Converter Module V1.4
- User Guide -

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

September 23, 2009

1 Converter : data conversion module

1.1 Preamble

This module provides routines for data conversion.
This module manipulates two different data structures: the first one is called an **array**, the second one is called a **pyTree**.

- An **array** can be a **structured** array defined by **['x,y,z,...', a, ni, nj, nk]**, where ni, nj, nk are the dimension of the grid and a is a (nfld, nixnjxnk) numpy array containing data, or an **unstructured** array defined by **['x,y,z,...', a, c, 'ELTTYPE']**. For this last type of arrays, c is the elements-to-nodes connectivity and a is a numpy array of data. If a stores fields on nodes, 'ELTTYPE' can be **'NODE', 'BAR', 'TRI', 'QUAD', 'TETRA', 'PYRA', 'PENTA', 'HEXA', 'MIX'**. If a stores field on elements, 'ELTTYPE' can be **'NODE*', 'BAR*', 'TRI*', 'QUAD*', 'TETRA*', 'PYRA*', 'PENTA*', 'HEXA*', 'MIX*'**
  In this documentation, we note a or b an array, and A or B a list of arrays.

  Important note : For accessing numpy arrays in python, the first index corresponds to the variable number, the second index to the cell index. For instance : a[0,0] for an 'x,y,z' array is variable x of first cell, a[1,0] is variable y of first cell, and so on...

  To use the array interface:

```
import Converter as C
```

- A **pyTree** is a python/CGNS tree, that is a mapping of the CGNS standard in python. Each node of the tree is a python list defined by **['name', a, [...], 'CGNSType_t']**, where a is the value stored by this node (a can be a numpy array, a float64, an int32 or a string), and [...] designates a list of nodes that are the children of the current node.

In this case, in the following, we note a or b a zone node, and A or B a list of zone nodes or a complete python tree.

To use the pyTree interface:

```
import Converter.PyTree as C
```

1.2 Name standardisation

Some functions of Converter, Geom, Generator, Transform and Post modules perform specific treatments for given variables. Recognized names are CGNS names. For instance, the "computeVariables" function in the Post module can compute the pressure automatically if density and velocity are defined with their CGNS name).Nevertheless, alternative names are also recognised. Name are described in the following table:

Description	CGNS	Alternative names
Coordinate in x direction	CoordinateX	x, X
Coordinate in y direction	CoordinateY	y, Y
Coordinate in z direction	CoordinateZ	z, Z
Density	Density	ro
Momentum in x direction	MomentumX	rou, rovx
Momentum in y direction	MomentumY	rov, rovy
Momentum in z direction	MomentumZ	row, rovz
Density times total energy	EnergyStagnationDensity	roE
Density times turbulence kinetic energy	TurbulentEnergyKineticDensity	rok
Density times dissipation rate of turbulence kinetic energy	TurbulentDissipationDensity	roeps
Static pressure	Pressure	
Dynamic pressure	PressureDynamic	
Static temperature	Temperature	
Enthalpy	Enthalpy	
Entropy	Entropy	
Stagnation pressure	PressureStagnation	
Stagnation temperature	TemperatureStagnation	
x-component of the absolute velocity	VelocityX	
y-component of the absolute velocity	VelocityY	
z-component of the absolute velocity	VelocityZ	
Absolute velocity magnitude	VelocityMagnitude	
Absolute Mach number	Mach	
Molecular viscosity	ViscosityMolecular	
Cell Nature Field (0:blanked, 1:discretised, 2:interpolated)		cellN, cellnf
Cell Nature Field (0:blanked, 1:discretised, -Id:-interpolation block Id)		cellNF, cellnf, cellNF, ichim
Cell Status (-1:orphan, 0:blanked, 1:discretised, 2:interpolated explicitly, 3:extrapolated, 4:interpolated implicitly)		status

1.3 Array creation and manipulation

Create a structured array containing variables x,y,z on a nixnjxnk grid:

```
a = C.array('x,y,z', ni, nj, nk)
```

Create a unstructured array containing variables x,y,z on a grid containing np points and ne elements of type ELTTYPE:

```
a = C.array('x,y,z', np, ne, ELTTYPE)
```

(See : Examples/converter/array.py)

Return the list of values defined in array a for point of index ind (for both structured and unstructured arrays). For structured arrays, you can specify (i,j,k) instead of ind. For unstructured arrays, the index ind corresponds to the location type of point defining array a: for instance, if array a describes a field at element vertices, ind is a vertex index.

```
v = C.getValue(a, ind)
```

ind starts at 0 and (i,j,k) start at 1.

(See : Examples/converter/getValue.py)

Set the values of one point of index ind in array a. v must be a list corresponding to the variables stored in array a:

```
C.setValue(a, ind, v)
```

(See : Examples/converter/setValue.py)

Add variable(s) to an array. var can be a string name ('ro') or a list of string names (['ro', 'rou']):

```
b = C.addVars(a, 'ro') .or. B = C.addVars(A, 'ro')
```

(See : Examples/converter/addVar.py)

Concatenate array fields. Variables defined by a list of arrays are put in the same array:

```
f = C.addVars([a, b, c])
```

(See : Examples/converter/addVars.py)

Extract variables named 'x','y','ro' from an array. Variables can be designated by a string or a list of strings. Result in a new array:

```
b = C.extractVars(a, ['x','y','ro']) .or. B = C.extractVars(A, ['x','y','ro'])
```

(See : Examples/converter/extractVars.py) Note : var index starts at 1.

Copy an array (return a new duplicated array):

```
b = C.copy(a) .or. B = C.copy(A)
```

(See : Examples/converter/copya.py)

1.4 pyTree creation and manipulation

Create a pyTree with two bases ('Base1' and 'Base2'). Base1 will contain surface meshes (dim=2) and Base2 will contain volume meshes (dim=3):

```
A = C.newPyTree(['Base1', 2, 'Base2', 3])
```

(See : Examples/converter/newPyTree.py)

Add a base named 'baseName' to a pyTree. Third argument specifies the cell dimension (3 for a volume mesh, 2 for a surface mesh):

```
B = C.addBase2PyTree(A, baseName, 3)
```

(See : Examples/converter/addBase2PyTree.py)

Add a boundary condition to a structured zone of a python tree. The boundary of type bndType will be named bndName in the tree. The range specifies the window where the boundary is defined. It can be specified as [imin, imax, jmin, jmax, kmin, kmax] or as 'imin', 'jmin',... which means the full imin, jmin face of a structured block. bndType can be a CGNS name of a boundary condition, or 'BCMatch' for a matching BC, or 'BCOverlap' for an overlap BC, or 'FamilySpecified:FAMILY' for a BC specified in a family BC named FAMILY:

```
b = C.addBC2Zone(a, bndName, bndType, range)
```

For a matching BC, donor zone and donor range must be specified, final argument specifying the transformation is optional:

```
b = C.addBC2Zone(a, bndName, 'BCMatch', range, donorZone, donorRange, [1,2,3])
```

For an overlap BC, donor zones can be specified. If not, automatic computation of donor zones is used. Option can be 'doubly_defined' for a boundary condition also defined with a physical BC:

```
b = C.addBC2Zone(a, bndName, 'BCOverlap', range, donorZones, option)
```

(See : Examples/converter/addBC2Zone.py)

Fill empty boundary conditions of a structured zone with the given boundary condition. Parameter dim can be '2D' or '3D' (default is '3D'):

```
b = C.fillEmptyBCWith(a, bndName, bndType, dim) .or. B = C.fillEmptyBCWith(A, bndName, bndType, dim)
```

(See : Examples/converter/fillEmptyBCWith.py)

Remove all boundaries of a given type from a pyTree:

```
b = C.rmBCOfType(a, 'BCWall') .or. B = C.rmBCOfType(A, 'BCWall')
```

(See : Examples/converter/rmBCOfType.py)

Extract all boundaries of a given type from a pyTree, return a list of zones:

```
Z = C.extractBCOfType(a, 'BCWall') .or. Z = C.extractBCOfType(A, 'BCWall')
```

(See : Examples/converter/extractBCOfTypePT.py)

Check if some boundary conditions are undefined in a zone node, a list of zone nodes or a complete pyTree. Return a list of the indices [imin,imax,jmin,jmax,kmin,kmax] of undefined boundaries for all zones. Lists are empty ([[],...,[]]) if all the boundary conditions have been defined. Parameter dim can be '2D' or '3D' (default is '3D'):

```
wins = C.checkBC(a, dim) .or. wins = C.checkBC(A, dim)
```

(See : Examples/converter/checkBC.py)

Add a family of BCs to a base node of a tree:

```
B = C.addFamily2Base(A, familyName, bndType)
```

(See : Examples/converter/addFamily2Base.py)

Remove nodes named 'name' from a:

```
b = C.rmNodes(a, name) .or. B = C.rmNodes(A, name)
```

(See : Examples/converter/rmNodes.py)

Return the list of values defined in a zone a for point of index ind (for both structured and unstructured zones). For structured zones, you can specify (i,j,k) instead of ind. For unstructured zones, the index ind corresponds to the location type of point defining zone a: for instance, if a describes a field at element vertices, ind is a vertex index. var is the name of the variable:

```
v = C.getValue(a, var, ind)
```

ind starts at 0 and (i,j,k) start at 1.

(See : Examples/converter/getValuePT.py)

Set the values of one point of index ind in a zone a. var is the name of the variables to be modified, value can be a float or a list of floats corresponding to the values of the variables to be modified:

```
C.setValue(a, var, ind, value)
```

(See : Examples/converter/setValuePT.py)

Add a variable(s) to a zone. var is a string name or a list of string names (e.g. 'Density',...), variable localisation ('nodes' or 'centers') can be specified in var:

```
b = C.addVars(a, var) .or. B = C.addVars(A, var)
```

(See : Examples/converter/addVarsPT.py)

Remove a variable(s). var is a string name or a list of string names:

```
b = C.rmVars(a, var) .or. B = C.rmVars(A, var)
```

(See : Examples/converter/rmVars.py)

Copy a variable from zone a1 with name var1 to zone a2 with name var2. The var location must not change:

```
b = C.cpVars(a1, var1, a2, var2)
```

(See : Examples/converter/cpVars.py)

1.5 Array / pyTree common manipulation

Return the list of variable names contained in a:

```
v = C.getVarNames(a) .or. V = C.getVarNames(A)
```

(See : Examples/converter/getVarNames.py) (See : Examples/converter/getVarNamesPT.py)

Init variable given by a string to a constant value val:

```
b = C.initVars(a, 'cellN', val) .or. B = C.initVars(A, 'cellN', val)
```

(See : Examples/converter/initVars.py) (See : Examples/converter/initVarsPT.py)

Init a variable (named 'x') with a function f. Function f has two arguments here, named 'x','y':

```
b = C.initVars(a, 'x', f, ['x','y']) .or. B = C.initVars(A, 'x', f, ['x','y'])
```

(See : Examples/converter/initVar.py)

Create tetra unstructured array from an array. 2D elements are made triangular, else they are made tetrahedrical:

```
b = C.convertArray2Tetra(a) .or. B = C.convertArray2Tetra(A)
```

(See : Examples/converter/convertStruct2Tetra.py) (See : Examples/converter/convertArray2TetraPT.py)
(See : Examples/converter/convertHexa2Tetra.py) (See : Examples/converter/convertPrism2Tetra.py)

Create hexa unstructured array from an array. 2D elements are made quadrangular, else they are made hexahedrical:

```
b = C.convertArray2Hexa(a) .or. B = C.convertArray2Hexa(A)
```

(See : Examples/converter/convertStruct2Hexa.py)

Convert data (grid coordinates and solution) defined at nodes in a to centers:

```
b = C.node2Center(a) .or. B = C.node2Center(A)
```

When using the pyTree interface, a varname can be additionaly specified. Then, only the variable 'varname' is computed at centers and set in returned zone FlowSolution#Centers location:

```
b = C.node2Center(a, 'Density') .or. B = C.node2Center(A, 'Density')
```

(See : Examples/converter/node2center.py) (See : Examples/converter/node2centerPT.py)

Change data defined at centers in a to nodes:

```
b = C.center2Node(a) .or. B = C.center2Node(A)
```

When using the pyTree interface, a varname can be additionaly specified. Then, only the variable 'varname' is computed at nodes set in zone FlowSolution location:

```
b = C.center2Node(a, 'Density') .or. B = C.center2Node(A, 'Density')
```

(See : Examples/converter/center2Node.py) (See : Examples/converter/center2NodePT.py)

1.6 Array / PyTree analysis

Given a solution A and a solution B both defined on the same mesh, return the differences:

```
C = C.diffArrays(A, B)
```

(See : Examples/converter/diffArrays.py)

Return the field value where variable 'ro' is minimum:

```
min = C.getArgMin(a, 'ro') .or. min = C.getArgMin(A, 'ro')
```

(See : Examples/converter/getArgMin.py)

Return the field value where variable 'ro' is maximum:

```
max = C.getArgMax(a, 'ro') .or. max = C.getArgMax(A, 'ro')
```

(See : Examples/converter/getArgMax.py)

Return the min value of variable 'ro':

```
min = C.getMinValue(a, 'ro') .or. min = C.getMinValue(A, 'ro')
```

(See : Examples/converter/getMinValue.py)

Return the max value of variable 'ro':

```
max = C.getMaxValue(a, 'ro') .or. max = C.getMaxValue(A, 'ro')
```

(See : Examples/converter/getMaxValue.py)

Return the mean value of variable 'ro':

```
mean = C.getMeanValue(a, 'ro') .or. mean = C.getMeanValue(A, 'ro')
```

(See : Examples/converter/getMeanValue.py)

L0 and L2 norms of a field defined in an array can be extracted. If cellnature field is defined in the array, then blanked points are not taken into account into the computation of the norm:

```
L0norm = C.normL0(a, 'ro') .or. L0norm = C.normL0(A, 'ro')
```

(See : Examples/converter/normL0.py)

```
L2norm = C.normL2(a, 'ro') .or. L2norm = C.normL2(A, 'ro')
```

(See : Examples/converter/normL2.py)

Normalize a vector defined by its 3 vector coordinates. Modify sx,sy,sz:

```
b = C.normalize(a, ['sx', 'sy', 'sz']) .or. B = C.normalize(A, ['sx', 'sy', 'sz'])
```

(See : Examples/converter/normalize.py) (See : Examples/converter/normalizePT.py)

1.7 Array / pyTree conversion

Return the list of zone paths (strings) contained in a python tree:

```
P = C.convertPyTree2ZoneNames(T)
```

(See : Examples/converter/convertPyTree2ZoneNames.py)

Convert a python tree node to an array. One have to provide the path of the corresponding node and the python tree:

```
a = C.convertPyTree2Array("Zone-001/GridCoordinates/CoordinateX", T)
```

(See : Examples/converter/convertPyTree2Array.py)

1.8 File / arrays or pyTree conversion

Read a file and return a list of arrays:

```
A = C.convertFile2Arrays("file.plt", "bin_tp", options)
```

For formatted files, only one block of file is read.
For format needing multiple files (for ex : plot3d), multiple files can be specified in file string as: "file.gbin,file.qbin".
In file format where variables name are undefined, the following one is adopted : **x, y, z, ro, rou, rov, row, roE, cellN**.
Options can be set as a list of option pairs to specify the discretization of vector elements (as defined in xfig or svg):

Option name	**Meaning**	**Default value**
'nptsCurve'	Number of discretization points for curved vector elements	20
'nptsLine'	Number of discretization points for line vector elements	2

Write a list of arrays to a file:

```
C.convertArrays2File(A, "file.tp", "fmt_tp", options)
```

options is a list of option pairs. In each option pair, first argument is a string and second argument corresponds to its value. Option pairs can be:

Option name	**Meaning**	**Possible values**	**Default value**
'int'	size of integer	4, 8	4
'real'	size of real	4, 8	8
'endian'	endianess	'little', 'big'	'big'
'colormap'	colormap style	'active', 'inactive'	'inactive'

Read a file to a pyTree:

```
A = C.convertFile2PyTree('in.cgns', 'bin_cgns')
```

(See : Examples/converter/convertFile2PyTree.py)

Write a python tree to a file:

```
C.convertPyTree2File(A, 'out.cgns', 'bin_cgns')
```

(See : Examples/converter/convertPyTree2File.py)

Recognised formats are:

- bin_tp : binary tecplot file.
  (See : Examples/converter/conv.py)
- fmt_tp : formatted tecplot file.

(See : Examples/converter/conv2.py)

- bin_v3d : binary v3d file.

- fmt_v3d : formatted v3d file.
  (See : Examples/converter/conv3.py)
- bin_plot3d : binary plot3d file.
  (See : Examples/converter/convPlot3d.py)
- fmt_pov : formatted povray (raytracer) file.
  (See : Examples/converter/convPov.py)
- bin_df3 : binary density file for povray (raytracer).
  (See : Examples/converter/convDf3.py)
- fmt_mesh : formatted mesh (INRIA-Gamma) file.
  (See : Examples/converter/convMesh.py)
- bin_stl : binary STL file.
  (See : Examples/converter/convSTL.py)
- fmt_obj : formatted Obj file (wavefront).
  (See : Examples/converter/convObj.py)
- bin_3ds : binary 3D studio file (very limited).
  (See : Examples/converter/conv3DS.py)
- bin_pickle : binary python pickle format.
  (See : Examples/converter/convPickle.py)
- bin_wav : binary wav 8-bits sound file.
  (See : Examples/converter/convWav.py)
- fmt_xfig : formatted xfig file.
  (See : Examples/converter/convXfig.py)
- fmt_svg : formatted svg file.
  (See : Examples/converter/convSvg.py)
- bin_cgns : binary ADF or HDF CGNS file (for pyTrees only).

1.9 Cassiopée Objects / array conversion

To use conversion from/to Cassiopée objects, you need import:

```
import Converter.Cassiopee
```

Convert Cassiopée Mesh object to array:

```
a = C.Cassiopee.convertMesh2Array( msh )
```

(See : Examples/converter/convertMesh2Array.py)

Convert array to Cassiopée Mesh object:

```
C.Cassiopee.convertArray2Mesh( a, msh )
```

(See : Examples/converter/convertArray2Mesh.py)

Convert Cassiopée Window object to an array containing the coordinates of the nodes of the window:

```
a = C.Cassiopee.convertWindow2Array( win )
```

(See : Examples/converter/convertWindow2Array.py)

Convert a Cassiopée DesFunction to a python function. Available DesFunctions are 'wing_motion', 'rotor_motion' and 'transl_motion' (see Body Kinematics Userguide):

```
F = C.Cassiopee.convertDesFunction2Function( desFunction )
```

(See : Examples/converter/convertDesFunction2Function.py)

Convert a list of Cassiopée objects to a string corresponding to the name of the objects :

```
string = C.Cassiopee.convertDesObjs2String( desObjs )
```

(See : Examples/converter/convertWins2String.py)

1.10 elsA objects / array conversion

To use conversion from/to elsA objects, you need to use elsA executable to execute script and import:

```
import Converter.Elsa
```

Convert elsA Mesh object to array:

```
a = C.Elsa.convertMesh2Array( msh )
```

(See : Examples/converter/convertEMesh2Array.py)

Convert array to elsA Mesh object:

```
C.Elsa.convertArray2Mesh( a, msh )
```

(See : Examples/converter/convertArray2EMesh.py)

Convert elsA Window object to an array containing the coordinates of the nodes of the window:

```
a = C.Elsa.convertWindow2Array( win )
```

(See : Examples/converter/convertEWindow2Array.py)

Convert an elsA DesFunction to a python function. Available DesFunctions are 'wing_motion' and 'rotor_motion'.

```
F = C.Elsa.convertDesFunction2Function( desFunction )
```

(See : Examples/converter/convertEDesFunction2Func.py)

1.11 Converter arrays / 3D arrays conversion

In some applications, arrays must be seen as 3D arrays, that is (ni,nj,nk) numpy arrays instead of (nfld, ni*nj*nk) arrays. An 3D array is defined as [['x','y',...],[ax, ay, ...]] where ax is a (ni,nj,nk) numpy array corresponding to variable x, and so on...

Convert arrays to 3D arrays (ni,nj,nk):

```
B = C.Array3D.convertArrays2Arrays3D(A)
```

(See : Examples/converter/convertArray2Array3D.py)

Convert 3D arrays to arrays:

```
B = C.Array3D.convertArrays3D2Arrays(A)
```

(See : Examples/converter/convertArray3D2Array.py)

1.12 More general examples of use

- See : Examples/converter/restart.py
- See : Examples/converter/conv8.py
- See : Examples/converter/shell.py

1.13 Example files

Example file : Examples/converter/array.py

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

# Structured
b = C.array('x,y,z', 12, 9, 12) ; print b

# Unstructured
a = C.array('x,y,z', 12, 9, 'QUAD') ; print a
```

Example file : Examples/converter/getValue.py

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

# Array structure
Ni = 40; Nj = 50; Nk = 20
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1./(Nk-1)), (Ni,Nj,Nk))
# Les variables contenues dans a (x,y,z) au point (10,1,1)
print C.getValue( a, (10,1,1) )
print C.getValue( a, 9 ) # C'est le meme point

# Array non structure
Ni = 40; Nj = 50; Nk = 20
a = G.cartTetra((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1./(Nk-1)), (Ni,Nj,Nk))
print C.getValue( a, 9 )
```

Example file : Examples/converter/setValue.py

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

a = G.cart( (0,0,0), (1,1,1), (5,5,1) )
# Set point (1,1,1) with value x=0.1, y =0.1, z=1.
C.setValue(a, (1,1,1), [0.1,0.1,1.]) ; print a
```

Example file : Examples/converter/addVar.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,11))

# add a variable defined by a string (structured)
a = C.addVars(a, 'ro')
a = C.addVars(a, 'cellN')
C.convertArrays2File([a], "out1.plt", "bin_tp")

# add variables defined by a list of varNames
a = C.addVars(a, ['rou','rov'])
C.convertArrays2File([a], "out2.plt", "bin_tp")
```

Example file : Examples/converter/addVars.py

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

a = G.cart( (0,0,0), (1,1,1), (10,10,11) )
b = C.array('cell', a[2], a[3], a[4])
c = C.array('t,u', a[2], a[3], a[4])
f = C.addVars([a, b, c])

C.convertArrays2File([f], "out.plt", "bin_tp")
```

Example file : Examples/converter/extractVars.py

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

a = G.cart( (0,0,0), (1,1,1), (10,10,10) )
r = C.extractVars(a, ['x','y'])
C.convertArrays2File([r], "out.plt", "bin_tp")
```

Example file : Examples/converter/copya.py

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

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

Example file : Examples/converter/newPyTree.py

```
# - newPyTree (pyTree) -
import Converter.PyTree as C

t = C.newPyTree(['Base1', 2, 'Base2', 3]); print t
```

Example file : Examples/converter/addBase2PyTree.py

```
# - addBase2PyTree (pyTree) -
import Converter.PyTree as C

t = C.newPyTree(['Base', 3])
t = C.addBase2PyTree(t, 'Base2', 2) ; print t
```

Example file : Examples/converter/addBC2Zone.py

```
# - addBC2Zone (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

a = G.cylinder((0,0,0), 1., 1.5, 0., 360., 1., (80,30,2))
b = G.cart((-0.1,0.9,0), (0.01,0.01,1.), (20,20,2))

# Physical BC (here BCWall)
a = C.addBC2Zone(a, 'wall1', 'BCWall', 'jmin')
# Matching BC
a = C.addBC2Zone(a, 'match1', 'BCMatch', 'imin', a, 'imax', [1,2,3])
a = C.addBC2Zone(a, 'match2', 'BCMatch', 'imax', a, 'imin', [1,2,3])
# Overlap BC (with automatic donor zones)
a = C.addBC2Zone(a, 'overlap1', 'BCOverlap', 'jmax')
# Overlap BC (with given donor zones and doubly defined)
a = C.addBC2Zone(a, 'overlap2', 'BCOverlap', 'jmin', [b], 'doubly_defined')

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

Example file : Examples/converter/fillEmptyBCWith.py

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

a = G.cart((0,0,0),(1,1,1),(10,10,10))
a = C.addBC2Zone(a, 'overlap', 'BCOverlap', 'imin')
a = C.addBC2Zone(a, 'match1', 'BCMatch', 'jmin', a, 'jmax', [1,2,3])
a = C.fillEmptyBCWith(a, 'wall', 'BCWall', '3D')

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

Example file : Examples/converter/rmBCOfType.py

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

a = G.cylinder((0,0,0), 1., 1.5, 0., 360., 1., (80,30,2))
b = G.cart((-0.1,0.9,0), (0.01,0.01,1.), (20,20,2))

a = C.addBC2Zone(a, 'wall1', 'BCWall', 'jmin')
a = C.addBC2Zone(a, 'match1', 'BCMatch', 'imin', a, 'imax', [1,2,3])
a = C.addBC2Zone(a, 'match2', 'BCMatch', 'imax', a, 'imin', [1,2,3])
a = C.addBC2Zone(a, 'overlap1', 'BCOverlap', 'jmax')
b = C.addBC2Zone(b, 'wall1', 'BCWall', 'imin')
t = C.newPyTree(['Base']) ; t[1][2] = t[1][2] + [a,b]

t = C.rmBCOfType(t, 'BCWall')
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/extractBCOfTypePT.py

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

a = G.cylinder((0,0,0), 1., 1.5, 360., 0., 1., (100,30,10))
a = C.addBC2Zone(a, 'wall1', 'BCWall', 'jmin')
Z = C.extractBCOfType(a, 'BCWall')
t = C.newPyTree(['Base',3,'Skin',2]); t[1][2].append(a); t[2][2] = t[2][2]+Z
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/checkBC.py

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

a1 = G.cart((0.,0.,0.), (0.1, 0.1, 0.1), (11, 21, 2)); a1[0] = 'cart1'
a1 = C.addBC2Zone(a1, 'wall1', 'BCWall', 'imin')
a2 = G.cart((1., 0.2, 0.), (0.1, 0.1, 0.1), (11, 21, 2)); a2[0] = 'cart2'
a2 = C.addBC2Zone(a2, 'wall1', 'BCWall', 'imax')
t = C.newPyTree(['Base']); t[1][2] = t[1][2] + [a1,a2]
t = X.connectMatch(t)

wins = C.checkBC(t,'2D') ; print wins
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/addFamily2Base.py

```
# - addFamily2Base (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

a = G.cylinder((0,0,0), 1., 1.5, 0., 360., 1., (50,20,20))
t = C.newPyTree(['Base'])

# Add family to tree
t[1] = C.addFamily2Base(t[1], 'flap', 'BCWall')
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/rmNodes.py

```
# - rmNodes (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
a = G.cart((0,0,0),(1,1,1),(10,10,10))
a = C.addVars(a, ['Density', 'centers:cellN', 'rou', 'rov', 'Hx', 'Hy'])
b = C.rmNodes(a, 'FlowSolution#Centers')

t = C.newPyTree(['Base']); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/getValuePT.py

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

# Array structure
Ni = 40; Nj = 50; Nk = 20
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1./(Nk-1)), (Ni,Nj,Nk))
# Les variables contenues dans a (x,y,z) au point (10,1,1)
print C.getValue( a, 'CoordinateX', (10,1,1) )
print C.getValue( a, 'CoordinateX', 9 ) # C'est le meme point
print C.getValue( a, 'nodes:CoordinateX', 9 ) # C'est le meme point
```

```
print C.getValue( a, 'GridCoordinates', 9 ) # retourne (x,y,z)

# Array non structure
Ni = 40; Nj = 50; Nk = 20
a = G.cartTetra((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1./(Nk-1)), (Ni,Nj,Nk))
print C.getValue( a, 'CoordinateX', 9 )
```

Example file : Examples/converter/setValuePT.py

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

# Array structure
Ni = 40; Nj = 50; Nk = 20
a = G.cart((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1./(Nk-1)), (Ni,Nj,Nk))
C.setValue(a, 'CoordinateX', (10,1,1), 0.25)
C.setValue(a, 'GridCoordinates', (11,1,1), [0.3,0.2,0.1]); print a

# Array non structure
Ni = 40; Nj = 50; Nk = 20
a = G.cartTetra((0,0,0), (1./(Ni-1), 0.5/(Nj-1),1./(Nk-1)), (Ni,Nj,Nk))
C.setValue(a, 'CoordinateX', 9, 0.1 ); print a
```

Example file : Examples/converter/addVarsPT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,11))
a = C.addVars(a, 'rou')
a = C.addVars(a, 'centers:cellN')
a = C.addVars(a, ['Density', 'Hx', 'centers:Hy'])
t = C.newPyTree(['Base']) ; t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/rmVars.py

```
# - rmVars (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
a = G.cart((0,0,0),(1,1,1),(10,10,10))
a = C.addVars(a, ['Density', 'centers:cellN', 'rou', 'rov', 'Hx', 'Hy'])
b = C.rmVars(a, 'Density')
b = C.rmVars(a, ['Hx', 'Hy'])
b = C.rmVars(b, 'FlowSolution#Centers')

t = C.newPyTree(['Base']); t[1][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/cpVars.py

```
# - cpVars (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
a = G.cart((0,0,0),(1,1,1),(10,10,10)) ; a[0] = 'cart1'
b = G.cart((0,0,0),(1,1,1),(10,10,10)) ; b[0] = 'cart2'
a = C.addVars(a, 'Density')
a = C.cpVars(a, 'Density', a, 'Density2')
c = C.cpVars(a, 'Density', b, 'Density')
t = C.newPyTree(['Base']); t[1][2] = t[1][2] + [a,c]
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/getVarNames.py

```
# - getVarNames (array) -
import Converter as C
a = C.array('x,y,z,ro', 12, 9, 12)
print C.getVarNames(a)
```

Example file : Examples/converter/getVarNamesPT.py

```
# - getVarNames (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
a = G.cart((0,0,0),(1,1,1),(10,10,10))
a = C.addVars(a, ['Density', 'centers:cellN'])
print C.getVarNames(a)
```

Example file : Examples/converter/initVars.py

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

a = C.array("x,y,z, ichim, celln", 10, 10, 10)
a = C.initVars(a, 'celln', 2.) ; print a
```

Example file : Examples/converter/initVarsPT.py

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

a = G.cart((0,0,0), (1,1,1), (10,10,10))
a = C.addVars(a, 'centers:celln')
a = C.addVars(a, 'Density')
a = C.initVars(a, 'centers:celln', 2.)

# Create a function
def F(x1, x2):
    return 3.*x1+2.*x2
a = C.initVars(a, 'Density', F, ['CoordinateX','CoordinateY'])

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

Example file : Examples/converter/initVar.py

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

# Create a function
def F(x1, x2):
    return 3.*x1+2.*x2

# Create a cartesian grid
a = G.cart( (0,0,0), (1,1,1), (11,11,1))

# init by function
a = C.initVars(a, 'F', F, ['x','y'])

C.convertArrays2File([a], "out.plt", "bin_tp")
```

Example file : Examples/converter/convertStruct2Tetra.py

```
# - convertArray2Tetra (array) -
import Converter as C
import Generator as G

# 2D : triangles
```

```
a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
b = C.convertArray2Tetra(a)
C.convertArrays2File([b], 'new1.plt', 'bin_tp')

# 3D : tetraedres
a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,10))
b = C.convertArray2Tetra(a)
C.convertArrays2File([b], 'new2.plt', 'bin_tp')
```

Example file : Examples/converter/convertArray2TetraPT.py

```
# - convertArray2Tetra (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

t = C.newPyTree(['Base1',2,'Base2',3])

# 2D : triangles
a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
b = C.convertArray2Tetra(a) ; t[1][2].append(b)

# 3D : tetraedres
a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,10))
b = C.convertArray2Tetra(a) ; t[2][2].append(b)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/convertHexa2Tetra.py

```
# - convertArray2Tetra (array) -
import Converter as C
import Generator as G

# 2D : quads -> triangles
a = G.cartHexa((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
b = C.convertArray2Tetra(a)
C.convertArrays2File([b], 'new1.plt', 'bin_tp')

# 3D : hexa->tetraedres
a = G.cartHexa((0.,0.,0.), (0.1,0.1,0.2), (10,10,10))
b = C.convertArray2Tetra(a)
C.convertArrays2File([b], 'new2.plt', 'bin_tp')
```

Example file : Examples/converter/convertPrism2Tetra.py

```
# - convertArray2Tetra (array) -
import Converter as C
import Generator as G

a = G.cartPenta((0.,0.,0.), (1,1,1), (10,10,3))
a = C.convertArray2Tetra(a)
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file : Examples/converter/convertStruct2Hexa.py

```
# - convertArray2Hexa (array) -
import Converter as C
import Generator as G

# 2D : quad
a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,1))
b = C.convertArray2Hexa(a)
C.convertArrays2File([b], 'new1.plt', 'bin_tp')

# 3D : hexaedres
a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,10,10))
b = C.convertArray2Hexa(a)
C.convertArrays2File([b], 'new2.plt', 'bin_tp')
```

Example file : Examples/converter/node2center.py

```
# - node2Center (array) -
import Converter as C
import Generator as G

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

ni = 30 ; nj = 40 ; nk = 1
a = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
a = C.addVars(a, 'ro')
a = C.initVars(a, 'ro', F, ['x','y'])
C.convertArrays2File([a], "node.plt", "bin_tp")

ac = C.node2Center(a)
C.convertArrays2File([ac], "center.plt", "bin_tp")
```

Example file : Examples/converter/node2centerPT.py

```
# - node2Center (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

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

ni = 30 ; nj = 40 ; nk = 3
a = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
a = C.initVars(a, 'Density', F, ['CoordinateX','CoordinateY'])

# node2Center : passe une variable en centres (dans la meme zone)
a = C.node2Center(a, 'Density')
t = C.newPyTree(['Base',3]); t[1][2].append(a)
#C.convertPyTree2File(t, "out.cgns", "bin_cgns")

# node2Center : cree une nouvelle zone contenant les centres
a = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
a = C.initVars(a, 'Density', F, ['CoordinateX','CoordinateY'])
b = C.node2Center(a) ; b[0] = a[0]+'_centers'
t = C.newPyTree(['Base',3]); t[1][2] = t[1][2] + [a,b]
C.convertPyTree2File(t, 'out.cgns')
```

Example file : Examples/converter/center2Node.py

```
# - center2Node (array) -
import Converter as C
import Generator as G

ni = 30 ; nj = 40 ; nk = 10
a = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
a = C.initVars(a, 'ro', 1.)
an = C.center2Node(a)
C.convertArrays2File([an], "out.plt", "bin_tp")
```

Example file : Examples/converter/center2NodePT.py

```
# - center2Node (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G

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

# center2Node : cree une nouvelle zone
ni = 30 ; nj = 40 ; nk = 2
```

```
a = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
a = C.initVars(a, 'centers:Density', 1.)
b = C.center2Node(a) ; b[0] = a[0]+'_nodes'
t = C.newPyTree(['Base1',3]); t[1][2].append(b)
C.convertPyTree2File(t, "out.cgns")

# center2Node : modifie une variable
ni = 30 ; nj = 40 ; nk = 3
a = G.cart((0,0,0), (10./(ni-1),10./(nj-1),1), (ni,nj,nk))
a = C.initVars(a, 'centers:Density', 1.)
a = C.center2Node(a, 'Density')

t = C.newPyTree(['Base',3]); t[1][2].append(a)
#C.convertPyTree2File(t, "out.cgns","bin_cgns")
```

Example file : Examples/converter/diffArrays.py

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

ni = 11 ; nj = 11 ; nk = 11
a = G.cart( (0,0,0), (1,1,1), (ni,nj,nk) )
a = C.addVars(a, "F")
a = C.initVars(a, "F", 1.)
a = C.addVars(a, "Q")
a = C.initVars(a, "Q", 1.2)

b = G.cart( (0,0,0), (1,1,1), (ni,nj,nk) )
b = C.addVars(b, "Q")
b = C.initVars(b, "Q", 2.)
b = C.addVars(b, "F")
b = C.initVars(b, "F", 3.)

ret = C.diffArrays([a], [b]) ; print ret
```

Example file : Examples/converter/getArgMin.py

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

a = G.cart( (0,0,0), (1.,1.,1.), (10,10,10) )
argmin = C.getArgMin(a, 'x') ; print argmin
```

Example file : Examples/converter/getArgMax.py

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

a = G.cart( (0,0,0), (1.,1.,1.), (10,10,10) )
argmax = C.getArgMax(a, 'x') ; print argmax
```

Example file : Examples/converter/getMinValue.py

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

a = G.cart( (0,0,0), (1.,1.,1.), (11,1,1) )
minval = C.getMinValue(a, 'x') ; print minval
```

Example file : Examples/converter/getMaxValue.py

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

a = G.cart( (0,0,0), (1.,1.,1.), (11,1,1) )
maxval = C.getMaxValue(a, 'x') ; print maxval
```

Example file : Examples/converter/getMeanValue.py

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

a = G.cart( (0,0,0), (1.,1.,1.), (11,1,1) )
meanval = C.getMeanValue(a, 'x') ; print meanval
```

Example file : Examples/converter/normL0.py

```
# - normL0 (array) -
import Converter as C
import Generator as G

a = G.cart( (0,0,0), (1,1,1), (11,11,11) )
a = C.initVars(a, "F", 1.)
print 'normL0 = ', C.normL0(a, "F")
```

Example file : Examples/converter/normL2.py

```
# - normL2 (array) -
import Converter as C
import Generator as G

ni = 11 ; nj = 11 ; nk = 11
a = G.cart( (0,0,0), (1,1,1), (ni,nj,nk) )
a = C.initVars(a, "F", 1.)
print 'normL2 = ', C.normL2(a, "F")

# La variable cellN est prise en compte
cellnf = C.array('celln', ni, nj, nk)
cellnf = C.initVars(cellnf, "celln", 1.)

cellnf[1][0][1] = 0.
cellnf[1][0][2] = 0.

a = C.addVars([a, cellnf])
print 'normL2 = ', C.normL2(a, "F")
```

Example file : Examples/converter/normalize.py

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

a = D.sphere( (0,0,0), 1., 50 )
n = G.getNormalMap(a)
n = C.center2Node(n)
n[1] = n[1]*10
n = C.normalize(n, ['sx','sy','sz'])
a = C.addVars([a, n])
C.convertArrays2File([a], 'out.plt', 'bin_tp')
```

Example file : Examples/converter/normalizePT.py

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

a = D.sphere( (0,0,0), 1., 50 )
a = C.addVars(a,'sx'); a = C.initVars( a, 'sx', 10.)
a = C.addVars(a,'sy'); a = C.initVars( a, 'sy', 20.)
a = C.addVars(a,'sz'); a = C.initVars( a, 'sz', 30.)
a = C.normalize(a, ['sx','sy','sz'])
t = C.newPyTree(['Base',2]); t[1][2].append(a)
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
```

Example file : Examples/converter/convertPyTree2ZoneNames.py

```
# - convertPyTree2Array (pyTree) -
import Converter.PyTree as C

t = C.convertFile2PyTree('SquaredNozzle-06-R.cgns', 'bin_cgns')
# Noms des zones contenues dans l'arbre t
zones = C.convertPyTree2ZoneNames(t) ; print zones
```

Example file : Examples/converter/convertPyTree2Array.py

```
# - convertPyTree2Array (pyTree) -
import Converter.PyTree as C
import Converter as C1

t = C.convertFile2PyTree('SquaredNozzle-06-R.cgns', 'bin_cgns')
zones = C.convertPyTree2ZoneNames(t)

# 1 - Get one field
arrays = []
for i in zones:
    a = C.convertPyTree2Array(i+"/GridCoordinates/CoordinateX", t)
    b = C.convertPyTree2Array(i+"/GridCoordinates/CoordinateY", t)
    c = C.convertPyTree2Array(i+"/GridCoordinates/CoordinateZ", t)
    x = C1.addVars([a,b,c])
    arrays.append(x)
C1.convertArrays2File(arrays, "out.plt", "bin_tp")

# 2 - Get a global field
arrays = []
for i in zones:
    a = C.convertPyTree2Array(i+"/GridCoordinates", t)
    arrays.append(a)
C1.convertArrays2File(arrays, "out2.plt", "bin_tp")

# 3 - Get the window of a boundary
a = C.convertPyTree2Array('SquaredNozzle/Zone-001/ZoneBC/Symmetry:South/PointRange', t)
print a
```

Example file : Examples/converter/convertFile2PyTree.py

```
# - convertFile2PyTree (pyTree) -
import Converter.PyTree as C

t1 = C.convertFile2PyTree('SquaredNozzle-06-R.cgns', 'bin_cgns')
t2 = C.convertFile2PyTree('in.plt', 'bin_tp')
```

Example file : Examples/converter/convertPyTree2File.py

```
# - convertPyTree2File (pyTree) -
import Converter.PyTree as C

t = C.convertFile2PyTree('SquaredNozzle-06-R.cgns', 'bin_cgns')
C.convertPyTree2File(t, 'out.cgns', 'bin_cgns')
C.convertPyTree2File(t, 'out.plt', 'bin_tp')
```

Example file : Examples/converter/conv.py

```
# - convertFile2Arrays (binary tecplot) -
# - convertArrays2File (binary tecplot) -
# Lecture d'un fichier sous forme d'un arrays
# selection de certains blocs dans le arrays
# ecriture du nouvel arrays
import Converter as C

# Lit le fichier in.plt dans arrays
arrays = C.convertFile2Arrays("in.plt", "bin_tp")
n = len(arrays)
print 'number of blocks in file : ', n

# arrays est une liste de n array.

# On affiche les dimensions du premier bloc : arrays[0]
print 'dimensions : ', arrays[0][2], arrays[0][3], arrays[0][4]

# On affiche la liste des variables du premier bloc : arrays[0]
print 'variables : ', arrays[0][0]

# On affiche les coord. du premier point du premier bloc.
# ! : Note that index is not natural for Fortran programmers
print 'x,y,z of first element : ',\
      arrays[0][1][0,0], arrays[0][1][1,0],  arrays[0][1][2,0]

# Prend les blocs 1 a 3 (le premier bloc est 0) et les met dans une liste
# appelee res
res = arrays[1:4]

# Ajoute le bloc 5 a la fin de la liste
res.append(arrays[5])

# Insert le bloc 6 au debut de la liste
res.insert(0, arrays[6])

# Sauvegarde la liste au format bin_tp
C.convertArrays2File(res, "out.plt", "bin_tp")
```

Example file : Examples/converter/conv2.py

```
# - convertFile2Arrays (formated tecplot) -
# - convertArrays2File (binary tecplot) -
import Converter as C

# Read a file into arrays
arrays = C.convertFile2Arrays("in360.tp", "fmt_tp")

# Write arrays to file
C.convertArrays2File(arrays, "out.plt", "bin_tp")
```

Example file : Examples/converter/conv3.py

```
# - convertFile2Arrays (binaire v3d) -
# - convertArrays2File (formated and binary v3d) -
import Converter as C

# Read a file into arrays
arrays = C.convertFile2Arrays("infmt.v3d", "fmt_v3d")

# Write a binary file with endian conversion
C.convertArrays2File(arrays, "out.v3d", "bin_v3d", ['endian','big'])
```

Example file : Examples/converter/convPlot3d.py

```
# - convertFile2Arrays (binary plot3d) -
# - convertArrays2File (binary plot3d) -
import Converter as C

# Read a file into arrays
arrays = C.convertFile2Arrays("in.dat", "bin_plot3d")
arrays = C.addVars(arrays, ["ro", "rou", "rov", "row", "roE"])

# Write arrays to file
C.convertArrays2File(arrays, "out.dat", "bin_plot3d", ['int',8])
C.convertArrays2File(arrays, "out.plt", "bin_tp")
```

Example file : Examples/converter/convPov.py

```
# - convertArrays2File (fmt_pov) -
import Converter as C

a = C.convertFile2Arrays("dauphin_skin.plt", "bin_tp")

# fmt_pov only supports triangle meshes
b = C.convertArray2Tetra(a)

# This converts coordinates and Density field to fmt_pov
# using colormap 1 (c1). c0 means no color pigment.
C.convertArrays2File(b, "out.pov", "fmt_pov", ['colormap',1])

# Reread this file
a = C.convertFile2Arrays("out.pov", "fmt_pov")
C.convertArrays2File(a, "out.plt", "bin_tp")

# Execute povray
import os ; os.system("povray -W800 -H600 +a0.3 +SP16 render.pov +P")
```

Example file : Examples/converter/convDf3.py

```
# - convertArrays2File (bin_df3) -
import Converter as C
import Generator as G

a = G.cart( (0,0,0), (1,1,1), (11,11,11) )
a = C.addVar(a, 'ro')

# Convert density field to povray density file
C.convertArrays2File([a], 'out.df3', 'bin_df3')
```

Example file : Examples/converter/convMesh.py

```
# - convertArrays2File (INRIA mesh format) -
import Converter as C

# Read mesh file
a = C.convertFile2Arrays("falcon.mesh", "fmt_mesh")
C.convertArrays2File(a, "out.plt", "bin_tp")

# Rewrite mesh file
C.convertArrays2File(a, "out.mesh", "fmt_mesh")
```

Example file : Examples/converter/convSTL.py

```
# - convertFile2Arrays (binary STL) -
import Converter as C

a = C.convertFile2Arrays('Data/piece-a-percer.stl', 'bin_stl')
C.convertArrays2File(a, 'out.plt', 'bin_tp')
```

Example file : Examples/converter/convObj.py

```
# - convertFile2Arrays (formatted Obj) -
import Converter as C

a = C.convertFile2Arrays('Data/cube.obj', 'fmt_obj')
C.convertArrays2File(a, 'out.plt', 'bin_tp')
```

Example file : Examples/converter/conv3DS.py

```
# - convertFile2Arrays (binary 3DS) -
import Converter as C

a = C.convertFile2Arrays('Data/box.3ds', 'bin_3ds')
C.convertArrays2File(a, 'out.plt', 'bin_tp')
```

Example file : Examples/converter/convPickle.py

```
# - convertFile2Arrays (binary python pickle) -
import Converter as C
import Generator as G

a = G.cart((0,0,0), (1,1,1), (100,100,100))
C.convertArrays2File([a], 'out.pickle', 'bin_pickle')
b = C.convertFile2Arrays('out.pickle', 'bin_pickle')
```

Example file : Examples/converter/convWav.py

```
# - convertArrays2File (wav) -
import Converter as C
import math

# Pas de temps d'echantillonage (s)
Deltat = 5.e-5

# Nombre d'echantillons
N = 500000

# Length
L = N*Deltat
print 'Total sound duration : ', L

# Sound frequency (Hertz)
# Oreille humaine entre 20 Hz et 20 kHz
f1 = 1000
f2 = 1100
f3 = 1200

# Signal
def F(time):
    if (time < L/3.):
        return math.cos(2*math.pi*f1*time)
    elif (time < 2.*L/3.):
        return math.cos(2*math.pi*f2*time)
    else:
        return math.cos(2*math.pi*f3*time)

a = C.array('Time, Pressure', N, 1, 1)

# Time
for i in xrange(N):
    a[1][0,i] = Deltat*i

# Pressure
a = C.initVars(a, 'Pressure', F, ['Time'])
```

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

# Convert in wav uses Time and Pressure field
C.convertArrays2File([a], 'out.wav', 'bin_wav')
```

Example file : Examples/converter/convXfig.py

```
# - convertArrays2File (fmt_xfig) -
import Converter as C
import Generator as G

a = C.convertFile2Arrays('test.fig', 'fmt_xfig',['Npts',100])
C.convertArrays2File(a, 'out.plt', 'bin_tp')
C.convertArrays2File(a, 'out.fig', 'fmt_xfig')

a = G.cart( (0,0,0), (1,1,1), (3, 3, 3) )
b = G.cart( (4,0,-2), (1,1,1), (3, 3, 1) )
c = G.cart( (0,-3,0),(1,1,1), (3,3,3) )
d = G.cart( (4,-3,-2),(1,1,1), (3,3,1) )
import Transform as T
a = T.rotate(a, (0,0,0), (1,1,0), 20.)
c = T.rotate(c, (0,-3,0), (1,1,0), 20.)
C.convertArrays2File([a,b,c,d], 'out1.fig', 'fmt_xfig')

b = C.convertArray2Tetra(b)
d = C.convertArray2Hexa(d)
C.convertArrays2File([a,b,c,d], 'out2.fig', 'fmt_xfig')

a = C.convertArray2Tetra(a)
c = C.convertArray2Hexa(c)
C.convertArrays2File([a,b,c,d], 'out3.fig', 'fmt_xfig')
```

Example file : Examples/converter/convSvg.py

```
# - convertArrays2File (fmt_svg) -
import Converter as C
import Generator as G

a = C.convertFile2Arrays('test.svg', 'fmt_svg',['Npts',130])
C.convertArrays2File(a, 'out.plt', 'bin_tp')
C.convertArrays2File(a, 'out.svg', 'fmt_svg')

a = G.cart( (0,0,0), (1.,1.,1.), (10, 10, 1) )
C.convertArrays2File([a], 'out0.svg', 'fmt_svg')

a = G.cart( (0,0,0), (100,100,100), (3, 3, 3) )
b = G.cart( (400,0,-2), (100,100,1), (3, 3, 1) )
c = G.cart( (0,300,0),(100,100,100), (3,3,3) )
d = G.cart( (400,300,-2),(100,100,1), (3,3,1) )
import Transform as T
a = T.rotate(a, (0,0,0), (1,1,0), 20.)
c = T.rotate(c, (0,300,0), (1,1,0), 20.)
C.convertArrays2File([a,b,c,d], 'out1.svg', 'fmt_svg')

b = C.convertArray2Tetra(b)
d = C.convertArray2Hexa(d)
C.convertArrays2File([a,b,c,d], 'out2.svg', 'fmt_svg')

a = C.convertArray2Tetra(a)
c = C.convertArray2Hexa(c)
C.convertArrays2File([a,b,c,d], 'out3.svg', 'fmt_svg')
```

Example file : Examples/converter/convertMesh2Array.py

```
# - convertMesh2Array (mesh de Cassiopee) -
from elsA_user import *
```

```
import Transform as T
import Converter as C
import Converter.Cassiopee as CC
import Generator as G

msh = mesh(name='msh')
msh.set('file', 'mesh.tp')
msh.set('format', 'fmt_tp')
msh.submit()

a = CC.convertMesh2Array( msh )
C.convertArrays2File([a], "mesh.plt", "bin_tp")
```

Example file : Examples/converter/convertArray2Mesh.py

```
# - convertArray2Mesh (Mesh Cassiopee) -
from elsA_user import *
import Converter.Cassiopee as CC
import Converter as C
import Transform as T

# Create a cartesian mesh
msh = mesh(name='msh')
msh.submit()

# Get arrays from file
import Generator as G
a = G.cart((0.,0.,0.), (0.1,0.1,1.), (10,10,10))

# Convert arrays to mesh
CC.convertArray2Mesh(a, msh)

# Save the mesh
a = CC.convertMesh2Array(msh)
C.convertArrays2File([a], "mesh.tp", "fmt_tp")
```

Example file : Examples/converter/convertWindow2Array.py

```
# - convertWindow2Array (Window de Cassiopee) -
from elsA_user import *
import Transform as T
import Converter as C
import Converter.Cassiopee as CC
import Generator as G

msh = mesh(name='msh')
msh.submit()
a = G.cart( (0,0,0), (1,1,1), (10,10,10))

CC.convertArray2Mesh(a,msh)

blk = block(name='blk')
blk.set('mesh', 'msh')
blk.submit()

win = window('blk', name='win')
win.set('wnd', [1, msh.getI('im'), 1, 1, 1, msh.getI('km')])
win.submit()

a = CC.convertWindow2Array( win )
C.convertArrays2File([a], "win.plt", "bin_tp")
```

Example file : Examples/converter/convertDesFunction2Function.py

```
# - convertDesFunction2Function (DesFunction Cassiopee) -
import Converter as C
```

```
import Converter.Cassiopee as CC
import Generator as G
import Transform as T
from elsA_user import *
from math import *

# Create a motion function
Func = function('wing_motion', name='Func')
Func.set("transl_speed_x",0.) # coord du centre de rotation dans Ra
Func.set("transl_speed_y",0.) # = transl*time
Func.set("transl_speed_z",0.)
Func.set("axis_pnt_x",5.) # coord du centre de rotation repere rel
Func.set("axis_pnt_y",5.)
Func.set("axis_pnt_z",0.)
Func.set("axis_vct_x",0.) # vecteur rotation
Func.set("axis_vct_y",0.)
Func.set("axis_vct_z",1.)
Func.set("alp0",0.) # rotation d'angle (alp0+cos(omega*t)+...)
Func.set("nhalp",2)
Func.set("alp1c",1.)
Func.set("alp1s",0.)
Func.set("alp2c",0.)
Func.set("alp2s",0.)
Func.set("omega",1.)

F = CC.convertDesFunction2Function( Func )

# Create a cartesian grid
array = G.cart( (0,0,0), (1,1,1), (11,11,1))

# Move the mesh
t = 0.

print F(t)
a = T.move(array, F, t)
C.convertArrays2File([a], "mesh.plt", "bin_tp")

# Equivalent to :
b = T.rotate(array, (1,0,0), (0,0,1), 3.14/180.*cos(t)*180./3.14)
c = T.translate(b, (0,0,0))
C.convertArrays2File([c], "mesh2.plt", "bin_tp")
```

Example file : Examples/converter/convertWins2String.py

```
# - convertDesObjs2String -
from elsA_user import *
import Connector as X
import Converter.Cassiopee as CC
import Converter as C
import Transform as T
import Generator as G

# build blk1
a = G.cart((0.,0.,0.), (0.1,0.1,1.), (10,10,2))
msh = mesh(name='msh')
msh.submit()
CC.convertArray2Mesh(a, msh)
blk = block(name='blk')
blk.set('mesh', 'msh')
blk.submit()

# build blk2
a2 = G.cart((0.2,0.,0.), (0.1,0.1,1.), (10,10,2))
msh2 = mesh(name='msh2')
msh2.submit()
```

```
CC.convertArray2Mesh(a2, msh2)
a = T.rotate(a2, (0.2,0.,0.), (0.,0.,1.), 18.)

C.convertArrays2File([a], "mesh.plt", "bin_tp")
C.convertArrays2File([a2], "mesh2.plt", "bin_tp")

blk2 = block(name='blk2')
blk2.set('mesh', 'msh2')
blk2.submit()

# create mask
wnd_mask = window('blk2', name='wnd_mask')
wnd_mask.set('wnd', [4, 6, 4, 6, 1, 2])

wins1 = [wnd_mask]
wins = CC.convertDesObjs2String( wins1 )

msk = mask(wins, name = 'msk')
msk.set('type', 'cart_elts')
msk.set('dim1', 100)
msk.set('dim2', 100)
msk.set('area', 'classical')
msk.set('proj_direction', 'y')
msk.attach('blk')
msk.submit()
```

Example file : Examples/converter/convertEMesh2Array.py

```
# - convertMesh2Array pour un Mesh d'elsA -
from elsA_user import *
import Transform as T
import Converter as C
import Converter.Elsa as CE
import Generator as G

pb = cfdpb(name='pb')
msh = mesh(name='msh')
msh.set('file', 'mesh.tp')
msh.set('format', 'fmt_tp')
msh.submit()

a = CE.convertMesh2Array( msh )
C.convertArrays2File([a], "mesh.plt", "bin_tp")
```

Example file : Examples/converter/convertArray2EMesh.py

```
# - convertArray2Mesh (Mesh elsA) -
from elsA_user import *
import Converter.Elsa as CE
import Converter as C
import Transform as T

pb = cfdpb(name='pb')

# Create a cartesian mesh
msh = mesh(name='msh')
msh.set('file', 'mesh.tp')
msh.set('format', 'fmt_tp')
msh.submit()

# Get arrays from file
import Generator as G
a = G.cart((0.,0.,0.), (0.1,0.1,1.), (10,10,2))
C.convertArrays2File([a], "mesh.tp", "fmt_tp")
```

```
# Convert arrays to mesh
CE.convertArray2Mesh(a, msh)

# Save the mesh
a = CE.convertMesh2Array(msh)
C.convertArrays2File([a], "mesh.plt", "bin_tp")
```

Example file : Examples/converter/convertEWindow2Array.py

```
# - convertWindow2Array pour une Window d elsA -
from elsA_user import *
import Transform as T
import Converter as C
import Converter.Elsa as CE
import Generator as G

pb = cfdpb(name='pb')
msh = mesh(name='msh')
msh.set('file','mesh.tp')
msh.set('format','fmt_tp')
msh.submit()

blk = block(name='blk')
blk.set('mesh', 'msh')
blk.submit()

win = window('blk', name='win')
win.set('wnd', [1, msh.getI('im'), 1, 1, 1, msh.getI('km')])
win.submit()

a = CE.convertWindow2Array( win )
C.convertArrays2File([a], "win.plt", "bin_tp")
```

Example file : Examples/converter/convertEDesFunction2Func.py

```
# - convertDesFunction2Function pour une DesFunction elsA -
from elsA_user import *
import Converter as C
import Converter.Elsa as CE
import Generator as G
import Transform as T
from math import *

pb = cfdpb(name='pb')

# Create a motion function
Func = function('wing_motion', name='Func')
Func.set("transl_speed_x",0.) # coord du centre de rotation dans Ra
Func.set("transl_speed_y",0.) # = transl*time
Func.set("transl_speed_z",0.)
Func.set("axis_pnt_x",5.) # coord du centre de rotation repere rel
Func.set("axis_pnt_y",5.)
Func.set("axis_pnt_z",0.)
Func.set("axis_vct_x",0.) # vecteur rotation
Func.set("axis_vct_y",0.)
Func.set("axis_vct_z",1.)
Func.set("alp0",0.) # rotation d'angle (alp0+cos(omega*t)+...)
Func.set("nhalp",2)
Func.set("alp1c",1.)
Func.set("alp1s",0.)
Func.set("alp2c",0.)
Func.set("alp2s",0.)
Func.set("omega",1.)

F = CE.convertDesFunction2Function( Func )
```

```
# Create a cartesian grid
a = G.cart( (0,0,0), (1,1,1), (11,11,1))
# Move the mesh
t = 0.
a = T.move(a, F, t)
C.convertArrays2File([a], "mesh.plt", "bin_tp")

# Equivalent to :
b = T.rotate(array, (1,0,0), (0,0,1), 3.14/180.*cos(t)*180./3.14)
c = T.translate(b, (0,0,0))
C.convertArrays2File([c], "mesh2.plt", "bin_tp")
```

Example file : Examples/converter/convertArray2Array3D.py

```
# - convertArrays2Arrays3D -
import Generator as G
import Converter.Array3D

a = G.cart( (0,0,0), (0.1, 0.2, 1.), (11, 4, 1))
b = Converter.Array3D.convertArrays2Arrays3D([a]) ; print b
```

Example file : Examples/converter/convertArray3D2Array.py

```
# - convertArrays3D2Arrays -
import Converter as C
import Generator as G
import Converter.Array3D

a = G.cart( (0,0,0), (0.1, 0.2, 1.), (11, 4, 2))
b = Converter.Array3D.convertArrays2Arrays3D([a])
c = Converter.Array3D.convertArrays3D2Arrays(b) ; print c
```

Example file : Examples/converter/restart.py

```
# Lit un fichier output.plt en centres
# Ecrit chaque bloc separement au format binaire v3d
import Converter as C

# Lecture d'une fichier binaire tecplot
arrays = C.convertFile2Arrays("output.plt", "bin_tp")

# Ecriture pour chaque zone dans un fichier separe v3d binaire
i = 1
for ar in arrays:
    if i < 10:
        C.convertArrays2File([ar], "rep0"+repr(i)+".v3d", "bin_v3d")
    else:
        C.convertArrays2File([ar], "rep"+repr(i)+".v3d", "bin_v3d")
    i=i+1
```

Example file : Examples/converter/conv8.py

```
# - convertFile2Arrays -
# Lecture de fichier structure en tecplot binaire
# Lecture de fichier non structure en tecplot binaire
# Ecriture d'un fichier global structure/non structure
import Converter as C

arrays = C.convertFile2Arrays("dauphin.plt", "bin_tp")
unsArrays = C.convertFile2Arrays("unstr.plt", "bin_tp")
arrays.append(unsArrays[0])
C.convertArrays2File(arrays, "unstr2.plt", "bin_tp")
```

Example file : Examples/converter/shell.py

```
#!/usr/bin/env python

# Vous savez ecrire un python transformant un fichier
# formate tecplot en fichier binaire tecplot :
# import Converter as C
# a = C.convertFile2Arrays("in.tp", "fmt_tp")
# C.convertArrays2File(a, "out.plt", "bin_tp")
#
# Voila comment le transformer en commande shell utilisable
# par tout le monde! :

# tout d'abord definir une fonction equivalente au script precedent :
def conv(fmttpFile, bintpFile):
  import Converter as C
  a = C.convertFile2Arrays(fmttpFile, "fmt_tp")
  C.convertArrays2File(a, bintpFile, "bin_tp")

# Puis une fonction main appelant la fonction precedente :
if (__name__ == "__main__"):
  import sys
  if (len(sys.argv) < 3):
    print "Two arguments are needed!"
  else:
    conv(sys.argv[1], sys.argv[2])

# Et voila le travail!
```