ONERA

THE FRENCH AEROSPACE LAB

Compressor Documentation

Release 3.7

/ELSA/MU-14010/V3.7

Oct 19, 2023

CONTENTS

1 Preamble **1**

2 List of functions **3**

3 Contents **5**
- 3.1 Index field compression 5
- 3.2 Object serialize/compression 6

4 Index **13**

CHAPTER

ONE

PREAMBLE

Compressor enables fields compression for arrays/pyTrees.

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

To use the module with the Compressor array interface:

```
import Compressor
```

To use the module with the CGNS/Python interface:

```
import Compressor.PyTree as Compressor
```

CHAPTER

TWO

LIST OF FUNCTIONS

– Index field compression

Compressor.deltaIndex(index, ref)	Return the delta between index and ref.

– Object serializer/compression

Compressor.pack(a[, method])	Serialize or compress a.
Compressor.unpack(a[, method])	Deserialize or decompress a.

– CGNS Zones/tree compression

Compressor.PyTree.compressCartesian(t[, ...])	For Cartesian grids, replace Grid Coordinates with a compressed node.
Compressor.PyTree.uncompressCartesian(t)	For Cartesian grids, recreate Grid Coordinates from compressed zones.
Compressor.PyTree.compressCellN(t[, varNames])	Compress cellN (0,1,2) lossless on 2 bits.
Compressor.PyTree.compressCoords(t[, tol, ctype])	Compress coordinates lossless or with a relative tolerance.
Compressor.PyTree.compressFields(t[, tol, ...])	Compress fields lossless or with a relative tolerance.
Compressor.PyTree.compressElements(t)	Compress lossless Element connectivity.
Compressor.PyTree.compressAll(t)	Compress coords, fields and connectivity (lossless).
Compressor.PyTree.uncompressAll(t)	Uncompress all compressed data.

CHAPTER

THREE

CONTENTS

3.1 Index field compression

Compressor.**deltaIndex**(*a*, *ref*)

Compress a list of indices using delta algorithm. The return Delta contains the number of added indices in a when compared to ref, the list of added indices, the number of suppressed indices, the list of suppressed indices.

Parameters

- **a** (numpy of ints) – input indices
- **ref** (numpy) – compared indices

Returns

list of added indices, the number of supressed indices, list of suppress indices

Return type

(numpy, int, numpy)

- Compression by delta (numpy):

```
# - deltaIndex -
import numpy
import Compressor

# Liste des indexes de reference
indRef = numpy.array([1,2,3,4,5], dtype='int32')

# Liste des indexes a comparer a la reference
index = numpy.array([1,2,3,4], dtype='int32')
```

(continues on next page)

(continued from previous page)

```
delta = Compressor.deltaIndex(index, indRef)
print(delta)
```

3.2 Object serialize/compression

Compressor.**pack**(*a*)

Serialize/compress a python object a. For now, this is only a general interface to pickle module.

Parameters
a (python object) – any python object

Returns
serialized stream

- Object serialization (numpy):

```
# - pack -
import Compressor
import Generator.PyTree as G
a = G.cart((0,0,0), (1,1,1), (1000,100,100))
b = Compressor.pack(a)
```

Compressor.**unpack**(*a*)

Deserialize/decompress a serialized stream b. For now, this is only a general interface to pickle module.

Parameters
a (serialized stream) – a serialized stream as produced by pack

Returns
python object

- Object deserialization (numpy):

```
# - unpack -
import Compressor
import Generator.PyTree as G
a = G.cart((0,0,0), (1,1,1), (1000,100,100))
```

(continues on next page)

(continued from previous page)

```
b = Compressor.pack(a)
c = Compressor.unpack(b)
```

Compressor.PyTree.**compressCartesian**(*a*)

Compress zones if they are regular Cartesian grids. Create a CartesianData node containing the 6 floats corresponding to first point and steps in 3 directions.

Exists also as an in-place version (_compressCartesian) which modifies a and returns None.

Parameters
a ([zone, list of zones, base, pyTree]) – input data

Returns
identical to input

- Cartesian compression (pyTree):

```
# - compressCartesian (pyTree) -
import Compressor.PyTree as Compressor
import Generator.PyTree as G
import Converter.Internal as Internal

a = G.cart((0,0,0), (1,1,1), (10,10,10))
Compressor._compressCartesian(a)
Internal.printTree(a)
```

Compressor.PyTree.**uncompressCartesian**(*a*)

Uncompress zones that has been compressed with compressCartesian. Exists also as an in-place version (_uncompressCartesian) which modifies a and returns None.

Parameters
a ([zone, list of zones, base, pyTree]) – input data

Returns
identical to input

- Cartesian uncompression (pyTree):

```
# - uncompressCartesian (pyTree) -
import Compressor.PyTree as Compressor
```

(continues on next page)

(continued from previous page)

```
import Generator.PyTree as G
import Converter.Internal as Internal

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

Compressor._uncompressCartesian(a)
Internal.printTree(a)
```

Compressor.PyTree.**compressCellN**(*a*, *varNames=['cellN']*)

Compress cellN fields (valued 0,1,2).

Exists also as an in-place version (_compressCellN) which modifies a and returns None.

Parameters

- **a** ([zone, list of zones, base, pyTree]) – input data
- **varNames** (list of strings) – list of 'cellN' names

Returns

identical to input

- CellN compression (pyTree):

```
# - compressCellN (pyTree) -
import Compressor.PyTree as Compressor
import Generator.PyTree as G
import Converter.PyTree as C
import Converter.Internal as Internal

a = G.cart((0,0,0), (1,1,1), (10,11,12))
C._initVars(a, '{centers:cellN}=1.')
Compressor._compressCellN(a)
Compressor._uncompressAll(a)
Internal.printTree(a)
C.convertPyTree2File(a, 'out.cgns')
```

Compressor.PyTree.**compressCoords**(*a*, *tol=1.e-8*, *ctype=0*)

Compress zone coordinates with sz, zfp or fpc libraries.

Fpc is a lossless compression and doesn't use tol. sz and zfp are approximative compressions controling error at a given relative tolerance.

Exists also as an in-place version (_compressCoords) which modifies a and returns None.

Parameters

- **a** ([zone, list of zones, base, pyTree]) – input data
- **tol** (float) – control relative error on output (sz, zfp)
- **ctype** (0 (sz), 1 (zfp), 5 (fpc)) – compression algorithm

Returns

identical to input

- Coordinates compression (pyTree):

```
# - compressCoords (pyTree) -
import Compressor.PyTree as Compressor
import Generator.PyTree as G
import Converter.PyTree as C
import Converter.Internal as Internal

a = G.cart((0,0,0), (1,1,1), (5,3,4))
Compressor._compressCoords(a, tol=1.e-7, ctype=0)
Compressor._uncompressAll(a)
Internal.printTree(a)
C.convertPyTree2File(a, 'out.cgns')
```

Compressor.PyTree.**compressFields**(*a*, *tol=1.e-8*, *ctype=0*, *varNames=None*)

Compress zone fields with sz, zfp or fpc libraries.

Fpc is a lossless compression and doesn't use tol. sz and zfp are approximative compressions controling error at a given relative tolerance.

Exists also as an in-place version (_compressFields) which modifies a and returns None.

Parameters

- **a** ([zone, list of zones, base, pyTree]) – input data
- **tol** (float) – control relative error on output
- **ctype** (0 (sz), 1 (zfp), 5 (fpc)) – compression algorithm
- **varNames** (list of strings) – optional list of variable names to compress (e.g. ['f', 'centers:G'])

Returns

identical to input

- Field compression (pyTree):

```
# - compressFields (pyTree) -
import Compressor.PyTree as Compressor
import Generator.PyTree as G
import Converter.PyTree as C
import Converter.Internal as Internal

a = G.cart((0,0,0), (1,1,1), (10,10,10))
C._initVars(a, '{F}={CoordinateX}')
C._initVars(a, '{centers:G}={centers:CoordinateY}')
Compressor._compressFields(a, tol=1.e-6, ctype=5)
Compressor._uncompressAll(a)
Internal.printTree(a)
C.convertPyTree2File(a, 'out.cgns')
```

Compressor.PyTree.**compressElements**(*a*)

Compress zone elements (connectivity).

Exists also as an in-place version (_compressElements) which modifies a and returns None.

Parameters
a ([zone, list of zones, base, pyTree]) – input data

Returns
identical to input

- Element compression (pyTree):

```
# - compressElements (pyTree) -
import Compressor.PyTree as Compressor
import Generator.PyTree as G
import Converter.PyTree as C
import Converter.Internal as Internal

a = G.cartHexa((0,0,0), (1,1,1), (25,23,24))
Compressor._compressElements(a)
Internal.printTree(a)
Compressor._uncompressAll(a)
Internal.printTree(a)
C.convertPyTree2File(a, 'out.cgns')
```

Compressor.PyTree.**compressAll**(*a*)

Compress zones (fields, connectivity) in the best and lossless way.

Exists also as an in-place version (_compressAll) which modifies a and returns None.

Parameters
a ([zone, list of zones, base, pyTree]) – input data

Returns
identical to input

- Zone compression (pyTree):

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

a = G.cart((0,0,0), (1,1,1), (10,10,10))
C._initVars(a, '{F}={CoordinateX}')
C._initVars(a, '{centers:G}={centers:CoordinateY}')
Compressor._compressAll(a)
C.convertPyTree2File(a, 'out.cgns')
```

Compressor.PyTree.**uncompressAll**(*a*)

Uncompress zones compressed with the previous compressors.

Exists also as an in-place version (_uncompressAll) which modifies a and returns None.

Parameters
a ([zone, list of zones, base, pyTree]) – input data

Returns
identical to input

- Zone decompression (pyTree):

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

a = G.cart((0,0,0), (1,1,1), (10,10,10))
C._initVars(a, '{F}={CoordinateX}')
```

(continues on next page)

(continued from previous page)

```
C._initVars(a, '{centers:G}={centers:CoordinateY}')
Compressor._compressCoords(a, tol=1.e-6)
Compressor._compressFields(a, tol=1.e-6)
Compressor._uncompressAll(a)
C.convertPyTree2File(a, 'out.cgns')
```

CHAPTER

FOUR

INDEX

- genindex
- modindex
- search