Computational Fluid Dynamics (CFD)

- Navier-Stokes Equations
- Reynolds Number
- Numerical Analysis of Navier-Stokes Equations
- SIMPLE and PISO algorithm
- OpenFoam Project
- Some results for the Lid-driven cavity flow

Navier Stokes Equations

For an Incompressible Newtonian Fluid

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{u} \right) = 0$$ Continuity Equation

$$\rho \underbrace{\left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \vec{\nabla} \vec{u} \right)}_{\frac{D\vec{u}}{Dt}} = \underbrace{-\vec{\nabla} p}_{\substack{\text{Pressure} \\ \text{Forces}}} + \underbrace{\mu \nabla^2 \vec{u}}_{\substack{\text{Viscous} \\ \text{Forces}}}$$

ρ : density, $\vec{u}$: velocity, p : pressure, μ : viscosity

Reynolds Number

In order to make some estimates, let us suppose we have a typical change in speed given by U over a typical length L.

$$\frac{\rho U^2}{L} \approx \text{Inertial Forces}$$

$$\frac{\mu U}{L^2} \approx \text{Viscous Forces}$$

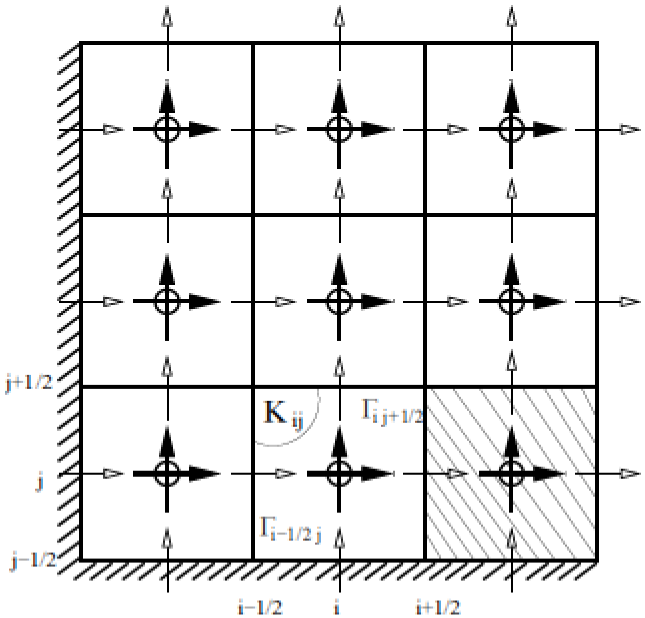
$$\frac{\text{Inertial Forces}}{\text{Viscous Forces}} \approx \frac{LU}{\mu / \rho} = \frac{LU}{\upsilon} = \text{Re} : \text{Reynolds Number}, \qquad \upsilon : \text{Kinematic Viscosity}$$

For low Reynolds number we have Stokes (or creeping) flow. This is the typical regime where cells move.

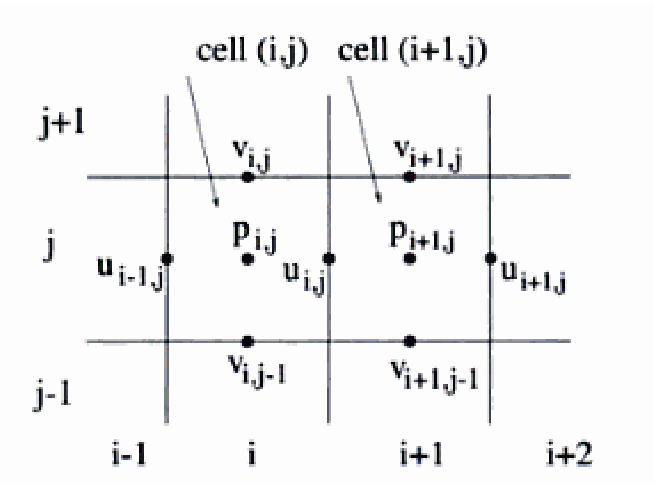
Numerical Analysis of Navier Stokes Equations

The first issue is to select the points in the domain at which the values of the unknown dependent variables are to be computed.

Collocated Arrangement



Staggered Arrangement



Issues solving Navier-Stokes Equations

- Lack of an independent equation for the pressure.
- Presence of non-linear quantities in the convective term.
- The equations are intricately coupled.

Discretisation Process

Time derivative: It is used the explicit Euler method unless detailed time history of the flow be required.

$$\frac{\partial(\rho u)}{\partial t} \approx \frac{(\rho u)^{n+1} - (\rho u)^{n}}{\Delta t}$$

Gradient, Divergence and Laplacian: The discretisation typically is performed using **Gauss'** theorem according to the following equations

$$\int_V \vec{\nabla}\phi dV = \int_s d\vec{S}\phi = \sum_f \vec{S}_f \phi_f \qquad \text{Gradient}$$

$$\int_V \vec{\nabla}\cdot\vec{u} dV = \int_S d\vec{S}\cdot\vec{u} = \sum_f \vec{S}_f \cdot \vec{u} \qquad \text{Divergence}$$

$$\int_V \vec{\nabla}\cdot\left(\Gamma\vec{\nabla}\phi\right) dV = \int_S d\vec{S}\cdot\left(\Gamma\vec{\nabla}\phi\right) = \sum_f \Gamma_f \vec{S}_f \cdot \left(\vec{\nabla}\phi\right)_f \qquad \text{Laplacian}$$

Algorithm: SIMPLE

The algorithm is essentially a guess-and-correct procedure for the calculation of pressure on the staggered grid.

$$\frac{\partial}{\partial x}(\rho uu)+\frac{\partial}{\partial y}(\rho vu)=\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right)-\frac{\partial p}{\partial x} \quad \Rightarrow \quad a_{i,J}u_{i,J}=\sum a_{nb}u_{nb}+\left(p_{I-1,J}-p_{I,J}\right)A_{i,J}$$

$$\frac{\partial}{\partial x}(\rho uv)+\frac{\partial}{\partial y}(\rho vv)=\frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right)-\frac{\partial p}{\partial y} \quad \Rightarrow \quad a_{I,j}v_{I,j}=\sum a_{nb}v_{nb}+\left(p_{I,J-1}-p_{I,J}\right)A_{I,j}$$

$$\frac{\partial}{\partial x}(\rho u)+\frac{\partial}{\partial y}(\rho v)=0 \quad \Rightarrow \quad \left[(\rho uA)_{i+1,J}-(\rho uA)_{i,J}\right]+\left[(\rho vA)_{I,j+1}-(\rho vA)_{I,j}\right]=0$$

From a guessed p^* we get velocities

$$a_{i,J}u^*_{i,J}=\sum a_{nb}u^*_{nb}+\left(p^*_{I-1,J}-p^*_{I,J}\right)A_{i,J} \qquad (1)$$

$$a_{I,j}v^*_{I,j}=\sum a_{nb}v^*_{nb}+\left(p^*_{I,J-1}-p^*_{I,J}\right)A_{I,j} \qquad (2)$$

We define the correction pressure p’ as the difference between the correct pressure and the guessed pressure. Analogously for the velocities.

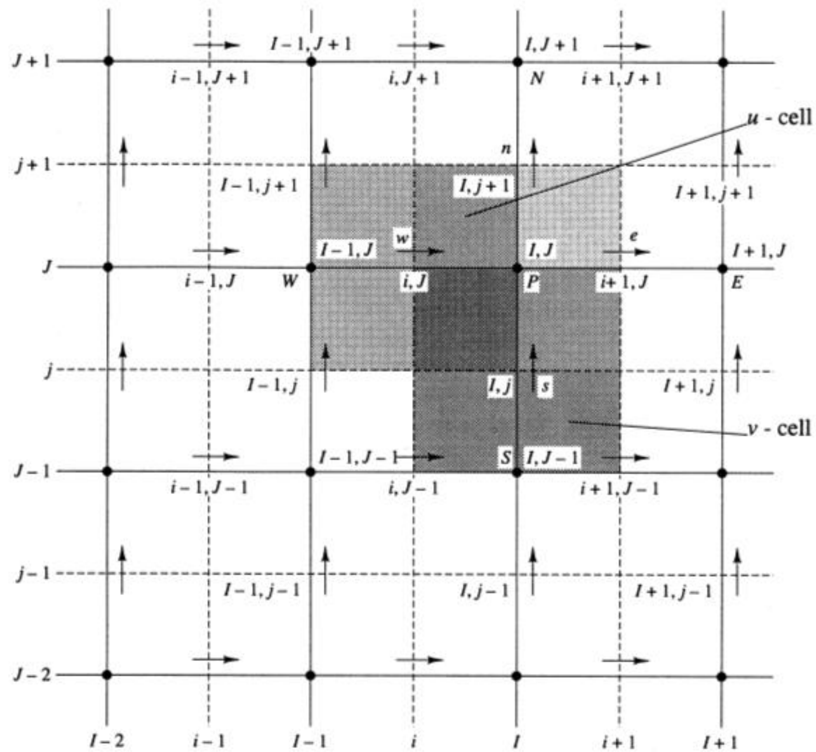
$$p = p^* + p' \qquad u = u^* + u' \qquad v = v^* + v'$$

By subtracting Eqs. (1) and (2) we get

$$a_{i,J} u'_{i,J} = \sum a_{nb} u'_{nb} + \left(p'_{I-1,J} - p'_{I,J}\right) A_{i,J} \quad (3)$$

$$a_{I,j} v'_{I,j} = \sum a_{nb} v'_{nb} + \left(p'_{I,J-1} - p'_{I,J}\right) A_{I,j}$$

$$\text{Key point}: \sum a_{nb} u'_{nb}, \sum a_{nb} v'_{nb} \approx 0$$



From (3) we get u and v in terms of p'

We use continuity Eq. to provide a relation for p'

$$a_{I,J}\,p'_{I,J} = a_{I+1,J}\,p'_{I+1,J} + a_{I-1,J}\,p'_{I-1,J} + a_{I,J+1}\,p'_{I,J+1} + a_{I,J-1}\,p'_{I,J-1}$$

The pressure correction equation is susceptible to divergence Unless some under-relaxation is used. We then have

$$p^{new} = p^{*} + \alpha_p p', \qquad 0 \le \alpha_p \le 1$$

We need some α_p not too small to have appropriate Computation speed and not too big to assure convergence. Unfortunately, the optimum value are flow dependent and must be sough on a case-by-case basis.

The PISO algorithm

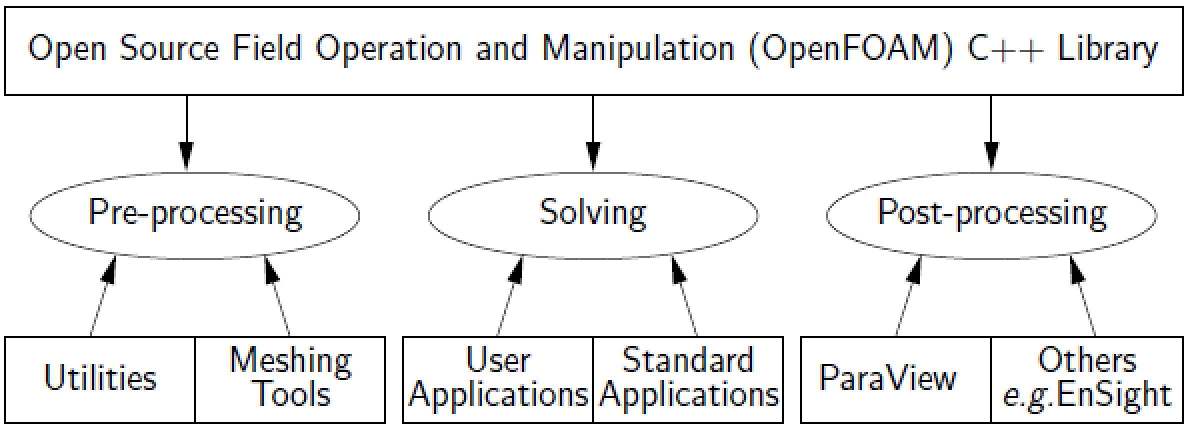
- PISO involves one predictor step and two corrector steps and may be seen as an extension of SIMPLE, with a further corrector step to enhance it.

- PISO was developed originally for the non-interative computation of unsteady compresible flows, but it has been adapted succesfully on steady state problems.

- Although the method implies a considerable increase in Computational effort it has been found to be efficient and fast.

OpenFoam Project

OpenFoam is first and foremost a C++ library for the customisation and extension of numerical solvers for continuum mechanics problems, including computational fluid dynamics (CFD).



Overview of OpenFOAM structure.

Apart from the standard solvers, one of the distinguishing features of OpenFOAM is its relative ease in creating custom solver applications. OpenFOAM allows the user to use syntax that closely resemble the partial differential equations being solved. For example the equation

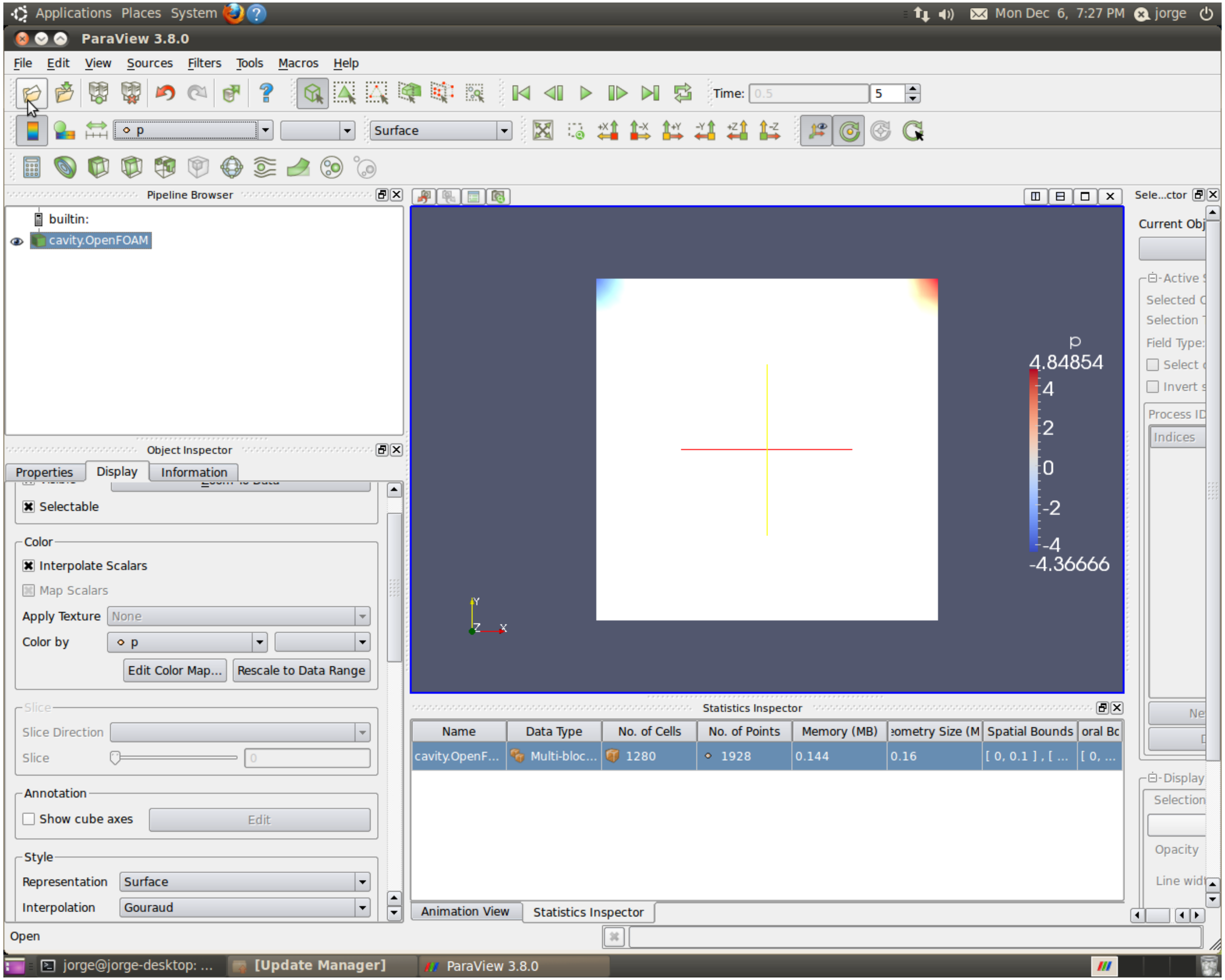
$$\frac{\partial(\rho\vec{u})}{\partial t}+\vec{\nabla}\cdot(\phi\vec{u})-\vec{\nabla}\cdot\mu\vec{\nabla}\vec{u}=-\vec{\nabla}p$$

is represented by the code

```
solve
(
     fvm::ddt(rho,U)
   + fvm::div(phi,U)
   - fvm::laplacian(mu,U)
     ==
   - fvc::grad(p)
);
```

Next I used the program to study the Lid-driven cavity flow. This is a typical phenomena to test CFD algorithms.

Pressure



Velocity flow field