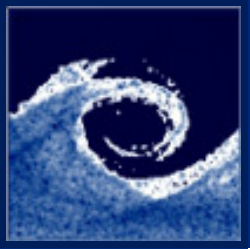


Programming in OpenFOAM

Lecture 7

Balogh Miklós

March 25, 2014

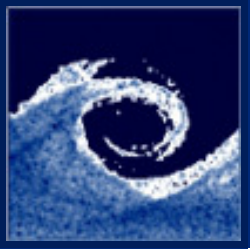


Table of Contents

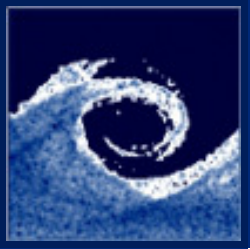
Basic concept

Programming

Variables

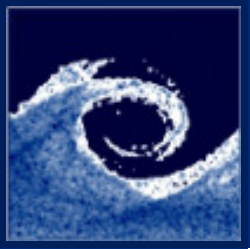
Loops

1. Basic concept
2. Programming
3. Variables
4. Loops



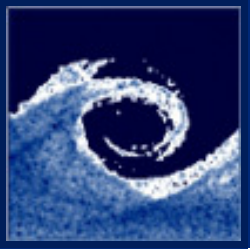
Difference between commercial and open-source

- Commercial solvers (Fluent, CFX, StarCCM++)
  - Black-box (way of implementation is not known)
  - UDF - User Defined Functions (ANSI C, Fortran)
  - Modification is limited
- OpenFOAM
  - The source code is accessible
  - Direct implementation is possible
  - No limitation (limited by common sense)



Applications and libraries

- Applications - standalone objects
  - Solvers - icoFoam, simpleFoam, etc.
  - Utilities - blockMesh, setFields, etc.
- Libraries
  - Turbulence models
  - Specific inlet/outlet boundary conditions
  - Specific wall functions
  - Additional source terms



Programming language of OpenFOAM

Programming

Balogh Miklós

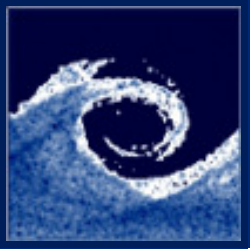
Basic concept

Programming

Variables

Loops

- OpenFOAM is a C++ library (toolbox)
  - Object-oriented, such as C++
  - Provide the mechanism (classes) to declare types and associated operations
  - In verbal and mathematical languages used in science and engineering
- Interpretation
  - Physics: Velocity field and magnitude
  - Math: U and $|U|$
  - OpenFOAM: volVectorField U and volScalarField Umag = mag(U)

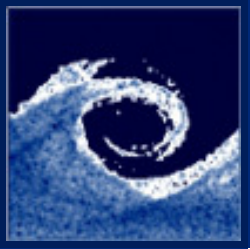


Equation representation in OpenFOAM

Example: $\frac{\partial \rho U}{\partial t} + \nabla \cdot \phi U - \nabla \cdot \mu \nabla U = -\nabla p$

Listing 1: solver code

```
1     solve
2     (
3         fvm::ddt(rho, U)
4       + fvm::div(phi, U)
5       - fvm::laplacian(mu, U)
6         ==
7       - fvc::grad(p)
8     );
```

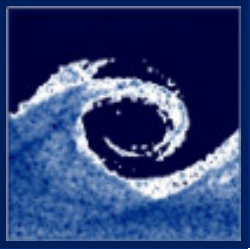


Variables

- Local single variables (e.g. in a loop)
  - scalar
  - vector
  - tensor
- Dimensioned single variables
  - dimensionedScalar
  - dimensionedVector
  - dimensionedTensor
- Dimensioned field variables
  - volScalarField (scalarField)
  - volvectorField (vectorField)
  - volTensorField (tensorField)

Listing 2: single variables

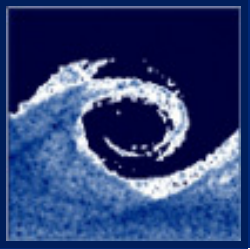
```
1    dimensionedScalar nu(transportProperties.lookup("nu"));
2    dimensionedVector flowDir(transportProperties.lookup("flowDir"));
```



Variables - volScalarField

Listing 3: volScalarField

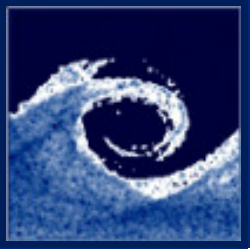
```
1     Info<< "Reading field p\n" << endl;
2     volScalarField p
3     (
4         IOobject
5         (
6             "p",
7             runTime.timeName(),
8             mesh,
9             IOobject::MUST_READ,
10            IOobject::AUTO_WRITE
11        ),
12        mesh
13    );
```



Variables - volVectorField

Listing 4: volVectorField

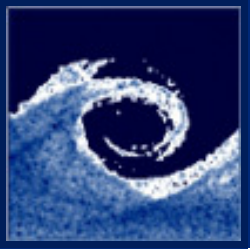
```
1     Info<< "Reading field U\n" << endl;
2     volVectorField U
3     (
4         IOobject
5         (
6             "U",
7             runTime.timeName(),
8             mesh,
9             IOobject::MUST_READ,
10            IOobject::AUTO_WRITE
11        ),
12        mesh
13    );
```



Variables - volTensorField

Listing 5: volTensorField

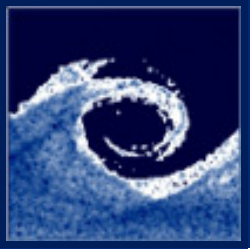
```
1     Info<< "Reading field Permeability\n" << endl;
2     volTensorField Permeability
3     (
4         IOobject
5         (
6             "Permeability",
7             runTime.timeName(),
8             mesh,
9             IOobject::MUST_READ,
10            IOobject::AUTO_WRITE
11        ),
12        mesh
13    );
```



Loops and cycles over cells

Listing 6: loop over cells

```
1     volScalarField centres_ = mag(wallNorm_ & mesh_.C());
2
3     // Limitation of centres to the ABL height in the cells
4     forAll(centres_,cellI)
5     {
6       if(centres_[cellI] > hABL_.value())
7       {
8         centres_[cellI] = hABL_.value();
9       }
10    }
11
12    // Similar to the following
13    volScalarField centres_ = min(mag(wallNorm_ & mesh_.C()),hABL_);
```



Loops and cycles over faces

Listing 7: loop over faces of patches

```
1     // Limitation of centres to the ABL height at the boundaries
2     forAll(patches, patchI)
3     {
4       const fvPatch& curPatch = patches[patchI];
5       forAll(curPatch, faceI)
6       {
7         vector cf = mesh_.Cf().boundaryField()[patchI][faceI];
8         scalar ctemp = mag(wallNorm_.value() & cf);
9         if(ctemp > hABL_.value())
10        {
11          centres_.boundaryField()[patchI][faceI] = hABL_.value();
12        }
13        else
14        {
15          centres_.boundaryField()[patchI][faceI] = ctemp;
16        }
17      }
18    }
```