OpenFOAM Workshop
Spring 2018

icoFoam solver

It is a transient solver for incompressible, laminar flow of Newtonian fluids. Let's explore the solver and derive the equations as are presented in the source code:

$FOAM_APP/solvers/incompressible/icoFoam/icoFoam.C

```
...
fvVectorMatrix UEqn
(
    fvm::ddt(U)
  + fvm::div(phi, U)
  - fvm::laplacian(nu, U)
);
...
fvScalarMatrix pEqn
(
    fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
);
...
U = HbyA - rAU*fvc::grad(p);
```

NS equations for incompressible and laminar flow are:

$$\partial_t \boldsymbol{u} + \nabla \bullet (\boldsymbol{u}\boldsymbol{u}) = \nabla \bullet (\nu \nabla \boldsymbol{u}) - \nabla p \tag{1}$$

$$\nabla \bullet \boldsymbol{u} = 0 \tag{2}$$

Complexities of the equations are:

- Non-linear term; $\nabla \bullet (\boldsymbol{u}\boldsymbol{u})$,
- Pressure-velocity coupling (Eq. 1).

These complexities can be addressed by:

- Using momentum predictor ($\boldsymbol{u}^* = -\nabla p$) which is not efficient so generally not recommended
- Freezing velocity ($\nabla \bullet (\boldsymbol{u}\boldsymbol{u}) \approx \nabla \bullet (\boldsymbol{u}^o \boldsymbol{u}^n)$) and applying pressure corrector

Let's define $r = \nabla \bullet (\nu \nabla \boldsymbol{u})$ and A as the discretization coefficient matrix, then discretize Eq. 1:

$$A_{ii}\boldsymbol{u}_i + \sum_j A_{ij}\boldsymbol{u}_j = r - \nabla p$$

$$A_{ii}\boldsymbol{u}_i = \underbrace{r - \sum_j A_{ij}\boldsymbol{u}_j}_{H(\boldsymbol{u})} - \nabla p$$

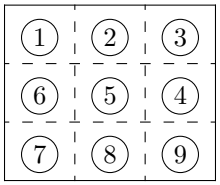
$$\boldsymbol{u}_i = \underbrace{A_{ii}^{-1}}_{\texttt{rAU}}(H(\boldsymbol{u}) - \nabla p)$$

$$\boldsymbol{u}_i = \underbrace{A_{ii}^{-1}H(\boldsymbol{u})}_{\texttt{HbyA}} - \underbrace{A_{ii}^{-1}\nabla p}_{\texttt{rAU*fvc::grad(p)}}$$

Let's take divergence of the equation and use Eq. 2 for calculating pressure:

$$\underbrace{\nabla \bullet (A_{ii}^{-1}\nabla p)}_{\texttt{fvm::laplacian(rAU, p)}} = \underbrace{\nabla \bullet (A_{ii}^{-1}H(\boldsymbol{u}))}_{\texttt{fvc::div(phiHbyA)}}$$

This procedure is repeated using the newly obtained pressure and velocity until u^o is "sufficiently" close to u^n. The criterion for terminating the loop can be controlled by setting desired tolerance and maximum number of iterations in fvSolution dictionary.
Regarding the velocity matrix coefficient, A, let's consider a mesh with 9 cells:



Considering that each face has an owner cell (normal pointing away from the cell) and a neighbor cell (normal pointing toward the cell) the coefficient matrix becomes as follows:

	1	2	3	4	5	6	7	8	9
1	D	N				N			
2	O	D	N		N				
3		O	D	N					
4			O	D	N				N
5		O		O	D	N		N	
6	O				O	D	N		
7						O	D	N	
8					O		O	D	N
9				O				O	D

where N is contribution from neighbor cells and O from owner cells. D represents the diagonal elements. For more information check out openfoamwiki.net website.