OpenFOAM selected solver

Roberto Pieri - SCS Italy

16-18 June 2014
Introduction to Navier-Stokes equations and RANS

Turbulence modelling

Numeric discretization
Navier-Stokes equations

\[
\begin{aligned}
\text{Convective term} & \quad \nabla \cdot (U \otimes U) - \nabla \cdot \nu \nabla U = -\frac{1}{\rho} \nabla P \\
\text{Viscous term} & \quad \nabla \cdot U = 0
\end{aligned}
\]

- Equations are directly derived from conservation laws.
- \( U \) is the velocity vector, \( P \) is pressure and \( \rho \) is density.
- System of partial differential equations.
- Equations are valid for viscous, incompressible, steady flows in laminar regime.
Reynolds Averaged Navier-Stokes (RANS) equations

\[
\begin{aligned}
\nabla \cdot (\overline{U} \otimes \overline{U}) + \nabla \cdot (U' \otimes U') - \nabla \cdot \nu \nabla \overline{U} &= -\nabla \overline{p} \\
\nabla \cdot \overline{U} &= 0
\end{aligned}
\]

- \( U = \overline{U} + U' \).
- Equations are obtained decomposing velocity vector and averaging.
- The term \( \nabla \cdot (U' \otimes U') \) represents a new unknown.
- A closure equation is required.
- \( \overline{p} = \overline{P} / \rho \), it is only a mathematical function (equation of state is not present).
**Turbulence modelling**

There are two different classes of models:

**Eddy-viscosity models**
- Based on Boussinesq hypothesis
- Very large number of models
- Different models for different flow conditions

**Reynolds stress models**
- More recent
- Equations for every term of Reynolds’ stress tensor are required

We are going to discuss the first class of models.
Turbulence modelling
Eddy-viscosity models (I)

- Effective viscosity $\nu_e$ is defined as follow:

$$\nu_e (x) = \nu + \nu_t (x)$$

- Reynolds’ stress tensor can be rewritten as follow:

$$\nabla \cdot (U' \otimes U') = \nabla \cdot (\nu_t (x) \nabla \bar{U})$$

- Momentum equation can be rewritten:

$$\nabla \cdot (\bar{U} \otimes \bar{U}) - \nabla \cdot \nu_e \nabla \bar{U} = -\nabla \bar{p}$$
The new system of equations is:

\[
\begin{aligned}
\nabla \cdot (\overline{U} \otimes \overline{U}) - \nabla \cdot \nu_e (x) \nabla \overline{U} &= -\nabla \overline{p} \\
\nabla \cdot \overline{U} &= 0
\end{aligned}
\]

with

\[
\nu_e (x) = \nu + \nu_t (x)
\]

- In this formulation the model is totally confined in \( \nu_t (x) \).
- A model for the effective viscosity is needed.
Turbulence modelling

**Eddy-viscosity models (III)**

Eddy-viscosity models are divided in three classes depending on the number of differential equations needed for the closure of the problem.

- **0-equation** models (mixing length).
- **1-equation** models (Spalart-Allmaras, $k$ equation, ...).
- **2-equations** models ($k - \varepsilon$, $k - \omega$, ...).
An example of a 2-equation model is $k - \omega$.

- An equation for $k$ is needed.
- An equation for $\omega$ is needed.
- The model is complete:

$$\nu_t = C_\mu \frac{k}{\omega}$$

where $C_\mu$ is a constant (possible tuning).
OpenFOAM solvers

- Large number of solvers.
- Choose the solver that best suits your case study (compressible/incompressible, heat transfer, multiphase...).
- A first setup is always given by the tutorials.
- **Attention:** tutorials’ setup may not work for your case.

One of the most used solvers is *simpleFoam*. 
**OpenFOAM solvers**

Semi-Implicit Method for Pressure-Linked Equations (simpleFoam)

- Suitable for incompressible, steady-state, viscous flows in laminar or turbulent regime.
- Used for internal and external flows.
- Very large documentation and test cases from the user community.
OpenFOAM solvers

Start time step

Momentum matrix

\[ [U \text{ Eqn}] \]

Under-relax \[ [U \text{ Eqn}] \]

Solve momentum

\[ [U \text{ Eqn}] = -\nabla p \]

Evaluate \( \mathbf{H(U)}, A \)

Pressure corrector

\[ \nabla \cdot \left( \frac{1}{A} \nabla p \right) = \nabla \cdot \left( \frac{\mathbf{H}}{A} \right) \]

Under-relax \( p \)

Momentum corrector and flux corrector

\[ U = \frac{\mathbf{H}}{A} - \frac{1}{A} \nabla p \]

End time step

\( \phi = S_f \cdot \left[ (\mathbf{H}/A)_f - (1/A)_f (\nabla p)_f \right] \)

NEW \( p, U, \phi \)

SIMPLE algorithm
OpenFOAM solvers
SIMPLE implementation in OpenFOAM

```
solve
(
    fvm::div(phi, U)
    + turbulence->divDevReff(U)
    ==
    - fvc::grad(p)
);
```

- Top level code represents the equations being solved.
- *OpenFOAM* has functions for derivatives. *e.g.* `div`, `grad`, `laplacian`, `curl`.
- `fvc::` returns a field, it is used to calculate the pressure gradient with current values (explicit).
- `fvm::` returns an `fvMatrix`, it is used in order to discretise a term into matrix equation you wish to solve (implicit).
- `solve` function solves the equation.
OpenFOAM solvers

Other solvers

- **pisoFoam**: transient solver for incompressible flow;
- **pimpleFoam**: merged PISO-SIMPLE
  - can run transient; no Courant number limited, unlike PISO;
  - can run pseudo-transient: big time step to reach steady-state with minimal under-relaxation;
  - can be used in substitution of SIMPLE, gaining in stability of the solver.
- **buoyantBoussinesqSimpleFoam**: steady-state solver for buoyant, turbulent flow of incompressible fluids including Boussinesq approximation for stratified flow

\[ \rho_k = 1 - \beta (\bar{T} - T_0) \]