RigidMotion Documentation

Release 3.7

/ELSA/MU-12016/V3.7

Oct 19, 2023

CONTENTS

CHAPTER

ONE

PREAMBLE

RigidMotion enables to define or compute rigid motions for arrays (as defined in Converter documentation) or for CGNS/Python trees (pyTrees).

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

For use with the array interface, you have to import RigidMotion module:

```
import RigidMotion
```

For use with the pyTree interface:

```
import RigidMotion.PyTree as RigidMotion
```

CHAPTER

TWO

LIST OF FUNCTIONS

– Prescribed motions

RigidMotion.PyTree.setPrescribedMotion1(t, name)	Define a motion of type 1 (time strings).
RigidMotion.PyTree.setPrescribedMotion2(t, name)	Define a motion of type 2 (rotor)
RigidMotion.PyTree.setPrescribedMotion3(t, name)	Define a motion of type 3 (constant rotation+translation speed).

– General functions

RigidMotion.PyTree.evalPosition(a, time[, F])	Move the mesh with defined motion to time t.
RigidMotion.PyTree.evalGridSpeed(a, time)	Eval grid speed at given time.
RigidMotion.PyTree.copyGrid2GridInit(t[, mode])	
RigidMotion.PyTree.copyGridInit2Grid(t)	

CHAPTER

THREE

CONTENTS

RigidMotion.PyTree.**setPrescribedMotion1**(*a*, *motionName*, *tx='0'*, *ty='0'*, *tz='0'*, *cx='0'*, *cy='0'*, *cz='0'*, *ex='0'*, *ey='0'*, *ez='0'*, *angle='0'*)

Set a prescribed motion defined by a translation of the origin (tx,ty,tz), the center of a rotation (cx,cy,cz), the second point of the rotation axis (ex,ey,ez) and the rotation angle in degrees. They can depend on time {t}.

Exists also as an in-place version (_setPrescribedMotion1) which modifies a and returns None.

Parameters

- **a** (`[array, list of arrays] or [pyTree, base, zone, list of zones]`) – Input data
- **tx** (`string`) – translation in x motion string
- **ty** (`string`) – translation in y motion string
- **tz** (`string`) – translation in z motion string
- **cx** (`string`) – rotation center x coordinate motion string
- **cy** (`string`) – rotation center y coordinate motion string
- **cz** (`string`) – rotation center z coordinate motion string
- **ex** (`string`) – rotation axis x coordinate motion string
- **ey** (`string`) – rotation axis y coordinate motion string
- **ez** (`string`) – rotation axis z coordinate motion string
- **angle** (`string`) – rotation angle motion string

Example of use:

- Set a prescribed motion of type 1 (pyTree):

```python
# - setPrescribedMotion1 (pyTree) -
# Motion defined by time string
import RigidMotion.PyTree as R
import Converter.PyTree as C
import Geom.PyTree as D

a = D.sphere((1.2,0.,0.), 0.2, 30)
a = R.setPrescribedMotion1(a, 'trans', tx="{t}")

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

`RigidMotion.PyTree.`**`setPrescribedMotion2`**(*a, motionName, transl_speed, psi0, pis0_b, alp_pnt, alp_vct, alp0, rot_pnt, rot_vct, rot_omg, del_pnt, del_vct, del0, delc, dels, bet_pnt, bet_vct, bet0, betc, bets, tet_pnt, tet_vct, tet0, tetc, tets, span_vct, pre_lag_pnt, pre_lag_vct, pre_lag_ang, pre_con_pnt, pre_con_vct, pre_con_ang*)

Set a prescribed motion defined by a rigid rotor motion. Arguments are identical to elsA rotor motion.

Exists also as an in-place version (_setPrescribedMotion2) which modifies a and returns None.

Parameters

- **a** (`[array, list of arrays] or [pyTree, base, zone, list of zones]`) – Input data
- **transl_speed** (`a 3-tuple of floats`) – translation speed
- **psi0** (`float`) – initial pitch angle (in degrees)
- **psi0_b** (`float`) – angle for blade position wrt leading blade (in degrees)
- **alp_pnt** (`a 3-tuple of floats`) – origin of rotor shaft
- **alp_vct** (`a 3-tuple of floats`) – axis of rotor shaft
- **alp0** (`float`) – rotor shaft angle (in degrees)
- **rot_pnt** (`3-tuple of floats`) – rotation center
- **rot_vct** (`3-tuple of floats`) – rotation axis
- **rot_omg** (`float`) – rotor angular velocity (in radians per sec)
- **del_pnt** (`3-tuple of floats`) – origin of lead-lag

- **del_vct** (3-tuple of floats) – lead-lag axis
- **del0** (float) – lead-lag angle (in degrees)
- **delc** (tuple of floats) – cosine part of harmonics for lead-lag
- **dels** (tuple of floats) – sine part of harmonics for lead-lag
- **bet_pnt** (3-tuple of floats) – origin of flapping motion
- **bet_vct** (3-tuple of floats) – flapping axis
- **bet0** (float) – flapping angle (in degrees)
- **betc** (tuple of floats) – cosine part of harmonics for conicity
- **bets** (tuple of floats) – sine part of harmonics for conicity
- **tet_pnt** (3-tuple of floats) – origin of pitching motion
- **tet_vct** (3-tuple of floats) – pitching axis
- **tet0** (float) – collective pitch angle (in degrees)
- **tetc** (tuple of floats) – cyclic pitch cosine part
- **tets** (tuple of floats) – cyclic pitch sine part
- **span_vct** (3-tuple of floats) – reference blade spanwise axis
- **pre_lag_pnt** (3-tuple of floats) – origin of pre-lag
- **pre_lag_vct** (3-tuple of floats) – pre-lag axis
- **pre_lag_ang** (float) – pre-lag angle (in degrees)
- **pre_con_pnt** (3-tuple of floats) – origin of pre-conicity
- **pre_con_vct** (3-tuple of floats) – pre-conicity axis
- **pre_con_ang** (float) – pre-conicity angle (in degrees)

Example of use:

- Set a prescribed motion of type 2 (pyTree):

```
# - setPrescribedMotion2 (pyTree) -
# Motion defined by a rotor motion
import RigidMotion.PyTree as R
import Converter.PyTree as C
import Generator.PyTree as G

# Mime une pale suivant x, quart avant
a = G.cart((0.2,-0.075,0), (0.01,0.01,0.1), (131,11,1))
RotorMotion={'Motion_Blade1':{'initial_angles' : [0.,0], #PSI0,PSI0_b
```

(continues on next page)

(continued from previous page)

```
                              'alp0': -12.013,'alp_pnt' : [0.,0.,0.],
↪'alp_vct':[0.,1.,0.],
                              'rot_pnt' : [0.,0.,0.],'rot_vct':[0.,0.,1.
↪],'rot_omg':104.71,
                              'span_vct' : [1.,0.,0.],
                              'pre_lag_pnt' : [0.075,0.,0.],'pre_lag_vct
↪' : [0.,0.,1.],'pre_lag_ang' : -4.,
                              'pre_con_pnt' : [0.,0.,0.],'pre_con_vct' :␣
↪[0.,1.,0.],'pre_con_ang' : 0.,
                              'del_pnt' : [0.075,0.,0.],'del_vct' : [0.,
↪0.,1.],'del0' : -0.34190,
                              'del1c' : 0.48992E-01 , 'del1s': -0.95018E-
↪01,
                              'bet_pnt' : [0.076,0.,0.],'bet_vct' : [0.,
↪1.,0.],'bet0' : -2.0890,
                              'bet1c' : 3.4534, 'bet1s' : 0.0,
                              'tet_pnt' : [0.156,0.,0.],'tet_vct' : [1.,
↪0.,0.],'tet0' : 12.807,
                              'tet1c' : 1.5450, 'tet1s' : -3.4534}}

dictBlade = RotorMotion["Motion_Blade1"]
init_angles = dictBlade["initial_angles"]
psi0 = init_angles[0]; psi0_b = init_angles[1]
transl_speed = (-87.9592,0.,0.)
alp_pnt = dictBlade["alp_pnt"]
alp_vct = dictBlade["alp_vct"]
alp0 = dictBlade["alp0"]
rot_pnt = dictBlade["rot_pnt"]
rot_vct = dictBlade["rot_vct"]
rot_omg = dictBlade["rot_omg"]
del_pnt = dictBlade["del_pnt"]
del_vct = dictBlade["del_vct"]
del0 = dictBlade["del0"]
delc = (dictBlade["del1c"],)
dels = (dictBlade["del1s"],)
bet_pnt = dictBlade["bet_pnt"]
bet_vct = dictBlade["bet_vct"]
bet0 = dictBlade["bet0"]
betc = (dictBlade["bet1c"],)
bets = (dictBlade["bet1s"],)
tet_pnt = dictBlade["tet_pnt"]
tet_vct = dictBlade["tet_vct"]
```

(continues on next page)

(continued from previous page)

```
tet0 = dictBlade["tet0"]
tetc = (dictBlade["tet1c"],)
tets = (dictBlade["tet1s"],)
span_vct = dictBlade['span_vct']
pre_lag_pnt = dictBlade["pre_lag_pnt"]
pre_lag_vct = dictBlade["pre_lag_vct"]
pre_lag_ang = dictBlade["pre_lag_ang"]
pre_con_pnt = dictBlade["pre_con_pnt"]
pre_con_vct = dictBlade["pre_con_vct"]
pre_con_ang = dictBlade["pre_con_ang"]
R._setPrescribedMotion2(a, 'Motion_Blade1', transl_speed=transl_speed,
                        psi0=psi0, psi0_b=psi0_b,
                        alp_pnt=alp_pnt, alp_vct=alp_vct, alp0=alp0,
                        rot_pnt=rot_pnt, rot_vct=rot_vct, rot_omg=rot_
↪omg,
                        del_pnt=del_pnt, del_vct=del_vct, del0=del0,
                        delc=delc, dels=dels,
                        bet_pnt=bet_pnt, bet_vct=bet_vct, bet0=bet0,
                        betc=betc, bets=bets,
                        tet_pnt=tet_pnt, tet_vct=tet_vct, tet0=tet0,
                        tetc=tetc, tets=tets,
                        span_vct=span_vct,
                        pre_lag_pnt=pre_lag_pnt, pre_lag_vct=pre_lag_vct,
↪ pre_lag_ang=pre_lag_ang,
                        pre_con_pnt=pre_con_pnt, pre_con_vct=pre_con_vct,
↪ pre_con_ang=pre_con_ang)
C.convertPyTree2File(a, 'out.cgns')
```

RigidMotion.PyTree.**setPrescribedMotion3**(*a*, *motionName*, *transl_speed*, *axis_pnt*, *axis_vct*, *omega*)

Set a precribed motion defined by a constant speed rotation and constant translation vector. omega is in rad/time unit. Since rotation is applied before translation, the center of rotation (axis_pnt) is moving with translation speed also.

Exists also as an in-place version (_setPrescribedMotion3) which modifies a and returns None.

Parameters

- **a** ([array, list of arrays] or [pyTree, base, zone, list of zones]) – Input data
- **transl_speed** (tuple of 3 floats) – translation vector

- **axis_pnt** (tuple of 3 floats) – rotation axis (constant in translated frame)
- **axis_vect** (tuple of 3 floats) – vector axis (constant in traslated frame)
- **omega** (float) – constant rotation speed

Example of use:

- Set a prescribed motion of type 3 (pyTree):

```
# - setPrescribedMotion3 (pyTree) -
# Motion defined by a constant rotation and translation speed
import RigidMotion.PyTree as R
import Converter.PyTree as C
import Geom.PyTree as D

a = D.sphere((1.2,0.,0.), 0.2, 30)
a = R.setPrescribedMotion3(a, 'mot', transl_speed=(1,0,0))

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

3.1 General functions

RigidMotion.PyTree.**evalPosition**(*a*, *time*)

Evaluate the position at time t according to a motion. The motion must be defined in a with setPrescribedMotion. If GridCoordinates#Init is present, it is used to compute position. Otherwise, Grid coordinates in a must be the coordinates at time=0.

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

Parameters

- **a** ([pyTree, base, zone, list of zones]) – input data
- **time** (float) – evaluation time

Returns
reference copy of a

Return type
identical to input

Example of use:

- Evaluate position (pyTree):

```python
# - evalPosition (PyTree) -
import RigidMotion.PyTree as R
import Generator.PyTree as G
import Converter.PyTree as C
from math import *

# Coordonnees du centre de rotation dans le repere absolu
def centerAbs(t): return [t, 0, 0]

# Coordonnees du centre de la rotation dans le repere entraine
def centerRel(t): return [5, 5, 0]

# Matrice de rotation
def rot(t):
    omega = 0.1
    m = [[cos(omega*t), -sin(omega*t), 0],
         [sin(omega*t), cos(omega*t),  0],
         [0,            0,             1]]
    return m

# Mouvement complet
def F(t): return (centerAbs(t), centerRel(t), rot(t))

a = G.cart((0,0,0), (1,1,1), (11,11,2))

# Move the mesh
time = 3.
b = R.evalPosition(a, time, F); b[0]='moved'
C.convertPyTree2File([a,b], "out.cgns")
```

Evaluate position at given time, when motion is described by a function. F(t) is a function describing motion. F(t) = (centerAbs(t), centerRel(t), rot(t)), where centerAbs(t) are the coordinates of the rotation center in the absolute frame, centerRel(t) are the coordinates of the rotation center in the relative (that is array's) frame and rot(t), the rotation matrix.

Parameters

- **a** ([pyTree, base, zone, list of zones]) – input data
- **time** (float) – evaluation time
- **F** (python function) – motion function

Returns

reference copy of a

Return type
identical to input

Example of use:

- Evaluate position with function (pyTree):

```
# - evalPosition pour motion 2 (pyTree) -
# Rotor motion
import RigidMotion.PyTree as R
import Converter.PyTree as C
import Generator.PyTree as G

time0 = 0.01
a = G.cart((0.2,-0.075,0), (0.01,0.01,0.1), (131,11,1))
# Mettre tous les parametres
RotorMotion={'Motion_Blade1':{'initial_angles' : [0.,0],#PSI0,PSI0_b
                              'alp0': -12.013,'alp_pnt' : [0.,0.,0.],
↪'alp_vct':[0.,1.,0.],
                              'rot_pnt' : [0.,0.,0.],'rot_vct':[0.,0.,1.
↪],'rot_omg':104.71,
                              'span_vct' : [1.,0.,0.],
                              'pre_lag_pnt' : [0.075,0.,0.],'pre_lag_vct
↪' : [0.,0.,1.],'pre_lag_ang' : -4.,
                              'pre_con_pnt' : [0.,0.,0.],'pre_con_vct' :␣
↪[0.,1.,0.],'pre_con_ang' : 0.,
                              'del_pnt' : [0.075,0.,0.],'del_vct' : [0.,
↪0.,1.],'del0' : -0.34190,
                              'del1c' : 0.48992E-01 , 'del1s': -0.95018E-
↪01,
                              'bet_pnt' : [0.076,0.,0.],'bet_vct' : [0.,
↪1.,0.],'bet0' : -2.0890,
                              'bet1c' : 3.4534, 'bet1s' : 0.0,
                              'tet_pnt' : [0.156,0.,0.],'tet_vct' : [1.,
↪0.,0.],'tet0' : 12.807,
                              'tet1c' : 1.5450, 'tet1s' : -3.4534}}

dictBlade = RotorMotion["Motion_Blade1"]
init_angles = dictBlade["initial_angles"]
psi0 = init_angles[0]; psi0_b = init_angles[1]
transl_speed = (-87.9592,0.,0.)
alp_pnt = dictBlade["alp_pnt"]
alp_vct = dictBlade["alp_vct"]
alp0 = dictBlade["alp0"]
rot_pnt = dictBlade["rot_pnt"]
```

(continues on next page)

(continued from previous page)

```
rot_vct = dictBlade["rot_vct"]
rot_omg = dictBlade["rot_omg"]
del_pnt = dictBlade["del_pnt"]
del_vct = dictBlade["del_vct"]
del0 = dictBlade["del0"]
delc = (dictBlade["del1c"],)
dels = (dictBlade["del1s"],)
bet_pnt = dictBlade["bet_pnt"]
bet_vct = dictBlade["bet_vct"]
bet0 = dictBlade["bet0"]
betc = (dictBlade["bet1c"],)
bets = (dictBlade["bet1s"],)
tet_pnt = dictBlade["tet_pnt"]
tet_vct = dictBlade["tet_vct"]
tet0 = dictBlade["tet0"]
tetc = (dictBlade["tet1c"],)
tets = (dictBlade["tet1s"],)
span_vct = dictBlade['span_vct']
pre_lag_pnt = dictBlade["pre_lag_pnt"]
pre_lag_vct = dictBlade["pre_lag_vct"]
pre_lag_ang = dictBlade["pre_lag_ang"]
pre_con_pnt = dictBlade["pre_con_pnt"]
pre_con_vct = dictBlade["pre_con_vct"]
pre_con_ang = dictBlade["pre_con_ang"]
R._setPrescribedMotion2(a, 'Motion_Blade1', transl_speed=transl_speed,
                        psi0=psi0, psi0_b=psi0_b,
                        alp_pnt=alp_pnt, alp_vct=alp_vct, alp0=alp0,
                        rot_pnt=rot_pnt, rot_vct=rot_vct, rot_omg=rot_
↪omg,
                        del_pnt=del_pnt, del_vct=del_vct, del0=del0,
                        delc=delc, dels=dels,
                        bet_pnt=bet_pnt, bet_vct=bet_vct, bet0=bet0,
                        betc=betc, bets=bets,
                        tet_pnt=tet_pnt, tet_vct=tet_vct, tet0=tet0,
                        tetc=tetc, tets=tets,
                        span_vct=span_vct,
                        pre_lag_pnt=pre_lag_pnt, pre_lag_vct=pre_lag_vct,
↪ pre_lag_ang=pre_lag_ang,
                        pre_con_pnt=pre_con_pnt, pre_con_vct=pre_con_vct,
↪ pre_con_ang=pre_con_ang)

b = R.evalPosition(a, time=time0); b[0]='moved'
C.convertPyTree2File(b, "out.cgns")
```

`RigidMotion.PyTree.`**`evalGridSpeed`**(*a*, *time*)

Evaluate grid speed at given time. The position must already have been evaluated at this time.

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

Parameters

- **a** (`[pyTree, base, zone, list of zones]`) – input data
- **time** (`float`) – evaluation time

Returns

reference copy of a

Return type

identical to input

Example of use:

- Evaluate speed (pyTree):

```
# - evalGridSpeed (pyTree) -
import RigidMotion.PyTree as R
import Converter.PyTree as C
import Geom.PyTree as D

a = D.sphere((1.2,0.,0.), 0.2, 30)
a = R.setPrescribedMotion3(a, 'motion', transl_speed=(1,0,0))
b = R.evalPosition(a, time=0.1)
R._evalGridSpeed(b, time=0.1)
C.convertPyTree2File(b, 'out.cgns')
```

`RigidMotion.PyTree.`**`copyGrid2GridInit`**(*a*, *mode=0*)

Copy GridCoordinates to GridCoordinates#Init. If mode=0, only if grid has a TimeMotion node. If mode=1, always copy.

Exists also as an in-place version (_copyGrid2GridInit) which modifies a and returns None.

Parameters

- **a** (`[pyTree, base, zone, list of zones]`) – input data
- **mode** (`0 or 1`) – behaviour

Returns
reference copy of a

Return type
identical to input

Example of use:

- Copy GridCoordinates to GridCoordinates#Init (pyTree):

```
# - copyGrid2GridInit (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import RigidMotion.PyTree as R

a = G.cart((0,0,0), (1,1,1), (10,10,10))
R._setPrescribedMotion3(a, 'motion', transl_speed=(1,0,0))
R._copyGrid2GridInit(a)
C.convertPyTree2File(a, 'out.cgns')
```

RigidMotion.PyTree.**copyGridInit2Grid**(*a*)

Copy GridCoordinates#Init to GridCoordinates if it exists.

Exists also as an in-place version (_copyGridInit2Grid) which modifies a and returns None.

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

Returns
reference copy of a

Return type
identical to input

Example of use:

- Copy GridCoordinates#Init to GridCoordinates (pyTree):

```
# - copyGridInit2Grid (pyTree) -
import Converter.PyTree as C
import Generator.PyTree as G
import RigidMotion.PyTree as R

a = G.cart((0,0,0), (1,1,1), (10,10,10))
R._copyGrid2GridInit(a, mode=1)
R._copyGridInit2Grid(a)
C.convertPyTree2File(a, 'out.cgns')
```

CHAPTER FOUR

INDEX

- genindex
- modindex
- search