Dist2Walls Documentation

Release 3.1

/ELSA/MU-10019/V3.1

May 28, 2020

CONTENTS

CHAPTER

ONE

PREAMBLE

Dist2Walls gathers efficient algorithms for computing the distance fields for arrays (as defined in Converter documentation) or for CGNS/python tree (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 Dist2Walls module:

```
import Dist2Walls
```

For use with the pyTree interface:

```
import Dist2Walls.PyTree
```

CHAPTER

TWO

LIST OF FUNCTIONS

– Wall distance computation

Dist2Walls.distance2Walls(zones, bodies[, ...])	Compute distance to walls.

CHAPTER

THREE

CONTENTS

3.1 Wall distance computation

Dist2Walls.**distance2Walls**(*a*, *bodies*, *type='ortho'*, *loc='centers'*, *signed=0*, *dim=3*)

Computes the distance field from a set of bodies. compute the distance field located at nodes or centers of zone a (or zones in A), provided a list of surfaces defining the bodies to which the distance is computed.

Two algorithms are available:

- type='ortho' means a distance computed by an orthogonal projection to the surface faces defined by bodies.
- type='mininterf' returns the minimum distance of the point to the vertices of bodies.

If loc='nodes', returns a distance computed at nodes of a (A), else if loc='centers, distance is computed at cell centers of a (A).

Parameter 'signed'=1 enables to compute a signed distance (negative inside bodies). When using signed distances, each body in bodies list must be a closed and watertight surface. In array version, cellnbodies provides the 'cellN' field for any vertex in bodies. Default value is 1. The algorithm 'ortho' does not take into account a body face if cellN=0 for all the vertices of that face. The algorithm 'mininterf' does not compute the distance to a vertex of cellN=0.

Parameters

- **a** (*[array, list of arrays] or [pyTree, base, zone, list of zones]*) – input data
- **bodies** (*[array, list of arrays] or [pyTree, base, zone, list of zones]*) – body definition
- **type** (*string*) – type of wall distance computation in ['ortho', 'mininterf']
- **loc** (*string*) – location of distance field in ['nodes', 'centers']

- **signed** (*int*) – if 0 absolut distance, if 1 signed distance (negative inside)

In the pyTree version, 'cellN' variable must be stored in bodies directly. If loc='nodes', the distance field is stored as a 'TurbulentDistance' field located at nodes, and if loc='centers', it is stored in nodes located at centers.

Exists also as an in-place version (_distance2Walls) that modifies a and returns None.

Example of use:

- Compute distance to walls (array):

```
# - distance2Walls (array) -
import Dist2Walls
import Generator as G
import Converter as C
import Geom as D

# Bloc dont on cherche la distance a la paroi
a = G.cart((0.,0.,0.),(0.1,0.1,0.1),(10,10,10))

# Paroi
sphere = D.sphere((1.2,0.,0.), 0.2, 30)
cellN = C.initVars(sphere,'cellN',1.)
# Calcul de la distance a la paroi
dist = Dist2Walls.distance2Walls(a, [sphere], cellnbodies=[cellN],
                                 loc='centers',type='ortho')
ac = C.node2Center(a)
ac = C.addVars([ac, dist])
C.convertArrays2File([ac], 'out.plt')
```

- Compute distance to walls (pyTree):

```
# - distance2Walls (pyTree) -
import Dist2Walls.PyTree as Dist2Walls
import Generator.PyTree as G
import Converter.PyTree as C
import Geom.PyTree as D

a = G.cart((0.,0.,0.),(0.1,0.1,0.1),(10,10,10))
sphere = D.sphere((1.2,0.,0.),0.2,100)
t = C.newPyTree(['Base',a])
t = Dist2Walls.distance2Walls(t, sphere)
C.convertPyTree2File(t, 'out.cgns')
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search