Transform 2.5

Stephanie Peron, Christophe Benoit, Gaelle Jeanfaivre, Pascal Raud
- Onera -

1 Transform: mesh transformation module

1.1 Preamble

Transform module performs simple transformations of meshes. It works on arrays (as defined in Converter documentation) or on CGNS/Python trees (pyTrees) containing grid coordinates information.

This module is part of Cassiopee, a free open-source pre- and post-processor for CFD simulations.

To use it with the array interface, you have to import Transform module:

```
import Transform as T
```

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

To use it with the pyTree interface:

```
import Transform.PyTree as T
```

Then, a defines a zone node, A defines a list of zone nodes or a pyTree.

1.2 Simple operations

Warning: the pyTree functions *subzone, oneovern, join, reorder* **destroy the 'BCMatch' and 'BCNearMatch' boundary conditions. They can be rebuilt by connectMatch and connectNearMatch functions of Connector module.**

T.subzone: extract a subzone.
For a structured array, you must specify the min and max indices in i,j,k directions of the subzone:

```
b = T.subzone(a, (imin,jmin,kmin), (imax,jmax,kmax))
```

For an unstructured array, you must specify the vertices belonging to subzone (index starts from 1):

```
b = T.subzone(a, [1,2,...])
```

Extract a subzone of an unstructured array providing the indices of elements (index starts from 0):

```
b = T.subzone(a, [0,1,...], type='elements')
```

Extract a subzone of an unstructured array providing the indices of faces (for unstructured zones with basic elements: indFace=indElt*numberOfFaces+noFace, for NGON zones: use the natural face indexing, starting from 1):

```
b = T.subzone(a, [1,2,...], type='faces')
```

(See: subzone.py) (See: subzonePT.py)

T.join: join two arrays in one (if possible) or join a list of arrays in one (if possible). *Warning:* for the pyTree version, join does not take into account modifications of boundary conditions:

```
c = T.join(a, b, tol=1.e-10) .or. c = T.join(A, tol=1.e-10)
```

(See: join.py) (See: joinPT.py)

T.merge: merge a set of curvilinear grids (surface grids must be k=1) if possible. Parameter sizeMax defines the maximum size of merged grids. dir is the constraint direction along which the merging is prefered. Default value is 0 (no prefered direction), 1 for i, 2 for j, 3 for k. alphaRef can be used for surface grids and avoids merging adjacent zones sharing an angle deviating of alphaRef to 180:

```
B = T.merge(A, sizeMax=1000000000, dir=0, tol=1.e-10, alphaRef=180.)
```

(See: merge.py) (See: mergePT.py)

T.mergeCart: merge a set of Cartesian grids (patch grids) if possible. Parameter sizeMax defines the maximum size of merged grids:

```
B = T.mergeCart(A, sizeMax=1000000000, tol=1.e-10)
```

(See: mergeCart.py) (See: mergeCartPT.py)

T.patch: patch (replace) an array b into an array a, from position (i,j,k) when dealing with structured arrays, and at given nodes (specified as a list or a numpy array of indices starting from 1) when dealing with structured or unstructured arrays:

```
c = T.patch(a, b, position=(i,j,k)) .or. c = T.patch(a, b, nodes=[1,2,3])
```

(See: patch.py) (See: patchPT.py)

T.oneovern: extract one point over N points from a:

```
b = T.oneovern(a, (Ni,Nj,Nk)) .or. B = T.oneovern(A, (Ni,Nj,Nk))
```

(See: oneovern.py) (See: oneovernPT.py)

T.collapse: for a TRI zone a, collapse smallest edges of each triangle. Return a BAR array:

```
b = T.collapse(a) .or. B = T.collapse(A)
```

(See: collapse.py) (See: collapsePT.py)

T.reorder: change the (i,j,k) ordering of structured array a. In the following example, i1 becomes k2, j1 becomes -j2, k1 becomes -i2:

```
b = T.reorder(a, (3,-2,-1)) .or. B = T.reorder(A, (3,-2,-1))
```

Change the ordering of a TRI- or QUAD-array or a 2D NGON array. All elements are numbered in the same way:

```
b = T.reorder(a, (-1,)) .or. B = T.reorder(A, (-1,))
```
For pyTree version, the global top tree a belongs to can be specified to update its matching boundaries impacted by the reordering of a (toptree is modified):
```
a = T.reorder(a, (3,-2,-1), toptree=[])
```
(See: reorder.py) (See: reorderPT.py)

T.reorderAll: reorient surfaces zones consistently between them. All zones must have the same type, either structured or unstructured (TRI currently). The zones must be abutting or overlapping and the reorientation of the first zone is used to reorient the other ones. For the unstructured case, if the zones represent a piecewise closed volume, reorientation is guaranteed to be outward (by default). Parameter 'dir' allows the user to change ordering (default value is 1). If dir=-1, the reordering is performed in the opposite direction:
```
B = T.reorderAll(A, dir)
```
NB: if reorder fails, A is returned.
(See: reorderAll.py) (See: reorderAllPT.py)

T.makeDirect: make a structured zone direct (change eventually k ordering):
```
b = T.makeDirect(a) .or. B = T.makeDirect(A)
```
(See: makeDirect.py) (See: makeDirectPT.py)

T.makeCartesianXYZ: align a Cartesian mesh such that i,j, k are aligned with x,y,z respectively:
```
b = T.makeDirect(a) .or. B = T.makeCartesianXYZ(A)
```
(See: makeCartesianXYZ.py) (See: makeCartesianXYZPT.py)

T.addkplane: add one or more (z+1) plane:
```
b = T.addkplane(a, N=1) .or. B = T.addkplane(A, N=1)
```
(See: addkplane.py) (See: addkplanePT.py)

1.3 Mesh positioning

The following functions are applicable to both structured and unstructured grids. **T.translate**:

translate of a vector (0.,1.,0.):
```
b = T.translate(a, (0.,1.,0.)) .or. B = T.translate(A, (0.,1.,0.))
```
(See: translate.py) (See: translatePT.py)

T.rotate: make a rotation of center (0.2,0.2,0.) around z-axis of angle of 18 degrees:
```
b = T.rotate(a, (0.2,0.2,0.), (0.,0.,1.), 18.,vectors=None) .or. B = T.rotate(A, (0.2,0.2,0.), (0.,0.,1.),
18.,vectors=[])
```
Make a rotation of center (0.2,0.2,0.) around z-axis of angle of 18 degrees. The velocity vector is also rotated:

```
b = T.rotate(a, (0.2,0.2,0.), (0.,0.,1.), 18.,vectors=[['VelocityX','VelocityY','VelocityZ']]) .or. B = T.rotate(A, (0.2,0.2,0.), (0.,0.,1.), 18.,vectors=[['VelocityX','VelocityY','VelocityZ']])
```

Make a rotation of center (0.2,0.2,0.) transforming frame vector (e1,e2,e3) in frame vector (f1,f2,f3):

```
b = T.rotate(a, (0.2,0.2,0.), (e1,e2,e3), (f1,f2,f3)) .or. B = T.rotate(A, (0.2,0.2,0.), (e1,e2,e3), (f1,f2,f3))
```

Make a rotation of center (0.2,0.2,0.) wrt three angles (alpha, beta, gamma). alpha is a rotation along X (Ox-¿Ox, Oy-¿Oy1, Oz-¿Oz1), beta along y (Ox1-¿Ox2, Oy1-¿Oy1, Oz1-¿Oz2), gamma along z (Ox2-¿Ox3, Oy2-¿Oy3, Oz2-¿Oz2):

```
b = T.rotate(a, (0.2,0.2,0.), (alpha,beta,gamma)) .or. B = T.rotate(A, (0.2,0.2,0.), (alpha,beta,gamma))
```

(See: rotate.py) (See: rotatePT.py)

1.4 Mesh transformation

The following functions, except "perturbate", are applicable to both structured and unstructured grids. **T.cart2Cyl**: transform a mesh in Cartesian coordinates into cylindrical coordinates, such that the origin of the frame is (x0,y0,z0) and (0,0,1) means the cylindrical axis is (0z).

```
b = T.cart2Cyl(a, (x0,y0,z0), (0,0,1)) .or. B = T.cart2Cyl(A, (x0,y0,z0), (0,0,1))
```

(See: cart2Cyl.py) (See: cart2CylPT.py)

T.homothety: make an homothety of center C (0.,0.,0.) and of multiplication factor alpha=0.1 (CM' = alpha * CM):

```
b = T.homothety(a, (0.,0.,0.), 0.1) .or. B = T.homothety(A, (0.,0.,0.), 0.1)
```

(See: homothety.py) (See: homothetyPT.py)

T.contract: make a contraction of a, regarding a plane defined by point (0.,0.,0.) and by dir1 (1,0,0) and dir2 (0,1,0) and of multiplication factor alpha=0.1:

```
b = T.contract(a, (0.,0.,0.), (1,0,0), (0,1,0), 0.1) .or. B = T.contract(A, (0.,0.,0.), (1,0,0), (0,1,0), 0.1)
```

(See: contract.py) (See: contractPT.py)

T.scale: scale a mesh with a given constant factor or per direction:

```
b = T.scale(a, factor=0.1) .or. B = T.scale(A, factor=(0.1,0.2,0.3))
```

(See: scale.py) (See: scalePT.py)

T.symetrize: make a symmetry of a, considering the plane passing by a point (1.,2.,3.) and defined by two vectors (1,0,0) and (0,1,0). Beware the (i,j,k) trihedra may be modified. Use reorder to avoid this:

```
b = T.symetrize(a, (1.,2.,3.), (1,0,0), (0,1,0)) .or. B = T.symetrize(A, (1.,2.,3.), (1,0,0), (0,1,0))
```

(See: symetrize.py) (See: symetrizePT.py)

T.perturbate: perturbate randomly a mesh with a given radius. If dim=2, z coordinates are fixed, if dim=1, y and z coordinates are fixed:

```
b = T.perturbate(a, 0.1, dim=3)
```

(See: perturbate.py) (See: perturbatePT.py)

T.smooth: perform a Laplacian smoothing on a 'QUAD', 'TRI' array or a list of structured arrays with a weight eps, and niter smoothing iterations. Type=0 means isotropic Laplacian, type=1 means scaled Laplacian, type=2 means taubin smoothing. Constraints can be defined in order to avoid smoothing some points (for instance the exterior faces of a):

```
b = T.smooth(a, eps=0.5, niter=4, type=0, fixedConstraints=[], projConstraints=[])
```

(See: smooth.py) (See: smoothPT.py)

T.splitCurvatureAngle: split a i-array following curvature angle. If angle is lower than 180-45 degrees or greater than 180+45 degrees, curve is split:

```
A = T.splitCurvatureAngle(a, 45.)
```

(See: splitCurvatureAngle.py) (See: splitCurvatureAnglePT.py)

T.splitCurvatureRadius: split a i-array following curvature radius, using B-splines approximation. The curve can be closed or not. Parameter Rs is a threshold curvature radius, so that the initial curve is split at points of curvature radius lower than than Rs:

```
B = T.splitCurvatureRadius(a, cs)
```

(See: splitCurvatureRadius.py) (See: splitCurvatureRadiusPT.py)

T.splitConnexity: split an unstructured array into connex parts:

```
B = T.splitConnexity(a)
```

(See: splitConnexity.py) (See: splitConnexityPT.py)

T.splitSharpEdges: split an array into smooth parts. AlphaRef specifies the split angle between neighbouring elements:

```
B = T.splitSharpEdges(a, alphaRef=30.)
```

(See: splitSharpEdges.py) (See: splitSharpEdgesPT.py)

T.splitBAR: split a 'BAR' array into 2 BAR arrays delimited by the node N (start 0):

```
B = T.splitBAR(a, N)
```

(See: splitBAR.py) (See: splitBARPT.py)

T.splitTBranches: split a 'BAR' array into several BAR arrays, at vertices where T-branches exist:

```
B = T.splitTBranches(a, tol=1.e-13) .or.  B = T.splitTBranches(A, tol=1.e-13)
```

(See: splitTBranches.py) (See: splitTBranchesPT.py)

T.splitTRI: split an unstructured TRI array into several TRI arrays delimited by the input poly lines which are a lists of index list. 2 consecutives indices in a list must define an existing edge in the mesh:

```
B = T.splitTRI(a, polyLines)
```
(See: splitTRI.py)

T.splitManifold: split an unstructured mesh (only TRI or BAR currently) into several manifold pieces:
```
B = T.splitManifold(a)
```
(See: splitManifold.py)

T.splitSize: split structured blocks when their number of points is greater than N. Argument 'multigrid' ensures the given multigrid level when splitting, if input grids respect this multigrid level:
```
B = T.splitSize(a, N, multigrid=0, dirs=[1,2,3]) .or. B = T.splitSize(A, N, multigrid=0, dirs=[1,2,3])
```
SplitSize can also be used to split blocks in order to best fit on a number of R=12 processors. MinPtsPerDir specifies the minimum number of points per direction the splitter must respect:
```
B = T.splitSize(a, R=12, multigrid=0, dirs=[1,2,3], type=2, minPtsPerDir=5) .or. B = T.splitSize(A, R=12, multigrid=0, dirs=[1,2,3], minPtsPerDir=5)
```
(See: splitSize.py) (See: splitSizePT.py)

T.splitNParts: split structured blocks in N parts. Argument 'multigrid' ensures the given multigrid level when splitting, if input grids respect this multigrid level. Parameter dirs enable to enforce splitting in one direction, i.e. dirs=[1] force splitting only in the first mesh direction.
Boundary conditions are split, but grid connectivity is removed:
```
B = T.splitNParts(a, N, multigrid=0, dirs=[1,2,3]) .or. B = T.splitNParts(A, N, multigrid=0, dirs=[1,2,3])
```
(See: splitNParts.py) (See: splitNPartsPT.py)

T.splitMultiplePts: split structured blocks at border points where they are connected to an even number of blocks. Parameter dim is the dimension of the problem:
```
B = T.splitMultiplePts(A, dim=3)
```
(See: splitMultiplePts.py) (See: splitMultiplePtsPT.py)

T.breakElements: break a NGON zone into a list of unstructured zones, as BAR, TRI, QUAD, TETRA, PENTA, PYRA, HEXA or NGON zones:
```
B = T.breakElements(a) .or. B = T.breakElements(A)
```
(See: breakElements.py) (See: breakElementsPT.py)

T.dual: returns the dual of a mesh. If extraPoints=1, points in the center of external faces are added:
```
b = T.dual(a, extraPoints=1)
```
(See: dual.py) (See: dualPT.py)

T.deformPoint: deform a, by moving point (x,y,z) of vector (dx,dy,dz). Fourth argument controls the depth of deformation. Last argument controls the width of deformation:

```
b = T.deformPoint(a, (x,y,z), (dx,dy,dz), 0.5, 0.4)
```
(See: deformPoint.py) (See: deformPointPT.py)

T.deform: deform a, by moving each point of a given vector array (array):

```
b = T.deform(a, vector)
```
Deform a, by moving each point of a given vector defined by variables 'dx','dy','dz', defined in a (pyTree):

```
b = T.deform(a, ['dx','dy','dz'])
```
(See: deform.py) (See: deformPT.py)

T.deformNormals: deform a surface a, by moving each point of the surface by a scalar field alpha times the surface normals in niter steps (array)¡/a¿:

```
b = T.deformNormals(a, alpha., niter=1)
```

Deform a surface a, by moving each point of the surface by a variable named alpha times the surface normals in niter steps (pyTree)¡/a¿:

```
b = T.deformNormals(a, alpha, niter=1)
```
(See: deformNormals.py) (See: deformNormalsPT.py)

T.deformMesh: deform a mesh defined by a, given surface or a set of surfaces for which a deformation is defined at nodes as 'dx,dy,dz'. Beta defined the deformation zone as multiplication factor of local deformation:

```
b = T.deformMesh(a, surfaces, beta=4.)
```
(See: deformMesh.py) (See: deformMeshPT.py)

T.projectAllDirs: project a (A) onto a list of surfaces S following a vector defined for each point of a (A). The three components of the vector are defined by the list of strings vect. If oriented=0, use both direction for projection else use vector direction only:

```
b = T.projectAllDir(a, S, vect=['nx','ny','nz'], oriented=0) .or. B = T.projectAllDirs(A, S, vect=['nx','ny','nz'], oriented=0)
```
(See: projectAllDirs.py) (See: projectAllDirsPT.py)

T.projectDir: project a (A) onto a list of surfaces S following a given direction. If smooth=1, unprojected points are smoothed (available only for structured arrays). If oriented=0, use both direction for projection else use vector direction only:

```
b = T.projectDir(a, S, (1.,0,0), smooth=0, oriented=0) .or. B = T.projectDir(A, S, (1.,0,0), smooth=0, oriented=0)
```
(See: projectDir.py) (See: projectDirPT.py)

T.projectOrtho: project a (A) onto a list of surfaces S following normals:

```
b = T.projectOrtho(a, S) .or. B = T.projectOrtho(A, S)
```
(See: projectOrtho.py) (See: projectOrthoPT.py)

T.projectOrthoSmooth: project a (A) onto a list of surfaces S following smoothed normals. niter is the number of smoothing iterations:

b = T.projectOrthoSmooth(a, S, niter=1) *.or.* B = T.projectOrthoSmooth(A, S, niter=1)

(See: projectOrthoSmooth.py) (See: projectOrthoSmoothPT.py)

T.projectRay: project a (A) onto a list of surfaces S following rays issued from P:

b = T.projectRay(a, S, P) *.or.* B = T.projectRay(A, S, P)

(See: projectRay.py) (See: projectRayPT.py)

1.5 Example files

Example file: subzone.py

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

# Structure
a = G.cart((0,0,0), (1,1,1), (10,20,10))
a = T.subzone(a, (3,3,3), (7,8,5))

# Structure avec indices negatif
e = G.cart((0,0,0), (1,1,1), (10,20,10))
e = T.subzone(e, (1,1,1), (-1,-1,-2)) # imax,jmax,kmax-1

# Non structure. Indices de noeuds -> retourne elts
b = G.cartTetra((0,0,0), (1,1,1), (2,2,2))
b = T.subzone(b, [2,6,8,5])

# Non structure. Indices d'elements -> Retourne elts
# Les indices d'elements commencent a 0
c = G.cartTetra((0,0,0), (1,1,1), (5,5,5))
c = T.subzone(c, [0,1], type='elements')

# Non structure. Indices de faces:
# Pour les maillages BAR, TRI, TETRA... indFace=indElt*nbreFaces+noFace
# les noFace commence a 1
# Pour les NGONS... indices des faces
# -> Retourne les faces
d = G.cartTetra((0,0,0), (1,1,1), (2,2,2))
d = T.subzone(d, [1,2,3], type='faces')

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

Example file: subzonePT.py

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

a = G.cart((0,0,0), (1,1,1), (10,20,1))
a = T.subzone(a, (3,3,1), (7,8,1))
C.convertPyTree2File(a, 'out.cgns')
```

Example file: join.py

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

a1 = G.cartTetra((0.,0.,0.), (1.,1.,1), (11,11,1))
a2 = G.cartTetra((10.,0.,0.), (1.,1.,1), (10,10,1))
a = T.join(a1, a2)
C.convertArrays2File([a], 'out.plt')
```

Example file: joinPT.py

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

a1 = D.naca(12., 5001)
a2 = D.line((1.,0.,0.),(20.,0.,0.),5001)
a = T.join(a1, a2)
C.convertPyTree2File(a, "out.cgns")
```

Example file: merge.py

```
# - merge (array) -
import Converter as C
import Generator as G
import Transform as T
import Geom as D
def f(t,u):
    x = t+u; y = t*t+1+u*u; z = u
    return (x,y,z)

a = D.surface(f)
b = T.splitSize(a, 100)
b = T.merge(b)
C.convertArrays2File(b, "out.plt")
```

Example file: mergePT.py

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

def f(t,u):
    x = t+u
    y = t*t+1+u*u
    z = u
    return (x,y,z)

a = D.surface(f)
b = T.splitSize(a, 100)
b = X.connectMatch(b, dim=2)
t = C.newPyTree(['Surface']); t[2][1][2] += b
b = T.merge(t)
t[2][1][2] = b
C.convertPyTree2File(t, "out.cgns")
```

Example file: mergeCart.py

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

dh = 0.1; n = 11
A = []
a1 = G.cart((0.,0.,0.),(dh,dh,dh),(n,n,n)); A.append(a1)
a2 = G.cart((1.,0.,0.),(dh,dh,dh),(n,n,n)); A.append(a2)
a3 = G.cart((1.,1.,0.),(dh,dh,dh),(n,n,n)); A.append(a3)
a4 = G.cart((0.,1.,0.),(dh,dh,dh),(n,n,n)); A.append(a4)
A[0] = T.oneovern(A[0],(2,2,2))
A[1] = T.oneovern(A[1],(2,2,2))
res = T.mergeCart(A)
C.convertArrays2File(res, "out.plt")
```

Example file: mergeCartPT.py

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

dh = 0.1; n = 11
A = []
a1 = G.cart((0.,0.,0.),(dh,dh,dh),(n,n,n)); a1 = T.oneovern(a1,(2,2,2))
a2 = G.cart((1.,0.,0.),(dh,dh,dh),(n,n,n)); a2 = T.oneovern(a2,(2,2,2))
a3 = G.cart((1.,1.,0.),(dh,dh,dh),(n,n,n))
a4 = G.cart((0.,1.,0.),(dh,dh,dh),(n,n,n))
A = [a1,a2,a3,a4]
for i in range(1,5): A[i-1][0] = 'cart'+str(i)

t = C.newPyTree(['Base']); t[2][1][2] += A
t[2][1][2] = T.mergeCart(t[2][1][2])
C.convertPyTree2File(t, "out.cgns")
```

Example file: patch.py

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

c1 = G.cart((0,0,0), (0.01,0.01,1), (201,101,1))
c2 = G.cart((0,0,0), (0.01,0.01,1), (51,81,1))
c2 = T.rotate(c2, (0,0,0),(0,0,1),0.2)
c3 = G.cart((0.0,1.,0), (0.01,0.01,1), (101,1,1))
c3 = T.rotate(c3, (0,0,0),(0,0,1),0.3)
# patch a region
a = T.patch(c1, c2,(1,1,1))
# patch some nodes
nodes = numpy.arange(20100, 20201, dtype=numpy.int32)
b = T.patch(c1, c3, nodes=nodes)
C.convertArrays2File([a,b], 'out.plt')
```

Example file: patchPT.py

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

c1 = G.cart((0,0,0), (0.01,0.01,1), (201,101,1))
c2 = G.cart((0,0,0), (0.01,0.01,1), (51,81,1))
a = T.patch(c1, c2,(1,1,1))
C.convertPyTree2File(a, 'out.cgns')
```

Example file: oneovern.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,1))
a2 = T.oneovern(a, (2,2,2))
C.convertArrays2File([a2], "out.plt")
```

Example file: oneovernPT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,1))
a2 = T.oneovern(a, (2,2,1)); a2[0] = 'cart2'
C.convertPyTree2File([a,a2], "out.cgns")
```

Example file: collapse.py

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

hi = 0.1; hj = 0.01; hk = 1.
ni = 20; nj = 2; nk = 1
a = G.cartTetra((0.,0.,0.),(hi,hj,hk),(ni,nj,nk))
b = T.collapse(a)
C.convertArrays2File([a,b], "out.plt")
```

Example file: collapsePT.py

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

hi = 0.1; hj = 0.01; hk = 1.
ni = 20; nj = 2; nk = 1
a = G.cartTetra((0.,0.,0.),(hi,hj,hk),(ni,nj,nk))
b = T.collapse(a)
C.convertPyTree2File(b, "out.cgns")
```

Example file: reorder.py

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

# Structured
a = G.cart((0,0,0),(1,1,1),(5,7,9))
a = T.reorder(a, (3,-2,-1))
C.convertArrays2File([a], 'out1.plt')

# Unstructured
a = G.cartTetra((0,0,0),(1,1,1),(3,3,1))
c = a[2]; c[1,0] = 5; c[2,0] = 2
a = T.reorder(a, (1,))
C.convertArrays2File([a], 'out2.plt')
```

Example file: reorderPT.py

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

a = G.cart((0,0,0), (1,1,1), (8,9,20))
a = T.reorder(a, (2,1,-3))
C.convertPyTree2File(a, "out.cgns")
```

Example file: reorderAll.py

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

ni = 30; nj = 40
m1 = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,1))
m2 = T.rotate(m1, (0.2,0.2,0.), (0.,0.,1.), 15.)
m2 = T.reorder(m2,(-1,2,3))
a = [m1,m2]
a = T.reorderAll(a,1)
C.convertArrays2File(a, "out.plt")
```

Example file: reorderAllPT.py

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

ni = 30; nj = 40; nk = 1
m1 = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk)); m1[0]='cart1'
m2 = T.rotate(m1, (0.2,0.2,0.), (0.,0.,1.), 15.)
m2 = T.reorder(m2,(-1,2,3)); m2[0]='cart2'
t = C.newPyTree(['Base',2,[m1,m2]])
t = T.reorderAll(t, 1)
C.convertPyTree2File(t, "out.cgns")
```

Example file: makeDirect.py

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

a = G.cart((0.,0.,0.),(1.,1.,1.),(10,10,10))
a = T.reorder(a, (1,2,-3)) # indirect now
a = T.makeDirect(a)
C.convertArrays2File([a], 'out.plt')
```

Example file: makeDirectPT.py

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

a = G.cart((0.,0.,0.),(1.,1.,1.),(10,10,10))
a = T.reorder(a, (1,2,-3)) # indirect now
a = T.makeDirect(a)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: makeCartesianXYZ.py

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

a = G.cart((0.,0.,0.),(1.,1.,1.),(11,12,13))
a = T.reorder(a, (3,2,-1))
a = T.makeCartesianXYZ(a)
C.convertArrays2File([a],'out.plt')
```

Example file: makeCartesianXYZPT.py

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

a = G.cart((0.,0.,0.),(1.,1.,1.),(11,12,13))
a = T.reorder(a, (3,2,-1))
a = T.makeCartesianXYZ(a)
C.convertPyTree2File(a,'out.cgns')
```

Example file: addkplane.py

```
# - addkplane (array) -
import Geom as D
import Transform as T
import Converter as C

a = D.naca(12., 501)
a = T.addkplane(a)
C.convertArrays2File([a], "out.plt")
```

Example file: addkplanePT.py

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

a = G.cart((0.,0.,0.),(0.1,0.1,1.),(10,10,2))
a = T.addkplane(a)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: translate.py

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

a = G.cart( (0,0,0), (1,1,1), (10,10,1))
b = T.translate(a, (-1.,0.,0.))
C.convertArrays2File([a,b], 'out.plt')
```

Example file: translatePT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,3))
T._translate(a, (10.,0.,0.))
C.convertPyTree2File(a, 'out.cgns')
```

Example file: rotate.py

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

a = G.cart( (0,0,0), (1,1,1), (10,10,1))
# Rotate with an axis and an angle
b = T.rotate(a, (0.,0.,0.), (0.,0.,1.), 30.)
# Rotate with axis transformations
c = T.rotate(a, (0.,0.,0.), ((1.,0.,0.),(0,1,0),(0,0,1)),
              ((1,1,0), (1,-1,0), (0,0,1)) )
# Rotate with three angles
d = T.rotate(a, (0.,0.,0.), (90.,0.,0.))
C.convertArrays2File([a,d], 'out.plt')
```

Example file: rotatePT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,2))
# Rotate with an axis and an angle
b = T.rotate(a, (0.,0.,0.), (0.,0.,1.), 30.); b[0] = 'cartRot1'
# Rotate with two axis
c = T.rotate(a, (0.,0.,0.), ((1.,0.,0.),(0,1,0),(0,0,1)),
             ((1,1,0), (1,-1,0), (0,0,1)) ); c[0] = 'cartRot2'
# Rotate with three angles
c = T.rotate(a, (0.,0.,0.), (0,0,90)); c[0] = 'cartRot3'
C.convertPyTree2File([a,b,c], 'out.cgns')
```

Example file: cart2Cyl.py

```
# - cart2Cyl (array) -
import Transform as T
import Generator as G
import Converter as C
a = G.cylinder((0.,0.,0.), 0.5, 1., 0., 359, 1., (360,20,10))
a = T.cart2Cyl(a, (0.,0.,0.),(0,0,1))
C.convertArrays2File([a], 'out.plt')
```

Example file: cart2CylPT.py

```
# - cart2Cyl (pyTree) -
import Transform.PyTree as T
import Generator.PyTree as G
import Converter.PyTree as C
a = G.cylinder((0.,0.,0.), 0.5, 1., 0., 360., 1., (360,20,10))
T._cart2Cyl(a, (0.,0.,0.),(0,0,1))
C.convertPyTree2File(a, 'out.cgns')
```

Example file: homothety.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,1))
b = T.homothety(a, (0.,0.,0.), 2.)
C.convertArrays2File([a,b], 'out.plt')
```

Example file: homothetyPT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,10))
b = T.homothety(a, (0.,0.,0.), 2.); b[0] = 'cart2'
C.convertPyTree2File([a,b], "out.cgns")
```

Example file: contract.py

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

a = G.cart( (0,0,0), (1,1,1), (10,10,10))
b = T.contract(a, (0.,0.,0.), (1,0,0), (0,1,0), 0.1)
C.convertArrays2File([a,b], 'out.plt')
```

Example file: contractPT.py

```
# - contract (pytree) -
import Generator.PyTree as G
import Transform.PyTree as T
import Converter.PyTree as C

a = G.cart((0,0,0), (1,1,1), (10,10,10))
b = T.contract(a, (0.,0.,0.), (1,0,0), (0,1,0), 0.1); b[0]='cart2'
C.convertPyTree2File([a,b], 'out.cgns')
```

Example file: scale.py

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

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

# scale in all directions
a = T.scale(a, factor=0.1)

# scale with different factors following directions
a = T.scale(a, factor=(0.1,0.2,0.3))

C.convertArrays2File(a, 'out.plt')
```

Example file: scalePT.py

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

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

# scale in all directions
T._scale(a, factor=0.1)

# scale with different factors following directions
T._scale(a, factor=(0.1,0.2,0.3))

C.convertPyTree2File(a, 'out.cgns')
```

Example file: symetrize.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,1))
b = T.symetrize(a, (0.,0.,0.), (1,0,0), (0,0,1))
C.convertArrays2File([a,b], "out.plt")
```

Example file: symetrizePT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,2))
b = T.symetrize(a, (0.,0.,0.), (1,0,0), (0,0,1)); b[0]='cart2'
C.convertPyTree2File([a,b], "out.cgns")
```

Example file: perturbate.py

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

```
a = G.cart((0,0,0), (1,1,1), (10,10,1))
a = T.perturbate(a, 0.1)
C.convertArrays2File([a], "out.plt")
```

Example file: perturbatePT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,2))
b = T.perturbate(a, 0.1); b[0]='cart2'
C.convertPyTree2File([a,b], "out.cgns")
```

Example file: smooth.py

```
# - smooth (array) -
import Transform as T
import Converter as C
import Geom as D
a = D.sphere6((0,0,0), 1, N=20)
b = T.smooth(a, eps=0.5, niter=20)
C.convertArrays2File(a+b, "out.plt")
```

Example file: smoothPT.py

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

a = D.sphere6((0,0,0), 1, N=20)
b = T.smooth(a, eps=0.5, niter=20)
C.convertPyTree2File(b, "out.cgns")
```

Example file: splitCurvatureAngle.py

```
# - splitCurvatureAngle (array) -
import Converter as C
import Transform as T
import Geom as D

line = D.line((0.,0.,0.), (1.,1.,0.), 51)
line2 = D.line((-1.,1.,0.), (0.,0.,0.), 51)
a = T.join(line2, line)
list = T.splitCurvatureAngle(a, 30.)
C.convertArrays2File(list, 'out.plt')
```

Example file: splitCurvatureAnglePT.py

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

a = D.naca(12,101)
a2 = D.line((1,0,0), (2,0,0), 50)
a = T.join(a, a2)
a2 = D.line((2,0,0), (1,0,0), 50)
a = T.join(a, a2)
zones = T.splitCurvatureAngle(a, 20.)
C.convertPyTree2File(zones+[a], 'out.cgns')
```

Example file: splitCurvatureRadius.py

```
# - splitCurvatureRadius (array) -
import Converter as C
import Transform as T
import Geom as D

pts = C.array('x,y,z', 7, 1, 1)
x = pts[1][0]; y = pts[1][1]; z = pts[1][2]
x[0]= 6.; x[1] = 5.4; x[2]=4.8; x[3] = 2.5; x[4]  = 0.3
y[0]=10.; y[1]=0.036; y[2]=-5.;y[3]=0.21;y[4]=0.26;y[5]=7.
z[0]=1.; z[1]=1.; z[2]=1.;z[3]=1.;z[4]=1.;z[5]=1.; z[6]=1.

a = D.bezier( pts, 50 )
L = T.splitCurvatureRadius(a)
C.convertArrays2File([a]+L, 'out.plt')
```

Example file: splitCurvatureRadiusPT.py

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

a = D.naca(12.5000)
zones = T.splitCurvatureRadius(a, 10.)
C.convertPyTree2File(zones+[a], 'out.cgns')
```

Example file: splitConnexity.py

```
# - splitConnexity (array) -
import Converter as C
import Transform as T
import Geom as D

a = D.text2D("CASSIOPEE")
B = T.splitConnexity(a)
C.convertArrays2File(B, 'out.plt')
```

Example file: splitConnexityPT.py

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

a = D.text2D("CASSIOPEE")
B = T.splitConnexity(a)
C.convertPyTree2File(B, 'out.cgns')
```

Example file: splitSharpEdges.py

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

a = D.text3D("A"); a = G.close(a, 1.e-4)
B = T.splitSharpEdges(a, 89.)
C.convertArrays2File(B, 'out.plt')
```

Example file: splitSharpEdgesPT.py

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

a = D.text3D("A"); a = G.close(a, 1.e-3)
B = T.splitSharpEdges(a, 89.)
C.convertPyTree2File(B, 'out.cgns')
```

Example file: splitBAR.py

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

a = G.cart( (0,0,0), (1,1,1), (50,1,1) )
a = C.convertArray2Tetra(a)
a = G.close(a)
b = T.splitBAR(a, 5)
C.convertArrays2File(b, 'out.plt')
```

Example file: splitBARPT.py

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

a = G.cart((0,0,0), (1,1,1), (50,1,1))
a = C.convertArray2Tetra(a)
a = G.close(a)
B = T.splitBAR(a, 5)
C.convertPyTree2File(B, 'out.cgns')
```

Example file: splitTBranches.py

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

a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,1,50))
c1 = T.subzone(a,(1,1,1),(50,1,1))
c2 = T.subzone(a,(1,1,50),(50,1,50))
c3 = T.subzone(a,(1,1,1),(1,1,50))
c = [c1,c2,c3]; c = C.convertArray2Hexa(c)
c = T.join(c)
res = T.splitTBranches(c)
C.convertArrays2File(res,"out.plt")
```

Example file: splitTBranchesPT.py

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

a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,1,50))
c1 = T.subzone(a,(1,1,1),(50,1,1))
c2 = T.subzone(a,(1,50,1),(50,50,1))
c3 = T.subzone(a,(1,1,1),(1,50,1))
c = [c1,c2,c3]; c = C.convertArray2Hexa(c)
c = T.join(c)
res = T.splitTBranches(c)
C.convertPyTree2File(res, "out.cgns")
```

Example file: splitTRI.py

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

a = D.circle( (0,0,0), 1, N=20 )
a = C.convertArray2Tetra(a)
a = G.close(a)
b = G.T3mesher2D(a)
#C.convertArrays2File([b], 'out.plt')
c = [[9, 25, 27, 30, 29, 28, 34, 38, 0], [29, 23, 19, 20, 24, 29]]
d = T.splitTRI(b, c)
C.convertArrays2File([d[0]], 'out1.plt')
C.convertArrays2File([d[1]], 'out2.plt')
```

Example file: splitManifold.py

```
# - splitManifold (array) -
# Conforming 1 or 2 TRI/BAR together (same type for both operands
import Converter as C
import Generator as G
import Intersector as XOR
import Geom as D
from Geom.Parametrics import base
import Transform as T

s1 = D.sphere( (0,0,0), 1, N=20 )

s2 = D.surface(base['plane'], N=30)
s2 = T.translate(s2, (0.2,0.2,0.2))

s1 = C.convertArray2Tetra(s1); s1 = G.close(s1)
s2 = C.convertArray2Tetra(s2); s2 = G.close(s2)

x = XOR.conformUnstr(s1, s2, 0., 2)
x = T.splitManifold(x)

C.convertArrays2File(x, 'outS.plt')

a = G.cylinder((0.,0.,0.), 0.5, 1., 360., 0., 10., (50,1,50))
c1 = T.subzone(a,(1,1,1),(50,1,1))
c2 = T.subzone(a,(1,1,50),(50,1,50))
c3 = T.subzone(a,(1,1,1),(1,1,50))
```

```
c = [c1,c2,c3]; c = C.convertArray2Hexa(c)
c = T.join(c)
C.convertArrays2File([c], 'B.plt')
x = T.splitManifold(c)

C.convertArrays2File(x, 'outB.plt')
```

Example file: splitSize.py

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

a = G.cart((0,0,0),(1,1,1),(50,20,10))
B = T.splitSize(a, 2000, type=0)
C.convertArrays2File(B, 'out.plt')
```

Example file: splitSizePT.py

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

a = G.cart((0,0,0),(1,1,1),(50,20,10))
t = C.newPyTree(['Base',a])
t = T.splitSize(t, 300, type=0)
C.convertPyTree2File(t, 'out.cgns')
```

Example file: splitNParts.py

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

a = G.cart((0,0,0), (1,1,1), (101,101,41))
b = G.cart((10,0,0), (1,1,1), (121,61,81))
c = G.cart((20,0,0), (1,1,1), (101,61,131))
res = T.splitNParts([a,b,c], 32, multigrid=0, dirs=[1,2,3])

C.convertArrays2File(res, 'out.plt')
```

Example file: splitNPartsPT.py

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

a = G.cart((0,0,0), (1,1,1), (81,81,81))
b = G.cart((80,0,0), (1,1,1), (41,81,41))
t = C.newPyTree(['Base',a,b])
t = T.splitNParts(t, 10, multigrid=0, dirs=[1,2,3])
C.convertPyTree2File(t, 'out.cgns')
```

Example file: splitMultiplePts.py

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

z0 = G.cart((0.,0.,0.),(0.1,0.1,1.),(10,10,1))
z1 = T.subzone(z0,(1,1,1),(5,10,1))
z2 = T.subzone(z0,(5,1,1),(10,5,1))
z3 = T.subzone(z0,(5,5,1),(10,10,1))
zones = [z1,z2,z3]
zones = T.splitMultiplePts(zones,dim=2)
C.convertArrays2File(zones, 'out.plt')
```

Example file: splitMultiplePtsPT.py

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

nk = 2
z0 = G.cart((0.,0.,0.),(0.1,0.1,1.),(10,10,nk))
z1 = T.subzone(z0,(1,1,1),(5,10,nk)); z1[0] = 'cart1'
z2 = T.subzone(z0,(5,1,1),(10,5,nk)); z2[0] = 'cart2'
z3 = T.subzone(z0,(5,5,1),(10,10,nk)); z3[0] = 'cart3'
z0 = T.translate(z0,(-0.9,0.,0.)); z0[0] = 'cart0'
z4 = G.cart((-0.9,0.9,0.),(0.1,0.1,1.),(19,5,nk)); z4[0] = 'cart4'
t = C.newPyTree(['Base',z1,z2,z3,z4])
t = X.connectMatch(t,dim=2)
t = C.fillEmptyBCWith(t, 'wall', 'BCWall', dim=2)
t = T.splitMultiplePts(t, dim=2)
C.convertPyTree2File(t, 'out.cgns')
```

Example file: breakElements.py

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

a = G.cartTetra((0,0,0),(1,1,1),(3,3,2))
a = C.convertArray2NGon(a)
a = G.close(a)
b = G.cartNGon((2,0,0),(1,1,1),(3,2,2))
res = T.join(a,b)
res = T.breakElements(res)
C.convertArrays2File(res, 'out.plt')
```

Example file: breakElementsPT.py

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

a = G.cartTetra((0,0,0),(1,1,1),(3,3,2))
a = C.convertArray2NGon(a)
a = G.close(a)
b = G.cartNGon((2,0,0),(1,1,1),(3,3,1))
res = T.join(a,b)
res = T.breakElements(res)
C.convertPyTree2File(res, 'out.cgns')
```

Example file: dual.py

```
import Converter as C
import Generator as G
import Transform as T
import Geom as D

ni = 5; nj = 5; nk = 1
a = G.cart((0,0,0),(1,1,1),(ni,nj,nk))
a = C.convertArray2NGon(a); a = G.close(a)
res = T.dual(a)
C.convertArrays2File([res],'out.tp')
```

Example file: dualPT.py

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

ni = 5; nj = 5
a = G.cart((0,0,0),(1,1,1),(ni,nj,1))
a = C.convertArray2NGon(a); a = G.close(a)
res = T.dual(a)
C.convertPyTree2File(res, 'out.cgns')
```

Example file: deformPoint.py

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

a1 = G.cart((0,0,0), (1,1,1), (10,10,1))
a2 = T.deformPoint(a1, (0,0,0), (0.1,0.1,0.1), 0.5, 2.)
C.convertArrays2File([a2], "out.plt")
```

Example file: deformPointPT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,1))
a = T.deformPoint(a, (0,0,0), (0.1,0.1,1.), 0.5, 0.4)
C.convertPyTree2File(a, "out.cgns")
```

Example file: deform.py

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

a = D.sphere((0,0,0), 1., 50)
```

```
n = G.getNormalMap(a)
n = C.center2Node(n); n[1] = n[1]*10
a = C.addVars([a,n])
b = T.deform(a,['sx','sy','sz'])
C.convertArrays2File([b], 'out.plt')
```

Example file: deformPT.py

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

a = G.cart((0.,0.,0.),(1.,1.,1.),(10,10,10))
C._initVars(a,'dx=10.')
C._initVars(a,'dy=0')
C._initVars(a,'dz=0')
b = T.deform(a)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: deformNormals.py

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

a = D.sphere((0,0,0), 1., 50)
a = C.convertArray2Hexa(a)
a = G.close(a)
b = C.initVars(a, 'alpha=0.5*{x}')
b = C.extractVars(b, ['alpha'])
b = T.deformNormals(a, b, niter=2)
C.convertArrays2File([b], 'out.plt')
```

Example file: deformNormalsPT.py

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

a = D.sphere6((0,0,0), 1., 10)
a = C.convertArray2Hexa(a)
a = T.join(a); a = G.close(a)

a = C.initVars(a, '{alpha}=0.5*{CoordinateX}')
a = T.deformNormals(a, 'alpha', niter=2)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: deformMesh.py

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

a1 = D.sphere6((0,0,0), 1, 20)
```

```
a1 = C.convertArray2Tetra(a1); a1 = T.join(a1)
point = C.getValue(a1, 0)
a2 = T.deformPoint(a1, point, (0.1,0.05,0.2), 0.5, 2.)
delta = C.addVars(a1, ['dx','dy','dz'])
delta = C.extractVars(delta, ['dx','dy','dz'])
delta[1][:,:] = a2[1][:,:]-a1[1][:,:]
a1 = C.addVars([a1, delta])
m = D.sphere6((0,0,0), 2, 20)
m = T.deformMesh(m, a1)
C.convertArrays2File(m, "out.plt")
```

Example file: deformMeshPT.py

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

a1 = D.sphere6((0,0,0),1,20)
a1 = C.convertArray2Tetra(a1); a1 = T.join(a1)
point = C.getValue(a1, 'GridCoordinates', 0)
a2 = T.deformPoint(a1, point, (0.1,0.05,0.2), 0.5, 2.)
delta = C.diffArrays(a2,a1)
deltax = C.getField('DCoordinateX',delta)
deltay = C.getField('DCoordinateY',delta)
deltaz = C.getField('DCoordinateZ',delta)
for noz in xrange(len(deltax)):
    deltax[noz][0] = 'dx'
    deltay[noz][0] = 'dy'
    deltaz[noz][0] = 'dz'
a1 = C.setFields(deltax,a1,'nodes')
a1 = C.setFields(deltay,a1,'nodes')
a1 = C.setFields(deltaz,a1,'nodes')

m = D.sphere6((0,0,0),2,20)
m = T.deformMesh(m, a1)
C.convertPyTree2File(m, "out.cgns")
```

Example file: projectAllDirs.py

```
# - projectAllDirs (array) -
import Geom as D
import Converter as C
import Generator as G
import Transform as T
a = D.sphere6((0,0,0), 1., 20)
b = G.cart((1.1,-0.1,-0.1),(0.03,0.03,0.03), (1,50,50))
n = G.getNormalMap(b)
n = C.center2Node(n)
b = C.addVars([b,n])
c = T.projectAllDirs([b], a, ['sx','sy','sz'])
C.convertArrays2File([b]+c, 'out.plt')
```

Example file: projectAllDirsPT.py

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

```
a = D.sphere((0,0,0), 1., 20)
b = G.cart((1.1,-0.1,-0.1),(0.1,0.1,0.1), (1,5,5))
b = G.getNormalMap(b)
b = C.center2Node(b,['centers:sx','centers:sy','centers:sz'])
c = T.projectAllDirs(b, a,['sx','sy','sz']); c[0] = 'projection'
C.convertPyTree2File([a,b,c], 'out.cgns')
```

Example file: projectDir.py

```
# - projectDir (array) -
import Geom as D
import Converter as C
import Generator as G
import Transform as T
a = D.sphere((0,0,0), 1., 20)
b = G.cart((1.1,-0.1,-0.1),(0.03,0.03,0.03), (1,50,50))
c = T.projectDir(b, [a], (1.,0,0))
d = T.projectDir([b], [a], (1.,0,0), smooth=1)
C.convertArrays2File([a,b,c]+d, 'out.plt')
```

Example file: projectDirPT.py

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

a = D.sphere((0,0,0), 1., 20)
b = G.cart((1.1,-0.1,-0.1),(0.1,0.1,0.1), (1,5,5))
c = T.projectDir(b, a, (1.,0,0)); c[0] = 'projection'
C.convertPyTree2File([a,b,c], 'out.cgns')
```

Example file: projectOrtho.py

```
# - projectOrtho (array) -
import Geom as D
import Converter as C
import Generator as G
import Transform as T
a = D.sphere((0,0,0), 1., 300)
b = G.cart((-0.5,-0.5,-1.5),(0.05,0.05,0.1), (20,20,1))
c = T.projectOrtho(b, [a])
C.convertArrays2File([a,b,c], 'out.plt')
```

Example file: projectOrthoPT.py

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

a = D.sphere((0,0,0), 1., 20)
b = G.cart((1.1,-0.1,-0.1),(0.1,0.1,0.1), (1,5,5))
c = T.projectOrtho(b, a); c[0] = 'projection'
C.convertPyTree2File([a,b,c], 'out.cgns')
```

Example file: projectOrthoSmooth.py

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

a = D.sphere((0,0,0), 1., 30)
b = G.cart((-0.5,-0.5,-1.5),(0.05,0.05,0.1), (20,20,1))
c = T.projectOrthoSmooth(b, [a], niter=2)
C.convertArrays2File([a,b,c], 'out.plt')
```

Example file: projectOrthoSmoothPT.py

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

a = D.sphere((0,0,0), 1., 30)
b = G.cart((-0.5,-0.5,-1.5),(0.05,0.05,0.1), (20,20,1))
c = T.projectOrthoSmooth(b, [a], niter=2)
C.convertPyTree2File([a,b,c], 'out.cgns')
```

Example file: projectRay.py

```
# - projectRay (array) -
import Geom as D
import Converter as C
import Generator as G
import Transform as T
a = D.sphere((0,0,0), 1., 20)
b = G.cart((1.1,-0.1,-0.1),(0.1,0.1,0.1), (1,5,5))
c = T.projectRay(b, [a], (0,0,0))
C.convertArrays2File([a,b,c], 'out.plt')
```

Example file: projectRayPT.py

```
# - projectRay (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C
import Generator.PyTree as G
import Transform.PyTree as T
a = D.sphere((0,0,0), 1., 20)
b = G.cart((1.1,-0.1,-0.1),(0.1,0.1,0.1), (1,5,5))
c = T.projectRay(b, a, (0,0,0)); c[0] = 'projection'
C.convertPyTree2File([a,b,c], 'out.cgns')
```