Py4Incompact3D Documentation

Release 0.0.0

Yorgos Deskos
Paul Bartholomew

October 03, 2018

CONTENTS:

CHAPTER

ONE

INTRODUCTION

Py4Incompact3D is a library for postprocessig data produced by Xcompact3D simulations. The aim of this project is to facillitate automated postprocessing of Xcompact3D simulations by providing, at first:

- Mesh class: this stores the domain data for the simulation
- Case class: this stores the information of the case: boundary conditions, fields etc.

With these building blocks, complex postprocessing tools may be built - for example, derivative calculateors to compute the vorticity and Q-criterion given the velocity field.

Installation

- Clone the git repository to a location on your `${PYTHONPATH}`
- Test module can be imported by python interpreter: `import Py4Incompact3D`

Documentation

Documentation of functions can be found under *doc/build/latex/*.

To regenerate documentation, from the project root type `make -C doc/ latexpdf` (requires sphinx).

Contributing

It is hoped that users of Xcompact3D will find this library useful and contribute to its development, for instance by adding additional functionality.

CHAPTER

TWO

API

class Py4Incompact3D.postprocess.postprocess.**Postprocess**(*input_file*)
Postprocess is the highest level class of the Py4Incompact3D package. Import this class and instantiate it with a path to an input file to begin running Py4Incompact3D. Use the ‘‘fields” attribute to access other objects within the model.

inputs: input_file: str - path to the nml input file

outputs: self: post - an instantiated post object

class Py4Incompact3D.postprocess.mesh.**Mesh**(*instance_dictionary*)
Mesh is a model object representing

compute_derivvars()
Compute variables required by derivative functions.

Py4Incompact3D.deriv.deriv.**compute_rhs**(*mesh*, *field*)
Compute the rhs for the derivative.

Parameters

- **mesh** (Py4Incompact3D.postprocess.mesh.Mesh) – The mesh on which derivatives are taken.
- **field** – The field for the variable who’s derivative we want.

Returns rhs – the right-hand side vector.

Return type numpy.ndarray

Py4Incompact3D.deriv.deriv.**deriv**(*postproc*, *mesh*, *phi*, *axis*)
Take the derivative of field ‘phi’ along axis.

Parameters

- **postproc** (Py4Incompact3D.postprocess.postprocess.Postprocess) – The basic Postprocess object.
- **mesh** (Py4Incompact3D.postprocess.mesh.Mesh) – The mesh on which derivatives are taken.
- **phi** (*str*) – The name of the variable who’s derivative we want.
- **axis** (*int*) – A number indicating direction in which to take derivative: 1=x; 2=y; 3=z.

Returns dphidx – the derivative

Return type numpy.ndarray

Py4Incompact3D.deriv.deriv.**tdma**(*a*, *b*, *c*, *rhs*)
The Tri-Diagonal Matrix Algorithm.

Solves tri-diagonal matrices using TDMA.

Parameters

- **a** (*numpy.ndarray*) –
- **b** –
- **c** –
- **rhs** –

Returns dphidx – the derivative

Return type numpy.ndarray

d

p

C

D

M

P

T