



# **Post.ExtraVariables2 Documentation**

## ***Release 3.4***

**/ELSA/MU-10019/V3.4**

**Jun 29, 2022**



# CONTENTS

- 1 List of functions** **3**
- 2 Contents** **5**
  - 2.1 Volume fields . . . . . 5
  - 2.2 Surface fields . . . . . 11



This module compute derived fields from primitive variables.



## LIST OF FUNCTIONS

### – Volume fields

---

<i>Post.ExtraVariables2. extractTree(t[, vars])</i>	Create a mirror tree with less vars.
<i>Post.ExtraVariables2. computeVorticity2(t[, ...])</i>	Compute vorticity from velocity in centers.
<i>Post.ExtraVariables2. computeVorticityMagnitude2(t)</i>	Compute vorticity magnitude from velocity in centers.
<i>Post.ExtraVariables2. computeQCriterion2(t[, ...])</i>	Compute Q criterion from velocity in centers.
<i>Post.ExtraVariables2. computeLambda2(t[, ...])</i>	Compute lambda2 criterion from velocity in centers.
<i>Post.ExtraVariables2. extractPressure(t)</i>	Extract Pressure.
<i>Post.ExtraVariables2. extractVelocityMagnitude(t)</i>	Extract velocity magnitude.
<i>Post.ExtraVariables2. extractMach(t)</i>	Extract Mach.
<i>Post.ExtraVariables2. extractViscosityMolecular(t)</i>	Extract Viscosity molecular.
<i>Post.ExtraVariables2. extractViscosityEddy(t)</i>	Extract eddy viscosity.

---

### – Surface fields

---

<i>Post.ExtraVariables2. extractShearStress(teff)</i>	Extract shearStress.
<i>Post.ExtraVariables2. extractTauN(teff)</i>	Extract tau.n.
<i>Post.ExtraVariables2. extractPn(teff)</i>	Extract p.n.

---

Continued on next page

Table 2 – continued from previous page

---

<i>Post.ExtraVariables2.</i> <i>extractForce</i> (teff[, ...])	Extract forces.
<i>Post.ExtraVariables2.</i> <i>extractFrictionVector</i> (teff)	Extract tangential friction vector.
<i>Post.ExtraVariables2.</i> <i>extractFrictionMagnitude</i> (teff)	Extract friction magnitude.

---

**– 1D profiles**



## CONTENTS

### 2.1 Volume fields

`Post.ExtraVariables2.extractTree` (*t*, vars=['centers:Density', 'centers:VelocityX', 'centers:VelocityY', 'centers:VelocityZ', 'centers:Temperature', 'centers:TurbulentSANuTilde'])

Keep only some variables from tree. This is just a reference tree (no extra memory is used).

#### Parameters

- **t** (*[zone, list of zones, base, tree]*) – input tree
- **vars** (*list of strings*) – list of vars to keep in returned tree

**Returns** tree with selected variables

**Return type** identical to input

*Example of use:*

- Extract tree (pyTree):

```
# - extractTree (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
tp = PE.extractTree(a, vars=['centers:Temperature'])
C.convertPyTree2File(tp, 'out.cgns')
```

Post.ExtraVariables2.**computeVorticity2** (*t*, *ghostCells=False*)

Compute vorticity on *t* from Velocity field in centers. If *t* contains ghost cells, set argument to True. Exists also as in place function (`_computeVorticity2`) that modifies *t* and returns None.

### Parameters

- **t** (*[zone, list of zones, base, tree]*) – input tree
- **ghostCells** (*boolean*) – must be true if *t* contains ghost cells

**Returns** tree with “VorticityX,”VorticityY”,”VorticityZ” in centers

**Return type** identical to input

*Example of use:*

- Compute vorticity (pyTree):

```
# - computeVorticity2 (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
PE._computeVorticity2(a, ghostCells=True)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**computeVorticityMagnitude2** (*t*, *ghost-Cells=False*)

Compute vorticity magnitude on *t* from Velocity field in centers. If *t* contains ghost cells, set argument to True. Exists also as in place function (`_computeVorticityMagnitude2`) that modifies *t* and returns None.

### Parameters

- **t** (*[zone, list of zones, base, tree]*) – input tree
- **ghostCells** (*boolean*) – must be true if *t* contains ghost cells

**Returns** tree with “VorticityMagnitude” in centers

**Return type** identical to input

*Example of use:*

- Compute vorticity magnitude (pyTree):

```
# - computeVorticityMagnitude2 (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
PE._computeVorticityMagnitude2(a, ghostCells=True)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**computeQCriterion2** (*t*, *ghostCells=False*)

Compute Q criterion on *t* from Velocity field in centers. If *t* contains ghost cells, set argument to True. Exists also as in place function (`_computeQCriterion2`) that modifies *t* and returns None.

#### Parameters

- **t** (*[zone, list of zones, base, tree]*) – input tree
- **ghostCells** (*boolean*) – must be true if *t* contains ghost cells

**Returns** tree with “QCriterion” in centers

**Return type** identical to input

*Example of use:*

- Compute Q criterion (pyTree):

```
# - computeQCriterion2 (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
PE._computeQCriterion2(a, ghostCells=True)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**computeLambda2** (*t*, *ghostCells=False*)

Compute lambda2 on *t* from Velocity field in centers. If *t* contains ghost cells, set argument to True. Exists also as in place function (`_computeLambda2`) that modifies *t* and returns None.

### Parameters

- **t** (*[zone, list of zones, base, tree]*) – input tree
- **ghostCells** (*boolean*) – must be true if t contains ghost cells

**Returns** tree with “lambda2” in centers

**Return type** identical to input

*Example of use:*

- Compute lambda2 (pyTree):

```
# - computeLambda2 (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
PE._computeLambda2(a, ghostCells=True)
C.convertPyTree2File(a, 'out.cgns')
```

---

### Post.ExtraVariables2.**extractPressure** (t)

Compute Pressure on t from Temperature and Density field in centers with  $P = \rho r T$ . The tree t must have a ReferenceState node. Cv and Gamma are taken from ReferenceState and  $r = C_v * (\text{Gamma}-1)$ . Exists also as in place function (`_extractPressure`) that modifies t and returns None.

**Parameters** **t** (*[zone, list of zones, base, tree]*) – input tree

**Returns** tree with “Pressure” in centers

**Return type** identical to input

*Example of use:*

- Extract pressure (pyTree):

```
# - extractPressure (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
```

(continues on next page)

(continued from previous page)

```
I._cons2Prim(a)
PE._extractPressure(a)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**extractVelocityMagnitude**(*t*)

Compute velocity magnitude on *t* from Velocity field in centers. Exists also as in place function (`_extractVelocityMagnitude`) that modifies *t* and returns None.

**Parameters** *t* (*[zone, list of zones, base, tree]*) – input tree

**Returns** tree with “VelocityMagnitude” in centers

**Return type** identical to input

*Example of use:*

- Extract velocity magnitude (pyTree):

```
# - extractVelocityMagnitude (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
PE._extractVelocityMagnitude(a)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**extractMach**(*t*)

Compute Mach on *t* from Velocity, Temperature and Density field in centers with  $M = u/\sqrt{\gamma p/\rho}$  and  $p = \rho r T$ . The tree *t* must have a ReferenceState node. *Cv* and *Gamma* are taken from ReferenceState and  $r = C_v * (\text{Gamma}-1)$ . Exists also as in place function (`_extractMach`) that modifies *t* and returns None.

**Parameters** *t* (*[zone, list of zones, base, tree]*) – input tree

**Returns** tree with “Mach” in centers

**Return type** identical to input

*Example of use:*

- Extract mach (pyTree):

```
# - extractMach (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
PE._extractMach(a)
C.convertPyTree2File(a, 'out.cgns')
```

---

### Post.ExtraVariables2.**extractViscosityMolecular** (*t*)

Compute ViscosityMolecular on *t* from Temperature field in centers with Sutherland law. The tree *t* must have a ReferenceState node. Cs, Mus, Ts are taken from ReferenceState. Exists also as in place function (`_extractViscosityMolecular`) that modifies *t* and returns None.

**Parameters** *t* (*[zone, list of zones, base, tree]*) – input tree

**Returns** tree with “ViscosityMolecular” in centers

**Return type** identical to input

*Example of use:*

- Extract viscosity molecular (pyTree):

```
# - extractViscosityMolecular (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
PE._extractViscosityMolecular(a)
C.convertPyTree2File(a, 'out.cgns')
```

---

### Post.ExtraVariables2.**extractViscosityEddy** (*t*)

Compute ViscosityEddy on *t* from TurbulentSANuTilde, ViscosityMolecular and Density field in centers with  $\kappa = \rho * \text{nutilde} / \mu$  and  $\mu_t = \rho * \text{nutilde} * \kappa^3 / (\kappa^3 + 7.1^3)$ . Exists also as in place function (`_extractViscosityEddy`) that modifies *t* and returns None.

**Parameters** *t* (*[zone, list of zones, base, tree]*) – input tree

**Returns** tree with “ViscosityEddy” in centers

**Return type** identical to input

*Example of use:*

- Extract viscosity eddy (pyTree):

```
# - extractViscosityEddy (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Initiator.PyTree as I
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
I._initLamb(a, position=(7.,7.), Gamma=2., MInf=0.8, loc='centers')
I._cons2Prim(a)
PE._extractViscosityEddy(a)
C.convertPyTree2File(a, 'out.cgns')
```

## 2.2 Surface fields

Post.ExtraVariables2.**extractShearStress** (*teff*)

Compute ShearStress on *teff* from ViscosityMolecular and gradxVelocityX,... in centers.  
Exists also as in place function (`_extractShearStress`) that modifies *t* and returns None.

**Parameters** *teff* (*[zone, list of zones, base, tree]*) – input tree

**Returns** tree with “ShearStressXX,XY,XZ,YY,YZ,ZZ” in centers

**Return type** identical to input

*Example of use:*

- Extract shearStress (pyTree):

```
# - extractShearStress (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,1))
for n in ['ViscosityMolecular', 'gradxVelocityX', 'gradxVelocityY',
        ↪ 'gradxVelocityZ',
        'gradyVelocityX', 'gradyVelocityY', 'gradyVelocityZ',
```

(continues on next page)

(continued from previous page)

```
'gradzVelocityX', 'gradzVelocityY', 'gradzVelocityZ']:
    C._initVars(a, '{centers:%s} = 1.'%n)
PE._extractShearStress(a)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**extractTaun** (*teff*)

Compute tau.n on teff from ShearStress in centers. Exists also as in place function (`_extractTaun`) that modifies t and returns None.

**Parameters** *teff* ([*zone*, *list of zones*, *base*, *tree*]) – input tree

**Returns** tree with “taunx,y,z” in centers

**Return type** identical to input

*Example of use:*

- Extract tau.n (pyTree):

```
# - extractTaun (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,1))
for n in ['ViscosityMolecular', 'gradxVelocityX', 'gradxVelocityY',
→ 'gradxVelocityZ',
        'gradyVelocityX', 'gradyVelocityY', 'gradyVelocityZ',
        'gradzVelocityX', 'gradzVelocityY', 'gradzVelocityZ']:
    C._initVars(a, '{centers:%s} = 1.'%n)
PE._extractShearStress(a)
PE._extractTaun(a)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**extractPn** (*teff*)

Compute P.n on teff from Pressure in centers. Exists also as in place function (`_extractPn`) that modifies t and returns None.

**Parameters** *teff* ([*zone*, *list of zones*, *base*, *tree*]) – input tree

**Returns** tree with “Pnx,y,z” in centers

**Return type** identical to input

*Example of use:*



- Extract P.n (pyTree):

```
# - extractPn (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,1))
for n in ['Pressure']:
    C._initVars(a, '{centers:%s} = 1.'%n)
PE._extractPn(a)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**extractForce** (*teff*, *withPInf=None*)

Compute the force field on *teff* from Pressure and ShearStress in centers. If *withPInf* is None:  $F = -p.n + \tau.n$  Else:  $F = -(p-pinf).n + \tau.n$  Exists also as in place function (`_extractForce`) that modifies *t* and returns None.

#### Parameters

- **teff** (*[zone, list of zones, base, tree]*) – input tree
- **withPInf** (*None or float*) – None or infinite field pressure

**Returns** tree with “Fx,y,z” in centers

**Return type** identical to input

*Example of use:*

- Extract Force (pyTree):

```
# - extractShearStress (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,1))
for n in ['ViscosityMolecular', 'Pressure',
          'gradxVelocityX', 'gradxVelocityY', 'gradxVelocityZ',
          'gradyVelocityX', 'gradyVelocityY', 'gradyVelocityZ',
          'gradzVelocityX', 'gradzVelocityY', 'gradzVelocityZ']:
    C._initVars(a, '{centers:%s} = 1.'%n)
PE._extractShearStress(a)
PE._extractForce(a)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**extractFrictionVector** (*teff*)

Compute the friction vector on *teff* from ShearStress in centers with  $\text{taut} = \text{tau.n} - (\text{n. tau.n})$ . Exists also as in place function (`_extractFrictionVector`) that modifies *t* and returns None.

**Parameters** *teff* (*[zone, list of zones, base, tree]*) – input tree

**Returns** tree with “FrictionX,FrictionY,FrictionZ” in centers

**Return type** identical to input

*Example of use:*

- Extract friction vector (pyTree):

```
# - extractFrictionVector (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,1))
for n in ['ViscosityMolecular', 'Pressure',
          'gradxVelocityX', 'gradxVelocityY', 'gradxVelocityZ',
          'gradyVelocityX', 'gradyVelocityY', 'gradyVelocityZ',
          'gradzVelocityX', 'gradzVelocityY', 'gradzVelocityZ']:
    C._initVars(a, '{centers:%s} = 1. '%n)
PE._extractShearStress(a)
PE._extractFrictionVector(a)
C.convertPyTree2File(a, 'out.cgns')
```

Post.ExtraVariables2.**extractFrictionMagnitude** (*teff*)

Compute the friction vector magnitude on *teff* from ShearStress in centers with norm of  $\text{taut} = \text{tau.n} - (\text{n. tau.n})$ . Exists also as in place function (`_extractFrictionMagnitude`) that modifies *t* and returns None.

**Parameters** *teff* (*[zone, list of zones, base, tree]*) – input tree

**Returns** tree with “FrictionMagnitude” in centers

**Return type** identical to input

*Example of use:*

- Extract friction magnitude (pyTree):

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

(continues on next page)

(continued from previous page)

```
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,1))
for n in ['ViscosityMolecular', 'Pressure',
         'gradxVelocityX', 'gradxVelocityY', 'gradxVelocityZ',
         'gradyVelocityX', 'gradyVelocityY', 'gradyVelocityZ',
         'gradzVelocityX', 'gradzVelocityY', 'gradzVelocityZ']:
    C._initVars(a, '{centers:%s} = 1.'%n)
PE._extractShearStress(a)

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