TOSCA

Generated by Doxygen 1.9.1

1 TOSCA 1

1.1 Recent Highlights/Additions . . . 2
1.2 Executables . . . 2
1.3 Boundary Conditions . . . 2
1.4 Wall Models . . . 2
1.5 Turbulence Models . . . 2
1.6 Turbine Models . . . 2
1.7 IBM Method . . . 3
1.8 Acquisition System . . . 3
1.9 Future Implementations: . . . 3
1.10 Notes: . . . 3
1.11 Installation: . . . 3
1.12 Contribute to the TOSCA Project . . . 4
1.13 Paraview-Catalyst2 OS-Rendering . . . 5
1.14 Usage . . . 5
1.15 What does it do . . . 5
1.16 Installation: . . . 5
1.17 1. install Catalyst2 . . . 6
1.18 2. install Paraview-5.11 . . . 6
1.19 3. Re-compile TOSCA . . . 7
1.20 4. Running . . . 7

2 Class Index 9

2.1 Class List . . . 9

3 File Index 13

3.1 File List . . . 13

4 Class Documentation 15

4.1 abl_ Struct Reference . . . 15
4.2 access_ Struct Reference . . . 21
4.2.1 Detailed Description . . . 22
4.3 Acell Struct Reference . . . 22
4.3.1 Detailed Description . . . 22
4.4 acquisition_ Struct Reference . . . 22
4.4.1 Detailed Description . . . 23
4.5 ADM Struct Reference . . . 23
4.5.1 Detailed Description . . . 25
4.6 AFM Struct Reference . . . 25
4.6.1 Detailed Description . . . 26
4.7 ALM Struct Reference . . . 26
4.7.1 Detailed Description . . . 27
4.8 anemometer Struct Reference . . . 27

4.8.1 Detailed Description . . . 28
4.9 bladeAeroInfo Struct Reference . . . 28
4.9.1 Detailed Description . . . 29
4.10 boundingBox Struct Reference . . . 29
4.10.1 Detailed Description . . . 29
4.11 Cabot Struct Reference . . . 29
4.11.1 Detailed Description . . . 30
4.12 cellIds Struct Reference . . . 30
4.12.1 Detailed Description . . . 30
4.13 cellList Struct Reference . . . 30
4.13.1 Detailed Description . . . 30
4.14 cellNode Struct Reference . . . 31
4.15 clock_ Struct Reference . . . 31
4.16 Cmpnts Struct Reference . . . 32
4.16.1 Detailed Description . . . 32
4.17 constants_ Struct Reference . . . 32
4.18 Cpt2D Struct Reference . . . 32
4.18.1 Detailed Description . . . 33
4.19 Dcell Struct Reference . . . 33
4.19.1 Detailed Description . . . 33
4.20 domain_ Struct Reference . . . 33
4.20.1 Detailed Description . . . 34
4.21 elementBox Struct Reference . . . 34
4.22 Face Struct Reference . . . 35
4.23 farm_ Struct Reference . . . 35
4.23.1 Detailed Description . . . 36
4.24 flags_ Struct Reference . . . 36
4.24.1 Detailed Description . . . 36
4.25 foilInfo Struct Reference . . . 37
4.25.1 Detailed Description . . . 37
4.26 HalfEdge Struct Reference . . . 37
4.27 ibm_ Struct Reference . . . 37
4.27.1 Detailed Description . . . 38
4.28 ibmFluidCell Struct Reference . . . 39
4.29 ibmMesh Struct Reference . . . 40
4.30 ibmNode Struct Reference . . . 40
4.31 ibmObject Struct Reference . . . 41
4.32 ibmPitchMotion Struct Reference . . . 42
4.33 ibmRotation Struct Reference . . . 42
4.34 ibmSineMotion Struct Reference . . . 42
4.35 inflowData Struct Reference . . . 43
4.35.1 Detailed Description . . . 43

4.36 inletFunctions Struct Reference . . . 43
4.36.1 Detailed Description . . . 43
4.37 inletFunctionTypes Struct Reference . . . 44
4.37.1 Detailed Description . . . 46
4.38 io_ Struct Reference . . . 46
4.38.1 Detailed Description . . . 47
4.39 les_ Struct Reference . . . 47
4.39.1 Detailed Description . . . 48
4.40 list Struct Reference . . . 48
4.40.1 Detailed Description . . . 48
4.41 LogLawAPG Struct Reference . . . 49
4.42 mapInfo Struct Reference . . . 49
4.42.1 Detailed Description . . . 49
4.43 mesh_ Struct Reference . . . 49
4.43.1 Detailed Description . . . 51
4.44 nacelleModel Struct Reference . . . 51
4.44.1 Detailed Description . . . 52
4.45 node Struct Reference . . . 52
4.45.1 Detailed Description . . . 52
4.46 overset_ Struct Reference . . . 52
4.46.1 Detailed Description . . . 53
4.47 oversetMotion Struct Reference . . . 53
4.47.1 Detailed Description . . . 54
4.48 patchVectorField Struct Reference . . . 54
4.48.1 Detailed Description . . . 54
4.49 peqn_ Struct Reference . . . 54
4.49.1 Detailed Description . . . 56
4.50 postProcess Struct Reference . . . 56
4.50.1 Detailed Description . . . 56
4.51 PowerLawAPG Struct Reference . . . 57
4.51.1 Detailed Description . . . 57
4.52 precursor_ Struct Reference . . . 57
4.52.1 Detailed Description . . . 57
4.53 scalarBC Struct Reference . . . 58
4.53.1 Detailed Description . . . 58
4.54 searchBox Struct Reference . . . 58
4.55 Shumann Struct Reference . . . 59
4.55.1 Detailed Description . . . 59
4.56 simInfo_ Struct Reference . . . 60
4.57 surface Struct Reference . . . 60
4.58 symmTensor Struct Reference . . . 60
4.58.1 Detailed Description . . . 61

4.59 teqn_ Struct Reference . . . 61
4.59.1 Detailed Description . . . 62
4.60 towerModel Struct Reference . . . 62
4.60.1 Detailed Description . . . 63
4.61 UADM Struct Reference . . . 63
4.61.1 Detailed Description . . . 64
4.62 ueqn_ Struct Reference . . . 64
4.62.1 Detailed Description . . . 66
4.63 upSampling Struct Reference . . . 66
4.63.1 Detailed Description . . . 67
4.64 vectorBC Struct Reference . . . 67
4.64.1 Detailed Description . . . 68
4.65 Vertex Struct Reference . . . 68
4.66 wallModel Struct Reference . . . 68
4.66.1 Detailed Description . . . 69
4.67 windTurbine Struct Reference . . . 69
4.67.1 Detailed Description . . . 73

5 File Documentation **75**

5.1 src/abl.c File Reference . . . 75
5.1.1 Detailed Description . . . 75
5.2 src/acquisition.c File Reference . . . 76
5.2.1 Detailed Description . . . 77
5.2.2 Function Documentation . . . 78
5.2.2.1 read3LMFields() . . . 78
5.3 src/boundary.c File Reference . . . 78
5.3.1 Detailed Description . . . 79
5.4 src/clock.c File Reference . . . 79
5.4.1 Detailed Description . . . 79
5.5 src/ibm.c File Reference . . . 79
5.5.1 Detailed Description . . . 82
5.6 src/ibmInput.c File Reference . . . 82
5.6.1 Detailed Description . . . 83
5.6.2 Function Documentation . . . 83
5.6.2.1 readIBMSurfaceFileAbaqusInp() . . . 83
5.7 src/include/abl.h File Reference . . . 83
5.7.1 Detailed Description . . . 83
5.8 src/include/acquisition.h File Reference . . . 84
5.8.1 Detailed Description . . . 86
5.8.2 Function Documentation . . . 86
5.8.2.1 read3LMFields() . . . 86
5.9 src/include/base.h File Reference . . . 86

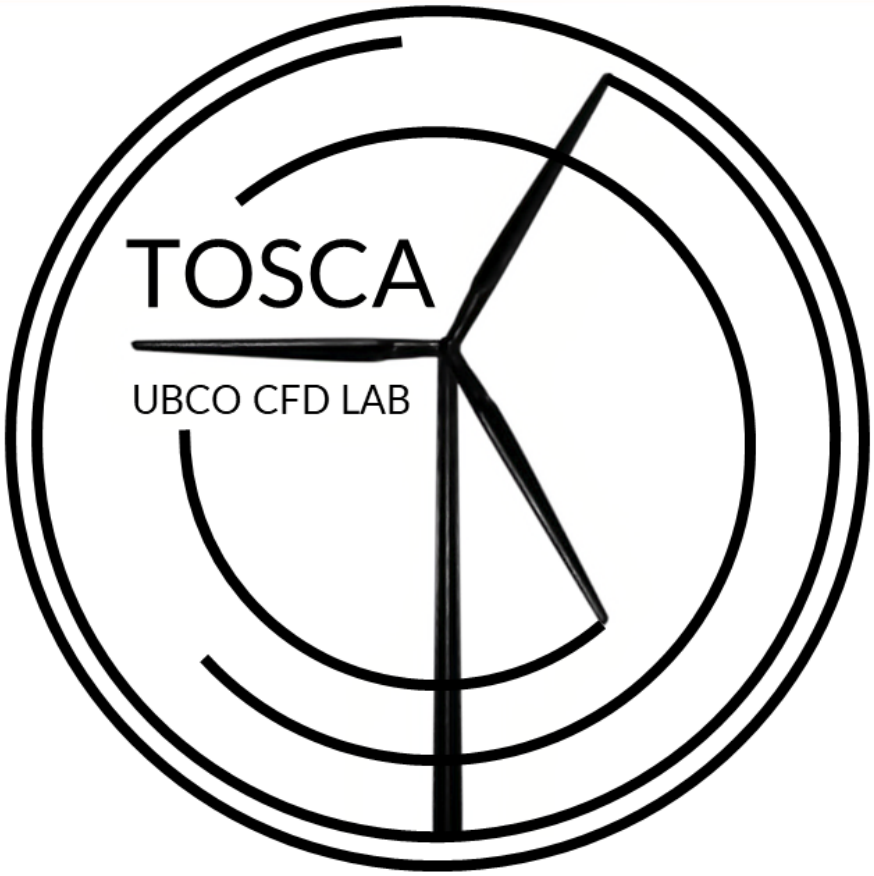
5.9.1 Detailed Description . . . 87
5.10 src/include/boundary.h File Reference . . . 87
5.10.1 Detailed Description . . . 88
5.11 src/include/clock.h File Reference . . . 88
5.11.1 Detailed Description . . . 89
5.12 src/include/domain.h File Reference . . . 89
5.12.1 Detailed Description . . . 89
5.13 src/include/ibm.h File Reference . . . 89
5.13.1 Detailed Description . . . 93
5.14 src/include/inflow.h File Reference . . . 93
5.14.1 Detailed Description . . . 93
5.15 src/include/initialField.h File Reference . . . 94
5.15.1 Detailed Description . . . 94
5.15.2 Function Documentation . . . 94
5.15.2.1 SetInitialField() . . . 94
5.16 src/include/initialization.h File Reference . . . 95
5.16.1 Detailed Description . . . 95
5.16.2 Function Documentation . . . 95
5.16.2.1 SetInitialField() . . . 95
5.16.2.2 simulationInitialize() . . . 96
5.17 src/include/inline.h File Reference . . . 96
5.17.1 Detailed Description . . . 101
5.17.2 Function Documentation . . . 101
5.17.2.1 computeEm() . . . 101
5.18 src/include/io.h File Reference . . . 101
5.18.1 Detailed Description . . . 103
5.19 src/include/mesh.h File Reference . . . 104
5.19.1 Detailed Description . . . 104
5.20 src/include/objects.h File Reference . . . 104
5.20.1 Detailed Description . . . 104
5.21 src/include/overset.h File Reference . . . 104
5.21.1 Detailed Description . . . 106
5.22 src/include/peqn.h File Reference . . . 106
5.22.1 Detailed Description . . . 107
5.23 src/include/precursor.h File Reference . . . 107
5.23.1 Detailed Description . . . 108
5.24 src/include/teqn.h File Reference . . . 108
5.24.1 Detailed Description . . . 109
5.25 src/include/tosca2PV.h File Reference . . . 109
5.25.1 Detailed Description . . . 113
5.25.2 Function Documentation . . . 113
5.25.2.1 getTimeList() . . . 113

5.26 src/include/ueqn.h File Reference . . . 113
5.26.1 Detailed Description . . . 114
5.27 src/include/wallfunctions.h File Reference . . . 114
5.27.1 Detailed Description . . . 115
5.28 src/include/wallmodel.h File Reference . . . 115
5.28.1 Detailed Description . . . 115
5.29 src/initialField.c File Reference . . . 116
5.29.1 Detailed Description . . . 116
5.29.2 Function Documentation . . . 116
5.29.2.1 SetInitialField() . . . 117
5.30 src/initialization.c File Reference . . . 117
5.30.1 Detailed Description . . . 118
5.30.2 Function Documentation . . . 118
5.30.2.1 simulationInitialize() . . . 118
5.31 src/io.c File Reference . . . 118
5.31.1 Detailed Description . . . 120
5.31.2 Function Documentation . . . 120
5.31.2.1 getTimeList() . . . 120
5.32 src/les.c File Reference . . . 121
5.32.1 Detailed Description . . . 121
5.32.2 Function Documentation . . . 121
5.32.2.1 UpdateCs() . . . 121
5.33 src/mesh.c File Reference . . . 122
5.33.1 Detailed Description . . . 122
5.34 src/overset.c File Reference . . . 122
5.34.1 Detailed Description . . . 123
5.35 src/peqn.c File Reference . . . 123
5.35.1 Detailed Description . . . 125
5.36 src/precursor.c File Reference . . . 125
5.36.1 Detailed Description . . . 126
5.37 src/teqn.c File Reference . . . 126
5.37.1 Detailed Description . . . 127
5.38 src/turbines.c File Reference . . . 127
5.38.1 Detailed Description . . . 130
5.39 src/ueqn.c File Reference . . . 130
5.39.1 Detailed Description . . . 131

Index **133**

Chapter 1

TOSCA



Toolbox fOr Stratified Convective Atmospheres

1.1 Recent Highlights/Additions

```
- Direct/Indirect profile assimilation techniques to drive LES with observations/mesoscale models
- New scale-dependent LES model
- New stability dependent wall model for IBM
- IBM also for concurrent precursor (to model terrain features)
- Lateral fringe region.
```

Now we can simulate real cases, wind rotating 360 degs, with terrain and turbines, all while damping gravity waves in all directions !!

1.2 Executables

```
- tosca    : transient solver for stratified incompressible flows. Temperature stratification is accounted via
- tosca2PV : post processor for ParaView visualization. Writes data in XMF/HDF format.
```

1.3 Boundary Conditions

```
- noSlip               : on all patches
- slip                 : on all patches
- zeroGradient         : on all patches
- fixedGradient        : on all patches (only T equation)
- inflowFunction       : only on kLeft patch. Different types of inflow can be prescribed: ABL inflow using te
                         simulation that can be also periodized and/or interpolated if meshes are not consiste
- velocityWallFunction : shear stress model (only U equation). Can be prescribed on jRight and jLeft patches
- thetaWallFunction    : potential temperature wall model (only T equation). Can be prescribed on jRight and j
```

1.4 Wall Models

```
- Shumann    : applies wall shear stress in the momentum equation according to similarity theory of Paulson an
               Velocity BC is dependent (velocityWallFunction/thetaWallFunction, type -3).
- Cabot      : prescribes velocity BC according to Cabot formulation (velocityWallFunction, type -1).
```

1.5 Turbulence Models

```
- Smagorinsky standard model (les 1).
- Dynamic Smagorinsky LES turbulence model with cubic (les 3) or lagrangian (les 4) averaging.
```

1.6 Turbine Models

```
- UADM : base actuator disk model with imposed Ct and yaw controller
- ADM  : advanced actuator disk model with rotor dynamics and yaw/pitch/rotation controllers
- ALM  : advanced actuator line model with rotor dynamics and yaw/pitch/rotation controllers (anisotropic gaus
- AFM  : anisotropic actuator farm model for coarser meshes, integral/point sampling available
```

1.7 IBM Method

```
TOSCA uses a sharp-interface IBM method with ghost cells applied on the flow side of the body. This allows for
thin bodies but complicated wall modeling. New wall models have been recently added, validated with both hi-Re
Moving IBM is also possible, we are currently validating the latter capability.
```

1.8 Acquisition System

```
- ABL wall-parallel planes-averaged statistics as a function of time
- field averages
- domain sections
- probes with advanced parallel I/O writing
- turbine data with advanced parallel I/O writing
- ABL perturbations w.r.t. reference state
- layer averaged fields (three layers available - 3LM)
- mechanical energy budgets within user-defined boxes (the code is not energy-conservative so MKE eq. is not f
```

1.9 Future Implementations:

```
- Overset with fringe interpolation
- IBM smooth normals and thin-body augmented search
```

1.10 Notes:

```
- test cases which contain an empty 'inflowDatabase' file require the inflow database. A sample database can b
  at https://drive.google.com/file/d/17F5wtI5Jc1XGh8crmOVJYVXabC8iQXq1/view?usp=sharing, simply substitute the
  file with the downloaded folder.
```

1.11 Installation:

In order to be installed, TOSCA requires a working C/C++ compiler, PETSc (version 3.14.x, 3.15.x), Open MPI (version 4.0.x, 4.1.x), HDF5 and HYPRE (needed by PETSs in order to build some of the matrix solvers we use). TOSCA has been tested with the above version combinations, it could work with other combinations or versions but it has not been tested (especially older versions). We recommend the following versions of the above libraries:

- gcc : 9.2.0 (https://gcc.gnu.org/).
- PETSc : 3.15.5 (https://ftp.mcs.anl.gov/pub/petsc/).
- Open MPI : 4.1.2 (https://www.open-mpi.org/software/ompi/v4.1/).
- HYPRE : 2.20.0 (https://github.com/hypre-space/hypre/tree/hypre_petsc) (check version in /src/CMakeLists.txt).
- HDF5 : 1.12.1 (https://www.hdfgroup.org/downloads/hdf5/).

Prior to install TOSCA, we suggest to create a folder named `Software` inside `$HOME`, where the PETSc, HYPRE and TOSCA folders will be located. In order to compile TOSCA on your system, please follow these steps:

- Check your compiler version with `gcc --version`

- Download PETSc into `$HOME/Software/`
- Download HYPRE `$HOME/Software/`
- Download Open MPI: you can download the binaries or compile from source (the latter is recommended if use `environment-modules`). If you have only one version of Open MPI installed on your system in the `/usr` directory (using sudo for example), you can omit the '--with-mpi-dir='your--path--to--mpicc'' at point 4: Open MPI will be found by the 'ld' library locator.
- Configure PETSc (will automatically compile HYPRE). We suggest the following configure options: './configure --with-fc=0 --download-f2cblaslapack --with-mpi-dir='your--path--to--mpicc' –download-hypre='your–path–to–hypre \ –with-64-bit-indices=1 –with-debugging=0`
- Make PETSc with `make all`
- Test PETSc with `make check`
- Save an environment variable that will tell TOSCA where PETSc is installed in your .bashrc: `echo "export PETSC_DIR=$HOME/your--path--to--petsc" >> $HOME/.bashrc`
- Save an environment variable that will tell TOSCA which PETSc architecture is required in your .bashrc. Note: this is the folder within $PETSC_DIR with a name beginning with "arch-". In a typical installation, it will be "arch-linux-c-opt": `echo "export PETSC_ARCH=arch-linux-c-opt" >> $↩ HOME/.bashrc`
- Add the PETSc shared libraries to your library path environment variable in your .bashrc: `echo "export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PETSC_DIR/$PETSC_ARCH/lib" >> $HOME/.bashrc`
- Reload the environment with `source $HOME/.bashrc`
- Go inside TOSCA/src directory and compile the executables with `make tosca` and `make tosca2PV`
- Test the installation by copying `tosca` and `tosca2PV` in one of the example cases and run the simulation and the post-processing with `./tosca` and `./tosca2PV` respectively. To run in parallel you have to use 'mpirun -np 'your-number-of-processors' ./tosca`

1.12 Contribute to the TOSCA Project

The TOSCA repository is open-source, so anyone can download and use the code. If you want to contribute to the project by adding code to TOSCA repository you need to open a pull-request that has to be approved by our team. In order to do so, please use the following steps:

- Clone the TOSCA package locally on your machine with `git clone https://github.↩ com/sebastipa/TOSCA.git`
- Create a new local branch with `git checkout -b your-branch-name`
- Make the desired changes to the TOSCA code, then check which files have been modified with `git status`
- Add changes to the git stack with `git add modified-files`
- Commit the changes using a short but exhaustive comment with `git commit -m "your-commit-description"`
- Push your local branch online with `git push origin your-branch-name`
- Go to github, select your branch, click on "Contribute" and open a pull-request describing the motivation of your changes, their effect on the code and the tests you performed.
- After approval of the pull-request by our team, commits will be added to the main TOSCA version
- To stay up-to-date, rebase your local master with the your new commits by first checking out in your local master branch with `git checkout master` and then rebase with `git pull --rebase origin master`
- Delete your local branch as it is not useful anymore with `git branch --delete -d your-branch-name` and operate on the master until you want to make new changes

1.13 Paraview-Catalyst2 OS-Rendering

TOSCA provides full interface with Paraview-catalyst2 through the USE_CATALYST flag in the makefile. If this is the case, the CATALYST environment variable should point to the catalyst2 installation directory. Paraview-catalyst is optional and can be disabled by setting USE_CATALYST=0

1.14 Usage

In order to activate off-screen rendering capabilities, -pvCatalyst=1 should be set in the control.dat file. A file called catalystProperties will be required inside the sampling directory. Entries to this file are

- ioType can be set to 'script' or 'general'
- outputType can be set to 'timeStep' or 'adjustableTime'
- startTime model-time at which catalyst actions start
- timeInterval acquisition period in seconds if outputType=adjustableTime or iterations if outputType=timeStep
- scriptName name of the catalyst actions python script. Only required if ioType=script

1.15 What does it do

If ioType=general, 3D fields of velocity magnitude, pressure and q-criterion are saved inside the catalyst/ folder. If ioType=script, Praview actions defined in the python script are executed and e.g. png images can be saved at runtime.

1.16 Installation:

In order to be installed, Paraview-catalyst2 requires a working C/C++ compiler, Open MPI (version 4.0.x, 4.1.x), Python3 and cmake. In order for Paraview to work, OpenGL must be available at runtime or mesa libraries are required to mimic some hardware components. These are usually available on supercomputers through the 'mesa' module, which should be loaded at runtime. Lastly, paraview and catalyst2 should be manually compiled of the system. As Paraview-5.10 contains a bug in the definition of rectilinear mesh (used by TOSCA), Paraview-5.11 or later is recommended.

Prior to install TOSCA, we suggest to create a folder named `Software` inside `$HOME`, where catalyst2 and paraview-5.11 will be located. In order to re-compile TOSCA with Paraview-catalyst2 capabilities on your system, please follow these steps:

1.17 1. install Catalyst2

- export LOCATION=$HOME/Software
- cd $LOCATION
- mkdir catalyst2 && cd catalyst2
- git clone https://gitlab.kitware.com/paraview/catalyst.git catalyst-src && cd catalyst-src
- mkdir -p build && cd build
- cmake .. -DCMAKE_INSTALL_PREFIX=$LOCATION/catalyst2/install
- make
- make install
- echo "export CATALYST=$LOCATION" >> $HOME/.bashrc
- Add the Catalyst2 shared libraries to your library path environment variable in your .bashrc: `echo "export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$LOCATION/catalyst2/install/lib" >> $HOME/.bashrc` Note: in some cases you may have to replace lib with lib64

1.18 2. install Paraview-5.11

- cd $LOCATION
- mkdir paraview-5.11.0 && cd paraview-5.11.0
- wget https://www.paraview.org/files/v5.11/ParaView-v5.11.0.tar.xz
- tar -xvf ParaView-v5.11.0.tar.xz
- mv ParaView-v5.11.0 paraview-src && cd paraview-src
- mkdir -p build && cd build
- export CMAKE_PREFIX_PATH=$CMAKE_PREFIX_PATH:$LOCATION/catalyst2/install/lib/cmake/ca Note: in some cases you may have to replace lib with lib64
- FLAGS=(-DCMAKE_INSTALL_PREFIX=$LOCATION/paraview-5.11.0/install -DVTK_↩ OPENGL_HAS_OSMESA=ON -DPARAVIEW_USE_MPI=ON -DBUILD_TESTING=OFF -DVTK↩ _USE_X=OFF -DPARAVIEW_USE_QT=OFF -DPARAVIEW_USE_PYTHON=ON -DPython3↩ _FIND_STRATEGY=LOCATION -DPython3_ROOT_DIR=$EBROOTPYTHON -DPARAVIEW↩ _BUILD_SHARED_LIBS=ON -DPARAVIEW_ENABLE_RAYTRACING=OFF -DPARAVIEW_↩ ENABLE_CATALYST=ON)
- cmake .. "${FLAGS[@]}"
- make -j8
- make install
- Add the Paraview shared libraries to your library path environment variable in your .bashrc: `echo "export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$LOCATION/paraview-5.11.0/install/lib" >> $HOME/.bashrc` Note: in some cases you may have to replace lib with lib64

1.19 3. Re-compile TOSCA

- Reload the environment with `source $HOME/.bashrc`
- Go inside `TOSCA/src` directory and recompile the solver with `make tosca` ensuring that `-DUSE_CATALYST=1` in the makefile

1.20 4. Running

In order for catalyst2 to find paraview library, the following envronment variables should be set at runtime:

- `export CATALYST_IMPLEMENTATION_PATHS=$LOCATION/paraview-5.11.0/install/lib/catalyst`
  Note: in some cases you may have to replace lib with lib64
- `export CATALYST_IMPLEMENTATION_NAME=paraview`

Credits and Copyright: Sebastiano Stipa - Arjun Ajay - Mohammad Hadi - The University of British Columbia

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

abl_ . . . 15
access_
Access database . . . 21
Acell
Overset acceptor cell . . . 22
acquisition_
Struct containing all acquisition data structures . . . 22
ADM
Actuator Disk Model . . . 23
AFM
Actuator Farm Model . . . 25
ALM
Actuator Line Model . . . 26
anemometer
Wind turbine nacelle mounted anemometer . . . 27
bladeAeroInfo
Blade aerodynamic properties (used to build the AD/AL models) . . . 28
boundingBox
Structure defining the domain bounding box (easy access to xmin, xmax, ymin, ymax, zmin, zmax) . . . 29
Cabot
Structure storing the Shumann wall models information . . . 29
cellIds
Cell indices . . . 30
cellList
Cell Node list . . . 30
cellNode . . . 31
clock_ . . . 31
Cmpnts
Defines the x, y, z components of a vector . . . 32
constants_ . . . 32
Cpt2D
Structure for defining the x, y components of a 2D vector . . . 32
Dcell
Overset donor cell . . . 33

domain_
Domain data structure definition . . . 33
elementBox . . . 34
Face . . . 35
farm_
Wind farm parameters . . . 35
flags_
Solution access flags . . . 36
foilInfo
Airfoil info . . . 37
HalfEdge . . . 37
ibm_
Struct storing IBM model . . . 37
ibmFluidCell . . . 39
ibmMesh . . . 40
ibmNode . . . 40
ibmObject . . . 41
ibmPitchMotion . . . 42
ibmRotation . . . 42
ibmSineMotion . . . 42
inflowData
Struct defining inflow information . . . 43
inletFunctions
Struct storing inlet functions data . . . 43
inletFunctionTypes
Struct storing the inlet function for a patch . . . 44
io_
Struct defining io settings (simulation checkpointing) . . . 46
les_
Struct storing LES model . . . 47
list
Node list . . . 48
LogLawAPG . . . 49
mapInfo
Concurrent precursor mapping info . . . 49
mesh_
Mesh structure storing mesh and curvilinear coordinates . . . 49
nacelleModel
Wind turbine nacelle actuator point model . . . 51
node
Node struct . . . 52
overset_
Structure for the Overset mesh method . . . 52
oversetMotion
Struct with the overset motion . . . 53
patchVectorField
Vector field on the patch (used to store wall models fields) . . . 54
peqn_
Struct storing pressure equation . . . 54
postProcess
Structure defining the variables for postProcessing . . . 56
PowerLawAPG
Structure storing the Shumann wall models information . . . 57
precursor_
Concurrent precursor database . . . 57
scalarBC
Structure dafining the type of boundary conditions for a scalar . . . 58
searchBox . . . 58

Shumann
Structure storing the Shumann wall models information for U and T . . . 59
simInfo_ . . . 60
surface . . . 60
symmTensor
Defines the components of a symmetric tensor . . . 60
teqn_
Struct storing temperature equation . . . 61
towerModel
Wind turbine tower actuator line model . . . 62
UADM
Uniform Actuator Disk model . . . 63
ueqn_
Structure storing momentum equation . . . 64
upSampling
Structure containing a coarser AD mesh located 2.5 D upstrem each turbine for velocity sampling 66
vectorBC
Structure dafining the type of boundary conditions for a vector . . . 67
Vertex . . . 68
wallModel
Wall models container . . . 68
windTurbine
Wind turbine parameters . . . 69

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

src/abl.c
Contains atmospheric boundary layer definition functions . . . 75
src/acquisition.c
Acquisition functions definition . . . 76
src/boundary.c
Contains boundary conditions . . . 78
src/clock.c
Contains simulation time function definitions . . . 79
src/ibm.c
Contains Immersed boundary method function definitions . . . 79
src/ibmInput.c
Contains Immersed boundary method input and read function definitions . . . 82
src/initialField.c
Contains initial field function definitions . . . 116
src/initialization.c
Contains inflow boundary condition function definitions . . . 117
src/io.c
Contains i/o operations function definitions . . . 118
src/les.c
Contains LES model function definitions . . . 121
src/mesh.c
Contains mesh definition functions . . . 122
src/overset.c
Overset function definitions . . . 122
src/peqn.c
Contains P equation function definitions . . . 123
src/precursor.c
Contains top to bottom level routines for the concurrent precursor method . . . 125
src/teqn.c
Contains T equation function definitions . . . 126
src/turbines.c
Contains top to bottom level routines for the wind farm modeling . . . 127
src/ueqn.c
Contains U equation function definitions . . . 130
src/include/abl.h
ABL-related object definition . . . 83

src/include/**access.h** . . . . . **??**
src/include/acquisition.h
Acquisition object declaration 84
src/include/base.h
Base header file including c/c++ libraries and PETSc and HYPRE libs 86
src/include/boundary.h
Boundary conditions header file 87
src/include/**catalystAdaptor.h** . . . . . **??**
src/include/clock.h
Simulation time struct header file 88
src/include/domain.h
Domain data structure definition 89
src/include/**flags.h** . . . . . **??**
src/include/ibm.h
IBM model header file 89
src/include/**ibmInput.h** . . . . . **??**
src/include/inflow.h
Contains inflow boundary condition headers 93
src/include/initialField.h
Initial field header file 94
src/include/initialization.h
Contains simulation initialization function headers 95
src/include/inline.h
Inline functions 96
src/include/io.h
Contains i/o operations function headers 101
src/include/**les.h** . . . . . **??**
src/include/mesh.h
Mesh header file 104
src/include/objects.h
Contains forward object declarations 104
src/include/overset.h
Overset Objects and functions, ... 104
src/include/peqn.h
P equation solution header file 106
src/include/precursor.h
Concurrent precursor header file 107
src/include/teqn.h
T equation solution header file 108
src/include/tosca2PV.h
Post processing header file 109
src/include/**turbines.h** . . . . . **??**
src/include/ueqn.h
U equation solution header file 113
src/include/wallfunctions.h
Wallfunction functions 114
src/include/wallmodel.h
Wall models header file 115

Chapter 4

Class Documentation

4.1 abl_ Struct Reference

Public Attributes

- PetscInt controllerActive

  activate velocity controller
- PetscInt controllerActiveT

  activate temperature controller
- PetscInt coriolisActive

  activate coriolis force
- PetscReal uTau

  friction Velocity
- PetscReal hRough

  equivalent roughness length
- PetscReal uRef

  reference velocity
- PetscReal hRef

  reference height
- PetscReal hGeo

  geostrophic height (required for geostrophic controller)
- PetscReal hInv

  inversion height
- PetscReal dInv

  inversion width
- PetscReal gInv

  delta T across inversion layer
- PetscReal gABL

  temperature gradient below the inversion layer
- PetscReal tRef

  reference potential temperature
- PetscReal gTop

  temperature gradient above the inversion layer
- PetscReal vkConst

  von Karman constant
- PetscReal smear

Rampanelli Zardi model parameter.

- PetscReal fc

  Coriolis parameter (omegaEarth ∗ sin(latitude) = 7.292115e-5 ∗ sin(latitude))
- PetscInt perturbations

  add turbulent perturbations (only if initialization is set to ABLFlow)
- PetscReal ∗ cellLevels

  heights of the averaging planes
- PetscReal ∗ tDes

  initial temperature to be maintained
- word controllerTypeT

  initial or directProfileAssimilation
- word controllerType

  velocity controller type: write/read (writes in postProcessing/momentumSource, reads from momentumSource)
- word **controllerAction**
- PetscReal relax

  source term relaxation factor
- PetscReal alpha

  proportional over integral controller action ratio
- PetscReal timeWindow

  time window of the integral part
- PetscReal ∗ totVolPerLevel

  total volume at each cell level
- PetscInt ∗ totCelPerLevel

  total number of cells per level
- PetscInt ∗ closestLabels

  closest heights w.r.t. controller height
- PetscReal ∗ levelWeights

  weights for variables interpolated at closest heights w.r.t. controller height
- PetscReal controllerMaxHeight

  max height of influence of the velocity controller
- PetscInt geostrophicDampingActive

  geosptrophic oscillation damping
- PetscReal geoDampAvgDT

  average time step from simulation start
- Cmpnts geoDampAvgS

  expected geostrophic velocity
- Cmpnts geoDampUBar

  expected geostrophic velocity
- Cmpnts ∗ geoDampU

  average horizontal velocity at current iteration
- PetscReal geoDampAlpha

  alpha = 1 critical damping (>1 over-damped, <1 under-damped)
- PetscReal geoDampH

  H at which damping begins to be applied.
- PetscReal geoDampDelta

  rising distance for the application function
- PetscReal geoDampC

  critical damping coefficient
- PetscReal geoDampStart

  starting time of geostrophic damping
- PetscReal geoDampWindow

averaging window for the geostrophic velocity

- PetscInt * closestLabelsGeo

  closest heights w.r.t. controller height
- PetscReal * levelWeightsGeo

  weights for variables interpolated at closest heights w.r.t. controller height
- Cmpnts uGeoBar

  desired geostrophic wind speed magnitude
- PetscReal omegaBar

  rotation velocity
- PetscReal hubAngle

  filtered angle at hub height
- PetscReal geoAngle

  cumulated wind angle given by the sum of all rotations and filtered
- Cmpnts **a**
- Cmpnts b

  the two constant parts of the controller (a = geo forcing, b = wind angle controller)
- PetscInt currentCloseIdx

  save the current closest index at each iteration to speed up the interpolation search
- PetscReal sourceAvgStartTime

  if controllerType is 'average', average sources from this time value
- PetscReal ** preCompSources

  table of given sources [ntimesteps][time|sourceX|sourceY|sourceZ] for velocity controller type = timeSeries/time↩ AverageSeries
- PetscReal *** timeHtSources

  table of given timeheight sources [time|sourceX|sourceY|sourceZ] at each cell level, controller type = timeHeight↩ Series
- PetscInt nSourceTimes

  number of times in the pre-computed sources
- Cmpnts cumulatedSource

  cumulated error of the velocity controller (equalt to gradP at steady state)
- Cmpnts * cumulatedSourceHt

  cumulated error of the velocity controller for every mesh level
- PetscReal avgTimeStep

  average time step from the momentum source file
- PetscReal zDampingStart

  starting height of the Rayleigh damping layer
- PetscReal zDampingEnd

  ending height of the Rayleigh damping layer
- PetscReal zDampingAlpha

  damping paramter (it is equal to the Rayleigh viscosity at hEnd)
- Cmpnts * uBarMeanZ

  array storing the reference velocity (averages along i at the k = 1 faces, size = my)
- PetscInt **avgWeight**
- PetscInt zDampingAlsoXY

  damp also x and y velocity components (if 0 only z component is damped)
- PetscInt zDampingXYType

  type 1 (default) averages at inlet, type (2) requires concurrent precursor and xDamping and uses planar averages
- PetscReal kLeftPatchDist

  width of the kLeft Rayleigh damping layer
- PetscReal kLeftDampingAlpha

  kLeft Rayleigh damping coefficient

- Cmpnts kLeftDampingUBar

  kLeft bar velocity with respect to which the flow is damped
- PetscReal kLeftDampingFilterHeight

  above this height damping is unity (transitions to zero at kLeftDampingFilterHeight - kLeftDampingFilterWidth)
- PetscReal kLeftDampingFilterWidth

  width of transition region from no damping to damping
- PetscReal kRightPatchDist

  width of the kRight Rayleigh damping layer
- PetscReal kRightDampingAlpha

  kRight Rayleigh damping coefficient
- Cmpnts kRightDampingUBar

  kRight bar velocity with respect to which the flow is damped
- PetscReal kRightDampingFilterHeight

  above this height damping is unity (transitions to zero at kRightDampingFilterHeight - kRightDampingFilterWidth)
- PetscReal kRightDampingFilterWidth

  width of transition region from no damping to damping
- PetscReal xDampingStart

  starting x of the fringe layer
- PetscReal xDampingEnd

  ending x of the fringe layer
- PetscReal xDampingDelta

  damping raise/decay distance (must be less than 0.5∗(xDampingEnd - xDampingStart))
- PetscReal xDampingAlpha

  damping paramter (see Inoue, Matheou, Teixeira 2014)
- word xDampingControlType

  type of controller: alphaFixed or alphaOptimized
- PetscInt advectionDampingType

  type of advection damping (0: none, 1: LanzilaoMeyers2022)
- PetscReal advDampingStart

  starting x of the adv damping layer
- PetscReal advDampingEnd

  ending x of the adv damping layer
- PetscReal advDampingDeltaStart

  damping raise/decay distance
- PetscReal advDampingDeltaEnd

  damping raise/decay distance
- PetscReal xDampingTimeWindow

  time constant for velocity filtering
- PetscReal xDampingVBar

  desired y-velocity sampled from precursor
- PetscReal **vEnd**
- PetscReal vStart

  line-averaged start and end y-velocity at the fringe region extrema
- PetscReal xDampingError

  error on y-velocity at fringe end w.r.t precursor
- PetscReal xDampingLineSamplingYmin

  starting y of the sampling lines
- PetscReal xDampingLineSamplingYmax

  ending y of the sampling lines
- PetscReal xDampingTimeStart

  last time that alpha was modified

- PetscReal xDampingDeltaV

  y-velocity jump across the fringe
- PetscReal xDampingCoeff

  coeff = |Vend - Vstart| / alpha
- PetscInt * closestLabelsFringe

  closest height w.r.t. fringe controller height
- PetscReal * levelWeightsFringe

  weights for variables interpolated at closest heights w.r.t. fringe controller height
- PetscReal yDampingStart

  starting y of the fringe layer
- PetscReal yDampingEnd

  ending y of the fringe layer
- PetscReal yDampingDelta

  damping raise/decay distance (must be less than 0.5(yDampingEnd - yDampingStart))*
- PetscReal yDampingAlpha

  damping paramter
- PetscInt yDampingNumPeriods

  number of periodizations in the streamwise direction of the x fringe region
- Vec uBarInstY

  instantaneous bar velocity for y-fringe region (only used for concurent precursor)
- Vec tBarInstY

  instantaneous bar temperature for y-fringe region (only used for concurent precursor)
- PetscInt numSourceProc

  global to all procs - number of processors within the source (part of x fringe region within the lateral fringe region)
- PetscInt * sourceProcList

  global to all procs - list of processors in the source
- MPI_Comm * yDamp_comm

  communicator that links each source processor to its corresponding periodization processors(destination) in the lateral fringe region - each source processor has a separate communicator for its set of source-destination processors
- PetscMPIInt * srcCommLocalRank

  global to communicator procs - local rank of the source processor within the source-destination communicator
- cellIds * srcMinInd

  global to communicator procs - minimum k,j,i index of each processor within the source domain
- cellIds * srcMaxInd

  local to each proc - maximum k,j,i index of each processor within the source domain
- PetscInt * isdestProc

  flag indicating if a given processor is within the destination region of a source processor
- cellIds ** destMinInd

  local to each proc - minimum k,j,i index of a processor in the destination domain
- cellIds ** destMaxInd

  local to each proc - maximum k,j,i index of a processor in the destination domain
- PetscInt * srcNumI

  global to communicator procs - number of i index for each processor in source
- PetscInt * srcNumJ

  global to communicator procs - number of j index for each processor in source
- PetscInt * srcNumK

  global to communicator procs - number of k index for each processor in source
- MPI_Request * mapRequest

  MPI variable to perform non blocking broadcast operation.
- Cmpnts ** velMapped

  one d array of the mapped velocity of each source processor

- PetscReal ** **tMapped**
- PetscInt ** closestKCell

  closest 2 k index of the fictitious mesh(mapped from source) to the k indexes of the sucessor domain
- PetscReal ** wtsKCell

  weights of the 2 closest k cells based on their distance
- PetscInt xFringeUBarSelectionType

  read type of fringe region in uBarSelectionType
- Cmpnts ** uBarInstX

  array storing the instantaneous velocity field for x damping layer
- PetscReal ** tBarInstX

  array storing the instantaneous temperature field for x damping layer
- PetscInt ** nProcsKLine

  number of processors in each k-line, used to average after MPI_Allreduce
- PetscReal **xStartCanopy**
- PetscReal xEndCanopy

  x start and ending coordinates of the region where the side force is applied
- PetscReal **yStartCanopy**
- PetscReal yEndCanopy

  x start and ending coordinates of the region where the side force is applied
- PetscReal **zStartCanopy**
- PetscReal zEndCanopy

  z start and ending coordinates of the region where the side force is applied
- PetscReal cftCanopy

  thrust coefficient of the entire wind canopy
- Cmpnts diskDirCanopy

  disk direction of turbines inside the canopy
- PetscReal **zPeak**
- PetscReal **deltaV**
- PetscReal **deltaU**
- PetscReal **Uperiods**
- PetscReal **Vperiods**
- PetscReal * **timeV**
- PetscReal * **hV**
- PetscReal * **timeT**
- PetscReal * **hT**
- Cmpnts ** **uMeso**
- PetscReal ** **tMeso**
- PetscInt **numhV**
- PetscInt **numhT**
- PetscInt **numtV**
- PetscInt **numtT**
- PetscInt ** **velInterpIdx**
- PetscReal ** **velInterpWts**
- PetscInt ** **tempInterpIdx**
- PetscReal ** **tempInterpWts**
- PetscInt **lowestIndV**
- PetscInt **lowestIndT**
- PetscInt **highestIndV**
- PetscInt **highestIndT**
- PetscInt **lMesoIndV**
- PetscInt **hMesoIndV**
- PetscReal **lowestSrcHt**
- PetscReal **highestSrcHt**

- Cmpnts * **luMean**
- Cmpnts * **guMean**
- Cmpnts * **srcPA**
- PetscInt closestTimeIndV

  closest index in time for the mesoscale timevarying data
- PetscReal **closestTimeWtV**
- PetscInt closestTimeIndT

  closest index in time for the mesoscale timevarying data
- PetscReal **closestTimeWtT**
- PetscInt **polyOrder**
- word **wtDist**
- PetscReal ** **polyCoeffM**
- PetscInt **averageSource**
- PetscReal **currAvgtime**
- PetscReal **tAvgWindow**
- Cmpnts * **avgsrc**
- Cmpnts * **avgVel**
- precursor_ * precursor

  concurrent precursor data structure
- access_ * **access**

The documentation for this struct was generated from the following file:

- src/include/abl.h

4.2 access_ Struct Reference

access database

```
#include <access.h>
```

Public Attributes

- clock_ * **clock**
- simInfo_ * **info**
- constants_ * **constants**
- flags_ * **flags**
- mesh_ * **mesh**
- io_ * **io**
- ueqn_ * **ueqn**
- peqn_ * **peqn**
- teqn_ * **teqn**
- les_ * **les**
- farm_ * **farm**
- overset_ * **os**
- abl_ * **abl**
- acquisition_ * **acquisition**
- ibm_ * **ibm**
- PetscInt * **domainID**

4.2.1 Detailed Description

access database

The documentation for this struct was generated from the following file:

- src/include/access.h

4.3 Acell Struct Reference

overset acceptor cell

`#include <overset.h>`

Public Attributes

- PetscInt **indi**
- PetscInt **indj**
- PetscInt **indk**
- PetscReal **coorx**
- PetscReal **coory**
- PetscReal **coorz**
- PetscMPIInt **rank**
- PetscReal **cell_size**
- PetscInt **face**

4.3.1 Detailed Description

overset acceptor cell

The documentation for this struct was generated from the following file:

- src/include/overset.h

4.4 acquisition_ Struct Reference

Struct containing all acquisition data structures.

`#include <acquisition.h>`

Public Attributes

- rakes ∗ probes
  probe rakes
- avgFields ∗ fields
  average and turbulence fields
- keFields ∗ keBudFields
  kinetic energy budjets
- sections ∗ kSections
  information about the k-sections
- sections ∗ jSections
  information about the j-sections
- sections ∗ iSections
  information about the i-sections
- udSections ∗ userSections
  information about the user defined sections
- dataABL ∗ statisticsABL
  ABL statistics.
- data3LM ∗ LM3
  3LM statistics
- perturbFields ∗ perturbABL
  ABL perturbation fields for gravity waves.
- PetscInt **isProbesActive**
- PetscInt **isSectionsActive**
- PetscInt **isAverageABLActive**
- PetscInt **isAverage3LMActive**
- PetscInt **isPerturbABLActive**
- access_ ∗ access
  access database

4.4.1 Detailed Description

Struct containing all acquisition data structures.

The documentation for this struct was generated from the following file:

- src/include/acquisition.h

4.5 ADM Struct Reference

Actuator Disk Model.

`#include <turbines.h>`

Public Attributes

- PetscInt nPoints
  total number of points
- Cmpnts * points
  array containing the AD point coordinates
- PetscReal * dr
  array containing the radial mesh size
- PetscInt nRadial
  number of AD points in the radial direction
- PetscInt nAzimuth
  number of AD points in the azimuthal direction
- PetscReal Uref
  reference velocity to compute CtInf (only used for data writing)
- PetscReal * chord
  array containing the chord at each AD point
- PetscReal * twist
  array containing the twist at each AD point
- PetscReal * solidity
  array containing the solidity at each AD point
- PetscInt ** foilIds
  labels of the 2 airfoils closest to each AD point
- PetscReal ** iw
  interp. weights for the the 2 airfoils closest to each AD point
- PetscInt * thisPtControlled
  flags telling if a point is controlled by this processor
- cellIds * closestCells
  indices of the closest cells to this turbine AD points
- PetscReal * Cd
  drag coefficient at each point of the AD
- PetscReal * Cl
  lift coefficient at each point of the AD
- PetscReal * alpha
  angle of attack at each point of the AD
- Cmpnts * U
  flow velocity at each point of the AD (relative to the blade)
- Cmpnts * B
  body force at each point of the AD mesh
- PetscReal * axialF
  rotor axial force at each point of the AD mesh
- PetscReal * tangtF
  rotor tangential force at each point of the AD mesh
- PetscReal rtrAvgMagU
  average velocity on the rotor (includes induction from CFD)
- PetscReal rtrTorque
  total rotor torque
- PetscReal rtrThrust
  total rotor thrust
- PetscReal aeroPwr
  total rotor aero power
- PetscInt dbg
  prints a lot of information

4.5.1 Detailed Description

Actuator Disk Model.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.6 AFM Struct Reference

Actuator Farm Model.

`#include <turbines.h>`

Public Attributes

- Cmpnts point
  point coordinates
- PetscReal Uref
  reference velocity to compute CtInf (only used for data writing)
- PetscReal Ct
  imposed thrust coefficient
- word projectionType
  gaussexp or anisotropic
- word sampleType
  velocity sampling type ("rotorDisk" or "givenVelocity")
- PetscReal rtrUFilterFreq
  frequency of the single-pole low pass filter for the rotor wind velocity (if sampling type is "rotorDisk")
- cellIds closestCell
  indices of the closest cells to this turbine AL points
- Cmpnts U
  flow velocity at the AF point
- Cmpnts B
  body force at the AF point
- PetscReal axialF
  rotor axial force at the AF point
- PetscReal rtrThrust
  total rotor thrust
- PetscReal aeroPwr
  total rotor aero power
- PetscInt thisPtControlled
  flag telling if this processor controls the AF point
- PetscInt searchDone
  flag telling if the closest cell search has been done
- PetscInt dbg
  prints a lot of information

4.6.1 Detailed Description

Actuator Farm Model.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.7 ALM Struct Reference

Actuator Line Model.

`#include <turbines.h>`

Public Attributes

- word projectionType
  isotropic or anisotropic (follows chord)
- word sampleType
  velocity sampling type ("rotorDisk" or "integral")
- PetscInt nPoints
  total number of points
- Cmpnts * points
  array containing the AL point coordinates
- PetscReal * dr
  array containing the radial mesh size
- PetscInt nRadial
  number of AL points in the radial direction
- PetscInt nAzimuth
  number of AL points in the azimuthal direction
- PetscReal Uref
  reference velocity to compute CtInf (only used for data writing)
- PetscReal * chord
  array containing the chord at each AL point
- PetscReal * twist
  array containing the twist at each AL point
- PetscReal * thick
  array containing the thickness at each AL point
- PetscReal * solidity
  array containing the solidity at each AL point
- PetscInt ** foilIds
  labels of the 2 airfoils closest to each AL point
- PetscReal ** iw
  interp. weights for the the 2 airfoils closest to each AL point
- PetscInt * thisPtControlled
  flags telling if a point is controlled by this processor
- cellIds * closestCells
  indices of the closest cells to this turbine AL points

- PetscReal * Cd
  drag coefficient at each point of the AL
- PetscReal * Cl
  lift coefficient at each point of the AL
- PetscReal * alpha
  angle of attack at each point of the AL
- Cmpnts * U
  flow velocity at each point of the AL (relative to the blade)
- Cmpnts * gWind
  sampled velocity at each point of the AL
- Cmpnts * B
  body force at each point of the AL mesh
- PetscReal * axialF
  rotor axial force at each point of the AL mesh
- PetscReal * tangtF
  rotor tangential force at each point of the AL mesh
- PetscReal rtrAvgMagU
  average velocity on the rotor (includes induction from CFD)
- PetscReal rtrTorque
  total rotor torque
- PetscReal rtrThrust
  total rotor thrust
- PetscReal aeroPwr
  total rotor aero power
- PetscReal azimuth
  azimuthal angle in deg
- PetscInt dbg
  prints a lot of information

4.7.1 Detailed Description

Actuator Line Model.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.8 anemometer Struct Reference

Wind turbine nacelle mounted anemometer.

`#include <turbines.h>`

Public Attributes

- Cmpnts samplePoint
  position of the anemometer
- cellIds closestCell
  cell from which to sample the velocity
- PetscInt anemometerControlled
  flag telling if this processor controls the anemometer
- Cmpnts U
  instantaneous velocity
- Cmpnts anemFilterFreq
  frequency of the single-pole low pass filter for U
- Cmpnts acquisitionFreq
  anemometer acquisition frequency

4.8.1 Detailed Description

Wind turbine nacelle mounted anemometer.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.9 bladeAeroInfo Struct Reference

Blade aerodynamic properties (used to build the AD/AL models)

`#include <turbines.h>`

Public Attributes

- PetscInt size
  number of points
- PetscReal * radius
  radius stations
- PetscReal * chord
  chord distribution
- PetscReal * twist
  twist distribution
- PetscReal * thick
  thickness distribution (only for anisotropic ALM projection)
- PetscInt * foilIds
  airfoil ids as provided in the dict, start from 0

4.9.1 Detailed Description

Blade aerodynamic properties (used to build the AD/AL models)

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.10 boundingBox Struct Reference

Structure defining the domain bounding box (easy access to xmin, xmax, ymin, ymax, zmin, zmax)

`#include <mesh.h>`

Public Attributes

- PetscReal **xmin**
- PetscReal **xmax**
- PetscReal **ymin**
- PetscReal **ymax**
- PetscReal **zmin**
- PetscReal **zmax**
- PetscReal **Lx**
- PetscReal **Ly**
- PetscReal **Lz**

4.10.1 Detailed Description

Structure defining the domain bounding box (easy access to xmin, xmax, ymin, ymax, zmin, zmax)

The documentation for this struct was generated from the following file:

- src/include/mesh.h

4.11 Cabot Struct Reference

structure storing the Shumann wall models information

`#include <wallmodel.h>`

Public Attributes

- PetscReal **roughness**
- PetscReal kappa

  von karman constant (usually 0.4)

4.11.1 Detailed Description

structure storing the Shumann wall models information

The documentation for this struct was generated from the following file:

- src/include/wallmodel.h

4.12 cellIds Struct Reference

Cell indices.

`#include <base.h>`

Public Attributes

- PetscInt **i**
- PetscInt **j**
- PetscInt **k**

4.12.1 Detailed Description

Cell indices.

The documentation for this struct was generated from the following file:

- src/include/base.h

4.13 cellList Struct Reference

cell Node list

`#include <ibm.h>`

Public Attributes

- cellNode * **head**

4.13.1 Detailed Description

cell Node list

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.14 cellNode Struct Reference

Public Attributes

- cellIds **Node**
- struct cellNode * next

  cell id

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.15 clock_ Struct Reference

Public Attributes

- PetscReal time

  current time value
- PetscReal startTime

  simulation start time
- PetscReal endTime

  simulation end time
- word startFrom

  choose if start from startTime value or latestTime (startTime is ignored)
- PetscReal dt

  time step
- PetscReal startDt

  time step
- PetscReal dtOld

  old time step
- PetscReal cfl

  cfl number
- PetscReal dxMin

  min cell side size
- PetscReal acquisitionDt

  uniform dt due to acquistion if applied from the start
- PetscInt it

  current iteration value
- PetscInt itStart

  start iteration value (zero)
- PetscInt timePrecision

  time write precision

The documentation for this struct was generated from the following file:

- src/include/clock.h

4.16 Cmpnts Struct Reference

Defines the x, y, z components of a vector.

`#include <base.h>`

Public Attributes

- PetscScalar **x**
- PetscScalar **y**
- PetscScalar **z**

4.16.1 Detailed Description

Defines the x, y, z components of a vector.

The documentation for this struct was generated from the following file:

- src/include/base.h

4.17 constants_ Struct Reference

Public Attributes

- PetscReal Pr
  Prantl number.
- PetscReal nu
  kinematic viscosity
- PetscReal rho
  flow density [Kg/m3]
- PetscReal tRef
  reference T, required when ABL is not active

The documentation for this struct was generated from the following file:

- src/include/base.h

4.18 Cpt2D Struct Reference

Structure for defining the x, y components of a 2D vector.

`#include <base.h>`

Public Attributes

- PetscReal **x**
- PetscReal **y**

4.18.1 Detailed Description

Structure for defining the x, y components of a 2D vector.

The documentation for this struct was generated from the following file:

- src/include/base.h

4.19 Dcell Struct Reference

overset donor cell

`#include <overset.h>`

Public Attributes

- PetscInt **indi**
- PetscInt **indj**
- PetscInt **indk**
- PetscMPIInt **rank**
- PetscReal **dist2p**

4.19.1 Detailed Description

overset donor cell

The documentation for this struct was generated from the following file:

- src/include/overset.h

4.20 domain_ Struct Reference

Domain data structure definition.

`#include <domain.h>`

Public Attributes

- simInfo_ **info**
- constants_ **constants**
- PetscInt **domainID**
- clock_ * clock
  time info
- mesh_ * mesh
  mesh data structure
- io_ * io
  io settings data structure
- ueqn_ * ueqn
  momentum equation data structure
- peqn_ * peqn
  pressure equation data structure
- teqn_ * teqn
  potential temperature transport equation
- les_ * les
  LES model data structure.
- overset_ * os
  overset data structure
- farm_ * farm
  wind farm data structure
- ibm_ * ibm
  IBM data structure.
- abl_ * abl
  atmospheric boundary layer data structure
- acquisition_ * acquisition
  acquisition data structure
- flags_ flags
  solution flags data structure
- access_ access
  access database data structure

4.20.1 Detailed Description

Domain data structure definition.

The documentation for this struct was generated from the following file:

- src/include/domain.h

4.21 elementBox Struct Reference

Public Attributes

- PetscInt * thisElemControlled
  flag telling if a ibm element is controlled by this processor
- PetscInt * thisElemTransfered
  flag telling that the control of this point is being transferred to another processor
- boundingBox **innerZone**
- boundingBox **outerZone**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.22 Face Struct Reference

Public Attributes

- PetscInt **faceId**
- HalfEdge * **edge**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.23 farm_ Struct Reference

Wind farm parameters.

`#include <turbines.h>`

Public Attributes

- word name
  wind farm name
- PetscInt size
  number of turbines in the farm
- Cmpnts * base
  base coordinates of each turbine
- PetscInt * farmControlActive
  wind farm controller active flag on each wind turbine
- word ** turbineTypes
  array of pointers to type of the wind turbines in the farm
- word ** turbineIds
  array of pointers to ID of the wind turbines in the farm
- word ** turbineModels
  array of pointers to name of the wind turbine models in the farm
- windTurbine ** wt
  array of pointers to wind turbines
- PetscReal timeStart
  start time of acquisition system
- word intervalType
  timeStep: sample at every (timeInterval) iter, adjustableTime sample at every (timeInterval) seconds
- PetscReal timeInterval
  acquisition time interval (overrides simulation time step if smaller and adjustableTime active)
- PetscInt writeNumber
  number of mesh files written up to now
- PetscInt dbg
  global debug switch, checks for point discrimination algorithm (slower)
- Vec lsourceFarmCat
  cartesian wind farm body force
- Vec sourceFarmCont
  contravariant wind farm body force
- access_ * **access**
- PetscInt checkCFL
  at least one actuator line model is present in the wind farm
- PetscReal maxTipSpeed
  maximum tip speed among all rotors

4.23.1 Detailed Description

Wind farm parameters.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.24 flags_ Struct Reference

solution access flags

`#include <flags.h>`

Public Attributes

- PetscInt **isOversetActive**
- PetscInt **isAdjustableTime**
- PetscInt **isLesActive**
- PetscInt **isTeqnActive**
- PetscInt **isAquisitionActive**
- PetscInt **isWindFarmActive**
- PetscInt **isAblActive**
- PetscInt **isIBMActive**
- PetscInt **isZDampingActive**
- PetscInt **isXDampingActive**
- PetscInt **isYDampingActive**
- PetscInt **isCanopyActive**
- PetscInt **isPrecursorSpinUp**
- PetscInt **isConcurrentPrecursorActive**
- PetscInt **isPvCatalystActive**
- PetscInt **isKLeftRayleighDampingActive**
- PetscInt **isKRightRayleighDampingActive**
- PetscInt **isAdvectionDampingActive**
- PetscInt **isSideForceActive**
- PetscInt **isNonInertialFrameActive**
- PetscInt **isGravityWaveModelingActive**

4.24.1 Detailed Description

solution access flags

The documentation for this struct was generated from the following file:

- src/include/flags.h

4.25 foilInfo Struct Reference

Airfoil info.

`#include <turbines.h>`

Public Attributes

- word name
  name of the airfoil
- PetscInt size
  number of points in the look up tables
- PetscReal * aoa
  angles of attack
- PetscReal * cl
  lift coefficients
- PetscReal * cd
  drag coefficients

4.25.1 Detailed Description

Airfoil info.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.26 HalfEdge Struct Reference

Public Attributes

- Vertex * **origin**
- HalfEdge * **next**
- HalfEdge * **twin**
- Face * **face**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.27 ibm_ Struct Reference

struct storing IBM model

`#include <ibm.h>`

Public Attributes

- Vec **INvertFixed**
- PetscInt numBodies

  number of bodies
- word IBInterpolationModel

  interpolation methodology
- word **curvibType**
- word **curvibOrder**
- ibmObject ** ibmBody

  array of pointers to ibm objects
- searchBox * sBox

  array of searchBox with number of search cells and their size for each ibm object
- ibmFluidCell * ibmFCells

  cell ids and supports cells of the ibm fluid cells within each processor
- PetscInt numIBMFluid

  number of ibm fluid cells within each processor
- access_ * **access**
- PetscInt **dbg**
- PetscInt **dynamic**
- PetscInt **computeForce**
- PetscInt **checkNormal**
- PetscInt **averageNormal**
- PetscInt **wallShearOn**
- PetscInt **writeSTL**
- PetscInt **ibmABL**
- PetscReal **interpDist**
- PetscReal timeStart

  start time of acquisition system
- word intervalType

  timeStep: sample at every (timeInterval) iter, adjustableTime sample at every (timeInterval) seconds
- PetscReal timeInterval

  acquisition time interval (overrides simulation time step if smaller and adjustableTime active)

4.27.1 Detailed Description

struct storing IBM model

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.28 ibmFluidCell Struct Reference

Public Attributes

- cellIds cellId

  cell id of the ibm fluid cell
- cellList flNodes

  list of the cell ids of the fluid cells as support to the ibm fluid cell
- list slNodes

  list of the ibm mesh nodes as support to the ibm fluid cell
- list bID

  list of the body id corresponding to the ibm mesh node in the slNodes list
- PetscInt numFl

  number of fluid cells in the support
- PetscInt numSl

  number of ibm mesh nodes in the support
- PetscReal rad

  support radius of the ibm fluid cell
- PetscInt closestElem

  closest ibm mesh element to the ibm fluid cell
- Cmpnts pMin

  projected point on the ibm mesh element from the ibm fluid cell
- PetscReal minDist

  dist to the ibm mesh element from the ibm fluid cell
- PetscInt **bodyID**
- Cmpnts **normal**
- PetscReal **cs1**
- PetscReal **cs2**
- PetscReal cs3

  ibm interpolation coefficient of the projected point from the ibm element nodes
- PetscReal **cr1**
- PetscReal **cr2**
- PetscReal cr3

  ibm interpolation coefficient of the background mesh point from the background plane triangle nodes
- PetscInt **i1**
- PetscInt **j1**
- PetscInt k1

  first node (cell center id of fluid mesh) of the background interpolation plane
- PetscInt **i2**
- PetscInt **j2**
- PetscInt k2

  second node (cell center id of fluid mesh) of the background interpolation plane
- PetscInt **i3**
- PetscInt **j3**
- PetscInt k3

  third node (cell center id of fluid mesh) of the background interpolation plane
- PetscInt imode

  indicates which triangle (out 0f 8) the interpolated point belongs in the background plane
- PetscReal dB

  distance to the background plane from ibm fluid node along the IBM element normal

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.29 ibmMesh Struct Reference

Public Attributes

- PetscInt nodes

  number of nodes in the IBM Body
- PetscInt elems

  number of elements in the IBM body
- Vertex * **vertices**
- HalfEdge * **halfEdges**
- Face * **faces**
- Cmpnts * nCoor

  pointer to the co-ordinates of the nodes
- PetscInt * **nID1**
- PetscInt * **nID2**
- PetscInt * nID3

  pointer to the 3 node ids of a triangular mesh element
- Cmpnts * eN

  pointers to the component of face normal in x, y and z direction of element
- Cmpnts * eT1

  pointers to the component of face tangential1 (eT1 = eN x k, where k is unit normal along z, which is taken as a generic direction)
- Cmpnts * eT2

  pointers to the component of face tangential2 (eT2 = eN x eT1)
- ibmNode * **ibmMeshNode**
- PetscInt * eSurface

  pointer to the surfaceBody that this element belongs to
- PetscReal * eA

  area of the element
- Cmpnts * eCent

  coordinate of the element center
- Cmpnts * nU

  velocity of the nodes
- Cmpnts * nUPrev

  velocity of the node at the previous timestep
- Cmpnts * eQVec

  center of the smallest bounding sphere of an ibm element
- PetscReal * eRVec

  radius of the smallest bounding sphere of an ibm element

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.30 ibmNode Struct Reference

Public Attributes

- PetscInt **elem** [MAX_ELEMENTS_PER_NODE]
- PetscInt **numConnected**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.31 ibmObject Struct Reference

Public Attributes

- word **bodyName**
- word **bodyType**
- PetscInt **numSurfaces**
- PetscInt **bodyID**
- PetscInt **thinBody**
- word **elementSet**
- ibmMesh ∗ **ibMsh**
- surface ∗∗ **ibmSurface**
- word **fileType**
- wallModel ∗ **ibmWallModelU**
- wallModel ∗ **ibmWallModelT**
- word **velocityBC**
- word **uBCSetType**
- PetscInt **wallFunctionTypeU**
- word **tempBC**
- word **tBCSetType**
- PetscInt **wallFunctionTypeT**
- PetscReal **fixedTemp**
- Cmpnts **baseLocation**
- word **bodyMotion**
- ibmRotation ∗ **ibmRot**
- ibmSineMotion ∗ **ibmSine**
- ibmPitchMotion ∗ **ibmPitch**
- PetscReal **searchCellRatio**
- boundingBox ∗ **bound**
- list ∗ **searchCellList**
- Cmpnts ∗ ibmPForce

  pressure force acting on each ibm element
- Cmpnts procBoundCenter

  center of the processor bounding box for the ibm body
- Cmpnts procBoundSize

  size of the processor bounding box in the x, y and z direction
- PetscInt ∗ thisPtControlled

  flag telling if a ibm mesh point is controlled by this processor based on normal projection
- PetscInt ∗ thisPtControlTransfer

  flag telling that the control of this point is being transferred to another processor
- cellIds ∗ closestCells

  closest cell id to the outward normal projection from the ibm mesh element
- PetscInt ibmControlled

  flag which tells if this proc controls this IBM body
- MPI_Comm IBM_COMM

  communicator for this IBM
- PetscInt ∗ **elementMapping**
- elementBox ∗ **eBox**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.32 ibmPitchMotion Struct Reference

Public Attributes

- PetscReal **amplitude**
- PetscReal **frequency**
- PetscReal **initAngPosition**
- Cmpnts **pitchAxis**
- Cmpnts **pitchCenter**
- PetscReal **tPrev**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.33 ibmRotation Struct Reference

Public Attributes

- PetscReal **angSpeed**
- PetscReal **angAcc**
- PetscReal **rotAngle**
- PetscReal **maxR**
- Cmpnts **rotAxis**
- Cmpnts **rotCenter**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.34 ibmSineMotion Struct Reference

Public Attributes

- PetscReal **amplitude**
- PetscReal **frequency**
- Cmpnts **motionDir**
- PetscReal **tPrev**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.35 inflowData Struct Reference

Struct defining inflow information.

`#include <boundary.h>`

Public Attributes

- PetscReal * inflowTimes
  array with the available times (stored at the beginning)
- PetscInt nInflowTimes
  number of times in the inflow database
- PetscInt currentCloseIdx
  save the current closest index at each iteration to speed up the interpolation search

4.35.1 Detailed Description

Struct defining inflow information.

The documentation for this struct was generated from the following file:

- src/include/boundary.h

4.36 inletFunctions Struct Reference

Struct storing inlet functions data.

`#include <boundary.h>`

Public Attributes

- inletFunctionTypes * **iLeft**
- inletFunctionTypes * **iRight**
- inletFunctionTypes * **jLeft**
- inletFunctionTypes * **jRight**
- inletFunctionTypes * **kLeft**
- inletFunctionTypes * **kRight**

4.36.1 Detailed Description

Struct storing inlet functions data.

The documentation for this struct was generated from the following file:

- src/include/boundary.h

4.37 inletFunctionTypes Struct Reference

Struct storing the inlet function for a patch.

`#include <boundary.h>`

Public Attributes

- PetscInt **typeU**
- PetscInt **typeT**
- PetscInt **typeNut**
- Cmpnts **Uref**
- PetscReal **Href**
- PetscReal **uPrimeRMS**
- Cmpnts Udir

  velocity direction vector
- PetscReal roughness

  equivalent roughness size
- PetscReal hInv

  height from bottom patch of inversion layer
- PetscReal uTau

  friction velocity used to compute log profile
- PetscReal dInv

  inversion width
- PetscReal gInv

  delta T across inversion layer
- PetscReal tRef

  reference potential temperature
- PetscReal gTop

  temperature gradient above the inversion layer
- PetscReal gABL

  temperature gradient below the inversion layer
- PetscReal smear

  Rampanelli Zardi model parameter.
- PetscReal latitude

  for Nieuwstadt model
- PetscReal fc

  Coriolis parameter.
- PetscInt n1

  number of inflow cells along the 1st index (ordered as accessed: k,j,i. Eg. for k-normal patch this is n cells along j)
- PetscInt n2

  number of inflow cells along the 2nd index (ordered as accessed: k,j,i. Eg. for k-normal patch this is n cells along i)
- PetscInt n1wg

  n1 with ghosts (n1 + 2)
- PetscInt n2wg

  n2 with ghosts (n2 + 2)
- PetscInt prds1

  indicates how many times inflow data should be periodized along 1st index
- PetscInt prds2

  indicates how many times inflow data should be periodized along 2nd index

- PetscInt merge1

  average data at 5 top cells
- PetscInt shift2

  apply shift on inflow slice data along direction 2
- PetscInt mapT

  flag telling if also T is mapped (if temperatureTransport is active is mandatory)
- PetscInt mapNut

  flag telling if also nut is mapped (optional)
- PetscReal *** inflowWeights

  array of 4 weights for each cell center (ordered as a 2D plane array)
- cellIds *** closestCells

  array of 4 closest cells for each cell center
- PetscReal *** inflowWeights_1

  array of 6 weights for each cell center (ordered as a 2D plane array) - j spline interp
- cellIds *** closestCells_1

  array of 6 closest cells for each cell center - j spline interp
- PetscReal *** inflowWeights_2

  array of 6 weights for each cell center (ordered as a 2D plane array) - i spline interp
- cellIds *** closestCells_2

  array of 6 closest cells for each cell center - i spline interp
- word sourceType

  source mesh type (uniform or grading)
- word interpMethod

  interpolation method (linear or nullDiv)
- PetscReal width1

  inflow cell width in the 1st direction
- PetscReal width2

  inflow cell width in the 2nd direction
- PetscReal inflowHeigth

  inflow height for gradient extrapolation aloft
- Cmpnts ** ucat_plane

  cartesian velocity inflow data
- PetscReal ** t_plane

  temperature inflow data
- PetscReal ** nut_plane

  nut inflow data
- inflowData inflowU

  velocity inflow database inflormation for unsteadyMappedInflow BC
- inflowData inflowT

  temperature inflow database inflormation for unsteadyMappedInflow BC
- inflowData inflowNut

  temperature inflow database inflormation for unsteadyMappedInflow BC
- Cmpnts * uBarAvgTopX

  velocity average from inflow database at top 10 points
- PetscReal * tBarAvgTopX

  temperature average from inflow database at top 10 points
- PetscReal * avgTopPointCoords

  z coordinates of the 10 average top points
- PetscReal avgTopDelta

  length of the five top cells
- PetscReal avgTopLength

height of the inflow slices

- PetscReal shiftSpeed

  speed of the i-direction shift
- PetscReal * ycent

  vector storing y cell center coordinates (assuming they don't vary vertically)
- PetscInt * yIDs

  IDs for the right interpolation point (the left would be IDs[i]-1)
- PetscReal * yWeights

  vector storing the right weight for the interpolation (left weight is 1.0 - right weight)
- MPI_Comm IFFCN_COMM

  communicator involving all processors touching k-boundaries
- PetscReal amplitude

  oscillation amplitude w.r.t. reference velocity magnitude
- PetscReal periods

  number of periods in the spanwise direction

4.37.1 Detailed Description

Struct storing the inlet function for a patch.

The documentation for this struct was generated from the following file:

- src/include/boundary.h

4.38 io_ Struct Reference

Struct defining io settings (simulation checkpointing)

`#include <io.h>`

Public Attributes

- word intervalType

  timeStep or adjustableTime
- PetscReal timeInterval

  in iterations if intervalType = timeStep, in seconds if intervalType = adjustableTime
- PetscInt runTimeWrite

  flag telling if must write at this iteration
- PetscInt purgeWrite

  deletes all other files after writing the current
- PetscInt averaging

  compute the time-averaged solution fields
- PetscReal avgPrd

  sampling period in seconds
- PetscReal avgStartTime

  start time of averaging procedure
- PetscInt phaseAveraging

compute the phase-averaged solution fields

- PetscReal phAvgPrd

  sampling period in seconds
- PetscReal phAvgStartTime

  start time of phase averaging procedure
- PetscInt keBudgets

  compute kinetic energy budget terms
- PetscInt writePForce

  compute pressure on individual elements
- PetscInt avgWeight

  number of average snapshots (cumulated at runtime)
- PetscInt pAvgWeight

  number of phase average snapshots (cumulated at runtime)
- PetscInt keAvgWeight

  number of keBudget average snapshots (cumulated at runtime)
- PetscInt **qCrit**
- PetscInt **l2Crit**
- PetscInt **windFarmForce**
- PetscInt **sources**
- PetscInt **buoyancy**
- PetscInt **continuity**
- word **lastModification**
- PetscReal **timeIntervalCatalyst**
- PetscReal **startTimeCatalyst**
- word **outputTypeCatalyst**
- word ioTypeCatalyst

  type of catalyst output (generic/script)
- word scriptNameCatalyst

  name of input script
- access_ * **access**

4.38.1 Detailed Description

Struct defining io settings (simulation checkpointing)

The documentation for this struct was generated from the following file:

- src/include/io.h

4.39 les_ Struct Reference

struct storing LES model

`#include <les.h>`

Public Attributes

- PetscReal **maxCs**
- Vec **lSx**
- Vec **lSy**
- Vec **lSz**
- Vec **lS**
- Vec **lLM**
- Vec **lMM**
- Vec **lQN**
- Vec **lNN**
- Vec **lLM_old**
- Vec **lMM_old**
- Vec **lQN_old**
- Vec **lNN_old**
- Vec **lCs**
- Vec **lN**
- Vec **lCh**
- Vec L

  length scale
- Vec lNu_t

  eddy viscosity
- word **initFieldType**
- access_ * access

  access database

4.39.1 Detailed Description

struct storing LES model

The documentation for this struct was generated from the following file:

- src/include/les.h

4.40 list Struct Reference

Node list.

`#include <ibm.h>`

Public Attributes

- node * **head**

4.40.1 Detailed Description

Node list.

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.41 LogLawAPG Struct Reference

Public Attributes

- PetscReal **roughness**
- PetscReal kappa

  von karman constant (usually 0.4)

The documentation for this struct was generated from the following file:

- src/include/wallmodel.h

4.42 mapInfo Struct Reference

concurrent precursor mapping info

`#include <precursor.h>`

Public Attributes

- PetscInt **kFringeStart**
- PetscInt kFringeEnd

  start and end ids of the fringe region in the successr indexing
- PetscInt **kStart**
- PetscInt kEnd

  start and end ids of this processor in the k direction in the successor indexing

4.42.1 Detailed Description

concurrent precursor mapping info

The documentation for this struct was generated from the following file:

- src/include/precursor.h

4.43 mesh_ Struct Reference

mesh structure storing mesh and curvilinear coordinates

`#include <mesh.h>`

Public Attributes

- word meshName

  name of the domain
- word meshFileType

  type of mesh file
- boundingBox bounds

  domain extensions and lengths information
- PetscReal grndLevel

  the ground level height - would be = bounds->zmin for normal simulation, but could change if using IBM
- PetscInt **IM**
- PetscInt **JM**
- PetscInt KM

  ncells in the GCC directions
- DM da

  data structure for scalars (include the grid geometry informaion, to obtain the mesh information, use DMDAGet↩ Coordinates)
- DM fda

  data Structure for vectors
- DM sda

  data Structure for symmetric tensors
- DMDALocalInfo info

  data struct that contains information about a the Distributed Array (essentially mesh information)
- Vec **Cent**
- Vec lCent

  cell centers coordinates in the cartesian frame
- Vec **lCsi**
- Vec **lEta**
- Vec **lZet**
- Vec lAj

  jacobian evaluated at cell centers (1 / cell volume)
- Vec **lICsi**
- Vec **lIEta**
- Vec **lIZet**
- Vec lIAj

  jacobian evaluated at the i-faces
- Vec **lJCsi**
- Vec **lJEta**
- Vec **lJZet**
- Vec lJAj

  jacobian evaluated at the j-faces
- Vec **lKCsi**
- Vec **lKEta**
- Vec **lKZet**
- Vec lKAj

  jacobian evaluated at the j-faces
- Vec **Nvert**
- Vec Nvert_o

  solid body field for IBM
- Vec **lNvert**
- Vec **lNvert_o**
- PetscInt **i_periodic**

- PetscInt **ii_periodic**
- PetscInt **j_periodic**
- PetscInt **jj_periodic**
- PetscInt **k_periodic**
- PetscInt **kk_periodic**
- scalarBC **boundaryNut**
- scalarBC **boundaryT**
- vectorBC **boundaryU**
- inletFunctions **inletF**
- Vec **fluxLimiter**
- MPI_Comm **MESH_COMM**
- access_ * **access**

4.43.1 Detailed Description

mesh structure storing mesh and curvilinear coordinates

The documentation for this struct was generated from the following file:

- src/include/mesh.h

4.44 nacelleModel Struct Reference

Wind turbine nacelle actuator point model.

```
#include <turbines.h>
```

Public Attributes

- PetscReal Cd
  tower drag coefficient
- PetscReal eps
  *spreading width of the gaussian projection function (good is 0.035 * hTower)*
- PetscReal prjNSigma
  confidence interval as number of std deviations for the projection function (hardcoded to 2.7)
- Cmpnts point
  nacelle point
- PetscReal A
  frontal area
- cellIds * controlledCells
  labels of the background mesh cells influenced by this nacelle in this processor
- PetscInt nControlled
  size of controlledCells
- PetscInt thisPtControlled
  flags telling if this nacelle is controlled by this processor
- cellIds closestCell
  indices of the closest cell to this turbine nacelle point
- Cmpnts U

flow velocity at nacelle point

- Cmpnts B

  body at nacelle point

- PetscReal tangF

  nacelle tangential force

- PetscReal nacThrust

  total nacelle thrust

- MPI_Comm NAC_COMM

  communicator for this nacelle

- PetscMPIInt nProcsNac

  size of the NAC_COMM communicator

4.44.1 Detailed Description

Wind turbine nacelle actuator point model.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.45 node Struct Reference

Node struct.

`#include <ibm.h>`

Public Attributes

- PetscInt **Node**
- struct node * next

  node id

4.45.1 Detailed Description

Node struct.

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.46 overset_ Struct Reference

Structure for the Overset mesh method.

`#include <overset.h>`

Public Attributes

- labelList parentMeshId

  Parent node of each level.
- labelList childMeshId

  Child node of each level.
- word interpolationType

  type of Interpolation
- PetscInt dynamicOverset

  switch for dynamic overset
- PetscInt procChange

  switch to check if the background processors which intersect with overset mesh have changed
- oversetMotion * **oMotion**
- PetscReal **cellAvg**
- PetscReal cellFactor

  cellFactor scales the cell size by a factor which will be used as the search radius
- std::vector< MPI_Comm > oset_comm

  communicator for overset - background processor interaction
- std::vector< Acell > aCell

  interpolated cells of the overset mesh
- std::vector< std::vector< Dcell > > dCell

  MLS interpolation: donor cells of the background mesh.
- std::vector< Dcell > closestDonor

  Trilinear interpolation: donor cells of the background mesh.
- std::vector< std::vector< PetscInt > > AcellProcMat

  rank matrix which indicates the processor connectivity between the Acell1 and Dcell0
- std::vector< PetscInt > NumAcellPerProc

  number of Acceptor cells in each processor
- std::vector< std::vector< PetscReal > > DWeights

  MLS interpolation weights.
- access_ * **access**

4.46.1 Detailed Description

Structure for the Overset mesh method.

The documentation for this struct was generated from the following file:

- src/include/overset.h

4.47 oversetMotion Struct Reference

struct with the overset motion

```
#include <overset.h>
```

Public Attributes

- Cmpnts **prescribedVel**
- word **motionType**
- PetscInt setMotion

  if set to true it uses the prescribed motion to move the overset mesh
- PetscInt ibmAttached

  set to true if the setMotion is false so motion based on the movement of the IBM attached to mesh

4.47.1 Detailed Description

struct with the overset motion

The documentation for this struct was generated from the following file:

- src/include/overset.h

4.48 patchVectorField Struct Reference

Vector field on the patch (used to store wall models fields)

`#include <base.h>`

Public Attributes

- PetscReal ** **x**
- PetscReal ** **y**
- PetscReal ** **z**

4.48.1 Detailed Description

Vector field on the patch (used to store wall models fields)

The documentation for this struct was generated from the following file:

- src/include/base.h

4.49 peqn_ Struct Reference

struct storing pressure equation

`#include <peqn.h>`

Public Attributes

- Vec phi

  phi is the actual solution of Poisson equation (then converted in Phi)
- Vec **Phi**
- Vec lPhi

  pressure correction for frac. step method
- Vec **P**
- Vec lP

  pressure at current and previous time step
- Vec **lLid**
- Vec lGid

  matrixFree - matrixBased connectivity (used to build the poisson coeff. matrix)
- HYPRE_Int thisRankSize

  number of cells owned by this processore (used to build the poisson coeff. matrix)
- HYPRE_Int thisRankStart

  first cell ID owned by this processor in global indexing (used to build the poisson coeff. matrix)
- HYPRE_Int thisRankEnd

  last cell ID owned by this processor in global indexing (thisRankStart + thisRankSize - 1)
- HYPRE_Int totalSize

  total number of cells (excluding physical ghost & IBM points, depending of the Hypre Poisson type, -1, -2, 1)
- PetscReal **initialPoissonRes**
- PetscReal finalPoissonRes

  initial and final residual of the Poisson iteration
- HYPRE_Int poissonIterations

  number of Poisson iterations
- PetscInt hypreSolverType

  1: GMRES, 2: PCG
- PetscInt poissonIt

  max number of poisson iterations per timestep
- PetscReal poissonTol

  relative exit tolerance
- PetscInt amgAgg

  aggresive coarsening is good for > 50mil grids
- PetscInt **amgCoarsenType**
- PetscReal amgThresh

  threshold value - 0.5 : Cartesian, 0.6 : Distorted
- word solverType

  HYPRE or PETSc.
- HYPRE_Solver hypreSlvr

  solver
- HYPRE_Solver hyprePC

  preconditioner
- HYPRE_IJMatrix hypreA

  coefficient matrix
- HYPRE_ParCSRMatrix hypreParA

  coefficient matrix
- HYPRE_IJVector **hypreP**
- HYPRE_IJVector hypreRhs

  unknwon and RHS
- HYPRE_ParVector **hypreParP**

- HYPRE_ParVector hypreParRhs
  unknown and RHS
- Mat petscA
  coefficient matrix
- KSP ksp
  linear krylov-subspace context
- PC petscPC
  preconditioner
- Vec petscRhs
  right hand side
- MatNullSpace petscNs
  null space
- access_ * access
  access database

4.49.1 Detailed Description

struct storing pressure equation

The documentation for this struct was generated from the following file:

- src/include/peqn.h

4.50 postProcess Struct Reference

Structure defining the variables for postProcessing.

`#include <tosca2PV.h>`

Public Attributes

- PetscInt **postProcessFields**
- PetscInt **writeRaster**
- PetscInt **samplingSections**
- PetscInt **postProcessPrecursor**
- precursor_ * **precursor**

4.50.1 Detailed Description

Structure defining the variables for postProcessing.

The documentation for this struct was generated from the following file:

- src/include/tosca2PV.h

4.51 PowerLawAPG Struct Reference

structure storing the Shumann wall models information

`#include <wallmodel.h>`

Public Attributes

- PetscReal **roughness**
- PetscReal kappa

  von karman constant (usually 0.4)

4.51.1 Detailed Description

structure storing the Shumann wall models information

The documentation for this struct was generated from the following file:

- src/include/wallmodel.h

4.52 precursor_ Struct Reference

concurrent precursor database

`#include <precursor.h>`

Public Attributes

- PetscInt thisProcessorInFringe

  1: this processors will solve, 0: this processor will idle
- mapInfo map

  precursor/successor mapping info
- domain_ * **domain**

4.52.1 Detailed Description

concurrent precursor database

The documentation for this struct was generated from the following file:

- src/include/precursor.h

4.53 scalarBC Struct Reference

Structure dafining the type of boundary conditions for a scalar.

`#include <boundary.h>`

Public Attributes

- word **iLeft**
- word iRight

  type of boundary condition
- word **jLeft**
- word **jRight**
- word **kLeft**
- word **kRight**
- PetscReal **iLval**
- PetscReal iRval

  value (defined if BC prefix is 'fixed' only)
- PetscReal **jLval**
- PetscReal **jRval**
- PetscReal **kLval**
- PetscReal **kRval**
- PetscInt **iLWF**
- PetscInt iRWF

  wall function type (defined for velocity BC only)
- PetscInt **jLWF**
- PetscInt **jRWF**
- PetscInt **kLWF**
- PetscInt **kRWF**

4.53.1 Detailed Description

Structure dafining the type of boundary conditions for a scalar.

The documentation for this struct was generated from the following file:

- src/include/boundary.h

4.54 searchBox Struct Reference

Public Attributes

- PetscInt **ncx**
- PetscInt **ncy**
- PetscInt ncz

  number of search cells in x,y and z direction
- PetscReal **dcx**
- PetscReal **dcy**
- PetscReal dcz

  search cell size in x,y and z direction

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.55 Shumann Struct Reference

structure storing the Shumann wall models information for U and T

`#include <wallmodel.h>`

Public Attributes

- word wfEvalType
  type of uStar evaluation localized/averaged
- PetscReal kappa
  von karman constant (usually 0.4)
- PetscReal thetaRef
  reference potential temperature
- PetscReal roughness
  equivalent roughness height
- PetscReal gammaM
  momentum gammaM from Paulson 1970 (stable BL)
- PetscReal betaM
  momentum betaM from Pauslon 1970 (unstable BL)
- PetscReal gammaH
  pot. temp. gammaH from Paulson 1970 (stable BL)
- PetscReal betaH
  pot. temp. betaH from Paulson 1970 (unstable BL)
- PetscReal alphaH
  pot. temp. alphaH from Paulson 1970 (stable BL)
- PetscReal tLast
  time at which the last theta update was done
- PetscReal heatingRate
  surface heating rate
- PetscReal ** surfaceTheta
  surface temperature
- PetscInt surfaceThetaSet
  surface temperature has been initialized
- PetscReal qWall
  prescribed wall heat flux
- PetscReal * **surfTemp**
- PetscReal * **surfL**
- PetscReal * **timeVec**
- PetscInt **numT**

4.55.1 Detailed Description

structure storing the Shumann wall models information for U and T

The documentation for this struct was generated from the following file:

- src/include/wallmodel.h

4.56 simInfo_ Struct Reference

Public Attributes

- PetscInt **nDomains**
- PetscInt **periodic**

The documentation for this struct was generated from the following file:

- src/include/base.h

4.57 surface Struct Reference

Public Attributes

- word **surfaceName**
- word **surfaceFileType**
- PetscInt **surfaceId**
- Cmpnts **baseLocation**
- word **elementSet**
- PetscInt * **elementMapping**
- ibmMesh * **ibMsh**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.58 symmTensor Struct Reference

Defines the components of a symmetric tensor.

```
#include <base.h>
```

Public Attributes

- PetscScalar **xx**
- PetscScalar **xy**
- PetscScalar **xz**
- PetscScalar **yy**
- PetscScalar **yz**
- PetscScalar **zz**

4.58.1 Detailed Description

Defines the components of a symmetric tensor.

The documentation for this struct was generated from the following file:

- src/include/base.h

4.59 teqn_ Struct Reference

struct storing temperature equation

`#include <teqn.h>`

Public Attributes

- SNES snesT
  non linear matrix free context
- Mat JT
  non linear matrix free preconditioner
- Mat **AT**
- Mat **CT**
- KSP ksp
  linear krylov-subspace context
- PC **pc**
- Vec **Rhs**
- Vec **Rhs_o**
- Vec TmprtTmp
  temporary solution
- Vec **Tmprt**
- Vec **ITmprt**
- Vec **Tmprt_o**
- Vec **ITmprt_o**
- Vec **IDivT**
- Vec **IViscT**
- Vec IViscIBMT
  viscous and divergence temperature equation fluxes
- Vec sourceT
  temperature sources
- Vec IRhoK
  rhok / rho0 field
- Vec ghGradRhok
  buoyancy term for momentum equation
- PetscReal absExitTol
  absolute exit tolerance
- PetscReal relExitTol
  relative exit tolerance
- word ddtScheme
  time derivative scheme

- wallModel * iLWM
  wall model on the i-left patch
- wallModel * iRWM
  wall model on the i-right patch
- wallModel * jLWM
  wall model on the j-left patch
- wallModel * jRWM
  wall model on the j-right patch
- PetscInt pTildeFormulation
  buoyancy term is expressed as gradient in momentum
- word **initFieldType**
- access_ * access
  access database

4.59.1 Detailed Description

struct storing temperature equation

The documentation for this struct was generated from the following file:

- src/include/teqn.h

4.60 towerModel Struct Reference

Wind turbine tower actuator line model.

`#include <turbines.h>`

Public Attributes

- PetscReal rBase
  radius at the base of the tower
- PetscReal rTop
  radius at the top of the tower
- PetscReal Cd
  tower drag coefficient
- PetscReal eps
  *spreading width of the gaussian projection function (good is 0.035 * hTower)*
- PetscReal prjNSigma
  confidence interval as number of std deviations for the projection function (hardcoded to 2.7)
- PetscInt nPoints
  number of tower points in the linear direction
- Cmpnts * points
  total number of points
- PetscReal * dA
  array containing the frontal area at each tower point
- cellIds * controlledCells

labels of the background mesh cells influenced by this tower in this processor

- PetscInt nControlled

  size of controlledCells
- PetscInt * thisPtControlled

  flags telling if a point is controlled by this processor
- cellIds * closestCells

  indices of the closest cells to this turbine tower points
- Cmpnts * U

  flow velocity at each point of the tower
- Cmpnts * B

  body force at each point of the tower mesh
- PetscReal * tangF

  tower tangential force at each point of the tower mesh
- PetscReal twrThrust

  total tower thrust
- MPI_Comm TWR_COMM

  communicator for this tower
- PetscMPIInt nProcsTwr

  size of the TWR_COMM communicator

4.60.1 Detailed Description

Wind turbine tower actuator line model.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.61 UADM Struct Reference

Uniform Actuator Disk model.

`#include <turbines.h>`

Public Attributes

- PetscInt nPoints

  total number of points
- Cmpnts * points

  array containing the AD point coordinates
- PetscReal * dA

  array containing the element area at each AD point (sums up to rotor area)
- PetscInt nRadial

  number of AD points in the radial direction
- PetscInt nAzimuth

  number of AD points in the azimuthal direction
- word sampleType

velocity sampling type ("rotorUpstream" or "givenVelocity")

- PetscReal Uref

  reference velocity to make the provided Ct dimensional
- PetscReal Ct

  imposed thrust coefficient
- PetscReal axiInd

  axial induction factor
- PetscInt * thisPtControlled

  flags telling if a point is controlled by this processor
- cellIds * closestCells

  indices of the closest cells to this turbine AD points
- Cmpnts * U

  flow velocity at each point of the AD
- Cmpnts * B

  body force at each point of the AD mesh
- PetscReal * axialF

  rotor axial force at each point of the AD mesh
- PetscReal rtrAvgMagU

  average velocity on the rotor (includes induction from CFD)
- PetscReal rtrThrust

  total rotor thrust
- PetscReal aeroPwr

  total rotor aero power
- PetscInt dbg

  prints a lot of information

4.61.1 Detailed Description

Uniform Actuator Disk model.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.62 ueqn_ Struct Reference

structure storing momentum equation

`#include <ueqn.h>`

Public Attributes

- SNES snesU

  non linear matrix free context for momentum equation
- Mat JU

  non linear matrix free preconditioner
- Mat **A**
- Mat **C**
- KSP ksp

  linear krylov-subspace context
- PC **pc**
- Vec Utmp

  temporary solution passed to the SNES evaluation function or used for RK4
- Vec Rhs

  rhs of the momentum equation (stores transport and viscous fluxes), low level use in FormU
- Vec Rhs_o

  rhs of the momentum equation at previous time step
- Vec lFp

  rhs of the momentum equation prior to dotting with curv. coords basis (becomes Rhs)
- Vec **lDiv1**
- Vec **lDiv2**
- Vec lDiv3

  Components of the convective term in momentum equation.
- Vec **lVisc1**
- Vec **lVisc2**
- Vec lVisc3

  Components of the viscous term in the momentum equation.
- Vec **lViscIBM1**
- Vec **lViscIBM2**
- Vec lViscIBM3

  Components of the viscous term in the momentum equation for IBM faces.
- Vec dP

  pressure term of the momentum equation
- Vec dPAGW

  pressure term of the momentum equation as calculated from provided atmopsheric gravity waves pressure
- Vec bTheta

  buoyancy field
- Vec sourceU

  source term to drive Uref at Zref with periodic BCs
- Vec gCont

  gravity vector in cuvilinear coordinates
- Vec **Ucont**
- Vec lUcont

  contravariant fluxes (contravariant velocity / J)
- Vec **Ucat**
- Vec lUcat

  cartesian velocity
- Vec Ucont_o

  contravariant fluxes at the previous time step
- Vec **lUstar**
- word ddtScheme

  time derivative scheme

- word divScheme
  divergence scheme
- PetscReal relExitTol
  relative exit tolerance
- PetscReal absExitTol
  absolute exit tolerance
- PetscInt inviscid
  inviscid run
- PetscInt buoyancy
  buoyancy term
- PetscInt coriolis
  coriolis term
- PetscInt fringe
  fringe region term
- PetscInt centralDiv
  linear divergence scheme
- PetscInt centralUpwindDiv
  blending between linear and upwind scheme
- PetscInt centralUpwindWDiv
  blending between linear and upwind scheme for non-uniform mesh
- PetscInt quickDiv
  3rd order QUICK scheme
- PetscInt weno3Div
  3rd order WENO scheme
- wallModel * iLWM
  wall model on the i-left patch
- wallModel * iRWM
  wall model on the i-right patch
- wallModel * jLWM
  wall model on the j-left patch
- wallModel * jRWM
  wall model on the j-right patch
- word **initFieldType**
- access_ * access
  access database

4.62.1 Detailed Description

structure storing momentum equation

The documentation for this struct was generated from the following file:

- src/include/ueqn.h

4.63 upSampling Struct Reference

Structure containing a coarser AD mesh located 2.5 D upstrem each turbine for velocity sampling.

`#include <turbines.h>`

Public Attributes

- PetscInt nPoints

  total number of points
- Cmpnts ∗ points

  array containing the upstream point coordinates
- PetscReal ∗ dA

  array containing the element area at each upstream sample point (sums up to rotor area)
- Cmpnts center

  center point of the sampling mesh
- PetscInt thisRigControlled

  true if this processor has controller cells, zero otherwise
- MPI_Comm UPW_COMM

  communicator for this sampling (TRB_COMM is a subset of UPW_COMM: all data contained in UPW_COMM are accessible from TRB_COMM)
- cellIds ∗ controlledCells

  labels of the background mesh cells influenced by this sample points in this processor
- PetscInt nControlled

  size of controlledCells
- PetscInt ∗ thisPtControlled

  flags telling if a point is controlled by this processor
- cellIds ∗ closestCells

  indices of the closest cells to this sample points
- PetscReal Uref

  wind velocity averaged on the sample points

4.63.1 Detailed Description

Structure containing a coarser AD mesh located 2.5 D upstrem each turbine for velocity sampling.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

4.64 vectorBC Struct Reference

Structure dafining the type of boundary conditions for a vector.

`#include <boundary.h>`

Public Attributes

- word **iLeft**
- word iRight

  type of boundary condition
- word **jLeft**
- word **jRight**
- word **kLeft**
- word **kRight**
- Cmpnts **iLval**
- Cmpnts iRval

  value (defined if BC prefix is 'fixed' only)
- Cmpnts **jLval**
- Cmpnts **jRval**
- Cmpnts **kLval**
- Cmpnts **kRval**
- PetscInt **iLWF**
- PetscInt iRWF

  wall function type (defined for velocity BC only)
- PetscInt **jLWF**
- PetscInt **jRWF**
- PetscInt **kLWF**
- PetscInt **kRWF**

4.64.1 Detailed Description

Structure dafining the type of boundary conditions for a vector.

The documentation for this struct was generated from the following file:

- src/include/boundary.h

4.65 Vertex Struct Reference

Public Attributes

- Cmpnts **nCoor**
- PetscInt **vertexId**
- HalfEdge ∗ **edge**

The documentation for this struct was generated from the following file:

- src/include/ibm.h

4.66 wallModel Struct Reference

wall models container

`#include <wallmodel.h>`

Public Attributes

- Shumann * **wmShumann**
- Cabot * **wmCabot**
- PowerLawAPG * **wmPowerLawAPG**
- LogLawAPG * **wmLogLawAPG**
- patchVectorField **tauWall**
- patchVectorField **qWall**
- patchVectorField **uFilt**
- PetscReal ** uMeanMag

  average velocity to compute filtering time scale
- PetscInt uFiltSet

  filtered velocity has been initialized

4.66.1 Detailed Description

wall models container

The documentation for this struct was generated from the following file:

- src/include/wallmodel.h

4.67 windTurbine Struct Reference

Wind turbine parameters.

`#include <turbines.h>`

Public Attributes

- word id

  id of the wind turbine
- word type

  type of the wind turbine
- PetscInt nBlades

  number of turbine blades
- PetscReal rTip

  tip radius from CoR
- PetscReal rHub

  hub radius from CoR
- PetscReal hTwr

  tower height
- PetscReal ovrHang

  nacelle overhang in the rotor direction (facing the wind)
- PetscReal precone

  blade precone (equal for all blades)
- Cmpnts twrDir

  unit vector pointing from base to tower top

- PetscReal upTilt
  nacell up-tilt (positive when blades get far from tower)
- PetscReal genEff
  electrical generator eficiency
- word rotDir
  rotation dir as seen lookingthrough the WT from the front
- Cmpnts rotCenter
  rotor rotation center
- PetscInt nFoils
  number of airfoils in the database
- word ** foilNames
  array of pointers of size(n-foils in turbine) to airfoil names
- foilInfo ** foils
  array of pointers of size(n-foils in turbine) to each airfoil's info
- bladeAeroInfo blade
  blade properties
- anemometer WDAS
  wind data acquisition system (nacelle mounted anemometer)
- Cmpnts rtrDir
  (all) unit vector pointing to the rotor orientation in non-tilted position (facing the wind)
- Cmpnts rtrAxis
  (all) unit vector pointing to the rotor orientation in tilted position (facing the wind)
- Cmpnts omega_hat
  (AD/AL) turbine angular velocity unit vector (directed as rtrAxis, pointed according to rotDir)
- PetscReal rtrOmega
  (AD/AL) turbine angular velocity in rad/sec
- ADM adm
  actuator disk model
- UADM uadm
  unform actuator disk model
- ALM alm
  actuator line model
- AFM afm
  actuator farm model
- towerModel twr
  actuator line tower model
- PetscInt includeTwr
  flag telling if tower is included
- nacelleModel nac
  actuator point for nacelle
- PetscInt includeNacelle
  flag telling if nacelle is included
- upSampling * upPoints
  struct containing the upstream sampling points information
- PetscReal deg2rad
  degrees to radiants conversion factor
- PetscReal rad2deg
  radiants to degrees conversion factor
- PetscReal rpm2RadSec
  RPM to rad/s conversion factor.
- PetscInt turbineControlled

flag which tells if this proc controls this wind turbine

- MPI_Comm TRB_COMM

  communicator for this turbine
- PetscMPIInt writerRank

  label of master rank of the TRB_COMM communicator in the MPI_COMM_WORLD rank list
- PetscMPIInt nProcsTrb

  size of the TRB_COMM communicator
- word genControllerType

  name of torque controller (if none preserves intial omega, else reads from control/genControllerType)
- PetscReal genOmega

  turbine angular velocity
- PetscReal rtrOmegaFilt

  turbine filtered angular velocity
- PetscReal rtrSpdFilterFreq

  frequency of the single-pole low pass filter for the rotor angular frequency
- PetscReal cutInGenSpd

  cut in generator speed
- PetscReal cutInGenTq

  cut in generator torque
- PetscReal regTwoStartGenSpd

  generator speed at the start of control region 2
- PetscReal regTwoEndGenSpd

  generator speed at the end of control region 2
- PetscReal ratedGenTq

  generator torque at the rated wind speed
- PetscReal omegaKP

  proportional gain of the generator torque controller
- PetscReal genTorque

  total generator torque
- PetscReal genPwr

  total rotor gen power
- PetscInt tqRateLimiter

  activate torque rate limiter (1 yes, 0 no)
- PetscInt rtrSpdLimiter

  activate generator speed limiter (1 yes, 0 no)
- PetscReal tqMaxRate

  maximum torque variation rate allowed for the generator torque controller
- PetscReal ratedRotorSpd

  rotor speed at rated wind speed
- PetscReal driveTrainInertia

  sum of all the inertias attached to the shaft
- PetscReal genInertia

  generator inertia
- PetscReal hubInertia

  hub intertia
- PetscReal bldInertia

  blade intertia
- PetscReal gbxRatioG2R

  gearbox generator-to-rotor ratio
- PetscReal gbxEff

  gearbox mechanical efficiency

- word pitchControllerType

  name of torque controller (if none preserves intial omega, else reads from control/pitchControllerType)
- PetscReal ∗ pitch

  blade pitch (array of size n-blades), could vary blade by blade
- PetscReal collPitch

  collective pitch (same for all blades)
- PetscReal pitchKP

  pitch controller proportional gain
- PetscReal pitchKI

  pitch controller integral gain
- PetscReal pitchKD

  pitch controller derivative gain
- PetscReal pitchS2R

  pitch at which the sensit. of power to pitch variations has doubled w.r.t. rated position
- PetscReal errPID

  error of the PID controller
- PetscReal intErrPID

  integrated error of the PID controller
- PetscInt pitchRateLimiter

  activate pitch rate limiter (1 yes, 0 no)
- PetscInt pitchAngleLimiter

  activate pitch angle limiter (1 yes, 0 no)
- PetscReal pitchMaxRate

  max rate of blade pitch motion
- PetscReal pitchMin

  minimum pitch angle
- PetscReal pitchMax

  maximum pitch angle
- word yawControllerType

  name of torque controller (if none preserves intial omega, else reads from control/yawControllerType)
- word yawSamplingType

  mode of velocity sampling (hubUpDist, anemometer)
- PetscReal yawAverageWindow

  time window used for misalignment averaging
- PetscReal yawAllowedError

  allows +-yawAllowedError flow misalignment in degrees
- PetscReal yawMin

  minimum yaw angle
- PetscReal yawMax

  maximum yaw angle
- PetscReal yawAngle

  actual yaw angle wrt xyz background ref frame
- PetscReal flowAngle

  actual averaged flow angle wrt xyz background ref frame
- PetscReal yawError

  yaw misalignment error
- PetscReal yawSpeed

  yaw speed in degs/s
- cellIds yawSampleIds

  ids of the point where the velocity for misalignment computation must be sampled
- PetscInt yawChanged

flag telling if must do the search on the turbine points due to yaw change

- PetscReal wfControlCollPitch

  delta pitch proscribed by wind farm controller (ADM and ALM)
- PetscReal wfControlCt

  delta Ct prescribed by wind farm controller (uniformADM, AFM)
- PetscInt wfControlNData

  number of entries in the control table
- PetscReal * wfControlTimes

  time from wind farm controller table
- PetscReal * wfControlValues

  values of wind farm controller variables (changes based on turbine model)
- PetscInt currentCloseIdx

  save the current closest index at each iteration to speed up the interpolation search
- cellIds * controlledCells

  labels of the background mesh cells influenced by this turbine in this processor
- PetscInt nControlled

  size of controlledCells
- PetscReal eps

  *spreading width of the gaussian projection function (good is 0.035 * dBlade)*
- PetscReal eps_x

  x spreading width of the gaussian projection function (for AFM and AALM)
- PetscReal eps_y

  y spreading width of the gaussian projection function (for AFM and AALM)
- PetscReal eps_z

  z spreading width of the gaussian projection function (for AFM and AALM)
- PetscReal flat

  flatness parameter for gaussexp AFM projection
- PetscReal r12

  half decay radius for gaussexp AFM projection
- PetscReal l

  normalization factor for gaussexp AFM projection
- PetscReal prjNSigma

  confidence interval as number of std deviations for the projection function (hardcoded to 2.7)
- PetscInt dbg

  this turbines info at the begining of the simulation and at each iteration

4.67.1 Detailed Description

Wind turbine parameters.

The documentation for this struct was generated from the following file:

- src/include/turbines.h

Chapter 5

File Documentation

5.1 src/abl.c File Reference

Contains atmospheric boundary layer definition functions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/inflow.h"
```

Functions

- PetscErrorCode InitializeABL (abl_ *abl)

  Read from ABLProperties.dat and initialize the ABL parameters.
- PetscReal NieuwstadtGeostrophicWind (abl_ *abl)

  Evaluate geostrophic speed using Nieuwstadt model.
- PetscErrorCode **readMesoScaleVelocityData** (abl_ *abl)
- PetscErrorCode readMesoScaleTemperatureData (abl_ *abl)

  read the mesoscale driving velocity and potential temperature profile
- PetscErrorCode **findTimeHeightSeriesInterpolationWts** (abl_ *abl)
- PetscErrorCode **findVelocityInterpolationWeights** (abl_ *abl)
- PetscErrorCode **findTemperatureInterpolationWeights** (abl_ *abl)
- PetscErrorCode **initializeYDampingMapping** (abl_ *abl)
- PetscErrorCode **setWeightsYDamping** (abl_ *abl)
- PetscErrorCode **computeLSqPolynomialCoefficientMatrix** (abl_ *abl)

5.1.1 Detailed Description

Contains atmospheric boundary layer definition functions.

5.2 src/acquisition.c File Reference

Acquisition functions definition.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
```

Functions

- PetscErrorCode InitializeAcquisition (domain_ *domain)

  Reads acquisition settings and calls individual iniitialization functions.
- PetscErrorCode InitializeAcquisitionPrecursor (domain_ *domain)

  Reads acquisition settings and calls individual iniitialization functions for concurrent precursor.
- PetscErrorCode WriteAcquisition (domain_ *domain)

  Write acquisition data.
- PetscErrorCode averageFieldsInitialize (acquisition_ *acquisition)

  Initialize average fields acquisition.
- PetscErrorCode averageKEBudgetsInitialize (acquisition_ *acquisition)

  Initializes ke budgets fields acquisition.
- PetscErrorCode setKeBoundsAndComms (mesh_ *mesh, keFields *ke)

  Search box bounds and define communicators.
- PetscErrorCode readKeBoxArray (keFields *ke)

  Read box array.
- PetscErrorCode averageFields (acquisition_ *acquisition)

  Average fields.
- PetscErrorCode averageKEBudgets (acquisition_ *acquisition)

  Compute MKE budgets.
- PetscErrorCode boxCumulateKEBudgets (acquisition_ *acquisition)

  Do box cumulation for mesh. energy budgets.
- PetscErrorCode averageKEBudgetsCat (acquisition_ *acquisition)

  Compute MKE budgets in cartesian form.
- PetscErrorCode averageKEBudgetsCont (acquisition_ *acquisition)

  Compute MKE budgets in generalized curvilinaer form w/o mesh stretching.
- PetscErrorCode sectionsInitialize (acquisition_ *acquisition)

  Initializes sections.
- PetscErrorCode writeSections (acquisition_ *acquisition)

  Writes down section data.
- PetscErrorCode iSectionSaveVector (mesh_ *mesh, sections *sec, PetscInt iplane, Vec &V, const char *fieldName)

  Saves i-section vector data.
- PetscErrorCode jSectionSaveVector (mesh_ *mesh, sections *sec, PetscInt jplane, Vec &V, const char *fieldName)

  Saves j-section vector data.
- PetscErrorCode kSectionSaveVector (mesh_ *mesh, sections *sec, PetscInt kplane, Vec &V, const char *fieldName)

  Saves k-section vector data.
- PetscErrorCode iSectionSaveScalar (mesh_ *mesh, sections *sec, PetscInt iplane, Vec &V, const char *fieldName)

Saves i-section scalar data.

- PetscErrorCode jSectionSaveScalar (mesh_ *mesh, sections *sec, PetscInt jplane, Vec &V, const char *fieldName)

  Saves j-section scalar data.
- PetscErrorCode kSectionSaveScalar (mesh_ *mesh, sections *sec, PetscInt kplane, Vec &V, const char *fieldName)

  Saves k-section scalar data.
- PetscErrorCode ProbesInitialize (domain_ *domain, PetscInt postProcessing)

  Initializes the array of probe rakes.
- PetscErrorCode InitRakeFile (probeRake *rake, const char *fieldName)

  Initialize probe rake file.
- PetscErrorCode writeProbes (domain_ *domain)

  Writes probes to file.
- PetscErrorCode computeQCritIO (acquisition_ *acquisition)

  Compute Q criteria for I/O in cartesian form.
- PetscErrorCode computeCoriolisIO (acquisition_ *acquisition)

  Compute coriolis force for I/O in cartesian form.
- PetscErrorCode computeDrivingSourceIO (acquisition_ *acquisition)

  Compute driving pressure force for I/O in cartesian form.
- PetscErrorCode computeXDampingIO (acquisition_ *acquisition)

  Compute x damping force for I/O in cartesian form.
- PetscErrorCode computeCanopyForceIO (acquisition_ *acquisition)

  Compute side force for I/O in cartesian form.
- PetscErrorCode computeVelocityDivergence (acquisition_ *acquisition)

  Compute velocity divergence field for I/O.
- PetscErrorCode perturbationABLInitialize (acquisition_ *acquisition)

  Initializes ABL perturbation fields w.r.t. given reference.
- PetscErrorCode averagePerturbationABL (acquisition_ *acquisition)

  Performs perturbation averaging and writes to memory.
- PetscErrorCode averaging3LMInitialize (domain_ *domain)

  Initialize 3LM data structure.
- PetscErrorCode read3LMFields (acquisition_ *acquisition)

  Reads velocity and pressure averages (they don't use auxiliary fields like TBL, LBL, IBL)
- PetscErrorCode writeAveraging3LM (domain_ *domain)

  Performs 3LM average and write 3LM fields.
- PetscErrorCode write3LMPoints (acquisition_ *acquisition)

  Write 3LM points to file inside postProcessing/3LM/points.
- PetscErrorCode write3LMFields (acquisition_ *acquisition)

  Write 3LM fields to file inside postProcessing/3LM/.
- PetscErrorCode findAvgLineIds (acquisition_ *acquisition)

  Inizializes the closest cells and the owner of the 3LM mesh points.
- PetscErrorCode averagingABLInitialize (domain_ *domain)

  Initialize ABL data structure.
- PetscErrorCode writeAveragingABL (domain_ *domain)

  Write spatial averaged ABL statistics.

5.2.1 Detailed Description

Acquisition functions definition.

5.2.2 Function Documentation

5.2.2.1 read3LMFields()

```
PetscErrorCode read3LMFields (
            acquisition_ * acquisition )
```

Reads velocity and pressure averages (they don't use auxiliary fields like TBL, LBL, IBL)

read field

5.3 src/boundary.c File Reference

Contains boundary conditions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/inflow.h"
#include "include/wallfunctions.h"
```

Functions

- PetscErrorCode readScalarBC (const word &location, const word &field, scalarBC *bc)
  Reads boundary conditions for a scalar field.
- PetscErrorCode readVectorBC (const word &location, const word &field, vectorBC *bc)
  Reads bundary conditions for a vector field.
- PetscErrorCode SetBoundaryConditions (mesh_ *mesh)
  Set boundary conditions.
- PetscErrorCode checkBoundaryConditions (mesh_ *mesh)
  Checks available boundary conditions.
- PetscErrorCode SetPeriodicConnectivity (mesh_ *mesh, word &meshFileName)
  Set periodicconnectivity type.
- PetscErrorCode UpdateContravariantBCs (ueqn_ *ueqn)
  Update contravariant fluxes boundary conditions.
- PetscErrorCode UpdateCartesianBCs (ueqn_ *ueqn)
  Update cartesian boundary conditions.
- PetscErrorCode UpdateTemperatureBCs (teqn_ *teqn)
  Update temperature boundary conditions.
- PetscErrorCode UpdateNutBCs (les_ *les)
  Update effective viscosity boundary conditions.
- PetscErrorCode UpdatePressureBCs (peqn_ *peqn)
  Update pressure boundary conditions.
- PetscErrorCode UpdatePhiBCs (peqn_ *peqn)
  Update pressure correction boundary conditions.

- PetscErrorCode UpdateWallModelsU (ueqn_ *ueqn)
  Update wall model for specified wall shear stress.
- PetscErrorCode UpdateWallModelsT (teqn_ *teqn)
  Update wall model for specified wall heat flux.
- PetscErrorCode UpdateImmersedBCs (ibm_ *ibm)
  Update ibm boundary conditions.
- PetscErrorCode SetWallModels (ueqn_ *ueqn)
  Set wall models.
- PetscErrorCode readSurfaceTempData (Shumann *wm)
  read surface temperature and obhukhov length data

5.3.1 Detailed Description

Contains boundary conditions.

5.4 src/clock.c File Reference

Contains simulation time function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
```

Functions

- PetscErrorCode **adjustTimeStep** (domain_ *domain)
- PetscErrorCode **timeStepInfo** (domain_ *domain, clock_ *clock, PetscReal &dxByU_min, PetscReal &maxU, cellIds &maxUCell)

5.4.1 Detailed Description

Contains simulation time function definitions.

5.5 src/ibm.c File Reference

Contains Immersed boundary method function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/wallfunctions.h"
#include "include/ibmInput.h"
```

Functions

- PetscErrorCode InitializeIBM (ibm_ *ibm)

  initialize ibm: top level function
- PetscErrorCode UpdateIBM (ibm_ *ibm)

  update imb: top level function
- PetscErrorCode setIBMWallModels (ibm_ *ibm)

  set IBM wall model type and properties
- PetscErrorCode **ComputeForceMoment** (ibm_ *ibm)
- PetscErrorCode writeIBMForceData (ibm_ *ibm, PetscInt b, PetscReal *gElemPressure, PetscReal net↩ Moment, PetscReal ibmPower, Cmpnts PresForce, Cmpnts ViscForce, Cmpnts momentVector)

  writes the IBM force data for element and net force
- PetscErrorCode UpdateIBMesh (ibm_ *ibm)

  update the ibm mesh when the ibm body is moving
- PetscErrorCode **recomputeIBMeshProperties** (ibm_ *ibm, PetscInt b)
- PetscErrorCode pitchingMotion (ibm_ *ibm, PetscInt b)

  prescribe pitching oscillation motion for IBM body
- PetscErrorCode sineMotion (ibm_ *ibm, PetscInt b)

  prescribe sinusoidal motion for IBM body
- PetscErrorCode rotateIBMesh (ibm_ *ibm, PetscInt b)

  rotate the ibm mesh based on the angular speed input
- PetscErrorCode writeIBMData (ibm_ *ibm, PetscInt b)

  writes the angular position when the body is rotating
- PetscErrorCode findInterceptionPoint (ibm_ *ibm)

  find the interceptionPoint on the background mesh plane for curvib interpolation
- PetscErrorCode interceptionPt (Cmpnts pCoor, Cmpnts pc[9], Cmpnts eNorm, ibmFluidCell *ibF)

  interception point algorithm based on dividing the plane into 8 triangles
- PetscErrorCode **findClosestIBMElement2Solid** (ibm_ *ibm)
- PetscErrorCode findClosestIBMElement (ibm_ *ibm)

  find the closest ibm mesh element to a IBM fluid node
- PetscErrorCode **CurvibInterpolationTriangular** (ibm_ *ibm)
- PetscErrorCode **CurvibInterpolationInternalCell** (ibm_ *ibm)
- PetscErrorCode CurvibInterpolation (ibm_ *ibm)

  CURVIB normal projection interpolation algorithm.
- PetscErrorCode **CurvibInterpolationQuadratic** (ibm_ *ibm)
- PetscErrorCode findIBMWallShearChester (ibm_ *ibm)

  compute shear stress at faces close to the IBM
- PetscErrorCode findIBMWallShear (ibm_ *ibm)

  compute shear stress at faces close to the IBM
- PetscErrorCode ibmSearch (ibm_ *ibm)

  ibm fluid node search performed using ray tracing algorithm
- PetscErrorCode MLSInterpolation (ibm_ *ibm)

  MLS interpolation algorithm.
- PetscErrorCode **checkIBMexists** (ibm_ *ibm)
- PetscErrorCode **createHalfEdgeDataStructure** (ibm_ *ibm)
- PetscErrorCode **computeIBMElementNormal** (ibm_ *ibm)
- PetscErrorCode findIBMControlledProcs (ibm_ *ibm)

  find in which processors the ibm body belongs
- PetscErrorCode initElementProjectionProcs (ibm_ *ibm)

  create list of ibm element processors and their closest ibm fluid cell to the ibm element normal projection
- PetscErrorCode IBMProjectionProcessorTransfer (ibm_ *ibm)

  transfer ibm mesh element across processors based on their movement - for dynamic case

- PetscErrorCode createProcessorBufferZones (ibm_ *ibm)

  create inner and outer buffer zones for ibm mesh element parallelization
- PetscErrorCode initElementProcs (ibm_ *ibm)

  create list of ibm element processors and their closest ibm fluid cell to the ibm element
- PetscErrorCode IBMElementProcessorTransfer (ibm_ *ibm)

  transfer ibm mesh element across processors based on their movement - for dynamic case
- PetscErrorCode elementBoundingSphere (ibmObject *ibmBody)

  find element bounding sphere
- PetscErrorCode findBodyBoundingBox (ibm_ *ibm)

  find the bounding box around an ibm body
- PetscErrorCode findFluidSupportNodes (ibm_ *ibm)

  find the fluid nodes that act as support nodes to the given ibm fluid node
- PetscErrorCode findIBMFluidCells (ibm_ *ibm)

  find list of ibm fluid nodes within each processor
- PetscErrorCode findIBMMeshSupportNodes (ibm_ *ibm)

  find the ibm mesh nodes that act as support node to the given ibm fluid node
- PetscErrorCode findSearchCellDim (ibm_ *ibm)

  find the search cell (coarser mesh around the ibm) dimension
- PetscErrorCode createSearchCellList (ibm_ *ibm)

  insert ibm mesh elements as a list into the search cell that they belong to
- PetscErrorCode destroyLists (ibm_ *ibm)

  destroy the ibm search cell list
- void insertIBMSupportNodes (Cmpnts pt, cellIds sCell, ibmFluidCell *ibF, ibmObject *ibBody, searchBox *sBox)

  insert ibm nodes into the list of support nodes around an ibm fluid node
- PetscErrorCode rayCastLocal (Cmpnts p, Cmpnts p1, Cmpnts p2, Cmpnts p3, Cmpnts p4, ibmMesh *ibMsh, cellIds sCell, searchBox *sBox, boundingBox *ibBox, list *searchCellList, PetscInt &intersect)

  ray casting algorithm in local positive cell neighbourhood
- PetscReal rayCastingTest (Cmpnts p, ibmMesh *ibMsh, cellIds sCell, searchBox *sBox, boundingBox *ibBox, list *searchCellList)

  ray casting algorithm
- bool isSearchCellSupport (PetscReal rad, Cmpnts pt, cellIds cID, boundingBox *ibBox, searchBox *sBox)

  inline function to check if a search cell is a support to an ibm fluid node
- Cmpnts randomdirection (Cmpnts p, cellIds sCell, boundingBox *ibBox, searchBox *sBox, PetscInt seed)

  inline function to generate a random direction along the z index of the search cell for the ray casting algorithm
- PetscInt intsectElement (Cmpnts p, Cmpnts dir, Cmpnts node1, Cmpnts node2, Cmpnts node3, PetscReal *t, PetscReal *u, PetscReal *v)

  inline function to check if the ray intersects an ibm element
- PetscBool isLineTriangleInt (Cmpnts p1, Cmpnts p2, ibmMesh *ibMsh, PetscInt e)

  inline function to check if the line intersects an ibm element
- PetscInt isPointInTriangle (Cmpnts p, Cmpnts p1, Cmpnts p2, Cmpnts p3, Cmpnts norm)

  check whether point p is inside the triangle - 3D
- PetscInt ISInsideTriangle2D (Cpt2D p, Cpt2D pa, Cpt2D pb, Cpt2D pc)

  check whether point p is inside the triangle - 2D
- PetscInt ISSameSide2D (Cpt2D p, Cpt2D p1, Cpt2D p2, Cpt2D p3)

  Check whether 2D point p is located on the same side of line p1p2.
- void disP2Line (Cmpnts p, Cmpnts p1, Cmpnts p2, Cmpnts *po, PetscReal *d, PetscReal *t)

  find projection of point on a line and the distance to it
- Cmpnts computeVertexAverageNormal (ibmMesh *ibMsh, PetscInt vertexId)

  find the angle averaged normal about a vertex
- Cmpnts computeEdgeAverageNormal (ibmMesh *ibMsh, PetscInt vertexId, PetscInt faceId)

find the angle averaged normal about an edge

- void triangleIntp (Cpt2D p, Cpt2D p1, Cpt2D p2, Cpt2D p3, ibmFluidCell *ibF)

  find the interpolation weights of a point inside a triangle from its nodes
- void triangleIntpBg (Cpt2D p, Cpt2D p1, Cpt2D p2, Cpt2D p3, ibmFluidCell *ibF)

  find the interpolation weights of a point inside a triangle from its nodes used for background grid

5.5.1 Detailed Description

Contains Immersed boundary method function definitions.

5.6 src/ibmInput.c File Reference

Contains Immersed boundary method input and read function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/ibmInput.h"
```

Functions

- PetscErrorCode readIBMProperties (ibm_ *ibm)

  read the IBMProperties.dat file , IBM mesh files and allocate memory for the objects
- PetscErrorCode **createHalfEdgeDataStructure** (ibm_ *ibm, PetscInt b)
- PetscErrorCode **readIBMObjectMesh** (ibm_ *ibm, PetscInt b)
- ibmNode * **initializeIBMNodes** (PetscInt numNodes)
- PetscErrorCode nodeElementConnectivity (ibm_ *ibm)

  find the reverse connectivity which lists the elements connected to a node.
- PetscErrorCode **combineMesh** (ibmObject *ibmBody)
- PetscErrorCode readIBMBodyFileAbaqusInp (ibmObject *ibmBody)

  read the ibm mesh in abaqus inp format
- PetscErrorCode readIBMSurfaceFileAbaqusInp (surface *ibmSurface)

  read the ibm mesh in abaqus inp format
- PetscErrorCode readIBMBodyFileASCII (ibmObject *ibmBody)

  read the ibm mesh in ASCII format
- PetscErrorCode readIBMBodyFileGRD (ibmObject *ibmBody)

  read the ibm mesh in GRD format
- PetscErrorCode readIBMSurfaceFileGRD (surface *ibmSurface)

  read the ibm mesh in .grd format
- PetscErrorCode writeSTLFile (ibm_ *ibm, PetscInt b)

  write the STL mesh
- PetscErrorCode readIBMBodyFileUCD (ibmObject *ibmBody)

  read the ibm mesh in ucd format
- PetscErrorCode readIBMBodyFileUCD2 (ibmObject *ibmBody)

  read the ibm mesh in ucd2 format (without 0 tri entries)
- PetscErrorCode readIBMSurfaceFileUCD (surface *ibmSurface)

  read the ibm mesh in ucd format
- PetscErrorCode readIBMSurfaceFileUCD2 (surface *ibmSurface)

  read the ibm mesh in ucd2 format (without 0 tri entries)

5.6.1 Detailed Description

Contains Immersed boundary method input and read function definitions.

5.6.2 Function Documentation

5.6.2.1 readIBMSurfaceFileAbaqusInp()

```
PetscErrorCode readIBMSurfaceFileAbaqusInp (
            surface * ibmSurface )
```

read the ibm mesh in abaqus inp format

combine the mesh nodes and elements of different surface bodies into one ibm body

5.7 src/include/abl.h File Reference

ABL-related object definition.

Classes

- struct abl_

Functions

- PetscErrorCode InitializeABL (abl_ *abl)
  Read from ABLProperties.dat and initialize the ABL parameters.
- PetscReal NieuwstadtGeostrophicWind (abl_ *abl)
  Evaluate geostrophic speed using Nieuwstadt model.
- PetscErrorCode readMesoScaleTemperatureData (abl_ *abl)
  read the mesoscale driving velocity and potential temperature profile
- PetscErrorCode **readMesoScaleVelocityData** (abl_ *abl)
- PetscErrorCode **findVelocityInterpolationWeights** (abl_ *abl)
- PetscErrorCode **findTemperatureInterpolationWeights** (abl_ *abl)
- PetscErrorCode **initializeYDampingMapping** (abl_ *abl)
- PetscErrorCode **setWeightsYDamping** (abl_ *abl)
- PetscErrorCode **computeLSqPolynomialCoefficientMatrix** (abl_ *abl)
- PetscErrorCode **findTimeHeightSeriesInterpolationWts** (abl_ *abl)

5.7.1 Detailed Description

ABL-related object definition.

5.8 src/include/acquisition.h File Reference

Acquisition object declaration.

```
#include "acquisition/averageAcquisition.h"
#include "acquisition/probesAcquisition.h"
#include "acquisition/sectionsAcquisition.h"
#include "acquisition/ablAcquisition.h"
#include "acquisition/keAcquisition.h"
#include "acquisition/perturbAcquisition.h"
```

Classes

- struct acquisition_

  Struct containing all acquisition data structures.

Functions

- PetscErrorCode InitializeAcquisition (domain_ *domain)

  Reads acquisition settings and calls individual iniitialization functions.
- PetscErrorCode InitializeAcquisitionPrecursor (domain_ *domain)

  Reads acquisition settings and calls individual iniitialization functions for concurrent precursor.
- PetscErrorCode WriteAcquisition (domain_ *domain)

  Write acquisition data.
- PetscErrorCode ProbesInitialize (domain_ *domain, PetscInt postProcessing=0)

  Initializes the array of probe rakes.
- PetscErrorCode InitRakeFile (probeRake *rake, const char *fieldName)

  Initialize probe rake file.
- PetscErrorCode writeProbes (domain_ *domain)

  Writes probes to file.
- PetscErrorCode sectionsInitialize (acquisition_ *acquisition)

  Initializes sections.
- PetscErrorCode writeSections (acquisition_ *acquisition)

  Writes down section data.
- PetscErrorCode iSectionSaveVector (mesh_ *mesh, sections *sec, PetscInt iplane, Vec &V, const char *fieldName)

  Saves i-section vector data.
- PetscErrorCode jSectionSaveVector (mesh_ *mesh, sections *sec, PetscInt jplane, Vec &V, const char *fieldName)

  Saves j-section vector data.
- PetscErrorCode kSectionSaveVector (mesh_ *mesh, sections *sec, PetscInt kplane, Vec &V, const char *fieldName)

  Saves k-section vector data.
- PetscErrorCode iSectionSaveScalar (mesh_ *mesh, sections *sec, PetscInt iplane, Vec &V, const char *fieldName)

  Saves i-section scalar data.
- PetscErrorCode jSectionSaveScalar (mesh_ *mesh, sections *sec, PetscInt jplane, Vec &V, const char *fieldName)

  Saves j-section scalar data.

- PetscErrorCode kSectionSaveScalar (mesh_ *mesh, sections *sec, PetscInt kplane, Vec &V, const char *fieldName)

  Saves k-section scalar data.
- PetscErrorCode averageFieldsInitialize (acquisition_ *acquisition)

  Initialize average fields acquisition.
- PetscErrorCode averageFields (acquisition_ *acquisition)

  Average fields.
- PetscErrorCode computeQCritIO (acquisition_ *acquisition)

  Compute Q criteria for I/O in cartesian form.
- PetscErrorCode computeCoriolisIO (acquisition_ *acquisition)

  Compute coriolis force for I/O in cartesian form.
- PetscErrorCode computeDrivingSourceIO (acquisition_ *acquisition)

  Compute driving pressure force for I/O in cartesian form.
- PetscErrorCode computeXDampingIO (acquisition_ *acquisition)

  Compute x damping force for I/O in cartesian form.
- PetscErrorCode computeCanopyForceIO (acquisition_ *acquisition)

  Compute side force for I/O in cartesian form.
- PetscErrorCode computeVelocityDivergence (acquisition_ *acquisition)

  Compute velocity divergence field for I/O.
- PetscErrorCode averageKEBudgetsInitialize (acquisition_ *acquisition)

  Initializes ke budgets fields acquisition.
- PetscErrorCode readKeBoxArray (keFields *ke)

  Read box array.
- PetscErrorCode setKeBoundsAndComms (mesh_ *mesh, keFields *ke)

  Search box bounds and define communicators.
- PetscErrorCode boxCumulateKEBudgets (acquisition_ *acquisition)

  Do box cumulation for mesh. energy budgets.
- PetscErrorCode averageKEBudgets (acquisition_ *acquisition)

  Compute MKE budgets.
- PetscErrorCode averageKEBudgetsCat (acquisition_ *acquisition)

  Compute MKE budgets in cartesian form.
- PetscErrorCode averageKEBudgetsCont (acquisition_ *acquisition)

  Compute MKE budgets in generalized curvilinaer form w/o mesh stretching.
- PetscErrorCode averaging3LMInitialize (domain_ *domain)

  Initialize 3LM data structure.
- PetscErrorCode writeAveraging3LM (domain_ *domain)

  Performs 3LM average and write 3LM fields.
- PetscErrorCode write3LMPoints (acquisition_ *acquisition)

  Write 3LM points to file inside postProcessing/3LM/points.
- PetscErrorCode write3LMFields (acquisition_ *acquisition)

  Write 3LM fields to file inside postProcessing/3LM/.
- PetscErrorCode findAvgLineIds (acquisition_ *acquisition)

  Inizializes the closest cells and the owner of the 3LM mesh points.
- PetscErrorCode read3LMFields (acquisition_ *acquisition)

  Reads velocity and pressure averages (they don't use auxiliary fields like TBL, LBL, IBL)
- PetscErrorCode perturbationABLInitialize (acquisition_ *acquisition)

  Initializes ABL perturbation fields w.r.t. given reference.
- PetscErrorCode averagePerturbationABL (acquisition_ *acquisition)

  Performs perturbation averaging and writes to memory.
- PetscErrorCode averagingABLInitialize (domain_ *domain)

  Initialize ABL data structure.
- PetscErrorCode writeAveragingABL (domain_ *domain)

  Write spatial averaged ABL statistics.

5.8.1 Detailed Description

Acquisition object declaration.

5.8.2 Function Documentation

5.8.2.1 read3LMFields()

```
PetscErrorCode read3LMFields (
            acquisition_ * acquisition )
```

Reads velocity and pressure averages (they don't use auxiliary fields like TBL, LBL, IBL)

read field

5.9 src/include/base.h File Reference

Base header file including c/c++ libraries and PETSc and HYPRE libs.

```
#include <vector>
#include <algorithm>
#include <assert.h>
#include <complex>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dirent.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <iomanip>
#include "petsctime.h"
#include "petscvec.h"
#include "petscdmda.h"
#include "petscksp.h"
#include "petscsnes.h"
#include "HYPRE_krylov.h"
#include "HYPRE.h"
#include "HYPRE_parcsr_ls.h"
#include "HYPRE_struct_ls.h"
#include "HYPRE_sstruct_ls.h"
#include "HYPRE_IJ_mv.h"
```

Classes

- struct simInfo_
- struct constants_
- struct cellIds

  Cell indices.
- struct Cmpnts

  Defines the x, y, z components of a vector.
- struct Cpt2D

  Structure for defining the x, y components of a 2D vector.
- struct symmTensor

  Defines the components of a symmetric tensor.
- struct patchVectorField

  Vector field on the patch (used to store wall models fields)

Macros

- #define **M_PI** 3.14159265358979323846264 3

Typedefs

- typedef std::string **word**
- typedef std::vector< PetscInt > **labelList**
- typedef std::vector< std::vector< PetscInt > > **labelListList**
- typedef std::complex< PetscReal > **complex**

5.9.1 Detailed Description

Base header file including c/c++ libraries and PETSc and HYPRE libs.

5.10 src/include/boundary.h File Reference

Boundary conditions header file.

Classes

- struct scalarBC

  Structure dafining the type of boundary conditions for a scalar.
- struct vectorBC

  Structure dafining the type of boundary conditions for a vector.
- struct inflowData

  Struct defining inflow information.
- struct inletFunctionTypes

  Struct storing the inlet function for a patch.
- struct inletFunctions

  Struct storing inlet functions data.

Functions

- PetscErrorCode readScalarBC (const word &location, const word &field, scalarBC *bc)

  Reads boundary conditions for a scalar field.
- PetscErrorCode readVectorBC (const word &location, const word &field, vectorBC *bc)

  Reads bundary conditions for a vector field.
- PetscErrorCode SetBoundaryConditions (mesh_ *mesh)

  Set boundary conditions.
- PetscErrorCode checkBoundaryConditions (mesh_ *mesh)

  Checks available boundary conditions.
- PetscErrorCode SetPeriodicConnectivity (mesh_ *mesh, word &meshFileName)

  Set periodicconnectivity type.
- PetscErrorCode SetWallModels (ueqn_ *ueqn)

  Set wall models.
- PetscErrorCode UpdateContravariantBCs (ueqn_ *ueqn)

  Update contravariant fluxes boundary conditions.
- PetscErrorCode UpdateCartesianBCs (ueqn_ *ueqn)

  Update cartesian boundary conditions.
- PetscErrorCode UpdateTemperatureBCs (teqn_ *teqn)

  Update temperature boundary conditions.
- PetscErrorCode UpdateNutBCs (les_ *les)

  Update effective viscosity boundary conditions.
- PetscErrorCode UpdatePressureBCs (peqn_ *peqn)

  Update pressure boundary conditions.
- PetscErrorCode UpdatePhiBCs (peqn_ *peqn)

  Update pressure correction boundary conditions.
- PetscErrorCode UpdateImmersedBCs (ibm_ *ibm)

  Update ibm boundary conditions.
- PetscErrorCode UpdateWallModelsU (ueqn_ *ueqn)

  Update wall model for specified wall shear stress.
- PetscErrorCode UpdateWallModelsT (teqn_ *teqn)

  Update wall model for specified wall heat flux.
- PetscErrorCode readSurfaceTempData (Shumann *wm)

  read surface temperature and obhukhov length data

5.10.1 Detailed Description

Boundary conditions header file.

5.11 src/include/clock.h File Reference

Simulation time struct header file.

Classes

- struct clock_

Functions

- PetscErrorCode **adjustTimeStep** (domain_ *domain)
- PetscErrorCode **timeStepInfo** (domain_ *domain, clock_ *clock, PetscReal &dxByU_min, PetscReal &maxU, cellIds &maxUCell)

5.11.1 Detailed Description

Simulation time struct header file.

5.12 src/include/domain.h File Reference

Domain data structure definition.

```
#include "objects.h"
#include "clock.h"
#include "access.h"
#include "flags.h"
#include "mesh.h"
#include "overset.h"
#include "acquisition.h"
#include "ibm.h"
#include "ueqn.h"
#include "peqn.h"
#include "teqn.h"
#include "les.h"
#include "precursor.h"
#include "turbines.h"
#include "abl.h"
```

Classes

- struct domain_

  Domain data structure definition.

5.12.1 Detailed Description

Domain data structure definition.

5.13 src/include/ibm.h File Reference

IBM model header file.

Classes

- struct Vertex
- struct HalfEdge
- struct Face
- struct ibmNode
- struct node

  Node struct.
- struct cellNode
- struct list

  Node list.
- struct cellList

  cell Node list
- struct searchBox
- struct ibmMesh
- struct ibmRotation
- struct ibmSineMotion
- struct ibmPitchMotion
- struct surface
- struct elementBox
- struct ibmObject
- struct ibmFluidCell
- struct ibm_

  struct storing IBM model

Macros

- #define **MAX_ELEMENTS_PER_NODE** 20

Typedefs

- typedef struct Vertex **Vertex**
- typedef struct HalfEdge **HalfEdge**
- typedef struct Face **Face**
- typedef struct node node

  Node struct.
- typedef struct cellNode **cellNode**
- typedef struct list list

  Node list.
- typedef struct cellList cellList

  cell Node list

Functions

- PetscErrorCode InitializeIBM (ibm_ *ibm)

  initialize ibm: top level function
- PetscErrorCode UpdateIBM (ibm_ *ibm)

  update imb: top level function
- PetscErrorCode writeIBMData (ibm_ *ibm, PetscInt b)

  writes the angular position when the body is rotating
- PetscErrorCode writeIBMForceData (ibm_ *ibm, PetscInt b, PetscReal *gElemPressure, PetscReal net↩ Moment, PetscReal ibmPower, Cmpnts PresForce, Cmpnts ViscForce, Cmpnts momentVector)

  writes the IBM force data for element and net force
- PetscErrorCode findIBMControlledProcs (ibm_ *ibm)

  find in which processors the ibm body belongs
- PetscErrorCode initElementProjectionProcs (ibm_ *ibm)

  create list of ibm element processors and their closest ibm fluid cell to the ibm element normal projection
- PetscErrorCode IBMProjectionProcessorTransfer (ibm_ *ibm)

  transfer ibm mesh element across processors based on their movement - for dynamic case
- PetscErrorCode createProcessorBufferZones (ibm_ *ibm)

  create inner and outer buffer zones for ibm mesh element parallelization
- PetscErrorCode initElementProcs (ibm_ *ibm)

  create list of ibm element processors and their closest ibm fluid cell to the ibm element
- PetscErrorCode IBMElementProcessorTransfer (ibm_ *ibm)

  transfer ibm mesh element across processors based on their movement - for dynamic case
- PetscErrorCode UpdateIBMesh (ibm_ *ibm)

  update the ibm mesh when the ibm body is moving
- PetscErrorCode rotateIBMesh (ibm_ *ibm, PetscInt b)

  rotate the ibm mesh based on the angular speed input
- PetscErrorCode sineMotion (ibm_ *ibm, PetscInt b)

  prescribe sinusoidal motion for IBM body
- PetscErrorCode pitchingMotion (ibm_ *ibm, PetscInt b)

  prescribe pitching oscillation motion for IBM body
- PetscErrorCode setIBMWallModels (ibm_ *ibm)

  set IBM wall model type and properties
- PetscErrorCode findBodyBoundingBox (ibm_ *ibm)

  find the bounding box around an ibm body
- PetscErrorCode findSearchCellDim (ibm_ *ibm)

  find the search cell (coarser mesh around the ibm) dimension
- PetscErrorCode ibmSearch (ibm_ *ibm)

  ibm fluid node search performed using ray tracing algorithm
- PetscErrorCode findClosestIBMElement (ibm_ *ibm)

  find the closest ibm mesh element to a IBM fluid node
- PetscErrorCode **findClosestIBMElement2Solid** (ibm_ *ibm)
- PetscErrorCode findInterceptionPoint (ibm_ *ibm)

  find the interceptionPoint on the background mesh plane for curvib interpolation
- PetscErrorCode interceptionPt (Cmpnts pCoor, Cmpnts pc[9], Cmpnts eNorm, ibmFluidCell *ibF)

  interception point algorithm based on dividing the plane into 8 triangles
- PetscErrorCode findIBMFluidCells (ibm_ *ibm)

  find list of ibm fluid nodes within each processor
- PetscErrorCode findFluidSupportNodes (ibm_ *ibm)

  find the fluid nodes that act as support nodes to the given ibm fluid node
- PetscErrorCode findIBMMeshSupportNodes (ibm_ *ibm)

find the ibm mesh nodes that act as support node to the given ibm fluid node

- PetscErrorCode MLSInterpolation (ibm_ *ibm)

  MLS interpolation algorithm.
- PetscErrorCode CurvibInterpolation (ibm_ *ibm)

  CURVIB normal projection interpolation algorithm.
- PetscErrorCode **CurvibInterpolationTriangular** (ibm_ *ibm)
- PetscErrorCode **CurvibInterpolationQuadratic** (ibm_ *ibm)
- PetscErrorCode **CurvibInterpolationInternalCell** (ibm_ *ibm)
- PetscErrorCode **ComputeForceMoment** (ibm_ *ibm)
- PetscErrorCode findIBMWallShear (ibm_ *ibm)

  compute shear stress at faces close to the IBM
- PetscErrorCode findIBMWallShearChester (ibm_ *ibm)

  compute shear stress at faces close to the IBM
- PetscErrorCode **checkIBMexists** (ibm_ *ibm)
- PetscErrorCode createSearchCellList (ibm_ *ibm)

  insert ibm mesh elements as a list into the search cell that they belong to
- PetscErrorCode destroyLists (ibm_ *ibm)

  destroy the ibm search cell list
- PetscErrorCode **computeIBMElementNormal** (ibm_ *ibm)
- PetscErrorCode **recomputeIBMeshProperties** (ibm_ *ibm, PetscInt b)
- PetscErrorCode **createHalfEdgeDataStructure** (ibm_ *ibm)
- PetscReal rayCastingTest (Cmpnts p, ibmMesh *ibMsh, cellIds sCell, searchBox *sBox, boundingBox *ib↩Box, list *searchCellList)

  ray casting algorithm
- PetscErrorCode rayCastLocal (Cmpnts p, Cmpnts p1, Cmpnts p2, Cmpnts p3, Cmpnts p4, ibmMesh *ibMsh, cellIds sCell, searchBox *sBox, boundingBox *ibBox, list *searchCellList, PetscInt &intersect)

  ray casting algorithm in local positive cell neighbourhood
- PetscErrorCode elementBoundingSphere (ibmObject *ibmBody)

  find element bounding sphere
- Cmpnts randomdirection (Cmpnts p, cellIds sCell, boundingBox *ibBox, searchBox *sBox, PetscInt seed)

  inline function to generate a random direction along the z index of the search cell for the ray casting algorithm
- PetscInt intsectElement (Cmpnts p, Cmpnts dir, Cmpnts node1, Cmpnts node2, Cmpnts node3, PetscReal *t, PetscReal *u, PetscReal *v)

  inline function to check if the ray intersects an ibm element
- PetscBool isLineTriangleInt (Cmpnts p1, Cmpnts p2, ibmMesh *ibMsh, PetscInt e)

  inline function to check if the line intersects an ibm element
- bool isSearchCellSupport (PetscReal rad, Cmpnts pt, cellIds cID, boundingBox *ibBox, searchBox *sBox)

  inline function to check if a search cell is a support to an ibm fluid node
- void insertIBMSupportNodes (Cmpnts pt, cellIds sCell, ibmFluidCell *ibF, ibmObject *ibBody, searchBox *sBox)

  insert ibm nodes into the list of support nodes around an ibm fluid node
- PetscInt ISSameSide2D (Cpt2D p, Cpt2D p1, Cpt2D p2, Cpt2D p3)

  Check whether 2D point p is located on the same side of line p1p2.
- PetscInt ISInsideTriangle2D (Cpt2D p, Cpt2D pa, Cpt2D pb, Cpt2D pc)

  check whether point p is inside the triangle - 2D
- PetscInt isPointInTriangle (Cmpnts p, Cmpnts p1, Cmpnts p2, Cmpnts p3, Cmpnts norm)

  check whether point p is inside the triangle - 3D
- Cmpnts computeVertexAverageNormal (ibmMesh *ibMsh, PetscInt vertexId)

  find the angle averaged normal about a vertex
- Cmpnts computeEdgeAverageNormal (ibmMesh *ibMsh, PetscInt vertexId, PetscInt faceId)

  find the angle averaged normal about an edge
- void disP2Line (Cmpnts p, Cmpnts p1, Cmpnts p2, Cmpnts *po, PetscReal *d, PetscReal *t)

find projection of point on a line and the distance to it

- void triangleIntp (Cpt2D p, Cpt2D p1, Cpt2D p2, Cpt2D p3, ibmFluidCell *ibF)

  find the interpolation weights of a point inside a triangle from its nodes
- void triangleIntpBg (Cpt2D p, Cpt2D p1, Cpt2D p2, Cpt2D p3, ibmFluidCell *ibF)

  find the interpolation weights of a point inside a triangle from its nodes used for background grid

5.13.1 Detailed Description

IBM model header file.

IBM input header file.

5.14 src/include/inflow.h File Reference

Contains inflow boundary condition headers.

Functions

- PetscErrorCode SetInflowFunctions (mesh_ *mesh)

  Initialize the inlet functions for inflow.
- PetscErrorCode printInflowMappingAction (mesh_ *mesh, inletFunctionTypes *ifPtr)

  print information regarding the inflow boundary mapping
- PetscErrorCode mappedInflowInitialize (inletFunctionTypes *ifPtr)

  allocate and set the inflow section fields
- PetscErrorCode SetInflowWeights (mesh_ *mesh, inletFunctionTypes *ifPtr)

  Calculate interpolation weights for the mapped interpolated inflow boundary condition.
- PetscErrorCode readInflowU (inletFunctionTypes *ifPtr, clock_ *clock)

  Reads inflow velocity from database and stores in inletFunction data.
- PetscErrorCode readInflowT (inletFunctionTypes *ifPtr, clock_ *clock)

  Reads inflow tempereature from database and stores in inletFunction data.
- PetscErrorCode readInflowNut (inletFunctionTypes *ifPtr, clock_ *clock)

  Reads inflow sgs viscosity from database and stores in inletFunction data.
- PetscErrorCode setShiftedInflowU (inletFunctionTypes *ifPtr, ueqn_ *ueqn)

  Sets the un-shifted boundary field to be used with shifted periodic BC (inflow function type 7)
- PetscErrorCode setShiftedInflowT (inletFunctionTypes *ifPtr, teqn_ *teqn)

  Sets the un-shifted boundary field to be used with shifted periodic BC (inflow function type 7)
- PetscErrorCode setShiftedInflowNut (inletFunctionTypes *ifPtr, les_ *les)

  Sets the un-shifted boundary field to be used with shifted periodic BC (inflow function type 7)
- Cmpnts NieuwstadtInflowEvaluate (inletFunctionTypes *ifPtr, PetscReal h)

  Nieuwstadt model for velocity inflow type 5.

5.14.1 Detailed Description

Contains inflow boundary condition headers.

5.15 src/include/initialField.h File Reference

initial field header file.

Functions

- PetscErrorCode SetInitialField (domain_ *domain)
  <
- PetscErrorCode SetInitialFieldPrecursor (abl_ *abl)
  set the initial internal contravariant and cartesian velocity field
- PetscErrorCode SetInitialFieldU (ueqn_ *ueqn)
  set the initial temperature field
- PetscErrorCode SetInitialFieldT (teqn_ *teqn)
  set the initial pressure field
- PetscErrorCode SetInitialFieldP (peqn_ *peqn)
  set the initial variables for LES
- PetscErrorCode SetInitialFieldLES (les_ *les)
  set uniform value for the cartesian velocity (add perturbations if applicable)
- PetscErrorCode SetUniformFieldU (ueqn_ *ueqn, Cmpnts &uRef, PetscInt &addPerturbations)
  set initial ABL flow U
- PetscErrorCode SetABLInitialFlowU (ueqn_ *ueqn)
  set the internal field as the spreaded inlet flow condition
- PetscErrorCode SpreadInletFlowU (ueqn_ *ueqn)
  set uniform value for the temperature
- PetscErrorCode SetUniformFieldT (teqn_ *teqn, PetscReal &tRef)
  set linear profile for the temperature
- PetscErrorCode SetLinearFieldT (teqn_ *teqn, PetscReal &tRef, PetscReal &tLapse)
  set initial ABL flow T
- PetscErrorCode SetABLInitialFlowT (teqn_ *teqn)
  set the internal field as the spreaded inlet flow condition
- PetscErrorCode **SpreadInletFlowT** (teqn_ *teqn)

5.15.1 Detailed Description

initial field header file.

5.15.2 Function Documentation

5.15.2.1 SetInitialField()

```
PetscErrorCode SetInitialField (
            domain_ * domain )
```

<

set the initial internal fields

set the initial internal fields for the concurrent precursor simulation

5.16 src/include/initialization.h File Reference

Contains simulation initialization function headers.

Functions

- PetscErrorCode PrintOkWindLogo ()
  Print OkWind logo.
- PetscErrorCode PrintNumberOfProcs ()
  Print number of processors.
- PetscErrorCode SetSimulationFlags (flags_ *flags)
  Set simulation flags.
- PetscErrorCode SetSimulationInfo (simInfo_ *info)
  Set simulation global info.
- PetscErrorCode SetDomainsAndAllocate (domain_ **domain, flags_ *flags, simInfo_ *info)
  Set number of domains (reads from OversetInput.dat file if overset is active)
- PetscErrorCode ReadTimeControls (clock_ *clock)
  Read time controls.
- PetscErrorCode SetStartTime (clock_ *clock, domain_ *domain, simInfo_ *info)
  Get startFrom parameter and set initial time (check multiple domains cosistency)
- PetscErrorCode ReadPhysicalConstants (domain_ *domain)
  Read physical constants.
- PetscErrorCode SetDomainMemory (domain_ *domain)
  Allocate memory for domain pointers.
- PetscErrorCode SetAccessPointers (domain_ *domain)
  Set access database pointers.
- PetscErrorCode simulationInitialize (domain_ **domainAddr, clock_ *clock, simInfo_ *info, flags_ *flags)
  Initialize the simulation parameters.
- PetscErrorCode SetInitialField (domain_ *domain)
  set the initial internal contravariant and cartesian velocity field
- PetscErrorCode **SetInitialField** (ueqn_ *ueqn)

5.16.1 Detailed Description

Contains simulation initialization function headers.

5.16.2 Function Documentation

5.16.2.1 SetInitialField()

```
PetscErrorCode SetInitialField (
            domain_ * domain )
```

set the initial internal contravariant and cartesian velocity field

set the initial internal contravariant and cartesian velocity field

set the initial internal fields

set the initial internal fields for the concurrent precursor simulation

5.16.2.2 simulationInitialize()

```
PetscErrorCode simulationInitialize (
            domain_ ** domainAddr,
            clock_ * clock,
            simInfo_ * info,
            flags_ * flags )
```

Initialize the simulation parameters.

set the initial internal fields

5.17 src/include/inline.h File Reference

Inline functions.

```
#include "io.h"
#include "ibm.h"
```

Functions

- PetscReal **currentDistanceToWriteTime** (clock_ *clock, PetscReal timeStart, PetscReal timeInterval)
- void **timeStepSet** (clock_ *clock, PetscReal timeStart, PetscReal timeInterval, PetscReal dByU, PetscInt &flag, PetscReal &cfl)
- PetscInt **mustWrite** (PetscReal time, PetscReal startTime, PetscReal timeInterval)
- void **matMatProduct** (PetscReal **A, PetscReal **B, PetscReal **C, PetscInt numRowA, PetscInt num↩ ColA, PetscInt numRowB, PetscInt numColB)
- void **matVecProduct** (PetscReal **A, PetscReal *b, PetscReal *c, PetscInt numRowA, PetscInt numColA, PetscInt numRowB)
- PetscInt **isPresent** (PetscInt arr[], PetscInt n, PetscInt elem)
- PetscReal **sign** (PetscReal a)
- PetscReal **signNonZero** (PetscReal a)
- PetscReal **gcd** (PetscReal a, PetscReal b)
- PetscReal **gcdN** (PetscReal *v, PetscInt n)
- complex **gamma** (complex x)
- complex **digamma** (complex x)
- complex **hypergeom** (complex a, complex b, complex c, PetscReal z)
- PetscReal **polyLog2** (PetscReal x)
- PetscReal **splineB1** (PetscReal a)
- PetscReal **splineB2** (PetscReal a)
- Cmpnts **nMax** (Cmpnts v, PetscReal c)
- Cmpnts **nAbs** (Cmpnts v)
- Cmpnts **nInv** (Cmpnts v)
- Cmpnts **nPow** (Cmpnts v, PetscReal c)
- Cmpnts **nScale** (PetscReal c, Cmpnts v)
- Cmpnts **nMultElWise** (Cmpnts v1, Cmpnts v2)
- Cmpnts **nDivElWise** (Cmpnts v1, Cmpnts v2)
- Cmpnts **nScaleX** (PetscReal c, Cmpnts v)
- Cmpnts **nScaleY** (PetscReal c, Cmpnts v)
- Cmpnts **nScaleZ** (PetscReal c, Cmpnts v)
- PetscReal **nDot** (Cmpnts v1, Cmpnts v2)

- PetscReal **nMag** (Cmpnts v)
- Cmpnts **nCross** (Cmpnts v1, Cmpnts v2)
- Cmpnts **nRot** (Cmpnts axis, Cmpnts vStart, PetscReal theta)
- Cmpnts **nTra** (Cmpnts point, Cmpnts translation)
- Cmpnts **nUnit** (Cmpnts v)
- Cmpnts **nSum** (Cmpnts v1, Cmpnts v2)
- Cmpnts **nSub** (Cmpnts v1, Cmpnts v2)
- Cmpnts **nSet** (Cmpnts value)
- Cmpnts **nSetFromComponents** (PetscReal vx, PetscReal vy, PetscReal vz)
- Cmpnts **nSetZero** ()
- void **mUnit** (Cmpnts &base)
- void **mSum** (Cmpnts &base, Cmpnts add)
- void **mSub** (Cmpnts &base, Cmpnts sub)
- void **mScale** (PetscReal c, Cmpnts &v)
- void **mScaleX** (PetscReal c, Cmpnts &v)
- void **mScaleY** (PetscReal c, Cmpnts &v)
- void **mScaleZ** (PetscReal c, Cmpnts &v)
- void **mSetScale** (PetscReal scale, Cmpnts &a, Cmpnts b)
- void **mRot** (Cmpnts axis, Cmpnts &v, PetscReal theta)
- void **mTra** (Cmpnts &point, Cmpnts translation)
- void **mSet** (Cmpnts &base, Cmpnts value)
- void **mSetValue** (Cmpnts &base, PetscReal value)
- void AxByC (PetscReal a, Cmpnts &X, PetscReal b, Cmpnts &Y, Cmpnts *C)

  C=aX+bY.
- bool **isInsideBoundingBox** (Cmpnts pt, const boundingBox &simBox)
- void **Calculate_Covariant_metrics** (PetscReal g[3][3], PetscReal G[3][3])
- void **calculateNormal** (Cmpnts &csi, Cmpnts &eta, Cmpnts &zet, PetscReal ni[3], PetscReal nj[3], PetscReal nk[3])
- bool isBoxIBMCell (int k, int j, int i, PetscReal ***nvert)

  check if ibm cell within the neighbouring box of i,j,k
- bool isFluidCell (PetscInt k, PetscInt j, PetscInt i, PetscReal ***nvert)

  check if purely a fluid cell
- bool isIBMCell (PetscInt k, PetscInt j, PetscInt i, PetscReal ***nvert)

  check if IBM cell - this includes IBM fluid cells (cell next to IBM body) and IBM solid cells
- bool isIBMFluidCell (PetscInt k, PetscInt j, PetscInt i, PetscReal ***nvert)

  check if IBM Fluid cell (cell next to IBM body)
- bool isIBMSolidCell (PetscInt k, PetscInt j, PetscInt i, PetscReal ***nvert)

  check if IBM solid cell
- bool isFluidKFace (PetscInt kL, PetscInt j, PetscInt i, PetscInt kR, PetscReal ***nvert)

  check is a fluid k face - shared by 2 fluid cells
- bool isFluidJFace (PetscInt k, PetscInt jL, PetscInt i, PetscInt jR, PetscReal ***nvert)

  check is a fluid j face - shared by 2 fluid cells
- bool isFluidIFace (PetscInt k, PetscInt j, PetscInt iL, PetscInt iR, PetscReal ***nvert)

  check is a fluid i face - shared by 2 fluid cells
- bool isIBMKFace (PetscInt kL, PetscInt j, PetscInt i, PetscInt kR, PetscReal ***nvert)

  check is a IBM k face - shared by atleast one IBM cell
- bool isIBMJFace (PetscInt k, PetscInt jL, PetscInt i, PetscInt jR, PetscReal ***nvert)

  check is a IBM j face - shared by atleast one IBM cell
- bool isIBMIFace (PetscInt k, PetscInt j, PetscInt iL, PetscInt iR, PetscReal ***nvert)

  check is a IBM i face - shared by atleast one IBM cell
- bool isIBMFluidKFace (PetscInt kL, PetscInt j, PetscInt i, PetscInt kR, PetscReal ***nvert)

  check if IBM fluid k face - atleast one of the shared cells is IBM fluid
- bool isIBMFluidJFace (PetscInt k, PetscInt jL, PetscInt i, PetscInt jR, PetscReal ***nvert)

check if IBM fluid j face - at least one of the shared cells is IBM fluid

- bool isIBMFluidIFace (PetscInt k, PetscInt j, PetscInt iL, PetscInt iR, PetscReal ***nvert)

  check if IBM fluid i face - at least one of the shared cells is IBM fluid
- bool **isIBMSolidKFace** (PetscInt kL, PetscInt j, PetscInt i, PetscInt kR, PetscReal ***nvert)
- bool isIBMSolidJFace (PetscInt k, PetscInt jL, PetscInt i, PetscInt jR, PetscReal ***nvert)

  check if IBM fluid j face - at least one of the shared cells is IBM fluid
- bool isIBMSolidIFace (PetscInt k, PetscInt j, PetscInt iL, PetscInt iR, PetscReal ***nvert)

  check if IBM fluid i face - at least one of the shared cells is IBM fluid
- void resetNonResolvedCellCentersScalar (mesh_ *mesh, Vec &V)

  Resets the value at the non-solved centers to zero.
- void ContravariantToCartesianPoint (Cmpnts &csi, Cmpnts &eta, Cmpnts &zet, Cmpnts &ucont, Cmpnts *ucat)

  Interpolates cartesian velocity at cell center from the contravariant flux around a cell.
- void resetNonResolvedCellFaces (mesh_ *mesh, Vec &V)

  Resets the value at the non-solved faces to zero except for zeroGradient, overset and periodic.
- bool **isOnNonResolvedCellCenters** (PetscInt i, PetscInt j, PetscInt k, DMDALocalInfo info)
- bool **isOnCornerCellCenters** (PetscInt i, PetscInt j, PetscInt k, DMDALocalInfo info)
- void getCell2Cell3StencilCsiNoGhost (mesh_ *mesh, PetscInt i, PetscInt mx, PetscInt *iL, PetscInt *iR)

  Get cell's idxs of a 3 cell-sencil in csi direction given the cell idx. NEVER USE GHOSTS.
- void getCell2Cell3StencilEtaNoGhost (mesh_ *mesh, PetscInt j, PetscInt my, PetscInt *jL, PetscInt *jR)

  Get cell's idxs of a 3 cell-sencil in eta direction given the cell idx. NEVER USE GHOSTS.
- void getCell2Cell3StencilZetNoGhost (mesh_ *mesh, PetscInt k, PetscInt mz, PetscInt *kL, PetscInt *kR)

  Get cell's idxs of a 3 cell-sencil in zet direction given the cell idx. NEVER USE GHOSTS.
- void getCell2Cell3StencilCsi (mesh_ *mesh, PetscInt i, PetscInt mx, PetscInt *iL, PetscInt *iR)

  Get cell's idxs of a 3 cell-sencil in csi direction given the cell idx.
- void getCell2Cell3StencilEta (mesh_ *mesh, PetscInt j, PetscInt my, PetscInt *jL, PetscInt *jR)

  Get cell's idxs of a 3 cell-sencil in eta direction given the cell idx.
- void getCell2Cell3StencilZet (mesh_ *mesh, PetscInt k, PetscInt mz, PetscInt *kL, PetscInt *kR)

  Get cell's idxs of a 3 cell-sencil in zet direction given the cell idx.
- void getFace2Cell4StencilCsi (mesh_ *mesh, PetscInt k, PetscInt j, PetscInt i, PetscInt mx, PetscInt *iL, PetscInt *iR, PetscReal *denom, PetscReal ***nvert)

  Get outmost cell's idxs of a 4 cell-sencil in csi direction given the face idx.
- void getFace2Cell4StencilEta (mesh_ *mesh, PetscInt k, PetscInt j, PetscInt i, PetscInt my, PetscInt *jL, PetscInt *jR, PetscReal *denom, PetscReal ***nvert)

  Get outmost cell's idxs of a 4 cell-sencil in eta direction given the face idx.
- void getFace2Cell4StencilZet (mesh_ *mesh, PetscInt k, PetscInt j, PetscInt i, PetscInt mz, PetscInt *kL, PetscInt *kR, PetscReal *denom, PetscReal ***nvert)

  Get outmost cell's idxs of a 4 cell-sencil in zet direction given the face idx.
- void getFace2Face3StencilCsi (mesh_ *mesh, PetscInt i, PetscInt mx, PetscInt *iL, PetscInt *iR)

  Get neighboring face idxs of a 3 face-sencil in csi direction given the central face idx.
- void getFace2Face3StencilEta (mesh_ *mesh, PetscInt j, PetscInt my, PetscInt *jL, PetscInt *jR)

  Get neighboring face idxs of a 3 face-sencil in eta direction given the central face idx.
- void getFace2Face3StencilZet (mesh_ *mesh, PetscInt k, PetscInt mz, PetscInt *kL, PetscInt *kR)

  Get neighboring face idxs of a 3 face-sencil in zeta direction given the central face idx.
- void **Compute_du_i** (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, Cmpnts ***ucat, PetscReal ***nvert, PetscReal *dudc, PetscReal *dvdc, PetscReal *dwdc, PetscReal *dude, PetscReal *dvde, PetscReal *dwde, PetscReal *dudz, PetscReal *dvdz, PetscReal *dwdz)
- void **Compute_du_j** (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, Cmpnts ***ucat, PetscReal ***nvert, PetscReal *dudc, PetscReal *dvdc, PetscReal *dwdc, PetscReal *dude, PetscReal *dvde, PetscReal *dwde, PetscReal *dudz, PetscReal *dvdz, PetscReal *dwdz)
- void **Compute_du_k** (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, Cmpnts ***ucat, PetscReal ***nvert, PetscReal *dudc, PetscReal *dvdc, PetscReal *dwdc, PetscReal *dude, PetscReal *dvde, PetscReal *dwde, PetscReal *dudz, PetscReal *dvdz, PetscReal *dwdz)

- void **Compute_dscalar_i** (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, PetscReal ***K, PetscReal ***nvert, PetscReal *dkdc, PetscReal *dkde, PetscReal *dkdz)
- void **Compute_dscalar_j** (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, PetscReal ***K, PetscReal ***nvert, PetscReal *dkdc, PetscReal *dkde, PetscReal *dkdz)
- void **Compute_dscalar_k** (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, PetscReal ***K, PetscReal ***nvert, PetscReal *dkdc, PetscReal *dkde, PetscReal *dkdz)
- void **Compute_du_center** (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, Cmpnts ***ucat, PetscReal ***nvert, PetscReal *dudc, PetscReal *dvdc, PetscReal *dwdc, PetscReal *dude, PetscReal *dvde, PetscReal *dwde, PetscReal *dudz, PetscReal *dvdz, PetscReal *dwdz)
- void **Compute_dscalar_center** (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, PetscReal ***K, PetscReal ***nvert, PetscReal *dkdc, PetscReal *dkde, PetscReal *dkdz)
- PetscReal **integrateTestfilterSimpson** (PetscReal val[3][3][3], PetscReal w[3][3][3])
- PetscReal **integrateTestfilterSimpson5x5** (PetscReal val[5][5][5], PetscReal w[5][5][5])
- void **Compute_du_dxyz** (mesh_ *mesh, PetscReal csi0, PetscReal csi1, PetscReal csi2, PetscReal eta0, PetscReal eta1, PetscReal eta2, PetscReal zet0, PetscReal zet1, PetscReal zet2, PetscReal ajc, Petsc↩ Real dudc, PetscReal dvdc, PetscReal dwdc, PetscReal dude, PetscReal dvde, PetscReal dwde, PetscReal dudz, PetscReal dvdz, PetscReal dwdz, PetscReal *du_dx, PetscReal *dv_dx, PetscReal *dw_dx, PetscReal *du_dy, PetscReal *dv_dy, PetscReal *dw_dy, PetscReal *du_dz, PetscReal *dv_dz, PetscReal *dw_dz)
- void **Compute_dscalar_dxyz** (mesh_ *mesh, PetscReal csi0, PetscReal csi1, PetscReal csi2, Petsc↩ Real eta0, PetscReal eta1, PetscReal eta2, PetscReal zet0, PetscReal zet1, PetscReal zet2, PetscReal ajc, PetscReal dkdc, PetscReal dkde, PetscReal dkdz, PetscReal *dk_dx, PetscReal *dk_dy, PetscReal *dk_dz)
- void **Compute_du_wmLocal** (mesh_ *mesh, Cmpnts eN, Cmpnts eT1, Cmpnts eT2, PetscReal du_dx, PetscReal dv_dx, PetscReal dw_dx, PetscReal du_dy, PetscReal dv_dy, PetscReal dw_dy, PetscReal du_dz, PetscReal dv_dz, PetscReal dw_dz, PetscReal *dut1dn, PetscReal *dut2dn, PetscReal *dundn, Petsc↩ Real *dut1dt1, PetscReal *dut2dt1, PetscReal *dundt1, PetscReal *dut1dt2, PetscReal *dut2dt2, PetscReal *dundt2)
- void **Comput_JacobTensor_i** (PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, Cmpnts ***coor, PetscReal *dxdc, PetscReal *dxde, PetscReal *dxdz, PetscReal *dydc, PetscReal *dyde, PetscReal *dydz, PetscReal *dzdc, PetscReal *dzde, PetscReal *dzdz)
- void **Comput_JacobTensor_j** (PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, Cmpnts ***coor, PetscReal *dxdc, PetscReal *dxde, PetscReal *dxdz, PetscReal *dydc, PetscReal *dyde, PetscReal *dydz, PetscReal *dzdc, PetscReal *dzde, PetscReal *dzdz)
- void **Comput_JacobTensor_k** (PetscInt i, PetscInt j, PetscInt k, PetscInt mx, PetscInt my, PetscInt mz, Cmpnts ***coor, PetscReal *dxdc, PetscReal *dxde, PetscReal *dxdz, PetscReal *dydc, PetscReal *dyde, PetscReal *dydz, PetscReal *dzdc, PetscReal *dzde, PetscReal *dzdz)
- void **Compute_du_Compgrid** (PetscReal dxdc, PetscReal dxde, PetscReal dxdz, PetscReal dydc, Petsc↩ Real dyde, PetscReal dydz, PetscReal dzdc, PetscReal dzde, PetscReal dzdz, PetscReal nx, PetscReal ny, PetscReal nz, PetscReal t1x, PetscReal t1y, PetscReal t1z, PetscReal t2x, PetscReal t2y, PetscReal t2z, PetscReal dut1dn, PetscReal dut2dn, PetscReal dundn, PetscReal dut1dt1, PetscReal dut2dt1, Petsc↩ Real dundt1, PetscReal dut1dt2, PetscReal dut2dt2, PetscReal dundt2, PetscReal *dudc, PetscReal *dvdc, PetscReal *dwdc, PetscReal *dude, PetscReal *dvde, PetscReal *dwde, PetscReal *dudz, PetscReal *dvdz, PetscReal *dwdz)
- PetscReal **minMod** (PetscReal m1, PetscReal m2)
- PetscReal **vanLeer** (PetscReal f0, PetscReal f1, PetscReal f2)
- PetscReal **central** (PetscReal f0, PetscReal f1)
- Cmpnts **centralVec** (Cmpnts f0, Cmpnts f1)
- symmTensor **centralSymmT** (symmTensor f0, symmTensor f1)
- PetscReal **wCentral** (PetscReal f0, PetscReal f1, PetscReal d0, PetscReal d1)
- Cmpnts **wCentralVec** (Cmpnts f0, Cmpnts f1, PetscReal d0, PetscReal d1)
- PetscReal **central4** (PetscReal f0, PetscReal f1, PetscReal f2, PetscReal f3)
- PetscReal **centralUpwind** (PetscReal f0, PetscReal f1, PetscReal f2, PetscReal f3, PetscReal wavespeed)
- Cmpnts **centralUpwindVec** (Cmpnts f0, Cmpnts f1, Cmpnts f2, Cmpnts f3, PetscReal wavespeed)
- PetscReal **centralUpwind** (PetscReal f0, PetscReal f1, PetscReal f2, PetscReal f3, PetscReal wavespeed, PetscReal limiter)
- Cmpnts **centralUpwindVec** (Cmpnts f0, Cmpnts f1, Cmpnts f2, Cmpnts f3, PetscReal wavespeed, PetscReal limiter)

- PetscReal **wCentralUpwind** (PetscReal f0, PetscReal f1, PetscReal f2, PetscReal f3, PetscReal d0, PetscReal d1, PetscReal d2, PetscReal d3, PetscReal wavespeed, PetscReal limiter)
- Cmpnts **wCentralUpwindVec** (Cmpnts f0, Cmpnts f1, Cmpnts f2, Cmpnts f3, PetscReal d0, PetscReal d1, PetscReal d2, PetscReal d3, PetscReal wavespeed, PetscReal limiter)
- PetscReal **quadraticUpwind** (PetscReal f0, PetscReal f1, PetscReal f2, PetscReal f3, PetscReal wavespeed)
- Cmpnts **quadraticUpwindVec** (Cmpnts f0, Cmpnts f1, Cmpnts f2, Cmpnts f3, PetscReal wavespeed)
- PetscReal **wQuadraticUpwind** (PetscReal f0, PetscReal f1, PetscReal f2, PetscReal f3, PetscReal d0, PetscReal d1, PetscReal d2, PetscReal d3, PetscReal wavespeed)
- PetscReal **weno3** (PetscReal f0, PetscReal f1, PetscReal f2, PetscReal f3, PetscReal wavespeed)
- Cmpnts **weno3Vec** (Cmpnts f0, Cmpnts f1, Cmpnts f2, Cmpnts f3, PetscReal wavespeed)
- PetscReal **weno5** (PetscReal f0, PetscReal f1, PetscReal f2, PetscReal f3, PetscReal f4, PetscReal f5, PetscReal wavespeed)
- void **resetNoPenetrationFluxes** (ueqn_ *ueqn)
- void **resetFacePeriodicFluxesVector** (mesh_ *mesh, Vec V, Vec lV, const char *scatterType)
- void resetCellPeriodicFluxes (mesh_ *mesh, Vec &V, Vec &lV, const char *type, const char *scatterType)

  Reset periodic values and scatter.
- PetscReal **viscRayleigh** (PetscReal &alpha, PetscReal &hS, PetscReal &hE, PetscReal &h)
- PetscReal **viscCosAscending** (PetscReal &alpha, PetscReal &hS, PetscReal &hE, PetscReal &h)
- PetscReal **viscCosDescending** (PetscReal &alpha, PetscReal &hS, PetscReal &hE, PetscReal &h)
- PetscReal **viscNordstrom** (PetscReal &alpha, PetscReal &hS, PetscReal &hE, PetscReal &delta, PetscReal &h)
- double **viscNordstromNoVertFilter** (double hS, double hE, double &delta, double &h)
- double **viscStipa** (double &hS, double &hE, double &delta, double &h, double &z, double &H)
- double **viscStipaDelta** (double &hS, double &hE, double &deltaS, double &deltaE, double &h, double &z, double &H)
- void **findInterpolationWeights** (PetscReal *weights, PetscInt *labels, PetscReal *pvec, PetscInt npts, PetscReal pval)
- void **findInterpolationWeightsWithExtrap** (PetscReal *weights, PetscInt *labels, PetscReal *pvec, PetscInt npts, PetscReal pval)
- PetscReal **scaleHyperTangBot** (PetscReal h, PetscReal H, PetscReal delta)
- PetscReal **scaleHyperTangTop** (PetscReal h, PetscReal H, PetscReal delta)
- PetscReal computeEm (Cmpnts &avgU, symmTensor &avgUprimeUprime, PetscReal avgP)

  Computes mechanical energy (Em = $\frac{1}{2}$ <u><u> + <v><v> + <w><w> + <u'u'> + <v'v'> + <w'w'> +.
- PetscReal computeEmTilde (Cmpnts &avgU, symmTensor &avgUprimeUprime)

  Computes modified mechanical energy (Em = $\frac{1}{2}$ <u><u> + <V><V> + <W><W> + <U'U'> + <V'V'> + <W'W'>) using contrav. fluxes.
- PetscReal **computeMKE** (Cmpnts &avgU)
- PetscReal **computeTKE** (symmTensor &avgUprimeUprime)
- void scalarPointLocalVolumeInterpolation (mesh_ *mesh, PetscReal px, PetscReal py, PetscReal pz, PetscInt ic, PetscInt jc, PetscInt kc, Cmpnts ***cent, PetscReal ***v, PetscReal &result)

  Trilinear volume interpolation function for scalars.
- void PointInterpolationWeights (mesh_ *mesh, PetscReal px, PetscReal py, PetscReal pz, PetscInt ic, PetscInt jc, PetscInt kc, Cmpnts ***cent, PetscReal *intWts, PetscInt *intId)

  Trilinear volume interpolation function for scalars to return interpolation Weights.
- void PointInterpolationCells (mesh_ *mesh, PetscReal px, PetscReal py, PetscReal pz, PetscInt ic, PetscInt jc, PetscInt kc, Cmpnts ***cent, PetscInt *intId)

  Trilinear volume interpolation function for scalars to return interpolation cells.
- void vectorPointLocalVolumeInterpolation (mesh_ *mesh, PetscReal px, PetscReal py, PetscReal pz, PetscInt ic, PetscInt jc, PetscInt kc, Cmpnts ***cent, Cmpnts ***v, Cmpnts &result)

  Trilinear volume interpolation function for vectors.
- void **initlist** (list *ilist)
- void **initCellList** (cellList *ilist)
- bool **insertnode** (list *ilist, PetscInt Node)
- void **insertnode1** (list *ilist, PetscInt Node)

- bool **insertCellNode** (cellList *ilist, cellIds Node)
- void **insertCellNode1** (cellList *ilist, cellIds Node)
- void **destroy** (list *ilist)
- void **destroyCellList** (cellList *ilist)
- void **reorderMatrix** (PetscReal **A, PetscReal **B, PetscInt N, PetscInt pivot)
- void **elimJordanAlg** (PetscReal **A, PetscReal **B, PetscInt N, PetscInt pivot)
- void **inverseMatrix** (PetscReal **A, PetscReal **inv_A, PetscInt size)
- void **inv_4by4** (PetscReal **A, PetscReal **inv_A, PetscInt size)
- void **inv_3by3** (PetscReal **A, PetscReal **inv_A, PetscInt size)
- void **reorder_20** (PetscReal A[ ][20], PetscReal B[ ][20], PetscInt N, PetscInt pivot)
- void **elim_jordan_20** (PetscReal A[ ][20], PetscReal B[ ][20], PetscInt N, PetscInt pivot)
- void **inv_20** (PetscReal A[20][20], PetscReal inv_A[ ][20], PetscInt N)
- void **reorder_10** (PetscReal A[ ][10], PetscReal B[ ][10], PetscInt N, PetscInt pivot)
- void **elim_jordan_10** (PetscReal A[ ][10], PetscReal B[ ][10], PetscInt N, PetscInt pivot)
- void **inv_10** (PetscReal A[10][10], PetscReal inv_A[ ][10], PetscInt N)
- void **mult_mats3_lin** (PetscReal **inv_A, PetscReal **B, PetscInt nsupport, PetscReal *PHI)

5.17.1 Detailed Description

Inline functions.

5.17.2 Function Documentation

5.17.2.1 computeEm()

```
PetscReal computeEm (
            Cmpnts & avgU,
            symmTensor & avgUprimeUprime,
            PetscReal avgP )  [inline]
```

Computes mechanical energy (Em = + <v><v> + <w><w> + <u'u'> + <v'v'> + <w'w'> +.

/rho)

5.18 src/include/io.h File Reference

Contains i/o operations function headers.

Classes

- struct io_

  Struct defining io settings (simulation checkpointing)

Functions

- void fatalErrorInFunction (const char *functionName, const char *errorMsg)

  Calls fatal error and exits.
- void warningInFunction (const char *functionName, const char *wrngMsg)

  Calls warning.
- word thisCaseName ()

  Retrieves this case name.
- PetscErrorCode getTimeList (const char *dataLoc, std::vector< PetscReal > &timeSeries, PetscInt &ntimes)

  Returns the time list (from folder names) and size contained in a folder.
- PetscErrorCode getFileList (const char *dataLoc, std::vector< word > &fileSeries, PetscInt &nfiles)

  Returns the file list (from folder names) and number of files contained in a folder.
- PetscInt foundInString (const char *str, word keyword)

  Finds a word inside a string.
- PetscInt file_exist (const char *str)

  Checks if file exists.
- PetscInt dir_exist (const char *str)

  Checks if directory exists.
- PetscInt count_files (const char *path)

  Count number of files.
- void createDir (MPI_Comm comm, const char *path)

  Creat directory (remove stuff it exists)
- void createDirNoRemove (MPI_Comm comm, const char *path)

  Creat directory (leave stuff it exists)
- void remove_dir (MPI_Comm comm, const char *path2dir)

  Removes directory.
- void remove_subdirs (MPI_Comm comm, const char *path2dir)

  Removes all subdirectory of a given directory.
- void remove_subdirs_except (MPI_Comm comm, const char *path2dir, const word name)

  Removes all subdirectory of a given directory except the name provided.
- void remove_subdirs_except2 (MPI_Comm comm, const char *path2dir, const word name1, const word name2)

  Removes all subdirectory of a given directory except the 2 names provided.
- void remove_subdirs_except3 (MPI_Comm comm, const char *path2dir, const word name1, const word name2, const word name3)

  Removes all subdirectory of a given directory except the 3 names provided.
- void remove_subdirs_except4 (MPI_Comm comm, const char *path2dir, const word name1, const word name2, const word name3, const word name4)

  Removes all subdirectory of a given directory except the 3 names provided.
- void SetWriteDir (mesh_ *mesh, const word str)

  Set write directory for a given mesh.
- void writeBinaryField (mesh_ *mesh, Vec &V, char *file)

  Write binary vector file.
- PetscErrorCode InitializeIO (io_ *io)

  Initializes io data structure.
- PetscErrorCode RereadIO (domain_ *domain)

  Re-read IO write parameters.
- PetscErrorCode UpdateInput (io_ *io, word &modified)

  Called by RereadIO, triggers updates in each domain.
- PetscErrorCode readFields (domain_ *domain, PetscReal timeValue)

  read and save the different fields stored in the fields/meshName/time folder

- word getTimeName (clock_ *clock)

  Get time name in string format with required digits.
- word getStartTimeName (clock_ *clock)

  Get start time name in string format with required digits.
- word getArbitraryTimeName (clock_ *clock, double timeValue)

  Get time value in string format with required digits.
- PetscErrorCode setRunTimeWrite (domain_ *domain)

  Sets the runTimeWrite flag and creates initial output directory.
- PetscErrorCode writeFields (io_ *io)

  Write output fields.
- PetscErrorCode readDictDouble (const char *dictName, const char *keyword, PetscReal *value)

  Read mandatory PetscReal from dictionary.
- PetscErrorCode readDictVector (const char *dictName, const char *keyword, Cmpnts *value)

  Read mandatory vector from dictionary.
- PetscErrorCode readDictVector2D (const char *dictName, const char *keyword, Cmpnts *value)

  Read mandatory 2D vector from dictionary.
- PetscErrorCode readDictInt (const char *dictName, const char *keyword, PetscInt *value)

  Read mandatory PetscInt from dictionary.
- PetscErrorCode readDictWord (const char *dictName, const char *keyword, word *value)

  Read mandatory word from dictionary.
- PetscErrorCode readDictWordAndDouble (const char *dictName, const char *keyword, word *value1, PetscReal *value2)

  Read mandatory word and PetscReal from dictionary.
- PetscErrorCode readDictWordAndVector (const char *dictName, const char *keyword, word *value1, Cmpnts *value2)

  Read mandatory word and vector from dictionary.
- PetscErrorCode readSubDictDouble (const char *dictName, const char *subdict, const char *keyword, PetscReal *value)

  Read mandatory PetscReal from sub-dictionary.
- PetscErrorCode readSubDictInt (const char *dictName, const char *subdict, const char *keyword, PetscInt *value)

  Read mandatory PetscInt from sub-dictionary.
- PetscErrorCode readSubDictVector (const char *dictName, const char *subdict, const char *keyword, Cmpnts *value)

  Read mandatory vector from sub-dictionary.
- PetscErrorCode readSubDictWord (const char *dictName, const char *subdict, const char *keyword, word *value)

  Read mandatory word from sub-dictionary.
- PetscErrorCode readSubDictIntArray (const char *dictName, const char *subdict, const char *keyword, labelList &value)

  Read mandatory array of PetscInt from sub-dictionary.
- std::string * **readSubDictWordArray** (const char *dictName, const char *subdict, const char *keyword, PetscInt numW)
- word trim (const word &str)

  Trims a string removing pre and trail spaces.
- bool isNumber (const word &str)

  Check if is a number.

5.18.1 Detailed Description

Contains i/o operations function headers.

5.19 src/include/mesh.h File Reference

Mesh header file.

```
#include "wallmodel.h"
#include "boundary.h"
```

Classes

- struct boundingBox

  Structure defining the domain bounding box (easy access to xmin, xmax, ymin, ymax, zmin, zmax)
- struct mesh_

  mesh structure storing mesh and curvilinear coordinates

Functions

- PetscErrorCode InitializeMesh (mesh_ *mesh)

  Initialize mesh.
- PetscErrorCode SetDistributedArrays (mesh_ *mesh)

  Set distributed arrays.
- PetscErrorCode SetMeshMetrics (mesh_ *mesh)

  Set curvilinear coordinates metrics.
- PetscErrorCode SetBoundingBox (mesh_ *mesh)

  Set bounding box.
- PetscErrorCode ghostnodesCellcenter (mesh_ *mesh)

  Find the cell center for the ghost nodes.
- PetscErrorCode DeformMeshBasedOnBLDisp (mesh_ *mesh)

  Deform the mesh according to prescribed BL Disp.

5.19.1 Detailed Description

Mesh header file.

5.20 src/include/objects.h File Reference

Contains forward object declarations.

5.20.1 Detailed Description

Contains forward object declarations.

5.21 src/include/overset.h File Reference

overset Objects and functions,

Classes

- struct Acell

  overset acceptor cell
- struct Dcell

  overset donor cell
- struct oversetMotion

  struct with the overset motion
- struct overset_

  Structure for the Overset mesh method.

Functions

- PetscErrorCode InitializeOverset (domain_ *dm)

  initialize the overset variables
- PetscErrorCode UpdateOversetInterpolation (domain_ *domain)

  perform the overset interpolation
- PetscErrorCode readOversetProperties (overset_ *overset)

  read the overset properties file to set them
- PetscErrorCode updateAcceptorCoordinates (overset_ *os)

  In dynamic overset, update the acceptor cell co-ordinates for the next time step.
- PetscErrorCode oversetMeshTranslation (overset_ *os)

  overset mesh translation based on the prescribed velocity
- PetscErrorCode createAcceptorCell (overset_ *os)

  Create the list of acceptor cells which will be interpolated.
- PetscErrorCode findClosestDonor (mesh_ *meshP, mesh_ *mesh)

  Find the closest donor cell to the acceptor cell in the overset mesh for trilinear interpolation.
- PetscErrorCode acellDcellConnectivity (mesh_ *meshP, mesh_ *mesh)

  Find the donor cell to the acceptor cell in the overset mesh for Least square interpolation.
- PetscErrorCode getLSWeights (mesh_ *meshP, mesh_ *mesh)

  Find the weight for each donor cell when using least square interpolation first order.
- PetscErrorCode getLSWeights_2nd (mesh_ *meshP, mesh_ *mesh)

  Find the weight for each donor cell when using least square interpolation second order.
- PetscErrorCode getLSWeights_3rd (mesh_ *meshP, mesh_ *mesh)

  Find the weight for each donor cell when using least square interpolation third order.
- PetscErrorCode **OversetInterpolation** (domain_ *domain)
- PetscErrorCode **interpolateACellInvD** (mesh_ *meshP, mesh_ *mesh)
- PetscErrorCode **interpolateACellTrilinear** (mesh_ *meshP, mesh_ *mesh)
- PetscErrorCode **interpolateACellLS** (mesh_ *meshP, mesh_ *mesh)
- PetscErrorCode oversetContravariantBC (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, Cmpnts ucart, PetscInt face)

  set the contravariant flux at the face of an overset interpolated cell
- PetscErrorCode **updateAcellCoordinates** (domain_ *domain)
- PetscErrorCode **updateIntersectingProcessors** (domain_ *domain)
- PetscErrorCode **updateDonorCells** (domain_ *domain)
- PetscErrorCode **oversetMeshTranslation** (domain_ *domain)
- void **sum_struct_Acell** (void *in, void *inout, int *len, MPI_Datatype *type)
- void **defineStruct_Acell** (MPI_Datatype *tstype)

5.21.1 Detailed Description

overset Objects and functions,

5.22 src/include/peqn.h File Reference

P equation solution header file.

Classes

- struct peqn_
  struct storing pressure equation

Functions

- PetscErrorCode InitializePEqn (peqn_ *peqn)
  Initializes Peqn environment.
- PetscErrorCode CreateHypreSolver (peqn_ *peqn)
  Initializes HYPRE solver.
- PetscErrorCode CreatePETScSolver (peqn_ *peqn)
  Initializes PETSc solver.
- PetscErrorCode CreateHypreMatrix (peqn_ *peqn)
  Initializes HYPRE matrix.
- PetscErrorCode CreatePETScMatrix (peqn_ *peqn)
  Initializes PETSc matrix.
- PetscErrorCode CreateHypreVector (peqn_ *peqn)
  Initializes HYPRE vector.
- PetscErrorCode CreatePETScVector (peqn_ *peqn)
  Initializes PETSc vector.
- PetscErrorCode DestroyHypreSolver (peqn_ *peqn)
  Destroys HYPRE solver.
- PetscErrorCode DestroyPETScSolver (peqn_ *peqn)
  Destroys PETSc solver.
- PetscErrorCode DestroyHypreMatrix (peqn_ *peqn)
  Destroys HYPRE matrix.
- PetscErrorCode DestroyHypreVector (peqn_ *peqn)
  Destroys HYPRE vector.
- PetscErrorCode DestroyPETScVector (peqn_ *peqn)
  Destroys PETSc vector.
- PetscErrorCode Petsc2HypreVector (Vec &A, HYPRE_IJVector &B, HYPRE_Int startID)
  Transfer vector values from Petsc 2 Hypre.
- PetscErrorCode Hypre2PetscVector (HYPRE_IJVector &B, Vec &A, HYPRE_Int startID)
  Transfer vector values from Hypre 2 Petsc.
- PetscErrorCode phiToPhi (peqn_ *peqn)
  Convert phi to Phi (1D to 3D vector)
- PetscReal L2NormHypre (peqn_ *peqn, HYPRE_IJMatrix &A, HYPRE_IJVector &X, HYPRE_IJVector &B)
  Compute L2 norm of system residual.

- PetscReal L2NormPETSc (peqn_ *peqn, Mat &A, Vec &X, Vec &B)

  Compute L2 norm of system residual.
- PetscErrorCode GradP (peqn_ *peqn)

  Compute pressure gradient term.
- PetscErrorCode SetPoissonConnectivity (peqn_ *peqn)

  Compute cell-matrix connectivity.
- PetscErrorCode SetCoeffMatrix (peqn_ *peqn)

  Compute coefficient matrix.
- PetscErrorCode ZeroCoeffMatrix (peqn_ *peqn)

  Zeroes coefficient matrix.
- PetscErrorCode SetRHS (peqn_ *peqn)

  Set RHS of the pressure equation.
- PetscErrorCode SubtractAverageHypre (peqn_ *peqn, HYPRE_IJVector &B)

  Subtract average from the solution.
- PetscErrorCode SubtractAveragePETSc (peqn_ *peqn, Vec &B)

  Subtract average from the solution.
- PetscErrorCode AdjustIBMFlux (peqn_ *peqn)

  Correct IBM volume flux.
- PetscErrorCode UpdatePressure (peqn_ *peqn)

  Update pressure and subtract average.
- PetscErrorCode **updateIBMPhi** (ibm_ *ibm)
- PetscErrorCode ProjectVelocity (peqn_ *peqn)

  Project Ucont into an incompressible space.
- PetscErrorCode SolvePEqn (peqn_ *peqn)

  Compute pressure gradient term.
- PetscErrorCode SetPressureReference (peqn_ *peqn)

  Set pressure = 0 at k = 0, j = 0, i = 0.
- PetscErrorCode ContinuityErrors (peqn_ *peqn)

  Compute continuity errors (also calculates which cell and processor has the max)
- PetscErrorCode ContinuityErrorsOptimized (peqn_ *peqn)

  Compute continuity errors (only prints the max)
- PetscErrorCode InitGravityWaveInducedPressure (peqn_ *peqn)

  Initialize pressure with given gravity wave large-scale pressure field from 3LM.
- cellIds GetIdFromStencil (int stencil, int k, int j, int i)

  get the cell id from stencil position

5.22.1 Detailed Description

P equation solution header file.

5.23 src/include/precursor.h File Reference

Concurrent precursor header file.

Classes

- struct mapInfo

  concurrent precursor mapping info
- struct precursor_

  concurrent precursor database

Functions

- PetscErrorCode concurrentPrecursorInitialize (abl_ *abl)

  initialize concurrent precursor
- PetscErrorCode SetSolutionFlagsPrecursor (domain_ *domain)

  Precursor solution flags definition.
- PetscErrorCode SetStartTimePrecursor (domain_ *domain, abl_ *abl)

  Checks that start time is available to read if spinUp == 0.
- PetscErrorCode InitializeMeshPrecursor (abl_ *abl)

  Precursor mesh initialization.
- PetscErrorCode SetBoundaryConditionsPrecursor (mesh_ *mesh)

  Precursor boundary conditions initialization function.
- PetscErrorCode SetInflowFunctionsPrecursor (mesh_ *mesh)

  Sets the inflow function on the precursor.
- PetscErrorCode MapInitialConditionPrecursor (abl_ *abl)

  Map fields from successor to precursor.
- PetscErrorCode successorPrecursorMapVectorField (abl_ *abl, Vec &Source, Vec &Target, Vec &lTarget)

  Map vector field from successor to precursor.
- PetscErrorCode successorPrecursorMapScalarField (abl_ *abl, Vec &Source, Vec &Target, Vec &lTarget)

  Map scalar tor field from successor to precursor.
- PetscErrorCode ABLInitializePrecursor (domain_ *domain)

  Initialize abl for precursor (x damping layer is not set)
- PetscErrorCode concurrentPrecursorSolve (abl_ *abl)

  Solve concurrent precursor.

5.23.1 Detailed Description

Concurrent precursor header file.

5.24 src/include/teqn.h File Reference

T equation solution header file.

Classes

- struct teqn_

  struct storing temperature equation

Functions

- PetscErrorCode InitializeTEqn (teqn_ *teqn)
  Initializes Teqn environment.
- PetscErrorCode SolveTEqn (teqn_ *teqn)
  Solve T equation.
- PetscErrorCode TeqnSNES (SNES snes, Vec T, Vec Rhs, void *ptr)
  SNES evaulation function.
- PetscErrorCode ghGradRhoK (teqn_ *teqn)
  *Computes g*h times gradient of rho_k / rho_0.*
- PetscErrorCode CorrectSourceTermsT (teqn_ *teqn, PetscInt print)
  Compute temperature control source term.
- PetscErrorCode dampingSourceT (teqn_ *teqn, Vec &Rhs, PetscReal scale)
  Apply fringe region damping.
- PetscErrorCode sourceT (teqn_ *teqn, Vec &Rhs, PetscReal scale)
  Apply temperature control.
- PetscErrorCode FormT (teqn_ *teqn, Vec &Rhs, PetscReal scale)
  RHS of the potential temperature transport equation.
- PetscErrorCode TeqnRK4 (teqn_ *teqn)
  solve Teqn using RungeKutta 4
- PetscErrorCode FormExplicitRhsT (teqn_ *teqn)
  Computed RHS of temperature equation using current ITmprt (updates Rhs), data put in ueqn->Rhs.
- PetscErrorCode correctDampingSourcesT (teqn_ *teqn)
  Compute tBar state for lateral damping region.

5.24.1 Detailed Description

T equation solution header file.

5.25 src/include/tosca2PV.h File Reference

post processing header file.

Classes

- struct postProcess
  Structure defining the variables for postProcessing.

Functions

- PetscErrorCode binary3DToXMF (domain_ *domain, postProcess *pp)

  Reads binary fields and writes paraview data into XMF folder.
- PetscErrorCode binaryISectionsToXMF (domain_ *domain)

  Reads binary i-section data and writes paraview data into XMF folder.
- PetscErrorCode fieldISectionsToXMF (domain_ *domain)

  Reads i-section data from average fields and writes paraview data into XMF folder.
- PetscErrorCode binaryISectionsPerturbToXMF (domain_ *domain)

  Reads binary i-section pertirbation data and writes paraview data into XMF folder.
- PetscErrorCode binaryJSectionsToXMF (domain_ *domain, postProcess *pp)

  Reads binary j-section data and writes paraview data into XMF folder.
- PetscErrorCode fieldJSectionsToXMF (domain_ *domain)

  Reads j-section data from average fields and writes paraview data into XMF folder.
- PetscErrorCode binaryJSectionsPerturbToXMF (domain_ *domain, postProcess *pp)

  Reads binary j-section perturbation data and writes paraview data into XMF folder.
- PetscErrorCode binaryKSectionsToXMF (domain_ *domain)

  Reads binary k-section data and writes paraview data into XMF folder.
- PetscErrorCode fieldKSectionsToXMF (domain_ *domain)

  Reads k-section data from average fields and writes paraview data into XMF folder.
- PetscErrorCode fieldUserDefinedPlaneToXMF (domain_ *domain)

  Reads a user defined surface file and writes paraview average field data at the surface points into the XMF folder.
- PetscErrorCode binaryKSectionsPerturbToXMF (domain_ *domain)

  Reads binary k-section perturbation data and writes paraview data into XMF folder.
- PetscErrorCode postProcessInitialize (domain_ **domainAddr, clock_ *clock, simInfo_ *info, flags_ *flags)

  Initialize post processing parameters.
- PetscErrorCode postProcessInitializePrecursor (postProcess *pp, clock_ *clock)

  Initialize concurrent precursor post processing parameters.
- PetscErrorCode **writeFieldsToXMF** (domain_ *domain, const char *filexmf, PetscReal time)
- PetscErrorCode postProcessWriteProbes (domain_ *domain)

  \broer On the fly probes cration of average and phase average fields only
- PetscErrorCode getTimeList (const char *dataLoc, std::vector< PetscReal > &timeSeries, PetscInt &ntimes)

  gets list of time folders contained in a directory
- PetscErrorCode sectionsReadAndAllocate (domain_ *domain)

  Reads i,j,k - sections info and allocates moemry. Some info are not necessary thus not read.
- PetscErrorCode kSectionLoadSymmTensorFromField (Vec &V, mesh_ *mesh, sections *sec, PetscInt kplane, const word &fieldName, PetscReal time)

  Generate the section on-the-fly in the post processing phase (for average fields that only have to be done at the end)
- PetscErrorCode **kSectionLoadVectorFromField** (Vec &V, mesh_ *mesh, sections *sec, PetscInt kplane, const word &fieldName, PetscReal time)
- PetscErrorCode **kSectionLoadScalarFromField** (Vec &V, mesh_ *mesh, sections *sec, PetscInt kplane, const word &fieldName, PetscReal time)
- PetscErrorCode **userSectionLoadVectorFromField** (Vec &V, mesh_ *mesh, uSections *uSection, const word &fieldName, PetscReal time)
- PetscErrorCode **userSectionLoadScalarFromField** (Vec &V, mesh_ *mesh, uSections *uSection, const word &fieldName, PetscReal time)
- PetscErrorCode kSectionLoadVector (mesh_ *mesh, sections *sec, PetscInt kplane, const word &fieldName, PetscReal time)

  \briefReads from k-slices time series and loads the velocity, temperature and nut planes. Important: assumes T and nut databases have the same times of U.
- PetscErrorCode **kSectionLoadScalar** (mesh_ *mesh, sections *sec, PetscInt kplane, const word &field↩ Name, PetscReal time)

- PetscErrorCode iSectionLoadSymmTensorFromField (Vec &V, mesh_ *mesh, sections *sec, PetscInt kplane, const word &fieldName, PetscReal time)

  Generate the section on-the-fly in the post processing phase (for average fields that only have to be done at the end)
- PetscErrorCode **iSectionLoadVectorFromField** (Vec &V, mesh_ *mesh, sections *sec, PetscInt iplane, const word &fieldName, PetscReal time)
- PetscErrorCode **iSectionLoadScalarFromField** (Vec &V, mesh_ *mesh, sections *sec, PetscInt iplane, const word &fieldName, PetscReal time)
- PetscErrorCode iSectionLoadVector (mesh_ *mesh, sections *sec, PetscInt iplane, const word &fieldName, PetscReal time)

  \briefReads from i-slices time series and loads the velocity, temperature and nut planes. Important: assumes T and nut databases have the same times of U.
- PetscErrorCode **iSectionLoadScalar** (mesh_ *mesh, sections *sec, PetscInt iplane, const word &fieldName, PetscReal time)
- PetscErrorCode jSectionLoadSymmTensorFromField (Vec &V, mesh_ *mesh, sections *sec, PetscInt kplane, const word &fieldName, PetscReal time)

  Generate the section on-the-fly in the post processing phase (for average fields that only have to be done at the end)
- PetscErrorCode **jSectionLoadVectorFromField** (Vec &V, mesh_ *mesh, sections *sec, PetscInt jplane, const word &fieldName, PetscReal time)
- PetscErrorCode **jSectionLoadScalarFromField** (Vec &V, mesh_ *mesh, sections *sec, PetscInt jplane, const word &fieldName, PetscReal time)
- PetscErrorCode jSectionLoadVector (mesh_ *mesh, sections *sec, PetscInt jplane, const word &fieldName, PetscReal time)

  \briefReads from j-slices time series and loads the velocity, temperature and nut planes. Important: assumes T and nut databases have the same times of U.
- PetscErrorCode **jSectionLoadScalar** (mesh_ *mesh, sections *sec, PetscInt jplane, const word &fieldName, PetscReal time)
- void xmfWriteFileStartTimeSection (FILE *Xmf, const char *FileXmf, PetscInt Size_x, PetscInt Size_y, PetscInt Size_z, const char *Topology, PetscReal Time)

  Opens a time section in the XMF file.
- void xmfWriteFileEndTimeSection (FILE *Xmf, const char *FileXmf)

  Closes a time section in the XMF file.
- void xmfWriteFileGeometry (FILE *Xmf, const char *FileXmf, PetscInt Size_x, PetscInt Size_y, PetscInt Size_z, const char *PathSave)

  Writes geometry info in the XMF file.
- void xmfWriteFileSymmTensor (FILE *xmf, const char *filexmf, PetscInt size_x, PetscInt size_y, PetscInt size_z, const char *PathSave, const char *symmTensorName, const char *XX, const char *YY, const char *ZZ, const char *XY, const char *XZ, const char *YZ, const char *center="Cell")

  Writes a symmetric tensor in the XMF files.
- void xmfWriteFileVector (FILE *xmf, const char *filexmf, PetscInt size_x, PetscInt size_y, PetscInt size_z, const char *PathSave, const char *Vecname, const char *V1, const char *V2, const char *V3, const char *center="Cell")

  Writes a vector in the XMF file.
- void xmfWriteFileScalar (FILE *Xmf, const char *Filexmf, PetscInt Size_x, PetscInt Size_y, PetscInt Size_z, const char *PathSave, const char *ScalName, const char *Scal, const char *Center="Cell")

  Writes a scalar in the XMF file.
- void hdfWriteDataset (hid_t *file_id, hid_t *dataspace_id, char const *var, float *x)

  Writes a dataset to HDF format file.
- PetscErrorCode writeScalarToXMF (domain_ *domain, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, Vec V)

  Writes a scalar field (appends to XMF and creates HDF)
- PetscErrorCode writeVectorToXMF (domain_ *domain, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, Vec V)

  Writes a vector field (appends to XMF and creates HDF)

- PetscErrorCode writeSymmTensorToXMF (domain_ *domain, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, Vec V)

  Writes a symmetric tensor field (appends to XMF and creates HDF)
- PetscErrorCode writeISectionScalarToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, PetscReal **field)

  Writes a scalar defined on an i-section (appends to XMF and creates HDF)
- PetscErrorCode writeJSectionScalarToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, PetscReal **field)

  Writes a scalar defined on an j-section (appends to XMF and creates HDF)
- PetscErrorCode writeKSectionScalarToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, PetscReal **field)

  Writes a scalar defined on an k-section (appends to XMF and creates HDF)
- PetscErrorCode writeUserSectionScalarToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, uSections *uSection)

  Writes a scalar defined on a user defined section (appends to XMF and creates HDF)
- PetscErrorCode writeISectionVectorToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, Cmpnts **field)

  Writes a vector defined on an i-section (appends to XMF and creates HDF)
- PetscErrorCode writeJSectionVectorToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, Cmpnts **field)

  Writes a vector defined on an j-section (appends to XMF and creates HDF)
- PetscErrorCode writeKSectionVectorToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, Cmpnts **field)

  Writes a vector defined on an k-section (appends to XMF and creates HDF)
- PetscErrorCode writeUserSectionVectorToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, uSections *uSection)

  Writes a vector defined on a user defined section (appends to XMF and creates HDF)
- PetscErrorCode writeISectionSymmTensorToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, symmTensor **field)

  Writes a vector defined on an i-section (appends to XMF and creates HDF)
- PetscErrorCode writeJSectionSymmTensorToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, symmTensor **field)

  Writes a vector defined on an j-section (appends to XMF and creates HDF)
- PetscErrorCode writeKSectionSymmTensorToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, const char *fieldName, symmTensor **field)

  Writes a vector defined on an k-section (appends to XMF and creates HDF)
- PetscErrorCode writePointsToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time)

  Writes 3D mesh (appends to XMF and creates HDF)
- PetscErrorCode writeISectionPointsToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, PetscInt iIndex)

  Writes 2D i-section mesh (appends to XMF and creates HDF)
- PetscErrorCode writeJSectionPointsToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, PetscInt jIndex)

  Writes 2D j-section mesh (appends to XMF and creates HDF)
- PetscErrorCode writeKSectionPointsToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, PetscInt kIndex)

  Writes 2D k-section mesh (appends to XMF and creates HDF)
- PetscErrorCode writeUserSectionPointsToXMF (mesh_ *mesh, const char *filexmf, const char *hdfilen, hid_t *file_id, hid_t *dataspace_id, PetscReal time, uSections *uSection)

  Writes 2D user-section mesh (appends to XMF and creates HDF)
- void **setToZero** (float *vec, PetscInt n)
- PetscErrorCode **writeJSectionToRaster** (mesh_ *mesh, PetscInt jIndex)

5.25.1 Detailed Description

post processing header file.

5.25.2 Function Documentation

5.25.2.1 getTimeList()

```
PetscErrorCode getTimeList (
            const char * dataLoc,
            std::vector< PetscReal > & timeSeries,
            PetscInt & ntimes )
```

gets list of time folders contained in a directory

gets list of time folders contained in a directory

5.26 src/include/ueqn.h File Reference

U equation solution header file.

Classes

- struct ueqn_
  structure storing momentum equation

Functions

- PetscErrorCode InitializeUEqn (ueqn_ *ueqn)
  Initializes Ueqn environment.
- PetscErrorCode UpdateFluxLimiter (ueqn_ *ueqn)
  Updates flux limiter.
- PetscErrorCode CorrectSourceTerms (ueqn_ *ueqn, PetscInt print)
  Update driving source terms.
- PetscErrorCode correctDampingSources (ueqn_ *ueqn)
  Correct damping source terms.
- PetscErrorCode mapYDamping (ueqn_ *ueqn)
  finish the mapping of the ydamping source
- PetscErrorCode contravariantToCartesian (ueqn_ *ueqn)
  Transform velocity from contravariant to cartesian.
- PetscErrorCode contravariantToCartesianGeneric (mesh_ *mesh, Vec &lCont, Vec &lCat)
  Transform generic local vector from contravariant to cartesian.
- PetscErrorCode adjustFluxes (ueqn_ *ueqn)
  Adjust fluxes to obey mass conservation.

- PetscErrorCode adjustFluxesOverset (ueqn_ *ueqn)

  Adjust fluxes to obey mass conservation in the overset domain.
- PetscErrorCode sourceU (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Compute driving source term.
- PetscErrorCode dampingSourceU (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Compute damping source terms (x and z)
- PetscErrorCode Coriolis (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Compute Coriolis source term.
- PetscErrorCode CanopyForce (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Compute Side Force source term.
- PetscErrorCode Buoyancy (ueqn_ *ueqn, PetscReal scale)

  Compute buoyancy term.
- PetscErrorCode FormU (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Viscous and divergence terms.
- PetscErrorCode SolveUEqn (ueqn_ *ueqn)

  Solve the momentum equation.
- PetscErrorCode UeqnSNES (SNES snes, Vec Ucont, Vec Rhs, void *ptr)

  SNES evaluation function.
- PetscErrorCode UeqnRK4 (ueqn_ *ueqn)

  Solves ueqn using 4 stages runge kutta.
- PetscErrorCode UeqnEuler (ueqn_ *ueqn)

  Solves ueqn using explicit euler.
- PetscErrorCode FormExplicitRhsU (ueqn_ *ueqn)

  Computed RHS of momentum equation using current lUcont (updates Rhs), data put in ueqn->Rhs.

5.26.1 Detailed Description

U equation solution header file.

5.27 src/include/wallfunctions.h File Reference

wallfunction functions

```
#include "io.h"
```

Functions

- PetscReal **uTauCabot** (PetscReal nu, PetscReal u, PetscReal y, PetscReal guess, PetscReal dpdn)
- PetscReal **utau_wf** (PetscReal nu, PetscReal ks, PetscReal sb, PetscReal Ut_mag)
- PetscReal **uTauCabotRoughness** (PetscReal nu, PetscReal u, PetscReal y, PetscReal guess, PetscReal dpdn, PetscReal ks)
- void **wallFunctionCabot** (PetscReal nu, PetscReal sc, PetscReal sb, Cmpnts Ua, Cmpnts Uc, Cmpnts *Ub, PetscReal *ustar, Cmpnts nf)
- void **wallFunctionCabotRoughness** (PetscReal nu, PetscReal ks, PetscReal sc, PetscReal sb, Cmpnts Ua, Cmpnts Uc, Cmpnts *Ub, PetscReal *ustar, Cmpnts nf)
- void **wallFunctionPowerlaw** (PetscReal nu, PetscReal sc, PetscReal sb, Cmpnts Ua, Cmpnts Uc, Cmpnts *Ub, PetscReal *ustar, Cmpnts nf)

- void **wallFunctionPowerlawAPG** (PetscReal nu, PetscReal sc, PetscReal sb, PetscReal roughness, PetscReal kappa, Cmpnts Ua, Cmpnts Uc, Cmpnts *Ub, PetscReal *ustar, Cmpnts nf, PetscReal dpdx, PetscReal dpdy, PetscReal dpdz)
- void **wallFunctionLogLawAPG** (PetscReal nu, PetscReal sc, PetscReal sb, PetscReal roughness, PetscReal kappa, Cmpnts Ua, Cmpnts Uc, Cmpnts *Ub, PetscReal *ustar, Cmpnts nf, PetscReal dpdx, PetscReal dpdy, PetscReal dpdz)
- void **slipBC** (PetscReal sc, PetscReal sb, Cmpnts Ua, Cmpnts Uc, Cmpnts *Ub, Cmpnts nf)
- void **wallFunctionSchumann** (PetscReal nu, PetscReal sc, PetscReal sb, PetscReal roughness, PetscReal kappa, Cmpnts Ua, Cmpnts Uc, Cmpnts *Ub, PetscReal *ustar, Cmpnts nf)
- void **wallShearVelocityBC** (PetscReal nu, PetscReal sc, PetscReal sb, PetscReal roughness, PetscReal kappa, Cmpnts Ua, Cmpnts Uc, Cmpnts *Ub, PetscReal *ustar, Cmpnts nf)
- void **wallShearVelocityBCQuadratic** (PetscReal nu, PetscReal sd, PetscReal sc, PetscReal sb, PetscReal roughness, PetscReal kappa, Cmpnts Ua, Cmpnts Ud, Cmpnts Uc, Cmpnts *Ub, PetscReal *ustar, Cmpnts nf)
- void **uStarShumann** (PetscReal &UParallelMeanMag, PetscReal &wallDist, PetscReal &roughness, PetscReal &gammaM, PetscReal &kappa, PetscReal &qwall, PetscReal &thetaRef, PetscReal &uStar, PetscReal &phiM, PetscReal &L)
- void **qWallShumann** (PetscReal &UParallelMeanMag, PetscReal &wallDist, PetscReal &z0, PetscReal &gammaM, PetscReal &gammaH, PetscReal &alphaH, PetscReal &thetaRef, PetscReal &deltaTheta, PetscReal &kappa, PetscReal &qWall, PetscReal &uStar, PetscReal &phiM, PetscReal &phiH, PetscReal &L)

5.27.1 Detailed Description

wallfunction functions

5.28 src/include/wallmodel.h File Reference

Wall models header file.

Classes

- struct Shumann

  structure storing the Shumann wall models information for U and T
- struct Cabot

  structure storing the Shumann wall models information
- struct PowerLawAPG

  structure storing the Shumann wall models information
- struct LogLawAPG
- struct wallModel

  wall models container

5.28.1 Detailed Description

Wall models header file.

5.29 src/initialField.c File Reference

Contains initial field function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/initialField.h"
```

Functions

- PetscErrorCode SetInitialField (domain_ *domain)
  <
- PetscErrorCode SetInitialFieldPrecursor (abl_ *abl)
  set the initial internal contravariant and cartesian velocity field
- PetscErrorCode SetInitialFieldU (ueqn_ *ueqn)
  set the initial temperature field
- PetscErrorCode SetInitialFieldT (teqn_ *teqn)
  set the initial pressure field
- PetscErrorCode SetInitialFieldP (peqn_ *peqn)
  set the initial variables for LES
- PetscErrorCode SetInitialFieldLES (les_ *les)
  set uniform value for the cartesian velocity (add perturbations if applicable)
- PetscErrorCode SetUniformFieldU (ueqn_ *ueqn, Cmpnts &uRef, PetscInt &addPerturbations)
  set initial ABL flow U
- PetscErrorCode SetABLInitialFlowU (ueqn_ *ueqn)
  set the internal field as the spreaded inlet flow condition
- PetscErrorCode SpreadInletFlowU (ueqn_ *ueqn)
  set uniform value for the temperature
- PetscErrorCode SetUniformFieldT (teqn_ *teqn, PetscReal &tRef)
  set linear profile for the temperature
- PetscErrorCode SetLinearFieldT (teqn_ *teqn, PetscReal &tRef, PetscReal &tLapse)
  set initial ABL flow T
- PetscErrorCode SetABLInitialFlowT (teqn_ *teqn)
  set the internal field as the spreaded inlet flow condition
- PetscErrorCode **SpreadInletFlowT** (teqn_ *teqn)

5.29.1 Detailed Description

Contains initial field function definitions.

5.29.2 Function Documentation

5.29.2.1 SetInitialField()

```
PetscErrorCode SetInitialField (
            domain_ * domain )
```

<

set the initial internal contravariant and cartesian velocity field

set the initial internal fields

set the initial internal fields for the concurrent precursor simulation

5.30 src/initialization.c File Reference

Contains inflow boundary condition function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/boundary.h"
#include "include/inflow.h"
#include "include/abl.h"
#include "include/turbines.h"
#include "include/initialization.h"
```

Functions

- PetscErrorCode PrintOkWindLogo ()
  Print OkWind logo.
- PetscErrorCode PrintNumberOfProcs ()
  Print number of processors.
- PetscErrorCode simulationInitialize (domain_ **domainAddr, clock_ *clock, simInfo_ *info, flags_ *flags)
  Initialize the simulation parameters.
- PetscErrorCode SetSimulationFlags (flags_ *flags)
  Set simulation flags.
- PetscErrorCode SetSimulationInfo (simInfo_ *info)
  Set simulation global info.
- PetscErrorCode SetDomainsAndAllocate (domain_ **domainAddr, flags_ *flags, simInfo_ *info)
  Set number of domains (reads from OversetInput.dat file if overset is active)
- PetscErrorCode ReadTimeControls (clock_ *clock)
  Read time controls.
- PetscErrorCode SetStartTime (clock_ *clock, domain_ *domain, simInfo_ *info)
  Get startFrom parameter and set initial time (check multiple domains cosistency)
- PetscErrorCode ReadPhysicalConstants (domain_ *domain)
  Read physical constants.
- PetscErrorCode SetDomainMemory (domain_ *domain)
  Allocate memory for domain pointers.
- PetscErrorCode SetAccessPointers (domain_ *domain)
  Set access database pointers.

5.30.1 Detailed Description

Contains inflow boundary condition function definitions.

Contains simulation initialization function definitions.

5.30.2 Function Documentation

5.30.2.1 simulationInitialize()

```
PetscErrorCode simulationInitialize (
            domain_ ** domainAddr,
            clock_ * clock,
            simInfo_ * info,
            flags_ * flags )
```

Initialize the simulation parameters.

set the initial internal fields

5.31 src/io.c File Reference

Contains i/o operations function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
```

Functions

- PetscErrorCode InitializeIO (io_ *io)
  Initializes io data structure.
- PetscErrorCode ReadIO (domain_ *domain)
  Re-read IO write parameters.
- PetscErrorCode UpdateInput (io_ *io, word &modified)
  Called by ReadIO, triggers updates in each domain.
- PetscErrorCode readFields (domain_ *domain, PetscReal timeValue)
  read and save the different fields stored in the fields/meshName/time folder
- void **writeBinaryField** (MPI_Comm comm, Vec V, const char *file)
- PetscErrorCode **VecScalarLocalToGlobalCopy** (mesh_ *mesh, Vec &lV, Vec &V)
- PetscErrorCode **VecVectorLocalToGlobalCopy** (mesh_ *mesh, Vec &lV, Vec &V)
- PetscErrorCode **VecSymmTensorLocalToGlobalCopy** (mesh_ *mesh, Vec &lV, Vec &V)
- PetscErrorCode writeFields (io_ *io)
  Write output fields.

- void fatalErrorInFunction (const char *functionName, const char *errorMsg)
  Calls fatal error and exits.
- void warningInFunction (const char *functionName, const char *wrngMsg)
  Calls warning.
- void createDir (MPI_Comm comm, const char *path)
  Creat directory (remove stuff it exists)
- void createDirNoRemove (MPI_Comm comm, const char *path)
  Creat directory (leave stuff it exists)
- word thisCaseName ()
  Retrieves this case name.
- PetscErrorCode getFileList (const char *dataLoc, std::vector< word > &fileSeries, PetscInt &nfiles)
  Returns the file list (from folder names) and number of files contained in a folder.
- PetscErrorCode getTimeList (const char *dataLoc, std::vector< PetscReal > &timeSeries, PetscInt &ntimes)
  Returns the time list (from folder names) and size contained in a folder.
- PetscInt foundInString (const char *str, word keyword)
  Finds a word inside a string.
- PetscInt file_exist (const char *str)
  Checks if file exists.
- PetscInt dir_exist (const char *str)
  Checks if directory exists.
- PetscInt count_files (const char *path)
  Count number of files.
- void remove_dir (MPI_Comm comm, const char *path2dir)
  Removes directory.
- void remove_subdirs (MPI_Comm comm, const char *path2dir)
  Removes all subdirectory of a given directory.
- void remove_subdirs_except (MPI_Comm comm, const char *path2dir, const word name)
  Removes all subdirectory of a given directory except the name provided.
- void remove_subdirs_except2 (MPI_Comm comm, const char *path2dir, const word name1, const word name2)
  Removes all subdirectory of a given directory except the 2 names provided.
- void remove_subdirs_except3 (MPI_Comm comm, const char *path2dir, const word name1, const word name2, const word name3)
  Removes all subdirectory of a given directory except the 3 names provided.
- void remove_subdirs_except4 (MPI_Comm comm, const char *path2dir, const word name1, const word name2, const word name3, const word name4)
  Removes all subdirectory of a given directory except the 3 names provided.
- word getTimeName (clock_ *clock)
  Get time name in string format with required digits.
- word getArbitraryTimeName (clock_ *clock, double timeValue)
  Get time value in string format with required digits.
- word getStartTimeName (clock_ *clock)
  Get start time name in string format with required digits.
- PetscErrorCode setRunTimeWrite (domain_ *domain)
  Sets the runTimeWrite flag and creates initial output directory.
- PetscErrorCode readDictDouble (const char *dictName, const char *keyword, PetscReal *value)
  Read mandatory PetscReal from dictionary.
- PetscErrorCode readDictVector2D (const char *dictName, const char *keyword, Cmpnts *value)
  Read mandatory 2D vector from dictionary.
- PetscErrorCode readDictVector (const char *dictName, const char *keyword, Cmpnts *value)
  Read mandatory vector from dictionary.

- PetscErrorCode readDictInt (const char *dictName, const char *keyword, PetscInt *value)

  Read mandatory PetscInt from dictionary.
- PetscErrorCode readDictWord (const char *dictName, const char *keyword, word *value)

  Read mandatory word from dictionary.
- PetscErrorCode readDictWordAndDouble (const char *dictName, const char *keyword, word *value1, PetscReal *value2)

  Read mandatory word and PetscReal from dictionary.
- PetscErrorCode readDictWordAndVector (const char *dictName, const char *keyword, word *value1, Cmpnts *value2)

  Read mandatory word and vector from dictionary.
- PetscErrorCode readSubDictDouble (const char *dictName, const char *subdict, const char *keyword, PetscReal *value)

  Read mandatory PetscReal from sub-dictionary.
- PetscErrorCode readSubDictInt (const char *dictName, const char *subdict, const char *keyword, PetscInt *value)

  Read mandatory PetscInt from sub-dictionary.
- PetscErrorCode readSubDictWord (const char *dictName, const char *subdict, const char *keyword, word *value)

  Read mandatory word from sub-dictionary.
- PetscErrorCode readSubDictVector (const char *dictName, const char *subdict, const char *keyword, Cmpnts *value)

  Read mandatory vector from sub-dictionary.
- PetscErrorCode readSubDictIntArray (const char *dictName, const char *subdict, const char *keyword, labelList &value)

  Read mandatory array of PetscInt from sub-dictionary.
- std::string * **readSubDictWordArray** (const char *dictName, const char *subdict, const char *keyword, PetscInt numW)
- word trim (const word &str)

  Trims a string removing pre and trail spaces.
- bool isNumber (const word &str)

  Check if is a number.

5.31.1 Detailed Description

Contains i/o operations function definitions.

5.31.2 Function Documentation

5.31.2.1 getTimeList()

```
PetscErrorCode getTimeList (
            const char * dataLoc,
            std::vector< PetscReal > & timeSeries,
            PetscInt & ntimes )
```

Returns the time list (from folder names) and size contained in a folder.

gets list of time folders contained in a directory

5.32 src/les.c File Reference

Contains LES model function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
```

Functions

- PetscErrorCode InitializeLES (les_ *les)
  Initializes LES environment.
- PetscErrorCode UpdateCs (les_ *les)
  Update Cs Smagorinky coefficient.
- PetscErrorCode UpdateNut (les_ *les)
  Update effective viscosity.

Variables

- const PetscReal **wall_cs** = 0.001
- const PetscReal **std_cs** = 0.0289

5.32.1 Detailed Description

Contains LES model function definitions.

5.32.2 Function Documentation

5.32.2.1 UpdateCs()

```
PetscErrorCode UpdateCs (
            les_ * les )
```

Update Cs Smagorinky coefficient.

isOnCornerCellCenters(I, J, K, mesh->info) <- use also ghost? Not for now

isOnCornerCellCenters(I, J, K, mesh->info) <- use also ghost? Not for now

5.33 src/mesh.c File Reference

Contains mesh definition functions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
```

Functions

- PetscErrorCode InitializeMesh (mesh_ *mesh)

  Initialize mesh.
- PetscErrorCode SetDistributedArrays (mesh_ *mesh)

  Set distributed arrays.
- PetscErrorCode DeformMeshBasedOnBLDisp (mesh_ *mesh)

  Deform the mesh according to prescribed BL Disp.
- PetscErrorCode SetMeshMetrics (mesh_ *mesh)

  Set curvilinear coordinates metrics.
- PetscErrorCode SetBoundingBox (mesh_ *mesh)

  Set bounding box.
- PetscErrorCode ghostnodesCellcenter (mesh_ *mesh)

  Find the cell center for the ghost nodes.

5.33.1 Detailed Description

Contains mesh definition functions.

5.34 src/overset.c File Reference

overset function definitions

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/initialField.h"
#include "include/overset.h"
```

Functions

- PetscErrorCode InitializeOverset (domain_ *domain)

  initialize the overset variables
- PetscErrorCode UpdateOversetInterpolation (domain_ *domain)

  perform the overset interpolation
- PetscErrorCode readOversetProperties (overset_ *os)

  read the overset properties file to set them
- PetscErrorCode **interpolateACellTrilinear** (mesh_ *meshP, mesh_ *mesh)
- PetscErrorCode **interpolateACellLS** (mesh_ *meshP, mesh_ *mesh)
- PetscErrorCode **interpolateACellInvD** (mesh_ *meshP, mesh_ *mesh)
- PetscErrorCode createAcceptorCell (overset_ *os)

  Create the list of acceptor cells which will be interpolated.
- PetscErrorCode acellDcellConnectivity (mesh_ *meshP, mesh_ *mesh)

  Find the donor cell to the acceptor cell in the overset mesh for Least square interpolation.
- PetscErrorCode findClosestDonor (mesh_ *meshP, mesh_ *mesh)

  Find the closest donor cell to the acceptor cell in the overset mesh for trilinear interpolation.
- PetscErrorCode getLSWeights (mesh_ *meshP, mesh_ *mesh)

  Find the weight for each donor cell when using least square interpolation first order.
- PetscErrorCode getLSWeights_2nd (mesh_ *meshP, mesh_ *mesh)

  Find the weight for each donor cell when using least square interpolation second order.
- PetscErrorCode getLSWeights_3rd (mesh_ *meshP, mesh_ *mesh)

  Find the weight for each donor cell when using least square interpolation third order.
- PetscErrorCode oversetContravariantBC (mesh_ *mesh, PetscInt i, PetscInt j, PetscInt k, Cmpnts ucart, PetscInt face)

  set the contravariant flux at the face of an overset interpolated cell
- PetscErrorCode updateAcceptorCoordinates (overset_ *os)

  In dynamic overset, update the acceptor cell co-ordinates for the next time step.
- PetscErrorCode oversetMeshTranslation (overset_ *os)

  overset mesh translation based on the prescribed velocity
- void **defineStruct_ACell** (MPI_Datatype *tstype)
- void **sum_struct_ACell** (void *in, void *inout, int *len, MPI_Datatype *type)

5.34.1 Detailed Description

overset function definitions

5.35 src/peqn.c File Reference

Contains P equation function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
```

Macros

- #define **matID**(i, j, k) (HYPRE_Int)(gid[k][j][i])
- #define **CP** 0
- #define **EP** 1
- #define **WP** 2
- #define **NP** 3
- #define **SP** 4
- #define **TP** 5
- #define **BP** 6
- #define **NE** 7
- #define **SE** 8
- #define **NW** 9
- #define **SW** 10
- #define **TN** 11
- #define **BN** 12
- #define **TS** 13
- #define **BS** 14
- #define **TE** 15
- #define **BE** 16
- #define **TW** 17
- #define **BW** 18

Functions

- PetscErrorCode InitializePEqn (peqn_ ∗peqn)

  Initializes Peqn environment.
- PetscErrorCode SetPoissonConnectivity (peqn_ ∗peqn)

  Compute cell-matrix connectivity.
- PetscErrorCode CreateHypreMatrix (peqn_ ∗peqn)

  Initializes HYPRE matrix.
- PetscErrorCode CreatePETScMatrix (peqn_ ∗peqn)

  Initializes PETSc matrix.
- PetscErrorCode CreateHypreVector (peqn_ ∗peqn)

  Initializes HYPRE vector.
- PetscErrorCode CreatePETScVector (peqn_ ∗peqn)

  Initializes PETSc vector.
- PetscErrorCode CreateHypreSolver (peqn_ ∗peqn)

  Initializes HYPRE solver.
- PetscErrorCode **MyKSPMonitorPoisson** (KSP ksp, PetscInt iter, PetscReal rnorm, void ∗dummy)
- PetscErrorCode CreatePETScSolver (peqn_ ∗peqn)

  Initializes PETSc solver.
- PetscErrorCode DestroyHypreMatrix (peqn_ ∗peqn)

  Destroys HYPRE matrix.
- PetscErrorCode DestroyHypreVector (peqn_ ∗peqn)

  Destroys HYPRE vector.
- PetscErrorCode DestroyPETScVector (peqn_ ∗peqn)

  Destroys PETSc vector.
- PetscErrorCode DestroyHypreSolver (peqn_ ∗peqn)

  Destroys HYPRE solver.
- PetscErrorCode DestroyPETScSolver (peqn_ ∗peqn)

  Destroys PETSc solver.

- cellIds GetIdFromStencil (int stencil, int k, int j, int i)
  get the cell id from stencil position
- PetscErrorCode SetCoeffMatrix (peqn_ *peqn)
  Compute coefficient matrix.
- PetscErrorCode Petsc2HypreVector (Vec &A, HYPRE_IJVector &B, HYPRE_Int startID)
  Transfer vector values from Petsc 2 Hypre.
- PetscErrorCode Hypre2PetscVector (HYPRE_IJVector &B, Vec &A, HYPRE_Int startID)
  Transfer vector values from Hypre 2 Petsc.
- PetscErrorCode phiToPhi (peqn_ *peqn)
  Convert phi to Phi (1D to 3D vector)
- PetscReal L2NormHypre (peqn_ *peqn, HYPRE_IJMatrix &A, HYPRE_IJVector &X, HYPRE_IJVector &B)
  Compute L2 norm of system residual.
- PetscErrorCode SubtractAverageHypre (peqn_ *peqn, HYPRE_IJVector &B)
  Subtract average from the solution.
- PetscErrorCode SubtractAveragePETSc (peqn_ *peqn, Vec &B)
  Subtract average from the solution.
- PetscErrorCode SetRHS (peqn_ *peqn)
  Set RHS of the pressure equation.
- PetscErrorCode AdjustIBMFlux (peqn_ *peqn)
  Correct IBM volume flux.
- PetscErrorCode ProjectVelocity (peqn_ *peqn)
  Project Ucont into an incompressible space.
- PetscErrorCode UpdatePressure (peqn_ *peqn)
  Update pressure and subtract average.
- PetscErrorCode GradP (peqn_ *peqn)
  Compute pressure gradient term.
- PetscErrorCode SolvePEqn (peqn_ *peqn)
  Compute pressure gradient term.
- PetscErrorCode SetPressureReference (peqn_ *peqn)
  Set pressure = 0 at k = 0, j = 0, i = 0.
- PetscErrorCode ContinuityErrors (peqn_ *peqn)
  Compute continuity errors (also calculates which cell and processor has the max)
- PetscErrorCode ContinuityErrorsOptimized (peqn_ *peqn)
  Compute continuity errors (only prints the max)

5.35.1 Detailed Description

Contains P equation function definitions.

5.36 src/precursor.c File Reference

Contains top to bottom level routines for the concurrent precursor method.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/inflow.h"
#include "include/initialization.h"
#include "include/initialField.h"
```

Functions

- PetscErrorCode SetSolutionFlagsPrecursor (domain_ *domain)

  Precursor solution flags definition.
- PetscErrorCode concurrentPrecursorInitialize (abl_ *abl)

  initialize concurrent precursor
- PetscErrorCode SetStartTimePrecursor (domain_ *domain, abl_ *abl)

  Checks that start time is available to read if spinUp == 0.
- PetscErrorCode InitializeMeshPrecursor (abl_ *abl)

  Precursor mesh initialization.
- PetscErrorCode concurrentPrecursorSolve (abl_ *abl)

  Solve concurrent precursor.
- PetscErrorCode SetBoundaryConditionsPrecursor (mesh_ *mesh)

  Precursor boundary conditions initialization function.
- PetscErrorCode SetInflowFunctionsPrecursor (mesh_ *mesh)

  Sets the inflow function on the precursor.
- PetscErrorCode MapInitialConditionPrecursor (abl_ *abl)

  Map fields from successor to precursor.
- PetscErrorCode successorPrecursorMapVectorField (abl_ *abl, Vec &Source, Vec &Target, Vec &lTarget)

  Map vector field from successor to precursor.
- PetscErrorCode successorPrecursorMapScalarField (abl_ *abl, Vec &Source, Vec &Target, Vec &lTarget)

  Map scalar tor field from successor to precursor.
- PetscErrorCode ABLInitializePrecursor (domain_ *domain)

  Initialize abl for precursor (x damping layer is not set)

5.36.1 Detailed Description

Contains top to bottom level routines for the concurrent precursor method.

5.37 src/teqn.c File Reference

Contains T equation function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
```

Functions

- PetscErrorCode **SNESMonitorT** (SNES snes, PetscInt iter, PetscReal rnorm, void *comm)
- PetscErrorCode InitializeTEqn (teqn_ *teqn)

  Initializes Teqn environment.
- PetscErrorCode CorrectSourceTermsT (teqn_ *teqn, PetscInt print)

  Compute temperature control source term.
- PetscErrorCode ghGradRhoK (teqn_ *teqn)

  *Computes g*h times gradient of rho_k / rho_0.*

- PetscErrorCode correctDampingSourcesT (teqn_ *teqn)

  Compute tBar state for lateral damping region.
- PetscErrorCode dampingSourceT (teqn_ *teqn, Vec &Rhs, PetscReal scale)

  Apply fringe region damping.
- PetscErrorCode sourceT (teqn_ *teqn, Vec &Rhs, PetscReal scale)

  Apply temperature control.
- PetscErrorCode FormT (teqn_ *teqn, Vec &Rhs, PetscReal scale)

  RHS of the potential temperature transport equation.
- PetscErrorCode TeqnSNES (SNES snes, Vec T, Vec Rhs, void *ptr)

  SNES evaulation function.
- PetscErrorCode FormExplicitRhsT (teqn_ *teqn)

  Computed RHS of temperature equation using current lTmprt (updates Rhs), data put in ueqn->Rhs.
- PetscErrorCode TeqnRK4 (teqn_ *teqn)

  solve Teqn using RungeKutta 4
- PetscErrorCode SolveTEqn (teqn_ *teqn)

  Solve T equation.

5.37.1 Detailed Description

Contains T equation function definitions.

5.38 src/turbines.c File Reference

Contains top to bottom level routines for the wind farm modeling.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/turbines.h"
```

Functions

- PetscErrorCode UpdateWindTurbines (farm_ *farm)

  Update wind turbines.
- PetscErrorCode InitializeWindFarm (farm_ *farm)

  Initialize the wind farm.
- PetscErrorCode checkTurbineMesh (farm_ *farm)

  Check that the mesh is resolved around the turbine.
- PetscErrorCode computeRotSpeed (farm_ *farm)

  Solve rotor dynamics and compute filtered rot speed.
- PetscErrorCode rotateBlades (windTurbine *wt, PetscReal angle, PetscInt updateAzimuth)

  Rotate blades (only for ALM)
- PetscErrorCode controlGenSpeed (farm_ *farm)

  Compute generator torque with 5-regions control system model.
- PetscErrorCode controlBldPitch (farm_ *farm)

  Compute blade pitch using the PID control.

- PetscErrorCode controlNacYaw (farm_ *farm)

  Compute nacelle yaw.
- PetscErrorCode windFarmControl (farm_ *farm)

  Compute wind turbine control based on wind farm controller.
- PetscErrorCode findControlledPointsRotor (farm_ *farm)

  Discrimination algorithm: find out which points of each rotor are controlled by this processor.
- PetscErrorCode findControlledPointsSample (farm_ *farm)

  Discrimination algorithm: find out which points of each sample rig are controlled by this processor.
- PetscErrorCode findControlledPointsTower (farm_ *farm)

  Discrimination algorithm: find out which points of each tower are controlled by this processor.
- PetscErrorCode findControlledPointsNacelle (farm_ *farm)

  Discrimination algorithm: find out which points of each nacelle are controlled by this processor.
- PetscErrorCode computeWindVectorsRotor (farm_ *farm)

  Compute wind velocity at the rotor mesh points.
- PetscErrorCode computeWindVectorsTower (farm_ *farm)

  Compute wind velocity at the tower mesh points.
- PetscErrorCode computeWindVectorsNacelle (farm_ *farm)

  Compute wind velocity at the nacelle mesh points.
- PetscErrorCode computeWindVectorsSample (farm_ *farm)

  Compute wind velocity at the sample mesh points.
- PetscErrorCode computeBladeForce (farm_ *farm)

  Compute aerodynamic forces at the turbine mesh points.
- PetscErrorCode projectBladeForce (farm_ *farm)

  Project the wind turbine forces on the background mesh.
- PetscErrorCode projectTowerForce (farm_ *farm)

  Compute and project the tower forces on the background mesh.
- PetscErrorCode projectNacelleForce (farm_ *farm)

  Compute and project the nacelle forces on the background mesh.
- PetscErrorCode bodyForceCartesian2Contravariant (farm_ *farm)

  Transform the cartesian body force to contravariant.
- PetscErrorCode computeMaxTipSpeed (farm_ *farm)

  Compute max tip speed and activate CFL control flag.
- PetscErrorCode windTurbinesWrite (farm_ *farm)

  Write output if applicable.
- PetscErrorCode windTurbinesWriteCheckpoint (farm_ *farm)

  Write checkpoint file.
- PetscErrorCode windTurbinesReadCheckpoint (farm_ *farm)

  Read checkpoint file and prepare wind turbines.
- PetscErrorCode **initADM** (windTurbine *wt, Cmpnts &base, const word meshName)
- PetscErrorCode initUADM (windTurbine *wt, Cmpnts &base, const word meshName)

  Initialize the Uniform Actuator Disk Model.
- PetscErrorCode initSamplePoints (windTurbine *wt, Cmpnts &base, const word meshName)

  Initialize the upstream sample points data structure.
- PetscErrorCode initALM (windTurbine *wt, Cmpnts &base, const word meshName)

  Initializes the ALM reading from files and allocating memory.
- PetscErrorCode initAFM (windTurbine *wt, Cmpnts &base, const word meshName)

  Initializes the AFM reading from files and allocating memory.
- PetscErrorCode initTwrModel (windTurbine *wt, Cmpnts &base)

  Initializes the tower model.
- PetscErrorCode initNacModel (windTurbine *wt, Cmpnts &base)

  Initializes the nacelle model.

- PetscErrorCode initControlledCells (farm_ *farm)

  Initialize sphere cells and see what proc controls which turbine.
- PetscErrorCode initSampleControlledCells (farm_ *farm)

  Initialize sphere cells and see what proc controls which sample point.
- PetscErrorCode writeFarmADMesh (farm_ *farm)

  Write the wind farm AD mesh to ucd file.
- PetscErrorCode writeFarmTwrMesh (farm_ *farm)

  Write the wind farm tower mesh to ucd file.
- PetscErrorCode writeFarmALMesh (farm_ *farm)

  Write the wind farm AL mesh to ucd file.
- PetscErrorCode readFarmProperties (farm_ *farm)

  Read the farm_Properties file and fill the structs.
- PetscErrorCode readTurbineArray (farm_ *farm)

  Read the wind turbines inside the farm_Properties file.
- PetscErrorCode readTurbineProperties (windTurbine *wt, const char *dictName, const word meshName, const word modelName)

  Fill the windTurbine struct reading in the file named as the wind turbine type.
- PetscErrorCode readAirfoilProperties (windTurbine *wt, const char *dictName)

  Reads the names of the airfoil used in the turbine (given in the airfoils subdict inside the file named as the wind turbine type)
- PetscErrorCode readAirfoilTable (foilInfo *af, const char *tableName)

  Reads the 2D airfoil tables given in the file named as the airfoil.
- PetscErrorCode readBladeProperties (windTurbine *wt, const char *dictName, const PetscInt read↩ Thickness)

  Reads the blades aero properties used in the turbine (given in the bladeData subdict inside the file named as the wind turbine type)
- PetscErrorCode readTowerProperties (windTurbine *wt, const char *dictName)

  Reads the tower properties used in the turbine (given in the towerData subdict inside the file named as the wind turbine)
- PetscErrorCode readNacelleProperties (windTurbine *wt, const char *dictName)

  Reads the nacelle properties used in the turbine (given in the nacelleData subdict inside the file named as the wind turbine)
- PetscErrorCode readGenControllerParameters (windTurbine *wt, const char *dictName, const char *mesh↩ Name)

  Reads generator torque controller parameters.
- PetscErrorCode readPitchControllerParameters (windTurbine *wt, const char *dictName, const char *meshName)

  Reads blade pitch controller parameters.
- PetscErrorCode readYawControllerParameters (windTurbine *wt, const char *dictName, const char *mesh↩ Name)

  Reads nacelle yaw controller parameters.
- PetscErrorCode readWindFarmControlTable (windTurbine *wt)

  Read wind farm controller table (1 header and time - value list)
- PetscErrorCode checkPointDiscriminationRotor (farm_ *farm)

  Debug check for the discrimination algorithm on the rotor.
- PetscErrorCode checkPointDiscriminationTower (farm_ *farm)

  Debug check for the discrimination algorithm on the tower.
- PetscErrorCode checkPointDiscriminationNacelle (farm_ *farm)

  Debug check for the discrimination algorithm on the nacelle.
- PetscErrorCode checkPointDiscriminationSample (farm_ *farm)

  Debug check for the discrimination algorithm on sample points.
- PetscErrorCode printFarmProperties (farm_ *farm)

  Print wind farm information.

5.38.1 Detailed Description

Contains top to bottom level routines for the wind farm modeling.

5.39 src/ueqn.c File Reference

Contains U equation function definitions.

```
#include "include/base.h"
#include "include/domain.h"
#include "include/io.h"
#include "include/inline.h"
#include "include/inflow.h"
```

Functions

- PetscErrorCode **SNESMonitorU** (SNES snes, PetscInt iter, PetscReal rnorm, void *comm)
- PetscErrorCode InitializeUEqn (ueqn_ *ueqn)

  Initializes Ueqn environment.
- PetscErrorCode UpdateFluxLimiter (ueqn_ *ueqn)

  Updates flux limiter.
- PetscErrorCode CorrectSourceTerms (ueqn_ *ueqn, PetscInt print)

  Update driving source terms.
- PetscErrorCode sourceU (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Compute driving source term.
- PetscErrorCode mapYDamping (ueqn_ *ueqn)

  finish the mapping of the ydamping source
- PetscErrorCode correctDampingSources (ueqn_ *ueqn)

  Correct damping source terms.
- PetscErrorCode dampingSourceU (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Compute damping source terms (x and z)
- PetscErrorCode Coriolis (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Compute Coriolis source term.
- PetscErrorCode CanopyForce (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Compute Side Force source term.
- PetscErrorCode Buoyancy (ueqn_ *ueqn, PetscReal scale)

  Compute buoyancy term.
- PetscErrorCode contravariantToCartesian (ueqn_ *ueqn)

  Transform velocity from contravariant to cartesian.
- PetscErrorCode contravariantToCartesianGeneric (mesh_ *mesh, Vec &lCont, Vec &lCat)

  Transform generic local vector from contravariant to cartesian.
- PetscErrorCode adjustFluxes (ueqn_ *ueqn)

  Adjust fluxes to obey mass conservation.
- PetscErrorCode adjustFluxesOverset (ueqn_ *ueqn)

  Adjust fluxes to obey mass conservation in the overset domain.
- PetscErrorCode FormU (ueqn_ *ueqn, Vec &Rhs, PetscReal scale)

  Viscous and divergence terms.
- PetscErrorCode UeqnSNES (SNES snes, Vec Ucont, Vec Rhs, void *ptr)

SNES evaluation function.

- PetscErrorCode FormExplicitRhsU (ueqn_ *ueqn)

  Computed RHS of momentum equation using current lUcont (updates Rhs), data put in ueqn->Rhs.

- PetscErrorCode UeqnEuler (ueqn_ *ueqn)

  Solves ueqn using explicit euler.

- PetscErrorCode UeqnRK4 (ueqn_ *ueqn)

  Solves ueqn using 4 stages runge kutta.

- PetscErrorCode SolveUEqn (ueqn_ *ueqn)

  Solve the momentum equation.

5.39.1 Detailed Description

Contains U equation function definitions.

Index

abl_, 15
access_, 21
Acell, 22
acquisition.c
    read3LMFields, 78
acquisition.h
    read3LMFields, 86
acquisition_, 22
ADM, 23
AFM, 25
ALM, 26
anemometer, 27

bladeAeroInfo, 28
boundingBox, 29

Cabot, 29
cellIds, 30
cellList, 30
cellNode, 31
clock_, 31
Cmpnts, 32
computeEm
    inline.h, 101
constants_, 32
Cpt2D, 32

Dcell, 33
domain_, 33

elementBox, 34

Face, 35
farm_, 35
flags_, 36
foilInfo, 37

getTimeList
    io.c, 120
    tosca2PV.h, 113

HalfEdge, 37

ibm_, 37
ibmFluidCell, 39
ibmInput.c
    readIBMSurfaceFileAbaqusInp, 83
ibmMesh, 40
ibmNode, 40
ibmObject, 41
ibmPitchMotion, 42
ibmRotation, 42
ibmSineMotion, 42
inflowData, 43
initialField.c
    SetInitialField, 116
initialField.h
    SetInitialField, 94
initialization.c
    simulationInitialize, 118
initialization.h
    SetInitialField, 95
    simulationInitialize, 95
inletFunctions, 43
inletFunctionTypes, 44
inline.h
    computeEm, 101
io.c
    getTimeList, 120
io_, 46

les.c
    UpdateCs, 121
les_, 47
list, 48
LogLawAPG, 49

mapInfo, 49
mesh_, 49

nacelleModel, 51
node, 52

overset_, 52
oversetMotion, 53

patchVectorField, 54
peqn_, 54
postProcess, 56
PowerLawAPG, 57
precursor_, 57

read3LMFields
    acquisition.c, 78
    acquisition.h, 86
readIBMSurfaceFileAbaqusInp
    ibmInput.c, 83

scalarBC, 58
searchBox, 58
SetInitialField
    initialField.c, 116

initialField.h, 94
initialization.h, 95
Shumann, 59
simInfo_, 60
simulationInitialize
initialization.c, 118
initialization.h, 95
src/abl.c, 75
src/acquisition.c, 76
src/boundary.c, 78
src/clock.c, 79
src/ibm.c, 79
src/ibmInput.c, 82
src/include/abl.h, 83
src/include/acquisition.h, 84
src/include/base.h, 86
src/include/boundary.h, 87
src/include/clock.h, 88
src/include/domain.h, 89
src/include/ibm.h, 89
src/include/inflow.h, 93
src/include/initialField.h, 94
src/include/initialization.h, 95
src/include/inline.h, 96
src/include/io.h, 101
src/include/mesh.h, 104
src/include/objects.h, 104
src/include/overset.h, 104
src/include/peqn.h, 106
src/include/precursor.h, 107
src/include/teqn.h, 108
src/include/tosca2PV.h, 109
src/include/ueqn.h, 113
src/include/wallfunctions.h, 114
src/include/wallmodel.h, 115
src/initialField.c, 116
src/initialization.c, 117
src/io.c, 118
src/les.c, 121
src/mesh.c, 122
src/overset.c, 122
src/peqn.c, 123
src/precursor.c, 125
src/teqn.c, 126
src/turbines.c, 127
src/ueqn.c, 130
surface, 60
symmTensor, 60

teqn_, 61
tosca2PV.h
getTimeList, 113
towerModel, 62

UADM, 63
ueqn_, 64
UpdateCs
les.c, 121
upSampling, 66
vectorBC, 67
Vertex, 68

wallModel, 68
windTurbine, 69