Science and Technology Facilities Council

Scientific Computing

CCP-WSI

a Collaborative Computational Project in Wave Structure Interaction

OpenFOAM Fundamentals

Dr Omar Mahfoze

omar-ahmed.mahfoze@stfc.ac.uk

Table of Content

- What is OpenFOAM?
- OpenFOAM Structure
- Numerics of OpenFOAM
  - Finite Volume Method
  - Time Discretisation
  - Pressure Velocity Coupling
  - Volume of Fluid

Science and Technology Facilities Council

Scientific Computing

What is OpenFOAM?

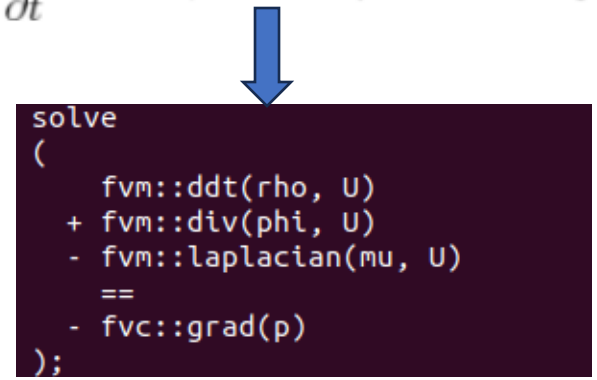
Science and Technology Facilities Council

Scientific Computing

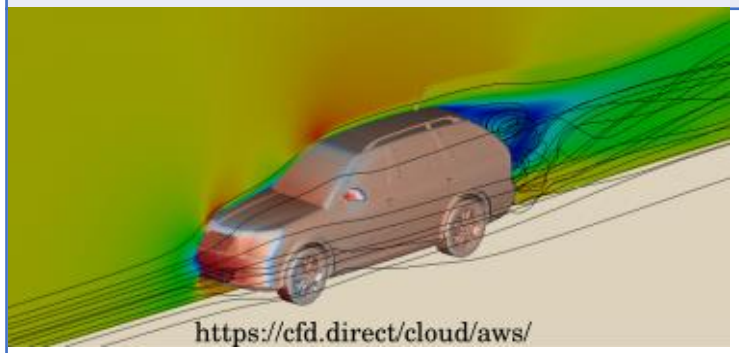
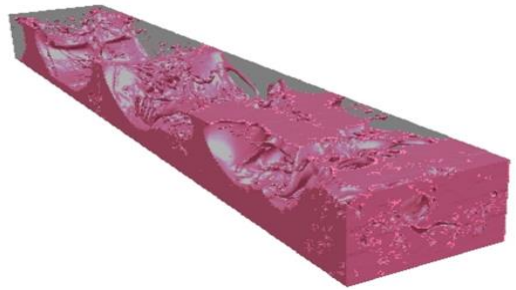
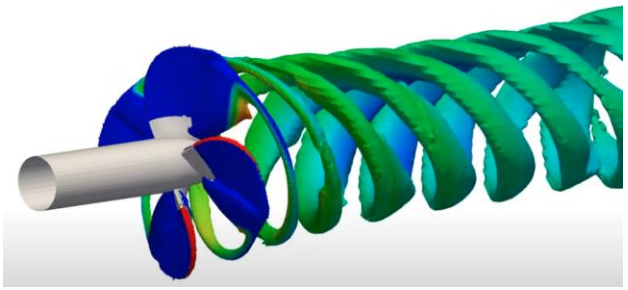
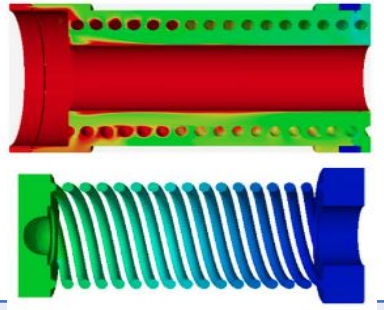
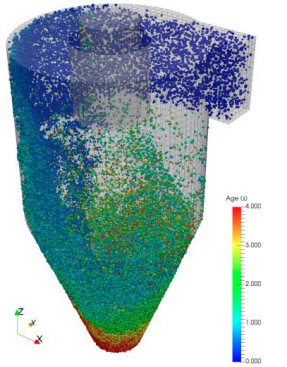
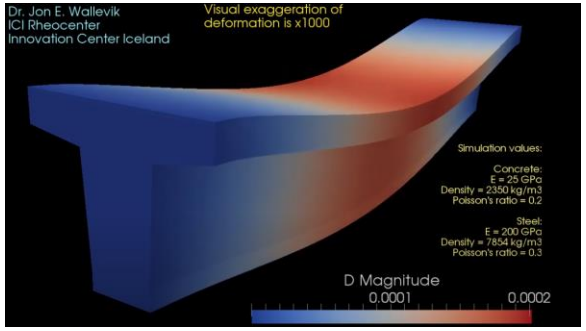
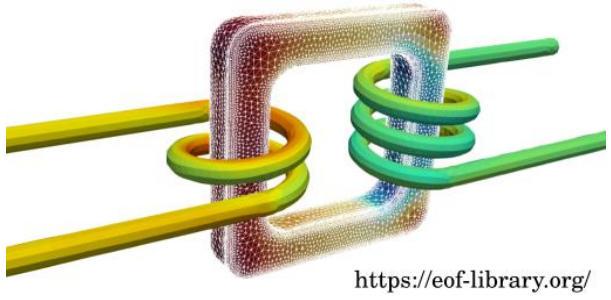
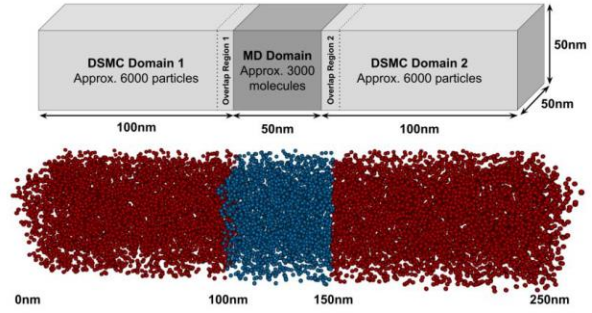
What is OpenFOAM

- FOAM stands for "Field Operation and Manipulation"
- It is a library of tools written in C++ with high utilisation of Object Oriented Programming
- There are different versions:
  - OpenFOAM ESI (OpenFOAM-v20xx) (www.openfoam.com)
  - OpenFOAM foundation(OpenFOAM x) (www.openfoam.org)
  - OpenFOAM Extend (https://github.com/Unofficial-Extend-Project-Mirror)
- The core solvers are similar
- Switching between solvers is easy, giving the capabilities are available in both versions.
- There are some differences
  - ESI has more solvers and utilities (overset method, interIsoFoam)
  - ESI do more frequent releases
  - Foundation has a more generalized implementation
  - Some solvers gives different results

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \bullet \phi \mathbf{U} - \nabla \bullet \mu \nabla \mathbf{U} = -\nabla p$$



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

Single Phase flow	MultiPhase flow [1]	Turbo-machinery [2]
		
Reaching flows [3]	**Conjugate heat transfer [4]**	**Mass methods [5]**
		
Solid Mechanics	**Electro-magnetics**	**Particle based simulations (MD & DSMC) [6]**
		

[1] https://www.mdpi.com/2076-3417/12/17/8481
[2] https://www.youtube.com/watch?v=9eNYtOQlixY&ab_channel=EngineerDo
[3]https://ecn.sandia.gov/diesel-spray-combustion/target-condition/spray-ab/
[4] https://www.sciencedirect.com/science/article/pii/S1359431117349840
[5] https://www.foamacademy.com/wp-content/uploads/2016/11/GOFUN2017_ParticleSimulations_slides.pdf
[6] https://www.sciencedirect.com/science/article/pii/S0045793020302966

Science and Technology Facilities Council

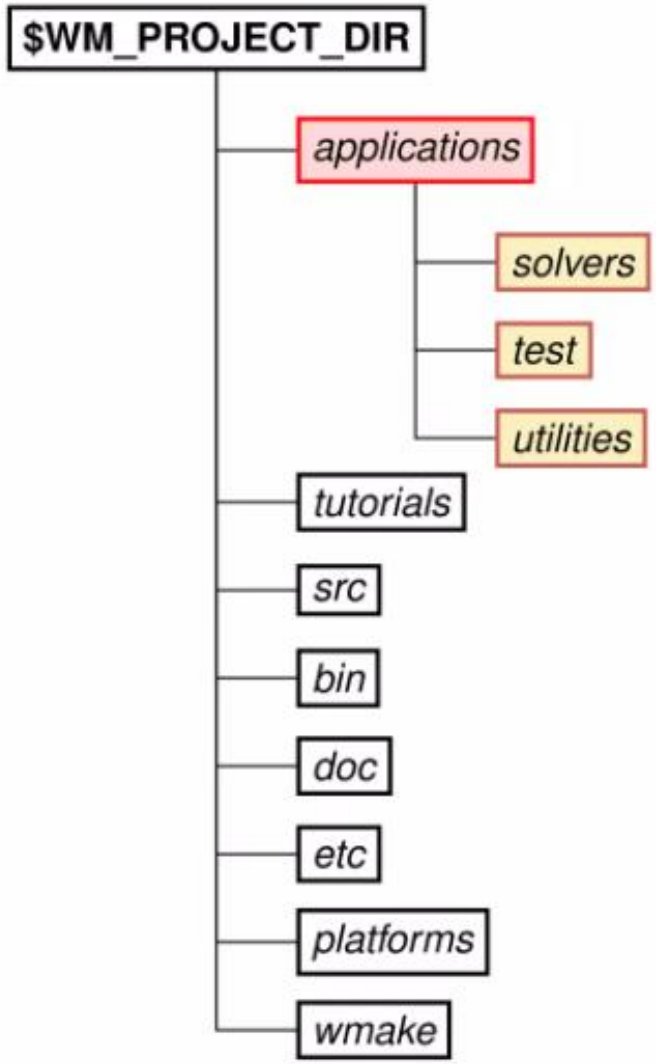
Scientific Computing

OpenFOAM Structure

Science and Technology Facilities Council

Scientific Computing

OpenFOAM Structure



1. *applications*:
   alias: *app* = 'cd $FOAM_APP'.
   This directory contains the source files of all the executables created using the **C++** libraries. It contains the following directories:

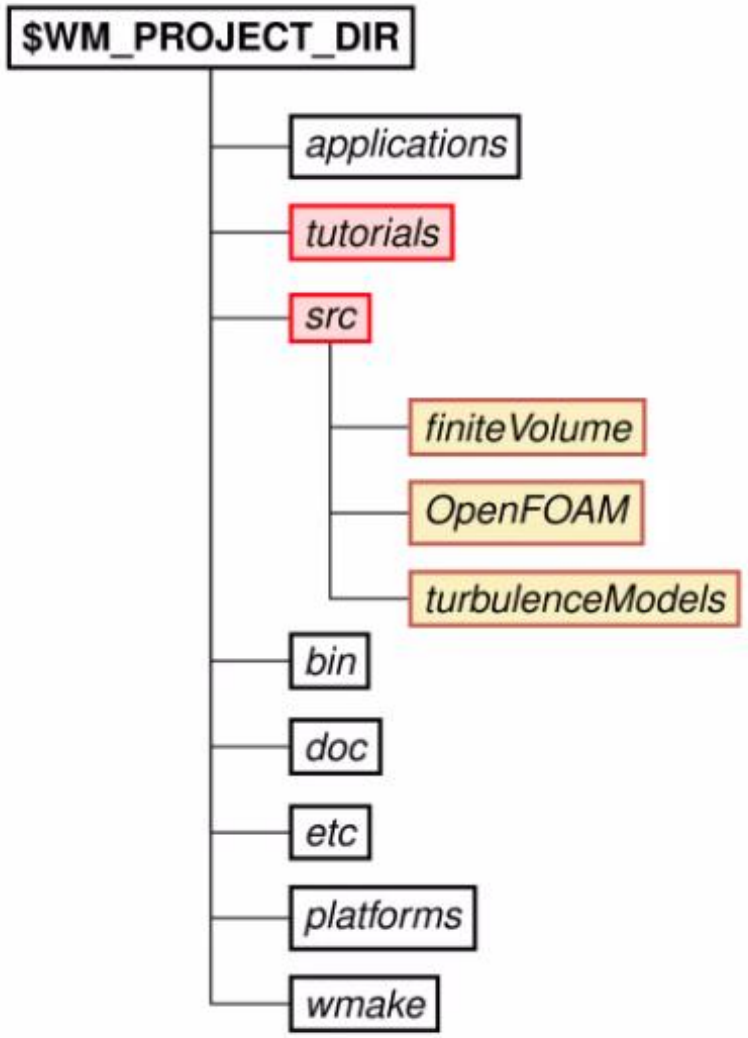
   1.1 *solvers*:
   alias: *sol* = 'cd $FOAM_SOLVERS'.
   Source code to solve a particular continuum mechanics problem.

   1.2 *test*:
   Sample codes to help understand the usage of OpenFOAM libraries.

   1.3 *Utilities*:
   alias: *util* = 'cd $FOAM_UTILITIES'.
   Source code to perform pre- and post-processing tasks involving data manipulation and algebraic manipulations.

Source: https://www.slideshare.net/pengding2/introdcution-to-openfoamworking-with-free-

OpenFOAM Structure



2. *tutorials*:
alias: *tut* = 'cd $FOAM_TUTORIALS'.
Contains tutorials that demonstrate the usage of all solvers and most of the utilities.

3. *src*:
alias: *src* = 'cd $FOAM_SRC'.
It contains several subdirectories which include the source code for all libraries. The important folders are:

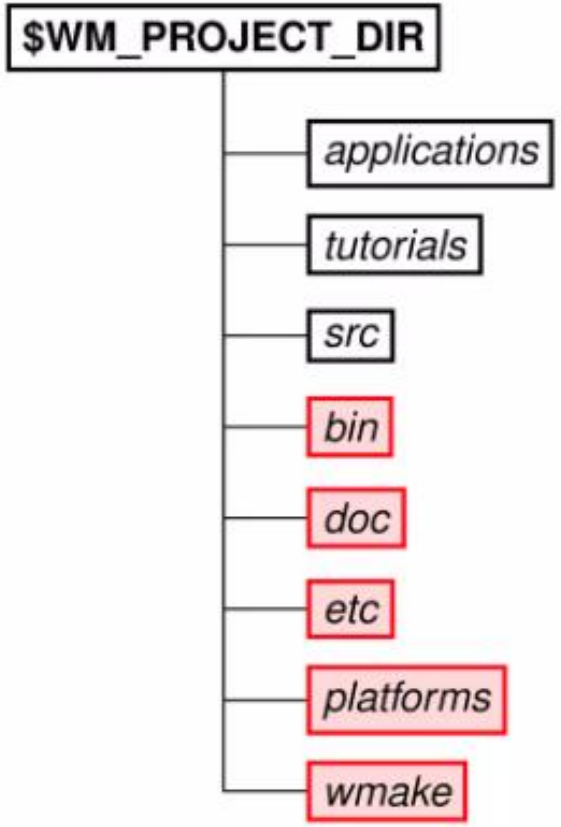
3.1 *finiteVolume*:
alias: *foamfv* = 'cd $FOAM_SRC/finiteVolume'.
Includes classes for finite volume space/time discretisation, boundary conditions etc.

3.2 *OpenFOAM*:
This core library includes important definitions.

3.3 *turbulenceModels*:
Contains libraries for turbulence models.

Source: https://www.slideshare.net/pengding2/introdcution-to-openfoamworking-with-free-software

OpenFOAM Structure



4. *bin*:
   This directory contains shell scripts such as *paraFoam*, *foamLog* etc.

5. *doc*:
   It contains all the documentation relevant to the version of **OpenFOAM** including:
   - 5.1 User and Programmer's guides
   - 5.2 Doxygen generated documentation
   - 5.3 **OpenFOAM** coding style guide

6. *etc*:
   It contains global **OpenFOAM** dictionaries and setup files.

7. *platforms*:
   The binaries generated during the compilation of the applications and the dynamic libraries are stored here.

8. *wmake*:
   Compiler settings are included in this directory including optimisation flags. It also contains *wmake*, a special make command which understands the **OpenFOAM** file structure.

Source: https://www.slideshare.net/pengding2/introdcution-to-openfoamworking-with-free-software

OpenFOAM Case Structure

- Basic OpenFOAM case

$FOAM_TUTORIALS/multiphase/interFoam/RAS/damBreak/damBreak

- This is not a comprehensive list of case files!
- Some files can be placed at different directory, i.g. blockMesh

Time Directories

- Contains the field values and boundary conditions of specific time

Constant directory

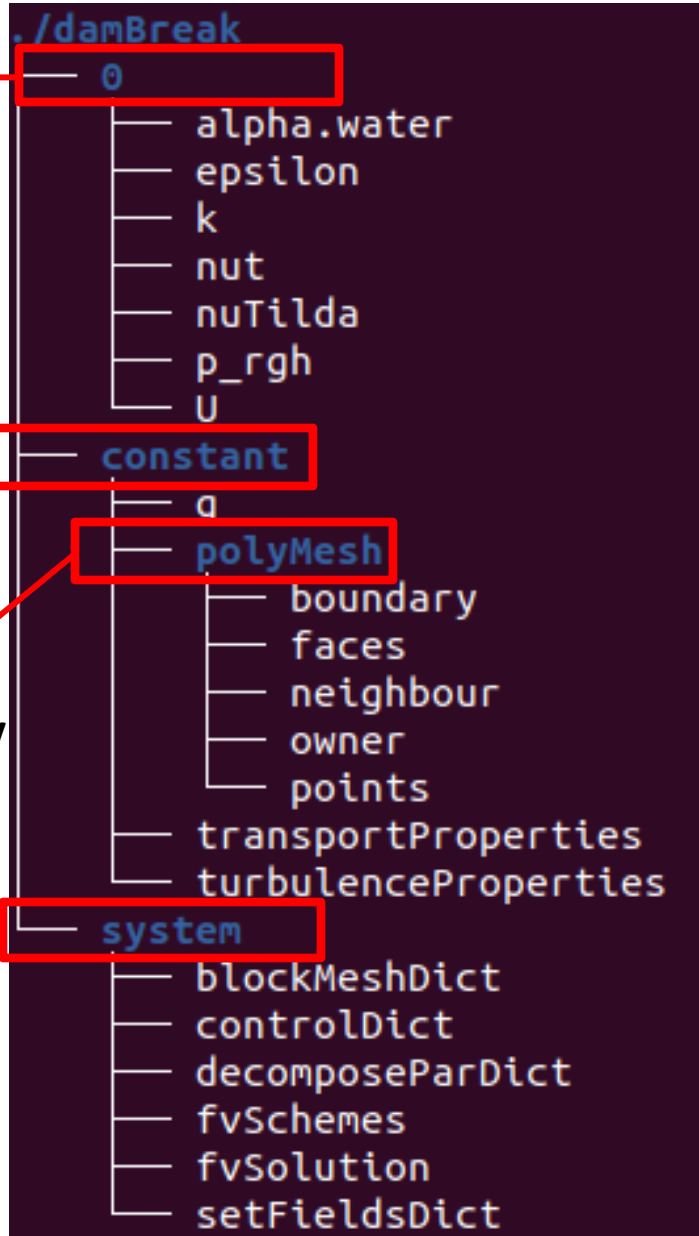
- Contains the physical properties and the mesh information

PolyMesh sub-directory

- Contains mesh info

System directory

- Mainly for defining the numerical schemes and their parameters.



Science and Technology Facilities Council

Scientific Computing

Numerics of OpenFOAM

Science and Technology Facilities Council

Scientific Computing

Governing Equation

- General conservation equation

$$\underbrace{\frac{\partial \rho\phi}{\partial t}}_{\text{Time derivative}} = -\underbrace{\nabla \cdot \rho\mathbf{u}\phi}_{\text{Convection}} + \underbrace{\nabla \cdot (D\nabla\phi)}_{\text{Diffusion}} + \underbrace{S_\phi}_{\text{Source term}}$$

- Momentum conservation equation

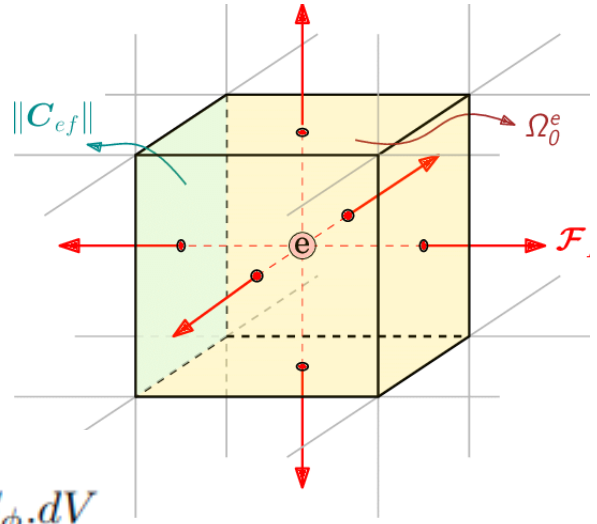
$$\frac{\partial \rho\mathbf{u}}{\partial t} = -\nabla \cdot \rho\mathbf{u}\mathbf{u} + \nabla \cdot (\rho\nu\nabla\mathbf{u}) + -\nabla p + S$$

- Mass Conservation Equation

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho\mathbf{u}$$

Finite Volume Method

- The domain is discretized into small cells, and the values of the cell is the integral of the function inside the cell.
- Insure flux conservation, and can be used for complex geometry and mesh refinement.
- **OF uses cell-centered finite volume method.**
- Integral form of the conservation equation:



$$\int_V \frac{\partial \rho\phi}{\partial t}.dV = -\int_V \nabla \cdot \rho\mathbf{u}\phi.dV + \int_V \nabla \cdot (D\nabla\phi).dV + \int_V S_\phi.dV$$

- Gauss's divergence theorem:

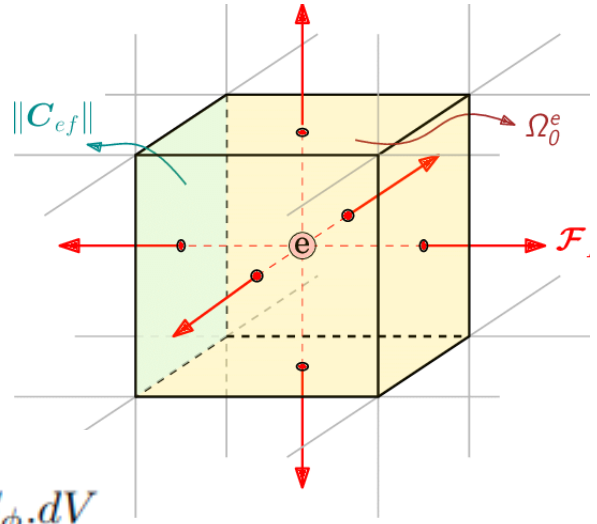
$$\int_V \nabla \cdot \mathbf{a}dV = \int_S \mathbf{a}dS$$

- Then the conservation equation cab be expressed as:

$$\int_V \frac{\partial \rho\phi}{\partial t}\mathrm{d}V = -\int_S \rho\mathbf{u}\phi\mathrm{d}S + \int_S (D\nabla\phi)\mathrm{d}S + \int_V S_\phi\mathrm{d}V$$

Finite Volume Method

- The domain is discretized into small cells, and the values of the cell is the integral of the function inside the cell.
- Insure flux conservation, and can be used for complex geometry and mesh refinement.
- **OF uses cell-centered finite volume method.**
- Integral form of the conservation equation:



$$\int_V \frac{\partial \rho\phi}{\partial t}.dV = -\int_V \nabla \cdot \rho\mathbf{u}\phi.dV + \int_V \nabla \cdot (D\nabla\phi).dV + \int_V S_\phi.dV$$

- Gauss's divergence theorem:

$$\int_V \nabla \cdot \mathbf{a}dV = \int_S \mathbf{a}dS$$

- Then the conservation equation cab be expressed as:

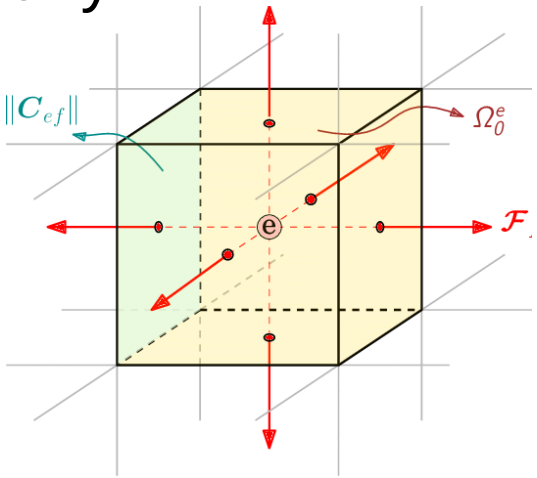
$$\int_V \frac{\partial \rho\phi}{\partial t}\mathrm{d}V = -\int_S \rho\mathbf{u}\phi\mathrm{d}S + \int_S (D\nabla\phi)\mathrm{d}S + \int_V S_\phi\mathrm{d}V$$

Finite Volume Method: Time Derivative

$$\int_V \frac{\partial \rho\phi}{\partial t} \mathrm{d}V = \frac{\partial}{\partial t} \int_V \rho\phi \mathrm{d}V$$

- In Second order discretisation we assume that ϕ changes linearly in space:

$$\phi(\mathbf{x}) = \phi_p + (\mathbf{x} - \mathbf{x}_p) \cdot \nabla\phi$$



- The volume integration of ϕ

$$\int_V \phi(\mathbf{x}) dV = \int_V \phi_p dV + \int_V (\mathbf{x} - \mathbf{x}_p) \cdot \nabla\phi_p dV$$

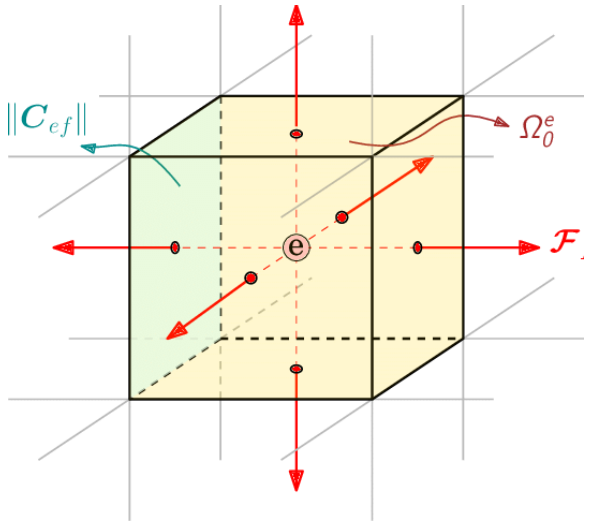
- Note: $\int_V (\mathbf{x} - \mathbf{x}_p) \cdot \nabla\phi_p dV = 0$ as x_p in the center of the cell and $\nabla\phi_p$ is constant

$$\int_V \phi(\mathbf{x}) dV = \phi_p V$$

$$\int_V \frac{\partial \rho\phi}{\partial t} \mathrm{d}V = \frac{\partial}{\partial t} \int_V \rho\phi \mathrm{d}V \rightarrow \frac{\partial}{\partial t} (\rho_p \phi_p) V$$

Finite Volume Method: Convection Term

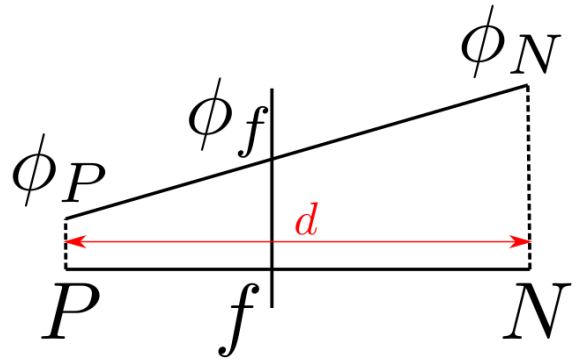
$$\int_V \nabla \cdot \rho \mathbf{u} \phi \, \mathrm{d}V = \int_S \rho \mathbf{u} \phi \, \mathrm{d}S = \sum_i S(\rho \mathbf{u} \phi)_f = \sum_i F \phi_f$$



- Where F is the flux, and i is the iterator over the cell,
- and $\boldsymbol{S}$ is surface area of the face
- The subscript `$f$` refers to the values on the cell face
- **Central Differencing (CD)**

$$\phi_f = \alpha \phi_P + (1 - \alpha)\phi_N$$

$$\alpha = \frac{\overline{fN}}{d}$$



  - Second order, unbounded, causes unphysical oscillations
- **Upwind differencing (UD)**

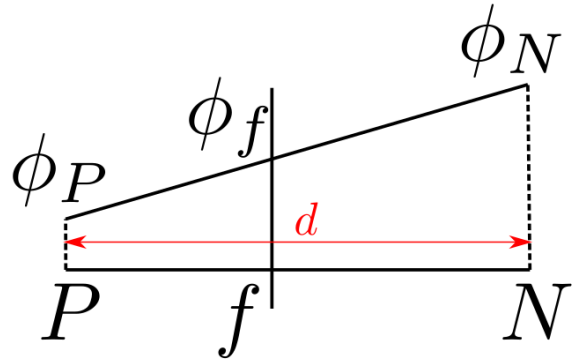
$$\phi_f = \begin{cases} \phi_P & , F \geq 0. \\ \phi_N & , F < 0. \end{cases}$$

  - First order, bounded faces

Finite Volume Method: Diffusion Term

$$\int_V \nabla \cdot (D\nabla\phi)\mathrm{d}V = \int_S D\nabla\phi \mathrm{d}S = \sum_i S \cdot (D\nabla\phi)_f$$

$$(\nabla\phi)_f = \begin{cases} \frac{\phi_N - \phi_P}{d}, & \text{For orthogonal grid.} \\ \alpha(\nabla\phi)_P + (1-\alpha)(\nabla\phi)_N, & \text{For non-orthogonal grid.} \end{cases}$$



- Both are second-order accurate but differ in boundness and suitability for non-orthogonal meshes

$$S \cdot (\nabla\phi)_f = \underbrace{\Delta \cdot (\nabla\phi)_f}_{\text{orthogonal contribution}} + \underbrace{\kappa \cdot (\nabla\phi)_f}_{\text{non-orthogonal contribution}}$$

$$S = \Delta + \kappa$$

$$\Delta = \begin{cases} \frac{d \cdot S}{d \cdot d} d & \text{Minimum correction approach.} \\ \frac{d}{|d|}|S| & \text{Orthogonal correction approach.} \\ \frac{d}{d \cdot S}|S|^2 & \text{Over-relaxed approach.} \end{cases}$$

Finite Volume Method: Usage in OF

- The gradient and divergence schemes are defined in *$case/system/fvSchemes*

```
gradSchemes
{
    default         none;
    grad(p)          <optional limiter> <gradient scheme> <interpolation scheme>;
    ......
    ......
}
```

It can be any scheme, i.e. *Gauss linear*

```
divSchemes
{
    default      none;
    div(phi,T)      Gauss <interpolation scheme>;
}
```

- There are different limiter, gradient and interpolations schemes.

https://www.openfoam.com/documentation/guides/v2112/doc/guide-schemes-gradient.html
https://www.openfoam.com/documentation/guides/v2112/doc/guide-schemes-divergence-example.html

Science and Technology Facilities Council

Scientific Computing

Numerics of OpenFOAM
Time discretisation

Science and Technology Facilities Council

Scientific Computing

Time Integration for Transient Flows

$$\int_t^{t+\Delta t} \left(\frac{\partial \rho \phi}{\partial t}\right)_P V_P \, \mathrm{d}t = \int_t^{t+\Delta t} \left[-\sum_f F\phi_f + \sum_f DS \cdot (\nabla \phi)_f + S_\phi V_P \right] \mathrm{d}t$$

- Backward time scheme
- Euler implicit time scheme
- Crank-Nicolson time scheme

- https://www.openfoam.com/documentation/guides/v2112/doc/guide-schemes-time.html

Time Integration: Explicit

$$\int_t^{t+\Delta t} \left(\frac{\partial \rho\phi}{\partial t}\right)_P V_P \, \mathrm{d}t = \int_t^{t+\Delta t} \left[-\sum_f F\phi_f + \sum_f DS \cdot (\nabla\phi)_f + S_\phi V_P \right] \mathrm{d}t$$

$$\left(\frac{\partial \rho\phi}{\partial t}\right)_P = \left(\frac{\rho^n\phi^n - \rho^o\phi^o}{dt}\right)_P$$

- The superscripts `*n*` and `*o*`, refer to values at new time `*t*+*dt*` and the previous time `*t*`.
- For clarity drop the density, i.e. consider incompressible flow

$$\left(\frac{\partial \phi}{\partial t}\right)_P = \left(\frac{\phi^n - \phi^o}{dt}\right)_P$$

- All the values are taken from the old time step

$$\phi_P^n = \phi_P^o + \frac{\Delta t}{V_P}\left[-\sum_f F^o\phi_f^o + \sum_f D^o S \cdot (\nabla\phi)_f^o + S_\phi^o V_P \right]$$

- First order, limited by the Courant number

$$Co = \frac{\mathbf{u}_f \cdot d}{\Delta t}$$

Time Integration: Euler Implicit

$$\int_t^{t+\Delta t} \left(\frac{\partial \rho \phi}{\partial t}\right)_P V_P \, \mathrm{d}t = \int_t^{t+\Delta t} \left[-\sum_f F\phi_f + \sum_f DS \cdot (\nabla\phi)_f + S_\phi V_P \right] \mathrm{d}t$$

$$\left(\frac{\partial \phi}{\partial t}\right)_P = \left(\frac{\phi^n - \phi^o}{dt}\right)_P$$

- All the values are expressed in terms of the new time step

$$\phi_f = \alpha \phi_P^n + (1-\alpha)\phi_N^n$$

$$S \cdot (\nabla\phi)_f = \Delta \cdot \frac{\phi_N^n - \phi_P^n}{d} + \kappa \cdot (\nabla\phi)_f^n$$

- The resulted equation in matrix form is

$$a_P \phi_P^n + \sum_N a_N \phi_N^n = R_P$$

- First order, Stable even for Co>1, Bounded

Time Integration: Backward Differencing

$$\int_t^{t+\Delta t} \left(\frac{\partial \rho \phi}{\partial t}\right)_P V_P \, \mathrm{d}t = \int_t^{t+\Delta t} \left[-\sum_f F\phi_f + \sum_f DS \cdot (\nabla \phi)_f + S_\phi V_P \right] \mathrm{d}t$$

- All the values on the RHS are expressed in terms of the new time step

$$\left(\frac{\partial \phi}{\partial t}\right)_P = \left(\frac{\phi^n - \phi^o}{dt}\right)_P \qquad \frac{\partial \phi}{\partial t} = \frac{1.5\phi^n - 2\phi^o + 0.5\phi^{oo}}{\Delta t}$$

- Need to solve a system of algebraic equations

- **Properties**: Implicit, Second order, Boundedness not guaranteed, Conditionally stable

Time Integration: Crank-Nicholson

$$\int_t^{t+\Delta t} \left(\frac{\partial \rho\phi}{\partial t}\right)_P V_P \, \mathrm{d}t = \int_t^{t+\Delta t} \left[-\sum_f F\phi_f + \sum_f DS \cdot (\nabla\phi)_f + S_\phi V_P\right] \mathrm{d}t$$

$$\left(\frac{\partial \phi}{\partial t}\right)_P = \left(\frac{\phi^n - \phi^o}{dt}\right)_P$$

- All the values are expressed in terms of the new and the old time steps

$$\int_t^{t+\Delta t} \phi dt = 0.5(\phi^n + \phi^o)\Delta t$$

$$\frac{\phi_P^n - \phi_P^o}{\Delta t} V_P = \frac{1}{2}\left[-\sum_f F^n \phi_f^n + \sum_f D^n S \cdot (\nabla\phi)_f^n + S_\phi^n V_P\right]$$
$$+ \frac{1}{2}\left[-\sum_f F^o \phi_f^o + \sum_f D^o S \cdot (\nabla\phi)_f^o + S_\phi^o V_P\right]$$

- Need to solve a system of algebraic equations
- **Properties:** Second order, Bounded

Time Integration: Usage

- The Time integration schemes are defined in *$case/system/fvSchemes*

```
ddtSchemes
{
    default         Euler;
   //default         CrankNicolson < a number \Psi =[0 1]>;
}
```

- Schemes list names
  - **Euler**: transient, first order implicit, bounded.
  - **backward**: transient, second order implicit, potentially unbounded.
  - **CrankNicolson**: transient, second order implicit, bounded.

$$\Psi = \begin{cases} 1 & \text{Corresponds to pure Crank-Nicolson} \\ 0 & \text{Corresponds to pure pure Euler} \end{cases}$$

  - https://www.openfoam.com/documentation/guides/v2112/doc/guide-schemes-time-example.html

Science and Technology Facilities Council

Scientific Computing

Numerics of OpenFOAM Pressure Velocity Coupling

Science and Technology Facilities Council

Scientific Computing

Pressure-Velocity Coupling

$$a_P U_P + \sum_N a_N U_N = R_P \quad , \text{or} \quad MU = R$$

- With 'N' being the eight

$$R = -\nabla p + S(U)$$

$$a_P U_P = -\nabla p + S(U_P) - \sum_N a_N U_N$$

$$a_P U_P = -\nabla p + H_N \quad , \text{or} \quad AU = H - \nabla p$$

- A is a diagonal matrix

$$U = A^{-1} H - A^{-1} \nabla p$$

- Substitute in the continuity equation $\nabla \cdot \mathbf{u} = 0$

$$\nabla \cdot A^{-1} \nabla p = \nabla \cdot (A^{-1} H)$$

Pressure velocity Coupling

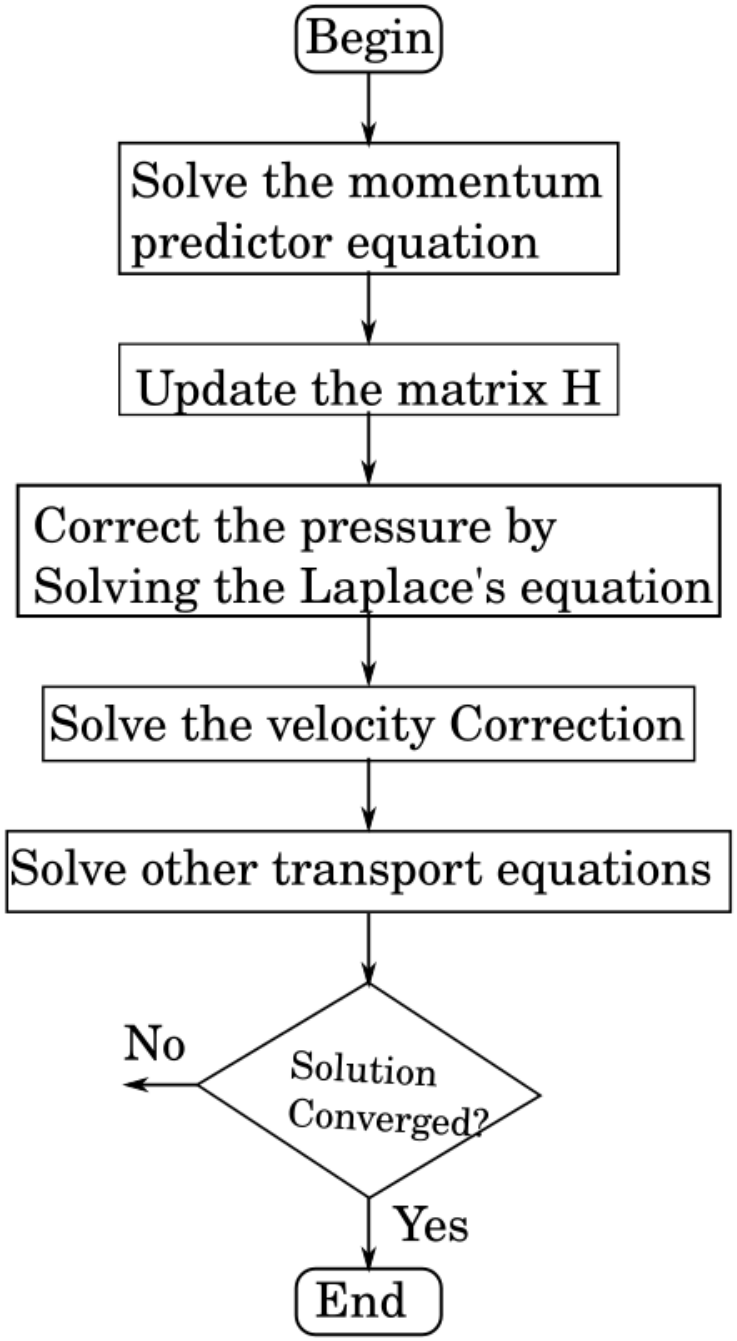
- Summary of the equation to be solved:

$$MU = -\nabla p - S(U) \quad \text{Momentum predictor}$$

$$H = AU - MU$$

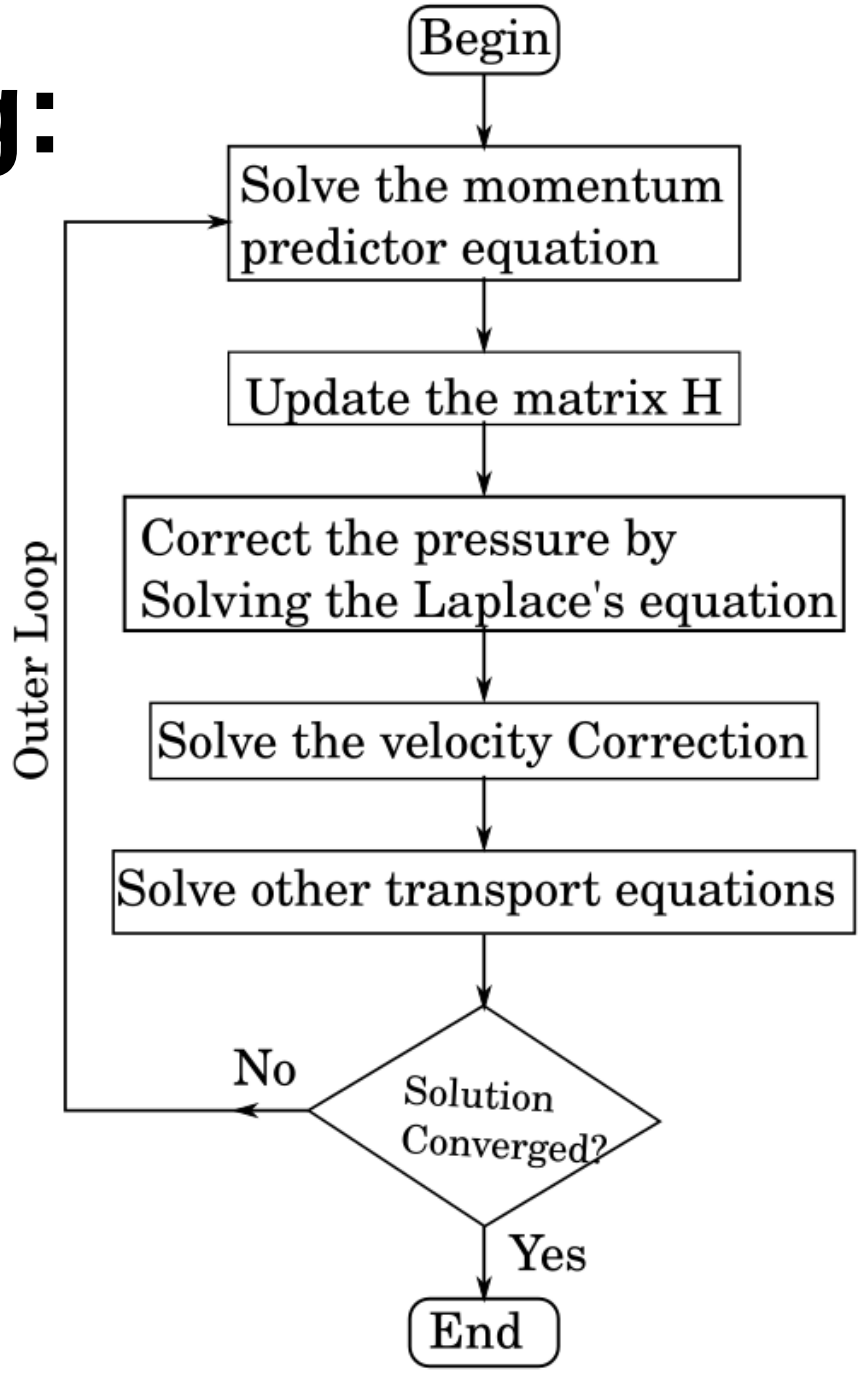
$$\nabla \cdot A^{-1}\nabla p = \nabla \cdot (A^{-1}H) \quad \text{Laplace's equation of the pressure}$$

$$U = A^{-1}H - A^{-1}\nabla p \quad \text{Velocity Corrector}$$



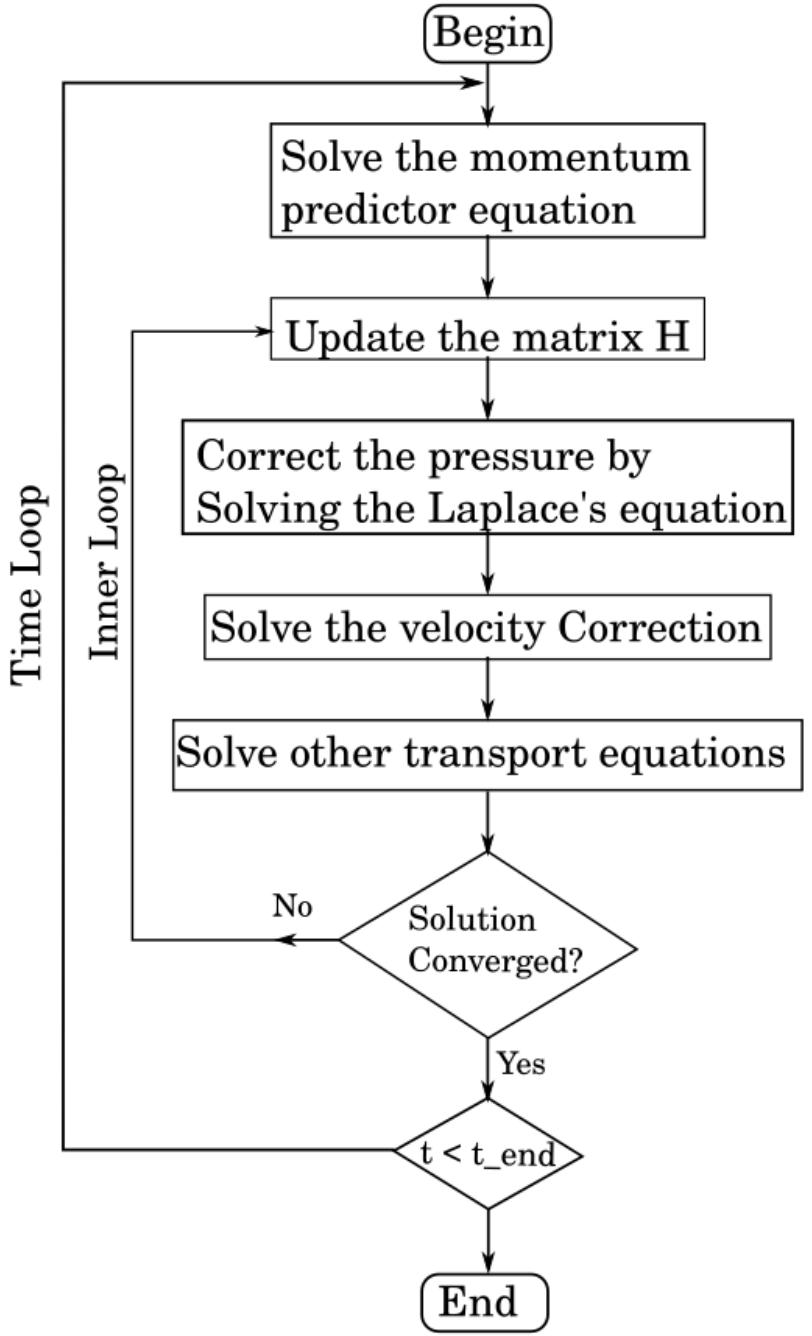
Pressure velocity Coupling: SIMPLE

- Semi-Implicit-Method-Of-Pressure-Linked-Equations.
- Used for Steady State simulations, but needs under-relaxation.
- Stable for large Co.
- https://www.openfoam.com/documentation/guides/v2112/doc/guide-applications-solvers-simple.html



Pressure velocity Coupling PISO

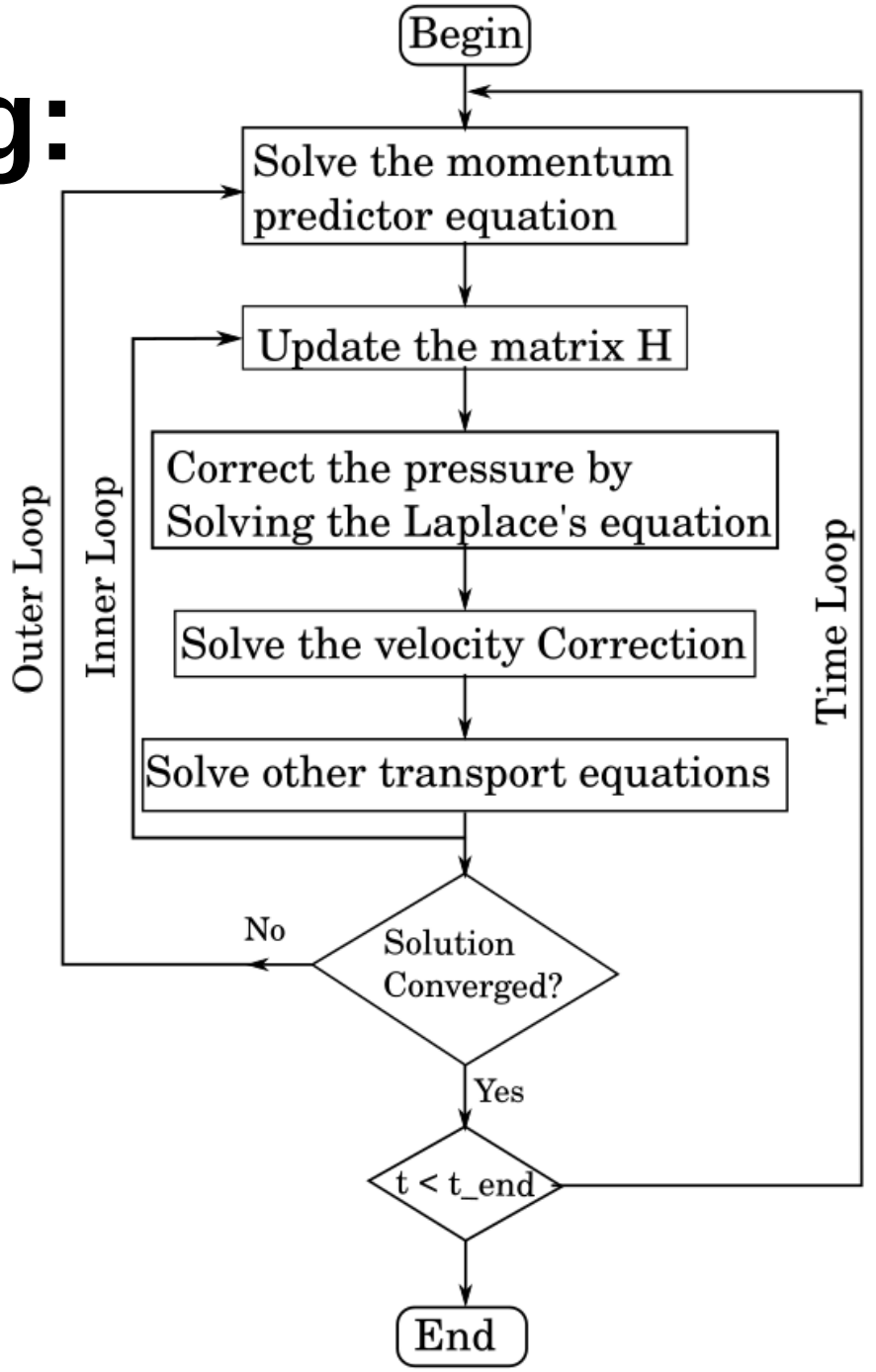
- Pressure Implicit with Splitting of Operators
- Unsteady simulation
- The momentum equation is solved once each time step
- The correction is repeated via the inner loop till convergence in each time step



- Issa, Raad I. "Solution of the implicitly discretised fluid flow equations by operator-splitting." Journal of computational physics 62.1 (1986): 40-65.

Pressure velocity Coupling: PIMPLE

- Combines SIMPLE and PISO algorithms
- Unsteady simulation
- The momentum equation can be solved multiple times time step (outer loop)
- The correction is repeated via the inner loop
- Allows large *Co*



Pressure velocity Coupling: Usage

- Defined in *$case/system/fvSolution*
- https://openfoamwiki.net/index.php/OpenFOAM_guide/The_PIMPLE_algorithm_in_OpenFOAM
- Monitor the residuals
  - *foamLog <logFile>*
  - In *$case/system/controlDict*, add residuals function

```
functions
{

    residuals
    {
        type                solverInfo;
        libs                (utilityFunctionObjects);
        writeResidualFields true;
        writeControl        writeTime;
        fields              (".*"); // (U, P)
    }
}
```

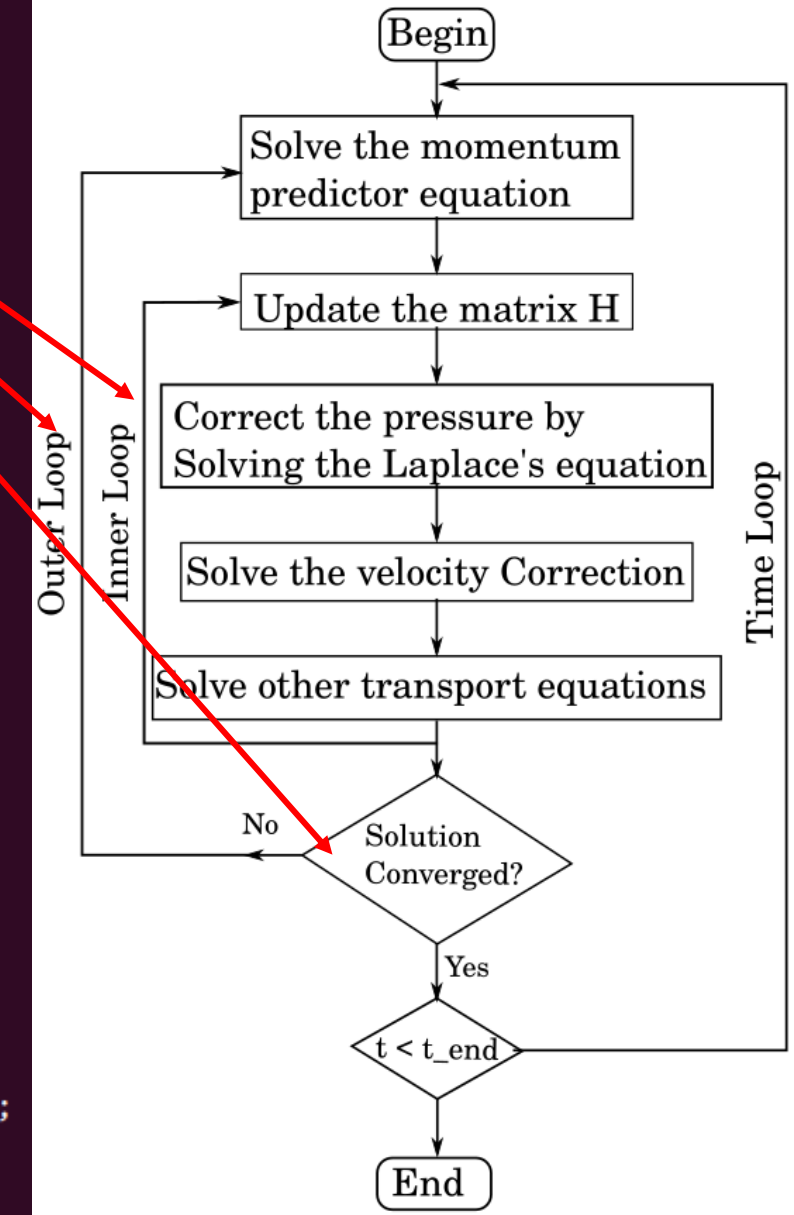
Get the residuals for all the available field

Get the residuals for specific fields

```
PIMPLE
{
    nCorrectors          5;
    nOuterCorrectors    50;

    residualControl
    {
        U
        {
            tolerance  1e-5;
            relTol      0;
        }
        p
        {
            tolerance  5e-4;
            relTol      0;
        }
    }
}

relaxationFactors
{
    fields
    {
        p      0.3;
        pFinal   1;
    }
    equations
    {
        "U|k|epsilon"      0.3;
        "(U|k|epsilon)Final"   1;
    }
}
```



Science and Technology Facilities Council

Scientific Computing

Numerics of OpenFOAM Volume of Fluid

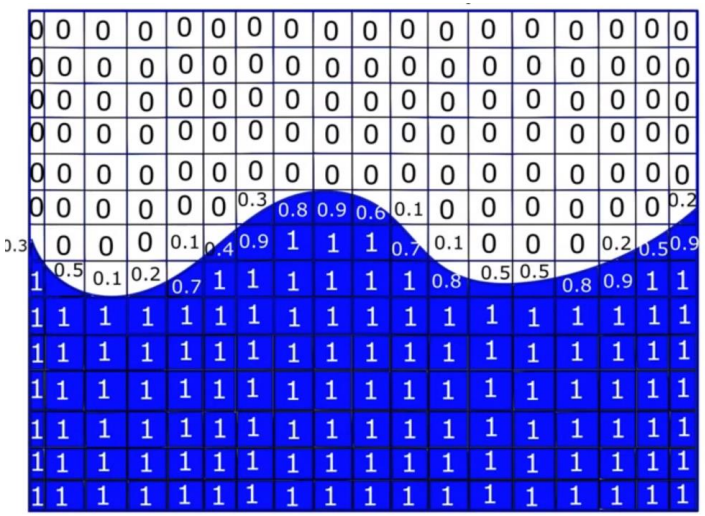
Science and Technology Facilities Council

Scientific Computing

Volume of Fluid (VOF)

- Interface capturing methods use a fixed grid
- Uses a labelling function for each cell having value [0 1]

$$\alpha = \frac{V_1}{V} = \begin{cases} 1 & \text{Fluid 1} \\ 0 < \alpha < 1 & \text{Interface Cells} \\ 0 & \text{Fluid 2} \end{cases}$$



Source: https://www.youtube.com/watch?v=7W8JqP1Le3I&ab_channel=AppliedComputationalFluidDynamics

- Assume that the local properties changes linearly with the volume fraction

$$\rho = \alpha\rho_1 + (1-\alpha)\rho_2$$

$$\mu = \alpha\mu_1 + (1-\alpha)\mu_2$$

- The governing equations can be used with no change
  - Note: Source term to account for the surface tension can be added

Volume of Fluid (VOF)

- There are two categories[1]:
  - **Algebraic Method** schemes are typically much simpler to implement, more efficient and are not restricted to structured meshes but not as accurate.
  - **Geometric methods** involving an explicit reconstruction of the interface from the volume fraction data. They usually involve complex geometric operation making their implementation cumbersome.

- OpenFOAM ESI (V1706 and later) has these two methods built-in
  - Algebraic Method using *MULES* scheme
  - Geometric Method using *isoAdvector*

- Note: isoAdvector is available as Third party solver for other versions of OpenFOAM
  - https://github.com/isoAdvector

[1] http://dx.doi.org/10.17196/OS_CFD#YEAR_2017

Volume of Fluid (VOF): Algebraic Method

- To track the interface, a transport equation of the volume fraction is needed

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u}\phi) = 0$$

- The diffusion is high leading to surface smearing

Volume of Fluid (VOF): Algebraic Method

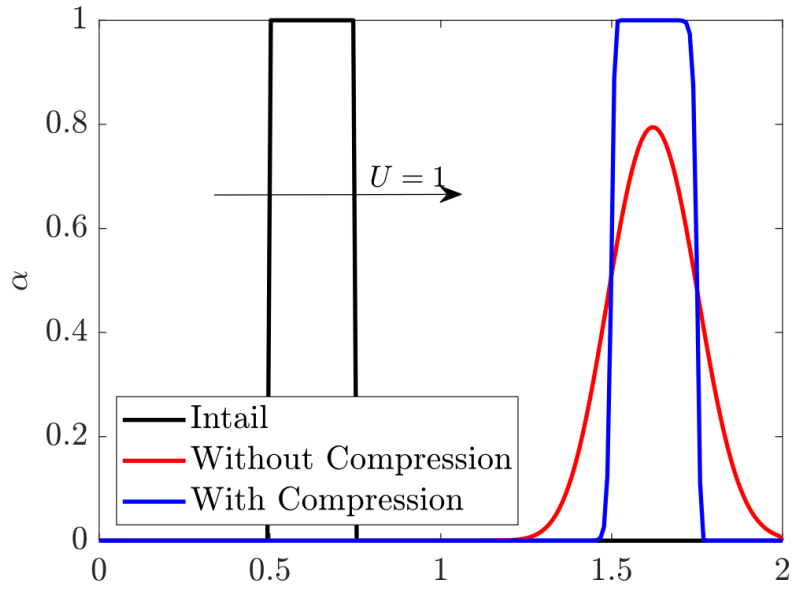
- To track the interface, a transport equation of the volume fraction is needed

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u}\phi) = 0$$

- The diffusion is high leading to surface smearing
- To reduce surface smearing, a compression term is added:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\mathbf{u}\phi) + \nabla \cdot (U_C \quad \alpha(1-\alpha)) = 0$$

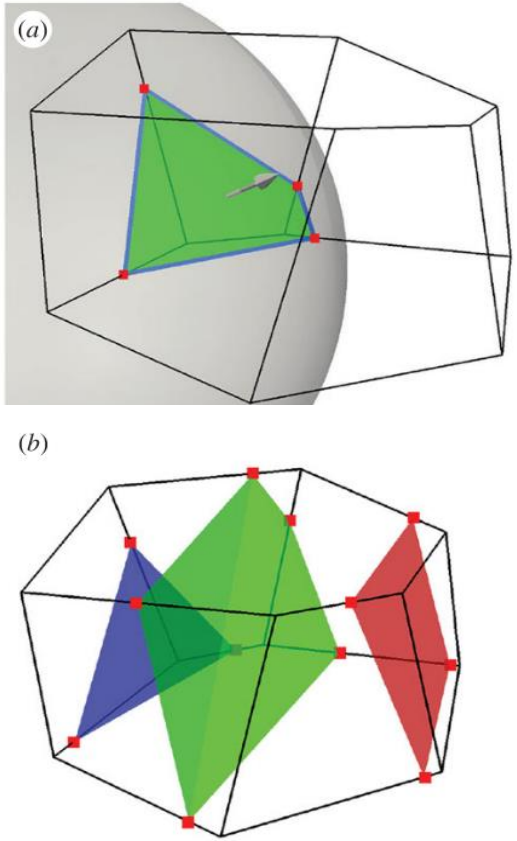
$$U_C = C_\alpha |\mathbf{u}| \frac{\nabla \alpha}{|\nabla \alpha|}$$



- Where C_α is user specific, 0 for no compression or 1 for strong compression
- Note: The actual implementation in OF uses two schemes
  - Upwind differencing away from the interface (reduce the computational effort)
  - High order scheme at the interface
- Available in *interFoam*

Volume of Fluid (VOF): isoAdvector Method

- **Geometric method**
  - Reconstruct the surface of the interface using the isosurface concept of the volume fraction
  - Model the motion of the constructed surface
- Use the solver *interIsoFoam*



- Olsson, E.: A description of isoAdvector - a numerical method for improved surface sharpness in two-phase flows. In Proceedings of CFD with Open Source Software, 2017,
- Roenby J, Bredmose H, Jasak H. A computational method for sharp interface advection. Royal Society open science. 2016 Nov 23;3(11):160405.

Important resources

- OpenFOAM Guides
  - https://www.openfoam.com/documentation/overview
    - **User Guide**: Gain understanding of how OpenFOAM cases are assembled and evaluated in the OpenFOAM user guide:
    - **Tutorial Guide**: A collection of tutorials to help users get started with OpenFOAM covering a range of topics, including incompressible, compressible and multiphase flows, and stress analysis
    - **Extended Code Guide**: see how OpenFOAM operates under-the-hood. As an open source code, users can directly see how the code is written and learn how the functionality is implemented.
- Finite volume and time discretisation
  - https://spiral.imperial.ac.uk/bitstream/10044/1/8335/1/Hrvoje_Jasak-1996-PhD-Thesis.pdf
- Numerics of OpenFOAM
  - https://www.researchgate.net/publication/307546712
- VOF
  - http://dx.doi.org/10.17196/OS_CFD#YEAR_2017

UKRI Science and Technology Facilities Council

Scientific Computing

Questions?

UKRI Science and Technology Facilities Council

Scientific Computing

Thank you

scd.stfc.ac.uk @SciComp_STFC STFC Scientific Computing Omar-ahmed.mahfoze@stfc.ac.uk