How to Run Cases in OpenFOAM® - interFoam

Rita Fernandes de Carvalho
ritalmfc@dec.uc.pt

LHRHA – Laboratório de Hidráulica, Recursos Hídricos e Ambiente
DEC-UC – Departamento de Engenharia Civil - Universidade de Coimbra
MARE – Marine and Environmental Sciences Centre
ARNET – Aquatic Research Network

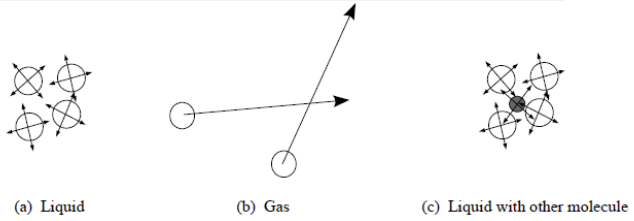
Contents

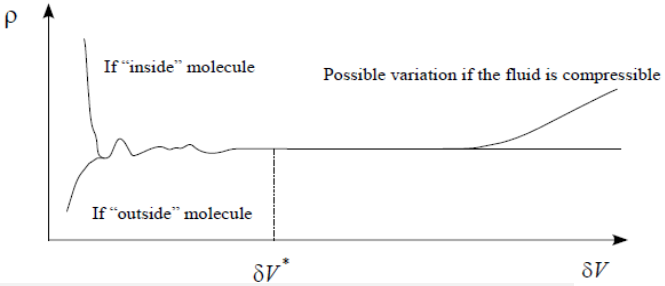
a bit of theory

Goals

- Understanding of **mathematical principles** related to numerical modelling of physical problems formulated in terms of partial derivative equations – turbulent free-surface flows - interFoam / VOF Method
- Use and understanding of **computational modelling**

Continuity





John D. Fenton. Hydraulics: science, knowledge, and culture. Journal of Hydraulic Research, 54(5):485-501, 2016

Matter is, in reality, an aggregate of molecules more or less spaced apart
➔ discontinuity
the definition of a quantity only makes sense, considering an average value in a certain region ➔ **continuous medium**

characteristic dimension of the molecular scale $L_M = 10^{-10}$ m
characteristic dimension of the macroscopic mechanical scale:

$$L_L = 10^{-4} \text{ to } 10^{-2} \text{ m}$$

Material particle is an elementary quantity (which comprises millions of molecules) and has dimensions, L_P, such that:

$$L_M << L_P << L_L$$

In the **Mechanics of Continuous Media**, the material is studied based on particle size (control volume),neglecting all dimension discontinuities and assuming the **continuum hypothesis**.

a bit of theory

Fluid Equations

OpenFOAM® a PDE library

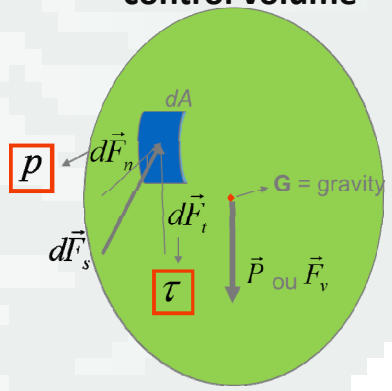
PDE – unknown function depends on several variables (x,y,z,t) - the equation can reflect an evolution phenomenon.

- ➔ relate **unknown quantities as functions of variables** and contain these functions and their total or partial derivatives (partial derivatives in space and time)
- ➔ in the case of fluid mechanics: velocity, acceleration, flow rate, intensity, flow height or depth, pressure, VoF, Concentration, Energy, Dissipation… - **generally a function of space (x, y, z), and time (t).**
  - ➔ Forces applied to the fluid
  - ➔ Stresses created in the fluid
  - ➔ Displacements and deformations

Stress theory
Behavior rheological

$$\sum_j \vec{F}_{ext} = \sum_j \vec{F}_{Sup} + \sum_j \vec{F}_{Vol}$$

control volume



N-S equations + …

continuity $\nabla \cdot \rho\bar{v} = 0$

momentum $\frac{\partial \rho\bar{v}}{\partial t} + \nabla \cdot (\rho\bar{v}\,\bar{v}) = -\nabla p^* - g \cdot x\nabla\rho + \nabla \cdot \tau + F$

VOF $\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha\bar{v}) + \nabla \cdot [v_c\alpha(1-\alpha)] = 0$

….

a bit of theory

Mathematical Modelling

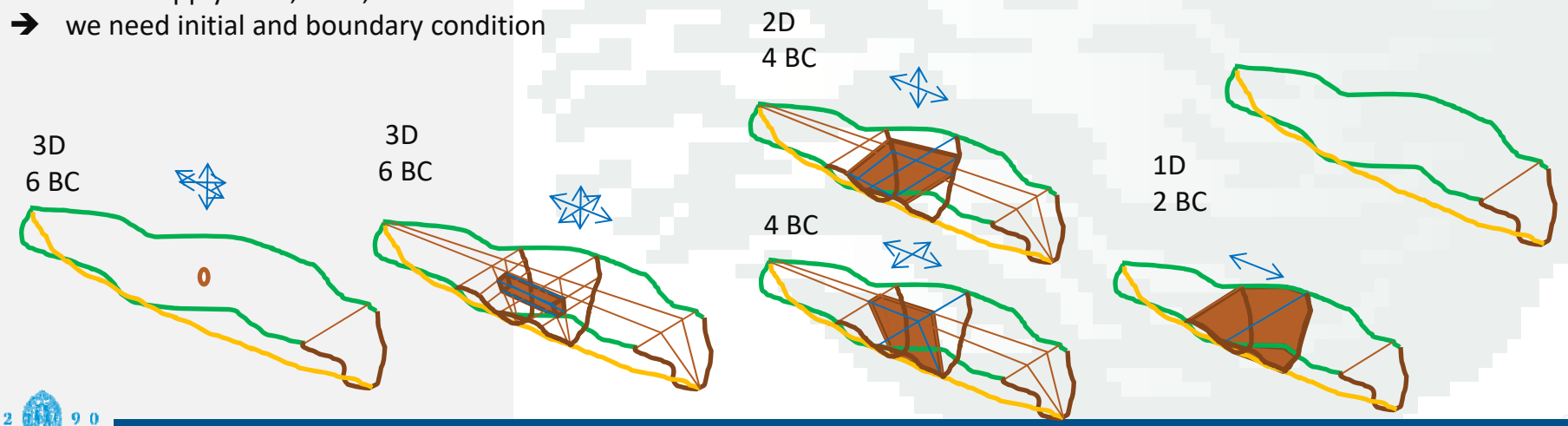
- ➔ Domain
- ➔ Mesh
- ➔ Boundary conditions

$$\nabla \cdot \rho\bar{\mathrm{v}} = 0$$

$$\frac{\partial \rho\bar{\mathrm{v}}}{\partial \mathrm{t}} + \nabla \cdot (\rho\bar{\mathrm{v}}\,\bar{\mathrm{v}}) = -\nabla p^* - g \cdot x\nabla\rho + \nabla \cdot \tau + F$$

$$\frac{\partial \alpha}{\partial \mathrm{t}} + \nabla \cdot (\alpha\bar{\mathrm{v}}) + \nabla \cdot [\mathrm{v}_c\alpha(1-\alpha)] = 0$$

- ➔ we replace the real molecular structure with a hypothetical continuous medium, divided into volumes with average properties of the molecules contained – “point”/control volume/finite element.
- ➔ we can apply FDM, **FVM**, FEM
- ➔ we need initial and boundary condition



a bit of theory

➔ How to deal with **free-surface** ?

- Fixed or mobile mesh
- Surface or volume methods

M. Surface
- interface marked by dots
- position between pts calculated by interpolation
- advection of points

M. Volume
- the entire volume is **marked by a function**
- calculation of the free surface position is not explicit (defined in a function)

Free-surface modelling

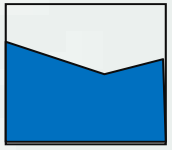
Volume of Fluid- F = *f* (*x*,*y*,*z*,*t*)

(Hirt and Nichols,1981)

- Empty cell-
  - alpha = 0

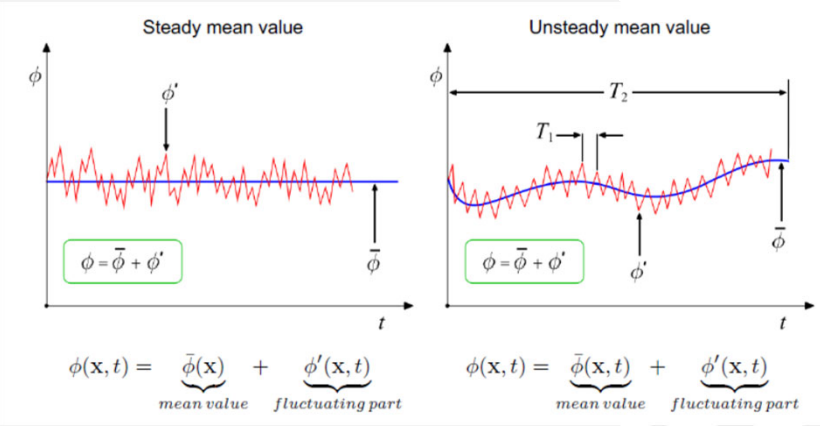
- Full cell-
  - alpha = 1

- Surface cell
  - 0 < alpha < 1



a bit of theory

Turbulence



- ➔ statistics at a point's time, at different times (**time scale**) at different points (**length scale**);
- ➔ **measurements/analysis** should be performed at high frequency to **include small eddies** and performed over a **period significantly longer** than the time characteristic of larger turbulence structures.
- ➔ fluid dynamics theory and CFD tools to be able to **understand the models** (eventually improve them) to **critically analyse results** and to better **evaluate energy dissipation**

$$\bar{u} = \frac{1}{n}\sum_{i=1}^{n} u\ ;\ \bar{v}, \bar{w},$$
$$u' = u - \bar{u}\ ;\ v', w', \phi'$$
$$var(u) = \overline{u'^2} = \frac{1}{n}\sum_{i=1}^{n} u'^2$$
$$cov(u,v) = \overline{u'v'} = \frac{1}{n}\sum_{i=1}^{n} u'v'$$

$$assi = \frac{\overline{u'^3}}{\sqrt{\overline{u'^2}^3}}, \qquad kur = \frac{\overline{u'^4}}{\overline{u'^2}^2}$$

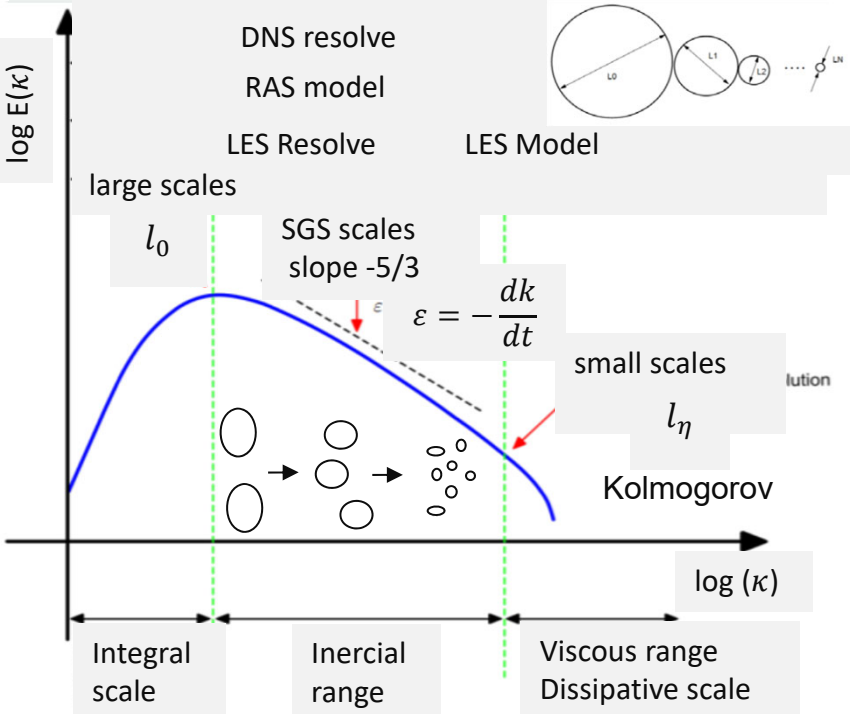
$$\rho(\tau) = \frac{\overline{u(t).u(t+\tau)}}{\overline{u^2(t)}}$$

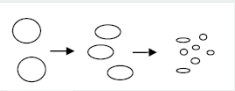
$$R_{ij}(r) = \overline{u(x).u(x+r)}$$

a bit of theory

(Flow instabilities) – seems a random movement - molecular origin - nature of constituents - vortices, fluctuation, unpredictable mixing - Extra transport of mass, momentum and energy by diffusion

Turbulence





Turbulence

- According to Kolmogorov's universal equilibrium theory, the **motion at the smallest scales** (at which the energy is dissipated) should depend only upon:
  - The rate at which the **larger eddies supply energy**, $\varepsilon = -\frac{dk}{dt}$, $\varepsilon = 2\nu\overline{s_{ij}s_{ij}}$, $s_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_j}\right)$
  - The **kinematic viscosity** v

Micro-scales $\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}$, $\tau_\eta = \left(\frac{\nu}{\varepsilon}\right)^{1/2}$, $u_\eta = (\nu\varepsilon)^{1/4}$, $Re_\eta = \frac{\eta u_\eta}{\nu}$=1 (Kolmogorov Reynolds number)

Large-scales $l_0 = \frac{k^{3/2}}{\varepsilon}$, $\tau_0 = \frac{k}{\varepsilon}$, $Re_T = \frac{k^{1/2} l_0}{\nu}$

- for large Reynolds number the length, time, and velocity scales of the **smallest eddies are small compared to those of the largest eddies**

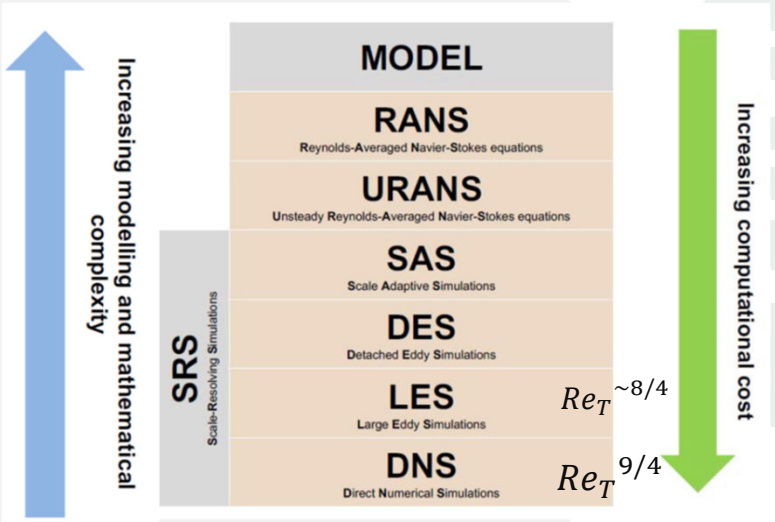
Largest eddies → $\frac{l_0}{\eta} \sim Re_T^{3/4}$ $\quad \frac{\tau_0}{\tau_\eta} \sim Re_T^{1/2}$ $\quad \frac{u_0}{u_\eta} \sim Re_T^{1/4}$

Smallest eddies →

a bit of theory

Turbulence Models

➔ translate the effects of instantaneous fields over time into equations, without directly calculating the complete turbulent flow as a function of time. ➔ remove small scales



"sufficiently resolved in LES/DES, or capture in RANS/URANS"

Reynolds ensemble averaging (RANS/URANS):

- 2D / 3D
- steady/ unsteady.

$$\tau^R = -\rho\left(\overline{\mathbf{u}'\mathbf{u}'}\right) = -\begin{pmatrix} \rho\overline{u'u'} & \rho\overline{u'v'} & \rho\overline{u'w'} \\ \rho\overline{v'u'} & \rho\overline{v'v'} & \rho\overline{v'w'} \\ \rho\overline{w'u'} & \rho\overline{w'v'} & \rho\overline{w'w'} \end{pmatrix}$$

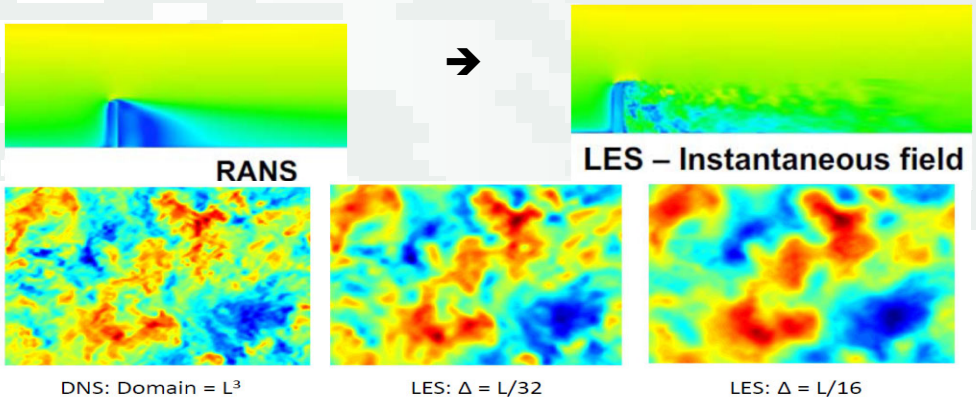
Filtering (LES/DES):

- calculate large eddies.
- model small eddies

  SGS – Sub Grid Scales Models
- always 3D unsteady.

$$l_0/\Delta$$

$$\Delta = \sqrt[3]{V_{célula}}$$



Numerical methods - Finite Volume method

Consider a simple, second-order equation, corresponding to the Poisson equation in 1D:

$$\frac{\partial^2 T}{\partial x^2} = f(x)$$

$$\int_{V_i} \frac{\partial^2 T}{\partial x^2} dx = \int_{V_i} f(x) dx$$

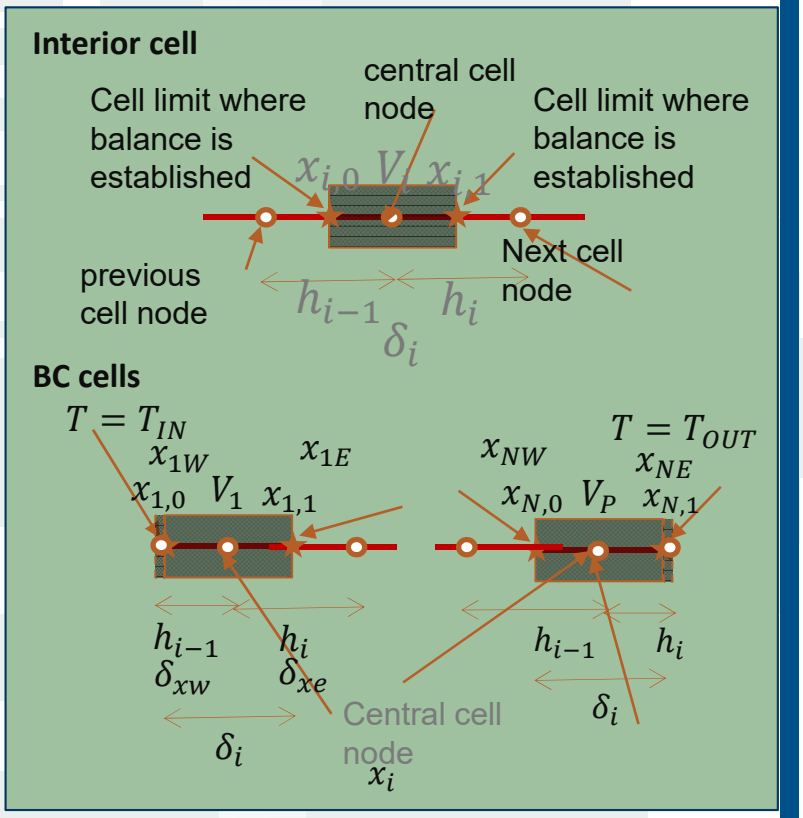
$$\left[\frac{\partial T}{\partial x}\right]_{x_{i,0}}^{x_{i,1}} = \int_{V_i} f(x) dx$$

$$\frac{T_{i+1} - T_i}{h_i} - \frac{T_i - T_{i-1}}{h_{i-1}} = f(x_i)\delta_i$$

$$\frac{T_{i+1} - T_i}{h_i} - \frac{\boldsymbol{T_i - T_{IN}}}{\boldsymbol{\delta x_i/2}} = f(x_i)\delta x_i$$

$$\frac{\boldsymbol{T_{OUT} - T_i}}{\boldsymbol{\delta x_i/2}} - \frac{T_i - T_{i-1}}{h_{i-1}} = f(x_i)\delta x_i$$

Approximation Method - fvscheme

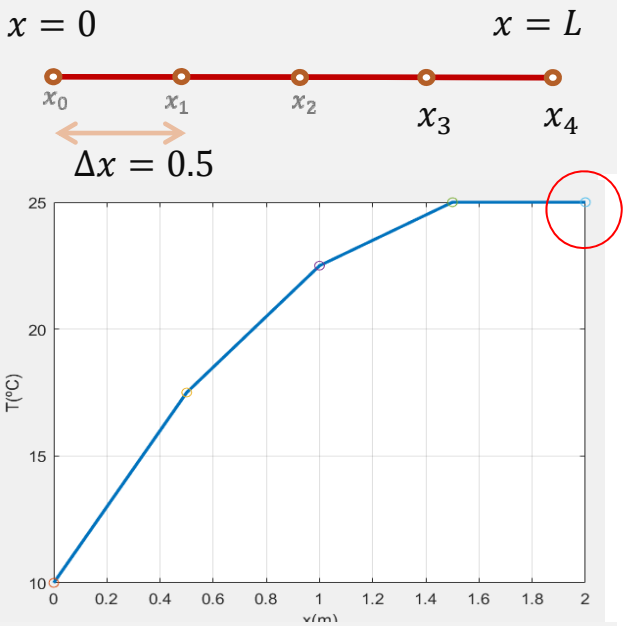


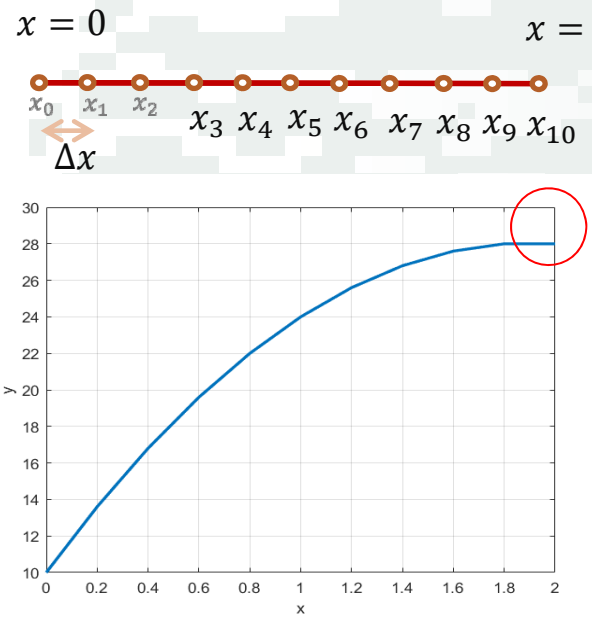
Numerical methods - Finite Volume method

Importance of the mesh - Example:

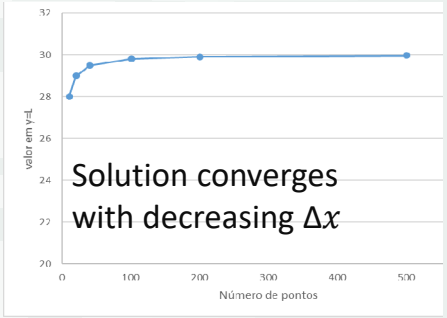
Equation $\frac{\partial^2 T}{\partial x^2} = 2.0.$ **domain:** in $x \in [0\ L]$

with BC: $y(x=0) = 10.0\,,\quad \frac{\partial y}{\partial x}(x=L) = 0.0$





Solving the same problem considering successively smaller intervals Δx



➔Data ➔ Mesh ➔ properties +BC ➔Matrix construction with BC+interior points of the domain ➔ Matrix resolution ➔ Mesh study

2

Solver and Turbulence models

Goals

- Understanding and knowing how to choose options to model a case, accordingly the pretended **mathematical principles**

Solver and Turbulence models

`interFoam – laminar`

(https://www.openfoam.com/documentation/guides/latest/api/interFoam_8C_source.html)

Solver for **two incompressible, isothermal immiscible fluids** using a **VOF (volume of fluid)** phase-fraction based **interface capturing approach**, with optional mesh motion and mesh topology changes including adaptive re-meshing

$$\nabla \cdot \rho\bar{\mathrm{v}} = 0$$

$$\frac{\partial \rho\bar{\mathrm{v}}}{\partial \mathrm{t}} + \nabla \cdot (\rho\bar{\mathrm{v}}\,\bar{\mathrm{v}}) = -\nabla p^* - g \cdot x\nabla\rho + \nabla \cdot \tau + F$$

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{t}} + \boldsymbol{\nabla \cdot (\alpha \bar{v})} + \boldsymbol{\nabla \cdot [v_c \alpha (1-\alpha)]} = \mathbf{0}$$

```
Description
    Solver for two incompressible, isothermal immiscible fluids using a VOF
```

In OpenFOAM®, complex concepts can be written in a familiar fashion example, systems of equations are implemented using a syntax that follows the mathematical notation, e.g.:

Time rate of change	$\frac{\partial}{\partial t}(\phi)$	`fvc::ddt(phi)`
Gradient	$\nabla\phi$	`fvc::grad(phi)`
Divergence	$\nabla \cdot \phi$	`fvc::div(phi)`
Laplacian	$\nabla^2\phi$	`fvc::laplacian(phi)`
Linearised sources	$s\phi$	`fvc::Sp(s,phi)`

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Solver and Turbulence models

Mathematical Modelling – VOF/Turbulence

➔ interFoam (laminar/RANS)

$$\nabla \cdot \rho\bar{\mathrm{v}} = 0$$

$$\frac{\partial \rho\bar{\mathrm{v}}}{\partial \mathrm{t}} + \nabla \cdot (\rho\bar{\mathrm{v}}\,\bar{\mathrm{v}}) = -\nabla p^{*} - g \cdot x\nabla\rho + \nabla \cdot \tau + F$$

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{t}} + \boldsymbol{\nabla \cdot (\alpha \bar{v})} + \boldsymbol{\nabla \cdot [v_c \alpha (1-\alpha)]} = \mathbf{0}$$

$$\frac{\partial \rho k}{\partial \mathrm{t}} + \nabla \cdot (\rho k\bar{\mathbf{v}}) = \nabla \cdot (\Gamma_k \nabla k) + P_k - Y_k$$

$$\frac{\partial \rho \varepsilon}{\partial \mathrm{t}} + \nabla \cdot (\rho \varepsilon\bar{\mathbf{v}}) = \nabla \cdot (\Gamma_\varepsilon \nabla \varepsilon) + P_\varepsilon - Y_\varepsilon + D_\varepsilon$$

$$\frac{\partial \rho \omega}{\partial \mathrm{t}} + \nabla \cdot (\rho \omega\bar{\mathbf{v}}) = \nabla \cdot (\Gamma_\omega \nabla \omega) + P_\omega - Y_\omega + D_\omega$$

Reynolds Average Navier Stokes – RANS/URANS ➔ **model of n-equations** represents a model that requires the solution of **n additional differential transport equations** to solve Reynolds stresses or tries to describe the effect of the viscous and Reynolds stresses considering **the turbulent viscosity** (Boussinesq hypotesis).

- ***k - ε***
- ***k – ω***
- ***k – ω SST***

Divergent
vector ➔ scalar

$$\nabla \cdot \mathrm{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}$$

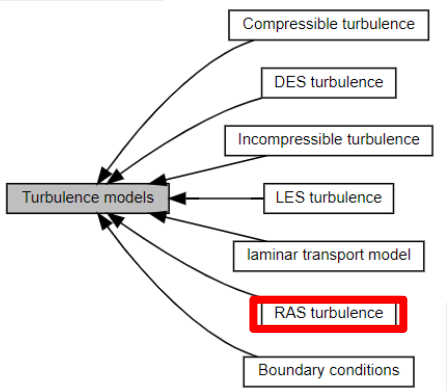
Gradient
scalar ➔ vector

$$\nabla p = \frac{\partial p}{\partial x}\hat{\imath} + \frac{\partial p}{\partial y}\hat{\jmath} + \frac{\partial p}{\partial z}\hat{k}$$

Laplaciano
vector ➔ vector

$$\nabla^2 \mathrm{v} = \frac{\partial^2 u}{\partial x^2}\hat{\imath} + \frac{\partial^2 v}{\partial y^2}\hat{\jmath} + \frac{\partial^2 w}{\partial z^2}\hat{k}$$

Solver and Turbulence models



`$WM_PROJECT_DIR/src/turbulenceModels/RAS`

Options

OpenFOAM includes Reynolds Averaged Simulation turbulence closures

- **Linear eddy viscosity models**
- **Non-linear eddy viscosity models**
- **Reynolds stress transport models**

Usage

RAS is selected by setting the `simulationType` entry

```
simulationType  RAS;

RAS
{
    \\ Model input parameters
}
```

Suitable for:

- 1-D, 2-D and 3-D cases
- steady-state or transient

Linear eddy viscosity turbulence model selections include:

- **k-epsilon**
- **k-epsilon-phit-f**
- **k-kl-omega**
- **Langtry-Menter k-omega Shear Stress Transport**
- **k-omega Shear Stress Transport (SST)**
- **Lien-Leschziner**
- **q-zeta**
- **Realizable k-epsilon**
- **RNG k-epsilon**
- **Spalart-Allmaras**
- **v2-f**

Non-linear eddy viscosity turbulence model selections include:

- **Lien cubic k-epsilon**
- **Shih quadratic k-epsilon**

Reynolds stress transport turbulence model selections include:

- **Launder, Reece and Rodi (LRR)**
- **Speziale, Sarkar and Gatski (SSG)**

Solver and Turbulence models

➔ interFoam (laminar/LES)

$\nabla \cdot \rho\bar{v} = 0$

$$\frac{\partial \rho\bar{v}}{\partial t} + \nabla \cdot (\rho\bar{v}\,\bar{v}) = -\nabla p^* - g \cdot x\nabla\rho + \nabla \cdot \tau + F$$

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{t}} + \boldsymbol{\nabla} \cdot (\boldsymbol{\alpha}\bar{\boldsymbol{v}}) + \boldsymbol{\nabla} \cdot [\boldsymbol{v_c}\boldsymbol{\alpha}(\mathbf{1}-\boldsymbol{\alpha})] = \mathbf{0}$$

$$\tau = 2\boldsymbol{\nu_{SGS}}\bar{S}_{ij}{}^*$$

$$\bar{S}_{ij}{}^* = 0.5\left(\frac{\partial \bar{v}_i}{\partial x_j} + \frac{\partial \bar{v}_j}{\partial x_i}\right)$$

turbulence viscosity is given by:

$$v_{SGS} = (C_s\bar{\Delta})^2[\bar{S}]$$

C_s – Smagorinsky coefficient,

$\bar{\Delta}$ – is the cut-off length which is set to:

1. the cube root of the cell volume

$$\Delta = \sqrt[3]{V_{célula}}$$

2. Van Driest damping coefficient

$$\Delta = \min\left(\left(1 - e^{-y^+/A^+}\right)\kappa y/C_s, \Delta_g\right)$$

Δ_g is a geometric-based delta function such as the cube root delta, κ=0.41, A^+=26

3. K equation / dynamic K equation

$$\frac{D}{Dt}(\rho k) = \nabla \cdot (\rho D_k \nabla k) + \rho G - \frac{2}{3}\rho k \nabla \cdot \mathbf{u} - \frac{C_e \rho k^{1.5}}{\Delta} + S_k$$

Solver and Turbulence models

`$WM_PROJECT_DIR/src/turbulenceModels/LES`

Options

- Dynamic k eqn — `LES { turbulence on; LESModel dynamicKEqn; }`
- k Equation — `LES { turbulence on; LESModel kEqn; }`
- Smagorinsky — `LES { turbulence on; LESModel Smagorinsky; }`
- WALE — `LES { turbulence on; LESModel WALE; }` ➔ C_e | C_k | C_w ----------—|----------—|----------— 1.048 | 0.094 | 0.325

C_e | C_k ----------—|----------— 1.048 | 0.094

Usage

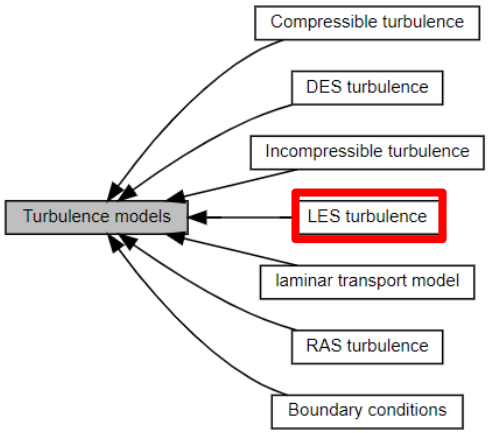
LES is selected by setting the `simulationType` entry

```
simulationType  LES;

LES
{
    \\ Model input parameters
}
```

Suitable for:

- 3-D cases, not appropriate for reduced-dimension cases
- transient, not appropriate for steady state

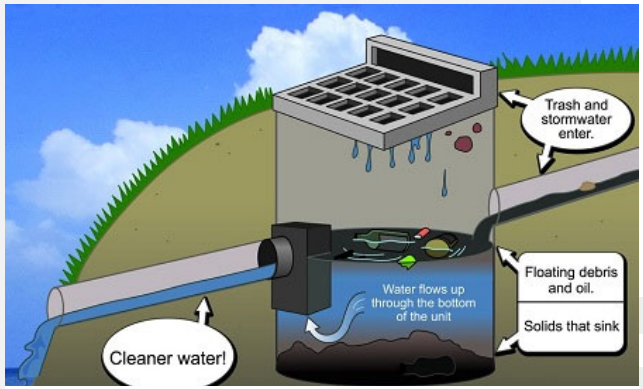


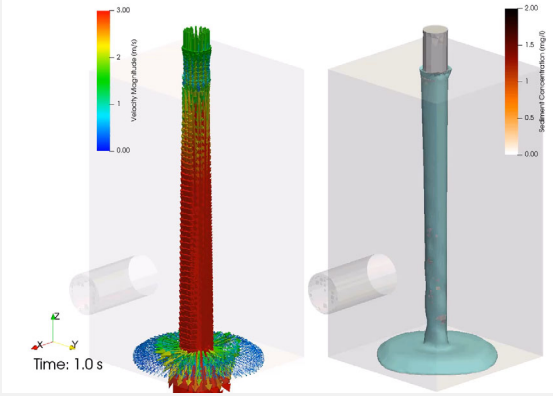
3

pre-processing
mesh, properties and IBC

- **Prepare and run a case** knowing the phenomena and choosen options – Fluid Mechanics / Mathematics / Numerics

Prepare a Case (solver ?)

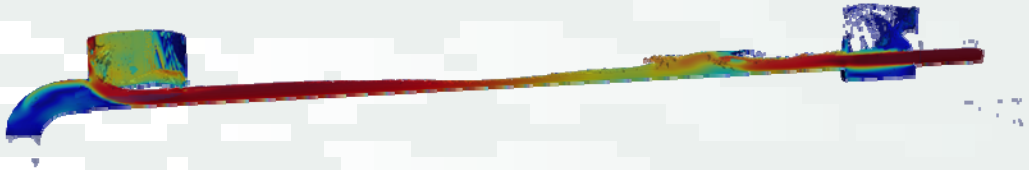


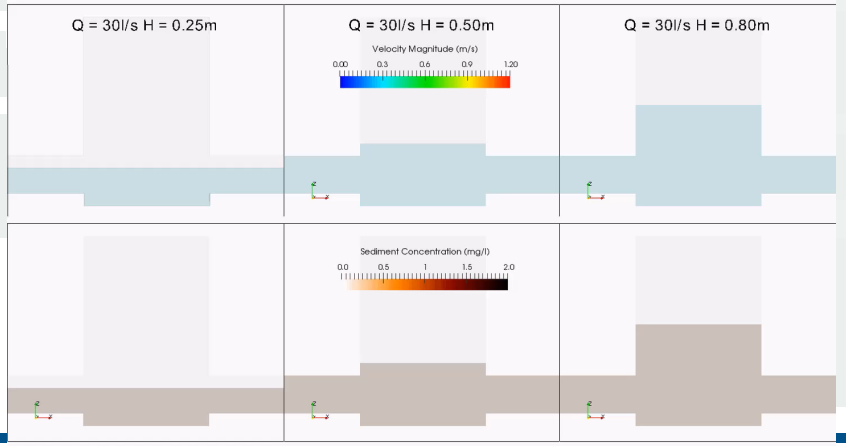


interFoam vs multiphaseEulerFoam

Choose solver

if unsteady





Prepare a Case (solver ?)

Choose solver

PDE Equations - OpenFOAM®

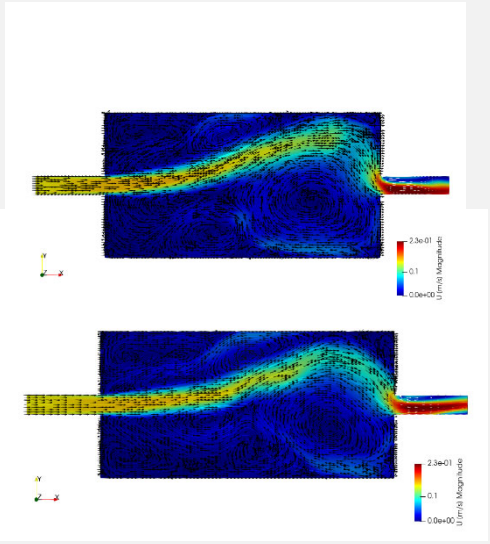
Requirements:
Adequate meshes
Adequate simplifications

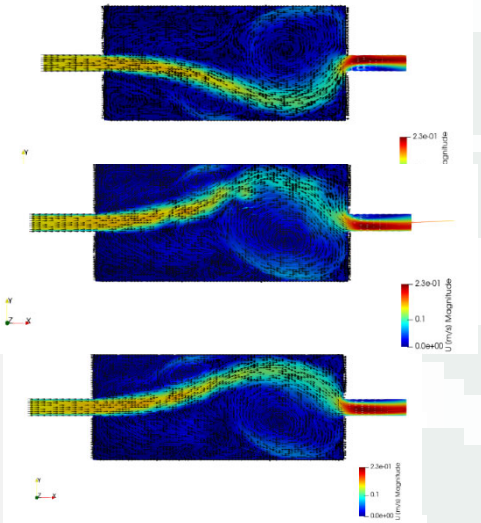
interFoam vs pimpleFoam/simpleFoam

if unsteady

Domain

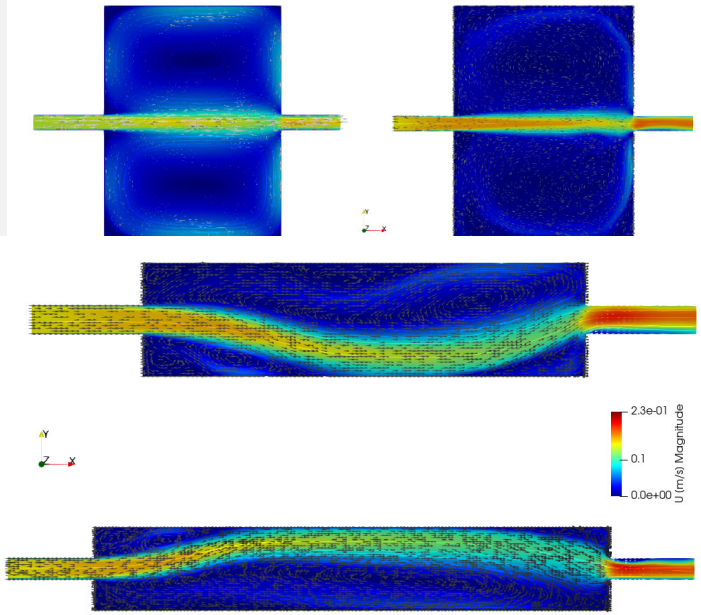
Shallow water reservoirs different dimensions - symmetry ?
Different relations b x l





2D/3D

3D with VOF



Case: start from Tutorial, planning the case

According solver interFoam

Go to file ➔cd $FOAM_RUN/tutorials/…

3

Laminar
- alpha.water
- alpha.water.orig
- p_rgh
- U

RAS - ***k - ε***
- alpha.water
- alpha.water.orig
- epsilon
- k
- nut
- nuTilda
- p_rgh
- U

2
- g
- transportProperties
- turbulenceProperties

1
- blockMeshDict

5
- controlDict
- decomposeParDict
- fvSchemes
- fvSolution
- sampling

4
- setFieldsDict

```
$ tree -L 3
.
├── 0
│   ├── alpha.water
│   └── U
├── constant
│   └── transportProperties
├── system
│   ├── blockMeshDict
│   ├── controlDict
│   ├── decomposeParDict
│   ├── fvSchemes
│   ├── fvSolution
│   └── PDRblockMeshDict

3 directories, 9 files
```

+ alfa.water
+ g + turbulenceProperties
+ sampling + setFieldsDict

Pre-processing

➔ Choose a tutorial

(…tutorials\multiphase**interFoam**\laminar\damBreak)

(…tutorials\multiphase**interFoam**\RAS\damBreak)

➔copy, rename and move

$ cp –r $FOAM_TUTORIALS/multiphase/interFoam/laminar/damBreak ./

rm damBreak reservoir

mv reservoir

➔prepare files

1. blockMeshDict in system ➔ blockMesh

2. properties in constant
- **g** ➔ **which direction is vertical?** value **(0 0 -9.81);**
- **transportProperties** ➔ phases (water air);
- **turbulenceProperties** ➔ **laminar/RAS/LES?**

3/4. IBC (**alpha.water; p_rgh; U**) in 0 and **setFieldsDict** in system

5. Options in controlDict, fvSchemes and fvSolution

Prepare a case - solver

Tip: start from a tutorial from the solver

PDE Equations - OpenFOAM®

Pre- processing

- **Mesh (blockMesh**, cartesianMesh, snappy) x**Mesh**
- Properties
- Initial and Boundary Conditions – IC/ BC
- **Control parameters and options**

Processing - Solver

- PDE resolution Gradients/divergents/ laplacians/interpolation x**Foam**

Post- processing

- Results analysis **paraFoam**

http://www.cfd-online.com/

http://www.openfoam.com/

Dictionary

Utilities

According solver

from previous course (blockMeshDict **System)**

Constant
g
transportProperties
- phases (water air);

turbulenceProperties
- Laminar
- RAS
- LES

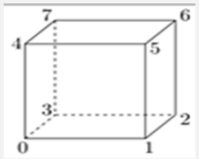
0
– one file for each variable

System
setFieldsDict
controlDict – define simulation and control parameters
fvScheme - approximations for the operators
fvSolution - how to solve the system of equations

Case: Mesh (blockMesh from previous course)

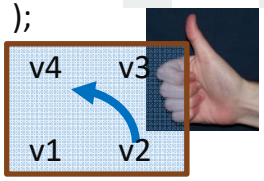
1. blockMeshDict

convertToMeters/scale value;
vertices
(
(x y z)
…
);
blocks
(
hex (v0 v1 v2 v3 v4 v5 v6 v7) (nx nt nz) simpleGrading (1 1 1)
);
edges
(
);
boundary
(
);
mergePatchPairs
(
);



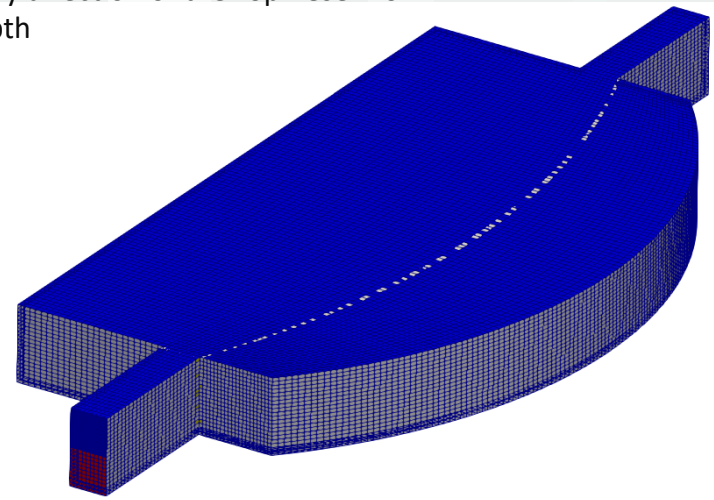
inlet
{
}
outlet
{
}
atmosphere
{
}
walls
{
}

type patch/wall;
faces
(
(v1 v2 v3 v4)
);



➔ **blockMesh**

NXI 12; // Number of cell in the x direction of the inlet Channel
NXR 63; //125 Number of cell in the x direction of the Reservoir Section
NXO 12; // Number of cell in the x direction of the outlet Channel
NYL 10; // Number of cell in the y direction of the Lower Reservior
NYM 5; // Number of cell in the y direction of the Middle Reservior
NYT 12; // Number of cell in the y direction of the Top Reservior
Nz 5; // Z direction Channel depth



paraview

Case: Properties - folder constant

2.

g

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           (0 0 -9.81);
```

dynamicMeshDict

```
dynamicFvMesh   staticMesh

or              dynamicRefineFvMesh;
                dynamicMotionSolverFvMesh;
...
```

transportProperties

```
...

phases (water air);
Water
{
    transportModel  Newtonian;
    nu              1e-06;
    rho             1000;
}
air
{
    transportModel  Newtonian;
    nu              1.48e-05;
    rho             1;
}
sigma           0.07;
```

turbulenceProperties

```
...

simulationType  laminar;
```

or

```
simulationType  RAS;
RAS
{
    RASModel        kEpsilon; //kOmega ...
    turbulence      on;
    printCoeffs     on;
}
```

Case: Boundary Conditions - folder 0

3.

U

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
   inlet
{
    type          fixedValue;
   value          uniform ( uinlet 0 0 );
 }
   outlet
   {
   type           zeroGradient;
   }
```

```
walls
  {
     type          fixedValue;          or     type          noSlip;
    value          uniform (0 0 0);
  }
  atmosphere
  {
     type          pressureInletOutletVelocity;
    value          uniform (0 0 0);
  }
```

Case: Boundary Conditions - folder 0

3.

p_rgh

```
dimensions      [1 -1 -2 0 0 0 0];
internalField   uniform 1e5;
boundaryField
{
   inlet
{
      type          fixedFluxPressure;
      value         $internalField;
 }
   outlet
   {
      type          prghPressure;
      p             $internalField;
      value         $internalField;
   }
walls
   {
      type          fixedFluxPressure;
      value         $internalField;
   }
atmosphere
   {
      type          totalPressure;
      p0            uniform 0;
   }
```

NOTE:

For cases that the **hydrostatic pressure contribution ($p(g.h)$) is important (multiphase cases)**, in OpenFOAM solver applications the p' pressure term is named **p_rgh**. The momentum equation is transformed to use p

$p' = p - p(g.h)$ $\quad -\nabla p = -\nabla p' - \rho g - g \cdot h \nabla \rho$

$$\frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho u \cdot u) - \underbrace{\nabla \cdot \left(\mu_{eff} \nabla u\right)}_{\nabla \tau} = -\nabla p^* - g \cdot h \nabla \rho + F$$

Case: Boundary Conditions - folder 0

3.

alpha.water / alpha.water.orig

```
dimensions      [0 0 0 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    inlet
    {
        type        fixedValue;
        value       uniform 1;
    }
    outlet
    {
        type        inletOutlet;
        inletValue  uniform 0;
        value       uniform 0;
    }
    walls
    {
        type        zeroGradient;
    }
    atmosphere
    {
        type        inletOutlet;
        inletValue  uniform 0;
        value       uniform 0;
    }
```

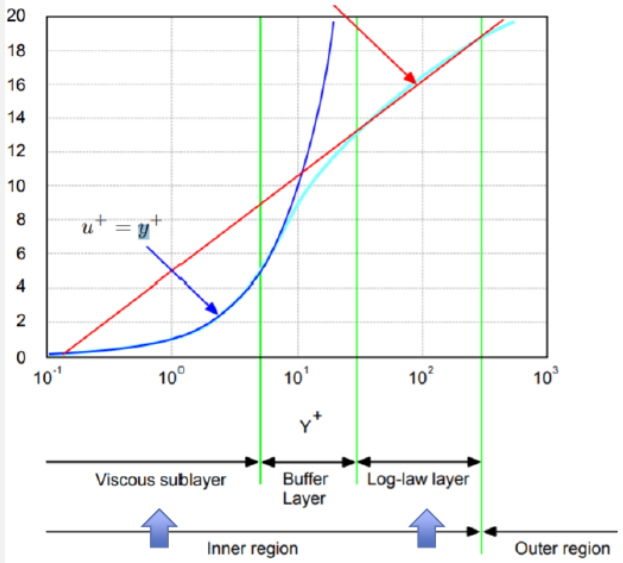
NOTE:
Due to initial conditions, it is better to define a file that will not change
.ORIG
Then

Case: Boundary Conditions - folder 0

example

3. If turbulence model RAS we need additional variables: nu, k, ε, ω

to activate the wall functions (applies for fixed and moving walls) - define wall patches as **wall** type in the dictionary file *constant/polyMesh/boundary*.



$$u^+ = \frac{1}{\kappa} ln y^+ + C^+$$

➔ quantify

$$y^+ = \sqrt{\tau_w/\rho}\; y/v$$

$$v^+ = v/\, v_\sigma,\; v_\sigma = \sqrt{\tau_w/\rho}$$

$$k^+ = k/\, {v_\sigma}^2,$$

$$\varepsilon^+ = \varepsilon v/\, {v_\sigma}^4,$$

$$\omega^+ = \omega v/\, {v_\sigma}^2$$

OpenFOAM tem preparado uma imensidão de tratamento de parede e permite na buffer layer resolver uma equação analítica para variáveis de turbulência com ν_t e preencher a lacuna entre as 2 regiões

Re	44000
L	0.06
visc	1.15E-05
U	7.33E+00

- k – turbulent kinetic energy
  - $k = \frac{3}{2}(u_{ref} T_i)^2$
  - u_{ref} - inlet flow velocity
  - T_i - turbulent intensity

K formula	1.5
U	7.333
T	0.05
k	0.201648

- ε – turbulent dissipation
  - $\varepsilon = 0.09^{3/4} \frac{k^{3/2}}{l}$
  - $l = 0.07L$
  - L - characteristic inlet scal

Epsilon formula	0.09
L	0.06
l	0.0042
epsilon	3.54262

- ω – specific dissipation rate
  - $\omega = \frac{\varepsilon}{k}$

omega	17.56831
	Should be better10-3

- *'Rule of thumb'* for estimation

Case: Initial Conditions – folder system

4.
setFieldsDict ➔ **setFields**

```
FoamFile{ version 2.0; format ascii; class dictionary;
location "system"; object setFieldsDict; }
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//defaultFieldValues
(
   volScalarFieldValue alpha.water 0
);
Regions
(
   boxToCell
   {
        box (-1.0   -1.5    0.0) (7   1.0   0.25);
     fieldValues
      (
         volScalarFieldValue alpha.water 1
      );
}
);
```

```
//Lenghts
L_in       1.00;
L_out      1.00;
L_Reservoir 4.50;

 //Heigths
        H_ReservTop   1.00;
        H_ReservMid   0.25;
        H_ReservLow   0.50;


//depth
        z0  0.0; // back plane
        z1  0.25; // front plane  ➔ 0.25
        z2  0.50; // front plane
```

Case: controlDict – folder system

5.
controlDict

```
FoamFile{ version 2.0; format ascii; class dictionary;
location "system"; object controlDict; }
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
application interFoam;
startFrom startTime;
startTime 0;
stopAt endTime;
endTime 1;
deltaT 0.001;
writeControl adjustableRunTime;
writeInterval 0.05;
purgeWrite 0;
writeFormat ascii;
writePrecision 6;
writeCompression off;

startFrom latestTime;

stopAt endTime;
endTime 10;

timeFormat general;
timePrecision 6;
runTimeModifiable yes;
adjustTimeStep yes;
maxCo 1;
maxAlphaCo 1;
maxDeltaT 1;
#sinclude "sampling"
```

```
sampling
functions
{
  sampleSets
  {
    type sets;
    libs ("libsampling.so");
    writeControl timeStep;
    writeInterval 1;
    setFormat vtk;
    interpolationScheme cellPointFace;
    fields ( alpha.water );
    sets
    (
      gauge_1
      {
        type face;
        axis y;
        start (0.02 0.20 0.005);
        end (0.02 0.25 0.005);
        nPoints 100;
      }
    ) } }
```

Case: fvSchemes – folder system

5.
fvSchemes

```
FoamFile{ version 2.0; format ascii; class dictionary;
location "system";
  object fvSchemes; }
// * * * * * * * * * *
 ddtSchemes
{
   default       Euler;
}
gradSchemes
{
   default       Gauss linear;
}
divSchemes
{
   div(rhoPhi,U)  Gauss linearUpwind grad(U);
   div(phi,alpha)  Gauss vanLeer;
   div(phirb,alpha) Gauss linear;
   div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
   default       Gauss linear corrected;
}
interpolationSchemes
{
   default       linear;
}
snGradSchemes
{
   default       corrected;
}
```

$$\frac{dT}{dt} \cong \frac{T(t+\Delta t) - T(t)}{\Delta t}$$

In OpenFOAM®, complex concepts can be writt
example, systems of equations are implement
follows the mathematical notation, e.g.:

Time rate of change	$\frac{\partial}{\partial t}(\phi)$
Gradient	$\nabla\phi$
Divergence	$\nabla \cdot \phi$
Laplacian	$\nabla^2\phi$
Linearised sources	$s\phi$

Numeric schemes

- CDS (linear, cubic,middle)
- UDS (upwind,…)
- TVD (VanLeer, MUSCL,NVD)
- SFCD

Time discretization

- Euler
- Backward
- Leapfrog
- Adams – Bashford
- Adams-Moulton
- Crank-Nicholson

Case: fvSolution – folder system

5.
fvSolution

```
FoamFile{   version    2.0;
  format     ascii;   class      dictionary;
   location   "system";  object     fvSolution; }
//**********************
Solvers
{
   "alpha.water.*“
   {
      nAlphaCorr     2;
      nAlphaSubCycles 1;
      cAlpha         1;
      MULESCorr      yes;
      nLimiterIter    5;
      solver         smoothSolver;
      smoother       symGaussSeidel;
      tolerance      1e-8;
      relTol         0;
   }
```

```
"pcorr.*"
 {
     solver          PCG;
     preconditioner  DIC;
     tolerance       1e-5;
     relTol          0;
 }
 p_rgh
 {
     solver          PCG;
     preconditioner  DIC;
     tolerance       1e-07;
     relTol          0.05;
 }
 p_rghFinal
 {
     $p_rgh;
     relTol          0;
}
```

iterative method

Depending on the structure of your matrix, this will allow a reduction on the number of iterations

Tolerance for stopping the iterative procedure

relTol is the relative tolerance between the initial and the final residual

```
{
   momentumPredictor   no;
   nOuterCorrectors    1;
   nCorrectors         3;
   nNonOrthogonalCorrectors 0;
}
relaxationFactors
{
   equations
   {
      ".*" 1;
   }
}
```

SOLVERS

- SOR
- Diagonal
- CG, BiCG
- GAMG
- GMRES
- BiCGStab

Preconditioning used to speed up the matrix solution:

- SOR
- Jacobi,
- ILU

Case: fvSolution – folder system

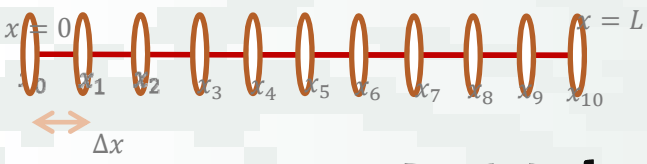
Resolution

Diferencial Equation: $2.0\frac{\partial^2 y}{\partial x^2} = -20.0$ defined in $x \in [0\ L]$

BC:

$y(x=0) = 10.0$ $\quad\quad$ $\frac{\partial y}{\partial x}(x=L) = 0.0$

Consider the domain discretized into 11 equally spaced points. $\Delta x = 0.2$.



$$\begin{cases} y_0 = 10 \\ 2\dfrac{y_0 - 2y_1 + y_2}{(\Delta x)^2} = -20 \\ 2\dfrac{y_1 - 2y_2 + y_3}{(\Delta x)^2} = -20 \\ \dots \\ 2\dfrac{y_8 - 2y_9 + y_{10}}{(\Delta x)^2} = -20 \\ \dfrac{y_{10} - y_9}{\Delta x} = 0 \end{cases}$$

$$\begin{bmatrix} 1 & & & & & & & & & & \\ 50 & -100 & 50 & & & & & & & & \\ & 50 & -100 & 50 & & & & & & & \\ & & 50 & -100 & 50 & & & & & & \\ & & & 50 & -100 & 50 & & & & & \\ & & & & 50 & -100 & 50 & & & & \\ & & & & & 50 & -100 & 50 & & & \\ & & & & & & 50 & -100 & 50 & & \\ & & & & & & & 50 & -100 & 50 & \\ & & & & & & & & -5 & 5 & \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \end{bmatrix} = \begin{bmatrix} 10.0 \\ -20.0 \\ -20.0 \\ -20.0 \\ -20.0 \\ -20.0 \\ -20.0 \\ -20.0 \\ -20.0 \\ -20.0 \\ 0.0 \end{bmatrix} \rightarrow \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \end{bmatrix} = \begin{bmatrix} 10.0 \\ 13.6 \\ 16.8 \\ 19.6 \\ 22.0 \\ 24.0 \\ 25.6 \\ 26.8 \\ 27.6 \\ 28.0 \\ 28.0 \end{bmatrix}$$

4 processing

- **Run a case** and analyse parameters and residuals during solving

Case: Processing

- ➔blockMesh
- ➔setFields
- ➔interFoam

or

- ➔blockMesh
- ➔setFields
- ➔decomposePar
- ➔interFoam
- ….
- ➔reconstruct

```
FoamFile{ version 2.0; format ascii; class dictionary;
object decomposeParDict;}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//numberOfSubdomains 4;
method simple;
Coeffs
{
  n (2 2 1);
}
distributed no;
roots ( );
```

Case: Processing

Residual analysis

```
Starting time loop

Time = 0.005

Courant Number mean: 3.05054e-05 max: 0.0128322
smoothSolver:  Solving for Ux, Initial residual = 1, Final residual = 9.73224e-06, No Iterations 8
smoothSolver:  Solving for Uy, Initial residual = 0, Final residual = 0, No Iterations 0
smoothSolver:  Solving for Uz, Initial residual = 0, Final residual = 0, No Iterations 0
DICPCG:  Solving for p, Initial residual = 1, Final residual = 0.0489658, No Iterations 567
time step continuity errors : sum local = 2.98744e-06, global = -1.0861e-08, cumulative = -1.0861e-08
DICPCG:  Solving for p, Initial residual = 0.000824352, Final residual = 9.64524e-07, No Iterations 635
time step continuity errors : sum local = 6.15632e-08, global = -2.72229e-11, cumulative = -1.08882e-08
ExecutionTime = 47.86 s  ClockTime = 48 s

Time = 0.01

Courant Number mean: 0.0310419 max: 0.15522
smoothSolver:  Solving for Ux, Initial residual = 0.737718, Final residual = 7.4179e-06, No Iterations 9
smoothSolver:  Solving for Uy, Initial residual = 0.342033, Final residual = 3.2086e-06, No Iterations 9
smoothSolver:  Solving for Uz, Initial residual = 0.341358, Final residual = 3.20142e-06, No Iterations 9
DICPCG:  Solving for p, Initial residual = 0.00400076, Final residual = 0.000170521, No Iterations 581
time step continuity errors : sum local = 3.43774e-06, global = 1.91352e-09, cumulative = -8.9747e-09
DICPCG:  Solving for p, Initial residual = 0.00128175, Final residual = 9.18694e-07, No Iterations 635
time step continuity errors : sum local = 3.89858e-08, global = 8.86936e-11, cumulative = -8.88601e-09
ExecutionTime = 89.88 s  ClockTime = 90 s

Time = 0.015

Courant Number mean: 0.0310739 max: 0.125886
smoothSolver:  Solving for Ux, Initial residual = 0.0411546, Final residual = 2.82741e-06, No Iterations 7
smoothSolver:  Solving for Uy, Initial residual = 0.066555, Final residual = 6.89743e-06, No Iterations 7
smoothSolver:  Solving for Uz, Initial residual = 0.0656764, Final residual = 6.77495e-06, No Iterations 7
DICPCG:  Solving for p, Initial residual = 0.0708021, Final residual = 0.00353566, No Iterations 37
time step continuity errors : sum local = 1.10025e-06, global = 6.72094e-08, cumulative = 5.83234e-08
DICPCG:  Solving for p, Initial residual = 0.0414173, Final residual = 9.60628e-07, No Iterations 684
time step continuity errors : sum local = 2.94561e-10, global = -9.69508e-13, cumulative = 5.83224e-08
ExecutionTime = 115.33 s  ClockTime = 115 s
```

$$\underbrace{\left(\frac{k_i A_i}{\Delta x}\right)}_{A_{21}} T_1 + \underbrace{\left(\frac{k_i A_i}{\Delta x} + \frac{k_r A_r}{\Delta x}\right)}_{A_{22}} T_2 + \underbrace{\left(\frac{k_r A_r}{\Delta x}\right)}_{A_{23}} T_3 = Q_{source}$$

- In general form we can write:

$$A_{21} T_1 + A_{22} T_2 + A_{23} T_3 + A_{24} T_4 = B_2$$

- Repeat this process for the other cells

$$A_{11} T_1 + A_{12} T_2 + A_{13} T_3 + A_{14} T_4 = B_1$$
$$A_{21} T_1 + A_{22} T_2 + A_{23} T_3 + A_{24} T_4 = B_2$$
$$A_{31} T_1 + A_{32} T_2 + A_{33} T_3 + A_{34} T_4 = B_3$$
$$A_{41} T_1 + A_{42} T_2 + A_{43} T_3 + A_{44} T_4 = B_4$$

Matrix Equation for the Residual

$$A_{11} T_1 + A_{12} T_2 + A_{13} T_3 + A_{14} T_4 - B_1 = r_1$$
$$A_{21} T_1 + A_{22} T_2 + A_{23} T_3 + A_{24} T_4 - B_2 = r_2$$
$$A_{31} T_1 + A_{32} T_2 + A_{33} T_3 + A_{34} T_4 - B_3 = r_3$$
$$A_{41} T_1 + A_{42} T_2 + A_{43} T_3 + A_{44} T_4 - B_4 = r_4$$

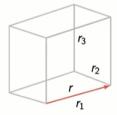
- These (modified) energy balance equations can also be assembled in matrix

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} - \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} \qquad \boldsymbol{AT} - \boldsymbol{B} = \boldsymbol{r}$$

The L_∞ norm

- The L_∞ norm is equal to the maximum absolute value.

$$\boldsymbol{r} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix} \qquad r = ||\boldsymbol{r}||_\infty = \max |r_i|$$



- The L_∞ norm only considers the worst (highest) residual in the mesh

5 post-processing

- **Analyse and discuss results** and enhance knowledge of phenomena – Fluid Mechanics

Post-processing

1. mesh

```
checkMesh -allTopology -allGeometry
```

➔polyMesh

Pipep › constant › polyMesh

Nome

- boundary
- cellLevel
- cellZones
- faces
- faceZones
- level0Edge
- neighbour
- owner
- pointLevel
- points
- pointZones
- surfaceIndex

➔ ***points***

- Nº points
- List of coordinates (X,Y,Z)

Every point belongs to a face

➔ ***faces***

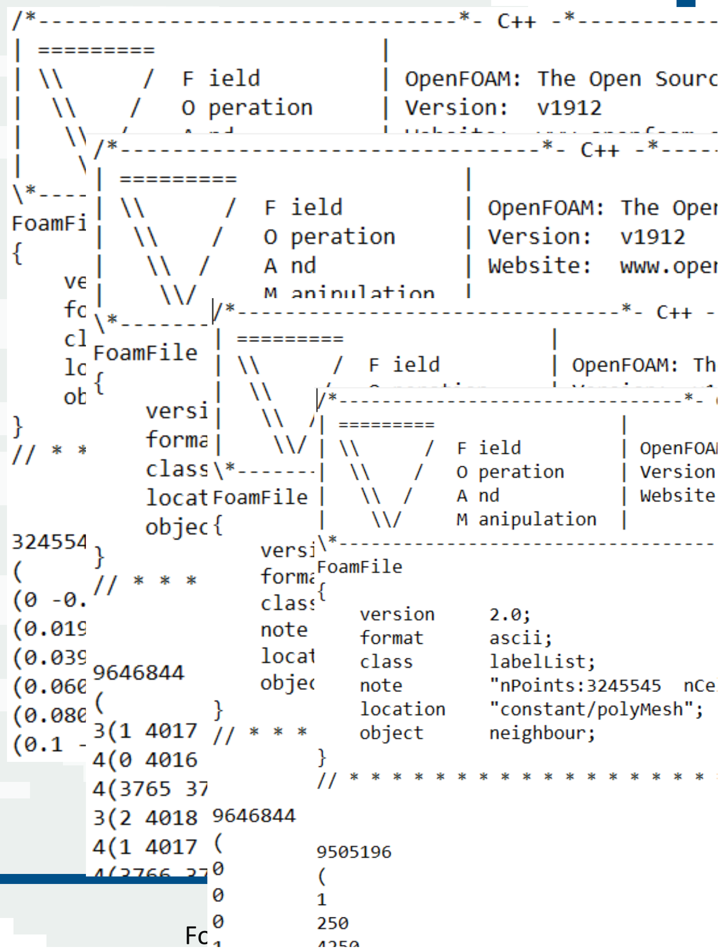
- cells number
- List of cells

nºvertices(vértices)

a list of indices for vertices in the list of points

➔ ***owner***

- Cells number
- List of cell labels

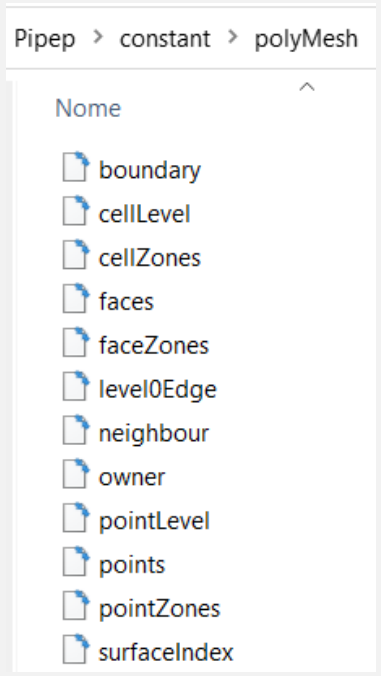


Post-processing

1. mesh

```
checkMesh -allTopology -allGeometry
```

➔polyMesh



➔***boundary***

- List of patches - boundaries
- For each
  - Type (patch/wall)
  - inGroup (Wall, symmetryPlane)
  - Nº faces (nFaces - faces in patch.)
  - startFace (faces list in patch)

➔cellLevel/pointLevel

number of cells

List of cells and level 0,1,2

(3245545{0})

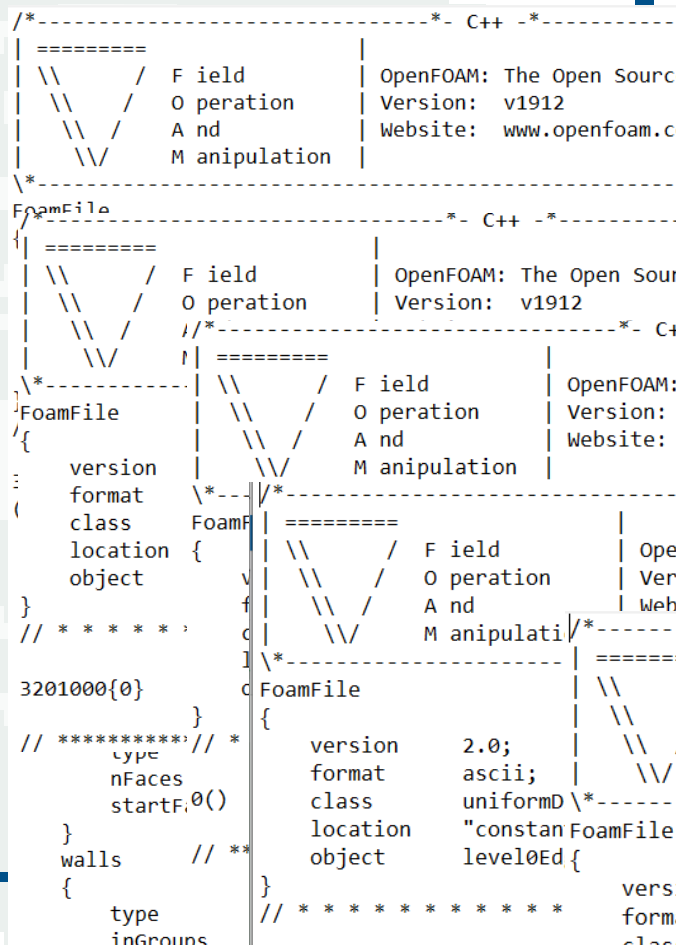
➔ cellZones/pointZones

0()

➔ faceZones

➔ level0Edges

➔ surfaceIndex



Post-processing

2 results

Formats:
Instant ➔variables

File with dimensions
Internal field
Internal point number
List
Boundary Field and type

p - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
/*--------------------------------*- C++ -*----------
| =========                 |
| \\      /  F ield         | OpenFOAM: The Open Source
|  \\    /   O peration     | Version:  v1912
|   \\  /    A nd           | Website:  www.openfoam.co
|    \\/     M anipulation  |
\*---------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "1.04";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * *

dimensions      [0 2 -2 0 0 0 0];


internalField   nonuniform List<scalar>
275584
(
0.741854
0.606258
0.5985
0.593906
0.590135
0.586526
0.582921
0.579313
0.575693
0.572066
0.568436
0.564804
0.561172
0.557539
0.553906
0.550273
0.546641

boundaryField
{
    inlet
    {
        type            zeroGra
    }
    outlet
    {
        type            fixedVa
        value           uniform
    }
    walls
    {
```

```
/*--------------------------------*- C++ -*----------
| =========                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD
|  \\    /   O peration     | Version:  v1912
|   \\  /    A nd           | Website:  www.openfoam.com
|    \\/     M anipulation  |
\*---------------------------------------------------
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "1.04";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * *

dimensions      [0 1 -1 0 0 0 0];


internalField   nonuniform List<vector>
275584
(
(0.034596 0.000930291 0.0131352)
(0.00758692 1.34549e-05 0.000212624)
(0.00493236 -1.96984e-05 5.34171e-05)
(0.004177 -4.01313e-06 2.64039e-05)
(0.00394785 -3.15725e-07 1.898e-05)
(0.00388558 -1.54128e-07 1.71267e-05)
(0.00386823 -1.42879e-07 1.71316e-05)
(0.00386574 -2.70659e-07 1.72545e-05)
(0.00386559 -3.82781e-07 1.73518e-05)
(0.00386574 -4.27992e-07 1.74007e-05)
(0.00386587 -4.40642e-07 1.7423e-05)
(0.00386595 -4.39235e-07 1.74329e-05)
(0.00386601 -4.34512e-07 1.74372e-05)
(0.00386603 -4.31895e-07 1.74389e-05)

boundaryField
{
    inlet
    {
        type
        value
    }
    outlet
    {
        type
    }
    walls
    {
        type
        value
```

Post-processing

2 results

analysis of the quality of results

- check mesh appearance - orthogonality, slope, symmetry, density, quality, type (e.g. coordinates, structured versus unstructured)
- mesh contains the relevant physics? – turbulence
- error analysis (discretization errors are the biggest concern, rounding errors- usually the definition of double precision is sufficient, iterative convergence are due to solving the equations against a specified convergence criterion
    - Mesh dependency analysis – must be done with the certainty of defining other variables
    - Analysis of the domain and type of boundary conditions2D versus 3D simulations/mesh types
    - Convergence study (The solution converged – RMS 1E-5 residual analysis – a turbulent simulation as laminar or an unstable simulation as stable)
    - sensitivity analysis of other relevant characteristics
        - Spatial discretization (or advection) schemes ➔ try different methods
        - Temporal discretization schemes
        - Turbulence model and turbulence intensity ➔ case of RANS, try different types: $k-\varepsilon$, realized $k-\varepsilon$, RNG$k-\varepsilon$,$k-\omega$,SST $k-\omega$
        - Other models such as heat transfer, free surface and entrainment
- Validate results

Post-processing

- OpenFOAM post-processing is performed with the open source cross-platform data analysis and visualization software, ParaView (ParaView, 2012), developed by Sandina National Laboratory, Kitware Inc., and Los Alamos National Laboratory under BSD license (Berkeley Software Distribution) and using the VTK visualization library. This viewer can be installed when installing OpenFOAM through the existing repositories provided by OpenCFD.

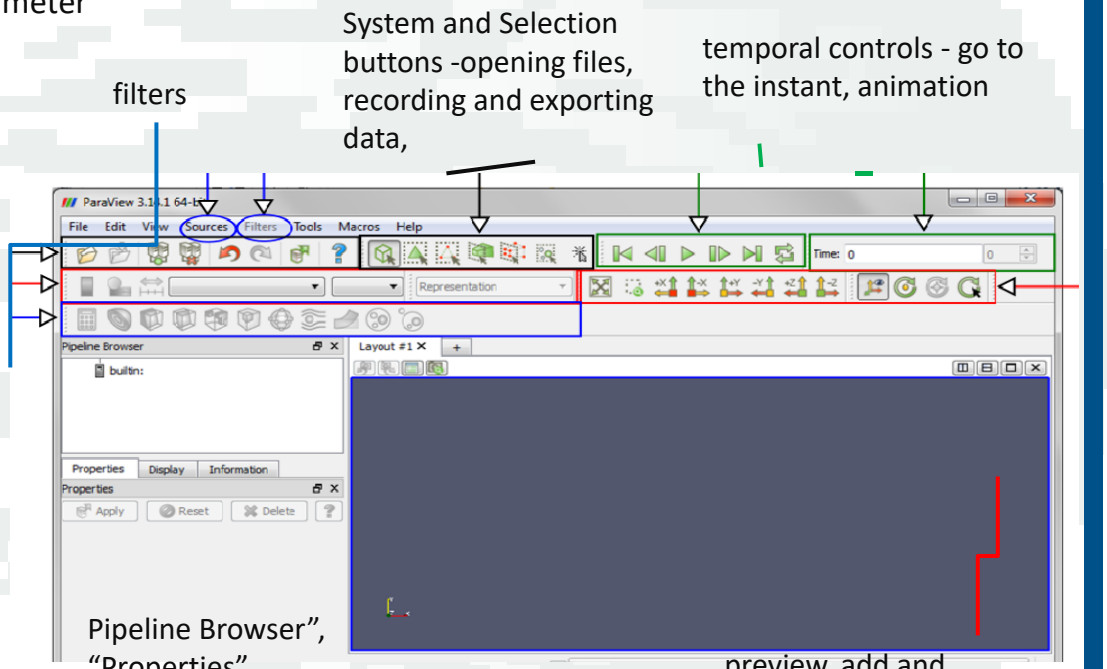
- ParaView allows, in a simple way, the processing of large-scale data, namely that generated by OpenFOAM, including the creation of contour lines, vectorization, definition of streamlines through a process of orienting successive filters. Due to its large-scale data processing nature, it also allows the same data to be processed through multiprocessors or different storage locations.

Post-processing

Modelling a case : geometry and mesh + properties + IC + BC ➔ integration/solvers ➔ results

- Our study: flow through pipes with different diameter
  PARAVIEW

ParaView is a tool that can be similar to CAD software from a conceptual point of view, as it allows the visualization of several layers (here known as filters) and the occlusion of others, always processing according to the previous filter. The interactivity of the "Layout" is customizable, however the settings by "default" are for the left mouse button the rotation, the right mouse button and the zoom wheel, and the middle button the displacement. Familiarization with these commands is essential as it will allow you to choose the viewing angle as well as the distance.

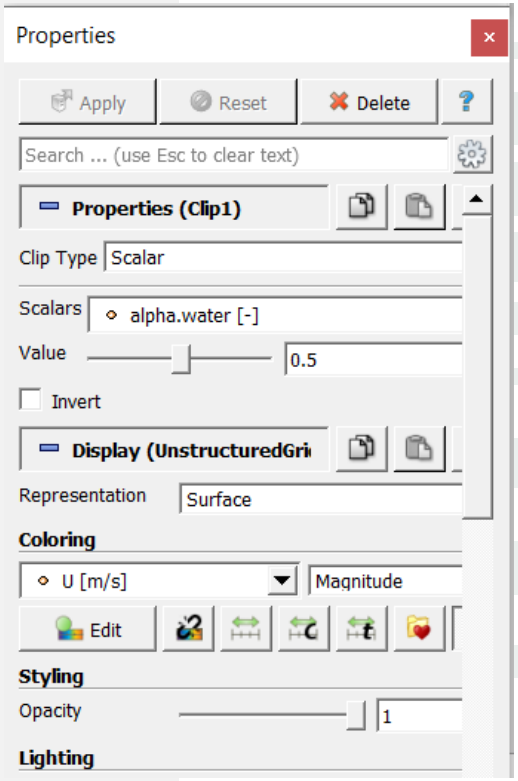


Post-processing

- Our study: flow through pipes with different diameter

2. Results - variables

clip



6

Conclusions and additional tips

- **Run a case** and enhance knowledge of phenomena – Fluid Mechanics

Conclusions and additional tips

- Run a case to enhance knowledge of phenomena – Fluid Mechanics
- Verify always all the files
- Verify the residuals
- Do a mesh independence study for RANS models
- Analyse the results properly

Acknowledgments

This work had the support of national funds through Fundação para a Ciência e Tecnologia, I. P (FCT), under the projects UIDB/04292/2020, UIDP/04292/2020 , granted to MARE, and LA/P/0069/2020, granted to the Associate Laboratory ARNET

END