Geom 2.4

Stephanie Peron, Christophe Benoit, Pascal Raud, Sam Landier
- Onera -

1 Geom: Geometry definition module

1.1 Preamble

In this module, a geometry is defined discretely with a great number of points. A geometry can be a i-array or a BAR-array for 2D simulations and a i,j-array, a QUAD-array or a TRI-array for 3D simulations. A **polyline** is defined as a C0 i-array which contains only the polyline points (with no extra discretization points). Geometry is stored in a Converter array (as defined in Converter documentation) or in a zone of a CGNS/python tree (pyTree), following the selected interface.

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

To use it with the Converter array interface, you must import the Geom module:

```
import Geom as D
```

Then, in the following, a is an array, and A a list of arrays.
To use it with the pyTree interface, you must import the module:

```
import Geom.PyTree as D
```

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

1.2 Analytical geometries

D.point: create a point of coordinates x,y,z:

```
a = D.point( (x,y,z) )
```

(See: point.py) (See: pointPT.py)

D.naca: create a naca i-array with less than N points and of thickness e (e=12, 15...) in the (x,y) plane:

```
a = D.naca(e, N=101)
```

(See: naca.py) (See: nacaPT.py)

D.line: create a line i-array between point (x1,y1,z1) and point (x2,y2,z2), discretized with N points:

```
a = D.line((x1,y1,z1), (x2,y2,z2), N=100)
```

(See: line.py) (See: linePT.py)

D.polyline: create a polyline i-array from a list of points [(x1,y1,z1),(x2,y2,z2),...,(xN,yN,zN)], containing no extra discretization points:

```
a = D.polyline([(x1,y1,z1),(x2,y2,z2),...,(xN,yN,zN)])
```

(See: polyline.py) (See: polylinePT.py)

D.circle: create an i-array made of a portion of circle with N points of center C, radius R, between angles tetas and tetae (in degrees) in the (x,y) plane:

```
a = D.circle((xc,yc,zc), R, tetas=0, tetae=360, N=100)
```

(See: circle.py) (See: circlePT.py)

D.bezier: create a Bezier i-array or i,j-array using control points. Either the number of points or the point density can be specified. Control points coordinates are defined in a i-array or a i,j-array designated here by c:

```
a = D.bezier(c, N=100, M=100) .or. a = D.bezier(c, density=10.)
```

(See: bezier.py) (See: bezierPT.py)

D.spline: create a spline i-array or i,j-array of using control points. Control points coordinates are defined in a i-array or i,j-array. Order is the order of spline:

```
a = D.spline(c, order=3, N=100, M=100) .or. a = D.spline(c, order=3, density=10.)
```

(See: spline.py) (See: splinePT.py)

D.nurbs: create a NURBS i-array or i,j-array using control points and associated weights. Control points coordinates and weight are defined in a i-array or i,j-array. Order is the order of basis functions:

```
a = D.nurbs(c, weight, order=3, N=100, M=100) .or. a = D.nurbs(c, weight, order=3, density=10.)
```

(See: nurbs.py) (See: nurbsPT.py)

D.curve: create an i-array from a user defined parametric function:

```
a = D.curve(f, N=100)
```

(See: curve.py) (See: curvePT.py)

D.cone: create a i,j-array made of a cone with NxN points of center C, basis Radius Rb, top Radius Rt and height H:

```
a = D.cone((xc,yc,zc), Rb, Rt, H, N=100)
```

(See: cone.py) (See: conePT.py)

D.sphere: create a sphere of center C and radius R, made of an i,j-array with Nx2N points:

```
a = D.sphere((xc,yc,zc), R, N=100)
```

(See: sphere.py) (See: spherePT.py)

D.sphere6: create a sphere of center C and radius R, made of 6 i,j-arrays with 6xNxN points:

```
A = D.sphere6((xc,yc,zc), R, N=100)
```

(See: sphere6.py) (See: sphere6PT.py)

D.sphereYinYang: create a sphere of center C and radius R, made of 2 overlapping i,j-arrays:

```
A = D.sphereYinYang((xc,yc,zc), R, N=100)
```

(See: sphereYinYang.py) (See: sphereYinYangPT.py)

D.torus: create an i,j-array (NRxNr points) made of a portion of a torus of center C, axis Z and radii R (main radius) and r (tube radius) between the angles alphas and alphae (on the XY-plane) and between betas and betae (on the RZ-plane):

```
A = D.torus((xc,yc,zc), R, r, alphas=0, alphae=360, betas=0, betae=360, NR=100, Nr=100)
```

(See: torus.py) (See: torusPT.py)

D.triangle: create a TRI-array made of a single triangle with 3 points P1, P2, P3:

```
a = D.triangle((x1,y1,z1), (x2,y2,z2), (x3,y3,z3))
```

(See: triangle.py) (See: trianglePT.py)

D.quadrangle: create a QUAD-array made of a single quadrangle with 4 points P1, P2, P3, P4:

```
a = D.quadrangle((x1,y1,z1), (x2,y2,z2), (x3,y3,z3), (x4,y4,z4))
```

(See: quadrangle.py) (See: quadranglePT.py)

D.surface: create a i,j-array from a user defined parametric function:

```
a = D.surface(f, N=100)
```

(See: surface.py) (See: surfacePT.py)

D.text1D: create i-arrays describing a given text, offset is the space between letters, font is the font name ('text1'), smooth is an integer indicating letter smoothness (0-4):

```
A = D.text1D(text, font='text1', smooth=0, offset=0.5)
```

(See: text1D.py) (See: text1DPT.py)

D.text2D: create a TRI-array describing given text in 2D:

```
a = D.text2D(text, font='text1', smooth=0, offset=0.5)
```

(See: text2D.py) (See: text2DPT.py)

D.text3D: create a TRI-array describing given text in 3D:

```
a = D.text3D(text, font='text1', smooth=0, offset=0.5)
```

(See: text3D.py) (See: text3DPT.py)

1.3 Simple operations on geometries

D.addSeparationLine: for two i-arrays a and b in contact, provides a list two i-arrays, with b defined in both arrays:

```
A = T.addSeparationLine(a, b)
```

(See: addSeparationLine.py) (See: addSeparationLinePT.py)

1.4 Surface (volume) mesh from curve (surface) mesh

D.lineGenerate: create a surface geometry by driving an i-array with an i-array curve or a set of curves. The initial i-array is in m, the driving curve is c, the resulting i,j-array is in a:

```
a = D.lineGenerate(m, c)
```

(See: lineGenerate.py) (See: lineGeneratePT.py)

D.axisym: create an axisymmetrical mesh from one of its borders. The input array a can be an (i,j) structured array, a ”TRI” or ”QUAD” unstructured array. Output array is respectively an (i,j,k) array, a ”PENTA” or an ”HEXA” array. Rotation center and axis are (xo,yo,zo) and (nx,ny,nz), teta is the azimuthal sector, nteta is the number of discretisation points in the azimuthal direction. A 1D curve describing the r a function of theta can alternatively be specified:

```
b = D.axisym(a, (xo,yo,zo), (nx,ny,nz), teta=360., Nteta=360) .or. b = D.axisym(a, (xo,yo,zo), (nx,ny,nz), rmod=c)
```

(See: axisym.py) (See: axisymPT.py)

D.volumeFromCrossSections: create a volume mesh from cross-section curves in the (x,y) plane. One curve (defined by a BAR-array) per cross-section is required. C must be a list of those curves corresponding to the different cross-sections of the solid body at different elevation z. a is the volume of the solid body:

```
a = D.volumeFromCrossSections(C)
```

(See: volumeFromCrossSections.py) (See: volumeFromCrossSectionsPT.py)

1.5 Information on geometries

D.getLength: return the length of an i-array or a BAR-array:

```
l = D.getLength(a) .or. l = getLength(A)
```

In the case of a list of meshes, the total length is returned.

(See: getLength.py) (See: getLengthPT.py)

D.getDistantIndex: return the index of the point distant of a signed distance l from point indexed ind in an i-array. Index starts at 1:

```
index = D.getDistantIndex(a, ind, l)
```

(See: getDistantIndex.py) (See: getDistantIndexPT.py)

D.getNearestPointIndex: let a point P be defined by (x,y,z). Return the index and the square distance of the nearest point of P in a:

```
(index, d2) = D.getNearestPointIndex(a, (x,y,z)) .or. (index, d2) = D.getNearestPointIndex(A, (x,y,z)) .or. [(index, d2)] = D.getNearestPointIndex(A, [(x,y,z)])
```

(See: getNearestPointIndex.py) (See: getNearestPointIndexPT.py)

D.getCurvilinearAbscissa: return the curvilinear abscissa for all points of an i-array or a BAR-array:

```
b = D.getCurvilinearAbscissa(a)
```

(See: getCurvilinearAbscissa.py) (See: getCurvilinearAbscissaPT.py)

D.getDistribution: return the curvilinear abscissa for all points of a structured i-array in a coordinate form:

```
b = D.getDistribution(a)
```

(See: getDistribution.py) (See: getDistributionPT.py)

D.getSharpestAngle: return the sharpest angle (in degrees and in [0-360]) for all nodes of an unstructured surface. The returned angle is the sharpest angle between pairs of adjacent elements to which the node belongs to:

```
b = D.getSharpestAngle(a) .or. B = D.getSharpestAngle(A)
```

(See: getSharpestAngle.py) (See: getSharpestAnglePT.py)

D.getCurvatureAngle: return the curvature angle (in degrees and in [0-360]) of all nodes of an i-array or a BAR-array. The returned angle is the angle between edges belonging to node:

```
b = D.getCurvatureAngle(a) .or. B = D.getCurvatureAngle(A)
```

(See: getCurvatureAngle.py) (See: getCurvatureAnglePT.py)

D.getCurvatureRadius: return the curvature radius of all the points in an i-array. This radius is signed positive for convex regions, negative for concave regions:

```
b = D.getCurvatureRadius(a) .or.  B = D.getCurvatureRadius(A)
```

(See: getCurvatureRadius.py) (See: getCurvatureRadiusPT.py)

D.getCurvatureHeight: return the curvature height 'hmax' of all nodes of a 1D array (structured or BAR) or a 2D array ((i,j)-array, TRI or QUAD):

```
b = D.getCurvatureHeight(a) .or. B = D.getCurvatureHeight(A)
```

(See: getCurvatureHeight.py) (See: getCurvatureHeightPT.py)

D.getTangent: return the unitary tangent vector of all nodes of a 1D array (only structured):

```
b = D.getTangent(a) .or. B = D.getTangent(A)
```

(See: getTangent.py) (See: getTangentPT.py)

1.6 Example files

Example file: point.py

```
# - point (array) -
import Geom as D
import Converter as C

a = D.point((0,0,0))
C.convertArrays2File([a], "out.plt")
```

Example file: pointPT.py

```
# - point (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.point((0,0,0))
b = D.point((1,1,1))
C.convertPyTree2File([a,b], "out.cgns")
```

Example file: naca.py

```
# - naca (array) -
import Geom as D
import Converter as C

a = D.naca(12.)
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file: nacaPT.py

```
# - naca (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.naca(12.)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: line.py

```
# - line (array) -
import Geom as D
import Converter as C

a = D.line((0,0,0), (1,0,0))
C.convertArrays2File([a], 'out.plt')
```

Example file: linePT.py

```
# - line (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.line((0,0,0), (1,0,0))
C.convertPyTree2File(a, 'out.cgns')
```

Example file: polyline.py

```
# - polyline (array) -
import Geom as D
import Converter as C

a = D.polyline([(0.,0.,0.),(1.,1.,0.),(2.,0.,0.)])
C.convertArrays2File([a], "out.plt")
```

Example file: polylinePT.py

```
# - polyline (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.polyline([(0.,0.,0.),(1.,1.,0.),(2.,0.,0.)])
C.convertPyTree2File(a, 'out.cgns')
```

Example file: circle.py

```
# - circle (array) -
import Geom as D
import Converter as C

a = D.circle((0,0,0), 1. , 0., 360.)
C.convertArrays2File([a], "out.plt")
```

Example file: circlePT.py

```
# - circle (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.circle((0,0,0), 1. , 0., 360.)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: bezier.py

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

# Bezier 1D
pts = D.polyline([(0.,0.,0.), (0.,1.,0.), (2.,1.,0.), (2.,0.,0.),\
                  (4.,-1.,0.), (5.,6.,0.)])
# With a specified number of points
a = D.bezier(pts, N=100)
# With a specified point density
b = D.bezier(pts, density=10.)
C.convertArrays2File([pts, a, b], 'out.plt')

# Bezier 2D
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.,4.])
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, density=10.)
C.convertArrays2File([a]+[b], 'out2.plt')
```

Example file: bezierPT.py

```
# - bezier (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

# Bezier 1D
pts = D.polyline([(0.,0.,0.), (0.,1.,0.), (2.,1.,0.), (2.,0.,0.),
                  (4.,-1.,0.), (5.,6.,0.),])
a = D.bezier(pts, 100); a[0] = 'bezier'
C.convertPyTree2File(a, 'out.cgns')
```

Example file: spline.py

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

# Spline 1D
c = D.polyline([(0.,0.,0.), (1.,1.,0.), (2.,1.,0.), \
                (3.,0.,0.), (4.,-1.,0.), (5.,6.,0.), \
                (6.,1.,0.), (7.,2.,0.), (8.,1.,0.), \
                (9.,-1.,0.), (10.,1.,0.), (11.,-1.,0.)])
# Avec un nombre de pts specifie
d = D.spline(c, 3, N=100)
# Avec une densite de points specifiee
e = D.spline(c, 3, density=10.)
C.convertArrays2File([c, d, e], 'out.plt')

# Spline 2D
ni = 4; nj = 4
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.,5.])
C.setValue(a, (1,4,1), [1.,4.,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.,5.])
C.setValue(a, (2,4,1), [2.,4.,2.])
C.setValue(a, (3,1,1), [3.,1.,2.])
C.setValue(a, (3,2,1), [3.,2.,5.])
C.setValue(a, (3,3,1), [3.,3.,5.])
C.setValue(a, (3,4,1), [3.,4.,2.])
C.setValue(a, (4,1,1), [4.,1.,2.])
C.setValue(a, (4,2,1), [4.,2.,5.])
C.setValue(a, (4,3,1), [4.,3.,5.])
C.setValue(a, (4,4,1), [4.,4.,2.])

b = D.spline(a, 4, N=30, M=30)
c = D.spline(a, 4, density=10.)
C.convertArrays2File([a, b, c], 'out2.plt')
```

Example file: splinePT.py

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

# Spline 1D
c = D.polyline([(0.,0.,0.), (1.,1.,0.), (2.,1.,0.), \
                (3.,0.,0.), (4.,-1.,0.), (5.,6.,0.), \
                (6.,1.,0.), (7.,2.,0.), (8.,1.,0.), \
                (9.,-1.,0.), (10.,1.,0.), (11.,-1.,0.)])
d = D.spline(c,3,100); d[0] = 'spline'
C.convertPyTree2File(d, 'out.cgns')
```

Example file: nurbs.py

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

a = D.polyline ([(4.1,0.1,1.1), (1.1,0.2,1.2), (1.1,1.3,1.3),
                 (1.1,1.5,1.4), (4.5,2.5,1.5), (5.6,1.5,1.6),
                 (6.7,1.7,1.7), (7.8,0.8,1.8), (8.9,-1.9,1.9), (9,0,1)])
w = C.array('W', 10, 1, 1)
w[1][:] = 1
b = D.nurbs(a, w, 4, N=100)
c = D.nurbs(a, w, 4, density=10.)
C.convertArrays2File([a, b, c], 'out.plt')

ni = 10; nj = 10
a = G.cart((0,0,0), (1,1,1), (ni,nj,1))
C.setValue(a, (1,1,1), [1.,1.,1.])
C.setValue(a, (1,2,1), [1.,2.,1.])
C.setValue(a, (1,3,1), [1.,3.,1.])
C.setValue(a, (1,4,1), [1.,4.,1.])
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.,5.])
C.setValue(a, (2,4,1), [2.,4.,2.])
C.setValue(a, (3,1,1), [3.,1.,2.])
C.setValue(a, (3,2,1), [3.,2.,5.])
C.setValue(a, (3,3,1), [3.,3.,12.])
C.setValue(a, (3,4,1), [3.,4.,2.])
C.setValue(a, (4,1,1), [4.,1.,2.])
C.setValue(a, (4,2,1), [4.,2.,5.])
C.setValue(a, (4,3,1), [4.,3.,5.])
C.setValue(a, (4,4,1), [4.,4.,2.])
C.setValue(a, (6,8,1), [4.,6.,14.])
C.setValue(a, (8,6,1), [4.,6.,-4.])
w = C.array('W', 10, 10, 1)

w[1][0][:] = 1
w[1][0][7] = 7
w[1][0][15] = 9

d = D.nurbs(a, w, 4, N=100, M=100)
e = D.nurbs(a, w, 4, density=20.)
C.convertArrays2File([a, d, e], 'out2.plt')
```

Example file: nurbsPT.py

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

ni = 10; nj = 10
a = G.cart((0,0,0), (1,1,1), (ni,nj,1));
a = C.initVars(a,'weight',1.)
C.setValue(a, 'weight', (7,1,1), 7.)
C.setValue(a, 'weight', (9,5,1), 9.)

d = D.nurbs(a,'weight',4,100,100)
C.convertPyTree2File(d, 'out.cgns')

a = D.polyline ([(4.1,0.1,1.1),(1.1,0.2,1.2),(1.1,1.3,1.3),(1.1,1.5,1.4),(4.5,2.5,1.5),(5.6,1.5,1.6),(6.7,1.7,1
a = C.initVars(a,'weight',1.)
C.setValue(a, 'weight', (7,1,1), 7.)
```

```
C.setValue(a, 'weight', (9,1,1), 9.)
b = D.nurbs(a,'weight',4,2000)
C.convertPyTree2File(b, 'out2.cgns')
```

Example file: curve.py

```
# - curve (array) -
import Converter as C
import Geom as D

# Definition of parametric curve by a function
def f(t):
    x = t
    y = t*t+1
    z = 0.
    return (x,y,z)
a = D.curve(f)

# Definition by equation
b = D.curve('{x}=cos(2*pi*{t}); {y}=sin(2*pi*{t}); {z} = 0.')

# Definition from data base
from Geom.Parametrics import base
c = D.curve(base['circle'])
C.convertArrays2File([a,b], "out.plt")
```

Example file: curvePT.py

```
# - curve (pyTree) -
import Converter.PyTree as C
import Geom.PyTree as D

# User definition of parametric curve
def f(t):
    x = t; y = t*t+1; z = 0.
    return (x,y,z)

a = D.curve(f)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: cone.py

```
# - cone (array) -
import Geom as D
import Converter as C

a = D.cone((0,0,0), 1. , 0.5, 1.)
C.convertArrays2File([a], "out.plt")
```

Example file: conePT.py

```
# - cone (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.cone((0,0,0), 1. , 0.5, 1.)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: sphere.py

```
# - sphere (array) -
import Geom as D
import Converter as C

a = D.sphere((0,0,0), 1., 20)
C.convertArrays2File([a], "out.plt")
```

Example file: spherePT.py

```
# - sphere (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.sphere((0,0,0), 1., 20)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: sphere6.py

```
# - sphere6 (array) -
import Geom as D
import Converter as C

a = D.sphere6((0,0,0), 1., 20)
C.convertArrays2File(a, "out.plt")
```

Example file: sphere6PT.py

```
# - sphere6 (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

A = D.sphere6((0,0,0), 1., 20)
C.convertPyTree2File(A, 'out.cgns')
```

Example file: sphereYinYang.py

```
# - sphereYinYang (array) -
import Geom as D
import Converter as C

a = D.sphereYinYang((0,0,0), 1., 50)
C.convertArrays2File(a, "out.plt")
```

Example file: sphereYinYangPT.py

```
# - sphereYinYang (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.sphereYinYang((0,0,0), 1., 50)
C.convertPyTree2File(a, "out.cgns")
```

Example file: torus.py

```
# - torus (array) -
import Geom as D
import Converter as C

a = D.torus((0,0,0), 5., 2.)
C.convertArrays2File([a], "out.plt")
```

Example file: torusPT.py

```
# - torus (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.torus((0.,0.,0.), 5., 2.)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: triangle.py

```
# - triangle (array) -
import Geom as D
import Converter as C

a = D.triangle((0,0,0), (0.1,0.,0.1), (0.05, 0.08, 0.1))
C.convertArrays2File([a], "out.plt", "bin_tp")
```

Example file: trianglePT.py

```
# - triangle (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.triangle((0,0,0), (0.1,0.,0.1), (0.05, 0.08, 0.1))
C.convertPyTree2File(a, 'out.cgns')
```

Example file: quadrangle.py

```
# - quadrangle (array) -
import Geom as D
import Converter as C

a = D.quadrangle((0,0,0.1), (0.1,0.,0.1), (0.05, 0.08, 0.1), (0.02,0.05,0.1))
C.convertArrays2File([a], "out.plt")
```

Example file: quadranglePT.py

```
# - quadrangle (PyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.quadrangle((0,0,0.1), (0.1,0.,0.1), (0.05, 0.08, 0.1), (0.02,0.05,0.1))
C.convertPyTree2File(a, 'out.cgns')
```

Example file: surface.py

```
# - surface (array) -
import Converter as C
import Geom as D

# User definition of parametric curve by a function
def f(t,u):
    x = t+u
    y = t*t+1+u*u
    z = u
    return (x,y,z)

a = D.surface(f)

# Definition by formula
b = D.surface('{x} = cos(pi*{t}); {y} = sin(pi*{u}); {z} = {t}*{u}')
C.convertArrays2File([a, b], 'out.plt')
```

Example file: surfacePT.py

```
# - surface (PyTree) -
import Converter.PyTree as C
import Geom.PyTree as D

# User definition of parametric curve
def f(t,u):
    x = t+u; y = t*t+1+u*u; z = u
    return (x,y,z)

a = D.surface(f)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: text1D.py

```
# - text1D (array) -
import Geom as D
import Converter as C
import Transform as T

a = D.text1D("Cassiopee - text1")
b = D.text1D("Cassiopee - text1 smoothed", smooth=4, offset=1.)
b = T.translate(b, (0,-12,0))
c = D.text1D("Cassiopee - vera", font='vera')
c = T.translate(c, (0,-24,0))
d = D.text1D("Cassiopee - chancery", font='chancery')
d = T.translate(d, (0,-36,0))
e = D.text1D("Cassiopee - courier", font='courier')
e = T.translate(e, (0,-48,0))
f = D.text1D("Cassiopee - nimbus", font='nimbus')
f = T.translate(f, (0,-60,0))

C.convertArrays2File(a+b+c+d+e+f, 'out.plt')
```

Example file: text1DPT.py

```
# - text1D (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.text1D("CASSIOPEE")
C.convertPyTree2File(a, 'out.cgns')
```

Example file: text2D.py

```
# - text2D (array) -
import Geom as D
import Converter as C

a = D.text2D("ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789", smooth=0, offset=1.)
C.convertArrays2File([a], 'out.plt')
```

Example file: text2DPT.py

```
# - text2D (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.text2D("Cassiopee")
C.convertPyTree2File(a, 'out.cgns')
```

Example file: text3D.py

```
# - text3D (array) -
import Geom as D
import Converter as C

a = D.text3D("Cassiopee", smooth=1)
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file: text3DPT.py

```
# - text3D (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

a = D.text3D("CASSIOPEE")
C.convertPyTree2File(a, 'out.cgns')
```

Example file: addSeparationLine.py

```
# - addSeparationLine (array)-
import Geom as D
import Converter as C

# Add a line to a circle
a1 = D.circle((0,0,0), 1, 0., 360, 1000)
a2 = D.line((0.,1.,0.), (0.,2.,0), 100)
arrays = D.addSeparationLine(a1, a2)
C.convertArrays2File(arrays, "out.plt")
```

Example file: addSeparationLinePT.py

```
# - addSeparationLine (pyTree)-
import Geom.PyTree as D
import Converter.PyTree as C

a1 = D.circle((0,0,0), 1, 0., 360, 1000)
a2 = D.line((0.,1.,0.), (0.,2.,0), 100)
zones = D.addSeparationLine(a1, a2)
C.convertPyTree2File(zones, 'out.cgns')
```

Example file: lineGenerate.py

```
# - lineGenerate (array) -
import Geom as D
import Converter as C

# With one driving curve
a = D.naca(12.)
b = D.line((0,0,0), (0.,0.,1.))
c = D.lineGenerate(a, b)
C.convertArrays2File([c], 'out.plt')

# With a set of driving curves
a = D.naca(12.)
d1 = D.line((0,0,0), (0.,0.,1.))
d2 = D.line((1,0,0), (2,0,1))
c = D.lineGenerate(a, [d1,d2])
C.convertArrays2File([c,d1,d2,a], 'out.plt')
```

Example file: lineGeneratePT.py

```
# - lineGenerate (pyTree)-
import Geom.PyTree as D
import Converter.PyTree as C

# With one driving curve
a = D.naca(12.)
l = D.line((0,0,0), (0,0.,1.))
a = D.lineGenerate(a, l)
C.convertPyTree2File(a, 'out.cgns')

# With a set of driving curves
a = D.naca(12.)
d1 = D.line((0,0,0), (0.,0.,1.))
d2 = D.line((1,0,0), (2,0,1))
a = D.lineGenerate(a, [d1,d2])
C.convertPyTree2File(a, 'out.cgns')
```

Example file: axisym.py

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

# Axisym a curve
a0 = D.line((0.5,0,0), (0.6,0,1))
a = D.axisym(a0,(0.,0.,0.),(0.,0.,1.),360.,360)
C.convertArrays2File([a], "out.plt")

# Axisym a curve with varying r
a0 = D.line((1.0,0,0), (0.,0,1))
a1 = D.circle((0,0,0), 2.)
import Modeler.Models as Models
a1 = Models.circle2(1, 0.8)
a = D.axisym(a0, (0.,0.,0.), (0.,0.,1.), rmod=a1)
C.convertArrays2File([a,a0,a1], "out.plt")

# Axisym a 2D cart grid
a0 = G.cart((0.,0.,0.), (0.1,0.1,0.2),(10,10,1))
a = D.axisym(a0,(1.,0.,0.),(0.,1.,0.),30.,4)
#C.convertArrays2File([a], "out.plt")
```

Example file: axisymPT.py

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

# Axisym a curve
a0 = D.line((0.5,0,0), (0.6,0,1))
a = D.axisym(a0,(0.,0.,0.),(0.,0.,1.),360.,360)
C.convertPyTree2File([a], "out.cgns")

# Axisym a curve with varying r
a0 = D.line((1.0,0,0), (0.,0,1))
a1 = D.circle((0,0,0), 2.)
a = D.axisym(a0, (0.,0.,0.), (0.,0.,1.), rmod=a1)
C.convertPyTree2File([a,a0,a1], "out.cgns")
```

```
# Axisym a 2D cart grid
a = G.cart((0.,0.,0.), (0.1,0.1,0.2),(10,10,1))
a = D.axisym(a,(1.,0.,0.),(0.,1.,0.),30.,4)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: volumeFromCrossSections.py

```
# - volumeFromCrossSections (array) -
import Converter as C
import Geom as D

contours = []
a = D.polyline([(0.,0.,0.),(1.,0.,0.),(1.,1.,0.),
                (0., 1., 0.),(0.,0.,0.)])
contours = [D.polyline([(0.,0.,2.),(1.,0.,2.),(1.,1.,2.),
                        (0., 1., 2.),(0.,0.,2.)])]
contours.append(a)

vol = D.volumeFromCrossSections(contours)
C.convertArrays2File([vol]+contours, 'out.plt')
```

Example file: volumeFromCrossSectionsPT.py

```
# - volumeFromCrossSection (pyTree) -
import Converter.PyTree as C
import Geom.PyTree as D

contours = []
a = D.polyline([(0.,0.,0.),(1.,0.,0.),(1.,1.,0.),(0., 1., 0.),(0.,0.,0.)])
contours = [D.polyline([(0.,0.,2.),(1.,0.,2.),(1.,1.,2.),(0., 1., 2.),(0.,0.,2.)])]
contours.append(a)

vol = D.volumeFromCrossSections(contours)
t = C.newPyTree(['Base',3,vol])
C.convertPyTree2File(t, 'out.cgns')
```

Example file: getLength.py

```
# - getLength (array) -
import Geom as D

a = D.line((0,0,0), (1,0,0))
print D.getLength(a)
```

Example file: getLengthPT.py

```
# - getLength (pyTree)-
import Geom.PyTree as D

a = D.line((0,0,0), (1,0,0)); print D.getLength(a)
```

Example file: getDistantIndex.py

```
# - getDistantIndex (array) -
import Geom as D

a = D.line((0.,0.,0.), (1.,0.,0), 100)
print 'distant Index:', D.getDistantIndex(a, 25, 0.2)
print 'distant Index:', D.getDistantIndex(a, 25, -0.2)
```

Example file: getDistantIndexPT.py

```
# - getDistantIndex (pyTree)-
import Geom.PyTree as D

a = D.line((0.,0.,0.), (1.,0.,0), 100)
print 'distant Index:', D.getDistantIndex(a, 25, 0.2)
```

Example file: getNearestPointIndex.py

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

a = G.cart((0.,0.,0.), (0.1,0.1,0.2),(10,10,1))

inds = D.getNearestPointIndex(a, (0.55,0.34,0))
print inds
inds = D.getNearestPointIndex(a, [(0.55,0.34,0), (0.56,0.32,0)])
print inds
```

Example file: getNearestPointIndexPT.py

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

a = G.cart((0.,0.,0.), (0.1,0.1,0.2),(10,10,1))
inds = D.getNearestPointIndex(a, (0.55,0.34,0))
print inds
```

Example file: getCurvilinearAbscissa.py

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

a = D.line((0.,0.,0.), (1.,0.,0), 100)
a2 = D.line((1.,0.,0.), (1.,1,0), 100)
a = T.join (a, a2)
a3 = D.getCurvilinearAbscissa( a )
a = C.addVars([a, a3])
C.convertArrays2File([a], "out.plt", "bin_tp")
```

Example file: getCurvilinearAbscissaPT.py

```
# - getCurvilinearAbscissa (pyTree)-
import Converter.PyTree as C
import Geom.PyTree as D

a = D.line((0.,0.,0.), (1.,0.,0), 100)
a = D.getCurvilinearAbscissa(a)
t = C.newPyTree(['Base',1]); t[2][1][2].append(a)
C.convertPyTree2File(t, 'out.cgns')
```

Example file: getDistribution.py

```
# - getDistribution (array) -
import Geom as D

Foil = D.naca(12., N=49)
print D.getDistribution(Foil)
```

Example file: getDistributionPT.py

```
# - getDistribution (PyTree) -
import Geom.PyTree as D

Foil = D.naca(12., N=49)
print D.getDistribution(Foil)
```

Example file: getSharpestAngle.py

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

N = 10
d1 = G.cart((0.,0.,0.), (0.05,1,1),(N,1,4))
d2 = G.cart((0.,0.,0.), (1.,0.001,1),(1,10*N,4))
d2 = T.rotate(d2,(0.,0.,0.),(0.,0.,1.),30.)
s = T.join(d1,d2)
s = C.convertArray2Hexa(s)
s = T.reorder(s,(-1,))
r = D.getSharpestAngle(s)
s = C.addVars([s,r])
C.convertArrays2File([s], "out.plt")
```

Example file: getSharpestAnglePT.py

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

N = 10
d1 = G.cart((0.,0.,0.), (0.05,1,1),(N,1,4))
d2 = G.cart((0.,0.,0.), (1.,0.001,1),(1,10*N,4))
d2 = T.rotate(d2,(0.,0.,0.),(0.,0.,1.),30.)
s = T.join(d1,d2)
s = C.convertArray2Hexa(s)
s = T.reorder(s,(-1,))
s = D.getSharpestAngle(s)
ts = C.newPyTree(['Base',2]); ts[2][1][2] +=[s]
C.convertPyTree2File(ts,"out.cgns")
```

Example file: getCurvatureAngle.py

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

a1 = D.line((0.,0.,0.), (1.,0.,0), 100)
a2 = D.line((1.,0.,0.), (1.,1,0), 100)
a = T.join (a1, a2)
a3 = D.getCurvatureAngle(a)
a = C.addVars([a, a3])
C.convertArrays2File([a], 'out.plt')
```

Example file: getCurvatureAnglePT.py

```
# - getCurvatureAngle (pyTree) -
import Converter.PyTree as C
import Geom.PyTree as D

a = D.polyline([(0.,0.,0.),(1.,1.,0.),(2.,0.,0.)])
a = D.getCurvatureAngle(a)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: getCurvatureRadius.py

```
# - getCurvatureRadius (array) -
import Geom as D

pts = D.polyline([(6,0.01,1), (5.4,0.036,1), (4.8,0.064,1), (2.5,0.21,1),
                  (0.3,0.26,1),(0,0.047,1),(0,0,0)])
a = D.bezier( pts, 100 )
rad = D.getCurvatureRadius(a); print rad
```

Example file: getCurvatureRadiusPT.py

```
# - getCurvatureRadius (pyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

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

Example file: getCurvatureHeight.py

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

a1 = D.line((0.,0.,0.), (1.,0.,0), 100)
a2 = D.line((1.,0.,0.), (1.,1,0), 100)
a = T.join (a1, a2)
hmax = D.getCurvatureHeight( a )
a = C.addVars([a,hmax])
C.convertArrays2File([a], 'out.plt')
```

Example file: getCurvatureHeightPT.py

```
# - getCurvatureHeight(pyTree) -
import Converter.PyTree as C
import Geom.PyTree as D

a = D.polyline([(0.,0.,0.),(1.,1.,0.),(2.,0.,0.)])
a = D.getCurvatureHeight(a)
C.convertPyTree2File(a, 'out.cgns')
```

Example file: getTangent.py

```
# - getTangent (array) -
import Geom as D

c = D.polyline([(0,0,0),(1,1,0),(2,-1,0)])
a = D.spline(c, order=3, density=10.)

print D.getTangent(a)
```

Example file: getTangentPT.py

```
# - getTangent (PyTree) -
import Geom.PyTree as D
import Converter.PyTree as C

c = D.polyline([(0,0,0),(1,1,0),(2,-1,0)])
a = D.spline(c, order=3, density=10.)

print D.getTangent(a)
```