Generator Module V1.4
- User Guide -

C. Benoit, G. Jeanfaivre, S. Peron et P. Raud
Onera / DSNA

September 23, 2009

1 Generator : Mesh generation module

1.1 Preamble

This module is used to generate meshes from geometries. A mesh is stored in a Converter array or in a zone node of a pyTree.

When using the array interface, import the Generator module:

```
import Generator as G
```

Then, a is an array and A is a list of arrays.

When using the pyTree interface, import the module:

```
import Generator.PyTree as G
```

Then, a is a zone node and A is a list of zone nodes or a complete pyTree.

1.2 Analytical grids creation

Create a structured Cartesian mesh with ni x nj x nk points starting from point (xo,yo,zo) and of step (hi,hj,hk):

```
a = G.cart((xo,yo,zo), (hi,hj,hk), (ni,nj,nk))
```

(See : Examples/generator/cart.py)

Create a unstructured hexaedrical mesh defined from a Cartesian grid of ni x nj x nk points starting from point (xo,yo,zo) and of step (hi,hj,hk). Type of elements are 'QUAD' for 2D arrays and 'HEXA' for 3D arrays:

```
a = G.cartHexa((xo,yo,zo), (hi,hj,hk), (ni,nj,nk))
```

(See : Examples/generator/cartHexa.py) (See : Examples/generator/cartHexaPT.py)

Create a unstructrued tetraedrical mesh defined from a Cartesian grid of ni x nj x nk points starting from point (xo,yo,zo) and of step (hi,hj,hk). Type of elements are 'TRI' for 2D arrays and 'TETRA' for 3D arrays:

```
a = G.cartTetra((xo,yo,zo), (hi,hj,hk), (ni,nj,nk))
```
(See : Examples/generator/cartTetra.py) (See : Examples/generator/cartTetraPT.py)

Create a pentahedrical prismatic mesh defined by an unstructured array starting from a regular Cartesian mesh. The initial Cartesian mesh is defined by ni x nj x nk points starting from point (xo,yo,zo) and of step (hi,hj,hk). Type of elements is 'PENTA':

```
a = G.cartPenta((xo,yo,zo), (hi,hj,hk), (ni,nj,nk))
```
(See : Examples/generator/cartPenta.py) (See : Examples/generator/cartPentaPT.py)

Create a regular cylindrical grid (or a portion of cylinder between tetas and tetae) with ni x nj x nk points, of center-bottom point (xo,yo,zo), of inner radius R1, outer radius R2 and height H. Return the mesh array:

```
a = G.cylinder((xo,yo,zo), R1, R2, tetas, tetae, H, (ni,nj,nk))
```
(See : Examples/generator/cylinder.py) (See : Examples/generator/cylinderPT.py)

Create an irregular cylindrical grid (or a portion of cylinder between tetas and tetae) with ni x nj x nk points, of center-bottom point (xo,yo,zo), of inner radius R1, outer radius R2, height H and with distributions in r, teta, z. Distributions are arrays defining 1D meshes (x and i varying) giving a distribution in [0,1]. Their number of points gives ni, nj, nk:

```
a = G.cylinder2((xo,yo,zo), R1, R2, tetas, tetae, H, arrayR, arrayTeta, arrayZ)
```
(See : Examples/generator/cylinder2.py) (See : Examples/generator/cylinder2PT.py)

Create an irregular cylindrical grid (or a portion of cylinder between tetas and tetae) from a xz plane mesh defined by a and a teta distribution defined by arrayTeta:

```
b = G.cylinder3(a, tetas, tetae, arrayTeta)
```
(See : Examples/generator/cylinder3.py) (See : Examples/generator/cylinder3PT.py)

Create a 2D Delaunay type mesh from an array. The array can be a 2D structured array, or an unstructured array of type 'NODE', 'TRI' or 'QUAD'. Epsilon is a geometric tolerance. Points nearer than epsilon are merged. If keepBB is set to 1, the bounding box is kept in the final triangulation:

```
b = G.delaunay(a, epsilon, keepBB)
```
(See : Examples/generator/delaunay.py) (See : Examples/generator/delaunayPT.py)

Create a constrained Delaunay triangulation of the convex hull of a contour c. Contour must be a BAR-array and must be in the plane (x,y). Epsilon is a geometric tolerance. Points nearer than epsilon are merged. If keepBB is set to 1, the bounding box is kept in the final triangulation:

```
b = G.constrainedDelaunay(c, epsilon, keepBB)
```
(See : Examples/generator/constrainedDelaunay.py) (See : Examples/generator/constrainedDelaunayPT.py)

Create a structured mesh from a curve defined by a i-array and a point:

```
b = G.pointedHat(a, (x,y,z))
```
(See : Examples/generator/pointedHat.py) (See : Examples/generator/pointedHatPT.py)

Create a stitched mesh from a curve defined by a i-array. The surface is stitched in the middle. Eps is the accuracy of the serach, eps2 is a merging tolerance and offx, offy, off z an optional offset:

```
b = G.stitchedHat(a, (offx,offy,offz), eps, eps2)
```
(See : Examples/generator/stitchedHat.py)

Simple Cartesian generator. Create a set of Cartesian grids (B) around a list of body grids (A). Those grids are patched with a ratio of 2. The user controls the number of levels, and the number of points for each level of grid. h is the spatial step on the finest level. Dfar is the maximal distance

to the body. nlvl is a list that provides the number of points per level, except for the finest level:

```
B = G.gencartmb(A, h, Dfar, nlvl)
```

(See : Examples/generator/gencartmb.py) (See : Examples/generator/gencartmbPT.py)

Create a set of overset Cartesian grids given a list of body grids (A). This requires Cassiopee Kernel:

```
import Generator.Cartesian
B = Generator.Cartesian.gencart(A, [options])
```

options argument can be:

- a Cassiopee numerics object:

(See : Examples/generator/gencart2.py) - a list of options:

```
B = Generator.Cartesian.gencart(A, ["vmin",8,"snear_type",3,"snear_mul",1.,...])
```

(See : Examples/generator/gencart.py) (See : Examples/generator/gencartPT.py)

See Cassiopee Kernel : Cartesian Solver documentation.

1.3 Information on generated meshes

Check regularity, orthogonality for a mesh defined by an array:

```
G.check(array)
```

(See : Examples/generator/check.py) (See : Examples/generator/checkPT.py)

Return the bounding box of A:

```
bb = G.bbox(A)
```

(See : Examples/generator/bbox.py) (See : Examples/generator/bboxPT.py)

Return the bounding box of all cells of a. The bounding box is located at centers of cells:

```
bb = G.bboxOfCells(a) .or. BB = G.bboxOfCells(A)
```

(See : Examples/generator/bboxOfCells.py) (See : Examples/generator/bboxOfCellsPT.py)

Test the Cartesian Elements Bounding Box (CEBB) intersection between a1 and a2. Return 0 if no intersection, 1 otherwise: Tolerance is a float given by tol.

```
intersect = G.CEBBIntersection(a1, a2, tol)
```

(See : Examples/generator/CEBBIntersection.py) (See : Examples/generator/CEBBIntersectionPT.py)

Test if bounding boxes of a1 and a2 intersects, within a tolerance tol (default value is 1.e-6). Return 0 if no intersection, 1 otherwise:

```
intersect = G.bboxIntersection(a1, a2, tol)
```

(See : Examples/generator/bboxIntersection.py) (See : Examples/generator/bboxIntersectionPT.py)

Test if a given point is in the CEBB of a:

```
inside = G.checkPointInCEBB(a, (x,y,z))
```

(See : Examples/generator/checkPointInCEBB.py) (See : Examples/generator/checkPointInCEBBPT.py)

Return the volume field of an array. Volume is located at centers of cells:

```
b = G.getVolumeMap(a) .or. B = G.getVolumeMap(A)
```

(See : Examples/generator/getVolumeMap.py) (See : Examples/generator/getVolumeMapPT.py)

Return the surface normals field of a surface array. It is located at centers of cells:

```
b = G.getNormalMap(a) .or. B = G.getNormalMap(A)
```

(See : Examples/generator/getNormalMap.py) (See : Examples/generator/getNormalMapPT.py)

Return a measure of cell planarity for each cell. It is located at centers of cells:

```
b = G.getCellPlanarity(a) .or. B = G.getCellPlanarity(A)
```
(See : Examples/generator/getCellPlanarity.py) (See : Examples/generator/getCellPlanarityPT.py)

1.4 Operations on distributions

Distributions are Cartesian meshes that can be used to be mapped on profiles to make curvilinear meshes for instance. Distributions in 2D (x,y) distribution represent the length and height of each cell. Distributions in 3D (x,y,z) represents the lengths in the three topological directions of each cell. Distributions can be modified by the enforce functions.

Enforce a region around a line x = x0. The size of the cell around the line is enforcedh. "supp" points are suppressed from the starting distribution on the left and right side. "add" points are added on the left and add points are added on the right. Add exactely add points:

```
b = G.enforceX(a, x0, enforcedh, (supp,add))
```
(See : Examples/generator/enforceX.py) (See : Examples/generator/enforceXPT.py)

Adjust add in order to have a monotonic distribution:

```
b = G.enforceX(a, x0, enforcedh, supp, add)
```
(See : Examples/generator/enforce.py) (See : Examples/generator/enforcePT.py)

Same as before but with a one sided distribution (left):

```
b = G.enforceMoinsX(a, enforcedh, (supp,add)) .or. b = G.enforceMoinsX(a, enforcedh, supp, add)
```
(See : Examples/generator/enforceMoinsX.py) (See : Examples/generator/enforceMoinsXPT.py) This can be usefull to create a boundary layer distribution in an Euler mesh.

Same as before but with a one sided distribution (right):

```
b = G.enforcePlusX(a, enforcedh, (supp,add)) .or. b = G.enforcePlusX(a, enforcedh, supp, add)
```
(See : Examples/generator/enforcePlusX.py) (See : Examples/generator/enforcePlusXPT.py)

Those functions are also available in Y and Z directions, except enforce functions in Z-directions with Py trees.

For instance, enforce a region around a the j line which is at y = y0:

```
b = G.enforceY(a, y0, enforcedh, (supp,add)) .or. b = G.enforceY(a, y0, enforcedh, supp, add)
```
(See : Examples/generator/enforceY.py) (See : Examples/generator/enforceYPT.py) (See : Examples/generator/enforceMoinsZ.py)

Enforce a curvilinear line defined by the array line in a distribution defined by the array a:

```
b = G.enforceLine(a, line, enforcedh, (supp,add))
```
(See : Examples/generator/enforceLine.py) (See : Examples/generator/enforceLinePT.py)

Enforce a point in the distribution. The index of enforced point is returned:

```
ind = G.enforcePoint(a, x0)
```
(See : Examples/generator/enforcePoint.py) (See : Examples/generator/enforcePointPT.py)

Enforce the curvature of an i-curve in a distribution defined by a. Power reflecting the power of stretching (typically 0.5):

```
b = G.enforceCurvature(a, curve, power)
```

(See : Examples/generator/enforceCurvature.py) (See : Examples/generator/enforceCurvaturePT.py)

Add a point in a distribution at index ind:

```
b = G.addPointInDistribution(a, ind)
```

(See : Examples/generator/addPointInDistribution.py) (See : Examples/generator/addPointInDistributionPT.

1.5 Operations on meshes

Close a mesh defined by array a. Points that are distant of eps maximum to one another are merged:

```
b = G.close(a, eps) .or. B = G.close(A, eps)
```

(See : Examples/generator/close.py) (See : Examples/generator/closePT.py)

Select elements of a TRI-array, whose centers are inside the given list of curves, defined by BAR-arrays:

```
b = G.selectInsideElts(a, curves)
```

(See : Examples/generator/selectInsideElts.py)

Map a distribution on a curve or on a structured surface:

```
b = G.map(a, distrib)
```

Map a i-array distribution in a direction (dir=1,2,3) in a surface or volume mesh:

```
b = G.map(a, distrib, dir)
```

(See : Examples/generator/map.py) (See : Examples/generator/mapPT.py)

Split a i-array and map a distribution on the splitted i-array. Sensibility is the variation of curvature triggering split (1.e-2 for instance) :

```
b = G.mapSplit(a, distrib, 1.e-2)
```

(See : Examples/generator/mapSplit.py) (See : Examples/generator/mapSplitPT.py)

Densify a i-array or a BAR-array with a new discretization step h. Discretization points from the original array are keeped :

```
b = G.densify(a, h)
```

(See : Examples/generator/densify.py) (See : Examples/generator/densifyPT.py)

Grow a surface array of one layer. Vector is the node displacement:

```
b = G.grow(a, vector)
```

(See : Examples/generator/grow.py) (See : Examples/generator/growPT.py)

Add normal layers to a surface mesh. d is a one-D distribution specifying the heigh of each layer:

```
b = G.addNormalLayers(a, d, check)
```

If check = 1, layers grow only of created cell volumes are positive.
(See : Examples/generator/addNormalLayers.py)

Generate a mesh by transfinite interpolation (TFI). Generated mesh can be 2D or 3D structured, or unstructured TRI or PENTA mesh. Warning : the boundaries can be in a different order from the examples below, except for the PENTA TFI meshes.
2D structured mesh is built from imin, imax, jmin, jmax boundaries.
3D structured mesh is built from imin, imax, jmin, jmax, kmin, kmax boundaries.
Dimensions must be equal for each pair (imin,imax), (jmin,jmax)...
TRI mesh is built from imin, jmin, diag boundaries. Each boundary is a structured array with the same dimension. PENTA mesh is built from Tmin, Tmax triangles boundary and imin, imax, diag

boundaries. Tmin, Tmax must be structured triangles of dimension nxn. imin, jmin, diag must be structured n*p arrays:

2D-struct : a = G.TFI([imin, imax, jmin, jmax])
3D-struct : a = G.TFI([imin, imax, jmin, jmax, kmin, kmax])
TRI : a = G.TFI([imin, jmin, diag])
PENTA : a = G.TFI([Tmin, Tmax, imin, imax, diag])

(See : Examples/generator/TFI.py) (See : Examples/generator/TFIPT.py)

Generate a mesh by elliptic grid generator:

a = G.TTM(imin, imax, jmin, jmax, nit)

(See : Examples/generator/TTM.py) (See : Examples/generator/TTMPT.py)

Generate an hyperbolic mesh (2D) of ”C” or ”O” type from a from a line defined by line a and from a distribution defined by distrib. The resulting mesh is nearly orthogonal:

b = G.hyper2D(line, distrib, ”C”)

(See : Examples/generator/hyper2d.py) (See : Examples/generator/hyper2dPT.py)

Generate a 2D mesh around a 2D polyline:

B = G.PolyLine.polyLineMesher(a, h, hf, density, ext)

where a is the input polyline (BAR-array), h is the height of the mesh, hf is the height of the first cell, density is the number of points per unity of length and ext the number of points for the regions where the mesh is extended.

In the ’array’ version, it returns a list where B[0] is the list of generated meshes, B[1] is the list of wall boundaries, B[2] is the list of overlap boundaries, B[3] is h, B[4] is density (eventually modified by the mesher).

In the pyTree version, it returns a list [t,hs,densities], where t is a CGNS python tree, containing the blocks, wall boundaries and overlap boundaries; hs is the list of heights (modified if necessary), and densities the list of densities (also modified if necessary).

(See : Examples/generator/polyLineMesher.py) (See : Examples/generator/polyLineMesherPT.py)

Generate a 2D mesh around a 2D polyC1 curve:

B = G.PolyC1.polyC1Mesher(A, h, hf, density, ext)

where A is a list of i-arrays each representing a C1 curve. All i-arrays put together must represent a polyC1 curve. Other arguments are similar to polyLineMesher. The function return is also similar to polyLineMesher.

(See : Examples/generator/polyC1Mesher.py) (See : Examples/generator/polyC1MesherPT.py)

1.6 More general examples of use

- See : Examples/generator/naca.py
- See : Examples/generator/gencart2.py

1.7 Example files

Example file : Examples/generator/cart.py

```
# - cart (array) -
import Converter as C
```

```
import Generator as G

a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,10))
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/cartHexa.py

```
# - cartHexa (array) -
import Generator as G
import Converter as C

a = G.cartHexa((0.,0.,0.), (0.1,0.1,0.2), (10,10,10))
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/cartHexaPT.py

```
# - cartHexa (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C

t = C.newPyTree(['Base1',2,'Base2',3])
a = G.cartHexa((0.,0.,0.), (0.1,0.1,0.2), (10,10,1)); t[1][2].append(a)
a = G.cartHexa((0.,0.,0.), (0.1,0.1,0.2), (10,10,10)); t[2][2].append(a)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/cartTetra.py

```
# - cartTetra (array) -
import Generator as G
import Converter as C

a = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/cartTetraPT.py

```
# - cartTetra (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C

t = C.newPyTree(['Base1',2,'Base2',3])
a = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
t[1][2].append(a)
a = G.cartTetra((0.,0.,0.), (1.,1.,1.), (2,2,2))
t[2][2].append(a)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/cartPenta.py

```
# - cartPenta (array) -
import Generator as G
import Converter as C

a = G.cartPenta((0.,0.,0.), (0.1,0.1,0.2), (10,10,10))
C.convertArrays2File([a],"out.plt","bin_tp")
```

Example file : Examples/generator/cartPentaPT.py

```
# - cartPenta (pyTree)-
import Generator.PyTree as G
import Converter.PyTree as C

a = G.cartPenta((0.,0.,0.), (0.1,0.1,0.2), (10,10,10))
t = C.newPyTree(['Base',3]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/cylinder.py

```
# - cylinder (array) -
import Generator as G
import Converter as C

a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,50,30))
C.convertArrays2File([a], "out.plt", "bin_tp")
```

Example file : Examples/generator/cylinderPT.py

```
# - cylinder (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,50,30))
t = C.newPyTree(['Base',3]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/cylinder2.py

```
# - cylinder2 (array) -
import Converter as C
import Generator as G

r = G.cart((0.,0.,0.), (0.1, 1., 1.), (11, 1, 1))
teta = G.cart((0.,0.,0.), (0.1, 1., 1.), (11, 1, 1))
z = G.cart((0.,0.,0.), (0.1, 1., 1.), (11, 1, 1))

cyl = G.cylinder2( (0.,0.,0.), 0.5, 1., 360., 0., 10., r, teta, z)
C.convertArrays2File([cyl], "out.plt", "bin_tp")
```

Example file : Examples/generator/cylinder2PT.py

```
# - cylinder2 (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

r = G.cart((0.,0.,0.), (0.1, 1., 1.), (11, 1, 1))
teta = G.cart((0.,0.,0.), (0.1, 1., 1.), (11, 1, 1))
z = G.cart((0.,0.,0.), (0.1, 1., 1.), (11, 1, 1))

cyl = G.cylinder2( (0.,0.,0.), 0.5, 1., 360., 0., 10., r, teta, z)
t = C.newPyTree(['Base',3]); t[1][2].append(cyl)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/cylinder3.py

```
# - cylinder3 (array) -
import Generator as G
import Converter as C

teta = G.cart((0.,0.,0.), (0.1, 1., 1.), (11, 1, 1))
xz = G.cart((0.1,0.,0.), (0.1,1.,0.2), (20, 1, 30))
cyl = G.cylinder3( xz, 0., 90., teta)
C.convertArrays2File([cyl], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/cylinder3PT.py

```
# - cylinder3 (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

teta = G.cart((0.,0.,0.), (0.1, 1., 1.), (11, 1, 1))
xz = G.cart((0.1,0.,0.), (0.1,1.,0.2), (20, 1, 30))
cyl = G.cylinder3( xz, 0., 90., teta)
t = C.newPyTree(['Base',3]); t[1][2].append(cyl)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/delaunay.py

```
# - delaunay (array) -
import Generator as G
import Converter as C

ni = 11; nj = 11; nk = 1
hi = 1./(ni-1); hj = 1./(nj-1); hk = 1.
a = G.cart((0.,0.,0.), (hi,hj,hk), (ni,nj,nk))
b = G.delaunay(a)
C.convertArrays2File([a,b], "out.plt", "bin_tp")
```

Example file : Examples/generator/delaunayPT.py

```
# - delaunay (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

ni = 11; nj = 11; nk = 1
hi = 1./(ni-1); hj = 1./(nj-1); hk = 1.
a = G.cart((0.,0.,0.), (hi,hj,hk), (ni,nj,nk))
b = G.delaunay(a); b[0] = 'delaunay'
t = C.newPyTree(['Base',2]); t[1][2].append(a); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/constrainedDelaunay.py

```
# - constrainedDelaunay (array) -
import Converter as C
import Generator as G
import Transform as T
import Geom as D

A = D.text1D('STEPHANIE')
A = C.convertArray2Tetra(A)
a = T.join(A)

# Triangulation avec respect du contour
tri = G.constrainedDelaunay(a)
C.convertArrays2File([a,tri], "out.plt")
```

Example file : Examples/generator/constrainedDelaunayPT.py

```
# - constrainedDelaunay (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Geom.PyTree as D
import Transform.PyTree as T

A = D.text1D('STEPHANIE')
A = C.convertArray2Tetra(A)
a = T.join(A)
# Triangulation avec respect du contour
tri = G.constrainedDelaunay(a); tri[0] = 'cdelaunay'
t = C.newPyTree(['Base',2,'Base2',1]); t[1][2].append(tri); t[2][2].append(a)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/pointedHat.py

```
# - pointedHat (array) -
import Geom as D
import Generator as G
import Converter as C

c = D.circle( (0,0,0), 1., 360., 0., 100)
surf = G.pointedHat(c,(0.,0.,1.))
C.convertArrays2File([surf], 'out.plt')
```

Example file : Examples/generator/pointedHatPT.py

```
# - pointedHat (pyTree) -
import Geom.PyTree as D
import Generator.PyTree as G
import Converter.PyTree as C

c = D.circle( (0,0,0), 1., 360., 0., 100)
surf = G.pointedHat(c,(0.,0.,1.))
t = C.newPyTree(['Base',3]); t[1][2].append(surf)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/stitchedHat.py

```
# - stitchedHat (array) -
import Geom as D
import Generator as G
import Transform as T
import Converter as C

c = D.circle( (0,0,0), 1., 360., 0., 100)
c = T.contract(c, (0,0,0), (0,1,0), (0,0,1), 0.1)
c = T.reorder(c, (1,2,3))

c = G.stitchedHat(c, (0,0,0), 1.e-3)

C.convertArrays2File([c], 'out.plt')
```

Example file : Examples/generator/gencartmb.py

```
# - gencartmb (array) -
import Generator as G
import Converter as C

# maillage de corps
a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,50,30))
bodies = [a]

# Pas de la grille cartesienne la plus fine
h = 1.e-1
# Distance d'eloignement au corps
Dfar = 20.

# Nb de points par niveau :
# ici 4 niveaux, mais le dernier est calcule automatiquement
nlvl = [10,10,5] # nlvl[0] : grille grossiere

# Retourne la liste des grilles cartesiennes
cartGrids = G.gencartmb(bodies, h, Dfar, nlvl)
C.convertArrays2File(cartGrids, 'out.plt', 'bin_tp')
```

Example file : Examples/generator/gencartmbPT.py

```
# - gencartmb (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C
import Transform.PyTree as T

# Maillage de corps
a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,50,30))

# Pas de la grille cartesienne la plus fine
h = 1.e-1
# Distance d'eloignement au corps
Dfar = 20.
```

```
# Nb de points par niveau :
# ici 4 niveaux, mais le dernier est calcule automatiquement
nlvl = [10,10,5] # nlvl[0] : grille grossiere

# Retourne la liste des zones correspondant aux grilles cartesiennes
t = C.newPyTree(['Bodies', 'Cart']); t[1][2].append(a)
zones = G.gencartmb(t[1], h, Dfar, nlvl)
t[2][2] = t[2][2] + zones
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/gencart2.py

```
# Test des fonctions :
# - gencart

from elsA_user import *
import Converter as C
import Generator as G
import Generator.Cartesian as GC

blkPaleArrays = []
a = C.convertFile2Arrays('pale_bo105_1.tp',"fmt_tp")
a1 = a[0]
a = C.convertFile2Arrays('pale_bo105_2.tp',"fmt_tp")
a2 = a[0]
del a

for i in range(4) :
    blkPaleArrays.append(a1)
    blkPaleArrays.append(a2)

# NUMERICS
Num1 = numerics(name='Num1')
Num1.set('automesh', 'active')
Num1.set('automesh_type', 'cartesian')     # 0 : cartesian, 1 : cylinder
Num1.set('dfxm', 20.)  # Distance for far boundary x-
Num1.set('dfxp', 30.)  # Distance for far boundary x+
Num1.set('dfym', 20.)  # Distance for far boundary y-
Num1.set('dfyp', 20.)  # Distance for far boundary y+
Num1.set('dfzm', 30.)  # Distance for far boundary z-
Num1.set('dfzp', 20.)  # Distance for far boundary z+
Num1.set('vmin', 8)                # minimum size of grids
Num1.set('snear_type', 1)
Num1.set('snear_mul', 1.)
Num1.set('step_factor', 2.)
Num1.set('start_from_level', 2)      # start proximity from level i

arrays = GC.gencart(blkPaleArrays, [Num1])
C.convertArrays2File(arrays, "new.plt", "bin_tp")
```

Example file : Examples/generator/gencart.py

```
# - gencart (array) -
import Converter as C
import Generator.Cartesian as GC
import Generator as G

a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,50,30))
for i in range(100):
    A = GC.gencart([a], ["dfxm", 20., "dfxp", 20.,
                         "dfym", 20., "dfyp", 20.,
                         "vmin", 8, "snear_type", 3,
                         "snear_mul", 1., "step_factor", 2.,
                         "start_from_level", 1])

C.convertArrays2File(A, "out.plt", "bin_tp")
```

Example file : Examples/generator/gencartPT.py

```
# - gencart (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

# Maillage de corps
a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,50,30))
t = C.newPyTree(['Bodies']); t[1][2].append(a)

# Ajoute les grilles cartesiennes dans l'arbre t
options = ["dfar", 20.,"vmin", 8, "snear_type", 3,"snear_mul", 1.,\
           "step_factor", 2., "start_from_level", 1]
t = G.gencart(t, options)

C.convertPyTree2File(t, "out.cgns", "bin_cgns")
```

Example file : Examples/generator/check.py

```
# - check (array) -
import Generator as G

a = G.cart((0,0,0), (1,1,1), (10,10,10))
G.check(a)
```

Example file : Examples/generator/checkPT.py

```
# - check (pyTree) -
import Generator.PyTree as G

a = G.cart((0,0,0), (1,1,1), (10,10,10))
G.check(a)
```

Example file : Examples/generator/bbox.py

```
# - bbox (array) -
import Generator as G
import Converter as C

a = G.cart((0.,0.,0.),(0.1,0.1,1.),(20,20,20))
print G.bbox([a])
```

Example file : Examples/generator/bboxPT.py

```
# - bbox (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C

a = G.cart((0.,0.,0.),(0.1,0.1,1.),(20,20,20))
t = C.newPyTree(['Base']); t[1][2] = t[1][2] + [a]
print G.bbox(t)
```

Example file : Examples/generator/bboxOfCells.py

```
# - bboxOfCells (array) -
import Generator as G
import Converter as C

a = G.cart((0.,0.,0.),(0.1,0.1,1.),(20,20,20))
b = G.bboxOfCells(a)
a = C.node2Center(a); a = C.addVars([a,b])
C.convertArrays2File([a], 'out.plt')
```

Example file : Examples/generator/bboxOfCellsPT.py

```
# - bboxOfCells (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C

a = G.cart((0.,0.,0.),(0.1,0.1,1.),(20,20,20))
a = G.bboxOfCells(a)
t = C.newPyTree(['Base',3]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/CEBBIntersection.py

```
# - CEBBIntersection (array) -
import Generator as G
import Converter as C
import Transform as T
ni = 11; nj = 3; nk = 11
a1 = G.cart((0.,0.,0.), (0.1,0.1,0.2),(ni, nj,nk))
a2 = G.cart((1.,0.,0.), (0.1,0.1,0.2),(ni, nj,nk))
a2 = T.rotate(a2, (0,0,0), (0,0,1), 12.)
print G.CEBBIntersection(a1, a2)
```

Example file : Examples/generator/CEBBIntersectionPT.py

```
# - CEBBIntersection (pyTree)-
import Generator.PyTree as G
import Converter.PyTree as C
import Transform.PyTree as T

ni = 11; nj = 3; nk = 11
a1 = G.cart((0.,0.,0.), (0.1,0.1,0.2),(ni, nj,nk))
a2 = G.cart((1.,0.,0.), (0.1,0.1,0.2),(ni, nj,nk))
a2 = T.rotate(a2, (0,0,0), (0,0,1), 12.)
print G.CEBBIntersection(a1, a2)
```

Example file : Examples/generator/bboxIntersection.py

```
# - bboxIntersection (array) -
import Generator as G

ni = 11; nj = 3; nk = 11
a1 = G.cart((0.,0.,0.), (0.1,0.1,0.2),(ni, nj,nk))
a2 = G.cart((0.5,0.05,0.01), (0.1,0.1,0.2),(ni, nj,nk))
intersect = G.bboxIntersection(a1,a2) ; print intersect
```

Example file : Examples/generator/bboxIntersectionPT.py

```
# - bboxIntersection (pyTree) -
import Generator.PyTree as G

ni = 11; nj = 3; nk = 11
a1 = G.cart((0.,0.,0.), (0.1,0.1,0.2),(ni, nj,nk))
a2 = G.cart((0.5,0.05,0.01), (0.1,0.1,0.2),(ni, nj,nk))
intersect = G.bboxIntersection(a1, a2) ; print intersect
```

Example file : Examples/generator/checkPointInCEBB.py

```
# - checkPointInCEBB (array) -
import Generator as G
import Transform as T
import Converter as C

Ni = 20 ; Nj = 20
a1 = G.cart((0,0,0),(1./Ni,0.5/Nj,1),(Ni,Nj,2))
a2 = G.cart((-0.1,0,0),(0.5/Ni, 0.5/Nj, 1), (Ni,Nj,2))
```

```
a2 = T.rotate(a2, (-0.1,0,0), (0,0,1), 0.22)

# Verifie si un point est dans la CEBB de mesh2
val = G.checkPointInCEBB(a2, (0.04839, 0.03873, 0.5)) ; print val
```

Example file : Examples/generator/checkPointInCEBBPT.py

```
# - checkPointInCEBB (pyTree) -
import Generator.PyTree as G
import Transform.PyTree as T
import Converter.PyTree as C

Ni = 20; Nj = 20
a2 = G.cart((-0.1,0,0),(0.5/Ni, 0.5/Nj, 1), (Ni,Nj,2))
a2 = T.rotate(a2, (-0.1,0,0), (0,0,1), 0.22)

# Verifie si un point est dans la CEBB de a2
val = G.checkPointInCEBB(a2, (0.04839, 0.03873, 0.5)); print val
```

Example file : Examples/generator/getVolumeMap.py

```
# - getVolumeMap (array) -
import Generator as G
import Converter as C

a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,3))
vol = G.getVolumeMap(a)
vol = C.center2Node(vol)
vol = C.addVars([a,  vol])
C.convertArrays2File([vol], "out.plt", "bin_tp")
```

Example file : Examples/generator/getVolumeMapPT.py

```
# - getVolumeMap (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C

a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,3))
a = G.getVolumeMap(a)
t = C.newPyTree(['Base',3]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/getNormalMap.py

```
# - getNormalMap (array) -
import Geom as D
import Generator as G
import Converter as C

# 2D structure
a = D.sphere((0,0,0), 1, 50)
n = G.getNormalMap(a)
n = C.center2Node(n)
n = C.addVars([a, n])
C.convertArrays2File([n], "out1.plt", "bin_tp")

# 2D non-structure
a = D.sphere((0,0,0), 1, 50)
a = C.convertArray2Tetra(a)
n = G.getNormalMap(a)
n = C.center2Node(n)
n = C.addVars([a, n])
C.convertArrays2File([n], "out2.plt", "bin_tp")
```

Example file : Examples/generator/getNormalMapPT.py

```
# - getNormalMap (pyTree) -
import Geom.PyTree as D
import Generator.PyTree as G
import Converter.PyTree as C

a = D.sphere((0,0,0), 1, 50)
a = G.getNormalMap(a)
t = C.newPyTree(['Base',2]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/getCellPlanarity.py

```
# - getCellPlanarity (array) -
import Converter as C
import Generator as G
import Geom as D

a = D.sphere( (0,0,0), 1., 10)
p = G.getCellPlanarity(a)
p = C.center2Node(p)
a = C.addVars([a, p])
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/getCellPlanarityPT.py

```
# - getCellPlanarity (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Geom.PyTree as D

a = D.sphere( (0,0,0), 1., 10)
a = G.getCellPlanarity(a)
t = C.newPyTree(['Base',2]); t[1][2].append(a)
C.convertPyTree2File(t, "out.cgns", "bin_cgns")
```

Example file : Examples/generator/enforceX.py

```
# - enforceX (array) -
import Generator as G
import Converter as C

Ni = 50 ; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
# Monotonic distribution
b = G.enforceX(a, 0.3, 0.001, (13,25))
C.convertArrays2File([b], "new.plt", "bin_tp")
```

Example file : Examples/generator/enforceXPT.py

```
# - enforceX (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C

Ni = 50; Nj = 50; Nk = 2
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,Nk))
a = G.enforceX(a, 0.3, 0.001, (13,25))
t = C.newPyTree(['Base',3]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/enforce.py

```
# - enforceX, enforceY, ... monotonic (array) -
import Converter as C
import Generator as G

a = G.cart( (0,0,0), (1,1,1), (20,20,10) )
b = G.enforceX(a, 5., 0.2, 10, 5 )
b = G.enforceX(b, 15., 0.2, 10, 5 )
b = G.enforceY(b, 10., 0.1, 5, 5 )
b = G.enforcePlusY(b, 0.01, 20, 5 )
C.convertArrays2File([a,b], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/enforcePT.py

```
# - enforceX, enforceY, ... monotonic (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C

a = G.cart( (0,0,0), (1,1,1), (20,20,10) )
b = G.enforceX(a, 5., 0.2, 10, 5 )
b = G.enforceX(b, 15., 0.2, 10, 5 )
b = G.enforceY(b, 10., 0.1, 5, 5 )
t = C.newPyTree(['Base']); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/enforceMoinsX.py

```
# - enforceMoinsX (array) -
import Generator as G
import Converter as C

Ni = 50; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
b = G.enforceMoinsX(a, 1.e-3, (10,15))
C.convertArrays2File([b], "out.plt", "bin_tp")
```

Example file : Examples/generator/enforceMoinsXPT.py

```
# - enforceMoinsX (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

Ni = 50; Nj = 50; Nk = 1
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,Nk))
b = G.enforceMoinsX(a, 1.e-3, (10,15))
t = C.newPyTree(['Base',2]); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/enforcePlusX.py

```
# - enforcePlusX (array) -
import Generator as G
import Converter as C

# Distribution
Ni = 50; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
b = G.enforcePlusX(a, 1.e-3, (10,20))
C.convertArrays2File([b], "new.plt", "bin_tp")
```

Example file : Examples/generator/enforcePlusXPT.py

```
# - enforcePlusX (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

# Distribution
```

```
Ni = 50; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
b = G.enforcePlusX(a, 1.e-3, (10,20))
t = C.newPyTree(['Base',2]); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/enforceY.py

```
# - enforceY (array) -
import Generator as G
import Converter as C

# Distribution
Ni = 50; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
b = G.enforceY(a, 0.3, 0.001, (10,15))
C.convertArrays2File([b], "new.plt", "bin_tp")
```

Example file : Examples/generator/enforceYPT.py

```
# - enforceY (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C

Ni = 50; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
b = G.enforceY(a, 0.3, 0.001, (10,15))
t = C.newPyTree(['Base',2]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/enforceMoinsZ.py

```
# - enforceMoinsZ (array) -
import Generator as G
import Converter as C

Ni = 50; Nj = 1; Nk = 50
a = G.cart((0,0,0), (1./(Ni-1), 1., 0.5/(Nk-1)), (Ni,Nj,Nk))
b = G.enforceMoinsZ(a, 1.e-3, 10,15)
C.convertArrays2File([b], "out.plt")
```

Example file : Examples/generator/enforceLine.py

```
# - enforceLine (array) -
import Generator as G
import Converter as C
import Geom as D

Ni = 50; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
b = D.line((0.,0.2,0.), (1.,0.2,0.), 20)
c = G.enforceLine(a, b, 0.01, (5,3))
C.convertArrays2File([c], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/enforceLinePT.py

```
# - enforceLine (pyTree)-
import Generator.PyTree as G
import Converter.PyTree as C
import Geom.PyTree as D

Ni = 50; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
b = D.line((0.,0.2,0.), (1.,0.2,0.), 20)
a = G.enforceLine(a, b, 0.01, (5,3))
t = C.newPyTree(['Base',2]); t[1][2].append(a)
C.convertPyTree2File(t,"out.cgns","bin_cgns")
```

Example file : Examples/generator/enforcePoint.py

```
# - enforcePoint (array) -
import Converter as C
import Generator as G

# distribution
Ni = 20 ; Nj = 20
a = G.cart((0,0,0), (1./(Ni-1),5./(Nj-1),1), (Ni,Nj,1))
b = G.enforcePoint(a, 0.5)
C.convertArrays2File([b], "new.plt","bin_tp")
```

Example file : Examples/generator/enforcePointPT.py

```
# - enforcePoint (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

Ni = 20; Nj = 20
a = G.cart((0,0,0), (1./(Ni-1),5./(Nj-1),1), (Ni,Nj,1))
b = G.enforcePoint(a, 0.5)
t = C.newPyTree(['Base',2]); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/enforceCurvature.py

```
# - enforceCurvature (array) -
import Geom as D
import Generator as G
import Converter as C
import Transform as T

# Naca profile with lines
a = D.naca(12., 501)
l1 = D.getLength(a)
a2 = D.line((1.,0.,0.),(2.,0.,0.), 500)
l2 = D.getLength(a2)
b = T.join(a, a2)
c = D.line((2.,0.,0.),(1.,0.,0.), 500)
res = T.join(c, b)

# Distribution on the profile
l = l1+2*l2
Ni = 100; Nj = 100
p1 = l2/l; p2 = (l2+l1)/l
h = (p2-p1)/(Ni-1)
distrib = G.cart((p1,0,0), (h, 0.25/Nj,1), (Ni,Nj,1))
distrib = G.enforceCurvature(distrib, res, 0.6)
C.convertArrays2File([distrib], "distrib.plt", "bin_tp")
```

Example file : Examples/generator/enforceCurvaturePT.py

```
# - enforceCurvature (pyTree) -
import Geom.PyTree as D
import Generator.PyTree as G
import Converter.PyTree as C

a = D.naca(12., 501)

# Distribution on the profile
Ni = 20; Nj = 20; Nk = 1; h = 1./(Ni-1)
b = G.cart((0,0,0), (h, 0.25/Nj,1), (Ni,Nj,Nk))
b = G.enforceCurvature(b, a, 0.6)
t = C.newPyTree(['Base',2]); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/addPointInDistribution.py

```
# - addPointInDistribution (array) -
import Generator as G
import Converter as C

# Distribution
Ni = 50 ; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
b = G.addPointInDistribution( a, Ni )
C.convertArrays2File([b], 'out.plt')
```

Example file : Examples/generator/addPointInDistributionPT.py

```
# - addPointInDistribution (pyTree)-
import Generator.PyTree as G
import Converter.PyTree as C

# Distribution
Ni = 50; Nj = 50
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,2))
b = G.addPointInDistribution( a, Ni )
t = C.newPyTree(['Base',2]); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/close.py

```
# - close (array) -
import Converter as C
import Generator as G
import Transform as T

# Close un maillage non-structure
a1 = G.cart((0,0,0), (1,1,1), (10,10,1))
b1 = C.convertArray2Tetra(a1)
a2 = G.cart((9+1.e-2,0,0), (0.2,1,1), (10,10,1))
b2 = C.convertArray2Tetra(a2)
b = T.join(b1, b2)
b = G.close(b, 1.e-1)
C.convertArrays2File([b], 'out.plt', 'bin_tp')

# Close une liste de maillages structures
B = G.close([a1,a2], 1e-1)
C.convertArrays2File(B, 'out2.plt', 'bin_tp')
```

Example file : Examples/generator/closePT.py

```
# - close (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

a1 = G.cart((0,0,0), (1,1,1), (10,10,1))
a2 = G.cart((9+1.e-2,0,0), (1,1,1), (10,10,1))
a3 = G.cart((0,-5.01,0),(1,1,1),(19,6,1))
a4 = G.cart((0,9.0001,0),(1,1,1),(10,6,1))
a5 = G.cart((9.01,9.0002,0),(1,1,1),(10,6,1))
t = C.newPyTree(['Base',2]); t[1][2] = t[1][2] + [a1,a2,a3,a4,a5]
t = G.close(t, 1.e-1)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/selectInsideElts.py

```
# - selectInsideElts (array) -
import Converter as C
import Generator as G
```

```
import Geom as D

a = G.cart( (0,0,0), (1,1,1), (10,10,1))
a = C.convertArray2Tetra(a)

b = D.circle( (5,5,0), 3.)
b = C.convertArray2Tetra(b)

a = G.selectInsideElts(a, [b])

C.convertArrays2File([a,b], 'out.plt')
```

Example file : Examples/generator/map.py

```
# - map (array) -
import Geom as D
import Generator as G
import Converter as C

# Map on a curve
l = D.line( (0,0,0), (1,1,0) )
Ni = 10
d = G.cart( (0,0,0), (1./(Ni-1),1.,1.), (Ni,1,1) )
m = G.map(l, d)
C.convertArrays2File([m], "out.plt", "bin_tp")

# Map on a structured surface
ni = 2 ; nj = 3
a = G.cart((0,0,0), (1,1,1), (ni,nj,1))
C.setValue(a, (1,1,1), [1.,1.,2.])
C.setValue(a, (1,2,1), [1.,2.,5.])
C.setValue(a, (1,3,1), [1.,3.,2.])
C.setValue(a, (2,1,1), [2.,1.,2.])
C.setValue(a, (2,2,1), [2.,2.,5.])
C.setValue(a, (2,3,1), [2.,3.,2.])
b = D.bezier(a, 10, 10)
Ni = 50; Nj = 30
d = G.cart( (0,0,0), (1./(Ni-1),1./(Nj-1),1.), (Ni,Nj,1) )
d = G.enforceX(d, 0.5, 0.01, (10,20))
d = G.enforceY(d, 0.5, 0.01, (10,20))
b = G.map(b, d)
C.convertArrays2File([b], "out2.plt", "bin_tp")

# Map in a direction
a = G.cylinder((0,0,0), 0.5, 2., 0, 60, 1., (20,20,1))
Ni = 10
d = G.cart( (0,0,0), (1./(Ni-1),1.,1.), (Ni,1,1) )
d = G.enforcePlusX(d, 0.01, (10,20))
a = G.map(a, d, 2)
C.convertArrays2File([a], "out3.plt", "bin_tp")
```

Example file : Examples/generator/mapPT.py

```
# - map (pyTree)-
import Geom.PyTree as D
import Generator.PyTree as G
import Converter.PyTree as C

l = D.line( (0,0,0), (1,1,0) )
Ni = 11; dist = G.cart( (0,0,0), (1./(Ni-1),1.,1.), (Ni,1,1) )
l = G.map(l, dist)
t = C.newPyTree(['Base',1]); t[1][2].append(l)
C.convertPyTree2File(t, "out.cgns", "bin_cgns")
```

Example file : Examples/generator/mapSplit.py

```
# - mapSplit (array) -
import Generator as G
import Transform as T
import Converter as C

# polyline
a = C.convertFile2Arrays("curve1.svg", "fmt_svg")
# distribution
ax0=a[0][1][0][0]
ay0=a[0][1][1][0]
az0=a[0][1][2][0]
Ni=41
dist=G.cart((0,0,0),(1./(Ni-1),1,1),(Ni,1,1))
dist=G.enforceX(dist, 15.5/(Ni-1), 0.005, 2,5)
dist=G.enforceX(dist, 27.5/(Ni-1), 0.005, 2,5)

# mapSlit
a= G.mapSplit(a[0],dist,3e-01)

# save in file
C.convertArrays2File(a,"out.plt")
```

Example file : Examples/generator/mapSplitPT.py

```
# - mapSplit (pyTree) -
import Generator.PyTree as G
import Transform.PyTree as T
import Converter.PyTree as C

# polyline
t = C.convertFile2PyTree("polyligne1.svg", "fmt_svg")
# distribution
Ni=41
dist=G.cart((0,0,0),(1./(Ni-1),1,1),(Ni,1,1))
dist=G.enforceX(dist, 6.5/(Ni-1), 0.005, 2,5)
dist=G.enforceX(dist, 30.5/(Ni-1), 0.005, 2,5)

# mapSlit
zones = G.mapSplit(t,dist,2e-01)
t[1][2] = t[1][2] + zones

# save in file
C.convertPyTree2File(t,"out.cgns")
```

Example file : Examples/generator/densify.py

```
# - densify (array) -
import Generator as G
import Converter as C
import Geom as D

a = D.circle( (0,0,0), 1., 10 )
b = G.densify(a, 0.01)
C.convertArrays2File([b], 'out.plt')
```

Example file : Examples/generator/densifyPT.py

```
# - densify (pyTree) -
import Generator.PyTree as G
import Converter.PyTree as C
import Geom.PyTree as D

a = D.circle( (0,0,0), 1., 10)
b = G.densify(a, 0.01)
t = C.newPyTree(['Base',1]); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/grow.py

```
# - grow (array) -
import Converter as C
import Generator as G
import Geom as D

a = D.sphere( (0,0,0), 1., 50 )
n = G.getNormalMap(a)
n = C.center2Node(n)
n[1] = n[1]*100.
b = G.grow(a, n)

C.convertArrays2File([b], 'out.plt')
```

Example file : Examples/generator/growPT.py

```
# - grow (pyTree)-
import Converter.PyTree as C
import Generator.PyTree as G
import Geom.PyTree as D

a = D.sphere( (0,0,0), 1., 50 )
a = G.getNormalMap(a)
a = C.center2Node(a)
a = C.rmVars(a, 'FlowSolution#Centers')
b = G.grow(a, ['sx','sy','sz'])

t = C.newPyTree(['Base1',2,'Base2',3]);
t[1][2].append(a); t[2][2].append(b)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/generator/addNormalLayers.py

```
# - addNormalLayers (array) -
import Generator as G
import Converter as C
import Geom as D

d = C.array('d', 3, 1, 1) ;
d[1][0,0] = 0.1 ; d[1][0,1] = 0.2 ; d[1][0,2] = 0.3

a = D.sphere( (0,0,0), 1, 50 )
a = G.addNormalLayers(a, d)
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/TFI.py

```
# - TFI 2D structuree (array)
# - TFI 3D structuree (array)
# - TFI TRI (array)
import Converter as C
import Generator as G
import Geom as D
import numpy as N
import Transform as T

#------------------
# TFI 2D structuree
#------------------
P0 = (0,0,0)
P1 = (5,0,0)
P2 = (0,7,0)
P3 = (5,7,0)

# Geometrie
```

```
d1 = D.line(P0, P1)
d2 = D.line(P2, P3)

pts = C.array('x,y,z',5,1,1)
x = pts[1][0]; y = pts[1][1]; z = pts[1][2]
x[0] = 0.; y[0] = 0.; z[0] = 0.
x[1] =-2.; y[1] = 2.; z[1] = 0.
x[2] =-3.; y[2] = 3.; z[2] = 0.
x[3] = 2.; y[3] = 5.; z[3] = 0.
x[4] = 0.; y[4] = 7.; z[4] = 0.
b1 = D.bezier(pts)

pts = C.array('x,y,z',5,1,1)
x = pts[1][0]; y = pts[1][1]; z = pts[1][2]
x[0] = 5.; y[0] = 0.; z[0] = 0.
x[1] = 3.; y[1] = 2.; z[1] = 0.
x[2] = 2.; y[2] = 3.; z[2] = 0.
x[3] = 6.; y[3] = 5.; z[3] = 0.
x[4] = 5.; y[4] = 7.; z[4] = 0.
b2 = D.bezier( pts )

C.convertArrays2File([d1, d2, b1, b2], "geom.plt", "bin_tp")

# Discretisation reguliere de chaque ligne
Ni = 20
Nj = 10
r = G.cart((0,0,0), (1./(Ni-1),1,1), (Ni,1,1))
q = G.cart((0,0,0), (1./(Nj-1),1,1), (Nj,1,1))
r1 = G.map(d1, r)
r2 = G.map(d2, r)
r3 = G.map(b1, q)
r4 = G.map(b2, q)

# TFI 2D
m = G.TFI([r1, r2, r3, r4])
C.convertArrays2File([r1,r2,r3,r4,m], 'tfi2d.plt', 'bin_tp')

#------------------
# TFI 3D structuree
#------------------
xo = 0.; yo = 0.; zo = 0.
nx = 21; ny = 21; nz = 21
hx = 1./(nx-1); hy = 1./(ny-1); hz = 1./(nz-1)

# grilles z = cste
fzmin = G.cart((xo,yo,zo), (hx,hy,1.), (nx,ny,1))
fzmax = T.translate(fzmin, (0.,0.,1.))

# grilles x = cste
fxmin = G.cart((xo,yo,zo),(1,hy,hz),(1,ny,nz))
fxmin = T.reorder(fxmin,(3,1,2))
fxmax = T.translate(fxmin, (1.,0.,0.))

# grilles y = cste
fymin = G.cart((xo,yo,zo),(hx,1.,hz),(nx,1,nz))
fymin = T.reorder(fymin,(1,3,2))
fymax = T.translate(fymin, (0.,1.,0.))

r = [fxmin,fxmax,fymin,fymax,fzmin,fzmax]
m = G.TFI(r)
C.convertArrays2File(r+[m],"tfi3d.plt","bin_tp")

#---------
# TFI tri
#---------
```

```
l1 = D.line((0,0,0),(0,1,0), 15)
l2 = D.line((0,0,0),(1,0,0), 15)
l3 = D.line((1,0,0),(0,1,0), 15)
tri = G.TFI([l1,l2,l3])
C.convertArrays2File([tri],"tfitri.plt","bin_tp")
```

Example file : Examples/generator/TFIPT.py

```
# - TFI (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Geom.PyTree as D
import numpy as N

# Geometrie
P0 = (0,0,0); P1 = (5,0,0); P2 = (0,7,0); P3 = (5,7,0)

d1 = D.line(P0, P1);d2 = D.line(P2, P3)
d3 = D.line(P0, P2);d4 = D.line(P1, P3)

# Discretisation reguliere de chaque ligne
Ni = 20; Nj = 10
r = G.cart((0,0,0), (1./(Ni-1),1,1), (Ni,1,1))
q = G.cart((0,0,0), (1./(Nj-1),1,1), (Nj,1,1))
r1 = G.map(d1, r); r2 = G.map(d2, r)
r3 = G.map(d3, q); r4 = G.map(d4, q)

m = G.TFI([r1, r2, r3, r4])
t = C.newPyTree(['Base',2]); t[1][2].append(m)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/TTM.py

```
# - TTM (array) -
import Converter as C
import Generator as G
import Geom as D

P0 = (0,0,0)
P1 = (5,0,0)
P2 = (0,7,0)
P3 = (5,7,0)

# Geometrie
d1 = D.line(P0, P1)
d2 = D.line(P2, P3)
pts = C.array('x,y,z', 5, 1, 1)
x = pts[1][0]; y = pts[1][1]; z = pts[1][2]

x[0] = 0.  ; y[ 0 ] = 0.; z[ 0 ] = 1.
x[1] =-2.  ; y[ 1 ] = 2.; z[ 1 ] = 1.
x[2] =-3.  ; y[ 2 ] = 3.; z[ 2 ] = 1.
x[3] = 2.  ; y[ 3 ] = 5.; z[ 3 ] = 1.
x[4] = 0.  ; y[ 4 ] = 7.; z[ 4 ] = 1.
b1 = D.bezier(pts)

x[0] = 5. ; y[ 0 ] = 0.; z[ 0 ] = 1.
x[1] = 3. ; y[ 1 ] = 2.; z[ 1 ] = 1.
x[2] = 2. ; y[ 2 ] = 3.; z[ 2 ] = 1.
x[3] = 6. ; y[ 3 ] = 5.; z[ 3 ] = 1.
x[4] = 5. ; y[ 4 ] = 7.; z[ 4 ] = 1.
b2 = D.bezier( pts )

C.convertArrays2File([d1, d2, b1, b2], "geom.plt", "bin_tp")
```

```
# Discretisation reguliere de chaque lignes
Ni = 20
Nj = 10
r = G.cart((0,0,0), (1./(Ni-1),1,1), (Ni,1,1))
q = G.cart((0,0,0), (1./(Nj-1),1,1), (Nj,1,1))
r1 = G.map(d1, r)
r2 = G.map(d2, r)
r3 = G.map(b1, q)
r4 = G.map(b2, q)

# TTM
m = G.TTM(r1, r2, r3, r4)
C.convertArrays2File([m], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/TTMPT.py

```
# - TTM (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Geom.PyTree as D

P0 = (0,0,0); P1 = (5,0,0); P2 = (0,7,0); P3 = (5,7,0)

# Geometrie
d1 = D.line(P0,P1); d2 = D.line(P2,P3)
d3 = D.line(P0,P2); d4 = D.line(P1,P3)

# Discretisation reguliere de chaque ligne
Ni = 20; Nj = 10
r = G.cart((0,0,0), (1./(Ni-1),1,1), (Ni,1,1))
q = G.cart((0,0,0), (1./(Nj-1),1,1), (Nj,1,1))
r1 = G.map(d1, r); r2 = G.map(d2, r)
r3 = G.map(d3, q); r4 = G.map(d4, q)

# TTM
m = G.TTM(r1, r2, r3, r4)
t = C.newPyTree(['Base',2]); t[1][2].append(m)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/generator/hyper2d.py

```
# - hyper2D (array) -
import Geom as D
import Generator as G
import Converter as C

msh = D.naca(12., 5001)

# Distribution
Ni = 300 ; Nj = 50
distrib = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))

a = G.hyper2D(msh, distrib, "C")
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/hyper2dPT.py

```
# - hyper2D (pyTree) -
import Geom.PyTree as D
import Generator.PyTree as G
import Converter.PyTree as C

line = D.naca(12., 5001)
# Distribution
Ni = 300 ; Nj = 50
distrib = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1), (Ni,Nj,1))
```

```
a = G.hyper2D(line, distrib, "C")
t = C.newPyTree(['Base',2]); t[1][2].append(a)
C.convertPyTree2File(t,"out.cgns","bin_cgns")
```

Example file : Examples/generator/polyLineMesher.py

```
# - polyLineMesher (array) -
import Converter as C
import Generator.PolyLine as GP
import Generator as G
import Post as P
import Geom as D
import Transform as T

# Lecture geometrie 2D cree avec tecplot
a = C.convertFile2Arrays('fusee.plt', 'bin_tp')

m = []
for i in a:
    i = T.reorder(i, (-1,2,3))
    i = C.convertArray2Tetra(i)
    i = G.close(i, 1.e-2)
    m.append(i)
C.convertArrays2File(m, 'input.plt', 'bin_tp')

# Donnees
h = 0.02
hf = 0.0001
density = 500
ext = 5

# Par familles
coords = []
walls = []
overlaps = []
for i in m:
    b = GP.polyLineMesher(i, h, hf, density, ext)
    coords.append(b[0])
    walls.append(b[1])
    overlaps.append(b[2])

# A plat
meshes = []
for i in coords:
    meshes = meshes + i

C.convertArrays2File(m+meshes, 'out.plt', 'bin_tp')
```

Example file : Examples/generator/polyLineMesherPT.py

```
# - polyLineMesher (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Transform.PyTree as T

tb = C.convertFile2PyTree('fusee.plt', 'bin_tp')
tb[1][2][0] = T.reorder(tb[1][2][0], (-1,2,3))

h = 0.02 ; hf = 0.0001 ; density = 500 ; ext = 5

res = G.polyLineMesher(tb, h, hf, density, ext)
t = res[0]; h = res[1]; density = res[2]
C.convertPyTree2File(t, "out.cgns", "bin_cgns")
```

Example file : Examples/generator/polyC1Mesher.py

```
# - polyC1Mesher (array) -
import Converter as C
import Generator.PolyC1 as GP
import Generator as G
import Post as P
import Geom as D
import Transform as T

def mesher(a, h, hp, density):
    Ni = 150
    d = G.cart( (0,0,0), (1./(Ni-1),1.,1.), (Ni,1,1) )
    a = G.map(a[0], d)
    A = [a]
    b = []

    size = 100
    cvs = 0.
    while ( size  > 10 and cvs < 15.) :
        b = []
        for i in A:
            c = T.splitSpline(i, cvs)
            b = b + c
            size = len(b)
            cvs = cvs + 1
    C.convertArrays2File(b, 'geom.plt', 'bin_tp')
    m = GP.polyC1Mesher(b, h, hp, density)

    return m

# Lecture de la geometrie
a = C.convertFile2Arrays('curve1.svg', 'fmt_svg')
a = T.homothety(a,(0,0,0),0.01)

h = 0.5 ; hp = 0.001 ; density = 50.
m = mesher(a, h, hp, density)

for i in m[0]:
    v = G.getVolumeMap(i)
    min = C.getMinValue(v, 'vol')
    if (min <= 0):
        print 'negative volume detected.'

C.convertArrays2File(m[0], 'out.plt', 'bin_tp')
```

Example file : Examples/generator/polyC1MesherPT.py

```
# - polyC1Mesher (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.PyTree as P
import Geom.PyTree as D
import Transform.PyTree as T

def mesher(a, h, hp, density):
    Ni = 150
    d = G.cart( (0,0,0), (1./(Ni-1),1.,1.), (Ni,1,1) )
    a = G.map(a, d)

    size = 100
    cvs = 0.
    while ( size  > 10 and cvs < 15.) :
        b = []
        c = T.splitSpline(a, cvs)
        size = len(c)
        cvs = cvs + 1
```

```
    tb = C.newPyTree(['Body',1]); tb[1][2] = tb[1][2] + c
    m = G.polyC1Mesher(tb, h, hp, density)
    return m

# Lecture de la geometrie
a = [D.naca(12., 101)]
a = T.reorder(a, (-1,2,3))

h = 10. ; hp = 0.001 ; density = 100.

res = mesher(a, h, hp, density)

C.convertPyTree2File(res[0], 'out.cgns')
```

Example file : Examples/generator/naca.py

```
# Test des fonctions :
# - naca
# - line
# - enforce
# - hyper2D
# Generateur de profil Naca0012
import Converter as C
import Generator as G
import Geom as D
import Transform as T

# Put a naca profile in msh
msh = D.naca( 12., 5001)
l1 = D.getLength(msh)

# Put a line in msh2
msh2 = D.line((1.,0.,0.),(20.,0.,0.),5001)
l2 = D.getLength(msh2)

# Join the two meshes
msh = T.join(msh, msh2)

# Add another line
msh2 = D.line((20.,0.,0.),(1.,0.,0.),5001)
l3 = D.getLength(msh2)

msh = T.join(msh2, msh)

C.convertArrays2File([msh], "naca.plt","bin_tp")

# distribution
Ni = 200
Nj = 100
distrib = G.cart( (0,0,0), (1./(Ni-1), 20./(Nj-1),1), (Ni,Nj,1))

distrib = G.enforcePlusY(distrib, 2.e-3, 50, 30)
distrib = G.enforceMoinsY(distrib, 2., 80, -60)
distrib = G.enforcePlusX(distrib, 1.e-4, 30, 50)
distrib = G.enforceMoinsX(distrib, 0.1, 80, -30)
distrib = G.enforceX(distrib, (l1*0.5)/(0.5*l1+l2), 1.25e-3, 30, 20)

distrib = G.enforcePoint(distrib, (l1*0.5)/(0.5*l1+l2))

distrib2 = T.symetrize(distrib, (0.,0.,0.), (0,1,0), (0,0,1))
distrib2 = T.reorder(distrib2, (-1,2,3))
distrib = T.join(distrib2, distrib)
distrib = T.contract(distrib, (0,0,0), (0,1,0), (0,0,1), 0.5)
distrib = T.translate(distrib, (0.5,0,0))
C.convertArrays2File([distrib], "distrib.plt", "bin_tp")
```

```
msh = G.hyper2D(msh, distrib, "C")
msh = G.close(msh, 1.e-6)

msh = T.addkplane(msh)
C.convertArrays2File([msh], "out.plt", "bin_tp")
```

Example file : Examples/generator/gencart2.py

```
# Test des fonctions :
# - gencart

from elsA_user import *
import Converter as C
import Generator as G
import Generator.Cartesian as GC

blkPaleArrays = []
a = C.convertFile2Arrays('pale_bo105_1.tp','fmt_tp")
a1 = a[0]
a = C.convertFile2Arrays('pale_bo105_2.tp','fmt_tp")
a2 = a[0]
del a

for i in range(4) :
    blkPaleArrays.append(a1)
    blkPaleArrays.append(a2)

# NUMERICS
Num1 = numerics(name='Num1')
Num1.set('automesh', 'active')
Num1.set('automesh_type', 'cartesian')     # 0 : cartesian, 1 : cylinder
Num1.set('dfxm', 20.)  # Distance for far boundary x-
Num1.set('dfxp', 30.)  # Distance for far boundary x+
Num1.set('dfym', 20.)  # Distance for far boundary y-
Num1.set('dfyp', 20.)  # Distance for far boundary y+
Num1.set('dfzm', 30.)  # Distance for far boundary z-
Num1.set('dfzp', 20.)  # Distance for far boundary z+
Num1.set('vmin', 8)                # minimum size of grids
Num1.set('snear_type', 1)
Num1.set('snear_mul', 1.)
Num1.set('step_factor', 2.)
Num1.set('start_from_level', 2)      # start proximity from level i

arrays = GC.gencart(blkPaleArrays, [Num1])
C.convertArrays2File(arrays, "new.plt", "bin_tp")
```