ONERA

THE FRENCH AEROSPACE LAB

Post.ExtraVariables2 Documentation

Release 3.7

/ELSA/MU-10019/V3.7

Oct 19, 2023

CONTENTS

This module compute derived fields from primitive variables.

CHAPTER

ONE

LIST OF FUNCTIONS

– Volume fields

Post.ExtraVariables2.extractTree(t[, vars])	Create a mirror tree with less vars.
Post.ExtraVariables2.computeVorticity2(t[, ...])	Compute vorticity from velocity in centers.
Post.ExtraVariables2.computeVorticityMagnitude2(t)	Compute vorticity magnitude from velocity in centers.
Post.ExtraVariables2.computeQCriterion2(t[, ...])	Compute Q criterion from velocity in centers.
Post.ExtraVariables2.computeLambda2(t[, ...])	Compute lambda2 criterion from velocity in centers.
Post.ExtraVariables2.computeLogGradField2(t, name)	Compute log(grad field) for field in centers.
Post.ExtraVariables2.extractPressure(t)	Extract Pressure.
Post.ExtraVariables2.extractVelocityMagnitude(t)	Extract velocity magnitude.
Post.ExtraVariables2.extractMach(t)	Extract Mach.
Post.ExtraVariables2.extractViscosityMolecular(t)	Extract Viscosity molecular.
Post.ExtraVariables2.extractViscosityEddy(t)	Extract eddy viscosity.

– Surface fields

Post.ExtraVariables2.extractShearStress(teff)	Extract shearStress.
Post.ExtraVariables2.extractTaun(teff)	Extract tau.n.
Post.ExtraVariables2.extractPn(teff)	Extract p.n.
Post.ExtraVariables2.extractForce(teff[, ...])	Extract forces.
Post.ExtraVariables2.extractFrictionVector(teff)	Extract tangential friction vector.
Post.ExtraVariables2.extractFrictionMagnitude(teff)	Extract friction magnitude.

– 1D profiles

CHAPTER

TWO

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 (_computeVoriticity2) 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*, *ghostCells=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 (_computeVoriticityMagnitude2) 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))
```

(continues on next page)

(continued from previous page)

```
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.**computeLogGradField2**(*t*, *name*, *ghostCells=False*)

Compute log(grad field) on t from field in centers. If t contains ghost cells, set argument to True. Exists also as in place function (_computeLogGradField2) that modifies t and returns None.

Parameters

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

- **name** (*string*) – name of field
- **ghostCells** (*boolean*) – must be true if t contains ghost cells

Returns
tree with "LogGrad"+name in centers

Return type
identical to input

Example of use:

- Compute log(grad pressure) (pyTree):

```
# - computeLogGradField2 (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._computeLogGradField2(a, 'centers:Density', ghostCells=True)
C.convertPyTree2File(a, 'out.cgns')
```

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

Compute Pressure on t from Temperature and Density field in centers with P = ro r T. The tree t must have a ReferenceState node. Cv and Gamma are taken from ReferenceState and r = Cv * (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
```

(continues on next page)

(continued from previous page)

```
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._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/ro) and p = ro r T. The tree t must have a ReferenceState node. Cv and Gamma are taken from ReferenceState and r = Cv * (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
```

(continues on next page)

(continued from previous page)

```
import Post.ExtraVariables2 as PE

a = G.cart((0.,0.,0.), (13./100.,13./100.,1.), (100,100,2))
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 = ro * nutilde / mu and mut = ro * 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',
          '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 (_extract-Taun) 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 friciton vector on teff from ShearStress in centers with taut = tau.n - (n. tau.n) 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 friciton vector magnitude on teff from ShearStress in centers with norm of taut = tau.n - (n. tau.n) 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
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',
```

(continues on next page)

(continued from previous page)

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

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