Post Module V1.4
- User Guide -

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

September 23, 2009

1 Post : solution post-processing module

1.1 Preamble

This module provides post-processing tools.

When using the Converter array interface, a (or b) denotes an array, and A (or B) denotes a list of arrays. Then, Post module must be imported:

```
import Post as P
```

When using the pyTree interface, import the module:

```
import Post.PyTree as P
```

In that case, a is a zone node. A is a list of zone nodes or a complete pyTree.

1.2 Variables computation

Some variables can be computed from conservative variables. The list of the names of the variables to compute and parameters required (e.g. 'gamma' and 'rgp' to compute the temperature) must be provided.
Parameters can be: - 'gamma' for the specific heat ratio (def. value: 1.4), - 'rgp' for the perfect gas constant (def. value: 287.053), - 'betas' and 'Cs' (Sutherland's law constants), or 'Cs','Ts' and 'mus' (def. values: betas = 1.458e-6, Cs = 110.4) - 's0' for a constant entropy (def. value: 0.), defined by:

```
s0 = sref - rgp gamma/(gamma-1) ln(Tref) + rgp ln(Pref),
where sref, Tref and Pref are defined for a reference state.
```

Computed variables are defined by their CGNS names:

- 'VelocityX', 'VelocityY', 'VelocityZ' for components of the absolute velocity,
- 'VelocityMagnitude' for the absolute velocity magnitude,
- 'Pressure' for the static pressure,
- 'Temperature' for the static temperature,
- 'Enthalpy' for the enthalpy,
- 'Entropy' for the entropy,
- 'Mach' for the Mach number,
- 'ViscosityMolecular' for the fluid molecular viscosity,
- 'PressureStagnation' for stagnation pressure,
- 'TemperatureStagnation' for stagnation temperature,
- 'PressureDynamic' for dynamic pressure.

```
b = P.computeVariables(a, ['varname1', 'varname2'], ['gamma',1.4,'rgp',1/1.4]) .or. B = P.computeVariables(A, ...)
```

(See : Examples/post/computeVariables.py) (See : Examples/post/computeVariablesPT.py)

Compute gradient with respect to each variable x, y or z of a varname field defined in a. The returned field is defined at cell centers for structured grids and elements centers for unstructured grids. When using pyTree interface, the gradient is stored in FlowSolution#Centers of the returned zone:

```
b = P.computeGrad(a, varname) .or. B = P.computeGrad(A, varname)
```

(See : Examples/post/computeGrad.py) (See : Examples/post/computeGradPT.py)

Compute curl of a 3D vector defined by its variable names ['vectx','vecty','vectz'] in a. The returned field is defined at cell centers for structured grids and elements centers for unstructured grids:

```
b = P.computeCurl(a, ['vectx','vecty','vectz']) .or. B = P.computeCurl(A, ['vectx','vecty','vectz'])
```

(See : Examples/post/computeCurl.py) (See : Examples/post/computeCurl.py)

1.3 Solution selection

Select cells with respect to a given criterion. The criterion must be defined as a python function returning True (cell is selected) or False (cell is not selected):

```
b = P.selectCells(a, F, ['var1', 'var2']) .or B = P.selectCells(A, F, ['var1', 'var2'])
```

(See : Examples/post/selectCells.py) (See : Examples/post/selectCellsPT.py)

Select the interior faces of a mesh. Interior faces are faces with two neighbours. If 'strict' is set to 1, select the interior faces that have only interior nodes. Default value is 'strict'=0:

```
b = P.interiorFaces(a, strict)
```

(See : Examples/post/interiorFaces.py) (See : Examples/post/interiorFacesPT.py)

Select the exterior faces of a mesh:

```
b = P.exteriorFaces(a)
```

(See : Examples/post/exteriorFaces.py) (See : Examples/post/exteriorFacesPT.py)

Coarsen a triangle mesh by a providing a coarsening indicator, defined by indic. The indic array must be a i-array, whose dimension is equal to the number of elements in the initial triangulation. It is equal to 1 if the element has to be coarsened, 0 elsewhere. Triangles are merged by edge con-

traction, if tagged to be coarsened by indic and if new triangles are closer than eps to the original triangle. Required mesh quality is controlled by argqual : argqual equal to 0.5 corresponds to an equilateral triangle, whereas a value near zero corresponds to a bad triangle shape.
Default values are argqual = 0.25 and eps = 1e-6:

```
b = P.mergeElts(a, indic, argqual, eps)
```

(See : Examples/post/mergeElts.py) (See : Examples/post/mergeEltsPT.py)

1.4 Solution extraction

Extract the field in one point given a solution A. The extracted field is returned as a list of values for each field. If the point (x,y,z) is not interpolable from a grid, then an empty list is returned.
In the PyTree version, extractPoint returns the extracted solution from solutions located at nodes followed by the solution extracted from solutions at centers.
If 'cellN', 'ichim', 'cellnf', 'status', or 'cellNF' variable is defined, it is returned in the last position in the output array. The interpolation order can be 2, 3, or 5. Default value is 2.
If some blocks in A define surfaces, a tolerance 'eps' for interpolation cell search can be defined. Its default value is 1.e-6:

```
field = P.extractPoint(A, (x,y,z), order, eps)
```

(See : Examples/post/extractPoint.py) (See : Examples/post/extractPointPT.py)

Slice a solution A with a plane. The extracted solution is interpolated from A. Interpolation order can be 2, 3, or 5 (but the 5th order is very time-consuming for the moment). The best solution is kept. Plane is defined by **c1 x + c2 y + c3 z + c4 = 0**:

```
b = P.extractPlane(A, (c1, c2, c3, c4), order, eps)
```

(See : Examples/post/extractPlane.py) (See : Examples/post/extractPlanePT.py)

Extract a solution on a given mesh. The input solution containing coordinates and field is A and the mesh on which solution is extracted is a. Output is a mesh consisting of extraction mesh coordinates and extracted fields. This routine makes use of 2nd, 3rd or 5th order interpolations (order='2','3' or '5'). Default value is 2 order:

```
b = P.extractMesh(A, a, order, eps)
```

(See : Examples/post/extractMesh.py) (See : Examples/post/extractMeshPT.py)

Build an unstructured unique surface mesh, given a list of structured overlapping surface grids A. Cell nature field is used to find blanked (0) and interpolated (2) cells:

```
a = P.zipper(A, options)
```

The options argument is a list of arguments such as ["argName", argValue]. Option names can be :
- 'overlapTol' for tolerance required between two overlapping grids : if the projection distance between them is under this value then the grids are considered to be overset. Default value is 1.e-5.
- For some cases, 'matchTol' can be set to modify the matching boundaries tolerance. Default value is set 1e-6. In most cases, one needn't modify this parameter.
(See : Examples/post/zipper.py) ¡a href = "Examples/post/zipperPT.py"¿ zipping an overset surface (pyTree) ¡/a¿.

An alternative to "zipper" is "usurp". Result is a ratio field located at cell centers. In case of no overset, ratio are set to 1, otherwise ratio represents the percentage of overlap of a cell by another

mesh.

When using the array interface, the input arrays are a list of grid arrays A, defining nodes coordinates and a corresponding list of arrays defining the chimera nature of cells at cell centers B. Blanked cells must be flagged by a null value. Other values are equally considered as computed or interpolated cells:

```
C = P.usurp(A, B)
```

When using the pyTree interface, chimera cell nature field must be defined as a center field in A:

```
B = usurp(A)
```

Warning : normal of surfaces grids defined by A must be oriented in the same direction.
(See : Examples/post/usurp.py) (See : Examples/post/usurpPT.py)

1.5 Streams

Compute the stream line starting from point (x0,y0,z0), given a solution A. Default value of nptsmax is 2000. Parameter 'dir' can be 1 : streamline follows velocity, -1 : streamline follows -velocity, 2 : streamline expands in both directions. If 'dir' is not specified by the user, the streamline is expanded in both directions. The output yields the set of extracted points on the streamline, and the input fields at these points. The streamline computation stops when the current point is not interpolable from the input grids:

```
b = P.streamLine(A, (x0,y0,z0), nptsmax, dir)
```

(See : Examples/post/streamLine.py) (See : Examples/post/streamLinePT.py)

Compute the stream ribbon starting from point (x0,y0,z0), of width and direction given by the vector (nx,ny,nz). This vector must be roughly orthogonal to the velocity vector at (x0,y0,z0). Default value of nptsmax is 2000. The output yields the set of extracted points on the streamribbon, and the input fields at these points. The streamribbon computation stops when the current point is not interpolable from the input grids:

```
b = P.streamRibbon(A, (x0,y0,z0), (nx,ny,nz), nptsmax, dir)
```

(See : Examples/post/streamRibbon.py) (See : Examples/post/streamRibbonPT.py)

1.6 Solution integration

For all integration functions, the interface is different when using Converter arrays interface or pyTree interface. For arrays, fields must be input separately, for pyTree, they must be defined in each zone.

Compute the integral of a scalar field (whose name is varString) over the geometry defined by arrays containing the coordinates + field (+ an optional ratio). Solution and ratio can be located at nodes or at centers. For array interface:

```
res = P.integ( [coord], [field], [ratio] )
```

For pyTree interface:

```
res = P.integ( A )
```

(See : Examples/post/integ.py) (See : Examples/post/integPT.py)

Compute the integral of each scalar field times the surface normal over the geometry defined by coord. For array interface:

```
res = P.integNorm( [coord], [field], [ratio] )
```

For pyTree interface:

```
res = P.integNorm( A )
```

(See : Examples/post/integNorm.py) (See : Examples/post/integNormPT.py)

Compute the integral of a vector field times the surface normal over the geometry defined by coord. The input field must have 3 variables. For array interface:

```
res = P.integNormProduct( [coord], [field], [ratio] )
```

For pyTree interface:

```
res = P.integNormProduct( A )
```

(See : Examples/post/integNormProduct.py) (See : Examples/post/integNormProductPT.py)

Compute the integral of a moment over the geometry defined by coord. The input field must have 3 variables. (cx,cy,cz) are the center coordinates. For array interface:

```
res = P.integMoment( [coord], [field], [ratio], (cx,cy,cz) )
```

For pyTree interface:

```
res = P.integMoment( A, (cx,cy,cz) )
```

(See : Examples/post/integMoment.py) (See : Examples/post/integMomentPT.py)

Compute the integral of a moment over the geometry defined by coord, taking into account the surface normal. The input field is a scalar. For array interface:

```
res = P.integMomentNorm( [coord], [field], [ratio], (cx,cy,cz) )
```

For pyTree interface:

```
res = P.integMomentNorm( A, (cx,cy,cz) )
```

(See : Examples/post/integMomentNorm.py) (See : Examples/post/integMomentNormPT.py)

1.7 Example files

Example file : Examples/post/computeVariables.py

```
# - computeVariables (array) -
import Converter as C
import Post as P
import Generator as G

ni = 30; nj = 40
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,2))
c = C.array('ro,rou, rov,row,roE', ni, nj, 2)
c = C.initVars(c, 'ro', 1.)
c = C.initVars(c, 'rou', 1.)
c = C.initVars(c, 'rov', 0.)
c = C.initVars(c, 'row', 0.)
c = C.initVars(c, 'roE', 1.)
m = C.addVars([m,c])

#-----------------------------------------
# Pressure and Mach number extraction
# default values of rgp and gamma are used
#-----------------------------------------
p = P.computeVariables(m, ['Mach'],[])
m = C.addVars([m,p])
C.convertArrays2File([m], "out.plt", "bin_tp")
```

Example file : Examples/post/computeVariablesPT.py

```
# - computeVariables (pyTree) -
import Converter.PyTree as C
import Post.PyTree as P
import Generator.PyTree as G

ni = 30; nj = 40
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,2))
vars = ['Density','MomentumX', 'MomentumY', 'MomentumZ', \
        'EnergyStagnationDensity']
for v in vars:
    m = C.addVars(m, v)
    m = C.addVars(m,'centers:Density')
m = C.initVars(m, vars[0],1.)

# Pressure and Mach number extraction
m = P.computeVariables(m, ['Mach', 'Pressure'], [])
t = C.newPyTree(['Base',3]); t[1][2].append(m)
C.convertPyTree2File(t, "out.cgns")
```

Example file : Examples/post/computeGrad.py

```
# - computeGrad (array) -
import Converter as C
import Post as P
import Generator as G

def F(x,y):
    return 2*x+x*y

ni = 30; nj = 40; nk = 3
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
m = C.addVars(m, 'ro')
m = C.initVars(m, 'ro', F, ['x','y'])
p = P.computeGrad(m,'ro') # p is defined on centers
p = C.center2Node(p) # back on initial mesh
p = C.addVars([m, p])
C.convertArrays2File([p], 'out.plt', 'bin_tp')
```

Example file : Examples/post/computeGradPT.py

```
# - computeGrad (pyTree) -
import Converter.PyTree as C
import Post.PyTree as P
import Generator.PyTree as G

def F(x,y):
    return 2*x+x*y

ni = 30; nj = 40; nk = 10
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
m = C.addVars(m, 'Density')
m = C.initVars(m, 'Density', F, ['CoordinateX','CoordinateY'])
m = P.computeGrad(m,'Density')
t = C.newPyTree(['Base',3]); t[1][2].append(m)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/post/computeCurl.py

```
# - computeCurl (array) -
import Converter as C
import Post as P
import Generator as G
```

```
def F(x,y,z):
    return 12*y*y + 4

ni = 30; nj = 40; nk = 3
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
m = C.addVars(m,'F1')
m = C.initVars(m,'F1',F,['x','y','z'])
m = C.addVars(m,'F2')
m = C.initVars(m,'F2',0.)
m = C.addVars(m,'F3')
m = C.initVars(m,'F3',0.)

varname = ['F1','F2','F3']
p = P.computeCurl(m, varname) # defined on centers
p = C.center2Node(p) # back on init grid
p = C.addVars([m,p])
C.convertArrays2File([p], "out.plt", "bin_tp")
```

Example file : Examples/post/computeCurl.py

```
# - computeCurl (array) -
import Converter as C
import Post as P
import Generator as G

def F(x,y,z):
    return 12*y*y + 4

ni = 30; nj = 40; nk = 3
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
m = C.addVars(m,'F1')
m = C.initVars(m,'F1',F,['x','y','z'])
m = C.addVars(m,'F2')
m = C.initVars(m,'F2',0.)
m = C.addVars(m,'F3')
m = C.initVars(m,'F3',0.)

varname = ['F1','F2','F3']
p = P.computeCurl(m, varname) # defined on centers
p = C.center2Node(p) # back on init grid
p = C.addVars([m,p])
C.convertArrays2File([p], "out.plt", "bin_tp")
```

Example file : Examples/post/selectCells.py

```
# - selectCells (array) -
import Converter as C
import Generator as G
import Post as P

a = G.cart( (0,0,0), (1,1,1), (11,11,11) )
def F(x, y, z):
    if (x + 2*y + z   > 20.):
        return True
    else:
        return False

b = P.selectCells(a, F, ['x', 'y', 'z'])
C.convertArrays2File([b], 'out.plt')
```

Example file : Examples/post/selectCellsPT.py

```
# - selectCells (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.PyTree as P
```

```
def F(x, y, z):
    if (x + 2*y + z   > 20.):
        return True
    else:
        return False

a = G.cart( (0,0,0), (1,1,1), (11,11,11) )
a = P.selectCells(a, F, ['CoordinateX', 'CoordinateY', 'CoordinateZ'])
t = C.newPyTree(['Base',3]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/post/interiorFaces.py

```
# - interiorFaces (array) -
import Converter as C
import Post as P
import Generator as G

# Test faces interieures au sens large :
# faces ayant 2 voisins
a = G.cartTetra((0,0,0), (1,1.,1), (20,2,1))
b = P.interiorFaces(a)
C.convertArrays2File([a,b], 'out1.plt', 'bin_tp')

# Test faces interieures au sens strict :
# faces n'ayant que des noeuds interieurs
a = G.cartTetra((0,0,0), (1,1.,1), (20,3,1))
b = P.interiorFaces(a,1)
C.convertArrays2File([a,b], 'out2.plt', 'bin_tp')
```

Example file : Examples/post/interiorFacesPT.py

```
# - interiorFaces (pyTree) -
import Converter.PyTree as C
import Post.PyTree as P
import Generator.PyTree as G

a = G.cartTetra((0,0,0), (1,1.,1), (20,20,1))
b = P.interiorFaces(a)
t = C.newPyTree(['Base',1]); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/post/exteriorFaces.py

```
# - exteriorFaces (array) -
import Converter as C
import Post as P
import Generator as G

a = G.cartTetra((0,0,0), (1,1,1), (4,4,6))
b = P.exteriorFaces(a)
C.convertArrays2File([b], 'out.plt', 'bin_tp')
```

Example file : Examples/post/exteriorFacesPT.py

```
# - exteriorFaces (pyTree)-
import Converter.PyTree as C
import Post.PyTree as P
import Generator.PyTree as G

a = G.cartTetra((0,0,0), (1,1,1), (4,4,6)); b = P.exteriorFaces(a)
t = C.newPyTree(['Base',2]); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/post/mergeElts.py

```
# - mergeElts (array) -
import Post as P
import Converter as C
import Generator as G
import Transform as T

eps = 1.e-2; argqual = 0.25

# deraffinement de toutes les cellules d'un carre
ni = 21; nj = 21; nk = 11
hi = 2./(ni-1); hj = 2./(nj-1); hk = 1./(nk-1)
m = G.cart((0.,0.,0.),(hi,hj,hk), (ni,nj,nk))

hi = hi/2; hj = hj/2; hk = hk/2
m2 = G.cart((0.,0.,0.),(hi,hj,hk), (ni,nj,nk))
m2 = T.subzone(m2,(3,3,6),(m[2]-2,m[3]-2,6))
m2 = T.translate(m2, (0.75,0.75,0.25))
#m2 = T.rotate(m2, (0.2,0.2,0.), (0.,0.,1.), 15.)
m2 = T.perturbate(m2, 0.51)
tri = G.delaunay(m2)

npts = tri[2].shape[1]
indic = C.array('indic',npts,1,1)
indic = C.initVars(indic,'indic',1)

sol = P.mergeElts(tri, indic, argqual, eps)
C.convertArrays2File([tri,sol], 'out.plt')
```

Example file : Examples/post/mergeEltsPT.py

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

ni = 21; nj = 21; nk = 1
hi = 2./(ni-1); hj = 2./(nj-1)
m = G.cart((0.,0.,0.),(hi,hj,1.), (ni,nj,nk))
m = T.perturbate(m, 0.51)
tri = G.delaunay(m)
npts = tri[1][0][1]

import Converter as CA
indica = CA.array('indic',npts,1,1)
indica = CA.initVars(indica,'indic',1)
indic = C.convertArrays2ZoneNode('indicZ',[indica])

tri = C.addVars(tri,'Density'); tri = C.addVars(tri,'centers:cellN')
sol = P.mergeElts(tri, indic); sol[0] = 'cart2'
t = C.newPyTree(['Base',2]); t[1][2].append(tri); t[1][2].append(sol)
C.convertPyTree2File(t,"out.cgns","bin_cgns")
```

Example file : Examples/post/extractPoint.py

```
# - extractPoint (array) -
import Converter as C
import Generator as G
import Post as P

ni = 10; nj = 10 ; nk = 10;
a = G.cart((0,0,0), (1./(ni-1),1./(nj-1),1./(nk-1)), (ni,nj,nk))
def F(x,y,z):
```

```
    return x*x*x*x + 2.*y + z*z
a = C.initVars(a, 'F', F, ['x','y','z'])
val = P.extractPoint([a], (0.55, 0.38, 0.12), 2); print val
```

Example file : Examples/post/extractPointPT.py

```
# - extractPoint (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.PyTree as P

ni = 10; nj = 10 ; nk = 10;
a = G.cart((0,0,0), (1./(ni-1),1./(nj-1),1./(nk-1)), (ni,nj,nk))

def F(x,y,z):
    return x + 2.*y + 3.*z

a = C.initVars(a, 'F', F, ['CoordinateX','CoordinateY','CoordinateZ'])
val = P.extractPoint(a, (0.55, 0.38, 0.12), 2)
print val[0]
```

Example file : Examples/post/extractPlane.py

```
# - extractPlane (array) -
import Converter as C
import Post as P
import Transform as T
import Generator as G

m = G.cylinder((0,0,0), 1, 5, 0., 360., 10., (50,50,50))
m = T.rotate(m , (0,0,0), (1,0,0), 35.)

array = P.extractPlane([m], (0.5, 1., 0., 1), 2)
C.convertArrays2File([m,array], "out.plt", "bin_tp")
```

Example file : Examples/post/extractPlanePT.py

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

# Cree un cylindre et le met dans des arrays
m = G.cylinder((0,0,0), 1, 5, 0., 360., 10., (50,50,50))
m = T.rotate(m, (0,0,0), (1,0,0), 35.)
m = C.addVars(m,['Density','centers:cellN'])
m = C.initVars(m,'Density',1)
m = C.initVars(m,'centers:cellN',1)
z = P.extractPlane(m, (0.5, 1., 0., 1), 2)
t = C.newPyTree(['Base',2]); t[1][2].append(z)
C.convertPyTree2File(t, "out.cgns")
```

Example file : Examples/post/extractMesh.py

```
# - extractMesh (array) -
import Converter as C
import Post as P
import Generator as G

ni = 30; nj = 40; nk = 10
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
m = C.initVars(m, 'ro', 1.)

# Cree un maillage d'extraction
```

```
a = G.cart((0.,0.,0.), (1., 0.1, 0.1), (20, 20, 1))

# Extrait la solution sur le maillage d'extraction
a2 = P.extractMesh([m], a)
C.convertArrays2File([m,a2], "out.plt", "bin_tp")
```

Example file : Examples/post/extractMeshPT.py

```
# - extractMesh (pyTree) -
import Converter.PyTree as C
import Post.PyTree as P
import Generator.PyTree as G

ni = 30; nj = 40 ; nk = 10
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
m = C.initVars(m, 'Density', 2.)
m = C.initVars(m, 'cellN', 1)

# maillage d'extraction
a = G.cart((0.,0.,0.), (1., 0.1, 0.1), (20, 20, 1)); a[0] = 'extraction'

# Extrait la solution sur le maillage d'extraction
a = P.extractMesh(m, a)
t = C.newPyTree(['Solution',3,'Extraction',2])
t[1][2].append(m); t[2][2].append(a)
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/post/zipper.py

```
# - zipper (array) -
import Converter as C
import Post as P
import Generator as G
import Transform as T

# Cree un cylindre
m1 = G.cylinder((0,0,0), 1, 5, 0., 360., 10., (50,50,1))
m1 = C.initVars(m1, 'ro', 1.)
m1 = C.initVars(m1, 'rou', 1.)
m1 = C.initVars(m1, 'rov', 0.)
m1 = C.initVars(m1, 'row', 0.)
m1 = C.initVars(m1, 'roe', 1.)
m1 = C.initVars(m1, 'cellN', 1.)

# Set cellN = 2 (interpolated points) to boundary
s = T.subzone(m1, (1,m1[3],1),(m1[2],m1[3],m1[4]))
s = C.initVars(s, 'cellN', 2)
m1 = T.patch(m1, s, (1,m1[3],1))
s = T.subzone(m1, (1,1,1),(m1[2],1,m1[4]))
s = C.initVars(s, 'cellN', 2)
m1 = T.patch(m1, s, (1,1,1))

# Cree un carre
ni = 30; nj = 40
m2 = G.cart((0,0,0), (10./(ni-1),10./(nj-1),-1), (ni,nj,1))
m2 = C.initVars(m2, 'ro', 1.1)
m2 = C.initVars(m2, 'rou', 0.5)
m2 = C.initVars(m2, 'rov', 0.)
m2 = C.initVars(m2, 'row', 0.)
m2 = C.initVars(m2, 'roe', 1.)
m2 = C.initVars(m2, 'cellN', 1.)

array = P.zipper([m1,m2],[])
C.convertArrays2File([array], "out.plt","bin_tp")
```

Example file : Examples/post/usurp.py

```
# - usurp (array) -
import Post as P
import Converter as C
import Generator as G
import Transform as T

cyln = []
# Creation des cylindres
a1 = G.cylinder((0,0,0), 0, 2, 360, 0, 1., (100,2,10))
a1 = T.subzone(a1, (1,2,1), (a1[2],2,a1[4]))
cyln.append(a1)

a2 = G.cylinder((0,0,0), 0, 2, 90, 0, 0.5, (10,2,10))
a2 = T.translate(a2, (0,0,0.2))
a2 = T.subzone(a2, (1,2,1), (a2[2],2,a2[4]))
cyln.append(a2)

C.convertArrays2File(cyln, "out.plt", "bin_tp")

c1 = cyln[0]
ib1 = C.array('cellN', c1[2]-1, c1[3], c1[4]-1)
ib1 = C.initVars(ib1,'cellN', 1)
ib1[1][0,586] = 0.

c2 = cyln[1]
ib2 = C.array('cellN', c2[2]-1, c2[3], c2[4]-1)
ib2 = C.initVars(ib2, 'cellN', 1)

ibc = [ib1, ib2]

r = P.usurp(cyln, ibc)

cylc = C.node2Center(cyln)
out = []
l = len(cylc)
for i in range(l) :
    out.append(C.addVars([cylc[i], ibc[i]]))

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

Example file : Examples/post/usurpPT.py

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

# Creation des cylindres
a1 = G.cylinder((0,0,0), 0, 2, 360, 0, 1., (100,2,10))
a1 = T.subzone(a1, (1,2,1), (100,2,10)); a1[0]='cyl1'

a2 = G.cylinder((0,0,0), 0, 2, 90, 0, 0.5, (10,2,10))
a2 = T.translate(a2, (0,0,0.2))
a2 = T.subzone(a2, (1,2,1),(10,2,10)); a2[0]='cyl2'

# creation des celln
a1 = C.addVars(a1,'Density')
a1 = C.addVars(a1, 'centers:cellN'); a1 = C.initVars(a1, 'centers:cellN',1.)
a2 = C.addVars(a2, 'centers:cellN'); a2 = C.initVars(a2, 'centers:cellN',1.)
t = C.newPyTree(['Base',2])
t[1][2].append(a1); t[1][2].append(a2)
t = P.usurp(t)
C.convertPyTree2File(t, "out.cgns", "bin_cgns")
```

Example file : Examples/post/streamLine.py

```
# - streamLine (array) -
import Converter as C
import Post as P
import Generator as G
import math as M

ni = 30; nj = 40

# Maillage en noeuds structure
m1 = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,2))
m2 = G.cart((5.5,0,0), (9./(ni-1),9./(nj-1),1), (ni,nj,2))

def F(x):
    return M.cos(x)

c = C.array('ro,rou, rov,row,roE', ni, nj, 2)
c = C.initVars(c, 'ro', 1.)
c = C.initVars(c, 'rou', 1.)
c = C.initVars(c, 'rov', 0.)
c = C.initVars(c, 'row', 0.)
c = C.initVars(c, 'roE', 1.)
m1 = C.addVars([m1,c])
m2 = C.addVars([m2,c])
m1 = C.initVars(m1, 'rov', F, ['x'])
m2 = C.initVars(m2, 'rov', F, ['x'])

x0=0.1; y0=5.; z0=0.
p = P.streamLine([m1,m2], (x0,y0,z0))

# meme chose pour un maillage non structure
m3 = C.convertStruct2Tetra(m1)
m4 = C.convertStruct2Tetra(m2)
p2 = P.streamLine([m3,m4],(x0,y0,z0))

# mixte
p3 = P.streamLine([m1,m4],(x0,y0,z0))
C.convertArrays2File([p,p2,p3], "out.plt", "bin_tp")
```

Example file : Examples/post/streamLinePT.py

```
# - streamLine (pyTree) -
import Converter.PyTree as C
import Post.PyTree as P
import Generator.PyTree as G
import math as M

ni = 30; nj = 40; nk = 2
m1 = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk)); m1[0] = 'cart1'
m2 = G.cart((5.5,0,0), (9./(ni-1),9./(nj-1),1), (ni,nj,nk)); m2[0] = 'cart2'

def F(x):
    return M.cos(x)

vars = ['Density','MomentumX', 'MomentumY','MomentumZ','EnergyStagnationDensity']
for s in vars :
    m1 = C.addVars(m1, s)
    m2 = C.addVars(m2, s)

    if ( s == 'MomentumY' or s == 'MomentumZ' ) :
        m1 = C.initVars(m1,s,F, ['CoordinateX'])
        m2 = C.initVars(m2,s,F, ['CoordinateX'])
    elif ( s == 'MomentumZ' ) :
        m1 = C.initVars(m1,s,0.)
        m2 = C.initVars(m2,s,0.)
    else:
```

```
        m1 = C.initVars(m1,s,1.)
        m2 = C.initVars(m2,s,1.)

x0=0.1; y0=5.; z0=0.
t = C.newPyTree(['Base',3,'Line',1]); t[1][2].append(m1); t[1][2].append(m2)
s = P.streamLine(t, (x0,y0,z0));t[2][2].append(s)
C.convertPyTree2File(t, "out.cgns","bin_cgns")
```

Example file : Examples/post/streamRibbon.py

```
# - streamRibbon (array) -
import Converter as C
import Post as P
import Generator as G
import math as M

ni = 30; nj = 40
def F(x):
    return M.cos(x)

m1 = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,2))
m1 = C.initVars(m1, 'ro', 1.)
m1 = C.initVars(m1, 'rou', 1.)
m1 = C.initVars(m1, 'rov', 0.)
m1 = C.initVars(m1, 'row', 0.)
m1 = C.initVars(m1, 'roE', 1.)
m1 = C.initVars(m1, 'rov', F, ['x'])

ni = 40; nj = 50
m2 = G.cart((0.5,0,0), (9./(ni-1),9./(nj-1),1), (ni,nj,2))
m2 = C.initVars(m2, 'ro', 1.)
m2 = C.initVars(m2, 'rou', 1.)
m2 = C.initVars(m2, 'rov', 0.)
m2 = C.initVars(m2, 'row', 0.)
m2 = C.initVars(m2, 'roE', 1.)
m2 = C.initVars(m2, 'rov', F, ['x'])

ni = 40; nj = 40
m3 = G.cart((2.5,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,2))
m3 = C.initVars(m3, 'ro', 1.)
m3 = C.initVars(m3, 'rou', 1.)
m3 = C.initVars(m3, 'rov', 0.)
m3 = C.initVars(m3, 'row', 0.)
m3 = C.initVars(m3, 'roE', 1.)
m3 = C.initVars(m3, 'rov', F, ['x'])

x0=0.55; y0=5.; z0=0.
m = [m1,m2,m3]
p = P.streamRibbon(m,(x0,y0,z0),(0.,0.2,0.))

# test non structure
m2 = []
for i in m :
    m2.append(C.convertStruct2Tetra(i))

p2 = P.streamRibbon(m2,(x0,y0,z0),(0.,0.2,0.))
C.convertArrays2File(m+[p,p2], "out.plt", "bin_tp")
```

Example file : Examples/post/streamRibbonPT.py

```
# - streamRibbon (pyTree) -
import Converter.PyTree as C
import Post.PyTree as P
import Generator.PyTree as G
import math as M
```

```
def F(x):
    return M.cos(x)

vars=['Density','MomentumX','MomentumY','MomentumZ','EnergyStagnationDensity']

t = C.newPyTree(['Base',3,'Base2',2])
ni = 30; nj = 40
m1 = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,2)); m1[0]='cart1'
t[1][2].append(m1)
ni = 40; nj = 50
m2 = G.cart((0.5,0,0), (9./(ni-1),9./(nj-1),1), (ni,nj,2)); m2[0]='cart2'
t[1][2].append(m2)

for i in xrange(len(t[1][2])):
    for s in vars :
        t[1][2][i] = C.addVars(t[1][2][i], s)

        if ( s == 'MomentumY' or s == 'MomentumZ' ) :
            t[1][2][i] = C.initVars(t[1][2][i],s,F, ['CoordinateX'])
        elif ( s == 'MomentumZ' ) :
            t[1][2][i] = C.initVars(t[1][2][i],s,0.)
        else:
            t[1][2][i] = C.initVars(t[1][2][i],s,1.)

x0 = 1.; y0 = 2.; z0 = 0.
s = P.streamRibbon(t,(x0,y0,z0),(0.,0.2,0.));t[2][2].append(s)
C.convertPyTree2File(t, "out.cgns","bin_cgns")
```

Example file : Examples/post/integ.py

```
# - integ (array) -
import Converter as C
import Generator as G
import Post as P

# Maillage structure en noeuds
ni = 30 ; nj = 40
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,1))

# Champ a integrer en centres
c = C.array('vx', ni-1, nj-1, 1)
c = C.initVars(c, 'vx', 1.)
resc = P.integ([m], [c], [])[0] ; print resc

# Champ a integrer en noeuds
cn = C.array('vx', ni, nj, 1)
cn = C.initVars(cn, 'vx', 1.)
resn = P.integ([m], [cn], [])[0] ; print resn
```

Example file : Examples/post/integPT.py

```
# - integ (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.PyTree as P

ni = 30 ; nj = 40
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,1))
m = C.initVars(m, 'vx', 1.); m = C.initVars(m, 'ratio', 1.)
resn = P.integ(m)
print resn
```

Example file : Examples/post/integNorm.py

```
# - integNorm (array) -
import Converter as C
import Generator as G
import Post as P

# Maillage et champs non structure, en noeuds
m = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
c1 = C.array('ro', 100, 162, 'TRI')
c = C.initVars(c1, 'ro', 1.)
res = P.integNorm([m], [c], [])
print res

# Maillage en noeuds
ni = 30 ; nj = 40
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,1))

# Champ a integrer en centres
c1 = C.array('vx', ni-1, nj-1, 1)
c = C.initVars(c1, 'vx', 1.)

# Integration de chaque champ
res = P.integNorm([m], [c], [])
print res

# Champ a integrer en noeuds
c1 = C.array('vx, vy', ni, nj, 1)
cn = C.initVars(c1, 'vx', 1.)
cn = C.initVars(c1, 'vy', 1.)
resn = P.integNorm([m], [cn], [])
print resn
```

Example file : Examples/post/integNormPT.py

```
# - integNorm (pyTree)-
import Converter.PyTree as C
import Generator.PyTree as G
import Post.PyTree as P

# Maillage et champs non structure, en noeuds
m = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
m = C.addVars(m, 'Density') ; m = C.initVars(m, 'Density', 1.)
t = C.newPyTree(['Base',2]); t[1][2].append(m)
res = P.integNorm(t) ; print res
```

Example file : Examples/post/integNormProduct.py

```
# - integNormProduct (array) -
import Converter as C
import Generator as G
import Post as P

# Maillage et champs non structure, en noeuds
m = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
c = C.array('vx,vy,vz', 100, 162, 'TRI')
c = C.initVars(c, 'vx,vy,vz', 1.)
res = P.integNormProduct([m], [c], []) ; print res

# Maillage en noeuds
ni = 30 ; nj = 40
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,1))

# Champ a integrer en centres
c = C.array('vx,vy,vz', ni-1, nj-1, 1)
c = C.initVars(c, 'vx,vy,vz', 1.)
```

```
# Integration de chaque champ
res = P.integNormProduct([m], [c], []) ; print res

# Champ a integrer en noeuds
cn = C.array('vx,vy,vz', ni, nj, 1)
cn = C.initVars(cn, 'vx,vy,vz', 1.)
resn = P.integNormProduct([m], [cn], []) ; print resn
```

Example file : Examples/post/integNormProductPT.py

```
# - integNormProduct (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.PyTree as P

ni = 30 ; nj = 40; m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,1))
m = C.addVars(m,'MomentumX'); m = C.initVars(m,'MomentumX',1.);
m = C.addVars(m,'MomentumY'); m = C.initVars(m,'MomentumY',1.);
m = C.addVars(m,'MomentumZ'); m = C.initVars(m,'MomentumZ',1.);
res = P.integNormProduct(m) ; print res
```

Example file : Examples/post/integMoment.py

```
# - integMoment (array) -
import Converter as C
import Generator as G
import Post as P

# Maillage et champs non structure, en noeuds
m = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
c = C.array('vx,vy,vz', 100, 162, 'TRI')
c = C.initVars(c, 'vx,vy,vz', 1.)
res = P.integMoment([m], [c], [], (5.,5., 0.)) ; print res

# Maillage en noeuds
ni = 30 ; nj = 40
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,1))
C.convertArrays2File([m], "new.plt", "bin_tp")

# Champ a integrer en centres
c = C.array('vx,vy,vz', ni-1, nj-1, 1)
c = C.initVars(c, 'vx,vy,vz', 1.)

# Integration de chaque champ
res = P.integMoment([m], [c], [], (5.,5., 0.) ) ; print res

# Champ a integrer en noeuds
cn = C.array('vx,vy,vz', ni, nj, 1)
cn = C.initVars(cn, 'vx,vy,vz', 1.)
resn = P.integMoment([m], [cn], [], (5.,5., 0.)) ; print resn
```

Example file : Examples/post/integMomentPT.py

```
# - integMoment (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.PyTree as P

m = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
m = C.addVars(m,'vx'); m = C.initVars(m,'vx',1.)
m = C.addVars(m,'vy'); m = C.initVars(m,'vy',0.)
m = C.addVars(m,'vz'); m = C.initVars(m,'vz',0.)
res = P.integMoment(m, (5.,5., 0.)) ; print res
```

Example file : Examples/post/integMomentNorm.py

```
# - integMomentNorm (array) -
import Converter as C
import Generator as G
import Post as P

# Maillage et champs non structure, en noeuds
m = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
c = C.array('ro', 100, 162, 'TRI')
c = C.initVars(c, 'ro', 1.)
res = P.integMomentNorm([m], [c], [], (5.,5., 0.)) ; print res

# Maillage en noeuds
ni = 30 ; nj = 40
m = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,1))
C.convertArrays2File([m], "new.plt", "bin_tp")

# Champ a integrer en centres
c = C.array('v', ni-1, nj-1, 1)
c = C.initVars(c, 'v', 1.)

# Integration de chaque champ
res = P.integMomentNorm([m], [c], [], (5.,5., 0.) ) ; print res

# Champ a integrer en noeuds
cn = C.array('v', ni, nj, 1)
cn = C.initVars(cn, 'v', 1.)
resn = P.integMomentNorm([m], [cn], [], (5.,5., 0.))
print resn
```

Example file : Examples/post/integMomentNormPT.py

```
# - integMomentNorm (pyTree)-
import Converter.PyTree as C
import Generator.PyTree as G
import Post.PyTree as P

m = G.cartTetra((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
m = C.addVars(m, 'Density'); m = C.initVars(m, 'Density', 1.)
res = P.integMomentNorm(m, (5.,5., 0.)) ; print res
```