ONERA

THE FRENCH AEROSPACE LAB

Geom.IBM Documentation

Release 3.4

/ELSA/MU-09021/V3.4

Jun 29, 2022

CONTENTS

Specific geometry modification functions for immersed boundaries.

These functions require a geometry tree “tb” or a connectivity tree “tc”.

CHAPTER

ONE

NOTES ON IBCTYPES

Table outlining the various IBCs currently supported. Please note that the "Name" are case sensitive (e.g. Slip is not supported)

IBC Type	Name	Integer Identifier
Wall slip	slip	0
Wall no slip	noslip	1
Wall model: Logarithmic	Log	2
Wall model: Musker	Musker	3
Outflow pressure	outpress	4
Injection	inj	5
Wall model: Turbulent Boundary Layer Equation (TBLE)	TBLE	6
Wall model: Mobile Musker	MuskerMob	7
Wall model: Pohlhausen	Pohlhausen	8
Wall model: Thwaites	Thwaites	9
Wall model: Mafzal	Mafzal	10
Wall model: Full TBLE	TBLE_FULL	11
Wall no slip with curvature radius	slip_cr	100

CHAPTER

TWO

LIST OF FUNCTIONS

– Setting Snear & Dfar

Geom.IBM.setSnear(t, value)	Set the value of snear in a geometry tree.
Geom.IBM.setDfar(t, value)	Set the value of dfar in a geometry tree.
Geom.IBM.snearFactor(t, sfactor)	Multiply the value of snear in a geometry tree by a sfactor.

– Setting IBC Type

Geom.IBM.setIBCType(t, value)	Set the IBC type in a geometry tree.
Geom.IBM.changeIBCType(tc, oldIBCType, ...)	Change the IBC type in a connectivity tree from oldIBCType to newIBCType.
Geom.IBM.initOutflow(tc, familyName, P_static)	Set the value of static pressure P_static for the outflow pressure IBC with family name familyName.
Geom.IBM.initInj(tc, familyName, P_tot, H_tot)	Set the total pressure P_tot, total enthalpy H_tot, and direction of the flow injDir for the injection IBC with family name familyName.
Geom.IBM.setFluidInside(t)	Set fluid inside a geometry tree.

CHAPTER

THREE

CONTENTS

Note that all the functions have an in-place version, modifying directly the data without copy. The function names must be prefixed by an '_' (e.g. _setSnear for the in-place version of setSnear)

3.1 Setting Snear & Dfar

Geom.IBM.**setSnear**(*tb*, *snear*)

Set the snear for a geometry defined by tb. Exists also as in-place (_setSnear). Snear is the local Cartesian spacing close to cells intersected by the immersed boundary.

Parameters

- **tb** ([zone, list of zones, tree]) – geometry tree
- **snear** (float) – snear value

Returns same as input

Example of use:

- Set the values of the snears (pyTree):

```
# - setSnear (pyTree) -
import Converter.PyTree as C
import Geom.IBM as D_IBM
import Geom.PyTree as D

a = D.circle((0,0,0), 1. , 0., 360.)
D_IBM._setSnear(a,0.01)
C.convertPyTree2File(a, 'out.cgns')
```

Geom.IBM.**setDfar**(*tb*, *dfar*)

Set the dfar for a geometry defined by tb. Exists also as in-place (_setDfar). Dfar is

the distance from the center of the bounding box of the immersed boundary to the edge of the domain.

Parameters

- **tb** ([zone, list of zones, tree]) – geometry tree
- **dfar** (float) – dfar value

Returns same as input

Example of use:

- Set the value of the dfar (pyTree):

```
# - setDfar (pyTree) -
import Converter.PyTree as C
import Geom.IBM as D_IBM
import Geom.PyTree as D

a = D.circle((0,0,0), 1. , 0., 360.)
a = D_IBM.setDfar(a, 10)

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

Geom.IBM.**snearFactor**(*tb, sfactor*)

Multiply the snears in the geometry defined by tb by a factor. Exists also as in-place (_snearFactor).

Parameters

- **tb** ([zone, list of zones, tree]) – geometry tree
- **sfactor** (float) – multiplying factor

Returns same as input

Example of use:

- Modifying the value of snears (pyTree):

```
# - snearFactor (pyTree) -
import Converter.PyTree as C
import Geom.IBM as D_IBM
import Geom.PyTree as D

a = D.circle((0,0,0), 1. , 0., 360.)
D_IBM._setSnear(a, 0.01)
D_IBM._snearFactor(a, 2)
C.convertPyTree2File(a, 'out.cgns')
```

3.2 Setting IBC Type

Geom.IBM.**setIBCType**(*tb*, *ibctype*)

Set the type of IBC for the geometry defined by tb. Exists also as in-place (_setIBCType). See the table in "Notes on IBCTypes" for the IBCs currently supported.

Parameters

- **tb** ([zone, list of zones, tree]) – geometry tree
- **ibctype** (string) – name of the type of IBC

Returns same as input

Example of use:

- Set the type of IBC (pyTree):

```
# - setIBCType (pyTree) -
import Converter.PyTree as C
import Geom.IBM as D_IBM
import Geom.PyTree as D

a = D.circle((0,0,0), 1. , 0., 360.)
D_IBM._setIBCType(a, "Musker")
C.convertPyTree2File(a, 'out.cgns')
```

Geom.IBM.**changeIBCType**(*tc*, *oldBCType*, *newBCType*)

Change the IBC type in a connectivity tree. Exists also as in-place (_changeIBCType). Please refer to the table in "Notes on IBCTypes" for details on the integer identifies for the various IBC types.

Parameters

- **tc** ([zone, list of zones, tree]) – connectivity tree
- **oldBCType** (integer) – type of ibc
- **newBCType** (integer) – type of ibc

Returns same as input

Example of use:

- Change the type of IBC (pyTree):

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

(continues on next page)

(continued from previous page)

```
import Geom.IBM as D_IBM
import Geom.PyTree as D
import numpy

a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,11,12))
a = C.node2Center(a)
for z in Internal.getZones(a):
    Internal._createChild(z, 'IBCD_2_'+z[0] , 'ZoneSubRegion_t', value=z[0])

Nlength = numpy.zeros((10),numpy.float64)
for z in Internal.getZones(a):
    subRegions = Internal.getNodesFromType1(z, 'ZoneSubRegion_t')
    for zsr in subRegions:
        Internal._createChild(zsr, 'ZoneRole', 'DataArray_t', value='Donor')
        Internal._createChild(zsr, 'GridLocation', 'GridLocation_t', value=
↪'CellCenter')
        zsr[2].append(['Pressure', Nlength, [], 'DataArray_t'])
        zsr[2].append(['Density', Nlength, [], 'DataArray_t'])
        zsr[2].append(['VelocityX', Nlength, [], 'DataArray_t'])
        zsr[2].append(['VelocityY', Nlength, [], 'DataArray_t'])
        zsr[2].append(['VelocityZ', Nlength, [], 'DataArray_t'])
a = D_IBM.changeIBCType(a,2,3)

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

Geom.IBM.**setFluidInside**(*tb*)

Define the fluid inside a surface defined by tb. In that case, the IBM mesh will be defined inside tb. Exists also as in-place (_setFluidInside).

Parameters **tb** ([zone, list of zones, tree]) – geometry tree

Returns same as input

Example of use:

- Change the type of IBC (pyTree):

```
# - setFluidInsides (pyTree) -
import Converter.PyTree as C
import Geom.IBM as D_IBM
import Geom.PyTree as D
# Geometry
a = D.circle((0,0,0), 1. , 0., 360.)

D_IBM._setFluidInside(a)
C.convertPyTree2File(a, 'out.cgns')
```

Geom.IBM.**initOutflow**(*tc*, *familyName*, *P_static*)

Set the value of the static pressure P_static for the outflow pressure IBC with family name familyName. Exists also as in-place (_initOutflow).

Parameters

- **tc** ([zone, list of zones, tree]) – connectivity tree
- **familyName** (string) – familyName
- **P_static** (float) – static pressure

Returns same as input

Example of use:

- Set outflow IBC (pyTree):

```
# - initOutflow (pyTree) -
import Converter.Internal as Internal
import Converter.PyTree as C
import Generator.PyTree as G
import Geom.IBM as D_IBM
import Geom.PyTree as D
import numpy

a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,11,12))
a = C.node2Center(a)
for z in Internal.getZones(a):
    Internal._createChild(z, 'IBCD_4_'+z[0] , 'ZoneSubRegion_t', value=z[0])


Nlength = numpy.zeros((10),numpy.float64)
for z in Internal.getZones(a):
    subRegions = Internal.getNodesFromType1(z, 'ZoneSubRegion_t')
    for zsr in subRegions:
        Internal._createChild(zsr, 'ZoneRole', 'DataArray_t', value='Donor')
        Internal._createChild(zsr, 'GridLocation', 'GridLocation_t', value=
↪'CellCenter')
        zsr[2].append(['Pressure', Nlength, [], 'DataArray_t'])
        Internal._createChild(zsr, 'FamilyName', 'FamilyName_t', value='CART_
↪LOCAL')

a=D_IBM.initOutflow(a,'CART_LOCAL',101325)

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

Geom.IBM.**initInj**(*tc, familyName, P_tot, H_tot, injDir=[1.,0.,0.]*)

Set the total pressure P_tot, total enthalpy H_tot, and direction of the flow injDir for the injection IBC with family name familyName. Exists also as in-place (_initInj).

Parameters

- **tc** ([zone, list of zones, tree]) – connectivity tree
- **familyName** (string) – familyName
- **P_tot** (float) – total pressure
- **H_tot** (float) – total enthalpy
- **injDir** (float list) – direction of the injection w.r.t to the reference coordinate axis

Returns same as input

Example of use:

- Set injection IBC (pyTree):

```
# - initInj (pyTree) -
import Converter.Internal as Internal
import Converter.PyTree as C
import Generator.PyTree as G
import Geom.IBM as D_IBM
import Geom.PyTree as D
import numpy


a = G.cart((0.,0.,0.), (0.1,0.1,0.2), (10,11,12))
a = C.node2Center(a)
for z in Internal.getZones(a):
    Internal._createChild(z, 'IBCD_5_'+z[0] , 'ZoneSubRegion_t', value=z[0])


Nlength = numpy.zeros((10),numpy.float64)
for z in Internal.getZones(a):
    subRegions = Internal.getNodesFromType1(z, 'ZoneSubRegion_t')
    for zsr in subRegions:
        Internal._createChild(zsr, 'ZoneRole', 'DataArray_t', value='Donor')
        Internal._createChild(zsr, 'GridLocation', 'GridLocation_t', value=
↪'CellCenter')



        zsr[2].append(['StagnationEnthalpy', Nlength, [], 'DataArray_t'])
        zsr[2].append(['StagnationPressure', Nlength, [], 'DataArray_t'])
        zsr[2].append(['dirx', Nlength, [], 'DataArray_t'])
```

(continues on next page)

(continued from previous page)

```
        zsr[2].append(['diry', Nlength, [], 'DataArray_t'])
        zsr[2].append(['dirz', Nlength, [], 'DataArray_t'])


        Internal._createChild(zsr, 'FamilyName', 'FamilyName_t', value='CART_
↪LOCAL')

a=D_IBM.initInj(a,'CART_LOCAL',10,20,injDir=[0.5,0.5,0.])

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

CHAPTER

FOUR

INDEX

- genindex
- modindex
- search