
Py4Incompact3D Documentation

Release 0.0.0

**Yorgos Deskos
Paul Bartholomew**

October 03, 2018

CONTENTS:

1	Introduction	1
1.1	Installation	1
1.2	Documentation	1
1.3	Contributing	1
2	API	3
	Python Module Index	5
	Index	7

INTRODUCTION

Py4Incompact3D is a library for postprocessing data produced by Xcompact3D simulations. The aim of this project is to facilitate 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 calculators 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.

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, axis)`

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 whose derivative we want.
- **axis** (`int`) – A number indicating direction in which to take derivative: 0=x; 1=y; 2=z.

Returns `rhs` – the right-hand side vector.

Return type `numpy.ndarray`

`Py4Incompact3D.deriv.deriv.compute_rhs_0(mesh, field, axis)`

Compute the rhs for the derivative for periodic BCs.

Parameters

- **mesh** (`Py4Incompact3D.postprocess.mesh.Mesh`) – The mesh on which derivatives are taken.
- **field** – The field for the variable whose derivative we want.
- **axis** (`int`) – A number indicating direction in which to take derivative: 0=x; 1=y; 2=z.

Returns `rhs` – the right-hand side vector.

Return type `numpy.ndarray`

`Py4Incompact3D.deriv.deriv.compute_rhs_1(mesh, field, axis)`

Compute the rhs for the derivative for free slip BCs.

Parameters

- **mesh** (`Py4Incompact3D.postprocess.mesh.Mesh`) – The mesh on which derivatives are taken.
- **field** – The field for the variable whose derivative we want.
- **axis** (`int`) – A number indicating direction in which to take derivative: 0=x; 1=y; 2=z.

Returns `rhs` – the right-hand side vector.

Return type `numpy.ndarray`

`Py4Incompact3D.deriv.deriv.compute_rhs_2(mesh, field, axis)`

Compute the rhs for the derivative for Dirichlet BCs.

Parameters

- **mesh** (`Py4Incompact3D.postprocess.mesh.Mesh`) – The mesh on which derivatives are taken.
- **field** – The field for the variable whose derivative we want.
- **axis** (`int`) – A number indicating direction in which to take derivative: 0=x; 1=y; 2=z.

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 whose derivative we want.
- **axis** (`int`) – A number indicating direction in which to take derivative: 0=x; 1=y; 2=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

`deriv`, 3

p

`Py4Incompact3D.deriv.deriv`, 3

`Py4Incompact3D.postprocess.mesh`, 3

`Py4Incompact3D.postprocess.postprocess`,
3

C

compute_derivvars() (Py4Incompact3D.postprocess.mesh.Mesh
method), 3
 compute_rhs() (in module Py4Incompact3D.deriv.deriv),
3
 compute_rhs_0() (in module
Py4Incompact3D.deriv.deriv), 3
 compute_rhs_1() (in module
Py4Incompact3D.deriv.deriv), 3
 compute_rhs_2() (in module
Py4Incompact3D.deriv.deriv), 4

D

deriv (module), 3
 deriv() (in module Py4Incompact3D.deriv.deriv), 4

M

Mesh (class in Py4Incompact3D.postprocess.mesh), 3

P

Postprocess (class in Py4Incompact3D.postprocess.postprocess),
3
 Py4Incompact3D.deriv.deriv (module), 3
 Py4Incompact3D.postprocess.mesh (module), 3
 Py4Incompact3D.postprocess.postprocess (module), 3

T

tdma() (in module Py4Incompact3D.deriv.deriv), 4