LMGC90v2_Pre : le pré-processeur de LMGC90

A. Martin, M. Bagnéris, F. Dubois, R. Mozul

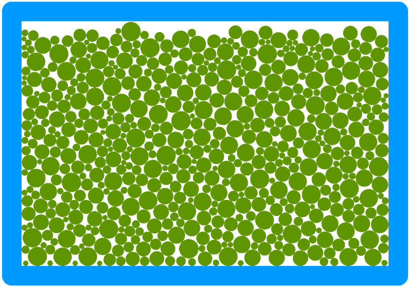
Laboratoire de Mécanique et Génie Civil

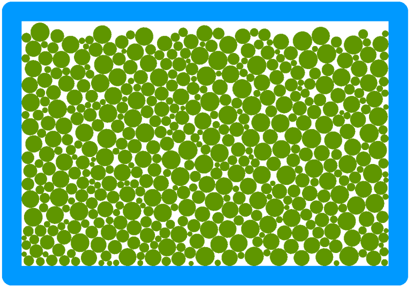
Formation LMGC90 - janvier 2013

Sommaire

Contraintes induites par une analyse statistique

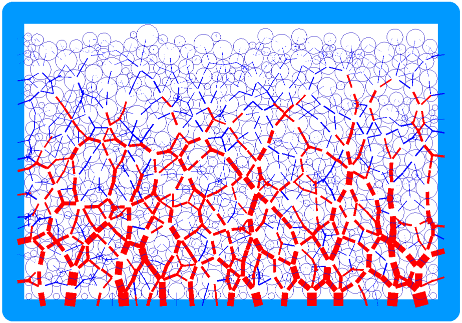
1. maîtrise de la représentativité statistique des échantillons
   $\Rightarrow$ maîtrise de la *granulométrie*,





Contraintes induites par une analyse statistique

1. maîtrise de la représentativité statistique des échantillons
   $\Rightarrow$ maîtrise de la *granulométrie*,
2. maîtrise des paramètres de textures (*e.g.* compacité, anisotropie des forces de contact)
   $\Rightarrow$ état initial obtenu par un *pré-calcul* (*e.g.* relaxation sous gravité, compression isotrope),



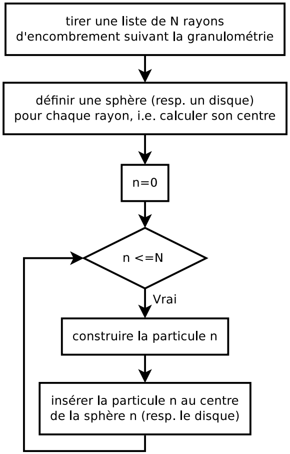
Contraintes induites par une analyse statistique

1. maîtrise de la représentativité statistique des échantillons
   $\Rightarrow$ maîtrise de la *granulométrie*,
2. maîtrise des paramètres de textures (*e.g.* compacité, anisotropie des forces de contact)
   $\Rightarrow$ état initial obtenu par un *pré-calcul* (*e.g.* relaxation sous gravité, compression isotrope),
3. influence de la forme des particules sur les paramètres de textures
   $\Rightarrow$ choix du *type* et des *paramètres* caractérisant la forme des particules.

Etat initial du pré-calcul

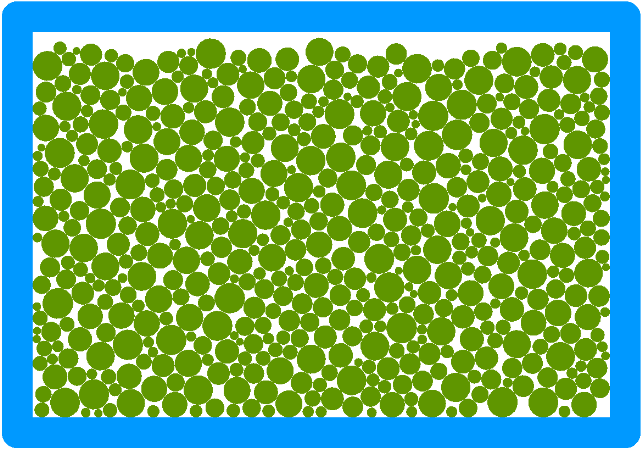
- rôle du pré-processeur : fournir une géométrie initiale adaptée au pré-calcul,
- idée : utiliser une méthode de répartition spatiale des particules, ou *méthode de dépôt*,
- principe :
  - calcul des coordonnées des centres de sphères (ou disques, en 2D) caractérisées par leur rayon,
  - insertion des particules dans les cavités sphériques (ou circulaires, en 2D),
  - particules non sphériques gérées en utilisant leur rayon d'encombrement,
- deux types de méthodes :
  - méthodes géométriques minimisant un potentiel (*e.g.* l'énergie potentielle de pesanteur),
  - méthodes de dépôt sur un réseau.

Algorithme générique de construction d'un échantillon granulaire



Exemple de dépôt géométrique minimisant l'énergie potentielle de pesanteur

- échantillon polydisperse de disques dans une boîte



Echantillon de disques : dépôt des particules

```
# imports, declarations (modeles, materiaux, conteneurs)

# parametres du script
#   * nombre de particules
nb_particles=1000
#   * taille de la boite (75 cm x 50 cm)
lx=0.75; ly=0.5

# tirage aleatoire des rayons entre 0.5 et 2 cm
radii=granulo_Random(nb_particles, 5.e-3, 2.e-2)

# depot dans une boite rectangulaire
[nb_part_in_box, coor]=depositInBox2D(radii, lx, ly)
```

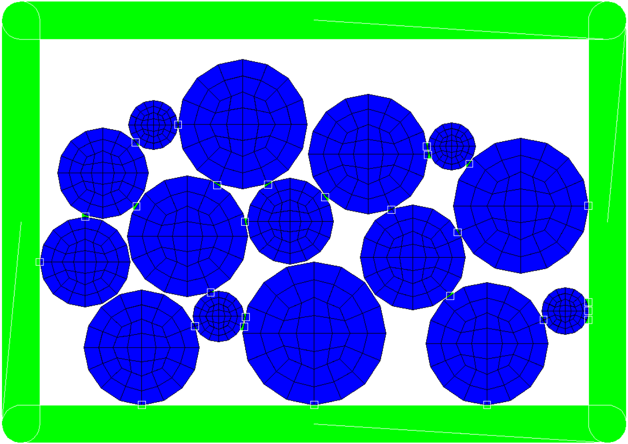
Echantillon de disques : boucle de génération des particules

```
# boucle d'ajout des particules deposees :
for i in range(0, nb_part_in_box):
   # creation d'un nouveau disque rigide
   body=rigidDisk(r=radii[i],
      center=coor[2*i : 2*(i + 1)],
      model=mR2D, material=plex, color='BLEUx')
   # ajout de la particule a l'ensemble de corps
   bodies.addAvatar(body)

# ajout des parois, definition des interactions
# (distance d'alerte calculee a partir de min(radii))
# et ecriture des fichiers
```

Exemple de dépôt géométrique minimisant l'énergie potentielle de pesanteur

- échantillon polydisperse de disques dans une boîte



Echantillon de disques : définition du matériau

```
# imports, premieres declarations (dimension, modele
# rigide)

# definition d'un modele elastique, lineaire,
# en grandes deformations
m2Dl = model(name='M2DLx', type='MECAx',
    element='Q4xxx', dimension=dim,
    external_model='yes__', kinematic='large',
    formulation='TotaL', material='neoh_',
    anisotropy='iso__', mass_storage='lump_',
    thermal_coupling='no___')

# fin des declarations (materiaux, remplissage des
# conteneurs)
```

Echantillon de disques : dépôt des particules

```
# parametres du script
#   * nombre de particules
nb_particles=1000
#   * taille de la boite (75 cm x 50 cm)
lx=0.75; ly=0.5

# tirage aleatoire des rayons entre 0.5 et 2 cm
radii=granulo_Random(nb_particles, 5.e-3, 2.e-2)

# depot dans une boite rectangulaire
[nb_part_in_box, coor]=depositInBox2D(radii, lx, ly)
```

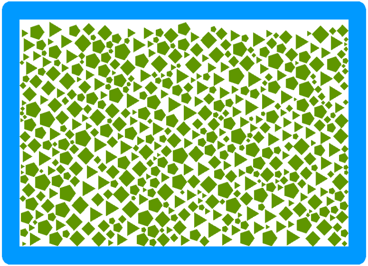
Echantillon de disques : boucle de génération des particules

```python
# boucle d'ajout des particules deposees :
for i in range(0, nb_part_in_box):
   # creation d'un nouveau disque deformable
   body=deformableParticle2D(r=radii[i],
      center=coor[2*i : 2*(i + 1)], type_part='Disk',
      model=m2Dl, material=steel, color='BLEUx')
   # ajout de la particule a l'ensemble de corps
   bodies.addAvatar(body)

# ajout des parois, definition des interactions
# (distance d'alerte calculee a partir de min(radii))
# et ecriture des fichiers
```

Méthodes de dépôt géométrique

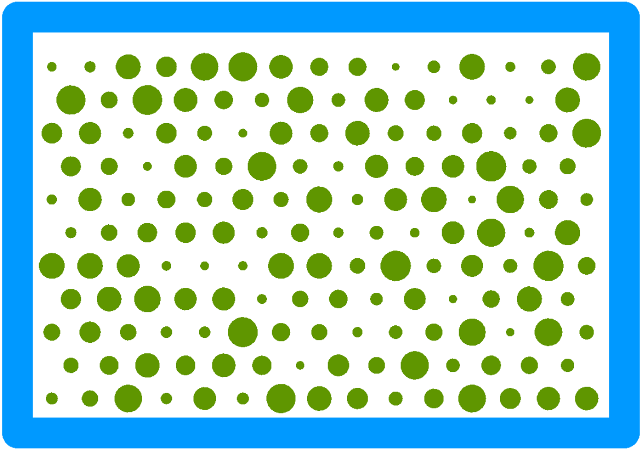
- dépôt dans différents conteneurs prédéfinis 2D (*e.g.* boîte, Couette, disque, tambour tournant) et 3D (*e.g.* boîte, sphère, cylindre),
- particules sphériques (ou circulaires en 2D) déposées au contact,



- l'anisotropie dépend du potentiel utilisé,
- nombre de particules nécessaires pour remplir le conteneur inconnu :
  - sous-estimation ⇒ conteneur pas complètement rempli,
  - sur-estimation ⇒ particules non déposées et granulométrie non respectée,
- méthode inadaptée au cas monodisperses (interpénétrations).

Exemple de dépôt sur un réseau triangulaire

- échantillon polydisperse de disques dans une boîte



Echantillon de disques : calcul du nombre de particules

```
# imports, declarations (modeles, materiaux, conteneurs)

# parametres du script
#   * taille de la boite (75 cm x 50 cm)
lx=0.75; ly=0.5
#   * taille d'un element du reseau (5 cm)
l=5.e-2
#   * choix de l'orientation du reseau
orient='up'
#   * nombre de particules sur la premiere couche
nb_pt=15
#   * nombre de couches
nb_layer=20

# calcul du nombre de particules sur le reseau
nb_particles=nbPointsInTriangularLattice2D(nb_ele=nb_pt,
    nb_layer=nb_layer, orientation=orient)
```

Echantillon de disques : boucle de génération des particules

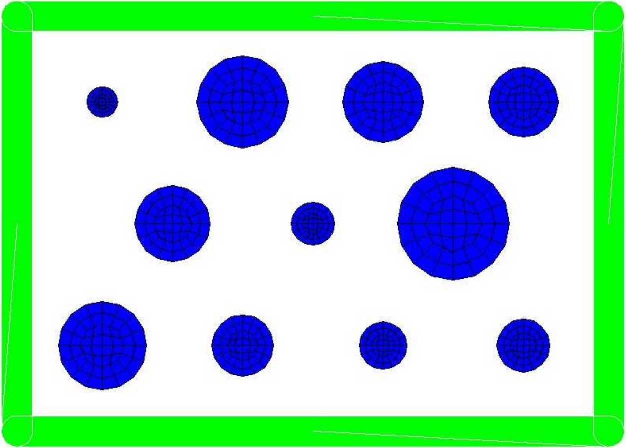
```
# tirage aleatoire des rayons entre 0.5 et 2 cm
radii=granulo_Random(nb_particles, 5.e-3, 2.e-2)
# depot des particules sur le reseau
coor=triangularLattice2D(nb_ele=nb_pt, l=l, x0=0.,
   y0=0., nb_layer=nb_layer, orientation=orient)

# boucle d'ajout des particules deposees :
for i in range(0, nb_particles):
   # creation d'un nouveau disque rigide
   body=rigidDisk(r=radii[i],
      center=coor[2*i : 2*(i + 1)],
      model=mR2D, material=plex, color='BLEUx')
   # ajout de la particule a l'ensemble de corps
   bodies.addAvatar(body)

# ajout des parois, definition des interactions et
# ecriture des fichiers
```

Exemple de dépôt sur un réseau triangulaire

- échantillon polydisperse de disques dans une boîte



Echantillon de disques : calcul du nombre de particules

```
# imports, declarations (modeles, materiaux, conteneurs)

# parametres du script
#   * taille de la boite (75 cm x 50 cm)
lx=0.75; ly=0.5
#   * taille d'un element du reseau (5 cm)
l=5.e-2
#   * choix de l'orientation du reseau
orient='up'
#   * nombre de particules sur la premiere couche
nb_pt=15
#   * nombre de couches
nb_layer=20

# calcul du nombre de particules sur le reseau
nb_particles=nbPointsInTriangularLattice2D(nb_ele=nb_pt,
    nb_layer=nb_layer, orientation=orient)
```

Echantillon de disques : boucle de génération des particules

```
# tirage aleatoire des rayons entre 0.5 et 2 cm
radii=granulo_Random(nb_particles, 5.e-3, 2.e-2)
# depot des particules sur le reseau
coor=triangularLattice2D(nb_ele=nb_pt, l=l, x0=0.,
   y0=0., nb_layer=nb_layer, orientation=orient)

# boucle d'ajout des particules deposees :
for i in range(0, nb_particles):
   # creation d'un nouveau disque deformable
   body=deformableParticle2D(r=radii[i],
      center=coor[2*i : 2*(i + 1)], type_part='Disk',
      model=m2Dl, material=steel, color='BLEUx')
   # ajout de la particule a l'ensemble de corps
   bodies.addAvatar(body)

# ajout des parois, definition des interactions et
# ecriture des fichiers
```

Méthodes de dépôt sur réseau

- dépôt sur réseaux triangulaires ou carrés en 2D et cubiques en 3D,
- pavage d'un rectangle en 2D et d'un parallépipède en 3D,
- génération d'échantillons lâches,
- possibilité de remplir des conteneurs quelconques,
- l'anisotropie dépend uniquement du chargement,
- nombre de particules du réseau calculable à l'avance
  $\Rightarrow$ granulométrie toujours respectée,
- méthode supportant les cas monodisperses.

Génération d'échantillons granulaires

- tirage de rayons suivant une granulométrie donnée ▸ catalogue : *e.g.* aléatoire, uniforme, partition entre deux tailles suivant un critère de volume (*cf.* aide en ligne des fonctions granulo_*)
- particules prédéfinies ▸ catalogue : *e.g.* disques (rigides ou déformables), sphères (rigides), polygones réguliers (rigides), clusters de disque (rigides) (*cf.* doc du package Pre.build_avatar.particles),
- méthodes de dépôts ▸ catalogue :
  - dépôt dans des contenants 2D : *e.g.* boîtes, Couette, disque, tambour tournant (*cf.* doc du package Pre.build_avatar.tools.macros.containers2D),
  - dépôt dans des containers 3D : *e.g.* boîtes, sphère, cylindre (*cf.* doc du package Pre.build_avatar.tools.macros.containers3D),
  - dépôt sur réseau : *e.g.* carré ou triangulaire, en 2D ou cubique en 3D (*cf.* doc des packages Pre.build_avatar.lattices2D et Pre.build_avatar.lattices3D),
- nombreux exemples disponibles dans le répertoire : examples/Pre/prepro_grains

Granulometry

Need to compile Pre extension... (cmake option BUILD_PRE : True by default)
List of functions generating a granulometry according to different methods :

1. granulo_Monodisperse
2. granulo_Random
3. granulo_Uniform
4. granulo_TwoSizesNumber
5. granulo_TwoSizesVolume
6. granulo_ReadFromFile

Granulometry

granulo_Monodisperse

Generates a list of radii following a monodisperese distribution.

$$radii = granulo\_Monodisperse(nb\_particles, radius)$$

where :

- nb_particles (input integer), number of particles
- radius (input double), radius of the particles
- radii (returned double array), generated radii list

Granulometry

granulo_Random

Used in every examples of the prepro_grains directory.
Generates a random list of radii between bounds.

$$radii = granulo\_Random(nb\_particles, min\_radius, max\_radius)$$

where :

- nb_particles (input integer), number of particles
- min_radius (input double), minimum radius of the particles
- max_radius (input double), maximum radius of the particles
- radii (returned double array), generated radii list

Granulometry

granulo_Uniform

Generates a list of radii following a uniform distribution between bounds.

$$radii = granulo\_Uniform(nb\_particles, min\_radius, max\_radius)$$

where :

- nb_particles (input integer), number of particles
- min_radius (input double), minimum radius of the particles
- max_radius (input double), maximum radius of the particles
- radii (returned double array), generated radii list

Granulometry

granulo_TwoSizesNumber

Generates a list of radii of two sizes specifiying a ratio on numbers of particles.

$$radii = granulo\_TwoSizesNumber(nb\_particles, radius1, radius2, ratio)$$

where :

- nb_particles (input integer), number of particles
- radius1 (input double), first radius of the particles
- radius2 (input double), second radius of the particles
- ratio (input doube), ratio of particles of first radius
- radii (returned double array), generated radii list

In the end, in the radii array, there are about $ratio * nb\_particles$ particles of the first radius

Granulometry

granulo_TwoSizesVolume

Generates a list of radii of two sizes specifiying a ratio on volume of particles.

$$radii = granulo\_TwoSizesVolume(nb\_particles, radius1, radius2, ratio)$$

where :

- nb_particles (input integer), number of particles
- radius1 (input double), first radius of the particles
- radius2 (input double), second radius of the particles
- ratio (input doube), ratio of particles of first radius
- radii (returned double array), generated radii list

In the end, in the radii array, there is a number of the particles of the first radius so that their volume is *ratio*∗the total volume of the particles.

Granulometry
granulo_ReadFromFile

Generates a list of radii from a file.

$$radii = granulo\_ReadFromFile(nb\_particles, file\_name)$$

where :

- nb_particles (input integer), number of particles
- file_name (input string), name of the file to read
- radii (returned double array), generated radii list

Particles generation

1. rigidDisk
2. rigidSpher
3. rigidCluster
4. rigidPolygon
5. rigidDiscreteDisk
6. deformableParticles2D

Particles generation

rigidDisk

Create an avatar of a rigid disk.

$$body = rigidDisk(r, center, model, material, \\ theta = 0., color = 'BLEUx', number = None)$$

where :

- r (input double), radius of the desired particle
- center (input double array), coordinates of the center of the desired particle
- model (input model object), rigid model for the particle
- material (input material object), material of the particle
- theta (optional input double), rotation angle in the inertia frame
- color (optional input string), color of the disk contactor (5 characters string)
- number (optional input integer) index of the avatar
- body (returned avatar object), the avatar

Particles generation

rigidSphere

Create an avatar of a rigid sphere.

$$body = rigidSphere(r, center, model, material,\\ color = \text{'}BLEUx\text{'}, number = None)$$

where :

- r (input double), radius of the desired particle
- center (input double array), coordinates of the center of the desired particle
- model (input model object), rigid model for the particle
- material (input material object), material of the particle
- color (optional input string), color of the disk contactor (5 characters string)
- number (optional input integer) index of the avatar
- body (returned avatar object), the avatar

Particles generation

rigidCluster

Create an avatar of a cluster of rigid disks.

$$body = rigidCluster(r, center, nb\_disk, model, material,$$
$$theta = 0., color = 'BLEUx', number = None)$$

where :

- r (input double), radius of the bounding disk
- center (input double array), coordinates of the center of the desired particle
- nb_disk (input integer), number of disks of the cluster
- model (input model object), rigid model for the particle
- material (input material object), material of the particle
- theta (optional input double), rotation angle in the inertia frame
- color (optional input string), color of the disk contactor (5 characters string)
- number (optional input integer) index of the avatar
- body (returned avatar object), the avatar

Particles generation

rigidPolygon

Create an avatar of a rigid cluster of disks.

$$body = rigidPolygon(r, center, nb\_vertex, model, material,$$
$$theta = 0., color = 'BLEUx', number = None)$$

where :

- r (input double), bounding radius of the particle
- center (input double array), coordinates of the center of the desired particle
- nb_vertex (input integer), number of vertex of the polygon
- model (input model object), rigid model for the particle
- material (input material object), material of the particle
- theta (optional input double), rotation angle in the inertia frame
- color (optional input string), color of the disk contactor (5 characters string)
- number (optional input integer) index of the avatar
- body (returned avatar object), the avatar

Particles generation

deformableParticles2D

Create an avatar of a deformable disk or pentagon.

$$body = deformableParticle2D(r, center, type\_part, model, material,$$
$$theta = 0., color = 'BLEUx', number = None)$$

where :

- r (input double), bounding radius of the particle
- center (input double array), coordinates of the center of inertia of the desired particle
- type_part (input 'Disk' or 'pent'), the type of particle to generate
- model (input model object), model of the particle
- material (input material object), material of the particle
- theta (optional input double), rotation angle in the inertia frame
- color (optional input string), color of the contactor (5 characters string) on the group 'skin'
- number (optional input integer) index of the avatar
- body (returned avatar object), the avatar

Deposit

Need to compile Pre extension... (cmake option BUILD_PRE : True by default)
List of functions deposing a list of radius in a container or on a lattice

1. Deposit in container 2D
   1.1 depositInBox2D
   1.2 depositInDisk2D
   1.3 depositInCouette2D
   1.4 depositInDrum2D
2. Deposit in container 3D
   2.1 depositInBox3D
   2.2 depositInCylinder3D
   2.3 depositInSphere3D
3. Deposit on lattices
   3.1 squareLattice2D
   3.2 triangularLattice2D
   3.3 cubicLattice3D

Warnings :

- the input granulometry may be changed on output if the number of remaining particles is less than the input number of particles
- to avoid interprenetration between particles and container, these function use a shrink based on the size of the particles

Deposit in container 2D

depositInBox2D

Deposit under gravity circular particles in a box center on 0 along x-axis.

$$[nb\_remaining\_particles, coor] = depositInBox2D(radii, lx, ly, deposited\_radii = None, deposited\_coor = None)$$

where :

- radii (input double array), list of radii
- lx (input double), length of the box in which to deposit
- ly (input double), height of the box in which to deposit
- deposited_radii (optional input double array), radii of particles supposed to be in the container before the deposit
- deposited_coor (optional input coordinates array), coordinates of deposited_radii particles
- nb_remaining_particles (returned integer), number of deposited particles in the container
- coor (returned coordinates array), coordinates of deposited particles

Deposit in container 2D

depositInDisk2D

Deposit under gravity of circular particles in a disk centered on 0.

$$[nb\_remaining\_particles, coor] = depositInDisk2D(radii, r, deposited\_radii = None, deposited\_coor = None)$$

where :

- radii (input double array), list of radii
- r (input double), radius of the container
- deposited_radii (optional input double array), radii of particles supposed to be in the container before the deposit
- deposited_coor (optional input coordinates array), coordinates of deposited_radii particles
- nb_remaining_particles (returned integer), number of deposited particles in the container
- coor (returned coordinates array), coordinates of deposited particles

Deposit in container 2D

depositInCouette2D

Deposit under gravity of circular particles in a container designed for a Couette shear.

$$[nb\_remaining\_particles, coor] = depositInCouette2D(radii, rint, rext, \\ deposited\_radii = None, \\ deposited\_coor = None)$$

where :

- radii (input double array), list of radii
- rint (input double), internal radius of the ring occupied by particles
- rext (input double), external radius of the ring occupied by particles
- deposited_radii (optional input double array), radii of particles supposed to be in the container before the deposit
- deposited_coor (optional input coordinates array), coordinates of deposited_radii particles
- nb_remaining_particles (returned integer), number of deposited particles in the container
- coor (returned coordinates array), coordinates of deposited particles

Deposit in container 2D

depositInDrum2D

Deposit under gravity of circular particles in a half drum.

$$[nb\_remaining\_particles, coor] = depositInDrum2D(radii, r, \\ deposited\_radii = None, \\ deposited\_coor = None)$$

where :

- radii (input double array), list of radii
- r (input double), radius of the drum
- deposited_radii (optional input double array), radii of particles supposed to be in the container before the deposit
- deposited_coor (optional input coordinates array), coordinates of deposited_radii particles
- nb_remaining_particles (returned integer), number of deposited particles in the container
- coor (returned coordinates array), coordinates of deposited particles

Deposit in container 3D

depositInBox3D

Deposit under gravity spherical particles in a box centered on 0 along x and y axis.

$$
\begin{aligned}
[nb\_remaining\_particles, coor] = depositInBox3D(radii, lx, ly, lz,& \\
deposited\_radii &= None, \\
deposited\_coor &= None)
\end{aligned}
$$

where :

- radii (input double array), list of radii
- lx (input double), width of the box along x-axis in which to deposit
- ly (input double), width of the box along y-axis in which to deposit
- lz (input double), height of the box in which to deposit
- deposited_radii (optional input double array), radii of particles supposed to be in the container before the deposit
- deposited_coor (optional input coordinates array), coordinates of deposited_radii particles
- nb_remaining_particles (returned integer), number of deposited particles in the container
- coor (returned coordinates array), coordinates of deposited particles

Deposit in container 3D

depositInCylinder3D

Deposit under gravity of spherical particles in a cylinder with bottom at 0.

$$[nb\_remaining\_particles, coor] = depositInCylinder3D(radii, R, lz, deposited\_radii = None, deposited\_coor = None)$$

where :

- radii (input double array), list of radii
- R (input double), radius of the cylinder
- lz (input double), heigth of the cylinder
- deposited_radii (optional input double array), radii of particles supposed to be in the container before the deposit
- deposited_coor (optional input coordinates array), coordinates of deposited_radii particles
- nb_remaining_particles (returned integer), number of deposited particles in the container
- coor (returned coordinates array), coordinates of deposited particles

Deposit in container 3D

depositInSphere3D

Deposit under gravity of spherical particles in a sphere.

$$[nb\_remaining\_particles, coor] = depositInSphere3D(radii, R, center, \\ deposited\_radii = None, \\ deposited\_coor = None)$$

where :

- radii (input double array), list of radii
- R (input double), radius of the sphere
- center (input double array), center of the sphere
- deposited_radii (optional input double array), radii of particles supposed to be in the container before the deposit
- deposited_coor (optional input coordinates array), coordinates of deposited_radii particles
- nb_remaining_particles (returned integer), number of deposited particles in the container
- coor (returned coordinates array), coordinates of deposited particles

Deposit on lattice

squareLattice2D

Generate a list of coordinates on a square lattice

$$coor = squareLattice2D(nb\_ele, nb\_layer, l, x = 0.0, y = 0.0)$$

where :

- nb_ele (input integer), number of particles on the first layer
- nb_layer (input integer), number of layers
- l (input double), length of the lattice element
- x (optional input double), x coordinate of the lower left corner of the bounding box
- y (optional input double), y coordinate of the lower left corner of the bounding box
- coor (returned double array), coordinates of lattice $[x_1, y_1, x_2, y_2, ...x_n, y_n]$

Deposit on lattice

triangularLattice2D

Generate a list of coordinates on a triangular lattice

$$coor = triangularLattice2D(nb\_ele, nb\_layer, l, x = 0.0, y = 0.0, orientation =' up')$$

where :

- nb_ele (input integer), number of particles on the first layer
- nb_layer (input integer), number of layers
- l (input double), length of the lattice element
- x (optional input double), x coordinate of the lower left corner of the bounding box
- y (optional input double), y coordinate of the lower left corner of the bounding box
- orientation (optional input 'up' or 'down'), orientation of the first lower triangular
- coor (returned double array), coordinates of lattice $[x_1, y_1, x_2, y_2, ...x_n, y_n]$

Deposit on lattice

cubicLattice3D

Generate a list of coordinates on a cubic lattice

$coor = cubicLattice3D(nb\_ele\_x, nb\_ele\_y, nb\_layer, l, x = 0.0, y = 0.0, z = 0.0)$

where :

- nb_ele_x (input integer), number of particles on the first layer along x-axis
- nb_ele_y (input integer), number of particles on the first layer along y-axis
- nb_layer (input integer), number of layers
- l (input double), length of the lattice element
- x (optional input double), x coordinate of the lower left corner of the bounding box
- y (optional input double), y coordinate of the lower left corner of the bounding box
- z (optional input double), z coordinate of the lower left corner of the bounding box
- orientation (optional input 'up' or 'down'), orientation of the first lower triangular
- coor (returned double array), coordinates of lattice $[x_1, y_1, z_1 \ldots x_n, y_n, z_n]$