation much slower?*
- *Replace the periodic boundary conditions from* `1x1y1z` *to* `0x0y0z`. *Can you see the difference on the simulation box limits (use* `vdp3D -i file_init` *and see Fig. 1 for an explanation of periodic conditions).*
- *Modify the simulation box size ratios to have a box which is 3 times larger in the z direction than in the other. Use the command* `cdp3D -info file_init` *to check that you obtained the desired shape.*

```
##########################################################################
#                general conditions                      #
##########################################################################
# coordinate file:
file_init

_____________________________________________________________
# packing density to reach:
# packing_density= relative_overlap=
packing_density=0.30

_____________________________________________________________
# boundary conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z cyl_x cyl_y
# cyl_z
1x1y1z

_____________________________________________________________
# simulation box size ratios:
# x= y= z=
x=5.00
y=5.00
z=5.00

_____________________________________________________________
# size distribution:
# plus_minus normal lognormal
plus_minus

_____________________________________________________________
# options:
# none initial_file= attach=(mat_n-mat_nn)
none

_____________________________________________________________
##########################################################################
#                classes                                 #
##########################################################################
# prop. of class (particle sizes in µm)          1:
# number= mat= name= particle_size= deviation= Temp= motif= max_angle=
number=2000
mat=1
name=cu
particle_size=1000.00
deviation=0.05

_____________________________________________________________
# prop. of class (particle sizes in µm)          2:
# number= mat= name= particle_size= deviation= Temp= motif= max_angle=
number=4000
mat=1
name=ar
particle_size=700.00
deviation=0.05

_____________________________________________________________
# prop. of class (particle sizes in µm)          3:
# number= mat= name= particle_size= deviation= Temp= motif= max_angle=
number=30000
mat=2
name=zr
particle_size=200.00
deviation=0.05

_____________________________________________________________
##########################################################################
#                      numerics                          #
##########################################################################
# random seed:
# random_seed=
random_seed=-2

_____________________________________________________________
```

2.2 From a gas of particles to a packing of particles

- directory : `gaz_to_pack`; coordinate file: `lognorm_400init`

This example consists in densifying a gas of particles, which has been generated with `cdp3D -gas` (see section 2.1). The initial coordinate packing (`lognorm_400init`) has no initial contact, it has a log normal size distribution (figure 3a). Because of the log normal size distribution, the initial relative density is already quite high ($D_i = 0.5$). The evolution from a gas to a dense packing is a CPU time consuming process if truly jammed packing is sought for.

To enforce `dp3D` calculation to look for fine equilibrium, you must impose the mode key word '`jamming`'. In this mode, only elastic interactions between particle are considered. Boundary conditions are fully periodic (set to `1x1y1z` in the loading conditions). The densification is conducted at a given macroscopic pressure, which is small as compared to the particle elastic modulus. This pressure is given by the command `pressure` in the loading conditions (here set to 0.02MPa). Also, note that some adhesion (with the DMT model) and friction (with a simplified Hertz-Mindlin model) has been given.

Since there is no way to know in advance the jammed packing density for a given size distribution. The `simulation termination` condition is given here by the volumic strain rate at which we consider that particle rearrangement becomes negligible. This condition is set by: `epsvdot`$\leq 1.E^{-09}$ (in sec^{-1}).

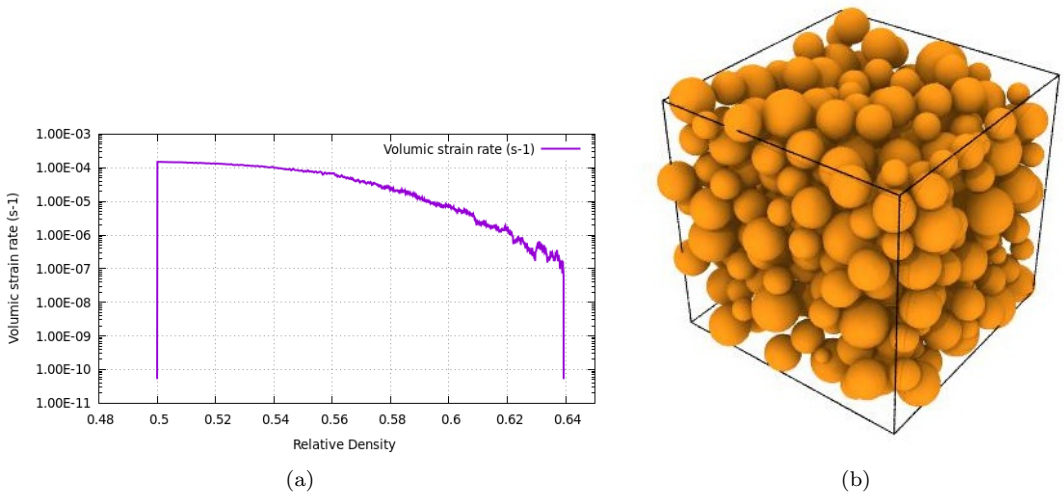
Fig. 3a, which gives the evolution of the volumic strain rate, has been generated using the output file `tstress` together with the `ddp3D` tool[1]. When the packing approaches the "jamming threshold", the volumic strain rate decreases drastically. At this point, the packing cannot densify further solely by particle rearrangement. An alternative for the condition to stop the simulation is to give a target relative density. This can be defined by setting `density`≥ 0.6, for example, instead of `epsvdot`$\geq 1.E^{-09}$ in `input_dp3D`. In that case, the packing may not be necessarily jammed. In most cases, such a packing is sufficiently dense to represent realistically a green (before sintering) powder. If you set too large a value for the target relative density, (for example larger than the RCP 0.635 value for a monomodal random packing), the calculation will not stop automatically. Note that is a test with a set pressure (`pressure=0.2E+05`)[2].

The packing generated at this stage may be used for further compaction or sintering simulations. You may choose any of the output coordinate files _coordxxxx that are generated by `dp3D` for further compaction[3].

[1] `ddp3D -dens -epsvdot`

[2] The controller is a simple Proportional controller by default. You may invoke the `PID_controller` keyword in the general keywords for a complete Proportional Integrator Derivator controller.

[3] You may get a lot of information on coordinate files with the command `cdp3D -info _coord`*iiii* where *iiii* is the number of the `_coord` file.



(a) (b)

Figure 3: (a) Evolution of the macroscopic volumic strain-rate during densification. (b) Initial gas type coordinate file.

Exercise: (it is wise to create a whole new directory for each modification)

- *Replace the* `frict(1,1)=0.` *and the* `adhes(1,1)=0.` *by 0.3 for* `frict` *and 0.2 J.m*$^{-2}$ *for adhesion. Rerun the test. What difference does these changes make on the final density attained during jamming? Use the command*
  `ddp3D -dens -epsvdot`
  to plot the volumic strain rate evolution with density.

- *Keeping friction and adhesion to zero, replace the simulation termination condition by* `density>=0.6.` *Compare the average coordination number of the last coordinate file generated under these conditions with the one generated with* `frict(1,1)=0.30` *and* `adhes(1,1)=2.` *Use the command* `ddp3D -dens -z`.

Contact laws used in this example:

In the *jamming* mode, only elastic interactions are used between particles (no plasticity). However, adhesion or friction can be introduced. Lets consider first the simplest case with neither adhesion nor friction. The normal force acting between two particles of radii R_1 and R_2 with elastic properties (E_1, ν_1) and (E_2, ν_2), respectively, is given by the Hertzian law:

$$N^{Hertz} = \frac{4}{3} E^* R^{*1/2} \delta_n^{3/2} \tag{1}$$

where $E^* = \left(\frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}\right)^{-1}$ and $R^* = \left(\frac{1}{R_1} + \frac{1}{R_2}\right)^{-1}$ and δ_n is the normal indentation (Fig. 4). When adhesion is included the adhesive model must be defined in `input_dp3D` (`adhesion model`). It may be the DMT or the JKR models [3, 18, 19], and a tensile term is added to Eq. (1):

$$\begin{aligned} N^{\mathrm{JKR}} &= \frac{4}{3} E^* R^{*1/2} \delta_n^{3/2} - 2\sqrt{2\pi w E^* a^3}, \\ N^{\mathrm{DMT}} &= \frac{4}{3} E^* R^{*1/2} \delta_n^{3/2} - 2\pi w R^*, \end{aligned} \tag{2}$$

with w the work of adhesion and a the contact radius, which for Hertzian contact writes:

$$a^2 = R^* a \tag{3}$$

The tangential force model is of the Hertz-Mindlin type in the sticking mode while the norm of the tangential force is limited during sliding by Coulomb friction (friction coefficient μ):

$$\begin{aligned} T &= -8G^* a \delta_t \qquad \texttt{if} \quad 8G^* a\delta_t < \mu N^{Hertz}, \\ T &= -\frac{\delta_t}{|\delta_t|} \mu N^{Hertz} \quad \texttt{if} \quad 8G^* a\delta_t \geq \mu N^{Hertz}, \end{aligned} \tag{4}$$

where G^* is the effective shear modulus and δ_t the accumulated tangential displacement.

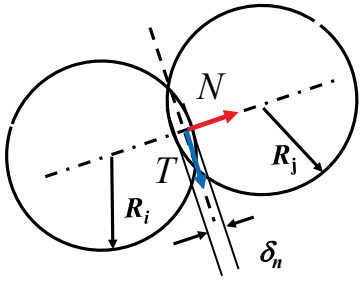


Figure 4: Contact between two indented particles.

```
################################################################################
#                    simulation conditions                                     #
################################################################################
# coordinate file:
lognorm_400init
--------------------------------------------------------------------------------
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
jamming
--------------------------------------------------------------------------------
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity thermal
none
--------------------------------------------------------------------------------
################################################################################
#                              models                                          #
################################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity linear_elasticity
none
--------------------------------------------------------------------------------
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
--------------------------------------------------------------------------------
# adhesion model:
# DMT JKR
DMT
--------------------------------------------------------------------------------
################################################################################
#                             outputs                                          #
################################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
epsvdot<=0.1000E-09
--------------------------------------------------------------------------------
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep= none end
density=0.01
--------------------------------------------------------------------------------
# writing output files:
# density= epsilon= pressure= aoR= time= timestep= none end
density=0.1000E-03
--------------------------------------------------------------------------------
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
--------------------------------------------------------------------------------
################################################################################
#                             loadings                                         #
################################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
--------------------------------------------------------------------------------
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=-0.1000E-03
epsydot=-0.1000E-03
epszdot=-0.1000E-03
pressure=0.2E+05
--------------------------------------------------------------------------------
################################################################################
#                          materials (from 0,1,2, ... to 9)                    #
################################################################################
# elastic parameters (Pa for stress):
# E(0)= poisson(0)= E(1)= poisson(1)= delta_c(0)= fact_mult(0)=
E(1)=0.2000E+11
poisson(1)=0.3000E+00
--------------------------------------------------------------------------------
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.3000E+09
Mstrain=0.0000E+00
Nvisco=0.0000E+00
--------------------------------------------------------------------------------
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.
--------------------------------------------------------------------------------
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.
--------------------------------------------------------------------------------
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.7890E-02
--------------------------------------------------------------------------------
################################################################################
#                              numerics                                        #
################################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
--------------------------------------------------------------------------------
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
--------------------------------------------------------------------------------
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
safe_dt=0.5000E-01
upscale(1)=0.1000E+01
damping=0.5000E-01
--------------------------------------------------------------------------------
```

2.3 Close-die compaction of a homogeneous packing of plastic particles

- directory : `closedie_compact`; coordinate file: `p400p6365ur`

Starting from a coordinate file that has been generated following the procedure exemplified above (section 2.2), this example shows how to compact uniaxially this packing to 0.85 relative density. Note that here, in contrast with the last example, the mode key word is set to 'elasto_plasticity'. This is to ensure that plasticity is allowed in the simulation. The strain rate is defined by `epszdot=-0.1000E-05` to mmimic close-die compaction[1]. By default in `dp3D`, negative strain values relate to compression. The strain-rate must be set to a sufficiently small value to ensure quasi-static conditions[2]. A larger value would result in a smaller CPU time but may not result in quasi-static conditions.

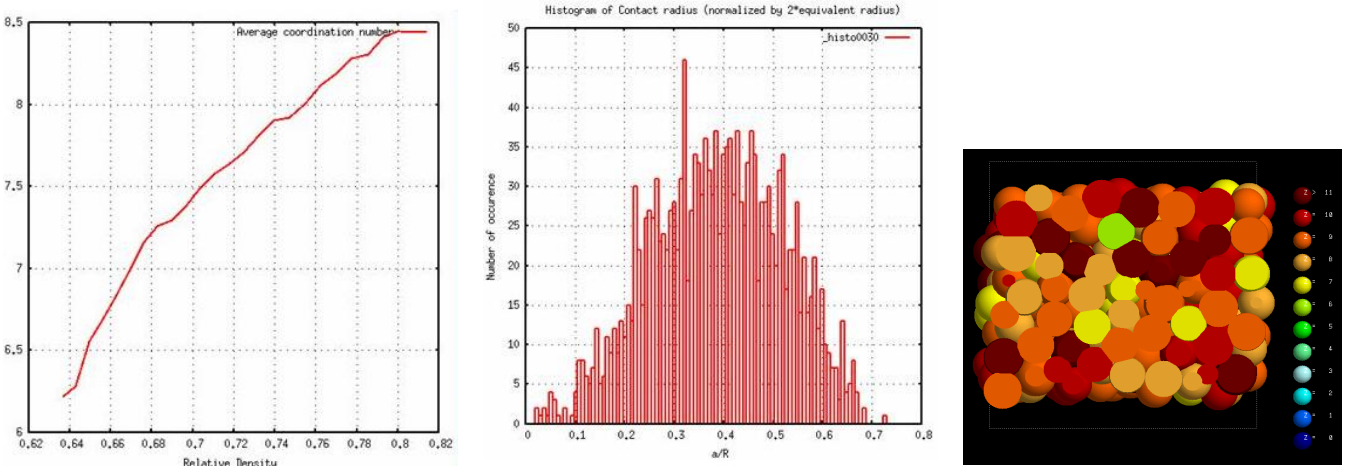
The material is considered as elasto-plastic with the plastic regime defined by the following equation:

$$\sigma = \sigma_1 \varepsilon^{1/m} \tag{5}$$

where σ_1 is a material parameter, m is the hardening coefficient and σ and ε are the stress and strain in the uniaxial case. In this exemple, the material is considered perfectly plastic ($M = 1/m = 0$).

The evolution of the average number of contacts in the packing is shown in fig. 5a. The figure has been generated using the the data stored in the output file `zave` (command `ddp3D -dens -z`), which gives the evolution of the packing coordination number with density. The information on average data can be refined by looking at the histogram. Fig. 5b shows an example of such a plot at 0.80 relative density for the size of the contacts [3]. The microstructure itself can be visualized by using the command `vdp3D`. Fig. 5c has been generated using the `vdp3D` command [4], it shows the coordination number for each particle at 0.84 relative density.

[1] only one non-zero component in the z direction
[2] see discussions on quasi-static conditions in sections 2.9 and 2.12.
[3] `ddp3D -histo -a _coord0030`
[4] `vdp3D -z _coord0040` or `ovdp3D -z _coord0040`



(a) (b) (c)

Figure 5: (a) Evolution of the average coordination number. (b) Contact size distribution (normalized by the particle radius average) at 0.80 relative density. (c) Slab of the sample at 0.84 relative density displaying the coordination number of each particle.

Exercise:

- *Plot the stress evolution versus density using* `ddp3D -dens -sig`
  Why is the σ_{zz} larger than the others?
- *Replace* `sigy(1)=0.4E+09` *by a very large value* `sigy(1)=0.4E+33`. *What is the effect on macroscopic stresses? Look for the status of contacts in the _tag files (plastic, hertzian, plast_unl).*

Contact laws used in this example:

In the `elasto_plasticity` mode, contacts may experience elasticity (Eq. (1)), plasticity and elastic unloading. The plastic contact force is given by the model of Storåkers et al. [4]:

$$N^{plast} = 2^{1-\frac{1}{m}} 3^{1-\frac{1}{m}} \pi \sigma^* c\left(m\right)^{2+\frac{1}{m}} R^{*1-\frac{1}{2m}} \delta_n^{1+\frac{1}{2m}} \tag{6}$$

where $\sigma^* = \left(\frac{1}{\sigma_1} + \frac{1}{\sigma_2}\right)^{-1}$ is the effective material parameter that defines the plastic behaviour of the contact (see eq. (5)) and $c\left(m\right)$ is a function of the hardening parameter m ($c\left(m\right)^2 = 1.43 \exp\left(m\right)$ [4]). Note that σ_1 and σ_2 have the unit of stress but that they represent the yield stress of materials 1 and 2 only for $m \to \infty$. In this simple case (perfectly plastic material), Eq. (6) reduces to:

$$N^{plast} = 6\pi \sigma^* c\left(m\right)^2 R^* \delta_n \tag{7}$$

σ_i are `sigy(i)` in `input_dp3D`[1]. The contact radius in plasticity is governed by:

$$a^2 = 2c\left(m\right)^2 R^* \delta_n \tag{8}$$

The transition from elasticity to plasticity is simply governed in `dp3D` by stating that it occurs when:

$$N^{plast} \leq N^{Hertz} \tag{9}$$

This criterion has the advantage of enforcing a continuous transition in force. However, this has the drawback of overestimating the value of the indentation δ_n at the elastic-plastic transition [18]. Also, the maximum stress under the contact experiences a discontinuous jump (you may observe this behaviour by plotting the contact stress of a tagged contact[2]). This is due to the use of Eqs. (3) and (8) that introduce a discontinuous jump in the size of the contact. `dp3D` is not able to reproduce accurately the complex stress state under contacts.

[1] Note that for a composite, m must be the same for all materials ($\frac{1}{m}$ is `Mstrain` in `input_dp3D`).
[2] `ddp3D -tag -indent -sigN`

```
################################################################################
#                      simulation conditions                                   #
################################################################################
# coordinate file:
p400p6365ur
--------------------------------------------------------------------------------
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
--------------------------------------------------------------------------------
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
none
--------------------------------------------------------------------------------
################################################################################
#                                models                                        #
################################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity
none
--------------------------------------------------------------------------------
# friction model:
# Hertz_Mindlin Coulomb shear
shear
--------------------------------------------------------------------------------
# adhesion model:
# DMT JKR
JKR
--------------------------------------------------------------------------------
################################################################################
#                              outputs                                         #
################################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
density>=0.8500E+00
--------------------------------------------------------------------------------
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep=
density=0.5000E-02
--------------------------------------------------------------------------------
# writing output files:
# density= epsilon= pressure= aoR= time= timestep=
density=0.1000E-02
--------------------------------------------------------------------------------
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
--------------------------------------------------------------------------------
################################################################################
#                              loadings                                        #
################################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
--------------------------------------------------------------------------------
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=0.0000E+00
epsydot=0.0000E+00
epszdot=-0.1000E-05
--------------------------------------------------------------------------------
################################################################################
#                              materials                                       #
################################################################################
# elastic parameters (Pa for stress):
# E(1)= poisson(1)= E(2)= poisson(2)=
E(1)=0.1200E+12
poisson(1)=0.3400E+00
--------------------------------------------------------------------------------
# plastic parameters (Pa for stress):
# sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.4000E+09
Mstrain=0.0000E+00
Nvisco=0.0000E+00
--------------------------------------------------------------------------------
# friction parameters:
# frict(1,1)= frict(1,2)= frict(2,2)= frict(object)=
frict(1,1)=0.1000E+00
--------------------------------------------------------------------------------
# work of adhesion parameters (J.m-2):
# adhes(1,1)= adhes(1,2)= adhes(2,2)= adhes(object)=
adhes(1,1)=0.0000E+00
--------------------------------------------------------------------------------
# density (g.mm-3):
# ro(1)= ro(2)=
ro(1)=0.8706E-02
--------------------------------------------------------------------------------
################################################################################
#                               numerics                                       #
################################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
--------------------------------------------------------------------------------
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
--------------------------------------------------------------------------------
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping=
safe_dt=0.1000E-01
upscale(1)=0.1000E+01
damping=0.5000E-01
--------------------------------------------------------------------------------
```

2.4 Isostatic compaction of a composite packing

- directory : `composite`; coordinate file: `p400p6365pct20_tag`

In the preceding examples, all particles had the same material properties. It is possible in `dp3D` to define up to 10 families of particles with different material properties (from 0 to 9). In this example, two materials (1, and 2) are introduced. For contact laws, this means that 3 types of properties have to be defined (contacts 1-1, 1-2, and 2-2). The composite packing `p400p6365pct20` has been generated using the homogeneous packing `p400p6365ur` and the `cdp3D -mixture` tool (see section 3.3). Type 1 and type 2 particles are defined in the coordinate file `p400p6365pct20` with "cu" and "ni" labels to distinguish visually particles of material 1 ("cu") and particles of material 2 ("ni") (see section 4 for a more detailed discussion on the labels of particles). Also, since three different contacts are present, it is interesting to tag specific contacts and follow their behavior during compaction. This is done by using the command `vdp3D -tag p400p6365pct20`, which generates the file `p400p6365pct20_tag` used here (see section 3).

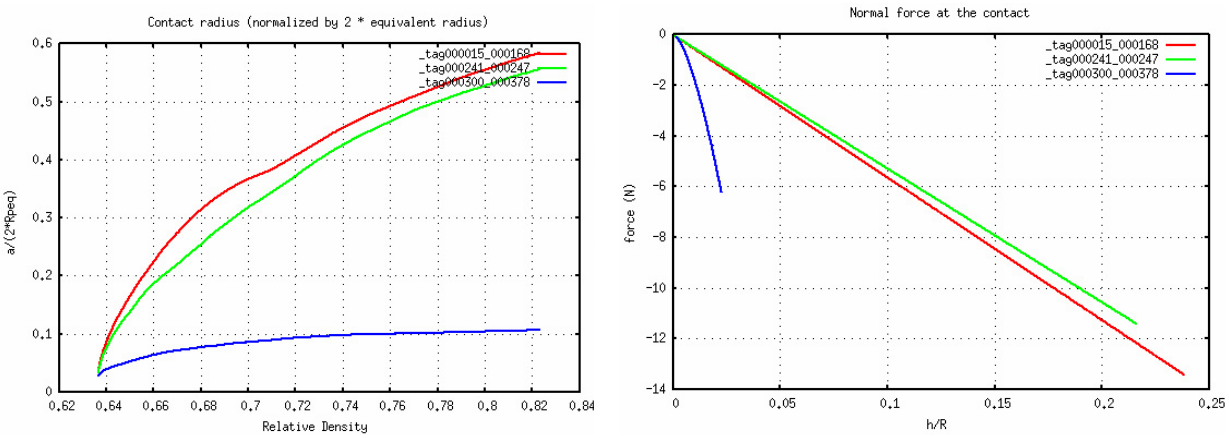
The two families of particles are described with different material properties in `input_dp3D` by the appropriate material data (highlighted data in `input_dp3D`, see next page). Note that the yield stress of material 2 is set to a very large value here to ensure that type 2 particles behave elastically all along the compaction. The friction and adhesion parameters are set to different values in `input_dp3D` for the three types of contacts. Also, note that for this example, the test is terminated by setting `pressure>=200E+06`, which means that the simulation stops when the pressure is above 200 MPa. All strain rates are set to the same value: this is an isostatic compaction test.

Figures 6a and b show the type of information given by tagging the three contacts [1]. Figure 6a demonstrates the effect of having contacts that plastify (large contacts) and contacts that stay fully elastic (small contacts). Figure 6b indicates also that the contact laws behave differently as a function of the indentation between the two particles (Hertzian and ideal plastic), as it should.

It is a good practice to tag some contacts at the early stage of a simulation to **ensure** that the simulated contacts behave as you expected.

Since the packing is a composite, it can be interesting to evaluate the contribution of each material to the overall stress response. This can be done using the command:
`ddp3D -dens -sigzz_mat_1 -sigzz_mat_2 -sigzz`
which displays the macroscopic stress contribution on zz for materials 1, 2 and the total stress. Fig. 7 shows the result of this command. Note that although material 2 plastic yield stress `sigy(2)` is larger than for material

[1] `ddp3D -tag -dens -a` and `ddp3D -tag -indent -forceN`



(a) (b)

Figure 6: (a) Evolution of contact size of three different contacts (plastic-plastic; plastic-elastic; elastic-elastic). (b) Normal force evolution for the same three contacts as a function of the indentation.

1, the overall contribution of material 1 is still larger as particles of material 1 are much more numerous. See section 3.6 for details on stress calculation in `dp3D`.

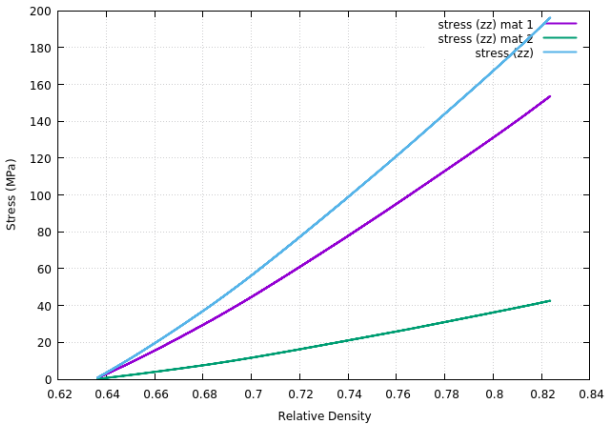


Figure 7: Stress partition with materials for the zz component.

Exercise:

- *Increase all the strain rates by a factor of 10* (`epsxdot, epsydot,epszdot`. *Is the simulation faster in terms of CPU time ? What is the effect on macroscopic stresses? Is this effect due to the behaviour of materials themselves (are there some rate effects introduced in the contact laws (Eqs. (1), (6)) or is it linked to inertia?)*
- *Replace* `density=0.5000E-02` *in the* `# writing coordinate files:` *by* `density=0.5000E-01`. *How many _coord files are written now ?*
- *Replace* `density=0.1000E-04` *in the* `# writing output files:` *by* `none`. *What is the effect on the output files ? (see the* `log` *file for example)*

```
sans titre
########################################################################
#                    simulation conditions                             #
########################################################################
# coordinate file:
p400p6365pct20_tag
________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
none
________________________________________________________________________
########################################################################
#                              models                                  #
########################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity
none
________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
________________________________________________________________________
# adhesion model:
# DMT JKR
JKR
________________________________________________________________________
########################################################################
#                            outputs                                   #
########################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
pressure>=200.E+06
________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep=
density=0.5000E-02
________________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep=
density=0.1000E-04
________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
________________________________________________________________________
########################################################################
#                            loadings                                  #
########################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=-0.1000E-05
epsydot=-0.1000E-05
epszdot=-0.1000E-05
________________________________________________________________________
########################################################################
#                            materials                                 #
########################################################################
# elastic parameters (Pa for stress):
# E(1)= poisson(1)= E(2)= poisson(2)=
E(1)=0.2000E+12
E(2)=0.2000E+12
poisson(1)=0.2200E+00
poisson(2)=0.2200E+00
________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.2000E+09
sigy(2)=0.2000E+12
Mstrain=0.0000E+00
Nvisco=0.0000E+00
________________________________________________________________________
# friction parameters:
# frict(1,1)= frict(1,2)= frict(2,2)= frict(object)=
frict(1,1)=0.1000E+00
frict(1,2)=0.1000E+00
frict(2,2)=0.0500E+00
________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(1,1)= adhes(1,2)= adhes(2,2)= adhes(object)=
adhes(1,1)=0.0000E+00
adhes(1,2)=0.0000E+00
adhes(2,2)=2.0000E+01
________________________________________________________________________
# density (g.mm-3):
# ro(1)= ro(2)=
ro(1)=0.8920E-02
ro(2)=0.1000E-01
________________________________________________________________________
########################################################################
#                             numerics                                 #
########################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
________________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
________________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping=
safe_dt=0.1000E-01
upscale(1)=0.1000E+01
upscale(2)=0.1000E+01
damping=0.5000E-01
________________________________________________________________________
```

2.5 Unloading a compact

- directory : `unl_rel`; coordinate file: `p400p85cd` and `p400p85cd_unl`; contact file: `p400p85cd_histc` and `p400p85cd_unl_histc`

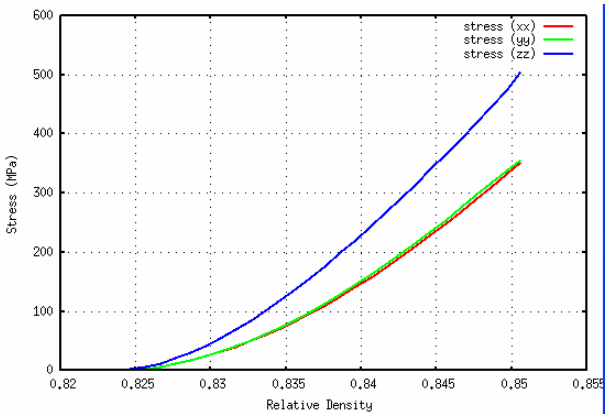
At the end of the compaction stage, the macroscopic stress acting on a packing, which has been densified to a given density, is non-zero. It is possible to unload the packing to obtain the state of the compact under nearly zero compressive stress for example. `dp3D` allows taking into account the local unloading of each contact that has been submitted to elastic (JKR model [2] or DMT [3]) or plastic [5] strains. Some contacts may experience decohesion during the unloading.

In this example, we use the last coordinate file that has been generated in section 2.3 (renamed to `p400p85cd` from the last coordinate file of `closedie_compact` directory test) in the present `input_dp3D` and unload it to 1. MPa by setting `pressure<=1.E+06` as the simulation termination condition.

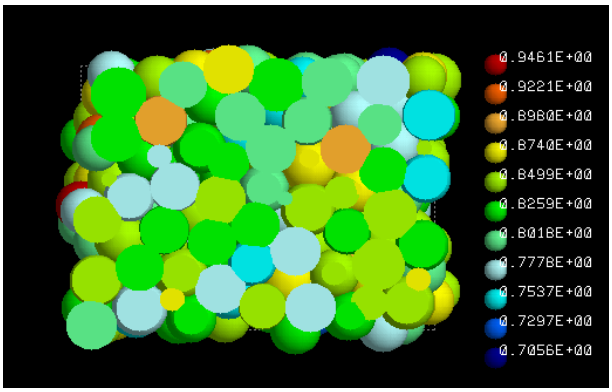
When using a coordinate file with a stress history, it is advisable to also use the contact history file which has been generated by the preceding simulation. It is not absolutely necessary to copy this file for the present calculation but if so, this will ensure that you start the unloading with the exact same contact history as the one calculated in the preceding example. Here, not using the _histc file would result in uncorrect results. Thus, you want to use this file and you need to rename it to `p400p85cd_histc` (or more generally `coordname_histc` if the coordinate file name is `coordname`) and store it in the same directory as p400p85cd so that `dp3D` understands that it has to use it. When the strain history of contacts is negligible for the overall packing behavior (typically elastic behavior), it is not important to save it for the next calculation. Also `sintering` and `viscoplastic` modes make no use, for the time being, of the `_histc` file.

Fig. 8a [1] shows how the stress components decreases during the unloading stage down to a 1 MPa pressure. Note that there exist more involved methods for unloading the compact that can be called with `equal_stress=`. This allows unloading the compact while ensuring that the stress state converges to isostatic. Fig. 8b [2] shows the local density of each particle. This is calculated by using a Voronoi tessellation (voro++ cell library [20]).

[1] `ddp3D -dens -sig`
[2] `vdp3D -dens _coord0003` or `ovdp3D -dens _coord0003`



(a)



(b)

Figure 8: (a) Evolution of the stress components during unloading. (b) Local density map of the sample after unloading.

Exercise:

- *Rename the* `p400p85cd_histc` *file to* `p400p85cd_not2beseen`. *What are the initial stresses at 0.85 relative density when unloading the* `p400p85cd` *compact without its contact history file* `p400p85cd_histc`*? Do they correspond to the final stresses at the end of compaction (see section 2.3)?*
- *Set* `equal_stress` *in* `loadings` *to* `equal_stress=3`. *What is the effect on the unloading?*

Contact laws used in this example:

Once plastified, contacts that unload follow an elastic unloading model (with or without adhesion) [5, 21] that leads to the following expression for the normal force:

$$N^{unload} = 2p_0 a_0^2 \left(\arcsin\left(\frac{a}{a_0}\right) - \frac{a}{a_0}\sqrt{1-\left(\frac{a}{a_0}\right)^2} \right) - 2\sqrt{2\pi w E^*}\, a^{\frac{3}{2}} \tag{10}$$

where the second term accounts for adhesion. Unloading contacts may snap (fracture) if the contact radius is lower than the critical value:

$$\left(\frac{a}{a_0}\right)_{snap} = \left(\frac{9\,(\pi-2)}{4}\right)^{1/3} \chi^{1/3} \tag{11}$$

where a_0 is the maximum contact radius attained during plasticity (see Eq. (8)) and:

$$\chi = \left(\frac{\pi}{2\pi-4}\right)\frac{wE^*}{9\sigma^{*2}a_0} \tag{12}$$

If $\chi \geq 0.1$, the unloading cannot be considered purely elastic and the above model may not be valid. If $\left(\frac{a}{a_0}\right)_{snap} > 1$ (typically if the work of adhesion w is too large), `dp3D` stops as it is not possible to treat this problem satisfactorily.

```
##########################################################################
#                    simulation conditions                               #
##########################################################################
# coordinate file:
p400p85cd
__________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
__________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
none
__________________________________________________________________________
##########################################################################
#                              models                                    #
##########################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity
none
__________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
shear
__________________________________________________________________________
# adhesion model:
# DMT JKR
JKR
__________________________________________________________________________
##########################################################################
#                            outputs                                     #
##########################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
pressure<=1.E+06
__________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep=
epsilon=0.500E-02
__________________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep=
epsilon=0.100E-02
__________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
__________________________________________________________________________
##########################################################################
#                           loadings                                     #
##########################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
__________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=0.100E-05
epsydot=0.100E-05
epszdot=0.100E-05
__________________________________________________________________________
##########################################################################
#                           materials                                    #
##########################################################################
# elastic parameters (Pa for stress):
# E(1)= poisson(1)= E(2)= poisson(2)=
E(1)=0.120E+12
poisson(1)=0.340E+00
__________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.400E+09
Mstrain=0.000E+00
Nvisco=0.000E+00
__________________________________________________________________________
# friction parameters:
# frict(1,1)= frict(1,2)= frict(2,2)= frict(object)=
frict(1,1)=0.100E+00
__________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(1,1)= adhes(1,2)= adhes(2,2)= adhes(object)=
adhes(1,1)=0.000E+00
__________________________________________________________________________
# density (g.mm-3):
# ro(1)= ro(2)=
ro(1)=0.870E-02
__________________________________________________________________________
##########################################################################
#                            numerics                                    #
##########################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
__________________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
__________________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping=
safe_dt=0.100E-01
upscale(1)=0.100E+01
damping=0.500E-01
__________________________________________________________________________
```

2.6 Use of cylinder objects together with periodic boundary conditions

- directory : `cylinder`; coordinate file: `cyl_p51`

In the preceding examples, boundary conditions were fully periodic (`1x1y1z` in `loadings`). This means that until this example, no rigid object was used. It is possible to introduce **objects** in `dp3D`. Objects can be: plane, cylinder or sphere[1]. Here we use two concentric cylinders which walls will interact with the particles. The command `cdp3D -cut_cyl` has been used to create a hollow cylinder by removing particles. Together with the two walls, the boundary conditions are completed by periodic boundary conditions in the z direction as declared in `input_dp3D` (`0x0y1z`, see highlighted). The material of the walls is given by type 0 material. As highlighted in the `input_dp3D` file, the yield stress of the wall is set deliberately to a large value.

```
particles:        400 objects:        2 bonded_contacts:          0  mean_radius:   0.100000012138E-02
 -0.846999242285E-02 -0.846999242285E-02 -0.971993215510E-02  0.846999242285E-02  0.846999242285E-02  0.971993215510E-02
1  cu'    -0.733720887308E-02  0.158331341239E-02 -0.788256418639E-02  0.963979476871E-03
1  cu     -0.680260222237E-02 -0.117598840021E-02  0.713358026952E-03  0.979030477261E-03
1  cu      0.742740359951E-02  0.869784479627E-04  0.583579314347E-02  0.104216860347E-02
1  cu      0.399951995384E-02 -0.420456248884E-02  0.386905133223E-02  0.969739255604E-03
1  cu      0.220817359483E-02 -0.479111722846E-02  0.316971545448E-02  0.104054897444E-02
1  cu     -0.193281238682E-02  0.723948918431E-02 -0.544106001577E-02  0.976973416482E-03
1  cu      0.560953970864E-02  0.443337803007E-02  0.920096909057E-02  0.101640327436E-02
1  cu      0.692625626780E-02  0.212054535549E-02  0.263842611128E-02  0.972508040059E-03
1  cu     -0.175307292944E-02 -0.723881985613E-02 -0.949918051006E-02  0.102194050879E-02

  list of objects:
** *****  ******************   ******************   ******************   ******************
0  cylin  0.000000000000E+00   0.000000000000E+00   0.100000000000E+01   0.846999242285E-02
   C_cyl  0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
** *****  ******************   ******************   ******************   ******************
0  cylin  0.000000000000E+00   0.000000000000E+00   0.100000000000E+01   0.423499621059E-02
   C_cyl  0.000000000000E+00   0.000000000000E+00   0.000000000000E+00
** *****  ******************   ******************   ******************   ******************
   bonded contacts (i,j):        0
```

cylinder 1 radius

cylinder 2 radius

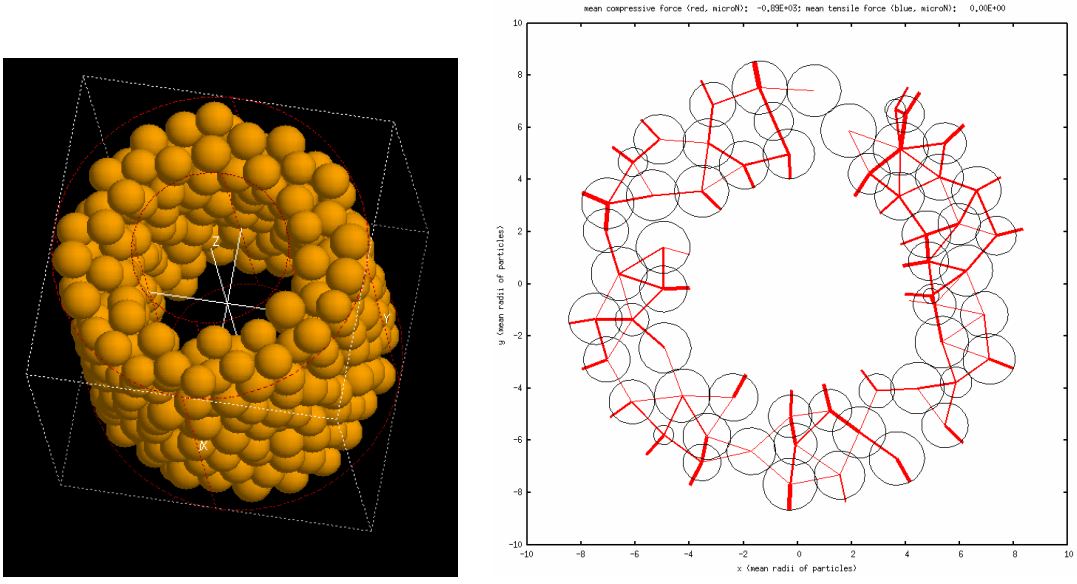
cylinder 1 axis

The objects themselves are declared in the initial coordinate file `cyl_p51` (see above the portion of the file `cyl_p51`). The number of objects (`2`) is declared on the first line of `cyl_p51`. Two lines are needed to describe the cylinders. The first line with the label `cylin` is followed by the unit axis vector and the radius of the cylinder. The second set needed to define the cylinder is given after the label `C_cyl` which defines the coordinates of a point on the axis of the cylinder.

The microstructure resulting from file `cyl_p51` is shown in fig. 9a. Fig. 9b shows the normal contact force network on a slab at mid height of the cylinder[2]. After the uniaxial compaction simulated in this example, we rename the last generated coordinate file to cylp85. This file will be used for the next example.

[1] denoted as `plane`, `cylin`, and `spher`, respectively in the `coord` file

[2] `vdp3D -Nnetwork` applied on the last coordinate file



(a) (b)

Figure 9: (a) Packing with two concentric cylinders objects and periodic B.C. on z axis. (b) Normal contact force network on the slab at mid height (`vdp3D -Nnetwork` on the last coordinate file).

Exercise:

- *Edit the* `cyl_p51` *file and add the following line under the two* `cylin` *lines that define the cylinders:*
  `irate 0.000000000E+00 0.000000000E+00 0.000000000E+00 -0.100000000E-05`
  What is the effect on the cylinder radii in the simulation ? Check the difference in the initial and final radius of the cylinder.
- *Remove the inner cylinder by careful edition of the* `cyl_p51` *file. Ensure that you define the correct number of objects.*

```
#########################################################################
#                      simulation conditions                            #
#########################################################################
# coordinate file:
cyl_p51
_________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
_________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity
none
_________________________________________________________________________
#########################################################################
#                             models                                    #
#########################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity linear_elasticity
none
_________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
_________________________________________________________________________
# adhesion model:
# DMT JKR
JKR
_________________________________________________________________________
#########################################################################
#                            outputs                                    #
#########################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
density>=0.8500E+00
_________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep= none end
density=0.1000E-01
_________________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep= none end
density=0.1000E-02
_________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
_________________________________________________________________________
#########################################################################
#                            loadings                                   #
#########################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
0x0y1z
_________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=0.000E+00
epsydot=0.000E+00
epszdot=-0.1000E-05
_________________________________________________________________________
#########################################################################
#                        materials (from 0,1,2, ... to 9)               #
#########################################################################
# elastic parameters (Pa for stress):
# E(0)= poisson(0)= E(1)= poisson(1)= E(2)= poisson(2)=
E(1)=0.2000E+12
E(0)=0.2000E+12
poisson(1)=0.3000E+00
poisson(0)=0.3000E+00
_________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.3000E+10
sigy(0)=0.3000E+12
Mstrain=0.0000E+00
Nvisco=0.0000E+00
_________________________________________________________________________
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.2000E+00
frict(0,1)=0.6000E+00
_________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.000E+00
adhes(0,1)=0.0000E+00
_________________________________________________________________________
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.7890E-02
_________________________________________________________________________
#########################################################################
#                             numerics                                  #
#########################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
_________________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
_________________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
safe_dt=0.1000E-01
upscale(1)=0.1000E+01
damping=0.5000E-01
_________________________________________________________________________
```

2.7 Plastic compaction in a cylinder up to large relative density

- directory : `large_dens`; coordinate file: `p51p4kpartp5`

The compaction applications in the preceding examples were stopped at intermediate relative densities (typically 0.85). In reality, many compaction processes go for for much higher densities. A priori, DEM is not well suited for these high densities as contact impingement, contact interactions will invalidate the simplifying assumption of pair interactions (i.e. a contact constitutive law (normal and tangential force laws) does not depend on neighboring contacts). Here, we use the keyword `large_dens` to invoke an elasto-plastic model with hardening that depends on the local relative density around the particles. The model is derived from [22] and the PhD thesis of Achraf Kallel, and addresses specifically high density compaction. The value given in `large_dens=1.` means that the material becomes incompressible when the `dp3D` calculated relative density approaches 1. It may be set to other values if needed (see (18)). It applies to particles which plastic behavior is defined by a simple constitutive equation of the type:

$$\sigma = \sigma_i \epsilon^{1/m} \tag{13}$$

with σ and ϵ the uniaxial stress and strain, and σ_i and $1/m$ the hardening parameters for the material of particle i (denominated `sigy` and `Mstrain` in `input_dp3D`).

An example of the use of such a model is given for a packing of 4000 particles in a cylinder and with planes acting as pistons in the axial direction. Note the use of the key word `large_dens` that calls for a calculation of the local density around each particle using the Voro++ program [20], and thus making this type of calculation more CPU intensive than the standard one.

Fig. 10 shows the result of the compaction simulation with and without the `large_dens` keyword. Note that stresses are calculated on the objects simply by adding the total contact forces on the planes (for the axial stress) divided by the surface. The keyword `stress_ref` is also invoked in the `input_dp3D` file. Together with the keyword `stress_ref` in the coordinate file `p4kpartp5` when defining objects (see section 4.3) it ensures that reference stresses (those that are used for example to set a stop condition on the simulation) are calculated from these objects (not from the total volume with the Love's in Eq. (63)).

The coordinate file `p4kpartp5` is necessarily accompanied by the file `p4kpartp5_voronoi_dens`. This is because the model with `large_dens` needs the initial Voronoi density around each particle. This file contains this information and can be generated by the command:

```
vdp3D -dens p4kpartp5
```

or

```
ovdp3D -dens p4kpartp5
```

The same file should be copied if an unloading is planned as explained in the exercise below.

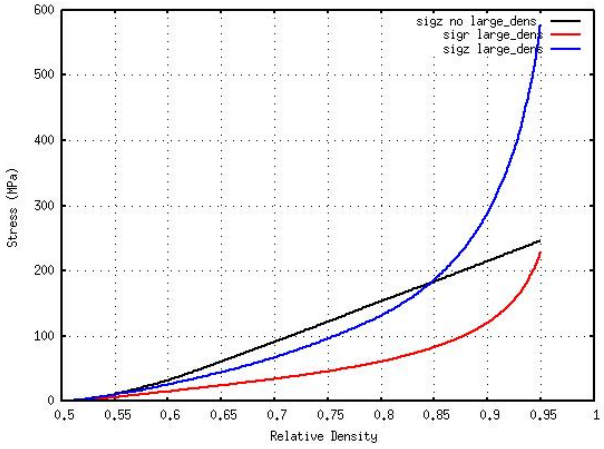


Figure 10: Axial and radial stress evolution of a packing in a cylinder without or with the `large_dens` keyword.

Exercise:

- *copy the last coordinate file from the compaction into an* `unloading` directory and rename it `cload`. *Copy the* `p4kpartp5_voronoi_dens` *file as* `cload_voronoi_dens` *in this directory. Copy the* `input_dp3D` *file and edit this file to impose a tensile strain rate* (`epszdot=0.1000E-04`). *Set a stop condition for the simulation at* `sigzz>=-50.E+06`. *Run the simulation to check that indeed you have imposed an elastic unloading and that it stopped at the correct 50 MPa compressive axial stress.*
- *Similarly, using the last coordinate file from the unloading, set up a simulation to reload up to 0.95 relative density. Check if indeed an elastic unloading-reloading sequence is simulated.*

Contact laws used in this example:

For two particles i and j, upon plastic loading or reloading, the normal repulsive force writes in incremental formulation at time $t + \mathrm{d}t$:

$$N^{plast}\left(t+\mathrm{d}t\right) = N^{plast}\left(t\right) - 2S_{ij}\mathrm{d}\delta_n \tag{14}$$

where the stiffness S_{ij} is assumed to be a sum of two independent terms (the factor 2 in Eq.(14) stems from the fact that [22] consider a different definition of δ_n):

$$S_{ij} = 2\sigma^* R^* \left(S_1 + S_2\right) \tag{15}$$

where σ^* is the equivalent stress parameter for particles i and j:

$$\sigma^* = 2^{\frac{1}{m}} \left(\sigma_i^{-m} + \sigma_j^{-m}\right)^{-\frac{1}{m}} \tag{16}$$

Note that the hardening parameter must be the same for the two particles even though the parameter σ_i may be material dependent. The S_1 and S_2 stiffnesses are written:

$$S_1 = \alpha_1\left(m\right)\exp\left(\beta_1\left(m\right)\frac{\delta_n}{2R^*}\right) + \gamma_1\left(m\right)\exp\left(\delta_1\frac{-\delta_n}{2R^*}\right) \tag{17}$$

$$S_2 = \alpha_2\left(m\right)\frac{\left[\max\left(0, \rho_{i,j} - \rho_{0i,j}\right)\right]^2}{d_1 - \rho_{i,j}} \tag{18}$$

where α_1, α_2, β_1, γ_1 are functions of m and δ_1 is a constant (see [22] for their values). Eq. (18) introduces an important aspect of this model: the local density $\rho_{i,j}$ and its initial value (before plastic indentation) $\rho_{0i,j}$. The S_2 stiffness tends to infinity when $\rho_{i,j} \to d_1$, thus providing a simple mean to verify the incompressibility condition. Note that the local density $\rho_{i,j}$ is given by using tessellations obtained with the Voro++ package [20], which is called periodically by `dp3D`. Also, note that if the keyword `large_dens` has been given a value different from unity ($d_1 \neq 1$), it means that the material becomes incompressible when approaching d_1.

Eq.(14) applies when the contact is loading plastically. Noting δ_1 the maximum indentation attained by a contact, Fig. 11 shows the sketch of N_{plast} up to indentation δ_1 when it unloads[1]. When the contact unloads (unloading branch defined by $\delta < \delta_1$), we consider that the elastic solution of a bond (see Eq.(25)) applies and can be added to the force of the maximum plastic indentation N_1:

$$N^{unl} = N_1 + \frac{E}{1-\nu^2} f_N\left(\frac{a_b}{R^*}, \nu\right) a_b \delta_{unl} \tag{19}$$

where a_b is the radius of the bond and δ_{unl} is the indentation difference with the maximum plastic indentation δ_1 ($\delta_{unl} = \delta_1 - \delta$). The elastic unloading thus depends on the elastic parameters of the particles and on the size a_b of this bond.

Note that to ensure that the elastic unloading is always stiffer than the plastic loading, S_{ij} is capped by the unloading stiffness:

$$S_{ij} = \min\left(S_{ij}, \frac{E}{1-\nu^2} f_N\left(\frac{a_b}{R^*}, \nu\right) a_b\right) \tag{20}$$

[1] Note that in Fig. 11 forces are positive when repulsive whereas in `dp3D` or in Eq.(14) the standard convention of positive tensile force applies.

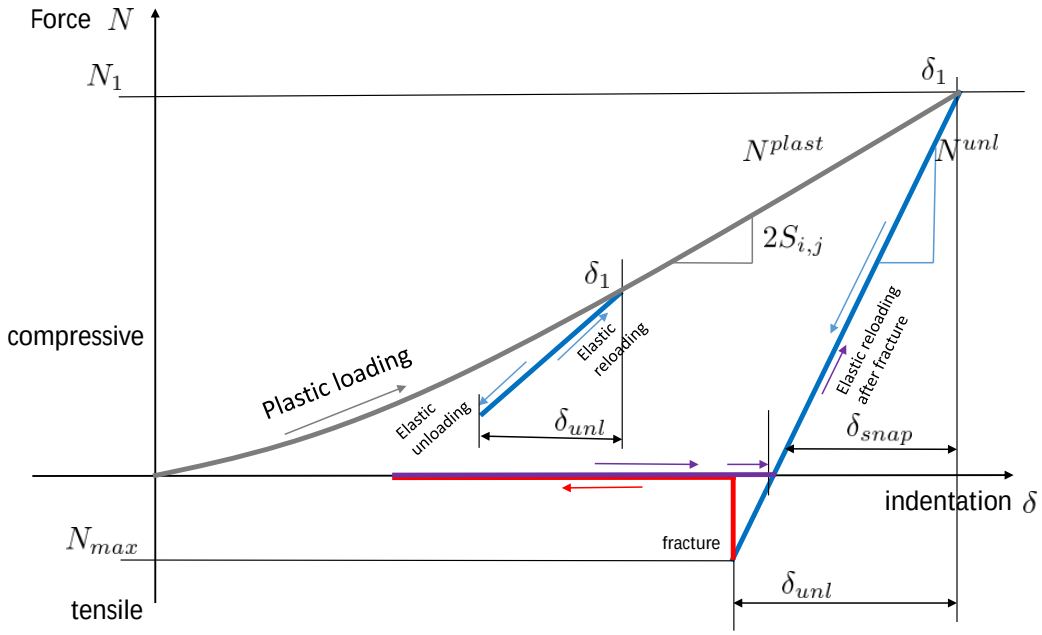


Figure 11: Sketch of the normal force evolution with indentation history. Note how the unloading stiffness increases as the bond size a_b (or plastic indentation δ_n) increases. Here forces are shown positive when repulsive.

With this scheme, it is assumed that a sufficiently strong bond has formed upon plastic deformation between particles. This bond may sustain tensile force as sketched in Fig. 11. The value for fracture is given by N_{max}:

$$N_{max} = \sigma_N \pi a_b^2 \tag{21}$$

where σ_N is the tensile strength (`sig_N` in `input_dp3D`). Note that for the time being no shear strength is introduced in this model. Bonds only fracture in tension. The value of the plastic contact radius is given by:

$$a_b^2 = 2c\,(m)^2\, R^* \delta_1 \tag{22}$$

where $c\,(m)^2$ is a hardening parameter that depends on m. When the tensile force N reaches N_{max} the bond breaks and does transmit any force. If the contact resumes, it will only transmit repulsive force when $\delta_{unl} = \delta_1 - \delta < \delta_{snap}$. The value of δ_{snap} is simply given by:

$$\delta_{snap} = \frac{N_1}{\frac{E}{1-\nu^2} f_N\left(\frac{a_b}{R^*}, \nu\right) a_b} \tag{23}$$

Once this value is reached, the contact reloads elastically following the N^{unl} branch in Fig.11 until the indentation increases above δ_1 and the contact re-plastifies.

```
################################################################################
#                      simulation conditions                                   #
################################################################################
# coordinate file:
p4kpartp5
--------------------------------------------------------------------------------
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
--------------------------------------------------------------------------------
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity thermal
large_dens=1.
no_rotation
stress_ref
--------------------------------------------------------------------------------
################################################################################
#                                models                                        #
################################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity linear_elasticity
none
--------------------------------------------------------------------------------
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
--------------------------------------------------------------------------------
# adhesion model:
# DMT JKR
JKR
--------------------------------------------------------------------------------
################################################################################
#                                outputs                                       #
################################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
density>=0.95
--------------------------------------------------------------------------------
# writing coordinate files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
density=0.1
--------------------------------------------------------------------------------
# writing output files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
sigzz=1.E+06
--------------------------------------------------------------------------------
# writing contact history files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
end
--------------------------------------------------------------------------------
################################################################################
#                                loadings                                      #
################################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
0x0y0z
--------------------------------------------------------------------------------
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=-0.
epsydot=-0.
epszdot=-0.1000E-03
--------------------------------------------------------------------------------
################################################################################
#                          materials (from 0,1,2, ... to 9)                    #
################################################################################
# elastic parameters (Pa for stress):
# E(0)= poisson(0)= E(1)= poisson(1)= delta_c(0)= fact_mult(0)=
E(1)=100.E+09
E(0)=200.000E+09
poisson(1)=0.3400E+00
poisson(0)=0.3400E+00
--------------------------------------------------------------------------------
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=100.E+06
sigy(0)=20.5E+33
Mstrain=0.
Nvisco=0.0000E+00
--------------------------------------------------------------------------------
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=.3
frict(2,2)=.3
frict(1,2)=.3
frict(0,1)=.3
--------------------------------------------------------------------------------
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.0000E+00
--------------------------------------------------------------------------------
# bond strength in tension (Pa):
# sig_N(0,0)= sig_N(0,1)= sig_N(1,1)= sig_N(1,2)= sig_N(2,2)=
sig_N(1,1)=5.E+09
sig_N(0,1)=0.
--------------------------------------------------------------------------------
# bond strength in shear (Pa):
# sig_T(0,0)= sig_T(0,1)= sig_T(1,1)= sig_T(1,2)= sig_T(2,2)=
sig_T(1,1)=0.
sig_T(0,1)=0.
--------------------------------------------------------------------------------
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.8706E-02
--------------------------------------------------------------------------------
################################################################################
#                                numerics                                      #
################################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
#0x0y0z
--------------------------------------------------------------------------------
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
--------------------------------------------------------------------------------
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
fixed_dt=0.05
upscale(1)=0.1000E+01
damping=0.5000E-01
--------------------------------------------------------------------------------
```

2.8 Generating bonds to form clusters

- directory : `gen_bonds`; coordinate file: `cylp85`

In the example on the hollow cylinder compaction (section 2.6), we have created large contacts by plastically deforming the particles. These contacts cannot hold much tensile stress since their tensile strength is only due to plastic deformation and adhesion. Suppose that the compact formed at the preceding stage is sintered without any further densification. This stage will create strong **bonds** that are able to sustain both tensile and compressive stresses (see [13, 23] for details on the bonds model). Some anisotropy characterizes the packing (due to the uniaxial strain rates involved) but that is not detrimental for the purpose of this example.

The present example shows how to create **bonds** by transforming all existing contacts between particles in the packing into sintered contacts (**bonds**). Additionally you may, during this stage, create bonds between particles and objects to model the sintering of particles on a substrate. This will not be attempted in this example.

We will use the bonded packing generated in this example to carry out a crushing test in the next example (section 2.9). Hence, the first step consists in modifying the boundary conditions in order to obtain suitable boundary conditions for crushing the 'sintered' body in between two planes which normals are in the x direction. This is done by first removing the objects (the two cylinders) [1]. Then planes are inserted [2]. Simply ask `cdp3D -bc` to keep the periodic boundary conditions only in the z direction. `cdp3D -bc` will introduce two planes that are tangent to the packing on the x direction and two planes that are away from the packing in the y direction. `cdp3D -bc` will create a new coordinate packing `cylp85_rm_bc`.

The last step consists in generating bonds [3].

Before running `cdp3D -bonds`, the packing `cylp85_rm_bc` had 400 isolated particles[4] and has no isolated particles when completing the bonding process. The bonded packing name is `cylp85_rm_bc_bonds`. The end of the new coordinate file contains a list of the bonds.

[1] `cdp3D -rmobj cylp85`, which creates the cylp85_rm file
[2] `cdp3D -bc cylp85_rm`
[3] `cdp3D -bonds cylp85_rm_bc`
[4] In the sense that none of these particles were bonded

2.9 Crushing of a cluster in between two planes

- directory : `crushing`; coordinate file: `cylp85_rm_bc_bonds_size_tag`

In this example, the bonded packing created in the last section is crushed in between two planes. The size of all particles is first reduced to 1μm [1]. Five particles have been 'tagged' to follow their contacts during the test [2]. The tagged coordinate file is `cylp85_rm_bc_bonds_tag`. The tagged particles are colored in deep pink when viewed with `vdp3D` (fig. 12). The parameters that define the contacts between tagged particles are stored and can be followed after the calculation as illustrated at the end of this section. Note that the only motion imposed is the motion of the two planes with x normal vectors. This motion is imposed like for the periodic conditions in the preceding example by setting the `epsxdot=-0.1000E-06` value. Other methods to impose object motion are described in section 4.

To invoke the bonding model, the `bonds` keyword is set in the `models` section of the `input_dp3D` file. Several bonding models are available in `dp3D` to simulate sintered elastic bonds between particles. The latest and most useful is the one given by the bond key words `large_bonds_full` and `psi=0.`. It uses the model described in [14] without interaction (`psi=0.`). This last option tells `dp3D` not to consider interactions between contacts. `large_bonds_full` with psi=0. model reproduces well the uniaxial response of bonded clusters but underestimates the Poisson's ratio. A more elaborated model for solid bonds, `large_bonds_full` without `psi=0.`, allows a better estimate of the Poisson's ratio (see section 2.10) but is much more CPU intensive (by a factor 2 approximately).

The keyword `beam` is used to ensure that the maximum stress calculation on the bond uses the beam model which takes into account the flexure moment on the bond (the second term in Eq. (6) of [24]). If this keyword is absent, the stress on the bond is calculated without the moment term (and thus the bond will break less easily).

The test is stopped when any of the three principal strains attains 0.05 (`epsilon>=0.5000E-01`) in absolute value as stated in the `input_dp3D` file. However, the `fracture=0.05` key word has also been set in the `simulation termination` keywords to indicate that the test may stop when macroscopic fracture is detected. Fracture is considered to happen when the stress has decreased below 0.05 times the maximum stress encountered during the test.

The yield properties of particles have been deliberately set to very large values to model elastic particles (typically ceramic type). The properties of objects are given by material 0 here and a large friction coefficient is set between the particles of the cylinder and the planes to ensure that the cylinder does not roll [3].

A new parameter is introduced in this example which deals with fracture: the strength of bonds. Bonds can fracture in normal or tangential mode. Here we set explicitly these two values in the `input_dp3D` file: `sig_N(1,1)=0.2000E+10` and `sig_T(1,1)=0.2000E+10`. When the normal (or tangential) bond stress is larger than the strength; the bond is broken (see [25] for more details on the fracture criterion). Another method to model fracture, using fracture energy, will be introduced in section 2.11. Note also that the Rankine criterion can be used (see [26] and 6) by setting the `Rankine` keyword in the `input_dp3D` file.

Finally, this examples introduces a method to ensure a **quasi-static** test. In `dp3D`, tests are considered quasi-static (inertia terms should play a negligible role), which allows a renormalization of masses and thus a large gain in CPU time. Quasi-static conditions can be ensured by setting a very low strain rate to the test. However this may not be sufficient when the test involves fracture of bonds that may trigger large kinetic energy release in the system. A normalized kinetic energy is defined as:

$$\tilde{E}_c = \frac{E_c}{n \max(NR)} \tag{24}$$

where E_c is the kinetic energy of the packing (taking into account translation and rotation), $\max(NR)$ is the maximum of the product of contact force and particle radius in the packing, and n is the number of particles in the packing. Our experience is that $\tilde{E}_c = 1.E^{-08} - 1.E^{-07}$ is a good range of values for ensuring quasi-static

[1] `cdp3D -resize cylp85_rm_bc_bonds`
[2] `vdp3D -tag cylp85_rm_bc_bonds_size`
[3] `frict(0,1)=0.9`

conditions (in that case, results do not depend anymore on the strain-rate for a rate independent constitutive law). When using `kin_energy=0.1000E-07` criterion, the strain-rate is recalculated starting from the one given by the user (`epsxdot=-0.1000E-06`). The value `ctrl_fact=100.` tells `dp3D` that the strain-rate cannot be multiplied or divided by more than 100. [1] Other criteria for ensuring a quasi-static simulation may be used: the fracture of a bond which size is larger than a given value (`aoamean=`, typically 0.5), a given strain (`epsilon=`), the maximum value of particle velocity in adimensionalized units (`vmax=`, typically between 50 and 200), or when a bond is approaching fracture (`break=`, typically 0.0001).

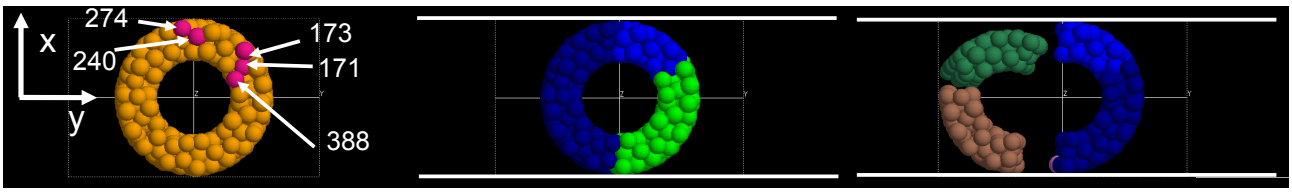
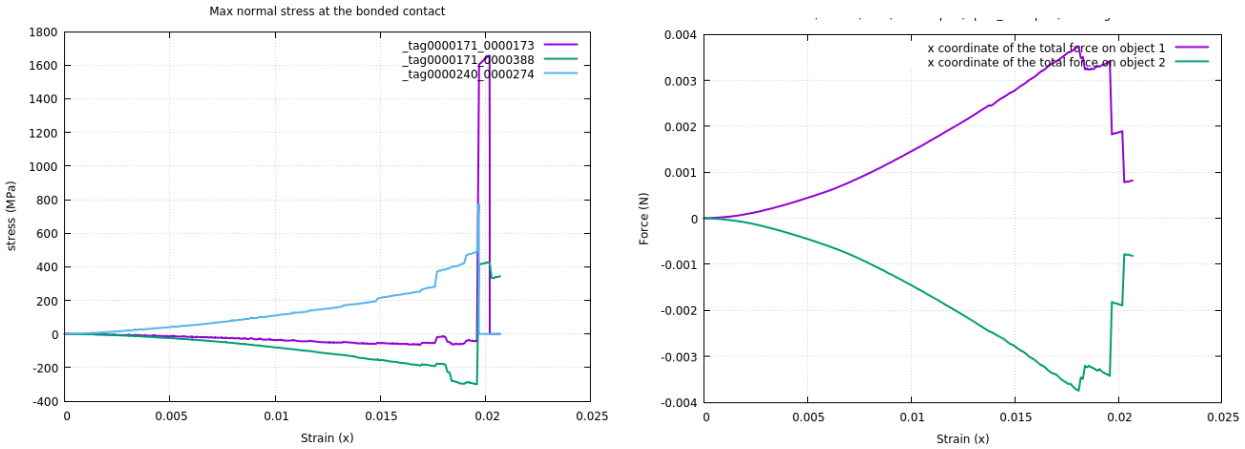


Figure 12: Evolution of the cylinder during the crush tests with the initial packing showing the tagged particles (command `vdp3D -class _coordxxxx` for the last two images).



(a) (b)

Figure 13: (a) Maximum stress at the periphery of bonds for tagged particles (b) x component of the total force on the four objects used in the test.

Note that the relative density is calculated here simply as the ratio of particle volume to the rectangular volume given by the most protruding particle in the simulation box. Consequently, the relative density calculated in this example is not meaningful[2]. Only macroscopic forces exerted on the objects are correct in this example. Fig. 12 shows the packing evolution during crushing [3]. A cluster is defined by the class of equivalence of bonds (if particle i is bonded to particles j and k, then i, j and k form a cluster and are shown with the same color).

The contact of the tagged particles may be followed [4]. For example, the maximum stress acting on the bond periphery is shown in fig.13a for the two contacts between the two pairs of particles that have been tagged. fig.13b shows the x component of the total force on the four objects used in the test[5]. Only object 1 and 2 (the two planes with normals in the x direction) actually experience any contact with the particles.

[1] A larger value of `kin_energy` can be used for CPU intensive simulations, but the user must realize that in that case quasi-static conditions are not fully ensured and that stresses might be over-evaluated.

[2] The macroscopic stress calculation is not meaningful either since it uses the simulation box volume as well

[3] `vdp3D -class _coordxxxx` which sets all particles that pertain to the same cluster to a given color

[4] `ddp3D -tag -epsx -sigN_b` (see section 3.5)

[5] ddp3D -epsx -forcex

Exercise:

- *Edit the* `input_dp3D` *file to remove all control on strain rates by removing the* `kin_energy` *and* `ctrl_fact` *key words and by setting instead the key word* `none`. *Increase the imposed strain-rate to:* `epsxdot=-1.E-04`.
  What is the effect on the cylinder fracture process in the simulation ?

Contact laws used in this example:

In this example, particles are bonded together by elastic bonds. These bonds impose normal and tangential forces that are spring-like. This means that starting from an initial indentation $\delta_{n,0}$, the normal force is symmetrically tensile or compressive depending on the relative indentation $u_n = \delta_n - \delta_{n,0}$:

$$N^{bond} = \frac{E}{1-\nu^2} f_N\left(\frac{a_b}{R^*}, \nu\right) a_b u_n \tag{25}$$

where $f_N\left(\frac{a_b}{R^*}, \nu\right)$ is a function that depends on the radius of the bond a_b and on the Poisson's ratio [13]. Typically, the correcting functions f_N, that characterizes the departure from the small-bond approximation is equal to 1.6 a large bond $a_b = 0.4R$, and tends to unity as the bond size decreases.

The tangential force law for the bond follows a similar law:

$$T^{bond} = -\frac{2E}{(1+\nu)(2-\nu)} f_T\left(\frac{a_b}{R^*}, \nu\right) a_b u_t \tag{26}$$

where u_t is the tangential accumulated relative displacement between the two particles and $f_T\left(\frac{a_b}{R^*}, \nu\right)$ is a correcting functions similar to f_N.

Resisting moments M_N and M_T oppose relative rotations between two bonded particles:

$$M_N = -\frac{E}{(2-\nu)(1+\nu)} f_T\left(\frac{a_b}{R^*}, \nu\right) a_b^3 \theta_N \tag{27}$$

and

$$M_T = -\frac{1}{4}\frac{E}{(1-\nu^2)} f_N\left(\frac{a_b}{R^*}, \nu\right) a_b^3 \theta_T \tag{28}$$

where θ_N and θ_T represent the accumulated relative rotation along the normal and tangential axis of the bond, respectively.

Bond fracture is dictated in this example by the strengths of the bond in tension and in shear (sig_N and sig_T in `input_dp3D`). Approximating the solid bridge to a cylindrical beam of radius a_b, the maximum tensile and shear stresses at the bond periphery may be evaluated by beam theory:

$$\sigma_N = \frac{N^{bond}}{\pi a_b^2} - \frac{4|M_T|}{\pi a_b^3}, \quad \sigma_T = \frac{T^{bond}}{\pi a_b^2} + \frac{2|M_T|}{\pi a_b^3} \tag{29}$$

Bond fracture arises when one of the two following criterion is met -force are negative in tension here):

$$-\sigma_N > \texttt{sig\_N} \ , \quad \sigma_T > \texttt{sig\_T} \tag{30}$$

A fractured bond does not transmit any tensile stress. However, a fractured bond may still transmit a compressive stress (Eq. (25)). A fractured bond may transmit a shear force according to a Hertz-Mindlin friction law. Correspondingly, a fractured bond continues to transmit a resisting moment in the tangential direction but none in the normal direction.

```
################################################################
#                    simulation conditions                     #
################################################################
# coordinate file:
cylp85_rm_bc_bonds_size_tag
________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity
none
________________________________________________________________
################################################################
#                           models                             #
################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity linear_elasticity
bonds
________________________________________________________________
# bond key words:
# large_bonds_full large_bonds geom toughness impinge only_bonds
# clump_cluster beam stiffness  Rankine iso_bonds= psi_bar=
# unload_stiff_ratio= plast_stiff_ratio= strength_deviation=
large_bonds_full
psi_bar=0.0000E+00
beam
________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
________________________________________________________________
# adhesion model:
# DMT JKR
JKR
________________________________________________________________
################################################################
#                          outputs                             #
################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
epsilon>=0.5000E-01
fracture=0.05
________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep= none end
epsilon=0.1000E-02
________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep= none end
epsilon=0.1000E-03
________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
________________________________________________________________
################################################################
#                          loadings                            #
################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
0x0y1z
________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure
epsxdot=-0.1000E-06
epsydot=0.0000E+00
epszdot=0.0000E+00
________________________________________________________________
################################################################
#                  materials (from 0,1,2, ... to 9)            #
################################################################
# elastic parameters (Pa for stress):
# E(0)= poisson(0)= E(1)= poisson(1)= E(2)= poisson(2)=
E(1)=0.2000E+12
E(0)=0.2000E+12
poisson(1)=0.2000E+00
poisson(0)=0.2000E+00
________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.8000E+11
sigy(0)=0.8000E+11
Mstrain=0.0000E+00
Nvisco=0.0000E+00
________________________________________________________________
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.5000E+00
frict(0,1)=0.9000E+00
________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.0000E+00
________________________________________________________________
# bond strength in tension (Pa):
# sig_N(0,0)= sig_N(0,1)= sig_N(1,1)= sig_N(1,2)= sig_N(2,2)=
sig_N(1,1)=0.2000E+10
________________________________________________________________
# bond strength in shear (Pa):
# sig_T(0,0)= sig_T(0,1)= sig_T(1,1)= sig_T(1,2)= sig_T(2,2)=
sig_T(1,1)=0.2000E+10
________________________________________________________________
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.8706E-02
________________________________________________________________
################################################################
#                          numerics                            #
################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
kin_energy=0.1000E-07
ctrl_fact=100.
________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
safe_dt=0.1
upscale(1)=0.1000E+01
damping=0.5000E-01
________________________________________________________________
```

2.10 Elastic effective properties of a bonded cluster

- directory : `eff_prop_bonded_aggregate`; coordinate file: `random_5k_D050_D76_bond`

In this example, the effective Young's modulus and Poisson's ratio of a bonded cluster of 5,000 particles is computed. The sample has been created via the weak jamming of an initial gas of particles (to a packing density of 0.5). This green sample has then sintered using Pan_Bouvard model up to a sintered density of 0.76 (see section 2.16 for sintering exemple). Bonds were defined as explained in section 2.8[1]. Periodic boundary conditions are used all along the sample generation. To obtain both Young's Modulus and Poisson ratio, a confined uniaxial test in the z direction is performed, with full periodic boundary conditions. The model with **interactions** is used: `large_bonds_full+geom`. Here the `geom` keyword is used leading to a geometrical overlap for the definition of the bond radius:

$$a_b^2 = 2R^*\delta_b \tag{31}$$

where R^* is the equivalent radius for two particles of radii R_1 and R_2 ($R^* = \frac{R_1 R_2}{R_1+R_2}$) and δ_b is the geometrical indentation between the two particles. This definition of the bond radius fits better the experimental values reported for partially sintered alumina [7, 14]. A consequence of using the `geom` option is that the density considering this bond model is lower that the "sintering density" using Coble option and is thus calculated when executing `dp3D` (here geometric density = 0.7178).

Hooke's law can be used to compute Young's modulus and Poisson's ratio :

$$\tilde{\nu} = \frac{\overline{\sigma}_{yy}}{\overline{\sigma}_{yy} + \overline{\sigma}_{zz}}, \qquad \tilde{E} = \frac{1}{\overline{\varepsilon}_{zz}}(\overline{\sigma}_{zz} - 2\tilde{\nu}\overline{\sigma}_{yy}) \tag{32}$$

The results obtained with and without interactions (with the option `psi=0.0`) are as following:

	`large_bonds_full+geom`	`large_bonds_full+geom+psi=0.0`
E GPa)	129.8	138.0
ν	0.173	0.122

The model with **interaction** is more involved numerically (and CPU intensive) and less stable. Thus, caution must be exercised when using it.

[1] `cdp3D -bonds`

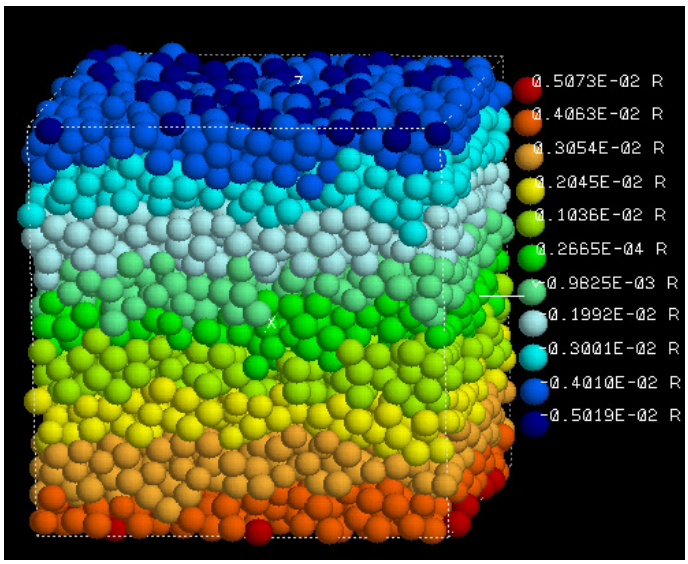


Figure 14: z displacement contour normalized by the mean radius (`vdp3D -dispz`).

```
###################################################################
#                   simulation conditions                         #
###################################################################
# coordinate file:
random_5k_D050_D76_bond
-------------------------------------------------------------------
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
-------------------------------------------------------------------
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
none
-------------------------------------------------------------------
###################################################################
#                            models                               #
###################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity
bonds
-------------------------------------------------------------------
# bond key words:
# large_bonds_full large_bonds small_bonds geom toughness impinge
# iso_bonds= psi_bar= strength_deviation=
large_bonds_full
geom
-------------------------------------------------------------------
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
-------------------------------------------------------------------
# adhesion model:
# DMT JKR
DMT
-------------------------------------------------------------------
###################################################################
#                           outputs                               #
###################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
epsilon>=0.5000E-03
-------------------------------------------------------------------
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep=
epsilon=0.1000E-04
-------------------------------------------------------------------
# writing output files:
# density= epsilon= pressure= aoR= time= timestep=
epsilon=0.1000E-05
-------------------------------------------------------------------
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
-------------------------------------------------------------------
###################################################################
#                           loadings                              #
###################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
-------------------------------------------------------------------
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure
epsxdot=0.0000E+00
epsydot=0.0000E+00
epszdot=-0.5000E-05
-------------------------------------------------------------------
###################################################################
#                           materials                             #
###################################################################
# elastic parameters (Pa for stress):
# E(1)= poisson(1)= E(2)= poisson(2)=
E(1)=0.4000E+12
poisson(1)=0.2000E+00
-------------------------------------------------------------------
# plastic parameters (Pa for stress):
# sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.8000E+13
Mstrain=0.0000E+00
Nvisco=0.0000E+00
-------------------------------------------------------------------
# friction parameters:
# frict(1,1)= frict(1,2)= frict(2,2)= frict(object)=
frict(1,1)=0.5000E+00
-------------------------------------------------------------------
# work of adhesion parameters (J.m-2):
# adhes(1,1)= adhes(1,2)= adhes(2,2)= adhes(object)=
adhes(1,1)=0.2000E+01
-------------------------------------------------------------------
# bond strength in tension (Pa):
# sig_N(1,1)= sig_N(1,2)= sig_N(2,2)= sig_N(object)=
sig_N(1,1)=0.4000E+13
-------------------------------------------------------------------
# bond strength in shear (Pa):
# sig_T(1,1)= sig_T(1,2)= sig_T(2,2)= sig_T(object)=
sig_T(1,1)=0.4000E+13
-------------------------------------------------------------------
# density (g.mm-3):
# ro(1)= ro(2)=
ro(1)=0.8706E-02
-------------------------------------------------------------------
###################################################################
#                           numerics                              #
###################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
-------------------------------------------------------------------
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
-------------------------------------------------------------------
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping=
safe_dt=0.1000E+00
upscale(1)=0.1000E+01
damping=0.5000E-01
-------------------------------------------------------------------
```

2.11 Toughness of a partially sintered ceramic

- directory : `toughness`; coordinate file: `plate_p75_crack`; histc file: `plate_p75_crack_histc`

The toughness of a partially sintered microstructure is obtained in this example by straining up to fracture a precracked sample. The sample is generated by packing and sintering with periodic boundary conditions. Bonds were installed between the sintered particles as in examples 2.9 and 2.10 [1]. The crack is introduced first by picking extrema particles [2] of the crack and then defining the crack [3] (see section 3.1.7) with a crack width of $4.0R$ (where R is the average particle radius). A `_histc` file is created after this stage, setting to `broken` the status of the bonds pertaining to the crack. In contrast to the other examples discussed in this guide, this is by necessity a large sample (10 000 particles). Thus, this simulation is CPU intensive. The keywords `kin_energy=1.E-08` and `break=0.0001`[4] are used to ensure a quasi-static simulation. When the normalized kinetic energy diverges from the value set by the user with the key word `kin_energy=1.E-08`, the strain rates (set in the section `loadings`) are increased or decreased to get back to this value. The maximum value the strain-rates may be multiplied or divided by, is given by the keyword `ctrl_fact`. Similarly, if the stress on a bond, σ_b is larger than $(1. - break)\sigma_c$, with σ_c the critical stress at which the bond breaks, the strain-rate is divided by `ctrl_fact`. This allows decreasing imposed strain-rates on the system before the breaking of a bond, which is an event that transforms elastic energy into kinetic energy.

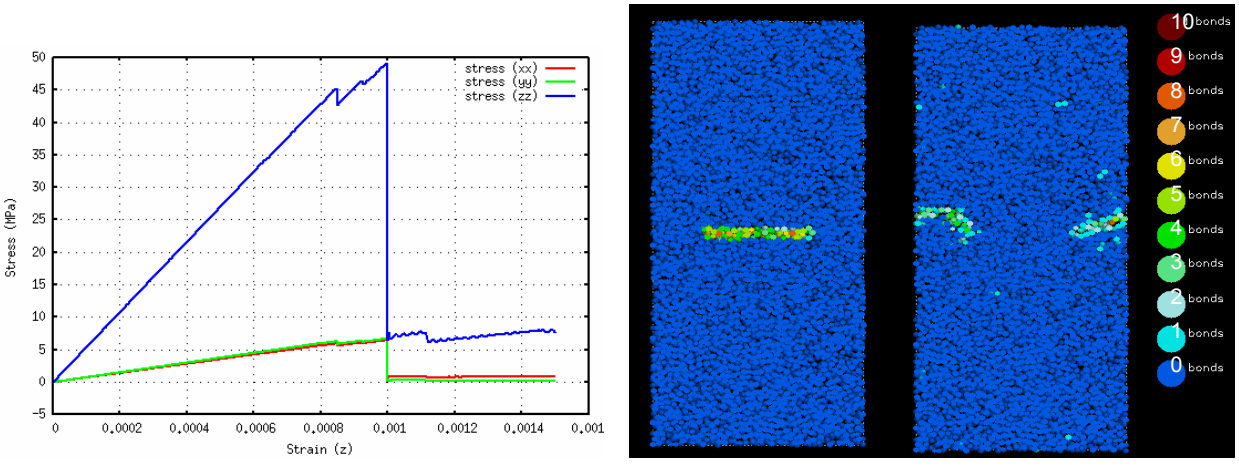
Periodic conditions are used in all three directions. Note that the bond toughness is given by the keyword `toughness` in J.m^{-2}. The expressions of the bond normal and tangential fracture stresses are given in [14].

The toughness can then be computed from the maximum stress and the half length of the crack a following linear elastic fracture mechanics:

$$K_{Ic} = \sigma_f Y \sqrt{\pi a} \tag{33}$$

The geometric factor Y depends on the configuration. For the collinear crack configuration used in this example $Y = \left[\frac{2W}{\pi a} \tan\left(\frac{\pi a}{2W}\right)\right]^{0.5}$, with W the half-width of the crack. Fig. 15b shows the fracture process[5].

[1] `cdp3D -bonds`
[2] `cdp3D -pick`
[3] `vdp3D -crack`
[4] The key word `break=` must be used with caution for very large samples as it may slow down the simulation dramatically. A 0.00001 value is a more reasonable choice for very large samples (> 200 000 particles).
[5] `vdp3D -fract_bonds`



(a) (b)

Figure 15: (a) Stress strain curves showing the fracture of a pre-cracked sample. (b)(left) Initial pre-cracked sample after the installation of the crack and (right) visualization of the fracture process . The colors show the number of bonds broken per particle during the simulation of tensile testing (i.e. the central pre-crack is not represented on the RHS image).

Contact laws used in this example:

The same contact laws are used here as in example `crushing` (section 2.9). However, two important differences must be noted concerning bond fracture when comparing the `input_dp3D` files. First the `beam` keyword is not used here. Hence the bond stress does not include the term due to resisting moments in Eqs. (29). Also, note that no normal or tangential strength is given `input_dp3D`. Instead, because the keyword `toughness` is used, the value of the toughness (`toughness(1,1)` in `input_dp3D`) of the bond is directly used to compute the critical stress (in tension and shear) at which the bond fails. Thus, The bond fractures when one of the following conditions is fulfilled:

$$\sigma_N = \frac{N^{bond}}{\pi a_b^2} > 2\sqrt{\frac{\Gamma}{\pi a_b}\frac{E}{1-\nu^2}f_N} \qquad \sigma_T = \frac{T^{bond}}{\pi a_b^2} > 2\sqrt{\frac{2}{\pi}}\sqrt{\frac{\Gamma}{\pi a_b}\frac{2E}{(1+\nu)(2-\nu)}f_T} \tag{34}$$

where Γ is given by `toughness(1,1)` in `input_dp3D`.

```
#########################################################################
#                      simulation conditions                            #
#########################################################################
# coordinate file:
plate_p75_crack
_____________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
_____________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
none
_____________________________________________________________________
#########################################################################
#                            models                                     #
#########################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity
bonds
_____________________________________________________________________
# bond key words:
# large_bonds_full large_bonds small_bonds geom toughness impinge
# iso_bonds= psi_bar= strength_deviation=
large_bonds_full
geom
toughness
psi_bar=0.
_____________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
_____________________________________________________________________
# adhesion model:
# DMT JKR
DMT
_____________________________________________________________________
#########################################################################
#                            outputs                                    #
#########################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
epsilon>=0.1500E-02
fracture=0.05
_____________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep= none end
epsilon=0.5000E-04
_____________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep= none end
epsilon=0.1000E-05
_____________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
_____________________________________________________________________
#########################################################################
#                           loadings                                    #
#########################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
_____________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=0.0000E+00
epsydot=0.0000E+00
epszdot=0.5000E-06
_____________________________________________________________________
#########################################################################
#                           materials                                   #
#########################################################################
# elastic parameters (Pa for stress):
# E(1)= poisson(1)= E(2)= poisson(2)=
E(1)=0.2000E+12
poisson(1)=0.2000E+00
_____________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.8000E+13
Mstrain=0.0000E+00
Nvisco=0.0000E+00
_____________________________________________________________________
# friction parameters:
# frict(1,1)= frict(1,2)= frict(2,2)= frict(object)=
frict(1,1)=0.5000E+00
_____________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(1,1)= adhes(1,2)= adhes(2,2)= adhes(object)=
adhes(1,1)=1.
_____________________________________________________________________
# bond toughness parameters (J.m-2):
# toughness(1,1)= toughness(1,2)= toughness(2,2)= toughness(object)=
toughness(1,1)=2.
_____________________________________________________________________
# density (g.mm-3):
# ro(1)= ro(2)=
ro(1)=0.8706E-02
_____________________________________________________________________
#########################################################################
#                           numerics                                    #
#########################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
_____________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
kin_energy=1.E-08
break=0.001
ctrl_fact=0.1000E+02
_____________________________________________________________________
```

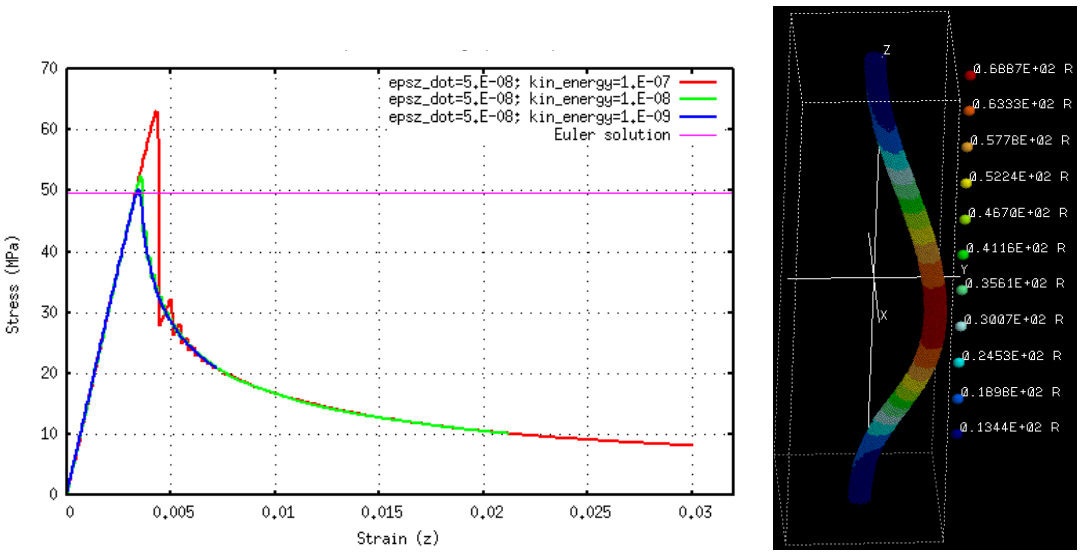
2.12 Buckling of a bar

- directory : `buckling`; coordinate file: `bar`

In granular-oriented applications described in the preceding sections, each modelled particle represents a clear physical entity. Another possible application for the `dp3D` code is the modelling of continuum mechanics with bonded particles. In this case, each modelled particle does not represent a physical entity. Instead, it is a set of microproperties that enables the continuum to be modelled. Interactions laws must be implemented and adjusted to correctly represent the material behaviour. To use this method, the keyword `stiffness` must be invoked in the bond key words section of the `input_dp3D` file. This is because in that case the contact interaction are not defined by elastic constants as for bonds but by normalized normal and tangential stiffness (K_N and K_T, see the subsection *Contact laws used in this example:* afterwards).

We take the example of the buckling of a bar. Particles are compacted and sintered to a relative density of 0.65. The preparation procedure is detailed in [27]. Particles are then bonded together and two planes with normal z are used to impose compression. Note that we use the possibility offered by `dp3D` to block the displacement of certain particles to impose fixed-fixed boundary conditions to the bar. This is indicated in the `input_dp3D` file which lists the particle numbers and displacement to block (as `deltax=nnnn,0.` where `nnnn` is the particle number, and 0. means that particle `nnn` has an imposed displacement of 0. in the x direction.). Bond strength has been set to a very large value. Note that when the `stiffness` keyword is on, the bond radius has no physical meaning. In all equation where it is used, its value is unity.

Fig. 16a shows the stress evolution for various conditions on the kinetic energy (see section 2.9 and `input_dp3D` file, the control of strain-rates to ensure quasi-static conditions may be defined with a target normalized kinetic energy). Three, decreasing, normalized kinetic energies have been tested. Fig. 16a demonstrates that the value of the buckling stress (in quasi-static mode) converges for very slow kinetic energy. The wrinkles after the peak stress are characteristic of a non quasi-static simulation. Also, the stress calculated by `dp3D` has to be corrected by a factor $4/\pi$ to take into account that the bar has a cylinder shape. Note that the stress predicted by Euler theory is in good accordance with the simulation in quasi-static conditions.



(a) (b)

Figure 16: Buckling of a bar. (a) Stress-strain curves for various strain rates. (b) Bar shape after buckling.

The imposed displacement possibility exemplified here with zero displacement, can be used as follows for more general cases:
`deltax=nnn,dispps`
where `nnn` is a particle number and `dispps` is a real means that in the direction x, particle `nnn` will be imposed the following displacement at each time step:
$\mathrm{d}x =$ `dispps` $\times \mathrm{d}t$
Rotations may also be imposed but only to a zero value (`deltar=nnn,0.`).

Contact laws used in this example:

The normal and tangential forces when the keyword `stiffness` is invoked in `input_dp3D` are given by:

$$N = 2K_N R^* u_n \quad ; \quad T = 2K_T R^* u_t \tag{35}$$

where R^* is the equivalent radius for two particles of radii R_1 and R_2 ($R^* = \frac{R_1 R_2}{R_1+R_2}$). $u_n = \delta_n - \delta_{n,0}$ is the normal and relative displacement between the two particles. Thus, K_N and K_T have the unit of a stress, which allows having macroscopic elastic properties that are not dependent on the size of the particles. Note that K_N and K_T must fulfill the conditions: $\frac{K_N}{K_T} \neq \frac{1}{2}; K_N \neq 0; K_T \neq 0$.

As for any bonds in `dp3D`, resisting moments are also included:

$$M_N = -4K_T \left(R^*\right)^3 \theta_N \tag{36}$$

and

$$M_T = -2K_N \left(R^*\right)^3 \theta_T \tag{37}$$

where θ_N and θ_T represent the accumulated relative rotation along the normal and tangential axis of the bond, respectively.

```
################################################################################
#                       simulation conditions                                  #
################################################################################
# coordinate file:
bar
________________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
________________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity thermal
none
________________________________________________________________________________
################################################################################
#                                 models                                       #
################################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity linear_elasticity
bonds
________________________________________________________________________________
# bond key words:
# large_bonds_full large_bonds geom toughness impinge only_bonds
# clump_cluster beam stiffness  Rankine iso_bonds= psi_bar=
# unload_stiff_ratio= plast_stiff_ratio= strength_deviation=
stiffness
________________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
________________________________________________________________________________
# adhesion model:
# DMT JKR
DMT
________________________________________________________________________________
################################################################################
#                                 outputs                                      #
################################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= sigxx>= sigxx<= sigyy>= sigyy<= sigzz>= sigzz<= fracture=
epsilon>=.03
________________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
epsilon=0.001
________________________________________________________________________________
# writing output files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
epsilon=0.0001
________________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
end
________________________________________________________________________________
################################################################################
#                                 loadings                                     #
################################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
0x0y0z
________________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=0.0000E+00
epsydot=0.0000E+00
epszdot=-0.5000E-07
________________________________________________________________________________
# imposed disp. (i,m/s), rotation (i,0.), force (i,N/s) on part i:
# deltax= deltay= deltaz= deltar= deltaFx= deltaFy= deltaFz=
deltax=544,0.
deltax=608,0.
...
deltax=16443,0.
deltay=544,0.
deltay=608,0.
...
deltay=16361,0.
deltay=16443,0.
________________________________________________________________________________
################################################################################
#                                 materials (from 0,1,2, ... to 9)             #
################################################################################
# bond stiffnesses (tension and shear) (Pa):
# K_N(1)= K_N(2)= K_T(1)= K_T(2)=
K_N(1)=0.25E+11
K_T(1)=0.2E+11
K_N(0)=0.25E+12
K_T(0)=0.2E+12
________________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(0)=0.8000E+33
sigy(1)=0.8000E+33
Mstrain=0.0000E+00
Nvisco=0.0000E+00
________________________________________________________________________________
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.5000E+00
________________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.
________________________________________________________________________________
# bond strength in tension (Pa):
# sig_N(0,0)= sig_N(0,1)= sig_N(1,1)= sig_N(1,2)= sig_N(2,2)=
sig_N(1,1)=0.4000E+33
sig_N(0,1)=0.4000E+33
________________________________________________________________________________
# bond strength in shear (Pa):
# sig_T(0,0)= sig_T(0,1)= sig_T(1,1)= sig_T(1,2)= sig_T(2,2)=
sig_T(1,1)=0.4000E+33
sig_T(0,1)=0.4000E+33
________________________________________________________________________________
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.8706E-02
________________________________________________________________________________
################################################################################
#                                 numerics                                     #
################################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
________________________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
kin_energy=1.E-08
ctrl_fact=100.
________________________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
```

2.13 Bending of a fiber

- directory : `fiber`; coordinate file: `fiber20`

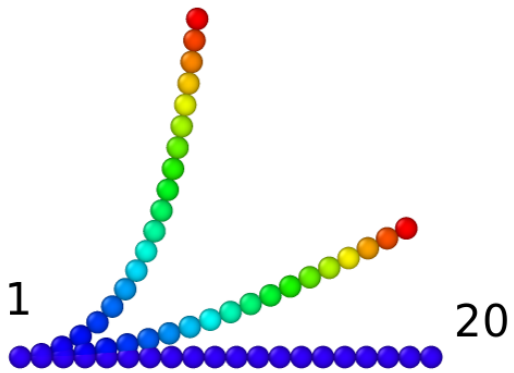
In this example, we continue to use the `stiffness` keyword to model a fiber. In contrast to the preceding example, the fiber is simply a necklace of 20 bonded spheres (Fig. 17a). This is a much simpler example with the advantage of a very small CPU time but with a simplistic description of the fiber surface.

Particle `1` must be pinned to impose the bending at its free end. This is done as in the example before using: `deltax=1.,0 ... deltar=1,0.` to impose zero displacement in x, y, z directions and to impose zero rotation (`deltar=1,0.`). Additionally, we need to impose a vertical force F_z on particle `20` at the free end of the fiber. This is done by stating in `input_dp3D`:
`deltaFz=20,1.25E-09`
which states that at each time step a force increment:
$\mathrm{d}F_z = 1.25\mathrm{E}^{-09} \times \mathrm{d}t$
is imposed on particle `20` in the z direction. Note here that the kinetics of the loading are NOT imposed by any strain-rate (the `epsxdot, epsydot, epszdot` are all zero in input_dp3D). The objects do not play any role (they are only there to bound the simulation box, they could be removed). The kinetics are solely imposed by the rate of increase of the vertical force F_z. Note that the time step dt is constant (`fixed_dt=0.1`. Its value has been evaluated by running `dp3D` with the standard `safe_dt` option first.

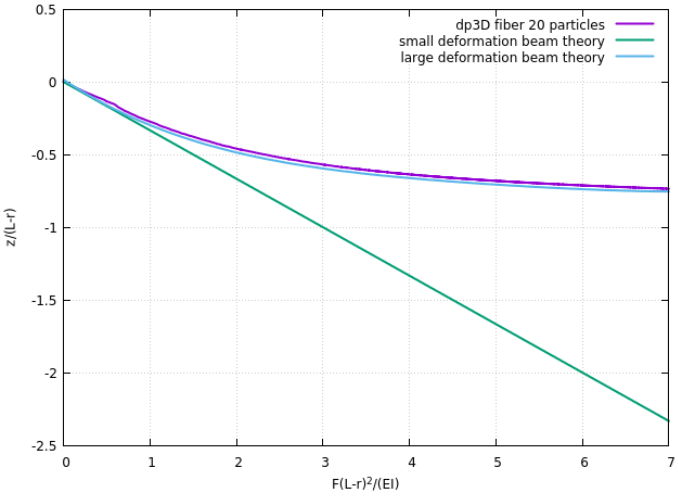
There are several ways to obtain the vertical deflection of the free end of the fiber, which is of interest here. The simplest is to use the `spy` possibility. Thus, particle `20` has been marked as spy [1]. This generates file `_spy0000020` which informs on the displacements (and many other things) of particle `20`.

Fig. 17b shows the result of the test in terms of the vertical deflection of the fiber at its free end normalized by $(L - r)$ where L is the total length of the fiber and r is the radius of the spheres. The force F is normalized by $\frac{EI}{(L-r)^2}$ where E is the Young's modulus of the fiber and I its moment of inertia ($I = \frac{\pi r^4}{4}$).

[1] `cdp3D -spy fiber20 20`



(a)



(b)

Figure 17: (a) Gradual bending of a fiber made of 20 particles (b) Vertical normalized deflection of the fiber at its free end vs the normalized load.

Contact laws used in this example:

The contact laws are the same as in the preceding example (Eqs. (35), (36)). However, it should be noted that in contrast to the preceding example, spheres indent each other only by a very small value.

Exercise:

- *Edit the* `input_dp3D` *file to increase the rate of change of the force* F_z *by a factor 10. Replot Fig. 17b using the* `plot_fiber` *gnuplot file in the directory (and modifying the* `deltaFz` *accordingly in this file). Is the fiber deflection still following the large deformation theory? Is the force equilibrium (quasi-static condition) still enforced?*

- *Use the same* `fiber20` *coordinate file to construct a simple tension (or compression) test of the fiber to obtain the Young's modulus of the fiber. This may be obtained by replacing the* `deltaF` *keyword in the* `input_dp3D` *file to impose a displacement to the free end of the fiber. You will need to tag the last two particles of the fiber (or actually any pair of particles in the fiber) to obtain the axial stress. Show that the macroscopic Young's modulus of the fiber discretized as in Fig. 17 is simply:*

$$E = \frac{2K\_N}{\pi} \tag{38}$$

where K_N *is the normal stiffness in the* `input_dp3D` *file.*

```
########################################################################
#                    simulation conditions                             #
########################################################################
# coordinate file:
fiber20
________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity
none
________________________________________________________________________
########################################################################
#                              models                                  #
########################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity linear_elasticity
bonds
________________________________________________________________________
# bond key words:
# large_bonds_full large_bonds geom toughness impinge only_bonds
# clump_cluster beam stiffness iso_bonds= psi_bar= strength_deviation=
# unload_stiff_ratio= plast_stiff_ratio= Rankine
stiffness
Rankine
________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
________________________________________________________________________
# adhesion model:
# DMT JKR
DMT
________________________________________________________________________
########################################################################
#                             outputs                                  #
########################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
timestep>=1.2E+07
________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep= none end
timestep=1.2E+05
________________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep= none end
timestep=1.2E+05
________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
________________________________________________________________________
########################################################################
#                            loadings                                  #
########################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
0x0y0z
________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=0.0000E+00
epsydot=0.0000E+00
epszdot=0.0000E+00
________________________________________________________________________
# imposed disp. (i,m/s), rotation (i,0.), force (i,N/s) on part i:
# deltax= deltay= deltaz= deltar= deltaFx= deltaFy= deltaFz=
deltax=1,0.
deltay=1,0.
deltaz=1,0.
deltar=1,0.
deltaFz=20,1.25E-09
________________________________________________________________________
########################################################################
#                    materials (from 0,1,2, ... to 9)                  #
########################################################################
# bond stiffnesses (tension and shear) (Pa):
# K_N(1)= K_N(2)= K_T(1)= K_T(2)=
K_N(0)=3.14160E+09
K_T(0)=3.14160E+09
K_N(2)=3.14160E+09
K_T(2)=3.14160E+08
________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(0)=0.8000E+33
sigy(2)=0.8000E+33
Mstrain=0.0000E+00
Nvisco=0.0000E+00
________________________________________________________________________
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(0,2)=0.000E+00
frict(2,2)=0.0000E+00
________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(0,2)=0.
adhes(2,2)=0.
________________________________________________________________________
# bond strength in tension (Pa):
# sig_N(0,0)= sig_N(0,1)= sig_N(1,1)= sig_N(1,2)= sig_N(2,2)=
sig_N(0,2)=20.0E+30
sig_N(2,2)=20.0E+30
________________________________________________________________________
# bond strength in shear (Pa):
# sig_T(0,0)= sig_T(0,1)= sig_T(1,1)= sig_T(1,2)= sig_T(2,2)=
sig_T(0,2)=20.0E+30
sig_T(2,2)=20.0E+30
________________________________________________________________________
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(2)=1.0E-03
________________________________________________________________________
########################################################################
#                             numerics                                 #
########################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
0x0y0z
________________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
________________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
fixed_dt=0.1
upscale(1)=0.1000E+01
damping=0.5000E-01
________________________________________________________________________
```

2.14 Thermal conduction and thermal expansion of a plate

- directory : `thermal_shock`; coordinate file: `plate_T250C`

In this example, we model a thermal shock experienced by a plate (here alumina) initially at a given uniform temperature which is quenched in a cold bath (20 deg C). Heat conduction is handle by setting the general key word `thermal` in `input_dp3D`. Each particle is given an initial temperature in the input coordinate file `plate`. All particles have initially the same temperature (250 deg C, see column `Temp`). As in the preceding example, particles are bonded together to model a continuous plate. The preparation procedure is detailed in [27].

Here, the external loading conditions are set by using marked particles (not by imposing a strain-rate). Marked particles are particles which name (here `cu`) is followed by a `+` as `cu+`. Here, we have chosen to mark particles that are on the surface of the plate to impose a thermal shock. This can be done after using the `cdp3D -pick` command for example. A surface temperature or a fixed temperature can be invoked for these marked particles by setting the key word `surface_Temp=20.` or `fixed_Temp=20.` in `input_dp3D`.

In `dp3D`, heat conduction for particle i with temperatures T_i leads to a temperature increment during timestep dt:

$$\mathrm{d}T_i = \delta D \sum_{contacts} \frac{4{R^*}^2}{d_{ij}R_i^3}\,(T_j - T_i)\,\mathrm{d}t \tag{39}$$

where D is the thermal diffusivity, d_{ij} is the center-center distance between particles i and j and δ is a fitting parameter (set to 0.385). The summation is made on all contacts j around particle i. The key word `surface_Temp=20.` make the marked particles to have an additional term in Eq. (39):

$$\mathrm{d}T_i^s = \delta D \frac{1}{R_i^2}\max\left(\bar{Z} - Z_i, 0.\right)(\mathrm{surface\_Temp} - T_i)\,\mathrm{d}t \tag{40}$$

where $\bar{Z}$ is the average coordination number in the packing and Z_i is the coordination number of particle i. This term accounts very simply of the free surface of particles.

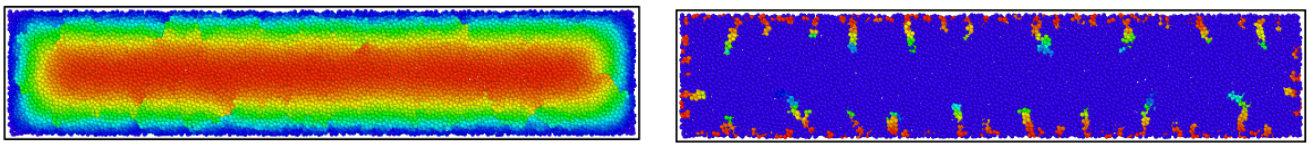
The necessary material parameters for the thermal mode are the thermal diffusivity (T_diffusivity) and the thermal expansion (T_expansion). Note that the linear thermal expansion parameter has been multiplied by a correction factor (2.75) to get back the correct Griffith length l_0 that describes the severity of the thermal shock:

$$l_0 = \left(\frac{K_{Ic}}{E\beta\Delta T}\right)^2 \tag{41}$$

with K_{Ic} the toughness, β the linear thermal expansion coefficient and ΔT the difference between the initial temperature of the slab (here 250°C) and the temperature of the bath (20°C).

Figs. 18 show the thermal field and the final crack system in the plate[1]. Fig. 18a indicates that when a bond is broken and does transmit any force, it does not transmit heat either (not counted anymore in the summation in Eq. (39)).

[1] commands `ovdp3D -Temp` and `ovdp3D -histo_bonds`



(a) (b)

Figure 18: Thermal shock of a plate. (a) Temperature field at time=0.27 sec. (b) Plate after thermal shock showing cracks. Colors indicate the timestep at which a bond has broken.

Contact laws used in this example:

The keyword `stiffness` is invoked in `input_dp3D`. Thus, the normal and tangential forces and the resisting moments are given by Eqs (35), (36) and (37) (see section 2.12). Concerning bond fracture, the `Rankine` keyword is invoked. This means that it is a combination of the normal and tangential stresses at the bond which is used to write the fracture criterion (Rankine's equivalent stress):

$$\sigma_{b,R} = \frac{1}{2}\left(\sigma_{b,N} + \sqrt{\sigma_{b,N}^2 + 4\sigma_{b,T}^2}\right) \tag{42}$$

where $\sigma_{b,N} = \frac{N}{4\pi R^{*2}}$ and $\sigma_{b,T} = \frac{T}{4\pi R^{*2}}$ are the normal and the tangential stresses at the bond scale. The bond breaks when the Rankine's equivalent stress $\sigma_{b,R}$ reaches a critical stress given by `sig_N` in `input_dp3D`. Note that when the Rankine criterion is used the `sig_T` has no use.

```
################################################################################
#                        simulation conditions                                 #
################################################################################
# coordinate file:
plate_T250C
________________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
elasto_plasticity
________________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity thermal
thermal
________________________________________________________________________________
################################################################################
#                                 models                                       #
################################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity linear_elasticity
bonds
________________________________________________________________________________
# bond key words:
# large_bonds_full large_bonds geom toughness impinge only_bonds
# clump_cluster beam stiffness  Rankine iso_bonds= psi_bar=
# unload_stiff_ratio= plast_stiff_ratio= strength_deviation=
stiffness
iso_bonds=1.
Rankine
________________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
________________________________________________________________________________
# adhesion model:
# DMT JKR
DMT
________________________________________________________________________________
################################################################################
#                                outputs                                       #
################################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
time>=0.4
________________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep= none end
time=0.02
________________________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep= none end
timestep=1000
________________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
________________________________________________________________________________
################################################################################
#                                loadings                                      #
################################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
0x0y0z
________________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=0.0000E+00
epsydot=0.0000E+00
epszdot=0.0000E+00
________________________________________________________________________________
# Properties of marked particles (Temp in deg C):
# fixed_Temp= surface_Temp=
surface_Temp=20.0000E+00
________________________________________________________________________________
################################################################################
#                          materials (from 0,1,2, ... to 9)                    #
################################################################################
# bond stiffnesses (tension and shear) (Pa):
# K_N(1)= K_N(2)= K_T(1)= K_T(2)=
K_N(0)=1810.E+09
K_T(0)=334.E+09
K_N(1)=1810.E+09
K_T(1)=334.E+09
________________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(0)=50000.E+06
sigy(1)=50000.E+06
Mstrain=0.0000E+00
Nvisco=0.0000E+00
________________________________________________________________________________
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.5000E+00
________________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.
________________________________________________________________________________
# bond strength in tension (Pa):
# sig_N(0,0)= sig_N(0,1)= sig_N(1,1)= sig_N(1,2)= sig_N(2,2)=
sig_N(1,1)=1189.E+06
sig_N(0,1)=1.E+33
________________________________________________________________________________
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.8706E-02
________________________________________________________________________________
# Thermal expansion:
# T_expansion(1)= T_expansion(2)=
T_expansion(1)=31.11E-06
________________________________________________________________________________
# Thermal diffusivity (m2.s-1):
# T_diffusivity(1)= T_diffusivity(2)=
T_diffusivity(1)=1.2E-05
________________________________________________________________________________
################################################################################
#                                numerics                                      #
################################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
________________________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
________________________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
safe_dt=0.100E+00
upscale(1)=0.1000E+01
damping=0.5000E-01
________________________________________________________________________________
```

2.15 Use of clumps instead of clusters

- directory : `clumps`; coordinate file: `file_init`

In this example, we explain how to use `clumps` to model non-deformable entities. `clumps` are defined like `clusters` by a set of bonds between some particles. However, they differ from clusters in that the interaction *inside* the `clumps` are not calculated. This is illustrated in Fig. 19. This means that the total number of contacts is much smaller in general in problems that use clumps as compared to those that use clusters. This translates into faster computation. This comes at the price of neglecting strains in the clump (it can only be indented by particles that do not pertain to the clump) and of impeding any fracture in the clump.

The `clumps` option still belongs to the `bonds` models as `bonds` need to be declared to define the `clumps`. Thus the `bonds` keyword must be invoked in `input_dp3D` and the `clump_cluster` keyword must be invoked in the `bond key words:` section.

When using the `clump_cluster` option, the coordinate file `filename` must be accompanied by a file `filename_clumps` (here `file_init_clumps`). This file has generally been generated initially when generating the gas[1]. Alternatively, the `filename_clumps` file may be generated afterwards through `cdp3D`[2], providing bonds have been defined.

In this example, several clumps (small spheres and platelets) and isolated particles are mixed together and compacted using the jamming mode. This illustrates a possible use of clumps: speeding the jamming of packings. You may test this by replacing in this example the `clump_cluster` key word by the key words:
`large_bonds_full`
`beam`
`psi_bar=0.`
The relative gain in CPU time is of the order of 100%.

[1] `cdp3D -gas`
[2] `cdp3D -clumps`

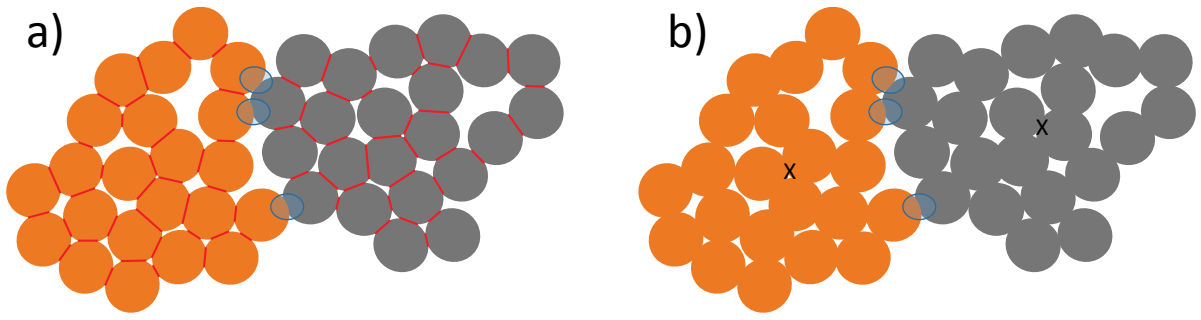


Figure 19: Schematic of: a) Two clusters that interact through three (circled) contacts. Interactions are calculated inside the clusters (red bonds). b) Two clumps that interact through three (circled) contacts. Interactions are *not* calculated inside the clumps. Crosses indicate the center of gravity of clumps where the total force (and moment) is calculated.

```
################################################################################
#                  simulation conditions                       #
################################################################################
# coordinate file:
file_init
_____________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
jamming
_____________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity thermal
none
_____________________________________________________________________
################################################################################
#                        models                                #
################################################################################
# elasto_plasticiy and jamming key words:
# none bonds no_elasticity linear_elasticity
bonds
_____________________________________________________________________
# bond key words:
# large_bonds_full large_bonds geom toughness impinge only_bonds
# clump_cluster beam stiffness  Rankine iso_bonds= psi_bar=
# unload_stiff_ratio= plast_stiff_ratio= strength_deviation=
clump_cluster
_____________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Hertz_Mindlin
_____________________________________________________________________
# adhesion model:
# DMT JKR
DMT
_____________________________________________________________________
################################################################################
#                        outputs                               #
################################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
epsvdot<=1.E-07
_____________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep= none end
density=0.01
_____________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep= none end
density=0.002
_____________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
_____________________________________________________________________
################################################################################
#                        loadings                              #
################################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
_____________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
epsxdot=-1.E-04
epsydot=-1.E-04
epszdot=-1.E-04
pressure=0.2E+06
_____________________________________________________________________
################################################################################
#                        materials (from 0,1,2, ... to 9)      #
################################################################################
# elastic parameters (Pa for stress):
# E(0)= poisson(0)= E(1)= poisson(1)= delta_c(0)= fact_mult(0)=
E(0)=10.E+09
poisson(0)=0.3000E+00
E(1)=10.E+09
poisson(1)=0.3000E+00
E(2)=10.E+09
poisson(2)=0.3000E+00
_____________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.3000E+19
Mstrain=0.0000E+00
Nvisco=0.0000E+00
_____________________________________________________________________
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.0
_____________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.
_____________________________________________________________________
# bond strength in tension (Pa):
# sig_N(0,0)= sig_N(0,1)= sig_N(1,1)= sig_N(1,2)= sig_N(2,2)=
sig_N(1,1)=0.2000E+33
_____________________________________________________________________
# bond strength in shear (Pa):
# sig_T(0,0)= sig_T(0,1)= sig_T(1,1)= sig_T(1,2)= sig_T(2,2)=
sig_T(1,1)=0.2000E+33
_____________________________________________________________________
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.7890E-02
_____________________________________________________________________
################################################################################
#                        numerics                              #
################################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
_____________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
_____________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
safe_dt=0.1
upscale(1)=0.1000E+01
damping=0.1000
_____________________________________________________________________
```

2.16 Sintering of a compact

- directory : `sinter_velocity`; coordinate file: `p400p6365R6p75`

The sintering mode is invoked in `dp3D` by setting the `mode key word` to `sintering` in `input_dp3D`. A model must also be chosen. There are currently three force models available for the sintering normal contact law (see the *Contact laws used in this example:* section afterwards): `Parhami_Mc_Meeking, Bouvard_Pan and viscous`.

Because sintering involves specific flow mechanisms that are quire different from elasticity or plasticity, a whole new set of material parameters are needed. The following table summarizes the material parameters that must be given for each type of model in `input_dp3D` to define sintering parameters. These are parameters that are sufficient when no particle coarsening is invoked.

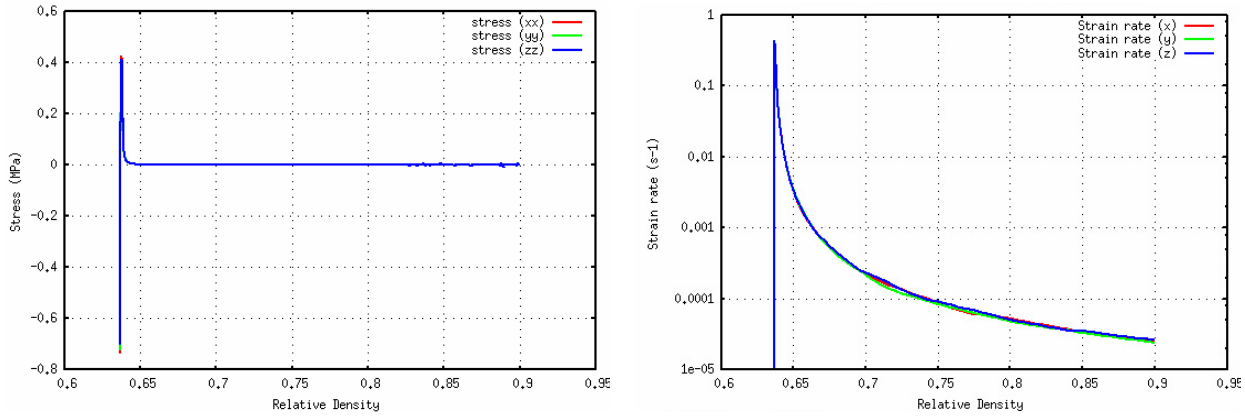
model	material parameter	eq. number	`input_dp3D` name	units
Parhami_Mc_Meeking, Bouvard_Pan, viscous	γ_s	(43,44,45)	`gamma_s`	J.m^{-2}
Bouvard_Pan	α	(44)	`alpha`	none
Parhami_Mc_Meeking, Bouvard_Pan	$\delta_b D_{0b}$	(46,47)	`DeltabD0b`	m^3.s^{-1}
Parhami_Mc_Meeking, Bouvard_Pan, viscous	Q_b, Q	(47,48)	`Qb`	KJ.mol^{-1}
Parhami_Mc_Meeking, Bouvard_Pan	μ	(50)	`frict`	none
viscous	η_0	(48,51)	`eta_0`	Pa.s
Parhami_Mc_Meeking, Bouvard_Pan, viscous	ψ	(49)	`chi`	$^\circ$
Parhami_Mc_Meeking, Bouvard_Pan	Ω	(47)	`omega`	m^3

In this example, only one material is used. The parameters are the `sintering material` (if two materials are defined, one may be a non-sintering inclusion), the `surface energy`, the `diffusion parameter`, the `activation energy`, the `dihedral angle`, the `atomic volume`, and the `coarsening parameter` which tells if coarsening is on or not (here set to zero, no coarsening). `Temperature` must also be defined (`loading conditions`).

In the sintering mode, the strain rate imposed on the simulation box is not given by the user in `input_dp3D` file. Instead, the user imposes an external stress state in `input_dp3D` and `dp3D` computes at each time step the strain-rate which will make the principal stresses to tend towards the desired value. In this example, the three stresses are set to 0 to simulate standard free sintering[1].

An important action that must be conducted after a sintering simulation is to check that indeed the imposed macroscopic stress has been reached by the simulation. Fig. 20a shows how the stress should evolve if the simulation is correct[2] . At the initial time-step, because the velocity of particles is zero, there exist tensile forces between particles and a macroscopic tensile stress is exerted on the sample. Afterwards, `dp3D` attempts to reach the imposed stress but overshoot it in this exemple and it is only after a small amount of densification that stresses converge toward the correct value. Note also that for large density (above 0.85), the stresses start to oscillate again a bit. In the present exemple, this is not an issue. However in more involved problems, the macroscopic stress may diverge. In that case, the simulation should be discarded. Fig. 20b shows the strain-rate evolution as densification proceeds[3] .

[1] in fact, `dp3D` will use a very small pressure value calculated from the material parameters ($4.10^{-3}\gamma_s/R$)
[2] `ddp3D -dens -sig`
[3] `ddp3D -dens -epsdot`



(a) (b)

Figure 20: (a) Evolution of the macroscopic stress versus density for a free sintering simulation. (b)Densification rate vs density.

Contact laws used in this example:

Three models are available in the sintering mode. The `Parhami_Mc_Meeking model` [8, 9]:

$$N_s = \frac{\pi a_s^4}{8\Delta_b}\frac{\mathrm{d}\delta_n}{\mathrm{d}t} - \pi\gamma_s\left[4R\left(1-\cos\frac{\psi}{2}\right) + a_s\sin\frac{\psi}{2}\right] \tag{43}$$

the `Bouvard_Pan` model: [15, 28, 29]

$$N_s = \frac{\pi a_s^4}{8\Delta_b}\frac{\mathrm{d}\delta_n}{\mathrm{d}t} - \frac{\alpha}{4}\pi R\gamma_s \tag{44}$$

and a purely `viscous` model [30, 31]

$$N_s = \frac{3\pi}{2}\eta a_s\frac{\mathrm{d}\delta_n}{\mathrm{d}t} - 3\pi a_s\gamma_s \tag{45}$$

In these models, a_s is the sintering contact radius, and $\frac{\mathrm{d}\delta_n}{\mathrm{d}t}$ is the rate of approach between the two particles. The driving force for all these models is the surface energy γ_s. The Parhami_Mc_Meeking and Bouvard_Pan models have very similar forms and a common material parameter:

$$\Delta_b = \frac{\Omega}{kT}\delta_b D_b \tag{46}$$

with

$$D_b = D_{0b}\exp(-Q_b/RT) \tag{47}$$

the diffusion coefficient for vacancy transport in the grain boundary with thickness δ_b and activation energy Q_b, Ω is the atomic volume, and T the temperature.

The viscous model introduces directly a viscosity η which varies with temperature in a similar Arhenius manner:

$$\eta = \eta_0\exp(Q/RT) \tag{48}$$

In all these models the dihedral angle ψ defines the equilibrium size of the sintering contact, a_{eq}:

$$a_{eq} = R\sin\frac{\psi}{2} \tag{49}$$

When a_s reaches a_{eq}, the second term in Eq. (43), (44), (45) is set to zero and any further growth in contact size necessitates a compressive contact force. For example, if ψ is set to 60°, the sintering part of the force will be set to zero when the contact radius is half the particle radius.

These models also introduce shear force at the contact, T_s, which opposes the tangential component of the relative velocity at the contact, $\mathrm{d}\delta_t/\mathrm{d}t$. For the Parhami_Mc_Meeking and Bouvard_Pan models, its takes the form:

$$T_s = -\mu \frac{\pi a_s^2 R^2}{8\Delta_b} \frac{\mathrm{d}\delta_t}{\mathrm{d}t} \tag{50}$$

where μ is a viscous parameter with no dimension [32]. For the viscous model, T_s is written with the same temperature dependent viscosity η as in Eq.(45) [30,31]

$$T_s = -\eta \frac{\pi}{2} a_s \frac{\mathrm{d}\delta_t}{\mathrm{d}t} \tag{51}$$

```
################################################################
#                   simulation conditions                      #
################################################################
# coordinate file:
p400p6365R6p75
----------------------------------------------------------------
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
sintering
----------------------------------------------------------------
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity thermal
no_rotation
----------------------------------------------------------------
################################################################
#                          models                              #
################################################################
# sintering model:
# Parhami_Mc_Meeking Bouvard_Pan viscous
Parhami_Mc_Meeking
----------------------------------------------------------------
# friction model:
# Hertz_Mindlin Coulomb shear
Coulomb
----------------------------------------------------------------
# adhesion model:
# DMT JKR
JKR
----------------------------------------------------------------
################################################################
#                         outputs                              #
################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= sigxx>= sigxx<= sigyy>= sigyy<= sigzz>= sigzz<= fracture=
density>=0.9000E+00
----------------------------------------------------------------
# writing coordinate files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
density=0.1000E-01
----------------------------------------------------------------
# writing output files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
density=0.1000E-03
----------------------------------------------------------------
# writing contact history files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
end
----------------------------------------------------------------
################################################################
#                         loadings                             #
################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
----------------------------------------------------------------
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
sigxx=0.0
sigyy=0.0
sigzz=0.0
----------------------------------------------------------------
# temperatures(°C) and duration (s) for stage 1:
# none T_init= T_final= time=
T_init=0.1100E+04
T_final=0.1100E+04
time=0.
----------------------------------------------------------------
################################################################
#                   materials (from 0,1,2, ... to 9)           #
################################################################
# elastic parameters (Pa for stress):
# E(0)= poisson(0)= E(1)= poisson(1)= delta_c(0)= fact_mult(0)=
E(1)=0.1000E+12
poisson(1)=0.2200E+00
----------------------------------------------------------------
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.1000E+10
Mstrain=0.0000E+00
Nvisco=0.0000E+00
----------------------------------------------------------------
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.0000E+00
----------------------------------------------------------------
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.
----------------------------------------------------------------
# sintering material(s):
# 0 1 2 ... 9
1
----------------------------------------------------------------
# surface energy parameters for sintering (J.m-2):
# gamma_s(0,0)= gamma_s(0,1)= gamma_s(1,1)= gamma_s(1,2)= gamma_s(2,2)=
gamma_s(1,1)=0.1720E+01
----------------------------------------------------------------
# diffusion parameter (m3.s-1), activation energy (KJ/mol):
# DeltabD0b(0,0)= DeltabD0b(0,1)= DeltabD0b(1,1)=
# Qb(0,0)= Qb(0,1)= Qb(1,1)=
DeltabD0b(1,1)=0.5120E-14
Qb(1,1)=0.1050E+03
----------------------------------------------------------------
# dihedral angle (°):
# chi(0,0)= chi(0,1)= chi(1,1)= chi(1,2)= chi(2,2)=
chi(1,1)=0.1460E+03
----------------------------------------------------------------
# atom vol(m3):
# omega(0)= omega(1)=
omega(1)=0.1180E-28
----------------------------------------------------------------
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.8706E-02
----------------------------------------------------------------
################################################################
#                          numerics                            #
################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
----------------------------------------------------------------
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
safe_dt=0.1000E-01
upscale(1)=0.1000E+01
damping=0.0000E+00
----------------------------------------------------------------
```

2.17 Sintering of a compact with a temperature profile

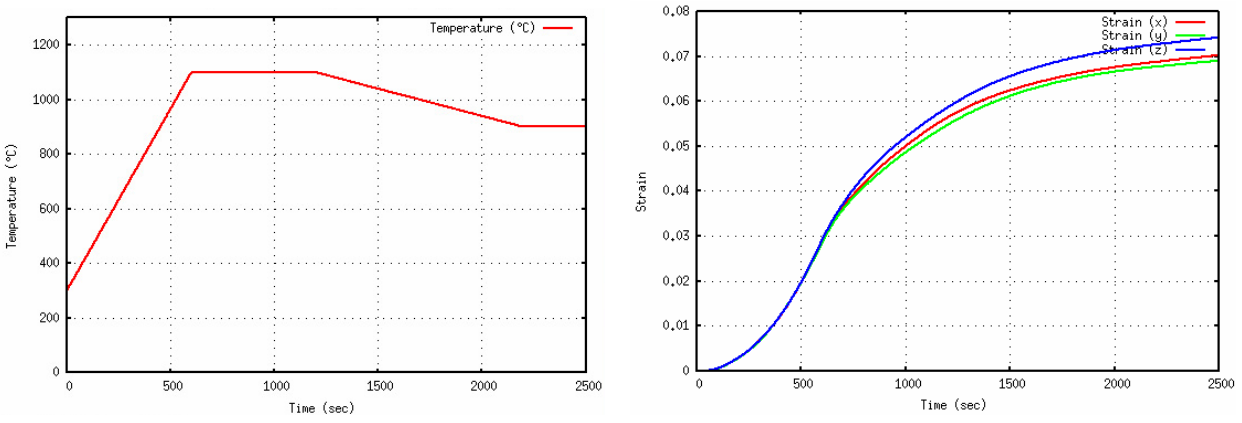
- directory : `sinter_ramp_temperature`; coordinate file: `p400p6365R6p75`

A temperature scheme may be given in the loadings key-words in `input_dp3D` to define a temperature ramp. The ramp is defined by its initial and final temperatures (`T_init, T_final`) and by the duration of this ramp (`time`). Note that it is not wise, from a numerical point of view, to start with too low a temperature since this will generate large viscoplastic forces to counter sintering forces. Here the lowest temperature is 300°C as compared to the highest sintering temperature of 1100°C.

This example also shows how to introduce tangential viscous forces at the contact when sintering. The value of the viscous parameter μ is set in the `frict` keyword in `input_dp3D`. A value of 0.001 leads to a standard shear resisting force. Note that for the time being, there is no resisting moment at the contact. Also, as in the previous example, rotations are not considered in the sintering mode. This is because we consider that finite size contacts grow very rapidly during sintering hindering rotations. In this example the `Bouvard_Pan` is invoked. For this model, an α parameter (named `alpha` in `input_dp3D`) must be provided as indicated by Eq. (44). Its value depends on the ratio between the grain-boundary and surface diffusion. A standard value for ceramics is 4.5.

As in other modes, it is possible to define a composite. This may be done in two ways. It is possible to define a matrix-inclusion mixture by defining only one sintering material [1] in the mixture [16]. The other one will behave elastoplastically as defined by the material parameters given for its family. It is also possible to define a mixture of two sintering materials with specific sintering material parameters for each family. In that case the two materials should be defined as sintering as in:

```
# sintering material(s):
# 1 2
1
2
```



(a) (b)

Figure 21: (a) Temperature profile imposed. (b) Strain evolution with time.

[1] by stating `# sintering material(s):1` in the materials

```
########################################################################
#                    simulation conditions                             #
########################################################################
# coordinate file:
p400p6365R6p75
________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
sintering
________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity thermal
no_rotation
________________________________________________________________________
########################################################################
#                              models                                  #
########################################################################
# sintering model:
# Parhami_Mc_Meeking Bouvard_Pan viscous
Bouvard_Pan
________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Coulomb
________________________________________________________________________
# adhesion model:
# DMT JKR
JKR
________________________________________________________________________
########################################################################
#                             outputs                                  #
########################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= sigxx>= sigxx<= sigyy>= sigyy<= sigzz>= sigzz<= fracture=
time>=2500.
________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
time=250.
________________________________________________________________________
# writing output files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
time=1.
________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
none
________________________________________________________________________
########################################################################
#                             loadings                                 #
########################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= none
sigxx=0.0
sigyy=0.0
sigzz=0.0
________________________________________________________________________
# temperatures(□C) and duration (s) for stage 1:
# none T_init= T_final= time=
T_init=300.
T_final=1100.
time=600.
________________________________________________________________________
# temperatures(□C) and duration (s) for stage 2:
# none T_init= T_final= time=
T_init=1100.
T_final=1100.
time=600.
________________________________________________________________________
# temperatures(□C) and duration (s) for stage 3:
# none T_init= T_final= time=
T_init=1100.
T_final=900.
time=1000.
________________________________________________________________________
########################################################################
#                          materials (from 0,1,2, ... to 9)            #
########################################################################
# elastic parameters (Pa for stress):
# E(0)= poisson(0)= E(1)= poisson(1)= delta_c(0)= fact_mult(0)=
E(1)=0.1000E+12
poisson(1)=0.2200E+00
________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.1000E+10
Mstrain=0.0000E+00
Nvisco=0.0000E+00
________________________________________________________________________
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.001
________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.1720E+01
________________________________________________________________________
# sintering material(s):
# 0 1 2 ... 9
1
________________________________________________________________________
# surface energy parameters for sintering (J.m-2):
# gamma_s(0,0)= gamma_s(0,1)= gamma_s(1,1)= gamma_s(1,2)= gamma_s(2,2)=
gamma_s(1,1)=0.1720E+01
________________________________________________________________________
# diffusion parameter (m3.s-1), activation energy (KJ/mol):
# DeltabD0b(0,0)= DeltabD0b(0,1)= DeltabD0b(1,1)=
# Qb(0,0)= Qb(0,1)= Qb(1,1)=
DeltabD0b(1,1)=0.5120E-14
Qb(1,1)=0.1050E+03
________________________________________________________________________
# dihedral angle (□):
# chi(0,0)= chi(0,1)= chi(1,1)= chi(1,2)= chi(2,2)=
chi(1,1)=0.1460E+03
________________________________________________________________________
# atom vol(m3):
# omega(0)= omega(1)=
omega(1)=0.1180E-28
________________________________________________________________________
# alpha parameter for sintering:
# alpha(0,0)= alpha(0,1)= alpha(1,1)= alpha(1,2)= alpha(2,2)=
alpha(1,1)=4.5
________________________________________________________________________
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.8706E-02
________________________________________________________________________
########################################################################
#                             numerics                                 #
```

2.18 Sintering of a compact including grain growth

- directory : `sinter_grain_growth`; coordinate file: `p4000dens59R02ln035`

The grain growth model is invoked in the sintering mode by adding directly the values of the grain growth parameters in `input_dp3D`. If surface diffusion is not introduced in `input_dp3D`, coarsening will not be taken into account. The following table summarizes the material parameters that must be given for grain growth. For the time being, the coarsening model is only compatible with the Bouvard_Pan (Eq.(44)) sintering model and for a single material.

Name	material parameter	eq. number	`input_dp3D` name	units
Pre-exponential factor of surface diffusion	D_{0S}	(55)	`D0s`	$m^{-2}.s$
Activation energy of surface diffusion	Q_S	(55)	`Qs`	$KJ.mol^{-1}$
Pre-exponential factor of GB mobility	M_{0GB}	(56)	`Mob0`	$m^3.N^{-1}.s^{-1}$
Activation energy of GB mobility	Q_{GBM}	(56)	`Qmob`	$KJ.mol^{-1}$

The driving force for grain growth is the difference in size among particles, therefore it is necessary to have an initial size distribution different from a purely monomodal packing, which is in accordance with real packings. The probability distribution type and its parameters are declared in the `input_gas` file, in this example a lognormal distribution is used. As coalescence between particles progresses, small particles will be 'eaten' away by large ones and gradually disappear. The total number of particles will decrease during the simulation. Thus, it is necessary to start with a significant number of particles in order to still obtain representative results. In this example for illustration purposes we have initially 4k particles, however we recommend at least 40k or even more.

Fig. 22a shows the evolution of the mean particle radius during densification[1] . The simulations stops either when the simulation termination condition declared in `input_dp3D` is reached or when a particle overgrows, reaching a quarter of the simulation periodic box size. Fig. 22b displays the evolution of the number of particles with time[2] .

As particle size changes or even disappears, it is needed to set a lower time step than in standard sintering simulations without grain growth. In addition, the tangential viscous force will influence the grain growth, a value of 0.01 for the `frict` keyword in `input_dp3D` leads to consistent values of the tangential force (Eq. (50)).

In coupled sintering and grain growth, significant changes on microstructure are expected. The command `dp3D_pixel` allows to generate 2D raw images of a given coordinate file. As the coordinate files still includes particles that have reached a critically small radius (their label is set to 00), first it is necessary to keep only particles that still exist[3], in this example `a1`. Fig. 23 shows the microstructures for different coordinate files of the current example.

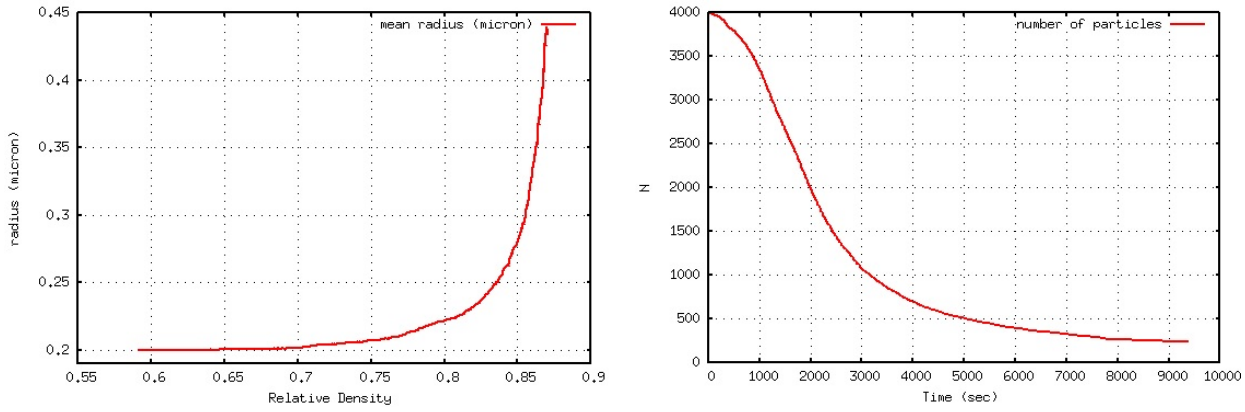
Contact laws used in this example:

Some equations from Section 2.16 are updated when considering grain growth in order to refine accuracy for large particle size ratios. Eq.(44) of Bouvard_Pan model is modified to [29]:

$$N_s = -\frac{\pi a_s^4}{\left(1+\frac{r_s}{r_l}\right)\beta\Delta_{GB}}\frac{\mathrm{d}h}{\mathrm{d}t} + \frac{\alpha}{4}\pi r_l \gamma_S \tag{52}$$

where r_s and r_l are the radius of the smaller and larger particle respectively for each contact. In grain growth, the parameter α is directly calculated by `dp3D` depending on the ratio of grain boundary and surface diffusion [28].

[1] `ddp3D -dens -rp`
[2] `ddp3D -time_s -Npart`
[3] `cdp3D -keep filename`



(a) (b)

Figure 22: (a) Evolution of mean particle radius versus density. (b)Number of particles vs time.

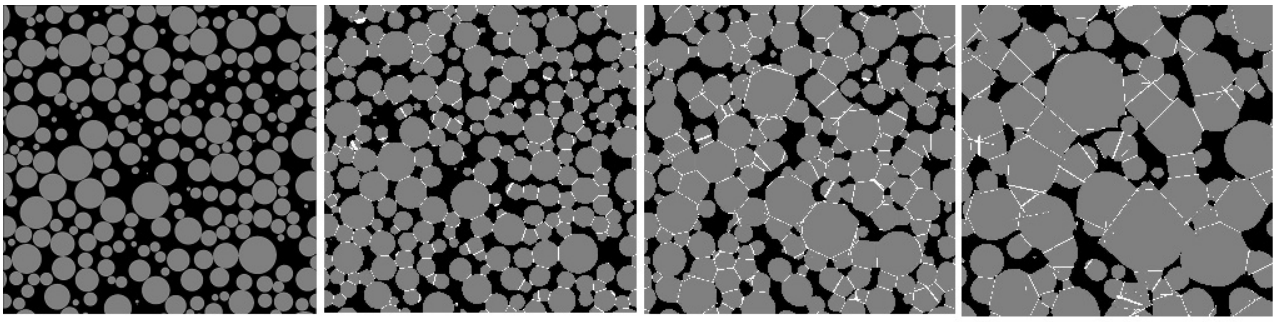


Figure 23: 2D microstructure evolution of a 4000 particles packing.

The sintering contact radius is calculated by [29]:

$$a_s^2 = \kappa \left[0.5\left(1 + \frac{r_s}{r_l}\right)\right]^{\zeta} r_l h \tag{53}$$

where $\kappa = 2.4$ and $\zeta = 1.5$ are fitted empirical values.

The Eq.(49) is replaced by :

$$a_{eq} = \frac{\Psi_{eq}}{\hat{\Psi}} \frac{r_s}{1 + \frac{r_s}{r_l}} \tag{54}$$

where $\hat{\Psi} = 92.937°$ is a fitted constant.

Grain growth laws:

The grain growth model implemented in `dp3D` considers two mechanisms of grain growth. When $a_s \geq a_{eq}$ the *surface diffusion* is activated:

$$\left(\frac{\mathrm{d}V_{l,s}}{\mathrm{d}t}\right)_S = -2\frac{D_S}{k_b T}\gamma_S \Omega \frac{\frac{1}{r_l} - \frac{1}{r_s}}{r_l + r_s - \delta}\left[\pi(a + \delta_S)^2 - \pi a^2\right] \tag{55}$$

If $a_s \geq r_s$ then *grain boundary migration* is switch on:

$$\left(\frac{\mathrm{d}V_{l,s}}{\mathrm{d}t}\right)_{GBM} = -2M_{GB}\gamma_{GB}\left(\frac{1}{r_l} - \frac{1}{r_s}\right)\left[\pi a^{*2}\right] \tag{56}$$

The second term of Eq.(52) is set to zero during surface diffusion and reactivated in GB migration.

```
#########################################################################
#                    simulation conditions                              #
#########################################################################
# coordinate file:
p4000dens59R02ln035
_________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
sintering
_________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
# constant_velocity gravity thermal
no_rotation
_________________________________________________________________________
#########################################################################
#                          models                                       #
#########################################################################
# sintering model:
# Parhami_Mc_Meeking Bouvard_Pan viscous
Bouvard_Pan
_________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
Coulomb
_________________________________________________________________________
# adhesion model:
# DMT JKR
JKR
_________________________________________________________________________
#########################################################################
#                          outputs                                      #
#########################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= sigxx>= sigxx<= sigyy>= sigyy<= sigzz>= sigzz<= fracture=
density>=0.90
_________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
density=0.1000E-01
_________________________________________________________________________
# writing output files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
density=0.1000E-03
_________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= sigxx= sigyy= sigzz=
# aoR= time= timestep= none end
end
_________________________________________________________________________
#########################################################################
#                          loadings                                     #
#########################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
_________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressure=
sigxx=0.0
sigyy=0.0
sigzz=0.0
_________________________________________________________________________
# temperatures(°C) and duration (s) for stage 1:
# none T_init= T_final= time=
T_init=0.1350E+04
T_final=0.1350E+04
time=0.
_________________________________________________________________________
#########################################################################
#                        materials (from 0,1,2, ... to 9)               #
#########################################################################
# elastic parameters (Pa for stress):
# E(0)= poisson(0)= E(1)= poisson(1)= delta_c(0)= fact_mult(0)=
E(1)=0.1000E+12
poisson(1)=0.2200E+00
_________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(0)= sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.1000E+10
Mstrain=0.0000E+00
Nvisco=0.0000E+00
_________________________________________________________________________
# friction parameters:
# frict(0,0)= frict(0,1)= frict(1,1)= frict(1,2)= frict(2,2)=
frict(1,1)=0.0100E+00
_________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(0,0)= adhes(0,1)= adhes(1,1)= adhes(1,2)= adhes(2,2)=
adhes(1,1)=0.
_________________________________________________________________________
# sintering material(s):
# 0 1 2 ... 9
1
_________________________________________________________________________
# surface energy parameters for sintering (J.m-2):
# gamma_s(0,0)= gamma_s(0,1)= gamma_s(1,1)= gamma_s(1,2)= gamma_s(2,2)=
gamma_s(1,1)=0.0905E+01
_________________________________________________________________________
# diffusion parameter (m3.s-1), activation energy (KJ/mol):
# DeltabD0b(0,0)= DeltabD0b(0,1)= DeltabD0b(1,1)=
# Qb(0,0)= Qb(0,1)= Qb(1,1)=
DeltabD0b(1,1)=1.3000E-08
Qb(1,1)=0.4750E+03
_________________________________________________________________________
# dihedral angle (°):
# chi(0,0)= chi(0,1)= chi(1,1)= chi(1,2)= chi(2,2)=
chi(1,1)=0.1380E+03
_________________________________________________________________________
# atom vol(m3):
# omega(0)= omega(1)=
omega(1)=0.2110E-28
_________________________________________________________________________
# mobility (m3/(N.s)), activation energy (KJ/mol):
# Mob0(0)= Qmob(0)= Mob0(1)= Qmob(1)=
Mob0(1)=0.0100E+00
Qmob(1)=0.443E+03
_________________________________________________________________________
# surface diffusion (m2/s), activation energy (KJ/mol):
# D0s(0)= Qs(0)= D0s(1)= Qs(1)=
D0s(1)=0.0900E+00
Qs(1)=0.3138E+03
_________________________________________________________________________
# density (g.mm-3):
# ro(0)= ro(1)= ro(2)=
ro(1)=0.3950E-02
_________________________________________________________________________
#########################################################################
#                          numerics                                     #
#########################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
_________________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping= fixed_dt= random_seed=
# potential_contact=
safe_dt=0.00500E+00
upscale(1)=0.1000E+01
damping=0.0000E+00
_________________________________________________________________________
```

2.19 Close-die compaction of a viscoplastic compact

- directory : `closedie_viscoplast`; coordinate file: `p400p6365ur`

The viscoplastic mode is invoked in `dp3D` by setting the mode to `viscoplasticity` in the `input_dp3D` file. No elasticity is considered. Thus, all contacts behave viscoplastically. We consider a particle made of a material that behaves in uniaxial tension with the following viscoplastic constitutive equation:

$$\sigma = \sigma_1 \varepsilon^M \dot{\varepsilon}^N, \tag{57}$$

where σ is the uniaxial stress, and ε and $\dot{\varepsilon}$ are the uniaxial strain and strain-rate, and σ_1, M and N are material constants.

Fig. 24 shows the stress evolution generated from the calculation with the `input_dp3D` shown above. Note how the stress evolves with no stress asymptote towards large density. This artificial effect is due to the power $1 + M/2 - N/2$ in Eq. (58) which for $N > 0$ tends to soften the response of the material. The option `large_dens` may be invoked to model the hardening effect arising at large density.

Contact laws used in this example:

The normal force between the two particles is written:

$$N_v = \eta_{part} \delta_n^{(1+M/2-N/2)} \dot{\delta}_n^N \tag{58}$$

where $\dot{\delta}_n$ is the normal rate of approach of the two particles. N_v opposes the relative motion of the two particles in tension and compression. η_{part} is defined as [4]:

$$\eta_{part} = 2^{1-\frac{M}{2}-\frac{3N}{2}} 3^{1-M-N} (1+2N) \pi c^{2+M+N} \sigma_0 (R^*)^{1-\frac{M}{2}-\frac{N}{2}} \tag{59}$$

where $\sigma_0^{-\frac{1}{M+N}} = \sigma_1^{-\frac{1}{M+N}} + \sigma_2^{-\frac{1}{M+N}} = 2\sigma_1^{-\frac{1}{M+N}}$ in the case of a homogeneous material ($\sigma_1 = \sigma_2$). The material constants σ_1, σ_2 (which units are Pa.sec^N), and N are set in the `input_dp3D` file. In the example shown here, `Mstrain=0.5` and `Nvisco=0.3`. A shear force at the contact is also implemented and its formulation derives from the writing of the shearing stress at the contact location.

$$T_v = -\mu \frac{2\pi}{\sqrt{3}} c^2 R^* \delta_n \sigma_1 \left(\frac{\|\mathrm{d}\delta_t/\mathrm{d}t\|}{R_1 + R_2 - \delta_n} \right)^N \frac{\mathrm{d}\delta_t/\mathrm{d}t}{\|\mathrm{d}\delta_t/\mathrm{d}t\|} \tag{60}$$

where $\mathrm{d}\delta_t/\mathrm{d}t$ is the tangential component of the relative velocity at the contact. μ is a viscous parameter with no dimension. The value of the viscous parameter is set in the `frict` line in `input_dp3D` (in the present example it is set to 1.0). A value of 1.0 leads to the standard shear resisting force. Note that for the time being, there is no resisting moment at the contact.

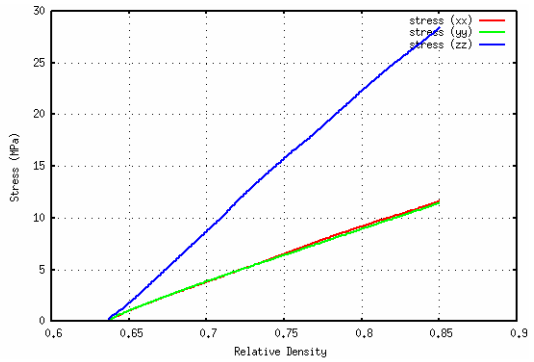


Figure 24: Stress evolution for viscoplastic aggregates under close-die compaction.

```
##########################################################################
#                    simulation conditions                               #
##########################################################################
# coordinate file:
p400p6365ur
__________________________________________________________________________
# mode key word:
# jamming elasto_plasticity viscoplasticity sintering
viscoplasticity
__________________________________________________________________________
# general key words:
# none stress_ref large_dens no_rotation rot_elast_only
none
__________________________________________________________________________
##########################################################################
#                                models                                  #
##########################################################################
# viscoplastic model:
# standard oriented
standard
__________________________________________________________________________
# friction model:
# Hertz_Mindlin Coulomb shear
shear
__________________________________________________________________________
# adhesion model:
# DMT JKR
JKR
__________________________________________________________________________
##########################################################################
#                               outputs                                  #
##########################################################################
# simulation termination:
# density>= epsilon>= pressure>= pressure<= epsvdot<= aoR>= time>=
# timestep>= fracture=
density>=0.8500E+00
__________________________________________________________________________
# writing coordinate files:
# density= epsilon= pressure= aoR= time= timestep=
density=0.1000E-01
__________________________________________________________________________
# writing output files:
# density= epsilon= pressure= aoR= time= timestep=
density=0.1000E-02
__________________________________________________________________________
# writing contact history files:
# density= epsilon= pressure= aoR= time= timestep= none end
end
__________________________________________________________________________
##########################################################################
#                             loadings                                   #
##########################################################################
# periodic conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
__________________________________________________________________________
# loading conditions (Pa for stress):
# epsxdot= epsydot= epszdot= sigxx= sigyy= sigzz= equal_stress= pressur
epsxdot=0.0000E+00
epsydot=0.0000E+00
epszdot=-0.1000E-03
__________________________________________________________________________
##########################################################################
#                             materials                                  #
##########################################################################
# elastic parameters (Pa for stress):
# E(1)= poisson(1)= E(2)= poisson(2)=
E(1)=0.2000E+22
poisson(1)=0.3400E+00
__________________________________________________________________________
# plastic parameters (Pa for stress):
# sigy(1)= sigy(2)= Mstrain= Nvisco=
sigy(1)=0.4000E+09
Mstrain=0.5000E+00
Nvisco=0.3000E+00
__________________________________________________________________________
# friction parameters:
# frict(1,1)= frict(1,2)= frict(2,2)= frict(object)=
frict(1,1)=0.1000E+01
__________________________________________________________________________
# work of adhesion parameters (J.m-2):
# adhes(1,1)= adhes(1,2)= adhes(2,2)= adhes(object)=
adhes(1,1)=0.0000E+00
__________________________________________________________________________
# density (g.mm-3):
# ro(1)= ro(2)=
ro(1)=0.7890E-02
__________________________________________________________________________
##########################################################################
#                              numerics                                  #
##########################################################################
# affine motion conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z
1x1y1z
__________________________________________________________________________
# control of strain-rates for quasi-static conditions:
# none aoamean= break= epsilon= ferror= kin_energy= vmax= ctrl_fact=
none
__________________________________________________________________________
# numerical parameters:
# safe_dt= upscale(1)= upscale(2)= damping=
safe_dt=0.1
upscale(1)=0.1000E+01
damping=0.0000E+00
__________________________________________________________________________
```

3 Additional tools with dp3D

3.1 Visualization tools: vdp3D and ovdp3D

vdp3D is a tool to create images and animations from the simulations that you have run with dp3D. vdp3D can also be used as a preprocessor to dp3D to modify a coordinate file. vdp3D uses the rather old but sturdy free software RASMOL 2.7 (RasMol is a program for molecular graphics visualisation originally developed by Roger Sayle (www.RasMol.org) and www.OpenRasMol.org).

The vdp3D command accepts many options. All options are listed below by alphabetical order (options that can be combined to make an animation are labelled (-anim)). The simplest one is vdp3D -i filename which simply shows the packing.

The visualization tool ovito (http://www.ovito.org/), [17], has been added since rasmol is becoming gradually obsolete. It is called with the command ovdp3D and has the same arguments as vdp3D. The quality of images is much better than ramol, and ovito downloads much faster large files. ovito is also a much better tool for generating animations. In the list below, (o)vdp3D means that the command can be either called with vdp3D or ovdp3D.

.....

vdp3D synopsis: `vdp3D [-option1] file1 file2 ...  filen range/xx:yy/`
ovdp3D synopsis: `vdp3D [-option1] file`

.....

3.1.1 (o)vdp3D -anim

- makes a series of images (gif files) that can be used to make an animation from the _coordxxxx files generated by dp3D. For ovdp3D -anim, the animation settings is created in ovito. ovito is much more powerful than RASMOL for animations.

3.1.2 (o)vdp3D -anim[n]

- same as above, but uses only a fraction ($n \leq 10$) of the _coordxxxx files (typically when too many _coord files have been generated)

- Example: in section 2.3 (directory crushing) the command:

  (o)vdp3D -anim2

  generates a gif file every two _coord file. These gif files may be used for creating an animation. The command may be combined with another option like:

  (o)vdp3D -anim2 -fract_bonds

  to obtain a series of images showing the evolution of broken bonds. To fix a given setting for all images a rasmol script may be given by copying the default ivdp3D rasmol script file to another file (here ivdp3D_90) and then:

  vdp3D -anim2 -fract_bonds ivdp3D_90

  where the rasmol command rotate x 90 has been embedded in the ivdp3D_90 file.

3.1.3 (o)vdp3D (-anim) -bonds filename

- generates an image showing number of non-broken bonds

3.1.4 `(o)vdp3D (-anim) center_clumps filename`

- generates an image showing the gravity center of clumps. Necessitates that file `filename_clumps` exists.

3.1.5 `(o)vdp3D (-anim) -class filename`

- generates an image showing non broken clusters of bonded particles. If file `filename_clumps` is provided, it only shows clumps.
- Example: in section 2.9 (directory `crushing`), the two figures on the RHS of Fig. 12 have been generated by the commands:

  `(o)vdp3D -class _coord0010`

  `(o)vdp3D -class _coord0020`

  and by rotating the figure in rasmol by the command :`rotate x 90`

3.1.6 `(o)vdp3D (-anim) -contact filename`

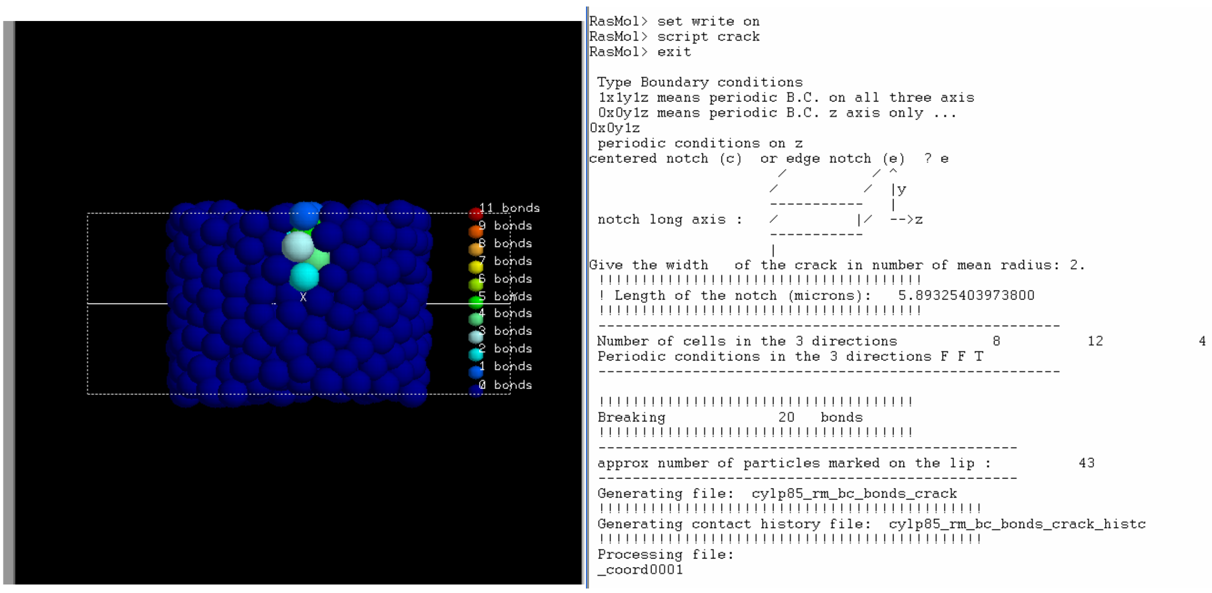
- generates an image of the packing showing sintered contacts. This is only an artificial viewing of the flow of matter near the contact.

3.1.7 `vdp3D -crack filename`

- generates a crack by breaking bonds between initially bonded particles from the coordinate filename.
- Example: in section 2.9 (directory `crushing`), you can create a crack by typing :

  `vdp3D -crack cylp85_rm_bc_bonds`

  then by selecting two particles that define the crack and then select the crack geometry (centered or edge), and its width. `vdp3D` will then display the sample with the number of bonds broken per particle. Note that this command generates two files: the coordinate filename cylp85_rm_bc_bonds_crack and the contact history file cylp85_rm_bc_bonds_crack_histc. This last file must be in the same directory as the cylp85_rm_bc_bonds_crack for further treatment (see sections 2.5 and 5.10).



3.1.8 `(o)vdp3D (-anim) -damage filename`

- generates damage_file.xyz with only damaged particles for viewing in ovito.

3.1.9 (o)vdp3D (-anim) -deltats filename

- generates an image showing maximum tangential strain accumulated for each particle.

3.1.10 (o)vdp3D -dens filename

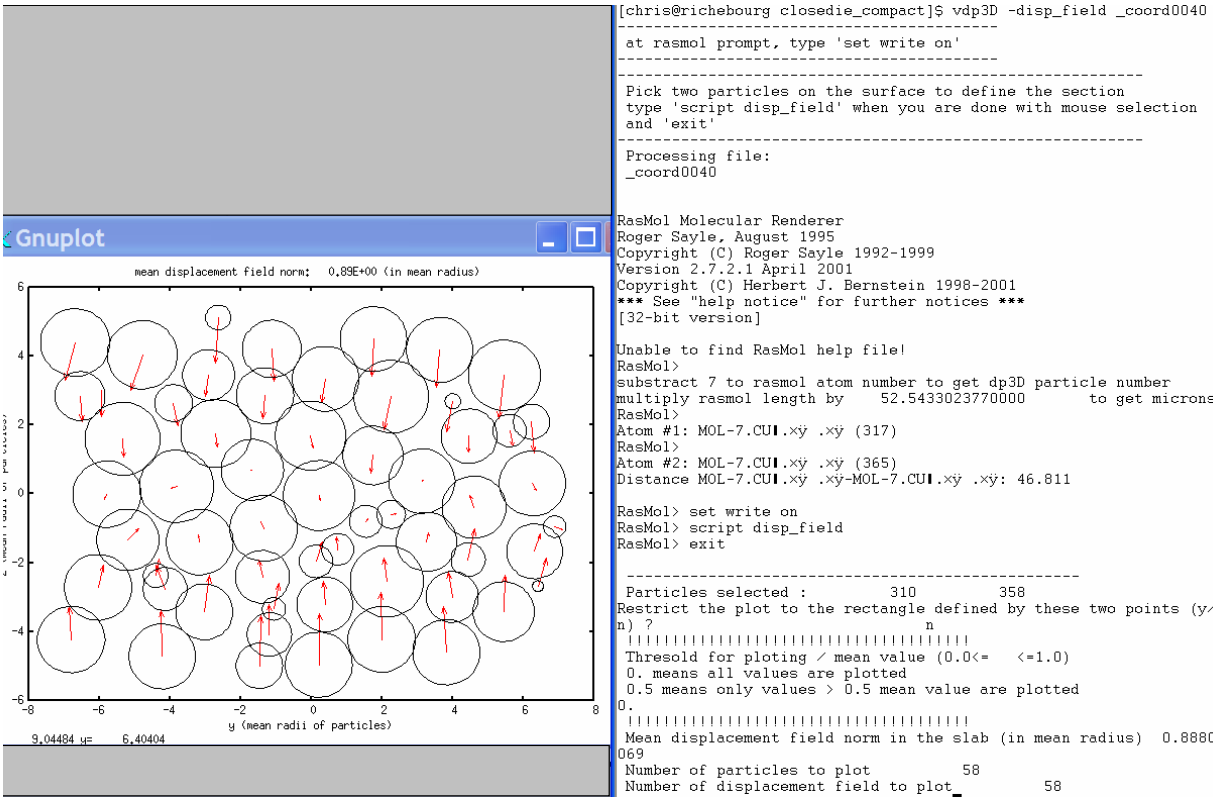
- Using the the radical Voronoi tesselation (Voro++ [20]) of the packing, generates an image showing the relative density of each particle (ratio of the particle volume to the Voronoi cell volume). This command will also generate a file named `filename_voronoi_dens` that contains the local Voronoi relative density for each particle. This file is necessary when using the `large_dens` keyword (see example 2.7).

3.1.11 vdp3D -disp_field filename

- displays the displacement field of a 2D section
- Example: in section 2.3 (directory `crushing`), type :

  `vdp3D -disp_field _coord0040`

  to obtain the displacement field on the section defined by the two particles that you have selected.



3.1.12 (o)vdp3D (-anim) -disp[x,y or z] filename

- generates an image of the packing showing particle displacements in x, y or z direction as compared to the initial position of particles.

3.1.13 vdp3D -ellipsoid filename

- generates a coordinate file with particles inside the ellipsoid selected in rasmol with a new name.

3.1.14 (o)vdp3D (-anim) -eps[x,y or z] filename

- generates an image of the packing showing local strains

3.1.15 (o)vdp3D (-anim) -force filename

- uses coordinate file to generate an image with maximum normal contact force for each particle.

3.1.16 (o)vdp3D (-anim) -fract_bonds

- generates an image with the number of fractured bonds for each particle.

3.1.17 (o)vdp3D -histo_bonds

- generates an image of the history of bonds fracture

3.1.18 (o)vdp3D -i filename

- uses coordinate file to generate an image

3.1.19 (o)vdp3D -iref filename

- generates an image of the packing showing the reference sphere or the reference rectangular slice, see -ref_sphere and -ref_rect options.

3.1.20 vdp3D -keep[n] filename

- keeps only the particles from file that you have selected in rasmol.
- vdp3D -keep3 means keeping particles 3 radii around the selected particle (and the selected particle itself).

3.1.21 vdp3D -Nnetwork filename

- displays the normal force network of a 2D section, similar to -disp_field option but needs a _histc file

3.1.22 vdp3D -notch filename

- generates a notch by removing particles from the coordinate file, similar to -crack option.

3.1.23 vdp3D -normal_axis filename

- displays the normal vector of clusters (needs the _clumps file).

3.1.24 vdp3D -object filename

- generates an image of the packing showing the cylin and spher objects

3.1.25 vdp3D -pick filename

- generates a coordinate file with picked particles having a new name (similar to cdp3D -pick).

3.1.26 vdp3D -probe filename

- generates a plot of the gradient of relative density along the axis normal to the plane chosen by picking two particles.

3.1.27 `vdp3D -ref_sphere` filename

- generates a reference sphere from two particles that you have specified in rasmol. The first one sets the center of the reference sphere, the second one defines the radius of the reference sphere. It generates a file `filename_ref`. When using this file, the reference sphere will be used to compute the density during the `dp3D` calculation. See also the option `stress_ref` in the general keywords of `input_dp3D` to use this volume to compute stresses during the calculation. This option may also be used to know the density, average coordination ... of the particles inside the reference volume. This option may be used to obtain properties of a specific part of the compact.

3.1.28 `vdp3D -ref_rect` filename

- generates a reference rectangular slice from two particles that you have specified in rasmol. The first one sets a corner of the slice, the second one defines the second corner. The normal to the slice has to be given by the user. It generates a file `filename_ref`. When using this file, the reference rectangular slice will be used to compute the density during the `dp3D` calculation. See also the option `stress_ref` in the general keywords of `input_dp3D` to use this volume to compute stresses during the calculation. This option may also be used to know the density, average coordination ... of the particles inside the reference volume. This option may be used to obtain properties of a specific part of the compact.

3.1.29 `vdp3D -rm[n] filename`

- generates a coordinate file with the particles, which you have selected in rasmol, removed.

- `vdp3D -rm3` means removing particles 3 radii around the selected particle (and the selected particle itself).

3.1.30 `(o)vdp3D (-anim) -sig[xx,yy,zz,xy,xz,...] or -press filename`

- generates an image of the packing showing the components of the stress tensor (or the pressure) for each particle. Note that the local stress tensor is taken from the _coord file. The local (per particle) stress tensor in the _coord file is calculated by using eqn (63) with:

$$V = \rho_{macro} \frac{4}{3} \pi R^3 \tag{61}$$

where ρ_{macro} is the density of the total sample or of the region delimited by a ref_sphere or ref_rectangle (see 3.1.27). However, as the local density around each particle may differ from the averaged ρ_{macro}, the vdp3D command for the stress will display the stress using a different volume:

$$V = \rho_{Voronoi} \frac{4}{3} \pi R^3 \tag{62}$$

where $\rho_{Voronoi}$ is the density calculated from the Voronoi cell (see 3.1.10) .

3.1.31 `vdp3D -sphere filename`

- generates a coordinate file with particles inside the sphere selected in rasmol with a new name.

3.1.32 `vdp3D -spy filename`

- Works similarly to -tag option. Here it will generate a coordinate file with some particles spyed during the `dp3D` calculation. These particles are followed all along the simulation. _spyxxxxx file will be generated during the `dp3D` calculation. Use `ddp3D -spy` command to display specific feature for these spyed particles.

3.1.33 `vdp3D -tag filename`

- generates a tagged coordinate file from filename. Tagged coordinate files have some particles tagged. The contacts that pertain to tagged particles are followed all along the simulation. `dp3D` generates a file: _tagxxxxx_yyyyyy for the contact between particles xxxxxx and particle yyyyyy if such a contact exists. Use `ddp3D -tag` command to display specific feature for these tagged contacts.

3.1.34 `vdp3D -Temp filename`

- generates an image with temperature of each particle.

3.1.35 `vdp3D -voronoi filename`

- generates the radical Voronoi tesselation (using Voro++ [20]) of the packing by showing particles together with the edges of the cells. For large packings (> 1000 particles), it is unwise to plot the Voronoi cells because Rasmol interprets each node as an atom which renders the visualisation extremely slow.

3.1.36 `(o)vdp3D (-anim) -z filename`

- uses coordinate file to generate an image with coordination numbers of each particle.

To obtain the last set of available options of `(o)vdp3D`, type `(o)vdp3D`.

The standard rasmol script loaded by `vdp3D` into rasmol is `/usr/local/dp3D/ivdp3D`. This script is for setting colors, zoom etc ... Each time, you run `vdp3D`, a copy of `ivdp3D` file is copied on the local directory. You may give it another name to use your own settings for your images (rasmol script commands are quite straightforward). For any options used with `vdp3D`, it is possible to give your own script by typing for example: `vdp3D -i file yourscript`.

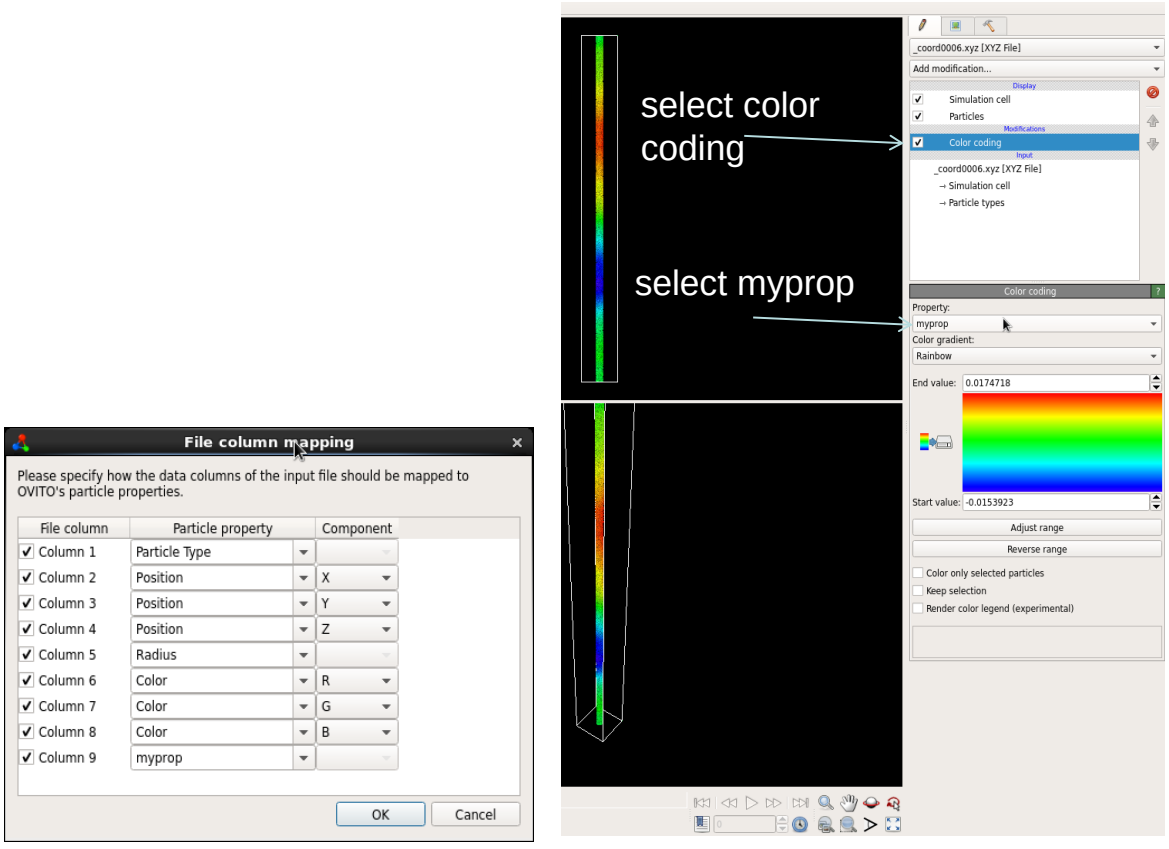
To use an animation, in a powerpoint presentation for example, coming from `vdp3D`. The simplest way is to proceed as following. Once the gif files have been generated by the `vdp3D` command, type on the linux prompt: `animate _coord*.gif`, then click on the animation and slow down the animation as desired. Click on the `save` menu and save the animation in gif format (for example `animation.gif`). The `animation.gif` file, once copied on your system, can be directly used in the powerpoint presentation.

Ovito can be downloaded from http://www.ovito.org/. The version should be 2.2.2 or higher. On the first use of ovito, you need to declare the file format that will be used. `ovdp3D` generates `.xyz` files that are readable by ovito with columns as given by Fig. 25a. The format must follow Fig. 25a. To visualize a given property (for example the coordination number with `ovdp3D -z _coord0006`), you need to select the ***`add modification`*** and ***`color coding`*** buttons as shown in Fig. 25b.

To save an animation on ovito with good quality graphics, use the `png` format. The whole animation must be screened in the renderer before saving the file.

- example of ovdp3D series of actions:

1. `ovdp3D -z filename` *to generate a plot of coordination numbers in the coordinate file* `filename`
2. *load (remote) file in ovito*
3. *fill the* `File column mapping` *(see Fig. 25) on the first use of ovito*
4. *select* `Add modification`
5. *select* `Color coding`
6. *select* `myprop`



(a) (b)

Figure 25: (a) Column format in `ovito` to read `xyz` files generated by `ovdp3D`. Reproduce the particle properties shown here on your first use of `ovito`(b) `ovito` screen shot with the use of `Add modification` and `myprop` to generate a plot of the `ovdp3D` option that you have selected.

3.2 Using an already existing coordinate file from dp3D_library

The densification of a gas of particles to form a jagged packing may be a CPU time consuming task (see section 2.2). Thus, before generating a whole new coordinate file (new microstructure), it is worth considering that there exists a library of coordinate files that have been created by `dp3D` users in the `dp3D_library` directory, which comes with the installation package. You may use/modify these files to suit your particular need. The `dp3D_library` directory is a series of coordinate files plus a `dp3D_library_log` file that gives information on each of these coordinate files. The file names already give some important information on the packing. For example:
`part4000_densp6153_size1p10_obj0_bond0_pct00_xr100_yr100_zr100`
indicates :

- `part:` the number of particles (here 4000);
- `dens:` the relative density (here 0.6153);
- `size:` the ratio of maximum size to minimum size of particles (here 1.10);
- `obj:` the number of objects (here none);
- `bond:` the total number of bonds in the packing (here none).
- `pct:` the volume fraction (percentage) of material 2 particles.
- `xr yr zr:` the packing relative geometry in the three directions, x, y, and z. Here for example the box is fully cubic with xr=yr=zr.

3.3 Working on a packing coordinate file using `cdp3D`

The command `cdp3D` allows to work on a `dp3D` coordinate file in order to carry a wide variety of operations or to obtain information on it.

.....

`cdp3D` synopsis: `cdp3D [-option] file`

.....

You may write a script (or a macro) with cdp3D to automatize a shell for example. This is carried out by:
`cdp3D -option_write file`
This will write the file `script_dp3D`, which can then be read and used with the command:
`cdp3D -option_read file`
This is a very useful option to automatize simulations and simulation preparations, using for exemple a shell or python like scripting.

This is a list of the current cdp3D options (the extension of the generated file is given in parenthesis):

- `cdp3D -bc filename` modifies Boundary Conditions in a packing. It replaces periodic B.C. by two plane objects which will be tangent (or further away) to the further located particle. (`_bc`).
- `cdp3D -bonds filename` creates bonds between particles and between particles and objects. (`_bonds`).
- `cdp3D -clumps filename`. If the file `filename_clumps` does not exist, it will create this file to allow for clumps use. If the file `filename_clumps` already exists, it will help you modify it to your needs. In that case, the previous `filename_clumps` is simply **overwritten**.
- `cdp3D -contact filename` translates objects to contact nearest particles. (`_trans`).

- `cdp3D -cut_cyl filename` modifies the simulation box coordinates. Particles that are inside or outside a cylinder are removed. (`_rm`).
- `cdp3D -cut_rect filename` modifies the simulation box coordinates. Particles that are outside a rectangular box are removed. (`_rm`)
- `cdp3D -export(_nxyz or _nxyzR) file` Exports a file from a dp3D format file to a `n x y z` or `n x y z R` format where `n` is an integer (particule number), `x, y, z` are the particle center coordinates and `R` is the particle radius.
- `cdp3D -full_info filename` Generate a file filename_full_info listing all contacts in the coordinate file with information on each contact. See also `cdp3D -info`.
- `cdp3D -gas` Generates a gas of particles or clusters or aggregates. Needs an input_gas file.
- `cdp3D -image filename` Given an initial coordinate file (generally a cube with a large number of particles), uses a raw image file to generate a dp3D format output file. Particles are of different types depending on the image level of grey. Generates two files: filename_image (with all particles from the initial coordinate files but with various atom names) and filename_image_rm where the particles originating from voxel with 0 value (black) are removed.
- `cdp3D -import(_xyz or _Geodict) file` Imports a file from various formats to generate a dp3D format output file. The standard command cdp3D -import_xyz accepts formats of the type: `x y z` (will ask a common radius for all particles), `x y z R` and `x y z R i` (where i is a integer which defines the color of the particle).
- `cdp3D -info filename` gives information on the coordinate file filename. See also `cdp3D -full_info`.
- `cdp3D -keep filename` keeps particles from a coordinate file by selecting them by atom name or particle number.
- `cdp3D -merge filename1 filename2` creates a file filename1_filename2_bc which merges filename1 and filename2.
- `cdp3D -mixture filename` generates a packing with a mixture of particles either. Inclusion particles may be arranged as homogeneous, clusters, with gradient, or layers. (`filename_mixt`).
- `cdp3D -noisolated filename` Enlarges isolated particles to form one contact or remove isolated particles.
- `cdp3D -part filename nn mm ...` gives coordinates, atom name and radius of particles nn mm ...
- `cdp3D -pick filename` pick particles by atom number or coordinates and rename them. (`_pick`).
- `cdp3D -probe_x (or y or z) filename` creates a plot of the relative density along the x, y or z axis.
- `cdp3D -rdf filename` creates a plot of the Radial Distribution Function.
- `cdp3D -renum filename` Optimize the numeration of particles to save on cache miss during the calculation. (`_renum`).
- `cdp3D -resize filename` allows to modify the size of the particles by a given size ratio. (`_size`).
- `cdp3D -rm filename` remove particles by atom name or number. (`_rm`).
- `cdp3D -rmobj filename` remove all objects from the coordinate file. (`_rm`).
- `cdp3D -rotate filename` rotates the packing of particles around an axis. (`_rot`).

- `cdp3D -sinter filename` "sinters" a packing by simply decreasing isostatically the distance between the particles. This will create larger contacts between particles. Of course such a microstructure is highly simplified as compared to one obtained by sintering with `dp3D` (with equilibrium of sintering forces ...). (`_size`)
- `cdp3D -spy filename nn mm ...` creates a spied coordinate file filename_spy with the particles nn mm ... spied. (`_spy`)
- `cdp3D -tag filename nn mm ...` creates a tagged coordinate file filename_tag with the particles nn mm ... tagged. (`_tag`)
- `cdp3D -trans filename` translates the whole packing (with objects if any). (`_trans`).
- `cdp3D -voxel filename` creates a raw file with voxels to create a 3D image of the packing. Contacts between particles may be filled using various options, depending on the mechanism for material flow.

New options in `cdp3D` are added routinely. Simply type `cdp3D` to get the latest description of the available options.

3.4 Generating a gas of clusters

- directory : `generate_gas_clusters`; coordinate file: `file_init`

The command `cdp3D -gas` has already been commented in section 2.1. Here we describe how to use the same command to create a gas of clusters (also termed aggregates in this guide). The `input_gas` example file is used to generate a gas of bonded clusters enclosed in a cylinder (axis z). Clusters of primary particles are sets of particles that are bonded together by strong elastic bonds as exemplified in sections 2.8, 2.9, and 2.11. Clusters are inserted in the gas as `motif` that are located randomly in the simulation box. Keywords have the following meaning:

- A `motif` file must be provided. The motif is given for each class, by a coordinate file. A set of such motif files (that have standard `dp3D` coordinate file format) are given in the `generate_gas_clusters` directory. The motif must already contain bonds which will be saved accordingly in the final gas of clusters.
- The `packing_density` is the packing density of the packing of motifs, taking into account the internal porosity of motifs.
- The `number` is the number of motifs in each class.
- The `name` of particles will be the name of all the particles of the imported motif. If this keyword is not provided the particle name(s) of the `motif` is(are) used.
- The `mat` of particles will be the material number of all the particles of the imported motif. If this keyword is not provided the particle material number(s) of the `motif` is used.
- The `particle_size` sets the average size of the primary particles contained in the `motif` (no deviation available). If this keyword is not provided the particle size of the `motif` is used.
- Each motif is introduced in the simulation box with a random rotation as compared to its original orientation. However, it can be interesting to impose some anisotropy to the initial gas. To do this, invoke `max_angle` to define a maximum cone for setting the cluster into the simulation box.

In the example shown here, three classes are used. The first two classes use `motif` to introduce clusters in the simulation box (`motif` files `large_sphere` and `platelet`). The last class is simply composed of particles.

Note that on top of the `file_init` file, the command `cdp3D -gas` has also generated a `file_init_clumps` file, which use has been described in section 2.15.

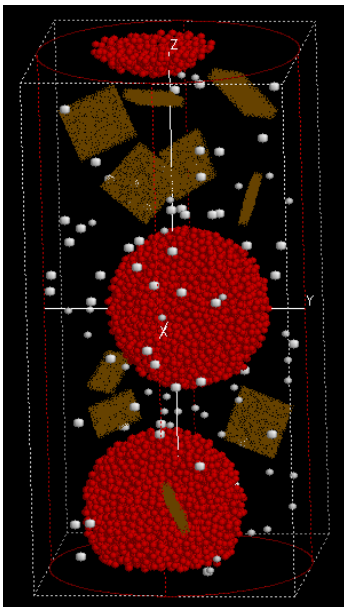


Figure 26: The gas of clusters generated from the `input_gas`. Note the effect of z periodic conditions on the large spherical aggregates, the use of a cylindrical object and the mixture of different shapes of aggregates.

```
########################################################################
#                    general conditions                                #
########################################################################
# coordinate file:
file_init
________________________________________________________________________
# packing density to reach:
# packing_density= relative_overlap=
packing_density=0.1
________________________________________________________________________
# boundary conditions:
# 1x1y1z 1x1y0z 1x0y0z 1x0y1z 0x1y1z 0x1y0z 0x0y1z 0x0y0z cyl_x cyl_y
# cyl_z
0x0y1z
cyl_z
________________________________________________________________________
# simulation box size ratios:
# x= y= z=
x=1.00
y=1.00
z=2.00
________________________________________________________________________
# size distribution:
# plus_minus normal lognormal
plus_minus
________________________________________________________________________
########################################################################
#                    classes                                           #
########################################################################
# prop. of class (particle sizes in µm)               1:
# number= mat= name= particle_size= deviation= Temp= motif= max_angle=
number=2
mat=1
name=ar
particle_size=0.20
motif=large_sphere
________________________________________________________________________
# prop. of class (particle sizes in µm)               2:
# number= mat= name= particle_size= deviation= Temp= motif= max_angle=
number=10
mat=1
name=cu
particle_size=0.05
motif=platelet
________________________________________________________________________
# prop. of class (particle sizes in µm)               3:
# number= mat= name= particle_size= deviation= Temp= motif= max_angle=
number=100
mat=2
name=ni
particle_size=0.3
________________________________________________________________________
########################################################################
#                            numerics                                  #
########################################################################
# random seed:
# random_seed=
random_seed=-5
________________________________________________________________________
```

3.5 Post-processing data: ddp3D

Many of the output files that are generated by dp3D can be viewed as 2D plots using the ddp3D tool which is a simple shell preparing and then calling gnuplot. In order to know which type of data ddp3D command can display, simply type ddp3D. The possible options will be listed.

The general synopsis is:

ddp3D -datatype(optional) -{x axis variable} -{first y axis variable} -{second y axis variable} ... xrange/minvalue:maxvalue/ yrange/minvalue:maxvalue/ anim

For example the commands:

ddp3D -dens -sigxx -sigyy -pressure will display the xx and yy stress components of the macroscopic stress tensor together with the pressure as a function of the relative density.

or

ddp3D -dens -z xrange/0.7:0.8/ yrange/6:7/ will display the average coordination number as a function of density with given ranges for x and y axis.

The possible data types, which are given before the x and y variable definitions, include:

- -tag
- -spy
- -histo

tag and spy types allow the specific information stored in the _tag**** and _spy**** files to be plotted for tagged contacts or spyed particles. For example: ddp3D -tag -epsx -sigN_b will display the normal stress at the bond of tagged particles as a function of the x strain component.

The data type -histo, allows histogram to be plotted. For example: ddp3D -histo -z filename will display the distibution of coordination numbers for the coordinate file filename.

valid x labels for ddp3D are :

- -dens : density -dgeom : geometric density
- -epsx | -epsy | -epsz : x y and z strain
- -time_s | -time_h | -time_y : time in sec, hour or year
- -timestep : time step
- -p : pressure
- -temperature : temperature

Type ddp3D -help to get the list of valid x labels. Type ddp3D together with a valid x label to obtain the list of valid y labels. New y labels are added routinely.

The options xrange/minvalue:maxvalue/ and yrange/minvalue:maxvalue/ allow ranges to be imposed to the plot. Note that if the range is not set, gnuplot will use default values. The option anim generates an animated plot with an animated point synchronized with _coord files.

3.6 Stress calculation in dp3D

The macroscopic stress tensor in the packing is calculated from Love's formulation [33, 34]:

$$\Sigma_{ij} = \frac{1}{V} \sum_{\text{contacts}} F_i l_{pq,j}, \tag{63}$$

where the summation is made on all contacts and where V is the sample volume, F_i is the i^{th} component of the total contact force (with normal and tangential terms), and $l_{pq,j}$ is the j^{th} component of the l_{pq} vector connecting the centers of two particles p and q (see Fig. 27 for the case of a bonded contact).

When using composites (different materials) or bonded contacts, it can be interesting to obtain some information on the contribution of various materials or on the contribution of various bond types on the overall stress. For example, the command:
`ddp3D -dens -sigzz_mat_1 -sigzz_mat_2 -sigzz`
displays the zz component of the macroscopic stress contribution for materials 1, 2 and the total stress. Fig. 7 shows the result of this command for a composite packing. For material 1, the stress contribution is calculated from an equation similar to Eq. (63):

$$\Sigma_{ij,\text{mat1}} = \frac{1}{V} \sum_{\text{contacts mat1}} F_i \left(R_p - \frac{1}{2}\delta_n \right) n_j, \tag{64}$$

where the summation is made here on all contacts that involve a particle of material 1. δ_n is the geometrical indentation between the two particles at the contact (see Fig. 4) and n_j is the jth component of the normal vector $\mathbf{n}$.

The same methodology applies for clusters. The command:
`ddp3D -dens -sigzz_0 -sigzz_1 -sigzz_2 -sigzz`
displays the zz component of the macroscopic stress contribution for Hertzian contacts (`-sigzz_0`), bonded contacts (`-sigzz_1`), broken contacts (`-sigzz_2`), and the total stress.

If objects are present in the simulation, stresses may also be calculated from the summation of the contact forces on these objects (for planes and cylinders) with a appropriate choice of surface normalization. See section 4.5 for a more detailed discussion on stress and objects.

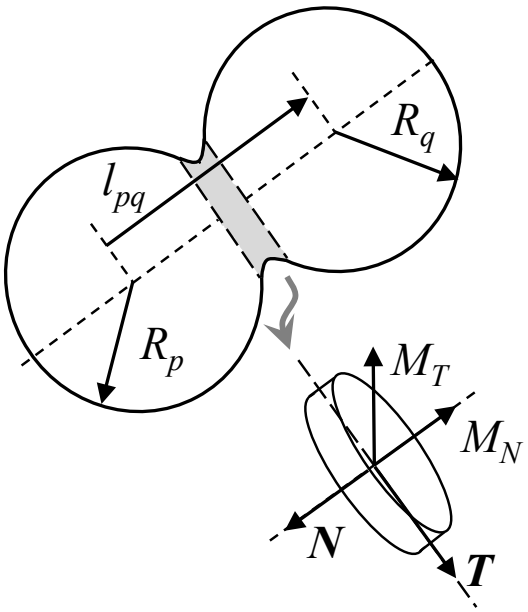


Figure 27: Sketch of the bond between two spherical particles p and q transmitting normal and tangential forces and resisting moments.

4 Coordinate file

dp3D needs two files to start a calculation: the initial coordinate file that keeps the location of each particles in the simulation box and the input_dp3D file that describes the conditions of the simulation. This last file has been described in the examples and its entries are detailed in section 6. Here we will focus on the coordinate file.

An example of a (very short and hypothetical) coordinate file is given in figure 29. Units in the coordinate files are in mm.

4.1 The two first lines

The first line of the file gives the number of particles, the number of objects, the number of bonds and the mean radius of the packing. Note that the mean particle radius is only given as a user information, dp3D does not use this information.

The second line of the file gives the coordinates of the rectangular simulation box.

4.2 The description of the particles of the packing

The first column gives the material index of the particle (integer from 0 to 9) . This refers to the material index used in the file input_dp3D to define material parameters. Atom labels are used in order to give colors to the particles in vdp3D (a list of colors associated to each atom type may be found in the ivdp3D file once you have run vdp3D once). Particles labeled:
'ac','al', 'ag', 'am', 'ar', 'at', 'au', 'ba', 'be', 'bi', 'bk', 'br', 'cd', 'cu', 'fe', 'fr', 'ge' 'in', 'li,'mg', 'ni', 'nb', 'os', 'pd', 'po', 'pt', 'pu', 'ra', 'sn', 'ti', 'tm', 'xi', 'zn', 'zr'
are permitted.

You may **tag** a particle by adding a star "*" after the label (this is exemplified for three particles "cu" in figure 29)[1]. This will tell dp3D to write detailed information on any contact between these particles and any other tagged particles. Typically information on contacts are size, contact forces, ... If you tag two particles that never make contact during the simulation, no _tag file will be written.

You may **spy** a particle by adding a hat "^" after the label (particle 9 is both spied and tagged in the exemple)[2]. This will tell dp3D to write detailed information on any particle which has been defined as spied. Typically information on particles are their position, velocity, total force acting on them ...

The label of the particle is followed by the 3 coordinates of the center of the sphere and by the radius of the sphere. After calculation, the _coord files may have some more columns after the 3 coordinates and radius.

4.3 The description of objects of the packing

Following the particles, is a list of the **objects**. By default, all objects are tagged, hence the object label does not need the "*". Objects are also ascribed a material index in the first column, as particles. Here material index for the three objects is 0.

A plane is characterized by the label "plane" and its equation:

$$n_x x + n_y y + n_z z = r \tag{65}$$

where the vector (n_x, n_y, n_z) is the normal to the plane pointing outward (as compared to the particles) (fig. 28). The four parameters (n_x, n_y, n_z, r) are given after the plane label. The parameter r may be negative or positive. The vector (n_x, n_y, n_z) is a unit vector.

A cylinder is characterized by the label "cylin" followed by its unit axis vector and its radius. The second set of parameters that are needed to define the cylinder is given after the label "C_cyl" which gives the coordinates of a point on the axis of the cylinder.

A sphere is characterized by the label "spher" followed by the coordinate of its center and its radius.

[1] A simpler way to tag particles for large packings, is to use vdp3D -tag as explained in section 3.1.
[2] A simpler way to spy particles for large packings, is to use vdp3D -spy as explained in section 3.1.

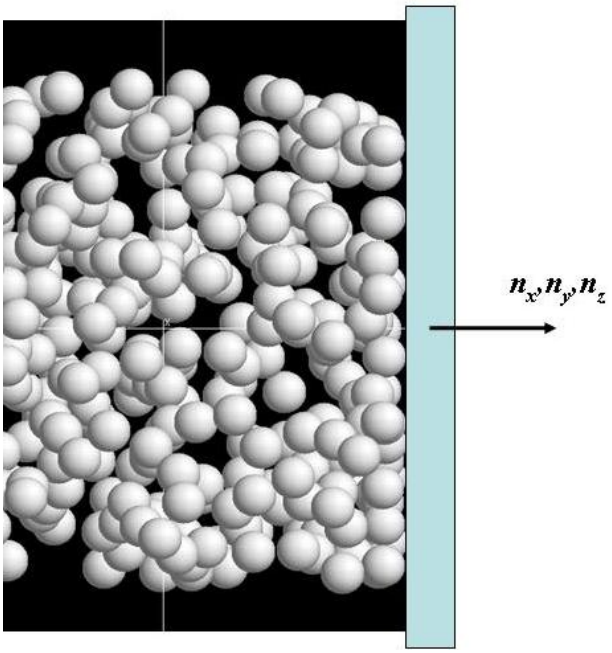


Figure 28: The normal to the plane points outward (as compared to the particles)

```
particles:       10 objects:       3 bonded_contacts:          8  mean_radius:   0.4996828330026E-04
 -0.227740355488E-03 -0.227740355488E-03 -0.167772889903E-03  0.227740355488E-03  0.227740355488E-03  0.167772889903E-03
1  cu     0.209994526031E-04  0.147964422698E-04  0.115622938462E-03  0.485437946009E-04
1  cu    -0.420316717224E-04 -0.606050323877E-04  0.118854545041E-03  0.489183448624E-04
1  cu     0.387305721265E-04  0.131111452603E-03 -0.101426536563E-03  0.522194444583E-04
1  cu    -0.516565004525E-05  0.106815568600E-03  0.899676514298E-04  0.514065728295E-04
1  cu*   -0.785830800535E-05  0.431746691846E-04 -0.118079873084E-03  0.496930168190E-04
1  cu*    0.189926912701E-04 -0.250221979960E-04 -0.530662259893E-04  0.492645440781E-04
1  cu    -0.280680366057E-04 -0.152462895878E-03  0.972732869980E-05  0.477583578957E-04
1  cu^    0.694794165134E-05 -0.662996103951E-04  0.347825060285E-04  0.495317430618E-04
1  cu^*  -0.109009713780E-04  0.148500591001E-03 -0.475270574720E-05  0.517206627734E-04
1  cu     0.386503314117E-05 -0.147754953831E-03  0.115496816044E-03  0.506263516469E-04
  list of objects:
** *****  ******************   ******************  ******************  ******************
0  plane  0.000000000000E+00  0.000000000000E+00  0.100000000000E+01  0.167772889903E-03
** *****  ******************   ******************  ******************  ******************
0  plane  0.000000000000E+00  0.000000000000E+00 -0.100000000000E+01  0.167772889903E-03
** *****  ******************   ******************  ******************  ******************
0  cylin  0.000000000000E+00  0.000000000000E+00  0.100000000000E+01  0.227740355488E-03
   C_cyl  0.000000000000E+00  0.000000000000E+00  0.000000000000E+00
** *****  ******************   ******************  ******************  ******************
   bonded contacts (i,j):         8 Coble
        1       4
        2       8
        2      10
        3       5
        5       6
        5      12
        6       8
        7       8
```

Figure 29: Example of a coordinate file with ten particles, seven bonds and three objects.

4.4 Imposing motion to objects

For planes, the rate of change of the parameter R (the distance from the origin) is given by default by the `loading conditions` of input_dp3D. However, the user may also specify a different rate of change for R by adding a line just after the `plane` or `cylin` line. Note that both an `srate` and an `irate` option may be specified. `srate` tells `dp3D` that the *initial* rate of change of R is given by this line (but may then be changed by the `dp3D` controller when a stress controlled test is imposed for example). `irate` tells that the rate of change of R is *imposed* strictly by this line. Thus, for example, the two lines defining a plane and its optional rate of change may be written:

```
***** ****************** ****************** ****************** ******************
0 plane 0.000000000E+00 0.000000000E+00 1.000000000E+00 0.123456789E-01
.  rate 0.000000000E+00 0.000000000E+00 0.000000000E+00 -0.100000000E-04
***** ****************** ****************** ****************** ******************
```

or

```
***** ****************** ****************** ****************** ******************
0 plane 0.000000000E+00 0.000000000E+00 1.000000000E+00 0.123456789E-01
.  irate 0.000000000E+00 0.000000000E+00 0.000000000E+00 -0.100000000E-04
***** ****************** ****************** ****************** ******************
```

The rate of change of R is given by:

$$\frac{1}{R}\frac{dR}{dt} = srate = -0.100000000\mathrm{E}^{-04} \tag{66}$$

Thus, it is a strain rate which defines the motion of the planes.

A second way to move planes is to write for example:

```
***** ****************** ****************** ****************** ******************
0 plane 0.000000000E+00 0.000000000E+00 1.000000000E+00 0.123456789E-01
.  irate 0.000000000E+00 1.000000000E-05 0.000000000E+00 0.000000000E+00
***** ****************** ****************** ****************** ******************
```

In that case the motion of the plane is in the y direction. Thus, it will generate a shear motion of the plane, which normal is in the z direction. The velocity of the plane in the y direction, V_y, is:

$$V_y = irate(y)R = 1.00000000\mathrm{E}^{-05}R \tag{67}$$

The same type of motion may be imposed to spheres. Typically by providing a strain-rate in one direction for the object *spher*:

```
***** ****************** ****************** ****************** ******************
0 sphere 0.123456789E-01 0.123456789E-01 0.123456789E-01 0.123456789E-01
.  irate 0.000000000E+00 -1.000000000E-05 0.000000000E+00 0.000000000E+00
***** ****************** ****************** ****************** ******************
```

which will impose a motion in the y direction to the sphere with the following velocity for the center of the sphere in the y direction:

$$V_y = irate(y)y = -1.00000000\mathrm{E}^{-05}y \tag{68}$$

Note that the velocity of the spher will not be constant (since y will be variable), only the strain rate is constant.

If several objects are used together to form a whole meta object, it may be useful to impose a rigid body motion to it. Typically, this may be useful if a series of `spher` objects are used to represent a complex shape meta object. In `dp3D`, rigid body motion may be imposed by declaring a *master* object and several *slave* objects:

```
1 cu 0.397619482720E-01 0.336330798024E-01 0.755523710191E-01 0.975257332123E-03
1 cu 0.494025233448E-01 -0.556199169059E-01 0.265963474629E-03 0.103153505529E-02
```

```
1 cu 0.498359740187E-01 0.139189457044E-01 -0.733279450527E-01 0.964238293021E-03
list of objects:
***** ****************** ****************** ****************** ******************
0 sphere 0.234567891E-01 0.234567891E-01 0.234567891E-01 0.234567891E-01
.  slave 5
***** ****************** ****************** ****************** ******************
0 sphere 0.123456789E-01 0.123456789E-01 0.123456789E-01 0.123456789E-01
.  irate 0.000000000E+00 -1.000000000E-05 0.000000000E+00 0.000000000E+00
***** ****************** ****************** ****************** ******************
```

In this example, object spher 4 (since there are three particles in the packing, the first object is n°4), is slaved to object spher 5, which is the master object (several master objects can be declared). In that case the velocity of object 4 is the same as the velocity of object 5:

$$V_y(4) = V_y(5) = irate(y(5))y(5) = -1.00000000\mathrm{E}^{-05}y(5) \tag{69}$$

A third way to impose motion to object is specific to cylinders. It consists in imposing rotation to the cylinder around its axis (for example to model a Couette rheometer) by writing:

```
***** ****************** ****************** ****************** ******************
0 cylin 0.000000000E+00 0.000000000E+00 1.000000000E+00 0.123456789E-01
.  irota 0.000000000E+00 0.000000000E+00 0.100000000E-04 0.000000000E+00
C_cyl 0.000000000E+00 0.000000000E+00 0.000000000E+00
***** ****************** ****************** ****************** ******************
```

In that case the rotation will be anti-clockwise (0.100000000E-04>0) and must be given around the axis of the cylinder (z axis here). The velocity at the cylinder frontier, V_{cyl} will be given by:

$$V_{rot} = irota(z)R = 0.100000000\mathrm{E}^{-04}R \tag{70}$$

4.5 Imposing stress on objects

By default stress imposed tests use the macroscopic stress to control strain rates. The macroscopic stress is calculated from [33–35] (see section 3.6):

$$\sigma_{jk} = \frac{1}{V}\left(\sum_{contacts}\left(R_1 + R_2 - \delta_n\right)Nn_j.n_k + \sum_{contacts}\left(R_1 + R_2 - \delta_n\right)Tn_j.t_k\right) \tag{71}$$

where V is the volume of the packing considered (a periodic box, or a spherical volume as defined by the option `stress_ref`), $\mathbf{n}$ and $\mathbf{t}$ are the unit vectors, respectively normal and parallel to the contact plane.

Alternatively, stresses can be calculated from the total contact forces applied onto objects and using an appropriate measure of the surface. The output file `object` contains this information for each object. A stress imposed test can thus use an object to control the strain rate. This option (only available for planes in the x, y and z principal directions and for cylinders in the radial directions) imposes to declare the keyword `stress_ref` in the general key words of the `input_dp3D` file and when declaring the object in the coordinate file:

```
***** ****************** ****************** ****************** ******************
0 plane 0.000000000E+00 0.000000000E+00 1.000000000E+00 0.123456789E-01
.  stress_ref
***** ****************** ****************** ****************** ******************
```

In the example above, the stress σ_{zz} on this plane will be used to control the strain-rate.

4.6 The list of bonds

The last set of information that may be needed is the list of particles or objects that are bonded together. Fig. 29 shows for example that particle 7 and 8 are bonded together and that particle 5 is bonded to object 12.

5 Output Files

Upon calculation, `dp3D` generates result or output files. Most of these files may not be interesting for the beginner. This is because the post-processing tool `ddp3D` will manage graphic visualization of the information contained in these files. Nevertheless, it may be useful to know where information is stored after the simulation. Most of these files are written when the `writing output files:` condition is met. A short description of these files is given hereafter.

5.1 `log` file

The `log` file is generated even if the simulation failed. It contains a summary of the evolution of the calculation.

The information contained in the log allows you to evaluate the most important aspects of the calculation. Each section of the log file is written when the `writing output files:` condition is met. The `log` file also tells when a coordinate file has been written and gives valuable information if the calculation crashes.

5.2 `tstress` file

The `tstress` file gives information on the evolution of several parameters, mostly related with strain, time, time-step and stress. Each column has a label which should be self-explanatory.

5.3 `zave` file

The `zave` file gives information on the evolution of several parameters, mostly related with coordination, contact size, and indentation. Each column has a label which should be self-explanatory.

5.4 `rupt` file

The `rupt` file is generated only if the cluster mode is on. It gives information on the evolution of several parameters, mostly related with cluster bonds. Each column has a label which should be self-explanatory.

5.5 `object` file

The `object` file is generated only if objects exist in the simulation. For each object, it gives information on the evolution of several parameters, mostly related with the total force and stress applied to objects. Each column has a label which should be self-explanatory.

5.6 `fract_bonds` file

The `fract_bonds` file is generated when bonds are present in the simulation. The file gives information on the fracture events of each bond. It is used by the `vdp3D -fract_bonds` option. Each line corresponds to a bond that has fractured. Each column has a label which should be self-explanatory.

5.7 `warnings` file

The `warnings` file is written if one or more warnings have been generated during the calculation.

5.8 `cstatus` file

There are a number (more than 30 to date) of different contact status in `dp3D`. For example `hertzian`, `broken_bond` or `sinter_equil`. The `cstatus` file gives the number of contacts for each status. The name of the status should be self explanatory and the complete list is available in `get_cstat.f90`.

5.9 `_coord` files

The `_coord` files are generated when the `writing coordinate files:` condition is met. These files have been described in section 4. These files may be renamed (highly recommended, since it may be overwritten) and used for another `dp3D` calculation. See next section on how to restart a calculation with the full contact information from the last run.

5.10 `_histc` files

A `_nnnn_histc` file is written at the end of each calculation (except for the sintering mode). The nnnn number corresponds to the number of the last _coordnnnn coordinate file written in the simulation. See section 2.5 on how to use this file to restart a calculation with the full contact information from the last run.

5.11 `_clumps` files

A `_nnnn_clumps` file is written at the end of each calculation (except for the sintering mode) if a `_clumps` file has been provided initially and if the `clump_cluster` option has been given. The nnnn number corresponds to the number of the last _coordnnnn coordinate file written in the simulation.

5.12 `_tag` files

The `_tagnnnnnn_mmmmmm` file exists if both particles (or object) nnnnnn mmmmmm have been tagged (see section 3.1 or 3.3 on how to tag particles prior to the simulation) and if they have formed a contact during the simulation.

The `_tag` file gives information on the contact evolution (typically indentation, contact size, maximum normal and tangential contact forces, ...). Each column has a label which should be self-explanatory. The `_tag` file is used by the command `ddp3D -tag` (see section 3.5).

5.13 `_spy` files

The `_spynnnnnn` file exists if particle nnnnnn particle has been spied (see section 3.1 or 3.3 on how to spy particles prior to the simulation). The `_spy` file gives information on the particle evolution (typically coordinates, velocity, acceleration, coordination number, ...). Each column has a label which should be self-explanatory. The _spy file is used by the command `ddp3D -spy` (see section 3.5).

6 The `input_dp3D` file

The `input_dp3D` file lists the input parameters needed to run a `dp3D` simulation. The example sections have shown the use of most of the key words in `input_dp3D`. Some new ones appear frequently. They are listed below. Key words are arranged in six classes:

- *simulation conditions*: general conditions for the simulation
- *models*: specifies the models used
- *outputs*: when outputs should be written
- *loadings*: loading conditions
- *materials*: material parameters
- *numerics*: numerical parameters

For each class, there are different key words (kw). For example in the `models` class, there are the `friction model` or `adhesion model` key words. For each of these key words, there are various input parameters asked for. Note that some may be useless for a given simulation and thus may be omitted. You may also, give more than one input parameter for a given parameter. For example, you may ask for the simulation to have both the `stress_ref` and the `gravity` key words.

In the following, we describe each class, key word and input parameter. The default value (if the input parameter is omitted) is indicated. *None* means that a kw must be given). A class may accept multiple key words. If it does not (last column of the following description), then if the user gives several kws, only the last one will be taken into account. Some key words accept values as: `kw=value`.

```
#########################################################################
#                          simulation conditions                        #
#########################################################################
```

	comment	default entry if omitted	accepts multiple kw
`# coordinate file:`	*The name of the input coordinate file*	must be defined	no
`# mode key word:`	*type of simulation to be run*	must be defined	no
`jamming`:	*jamming mode*	none	
`elasto-plasticity`:	*elasto-plastic mode*	none	
`viscoplasticity`:	*viscoplastic mode*	none	
`sintering`:	*sintering mode, forbids rotations*	none	
`# general key words:`	*kw that apply to the simulation*		yes
`none`:	*no specific kw given*		
`stress_ref` :	*A reference sphere or rectangle or an object is used to compute stresses*	stress computed on the entire simulation box	
`large_dens=` :	*Large density type of (visco)-plastic models, uses Voronoi cells to compute local density. The value after the kw gives the density at which the material becomes incompressible.*	Standard models.	
`no_rotation` :	*Forbids rotations. Not available with bonds.*	standard rotations	
`rot_elast_only` :	*Forbids rotations for non-elastic contacts (plastic contacts). Not available with bonds.*	Standard rotations.	
`constant_velocity` :	*Velocity is constant and determined from initial length* $v = l_0\dot{\epsilon}$	strain rate is constant and velocities determined from actual length $v = l\dot{\epsilon}$. Applies also to dilatation rates.	
`gravity` :	*gravity is accounted for*	No gravity.	
`thermal` :	*Solid thermal conduction may be used*	No thermal conduction	

```
#########################################################################
#                                models                                 #
#########################################################################
```

	comment	default entry if omitted	accepts multiple kw
`# elasto_plasticity and jamming key words:`	*kws attached to elasto_plasticity and jamming modes*		no
`none:`	*no specific kw*	none	
`bonds:`	*use bonds between particles (if they have been listed in the coordinated file)*	no bonds	
`no_elasticity:`	*does not consider Hertzian interactions, only plasticity for non-bonded contacts*	Hertzian interactions considered	
`linear_elasticity:`	*Elastic repulsion is linear for non-bonded contacts*	Hertzian interaction	
`# bond key words:`	*define kws for bonds. A bond model must be defined in the list:* ***`large_bonds_full`****,* ***`large_bonds`****,* ***`stiffness`*** *or* ***`clump_cluster`***		yes
`large_bonds_full :`	*Standard bond model described in [7,24]. May take bond interactions into account (cf* ***`psi_bar`*** *parameter).*	none	
`large_bonds :`	*Older bond model described in [23, 36]. No bond interactions.*	none	
`geom:`	*The contact size is defined by the geometric intersection of the two particles.*	Contact type (`geom` or `Coble`) defined in the coordinate file	
`toughness :`	*Uses the toughness model for fracture of bonds [14]. Needs bond toughness in material parameters*	Bond strength explicitly defined (see bond strength)	
`only_bonds:`	*Particles that are not bonded together do not transmit contact forces*	Hertzian contacts between non-bonded particles	
`clump_cluster :`	*bonded particles are clumped together. No force or fracture inside the clump.*	none	
`beam :`	*Resisting moments are used to compute the bond stress and to compare with bond strength [24]*	Bond stress computed without resisting moments, typically when using the toughness model [14]	
`stiffness :`	*Linear stiffness explicitly defined for bonds, see bond stiffnesses (tension and shear) in materials*	Young's moduli and Poisson's ratio used to compute bond stiffness	
`Rankine :`	*Rankine criterion is used for bond failure [26]:* $\sigma_{Rankine} = \frac{1}{2}\left(\sigma_N + \sqrt{\sigma_N^2 + 4\sigma_T^2}\right) >$ ***`sig_N`****, needs* ***`sig_N`*** *in the material properties.*	Bond fractures when either normal or tangential bond stress > `sig_N` or `sig_T`.	
`iso_bonds= :`	*All bonds, whatever their geometry are affected with the same a/R value iso_bonds*	Bond size computed from the actual geometry	
`psi_bar= :`	*Value of $\bar{\Psi}$ in Eq.(A5) of [7]. When a $\bar{\Psi}$ value is given, bond interactions are not taken into account explicitly. The standard value is $\bar{\Psi} = 0$.*	$\bar{\Psi}$ calculated with the full model (Eq. (15) of [7].	
`unload_stiff_ratio= :`	*The normal stiffness of a unloading bond is* ***`unload_stiff_ratio`*** $\times$ *the loading stiffness (defined by material parameters).*	Normal stiffness of a unloading bond is equal to the loading stiffness.	
`strength_deviation= :`	*Generates a Gaussian distribution of bond strength with mean value dictated by* ***`sig_N`*** *or* ***`toughness`*** *(see material parameters) depending on the model (toughness model or standard model).* ***`strength_deviation`*** *gives the standard deviation from the mean value*	Bond strength strictly given by material parameters.	

```
##########################################################################
#                                 models                                 #
##########################################################################
```

	comment	default entry if omitted	accepts multiple kw
`# viscoplastic model:`	*defines the viscoplastic model*		no
`standard`:	*standard viscoplastic model for a constitutive eq. of the type:* $\sigma = \sigma_1 \varepsilon^M \dot{\varepsilon}^N$	standard viscoplastic model	
`oriented`:	*viscoplastic model taking crystal orientation into account. Specific to ice.* σ_1 *depends on the crystal orientation.*	standard viscoplastic model	
`# sintering model:`	*defines the sintering model*		no
`Parhami_Mc_Meeking` :	*Parhami and McMeeking sintering model [8, 9]*	Bouvard_Pan	
`Bouvard_Pan`:	*Bouvard and Pan model for sintering [15, 28, 29]*		
`viscous`:	*viscous model for sintering*	Bouvard_Pan	
`# friction model:`	*defines the friction model*		no
`Hertz_Mindlin`:	*simplified Hertz Mindlin model used. Uses Young's modulus and Poisson's ratio to compute shear stiffness in stick region. Slip is dictated by the friction coefficient (see materials, friction parameters). Direction given by accumulated tangential relative displacement at contact.*	Hertz Mindlin	
`Coulomb`:	*Contact slips always active and dictated by the friction coefficient (see materials, friction parameters). Direction given by accumulated tangential relative displacement at contact.*	Hertz Mindlin	
`shear`:	*tangential force opposes actual relative shear velocity, value dictated by the friction coefficient (see materials, friction parameters)*	Hertz Mindlin	
`# adhesion model:`	*defines the adhesion model*		no
`JKR`:	*JKR [2, 19] adhesion model used. Adhesion is dictated by the work of adhesion (see materials, work of adhesion parameters).*	JKR	
`DMT`:	*DMT [3, 19] adhesion model used. Adhesion is dictated by the work of adhesion (see materials, work of adhesion parameters).*	JKR	

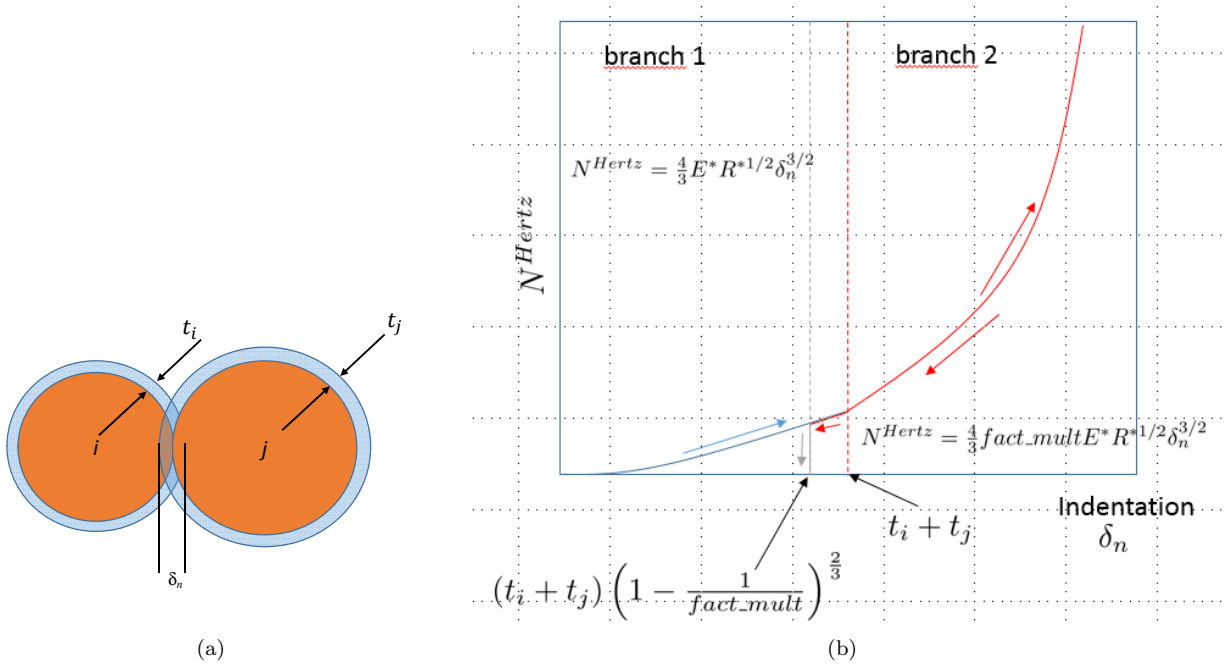
## # outputs # ##	*comment*	default entry if omitted	accepts multiple kw
# simulation termination:	*defines when simulation stops*	must be defined	fracture= can be combined
density>=	*simulation stops when relative density ≥ value*		
epsilon>=	*simulation stops when the absolute value of the strain in x, y or z directions ≥ value*		
pressure>=,<=	*simulation stops when pressure is ≥ or ≤ value (Pa)*		
epsvdot<=	*simulation stops when the volumic strain rate is ≤ value in sec^{-1}. Only used in jamming mode.*		
aoR>=	*simulation stops when the average contact size (normalized by the mean radius) is ≥ value.*		
time>=	*simulation stops when time (in seconds) is ≥ value.*		
timestep>=	*simulation stops when timestep is ≥ value (integer).*		
sig[xx, yy, zz]>=,<=	*simulation stops when the diagonal stress is ≥ or ≤ value (Pa).*		
fracture=	*simulation stops when a stress maximum (σ_{max}) has been reached and the stress ≤ value $\times\sigma_{max}$. Must be combined with another criterion.*		
# writing coordinate files:	*defines when to write _coordnnn files*		no
density=	*writing each increment (value) of relative density*	none	
epsilon=	*writing each increment (value) of strain*	none	
pressure=	*writing each increment (value) of pressure*	none	
aoR=	*writing each increment (value) of average contact size (normalized by the mean radius)*	none	
time=	*writing each increment (value) of time (seconds)*	none	
timestep=	*writing each increment (value) of timestep*	none	
sig[xx, yy, zz]=	*writing each increment (value) of sig (Pa)*	none	
none	*does not write any _coordnnn file*	none	
end	*writes a _coordnnn file only upon termination*	none	
# writing output files:	*defines when to write output files (log, tstress, zave, rupt ...)*		no
density=	*writing each increment (value) of relative density*	none	
epsilon=	*writing each increment (value) of strain*	none	
pressure=	*writing each increment (value) of pressure*	none	
aoR=	*writing each increment (value) of average contact size (normalized by the mean radius)*	none	
time=	*writing each increment (value) of time (seconds)*	none	
timestep=	*writing each increment (value) of timestep*	none	
sig[xx, yy, zz]=	*writing each increment (value) of sig (Pa)*	none	
none	*does not write any _coordnnn file*	none	
end	*writes a _coordnnn file only upon termination*	none	
# writing contact history files:	*defines when to write nnnn_histc and nnnn_clumps files*		no
density=	*writing each increment (value) of relative density*	none	
epsilon=	*writing each increment (value) of strain*	none	
pressure=	*writing each increment (value) of pressure*	none	
aoR=	*writing each increment (value) of average contact size (normalized by the mean radius)*	none	
time=	*writing each increment (value) of time (seconds)*	none	
timestep=	*writing each increment (value) of timestep*	none	
sig[xx, yy, zz]=	*writing each increment (value) of sig (Pa)*	none	
none	*does not write any _coordnnn file*	none	
end	*writes a _coordnnn file only upon termination*	none	

```
#########################################################################
#                                loadings                               #
#########################################################################
```

	comment	default entry if omitted	accepts multiple kw
`# periodic conditions:`	*defines periodic conditions in the simulation box*		no
`1x1y1z`	*periodic conditions on all three axis*	1x1y1z	
`1x1y0z`	*no periodic conditions on z axis*	1x1y1z	
`1x0y0z`	*periodic conditions only on x axis*	1x1y1z	
`...`	*etc ...*	1x1y1z	
`# loading conditions (Pa for stress):`	*defines the loading conditions. When a stress is imposed it takes predominance over imposed strain-rates (Strain-rates still must be given to help the PID controller).*		yes
`epsxdot=`	*imposed strain-rate on axis x (in 1/sec)*	0.	
`epsydot=`	*imposed strain-rate on axis y (in 1/sec)*	0.	
`epszdot=`	*imposed strain-rate on axis z (in 1/sec)*	0.	
`sigxx=`	*imposed σ_{xx} (in Pa)*	no imposed stress	
`sigyy=`	*imposed σ_{yy} (in Pa)*	no imposed stress	
`sigyy=`	*imposed σ_{zz} (in Pa)*	no imposed stress	
`equal_stress=`	*when imposing stress, special controls. See compute_strnx_nstrn.f90*	0	
`pressure=`	*imposed pressure (in Pa). Only used in jamming.*	no imposed pressure	
`# imposed disp. (i,m/s), rotation (i,0.), force (i,N/s) on part i:`	*defines the particles on which increment of displacement (in m/s) or increment of force (in N/s) can be imposed*	can be entirely omitted if no imposed displacement nor force is sought for	yes
`none`	*no imposed displacement or force*		
`deltax=10,0.`	*0 imposed disp for particle 10 on x*	no imposed displacement	
`deltay=11,1.E-12`	1.E^{-12} m.s^{-1} *imposed disp for particle 11 on y*	no imposed displacement	
`deltaz=12,-1.E-12`	-1.E^{-12} m.s^{-1} *imposed disp for particle 12 on y*	no imposed displacement	
`deltar=12,0.`	*zero imposed rotation for particle 12*	no imposed rotation	
`deltaFx=2,1.E-12`	1.E^{-12} N.s^{-1} *imposed force increment for particle 2 on x*	no imposed force	
`deltaFy=3,1.E-12`	1.E^{-12} N.s^{-1} *imposed force increment for particle 3 on y*	no imposed force	
`deltaFz=2,1.E-12`	1.E^{-12} N.s^{-1} *imposed force increment for particle 2 on z*	no imposed force	
`# temperatures(deg C) and duration (s) for stage`	*defines the temperature profile to be imposed*	Used only for sintering for the time being.	yes
`none`	*no imposed temperature*	none	
`T_init=`	*initial temperature of stage, should be equal to final temperature of preceding stage*	0.	
`T_final=`	*final temperature of stage, should be equal to initial temperature of preceding stage*	0.	
`time=`	*duration of stage in sec. A zero value means that no time limit is imposed for this stage*	0.	
`# Properties of marked particles (Temp in deg C):`	*Defines properties to impose to marked particles (marked with + in the particle name of the coordinate file)*	No specific property imposed even if some particles are marked	yes
`fixed_Temp=`	*marked particles have a fixed temperature*	none	
`surface_Temp=`	*marked particles are in contact with a surface with a given temperature*	none	

```
########################################################################
#                materials (from 0,1,2, ...  to 9)                     #
########################################################################
```

	comment	default entry if omitted	accepts multiple kw
`# elastic parameters (Pa for stress):`	*Elastic properties*	Young's modulus of material 1 must be defined	yes
`E(1)`	*Young's modulus of material 1*	0.D0	
`poisson(1)`	*Poisson's ratio of material 1*	0.D0	
`E(2)`	*Young's modulus of material 2*	0.D0	
`poisson(2)`	*Poisson's ratio of material 2*	0.D0	
`delta_c(1)`	*indentation (normalized by R) at which E(1) is multiplied by fact_mult(1)*	0.D0	
`fact_mult(1)`	*factor by which E(1) is multiplied (see Fig. 30)*	1.D0	
`# plastic parameters (Pa for stress):`	*Plastic properties*		yes
`sigy(1)=`	σ_1 *of material 1 for a material with constitutive equation:* $\sigma = \sigma_1 \varepsilon^M \dot{\varepsilon}^N$	10^{33}	
`sigy(2)=`	σ_2 *of material 2 for a material with constitutive equation:* $\sigma = \sigma_2 \varepsilon^M \dot{\varepsilon}^N$	10^{33}	
`Mstrain=`	*hardening coefficient for all materials:* $\sigma = \sigma_{1,2} \varepsilon^{Mstrain} \dot{\varepsilon}^N$	0.D0	
`Nvisco=`	*creep indicia for all materials:* $\sigma = \sigma_{1,2} \varepsilon^M \dot{\varepsilon}^{Nvisco}$	0.D0	
`# friction parameters:`	*friction coefficients, see models for the definition of the associated friction model. In sintering mode (not viscous model),* `frict` *is the* μ *parameter in Eq. (50)*		yes
`frict(1,1)=`	*friction coefficient between materials 1 and 1*	0.D0	
`frict(1,2)=`	*friction coefficient between materials 1 and 2 (=frict(2,1))*	0.D0	
`frict(1,1)=`	*friction coefficient between materials 1 and 1*	0.D0	
`frict(2,2)=`	*friction coefficient between materials 2 and 2*	0.D0	
`# work of adhesion parameters (J.m-2):`	*work of adhesion, see models for the definition of the associated adhesion model [11]*		yes
`adhes(1,1)=`	*work of adhesion between materials 1 and 1*	0.D0	
`adhes(1,2)=`	*work of adhesion between materials 1 and 2 (=adhes(2,1))*	0.D0	
`adhes(1,1)=`	*work of adhesion between materials 1 and 1*	0.D0	
`adhes(2,2)=`	*work of adhesion between materials 2 and 2*	0.D0	
`# bond toughness parameters (J.m-2):`	*toughness of a bond in the toughness model for the fracture of bonds, see models definition [14]*		yes
`toughness(1,1)=`	*bond toughness between materials 1 and 1*	0.D0	
`toughness(1,2)=`	*bond toughness between materials 1 and 2 (=toughness(2,1))*	0.D0	
`toughness(1,1)=`	*bond toughness between materials 1 and 1*	0.D0	
`toughness(2,2)=`	*bond toughness between materials 2 and 2*	0.D0	
`# bond stiffnesses (tension and shear) (Pa):`	*stiffness of a bond in the stiffness model [37]*		yes
`K_N(1)=`	*normal bond stiffness for material 1*	0.D0	
`K_N(2)=`	*normal bond stiffness for material 2*	0.D0	
`K_T(1)=`	*tangential bond stiffness for material 1*	0.D0	
`K_T(2)=`	*tangential bond stiffness for material 2*	0.D0	
`# bond strength in tension (Pa):`	*bond strength in tension for the fracture of bonds. Default model (not toughness)*		yes
`sig_N(1,1)=`	*strength in tension between materials 1 and 1*	0.D0	
`sig_N(1,2)=`	*strength in tension between materials 1 and 2 (=sig_N(2,1))*	0.D0	
`sig_N(1,1)=`	*strength in tension between materials 1 and 1*	0.D0	
`sig_N(2,2)=`	*strength in tension between materials 2 and 2*	0.D0	



(a) (b)

Figure 30: (a) Two elastic particles coated with thickness $t_i = delta\_c \times R_i$ and $t_j = delta\_c \times R_j$. (b) When the indentaion reaches the critical indentation $\delta_n \geq \delta_c = t_i + t_j$, the elastic moduli of particles i and j are multiplied by $fact\_mult$ and the contact enters branch 2. Upon unloading of a contact that has entered branch 2, when the condition $\delta_n \leq \delta_c^u = (t_i + t_j)\left(1 - \frac{1}{fact\_mult}\right)^{\frac{2}{3}}$ applies, the contact fails and transmits no normal force. See elastic properties.

```
#########################################################################
#              materials (from 0,1,2, ...  to 9)                        #
#########################################################################
```

	comment	default entry if omitted	accepts multiple kw
`# sintering material(s):`	*Lists materials that sinter. See Yan et al. [16] for interactions between sintering and non-sintering particles.*	no sintering material	yes
`1`	*material 1 sinters*	material 1 does not sinter	
`2`	*material 2 sinters*	material 2 does not sinter	
`# surface energy parameters for sintering (J.m-2):`	*surface energies for sintering contacts [15, 28, 29]*		yes
`gamma_s(1,1)=`	*surface energy between materials 1 and 1*	0.D0	
`gamma_s(1,2)=`	*surface energy between materials 1 and 2 (=gamma_s(2,1))*	0.D0	
`gamma_s(1,1)=`	*surface energy between materials 1 and 1*	0.D0	
`gamma_s(2,2)=`	*surface energy between materials 2 and 2*	0.D0	
`# diffusion parameter (m3.s-1):`	*diffusion parameter for sintering contacts. Objects cannot sinter.*		yes
`DeltabD0b(1,1)=`	*diffusion parameter between materials 1 and 1*	0.D0	
`DeltabD0b(1,2)=`	*diffusion parameter between materials 1 and 2 (=DeltabD0b(2,1))*	0.D0	
`DeltabD0b(1,1)=`	*diffusion parameter between materials 1 and 1*	0.D0	
`DeltabD0b(2,2)=`	*diffusion parameter between materials 2 and 2*	0.D0	
`# pre-exponential viscosity coefficient (Pa.s):`	*pre-exponential viscosity coefficient for the viscous model [30, 31]*		yes
`eta_0(1,1)=`	*pre-exponential viscosity coefficient between materials 1 and 1*	0.D0	
`eta_0(1,2)=`	*pre-exponential viscosity coefficient between materials 1 and 2 (=eta_0(2,1))*	0.D0	
`eta_0(1,1)=`	*pre-exponential viscosity coefficient between materials 1 and 1*	0.D0	
`eta_0(2,2)=`	*pre-exponential viscosity coefficient between materials 2 and 2*	0.D0	
`# activation energy (KJ/mol):`	*Activation energy for diffusion or viscosity.*		yes
`Qb(1,1)=`	*Activation energy for diffusion or viscosity between materials 1 and 1*	0.D0	
`Qb(1,2)=`	*Activation energy for diffusion or viscosity between materials 1 and 2 (=Qb(2,1))*	0.D0	
`Qb(1,1)=`	*Activation energy for diffusion or viscosity between materials 1 and 1*	0.D0	
`Qb(2,2)=`	*Activation energy for diffusion or viscosity between materials 2 and 2*	0.D0	
`# dihedral angle (°):`	*Dihedral angle [8, 9]*		yes
`chi(1,1)=`	*Dihedral angle between materials 1 and 1*	180	
`chi(1,2)=`	*Dihedral angle between materials 1 and 2 (=chi(2,1))*	180	
`chi(1,1)=`	*Dihedral angle between materials 1 and 1*	180	
`chi(2,2)=`	*Dihedral angle between materials 2 and 2*	180	
`# atomic volume (m3) and coarsening parameter:`	*atomic volume (m3) and coarsening parameter for sintering models. Coarsening model is described in [9]*		yes
`omega(1)=`	*atomic volume for material 1*	0.D0	
`omega(2)=`	*atomic volume for material 2*	0.D0	
`coarsen(1)=`	*coarsening parameter for material 1*	0.D0	
`coarsen(2)=`	*coarsening parameter for material 2*	0.D0	

```
########################################################################
#              materials (from 0,1,2, ...  to 9)                       #
########################################################################
```

	comment	default entry if omitted	accepts multiple kw
`# dilatation parameter (s-1):`	*Particle radius (R) may dilate according to:* $\frac{1}{R}\frac{dR}{dt} = dilate(material)$. *If constant_velocity is given (see general key words), initial radius* R_0 *is used:* $\frac{1}{R_0}\frac{dR}{dt} = dilate(material)$		yes
`dilate(1)=`	*dilatation rate for material 1*	0.D0	
`dilate(2)=`	*dilatation rate for material 2*	0.D0	
`# density (g.mm-3):`	*Material density. Density is multiplied by an upscaling factor (see upscale in numerics) for quasi-static conditions*		yes
`ro(1)=`	*density for material 1*	7.89D-03	
`ro(2)=`	*density for material 2*	7.89D-03	
`# dissipation parameter (s):`	*Dissipation. Model is from Pöschel [38]. Applies to the normal component of the Hertzian elastic model of contact.*		yes
`dissip(1,1)=`	*dissipation between materials 1 and 1*	0.D0	
`dissip(1,2)=`	*dissipation between materials 1 and 2 (=dissip(2,1))*	0.D0	
`dissip(1,1)=`	*dissipation between materials 1 and 1*	0.D0	
`dissip(2,2)=`	*dissipation between materials 2 and 2*	0.D0	
`# Thermal expansion:`	*Linear thermal expansion coefficient*		yes
`T_thermal(1)=`	*Thermal dilatation for material 1*	0.D0	
`T_thermal(2)=`	*Thermal dilatation for material 2*	0.D0	

The list given here is limited to two materials (1 and 2). However `dp3D` can handle 10 different materials ranging from material number 0 to 9.

## # numerics # ##	*comment*	default entry if omitted	accepts multiple kw
# affine motion conditions:	*Defines on which axis, affine motion is imposed on particles in the first half time-step*		yes
1x1y1z	*affine conditions on all three axis*	1x1y1z	
1x1y0z	*affine conditions only on x and y axis*	1x1y1z	
1x0y0z	*affine conditions only on x axis*	1x1y1z	
...	*etc ...*	1x1y1z	
# control of strain-rates for quasi-static conditions:	*Defines the type of control to ensure quasi-static conditions. Multiple conditions can be defined. Imposed strain-rate is calculated from the condition and from ctrl_fact factor. Actual strain-rate $\dot{\epsilon}$ is bounded by $\left(\frac{\dot{\epsilon_0}}{ctrl\_fact}, \dot{\epsilon_0} \times ctrl\_fact\right)$ where $\dot{\epsilon_0}$ is defined in the loading conditions.*		yes
none	*Strain-rates strictly given by loading conditions*		
aoamean=	*strain-rates are divided by ctrl_fact when a bond of size $\frac{a}{a_{mean}}$ has broken a_{mean} is the mean bond size.*	none	
break=	*strain-rates are divided by ctrl_fact when $\sigma_b > (1. - break) \times sigma_b^c$, where σ_b is the maximum stress on the bond (tensile or shear) and $sigma_b^c$ is the strength of the bond (see materials). Condition is released after bond breakage. This condition is best used with the kin_energy condition to ensure that strain-rates are not too high after bond fracture.*	none	
epsilon=	*strain-rates are divided by ctrl_fact when any strain is greater than epsilon.*	none	
ferror=	*A proportional controller is used to control strain rates $\dot{\epsilon}$: $\dot{\epsilon} = \dot{\epsilon} \times \frac{ferror}{ferror_{actual}}$. The strain-rate multiplier is in the interval $\left[\frac{1}{ctrl\_fact}, ctrl\_fact\right]$*	none	
kin_energy=	*A proportional controller is used to control strain rates $\dot{\epsilon}$: $\dot{\epsilon} = \dot{\epsilon} \times \frac{kin\_energy}{kin\_energy_{actual}}$. The strain-rate multiplier is in the interval $\left[\frac{1}{ctrl\_fact}, ctrl\_fact\right]$*	none	
vmax=	*A proportional controller is used to control strain rates $\dot{\epsilon}$: $\dot{\epsilon} = \dot{\epsilon} \times \frac{vmax}{vmax_{actual}}$. The strain-rate multiplier is in the interval $\left[\frac{1}{ctrl\_fact}, ctrl\_fact\right]$*	none	
ctrl_fact=	*The multiplier or the dividing factor used for the strain-rate controllers described above.*	none	
# numerical parameters:	*numerical parameters*	none	yes
safe_dt=	*Safety factor which multiplies the standard time step value calculated (typical value should be in between 0.001 and 0.1)*	5.10^{-3}	
upscale(1)=	*If set to 1., dp3D computes the upscaling. If set to zero, no upscaling is used. If $\neq$ from 0 or 1, the dp3D upscaling is $\times$ by upscale(1).*	1.	
damping=	*Numerical damping, Typical value is 0.05.*	0.05	
fixed_dt=	*User-imposed timestep in second.*	dp3D computes the timestep	
random_seed=	*Imposes a random seed (integer) for distributions (see oriented viscoplastic model or strength_deviation of bond strength).*	dp3D chooses the random seed	
potential_contact=	*Imposes the multiplying factor for particle radii to list potential contacts. Should be larger than 1.*	dp3D computes the multiplying factor	
aff_fact=	*When using affine conditions, multiply the affine displacement by aff_fact*	1.	
P_fact=, I_fact=, D_fact=	*When calling for a stress controlled test, multiply the standard dp3D calculated P, I and D parameters of the PID controller by P_fact, I_fact, D_fact, respectively*	1., 1., 1.	

7 dp3D on linux

7.1 dp3D and parallel computation

Discrete element simulations can be CPU demanding and dp3D is no exception in that regard. The code has been parallelized with a fine-grained method (at the loop level) with openMP directives. The scalability is far from fully satisfactory as shown by Fig. 31. Depending on the architecture of the shared memory, you might choose to use 2, 4, 6, 8, 12 or even 16 processes. Here four exemples are shown:

- 2×4 cores (typically machine richebourg, Intel 5560, 2009).
- 2×6 cores (typically machine charlemagne, Intel 5660, 2010), not hyperthreaded.
- cluster on nodes with 12 cores (typically ZEBULON cores, Intel 5660, 2012).
- cluster on nodes with 8 cores (typically DP3D cores, Intel Xeon, 2009).

Fig. 31 shows that depending on the available number of processes, the machine used and the number of simulations to be run, an optimized number of processes can be found.

7.2 Some useful tips and commands

Since some simulations can be long, and one may not have properly set the interval between two coordinate file writes or two outputs for the result files (see `simulation termination` conditions in 6), it is useful to be able to force dp3D to write these files *during a simulation*. This is invoked by the commands:

`dp3D -write_log`	writes into the result files (`log tstress zave ...`)
`dp3D -write_coord`	writes a coordinate file (and a _histc)
`dp3D -clean_stop`	kills 'cleanly' the simulation, writes a coordinate file and a last line in the result files

which must be executed in the directory where dp3D is running.

A more direct way to kill a dp3D job that is running with PID nnnn, is to invoke the command:

```
kill -s 2 nnnn
```

that will send a signal to dp3D to write a coord file before killing the job. kill signal works when using the

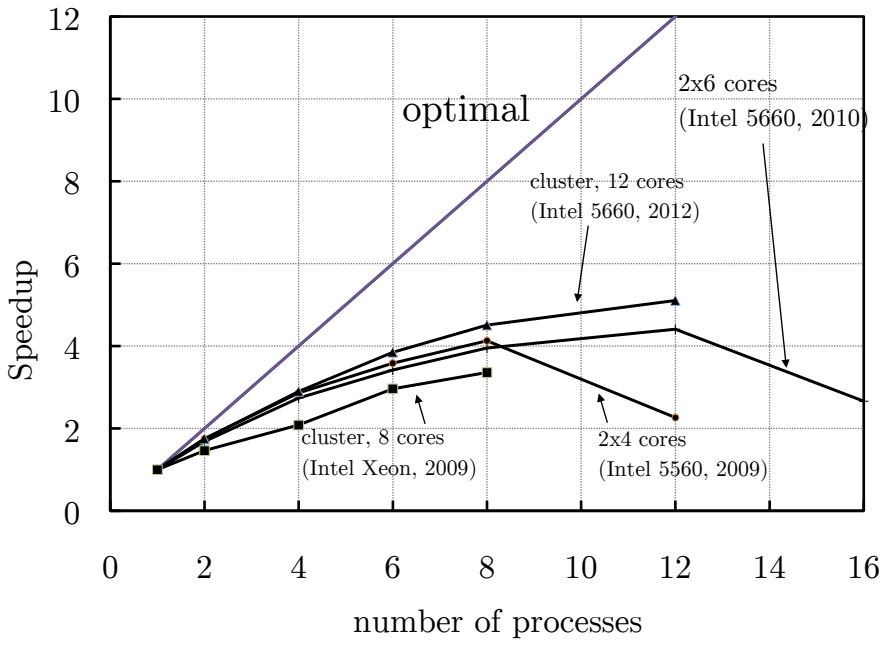


Figure 31: Scalability of a typical calculation (with bonds) of dp3D for three types of architectures.

`cdp3D -gas` command. When working on qsub environment on a cluster, the command becomes:
`qsig -s 2 nnnn`

The command: `mydp3D` tells in which directory and the PID of the current `dp3D` jobs of the user on the machine.

The command: `lastdp3D -nn` tells the directory of the `dp3D` jobs that ran in the last nn hours. If nn is not provided, the last 48 hours are looked for.

The command: `dp3D -v` gives the version of the dp3D executable which is running.

`rasmol` is an old but useful visualization tool. It is a 32 bit executable. To make it run on a 64 bit machine you need to install the following packages:

```
sudo apt-get install gcc-multilib
sudo apt-get install libxrender1:i386 libxtst6:i386 libxi6:i386
```

References

[1] P A Cundall and O D L Strack. A discrete numerical model for granular assemblies. *Géotechnique*, 29:47–65, 1979. 5

[2] K L Johnson, K Kendall, and A D Roberts. Surface energy and the contact of elastic solids. *Proc. R. Soc. Lond. A*, A324:301–313, 1971. 5, 23, 90

[3] B V Derjaguin, V M Muller, and Y P Toporov. Effect of contact deformations on adhesion of particles. *J. Colloid Interface Sci.*, 53:314–326, 1975. 5, 15, 23, 90

[4] B Storakers, N A Fleck, and R M McMeeking. The viscoplastic compaction of composite powders. *J. Mech. Phys. Solids*, 47:785–815, 1999. 5, 18, 64

[5] S Dj. Mesarovic and K L Johnson. Adhesive contact of elastic-plastic spheres. *J. Mech. Phys. Solids*, 48:2009–2033, 2000. 5, 23, 24

[6] G Jefferson, G K Haritos, and R M McMeeking. The elastic response of a cohesive aggregate - a discrete element model with coupled particle interaction. *J. Mech. Phys. Solids*, 50:2539–2575, 2002. 5

[7] David Jauffrès, Christophe L Martin, Aaron Lichtner, and Rajendra K Bordia. Simulation of the elastic properties of porous ceramics with realistic microstructure. *Modell. Simul. Mater. Sci. Eng.*, 20(4):45009, jun 2012. 5, 6, 38, 89

[8] F Parhami and R M McMeeking. A network model for initial stage sintering. *Mech. Mater.*, 27:111–124, 1998. 5, 55, 90, 95

[9] C L Martin, L C R Schneider, L Olmos, and D Bouvard. Discrete element modeling of metallic powder sintering. *Scripta Mater.*, 55:425–428, 2006. 5, 55, 90, 95

[10] Brayan Paredes-Goyes, David Jauffres, Jean-Michel Missiaen, and Christophe L. Martin. Grain growth in sintering: a discrete element model on large packings. *Acta Materialia*, 218:117182, 2021. 5, 6

[11] C L Martin and R K Bordia. Influence of adhesion and friction on the geometry of packings of spherical particles. *Phys. Rev. E*, 77:31307, 2008. 6, 93

[12] C L Martin. Elasticity, fracture and yielding of cold compacted metal powders. *J. Mech. Phys. Solids*, 52:1691–1717, 2004. 6

[13] Xiaoxing Liu, Christophe L. Martin, Gérard Delette, and Didier Bouvard. Elasticity and strength of partially sintered ceramics. *J. Mech. Phys. Solids*, 58(6):829–842, jun 2010. 6, 33, 36

[14] David Jauffres, Christrophe L Martin, Aaron Lichtner, Rajendra K Bordia, D Jauffrès, Christrophe L Martin, Aaron Lichtner, and Rajendra K Bordia. Simulation of the toughness of partially sintered ceramics with realistic microstructures. *Acta Mater.*, 60:4685–4694, 2012. 6, 34, 38, 40, 89, 93

[15] C L Martin and R K Bordia. The effect of a substrate on the sintering of constrained films. *Acta Mater.*, 57:549–558, 2009. 6, 55, 90, 95

[16] Z Yan, C L Martin, O Guillon, and D Bouvard. Effect of size and homogeneity of rigid inclusions on the sintering of composites. *Scripta Mater.*, 69:327–330, 2013. 6, 58, 95

[17] Alexander Stukowski. Visualization and analysis of atomistic simulation data with OVITO - the Open Visualization Tool. *Modelling and Simulation in Materials Science and Engineering*, 18(1):015012, jan 2010. 8, 66

[18] K L Johnson. *Contact Mechanics*. Cambridge University Press, 1985. 15, 18

[19] E Barthel. Adhesive elastic contacts - JKR and more. *J. Phys. D: Appl. Phys.*, 41:163001, 2008. 15, 90

[20] C H Rycroft. VORO++: A three-dimensional Voronoi cell library. *CHAOS*, 19:41111, 2009. 23, 29, 30, 68, 71

[21] C L Martin. Unloading of powder compacts and their resulting tensile strength. *Acta Mater.*, 51:4589–4602, 2003. 24

[22] B Harthong, J.-F. Jérier, P Dorémus, D Imbault, and F.-V. Donzé. Modeling of high-density compaction of granular materials by the Discrete Element Method. *International Journal of Solids and Structures*, 46:3357–3364, 2009. 29, 30

[23] C L Martin, D Bouvard, and G Delette. Discrete Element simulations of the Compaction of aggregated ceramic powders. *J. Am. Ceram. Soc.*, 89:3379–3387, 2006. 33, 89

[24] P Pizette, C L Martin, G Delette, F Sans, and T Geneves. Green strength of binder-free ceramics. *J. Eur. Ceram. Soc.*, 33:975–984, 2013. 34, 89

[25] P Parant, E Remy, S Picart, J P Bayle, E Brackx, A Ayral, T Delahaye, and C.L. L Martin. Mechanical behaviour of porous lanthanide oxide microspheres : Experimental investigation and numerical simulations. *Journal of the European Ceramic Society*, 38(2):695–703, 2018. 34

[26] Damien André, Mohamed Jebahi, Ivan Iordanoff, Jean-luc Charles, and Jérôme Néauport. Using the discrete element method to simulate brittle fracture in the indentation of a silica glass with a blunt indenter. *Computer Methods in Applied Mechanics and Engineering*, 265:136–147, 2013. 34, 89

[27] Rishi Kumar, Sarshad Rommel, David Jauffrès, Pierre Lhuissier, and Christophe L. Martin. Effect of packing characteristics on the discrete element simulation of elasticity and buckling. *International Journal of Mechanical Sciences*, 110:14–21, 2016. 43, 49

[28] D Bouvard and R M McMeeking. The deformation of interparticle necks by diffusion controlled creep. *J. Am. Ceram. Soc.*, 79:666–672, 1996. 55, 60, 90, 95

[29] J Pan, H Le, S Kucherenko, and J A Yeomans. A model for the sintering of spherical particles of different sizes by solid state diffusion. *Acta Mater.*, 46:4671–4690, 1998. 55, 60, 61, 90, 95

[30] Anand Jagota, KR Mikeska, and RK Bordia. Isotropic constitutive model for sintering particle packings. *Journal of the American Ceramic Society*, 73(8):2266–2273, 1990. 55, 56, 95

[31] Anand Jagota and George W Scherer. Viscosities and Sintering Rates of Composite Packings of Spheres. *Journal of the American Ceramic Society*, 78(3):521–528, 1995. 55, 56, 95

[32] C L Martin, H Camacho-Montes, L Olmos, D Bouvard, and R K Bordia. Evolution of Defects During Sintering: Discrete Element Simulations. *J. Am. Ceram. Soc.*, 92:1435–1441, 2009. 56

[33] J Weber. Recherches concernant les contraintes intergranulaires dans les milieux pulvérulents. *Bulletin de liaison des Ponts et Chaussées*, pages 1–20, 1966. 78, 83

[34] J Christoffersen, M M Mehrabadi, and S Nemat-Nasser. A micromechanical description of granular material behavior. *Journal of Applied Mechanics*, 48:339–344, 1981. 78, 83

[35] P Dantu. Etude statistique des forces intergranulaires dans un milieu pulv{é}rulent. *G{é}otechnique*, 18:50–55, 1968. 83

[36] Xiaoxing Liu, Christrophe L. Martin, Didier Bouvard, Stéphane Di Iorio, Jérome Laurencin, Gérard Delette, S Di Iorio, Jérome Laurencin, and Gérard Delette. Strength of Highly Porous Ceramic Electrodes. *J. Am. Ceram. Soc.*, 94(10):3500–3508, oct 2011. 89

[37] L. Hedjazi, C. L. Martin, S. Guessasma, G. Della Valle, and R. Dendievel. Experimental investigation and discrete simulation of fragmentation in expanded breakfast cereals. *Food Research International*, 55:28–36, 2014. 93

[38] H Kruggel-Emden, E Simsek, S Rickelt, S Wirtz, and V Scherer. Review and extension of normal force models for the Discrete Element Method. *Powder Technol.*, 171:157–173, 2007. 96

Index