Online Training – Advanced session
February 2022

Multiphase flows modeling in OpenFOAM: Guided Tutorials

wolf dynamics

Guided tutorials

From this point on and unless it is strictly necessary, or we introduce new features, we will not explain every single step or comment on the directories and files to be used.

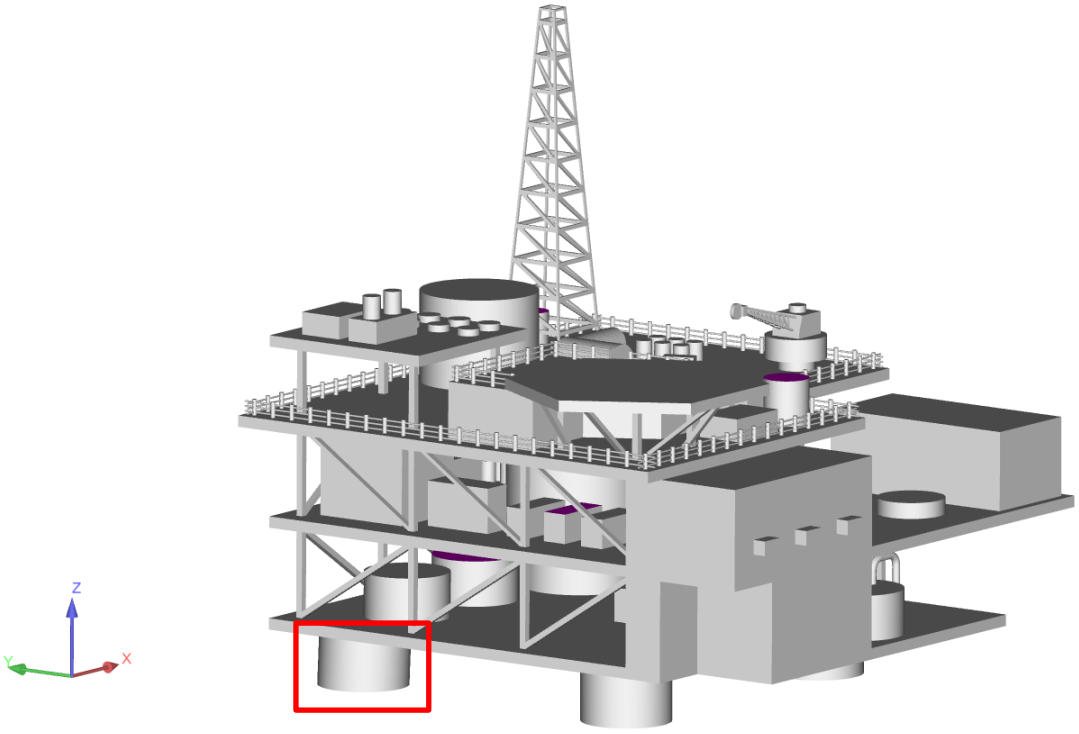
Guided tutorials

- **Guided tutorial 1.** Wave impact in a column. VOF Large scale.
- This case is ready to run.
- The case is located in the directory:

$TM/multiphase/guided_tutorials/GT1/

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

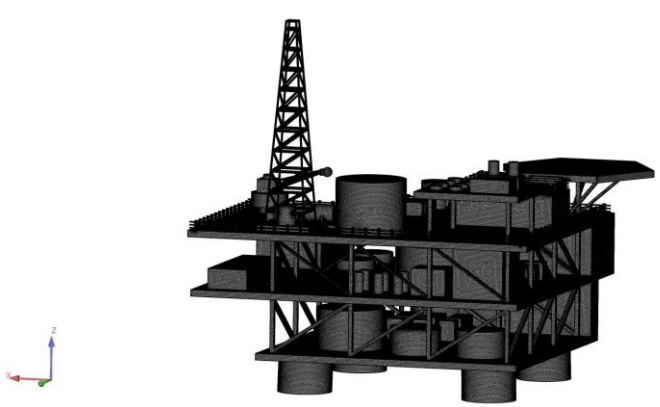
Guided tutorial 1 – Wave impact in a column

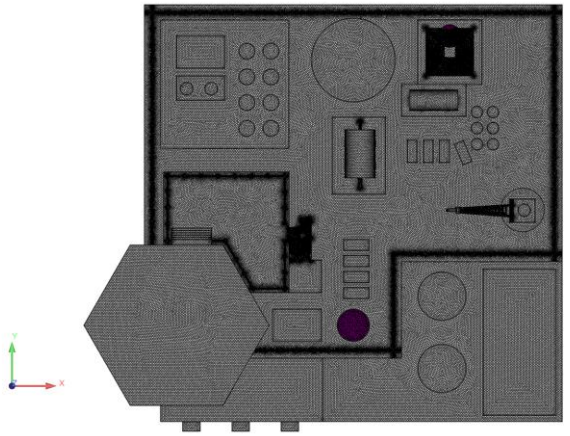


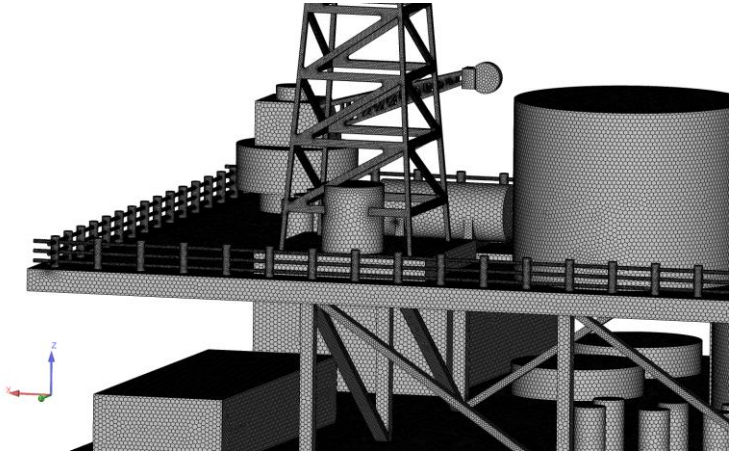
- This is the big picture.
- For this case we are going to simulate only one support column (the column in the red square).

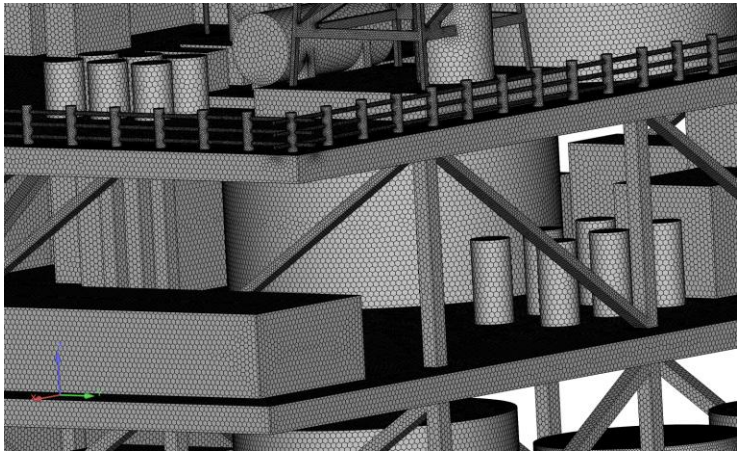
Guided tutorials

Guided tutorial 1 – Wave impact in a column





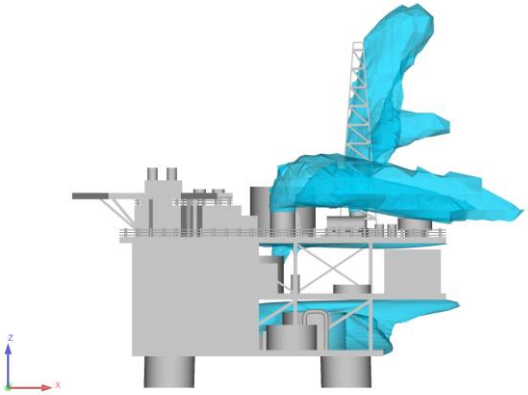


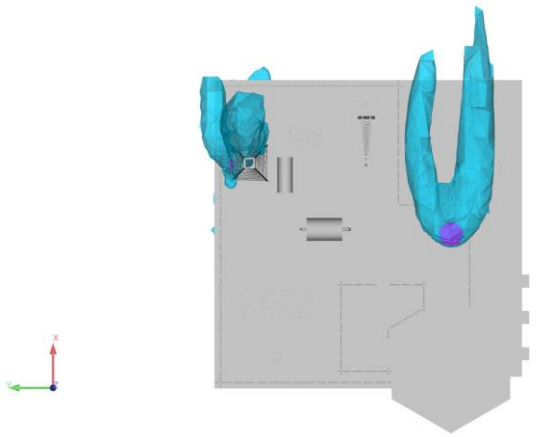


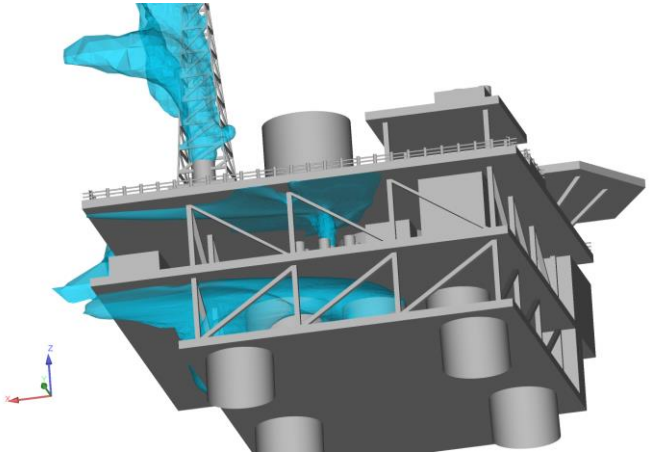
- And in case you were wondering. Yes, we do have a mesh and a solution using OpenFOAM.

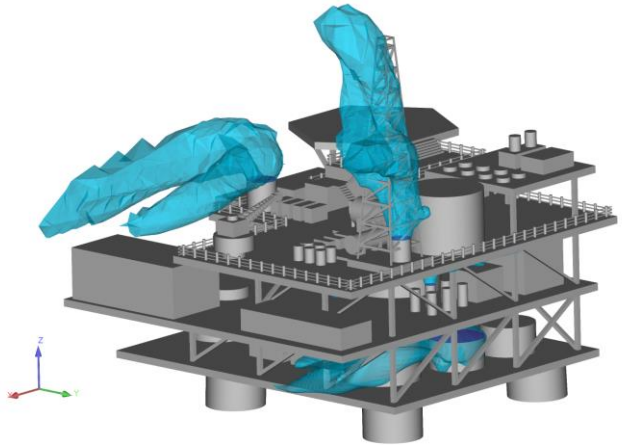
Guided tutorials

Guided tutorial 1 – Wave impact in a column





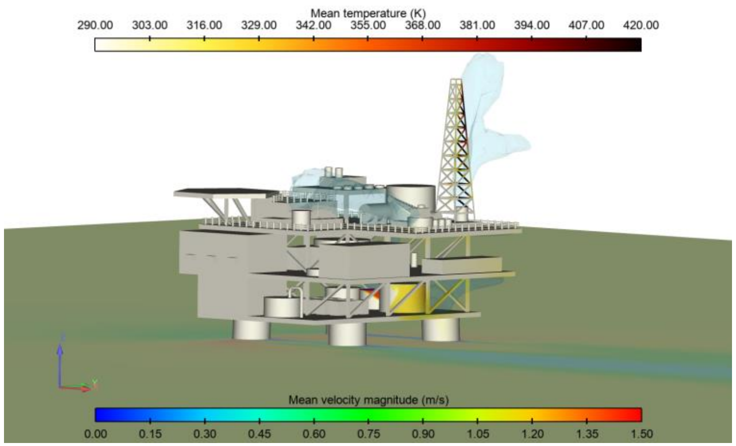


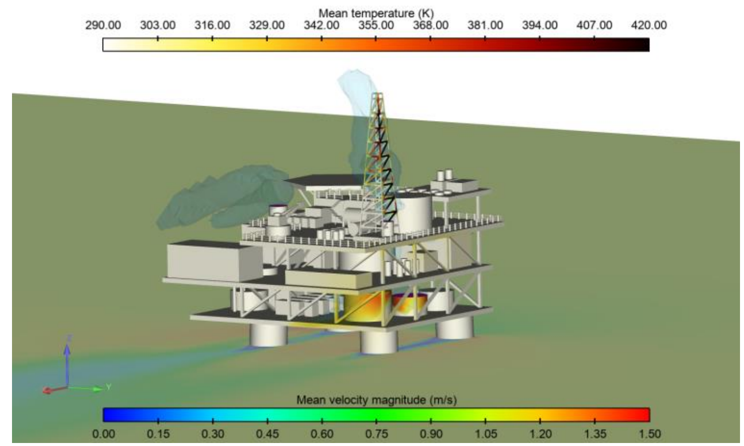


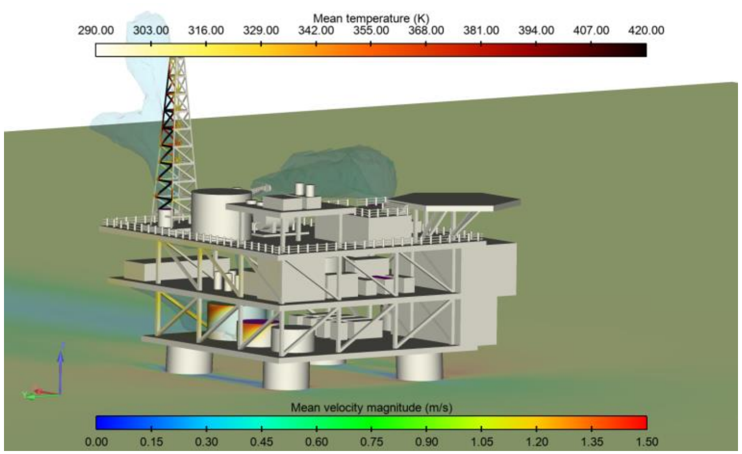
- And in case you were wondering. Yes, we do have a mesh and a solution using OpenFOAM.

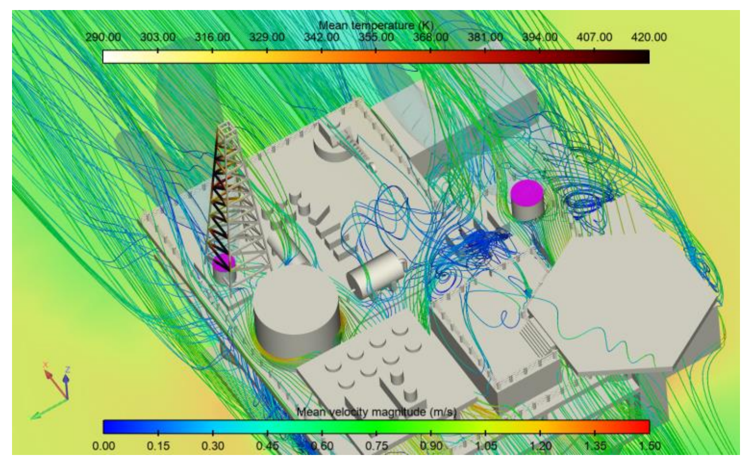
Guided tutorials

Guided tutorial 1 – Wave impact in a column



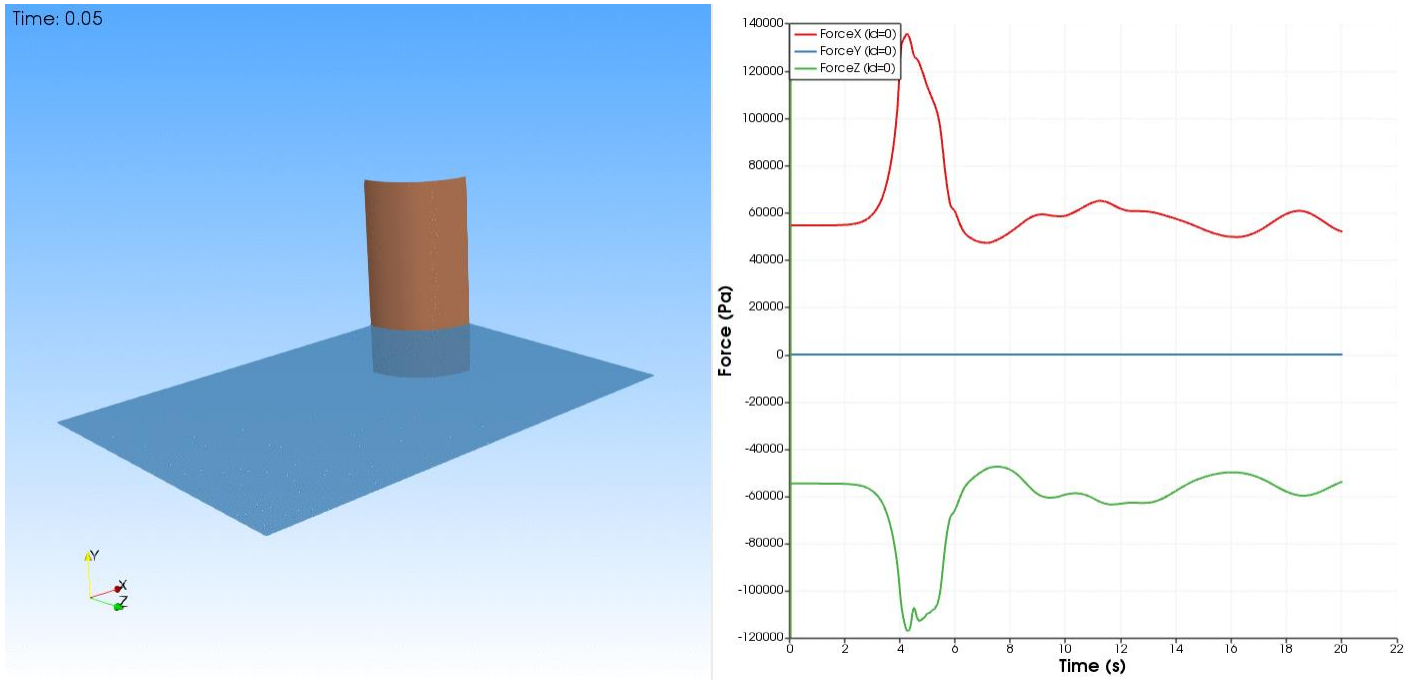






- And in case you were wondering. Yes, we do have a mesh and a solution using OpenFOAM.

Guided tutorial 1 – Wave impact in a column



Hexahedral mesh.

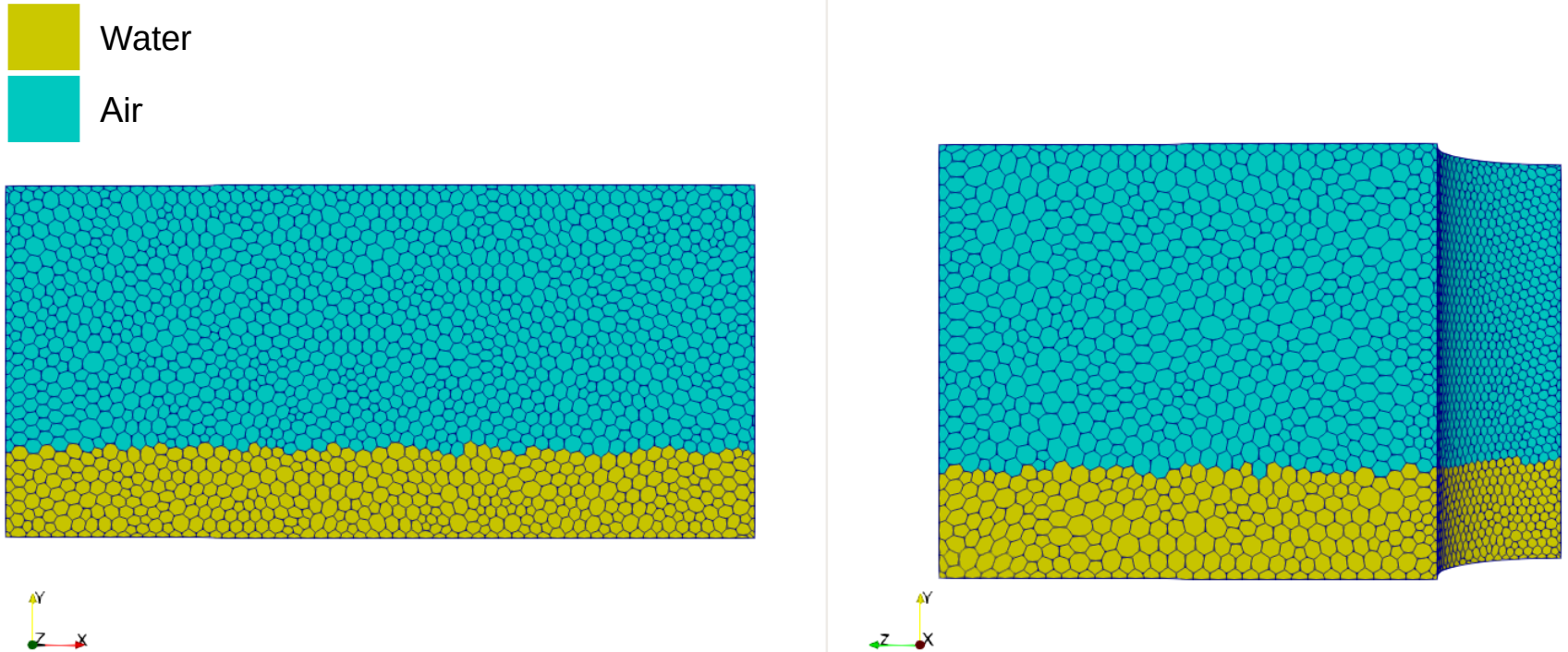
Left figure: free surface interaction with solid structure. Right figure: forces acting on the column.

http://www.wolfdynamics.com/training/mphase/gt1/hexa/ani2.gif

- The incoming wave is a solitary wave.
  - We are not going to address the theory or how to setup the wave*.
- We will address the influence of the cell type on the initialization and solution accuracy and stability, basic case setup, and postprocessing using paraview.

* More on waves at this link https://cfd.direct/openfoam/free-software/waves/

Guided tutorial 1 – Wave impact in a column

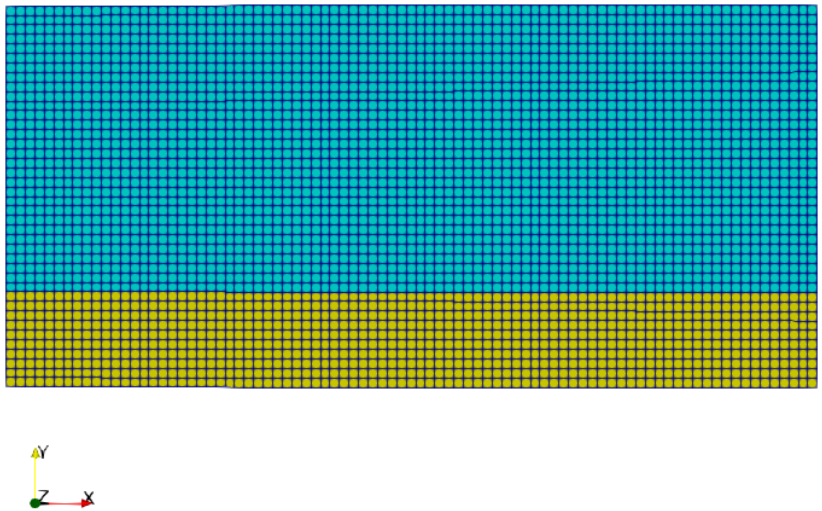


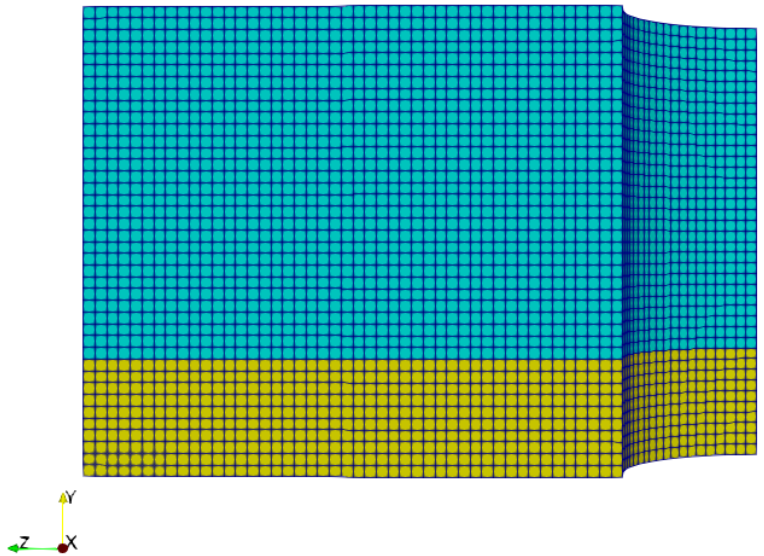
- When conducting free surface simulations, it is extremely important to generate a mesh that is aligned with the free surface level (at least at the free surface level).
- If the cells are not aligned, this may introduce spurious oscillations which might or might not be significant.
- In this case, the polyhedral cells are not aligned with the free surface.
  - This does not mean that the mesh is wrong, we need to assess the final solution.

Guided tutorial 1 – Wave impact in a column

Water

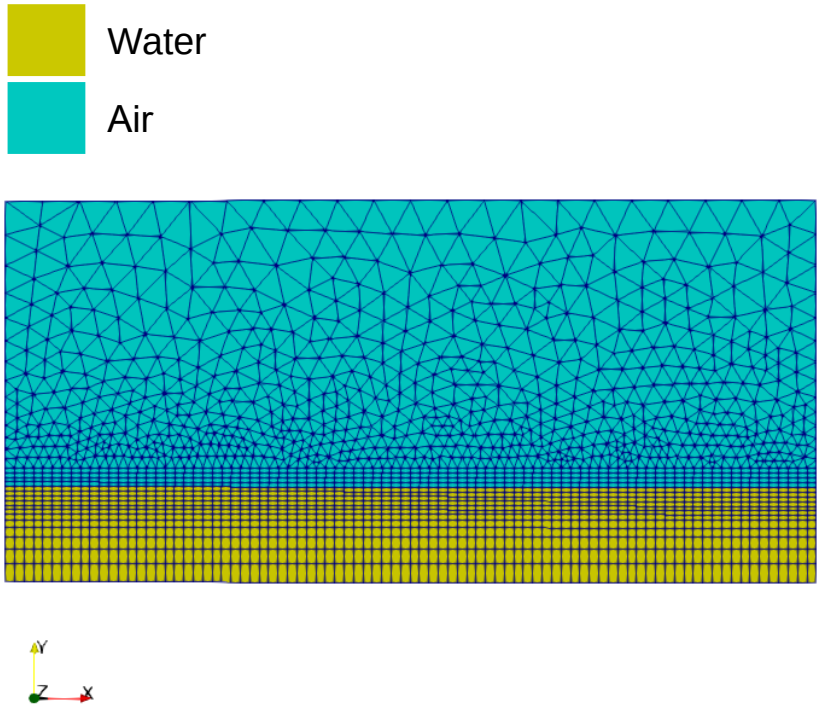
Air

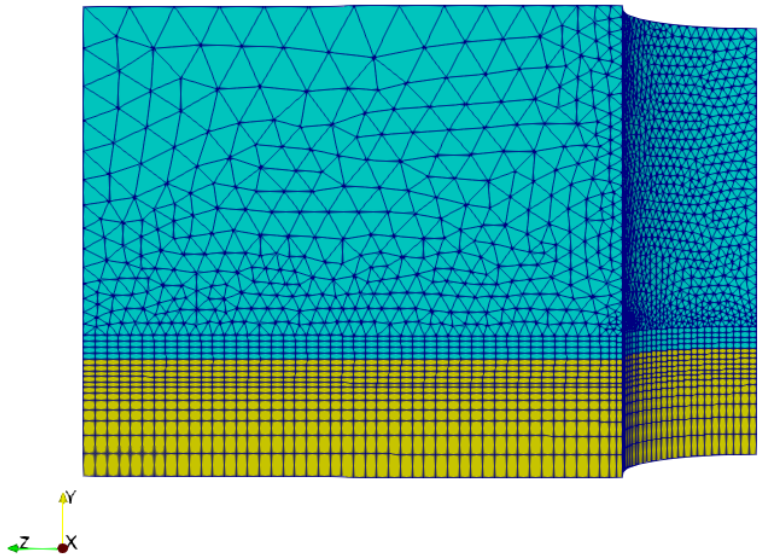




- When conducting free surface simulations, it is extremely important to generate a mesh that is aligned with the free surface level (at least at the free surface level).
- If the cells are not aligned, this may introduce spurious oscillations which might or might not be significant.
- <u>In this case, the hexahedral cells are aligned with the free surface.</u>
  - The quality of the solution at the free surface is much better than the solution with the polyhedral mesh.
  - Also, the error associated with this mesh is much lower that the error associated with the polyhedral mesh.

Guided tutorial 1 – Wave impact in a column

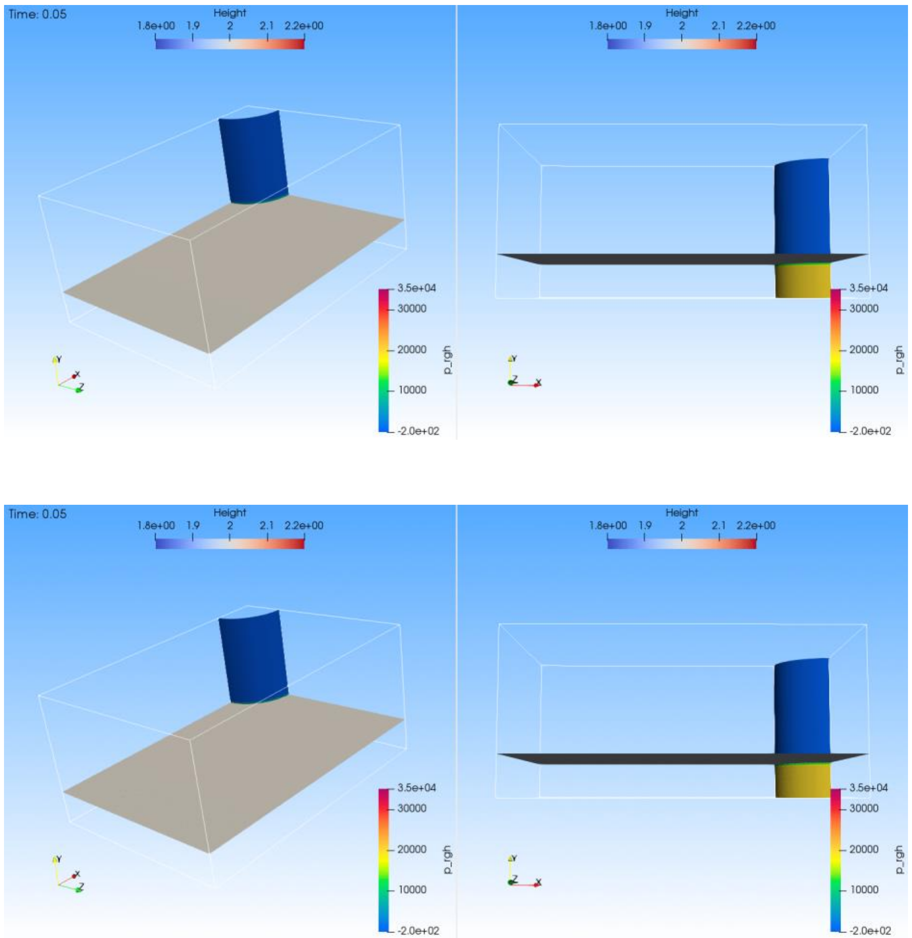




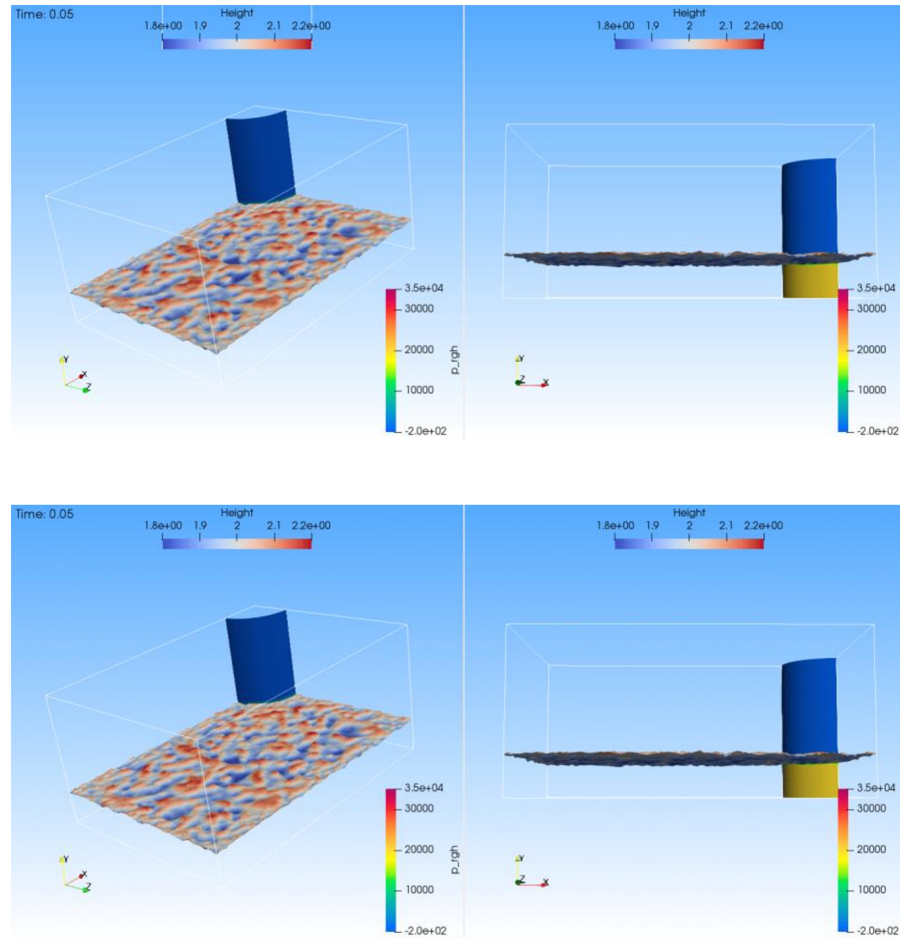
- When conducting free surface simulations, it is extremely important to generate a mesh that is aligned with the free surface level (at least at the free surface level).
- If the cells are not aligned, this may introduce spurious oscillations which might or might not be significant.
- Hybrid meshes can be used with no problem.
  - It is important to have in mind that at the free surface level, the cells should be aligned with the free surface.

Guided tutorial 1 – Wave impact in a column

Solution comparison – Hexahedral mesh against polyhedral mesh



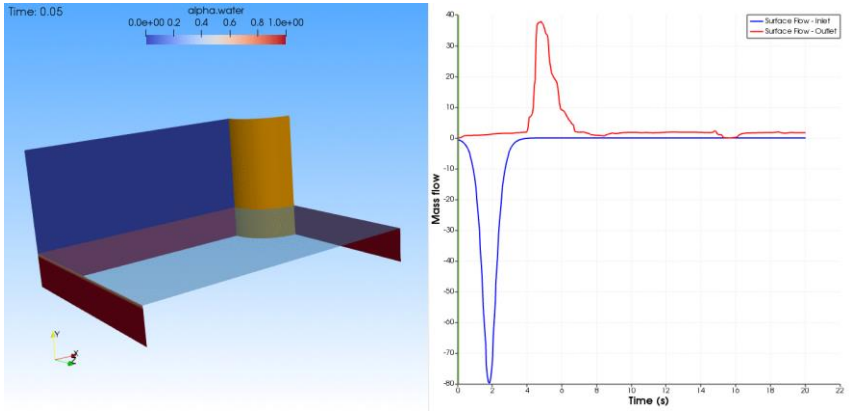
Hexahedral mesh
http://www.wolfdynamics.com/training/mphase/gt1/hexa/ani1.gif



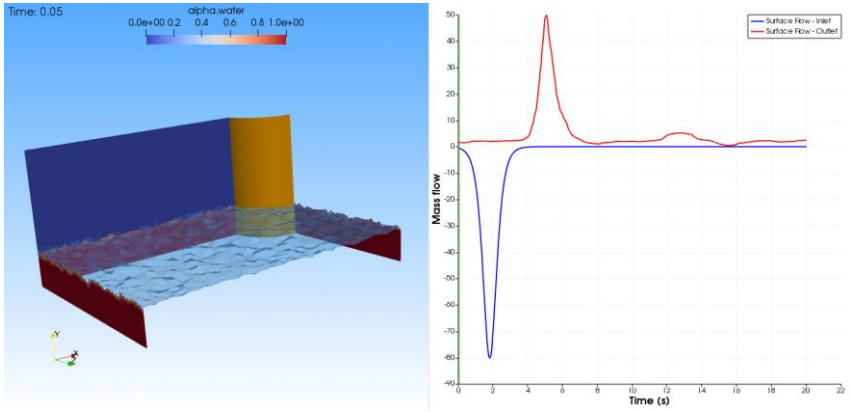
Polyhedral mesh
http://www.wolfdynamics.com/training/mphase/gt1/poly/ani1.gif

Guided tutorial 1 – Wave impact in a column

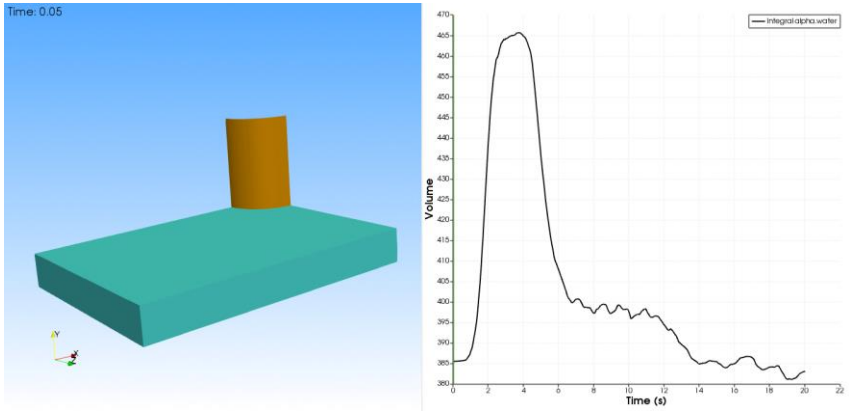
Solution comparison – Hexahedral mesh against polyhedral mesh



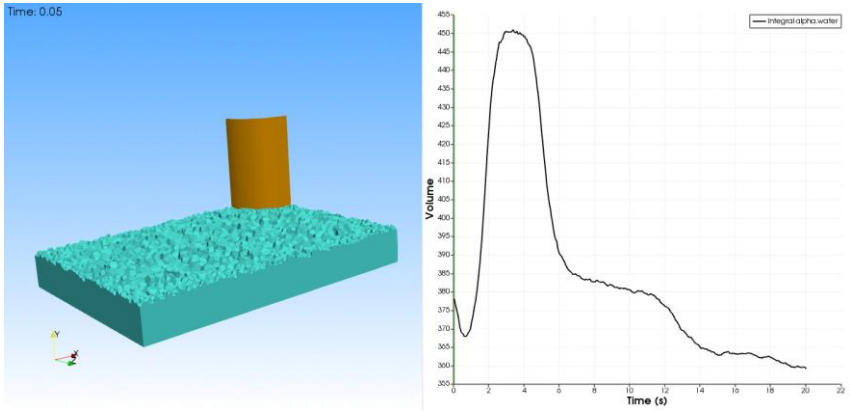
http://www.wolfdynamics.com/training/mphase/gt1/hexa/ani4.gif



http://www.wolfdynamics.com/training/mphase/gt1/poly/ani4.gif



http://www.wolfdynamics.com/training/mphase/gt1/hexa/ani5.gif



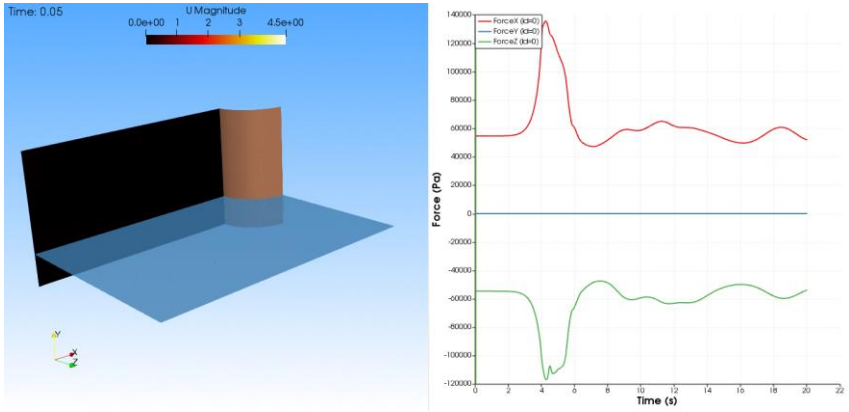
http://www.wolfdynamics.com/training/mphase/gt1/poly/ani5.gif

Hexahedral mesh

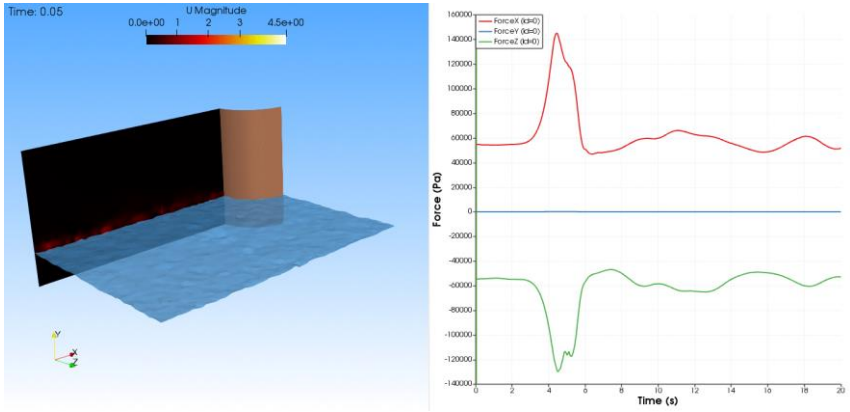
Polyhedral mesh

Guided tutorial 1 – Wave impact in a column

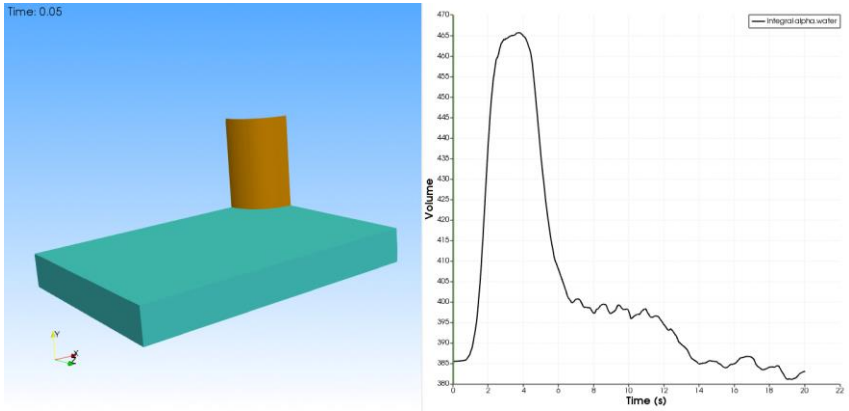
Solution comparison – Hexahedral mesh against tetrahedral mesh



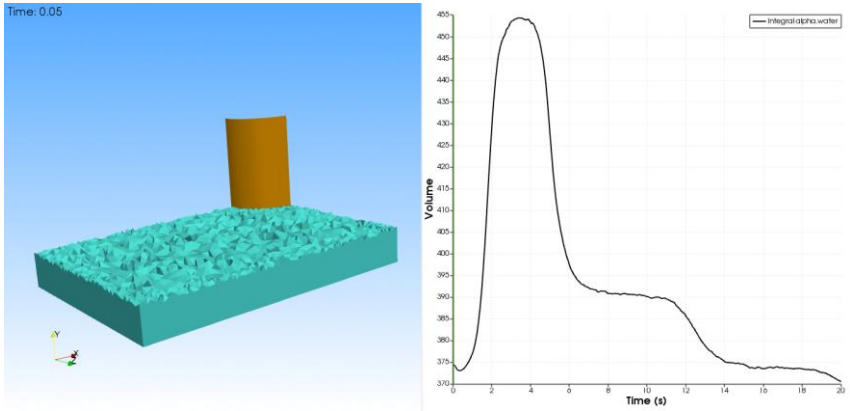
http://www.wolfdynamics.com/training/mphase/gt1/hexa/ani3.gif



http://www.wolfdynamics.com/training/mphase/gt1/tetra/ani3.gif



http://www.wolfdynamics.com/training/mphase/gt1/hexa/ani5.gif



http://www.wolfdynamics.com/training/mphase/gt1/tetra/ani5.gif

Hexahedral mesh

Tetrahedral mesh

Guided tutorial 1 – Wave impact in a column

- Turbulence kinetic energy can be over-predicted in VOF solvers at the phase interface.
- This well know deficiency of the turbulence models when dealing with multiphase flows can be corrected using multiphase stabilization techniques.
- If you are using the ESI OpenFOAM version, you can use the fvOption multiphaseStabilizedTurbulence option to apply corrections to the turbulence kinetic energy equation and turbulence viscosity field.
- The implementation is based on references [1,2].
- At the following link, you can find the release notes addressing this correction,
  - https://www.openfoam.com/news/main-news/openfoam-v1912/solver-and-physics

[1] Devolder, B., Rauwoens, P., and Troch, P. (2017). Application of a buoyancy-modified k-w SST turbulence model to simulate wave run-up around a monopile subjected to regular waves using OpenFOAM. Coastal Engineering, 125, 81-94.
[2] Larsen, B.E. and Fuhrman, D.R. (2018). On the over-production of turbulence beneath surface waves in Reynolds-averaged Navier-Stokes models J. Fluid Mech, 853, 419-460

Guided tutorial 1 – Wave impact in a column

- Turbulence kinetic energy can be over-predicted in VOF solvers at the phase interface.
- This well know deficiency of the turbulence models when dealing with multiphase flows can be corrected using multiphase stabilization techniques.
- If you are using OpenFOAM Dev (and soon in OpenFOAM 9), you can use turbulence damping via fvModels.
- This stabilization technique will add an extra source term to the mixture or phase epsilon or omega equation to reduce turbulence generated near a free-surface.
- The implementation is based on reference [1].
- At the following link, you can find the release notes addressing this correction,
  - https://github.com/OpenFOAM/OpenFOAM-dev/commit/4f969432ad715603043e0e38abfd16c831d023db

[1] Frederix, E. M. A., Mathur, A., Dovizio, D., Geurts, B. J., Komen, E. M. J. (2018). Reynolds-averaged modeling of turbulence damping near a large-scale interface in two-phase flow. Nuclear engineering and design, 333, 122-130.

Guided tutorial 1 – Wave impact in a column

- In this case, we are going to focus our attention on the influence of the cell type on the free surface resolution.
- We are also going to illustrate how to do advanced post-processing in paraview.
- Additionally, we will see how to do basic field initialization.
- We will address details of the numerics in the next tutorials.
- At this point, we are ready to run the simulation.
- To run the turbulent case, go to the **`$TM/multiphase/guided_tutorials/GT1/`** directory.
- After running the simulation, launch paraFoam/paraview and load the predefined paraview states located in the directory **`paraview`**.
  - When asked for the `Load State Data File Options`, select `Choose File Names`, and load the file *`GT1.OpenFOAM`* located in the case directory.
  - In the state file, all the steps to conduct the post-processing have been saved.

Guided tutorial 1 – Wave impact in a column

- At this point, we are ready to run the simulation.
- To run the turbulent case, go to the **`$TM/multiphase/guided_tutorials/GT1/`** directory.
- You can use the script `run_all.sh` to run the case automatically.
- Type in the terminal:

    - `$> sh run_all.sh`

- Feel free to open the file `run_all.sh` to see all the commands used.
- Or if you prefer, you can insert the commands manually in the terminal window.

Guided tutorial 1 – Wave impact in a column

- To do field initialization, we use the command `setFields`.
- This command, will overwrite the input dictionaries that we want to initialize, *e.g.*, water volume fraction (*`alpha.water`*) or velocity field (*`U`*).
- It is always recommended to keep a backup of the un-initialized fields.
- To initialize the solution, you need to follow these steps,

```
1. $> sh run_mesh.sh
2. $> cp 0/alpha.water.org 0/alpha.water
3. $> setFields
```

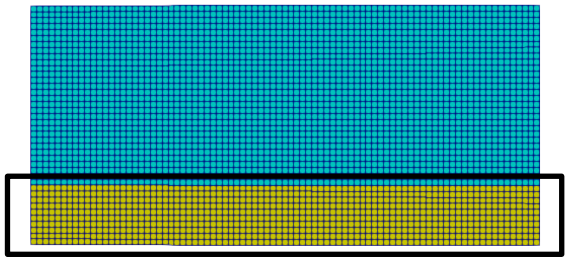
- In step 1, we generate the mesh using the script `run_mesh.sh`.
  - Remember, in order to initialize the solution, you need to have a valid mesh.
- In step 2, we copy the original files (un-initialized fields or *`0/alpha.water.org`*) to the field input file to be initialize (*`alpha.water`* in this case).
  - We do this to keep a backup of the original files, as the file *`0/alpha.water`* will be overwritten when using `setFields`.
- In step 3, we initialize the solution using `setFields`.

Guided tutorial 1 – Wave impact in a column

The `setFieldsDict` dictionary

```
17  defaultFieldValues
18  (
19      volScalarFieldValue alpha.water 0
20  );
21
22  regions
23  (
24      boxToCell
25      {
26          box (-100 -100 -100) (100 2 100);
27          fieldValues
28          (
29              volScalarFieldValue alpha.water 1
30          );
31      }
32
33      boxToFace
34      {
35          box (-100 -100 -100) (100 2 100);
36          fieldValues
37          (
38              volScalarFieldValue alpha.water 1
39          );
40      }
41
42  );
```

- This dictionary file is located in the directory **system**.
- The field to be initialized can be a scalar or a vector (or any of the fields supported in OpenFOAM).
- You can also initialize many fields at the same time. In this case, we are only initializing the field *`alpha.water`*, but if you want, you can also initialize the velocity field (have in mind that this is a vector).
- In lines 17-20 we set the default value of the field in the whole domain (*`alpha.water`* in this case).
- In lines 22-42, we initialize regions. You can initialize as many regions as you like.
- In lines 24-31 we initialize a rectangular region (cell values).
  - In this case, we are using a box. The input values are the absolute minimum and maximum coordinates of the region.
- In lines 33-40 we apply the same initialization to the face boundaries.
- There are many initialization types implemented in OpenFOAM.
  - Use the banana trick to find the options available or read the source code.



boxToCell region

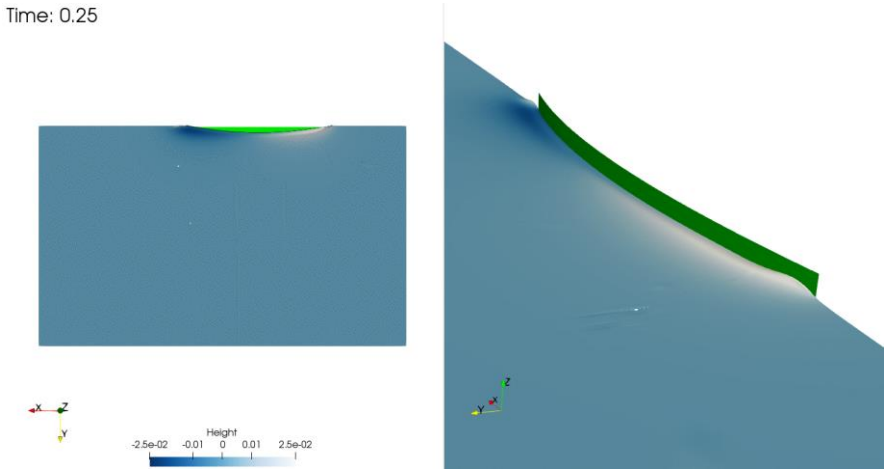
Guided tutorials

- **Guided tutorial 2.** Wigley hull – Towing tank. VOF Medium scale.
- This case is ready to run.
- The case is located in the directory:

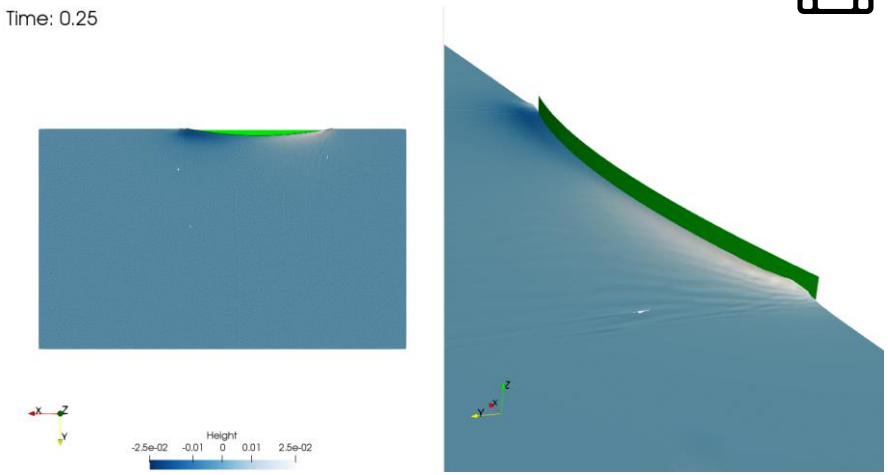
$TM/multiphase/guided_tutorials/GT2/

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

Guided tutorial 2 – Wigley Hull – Towing tank

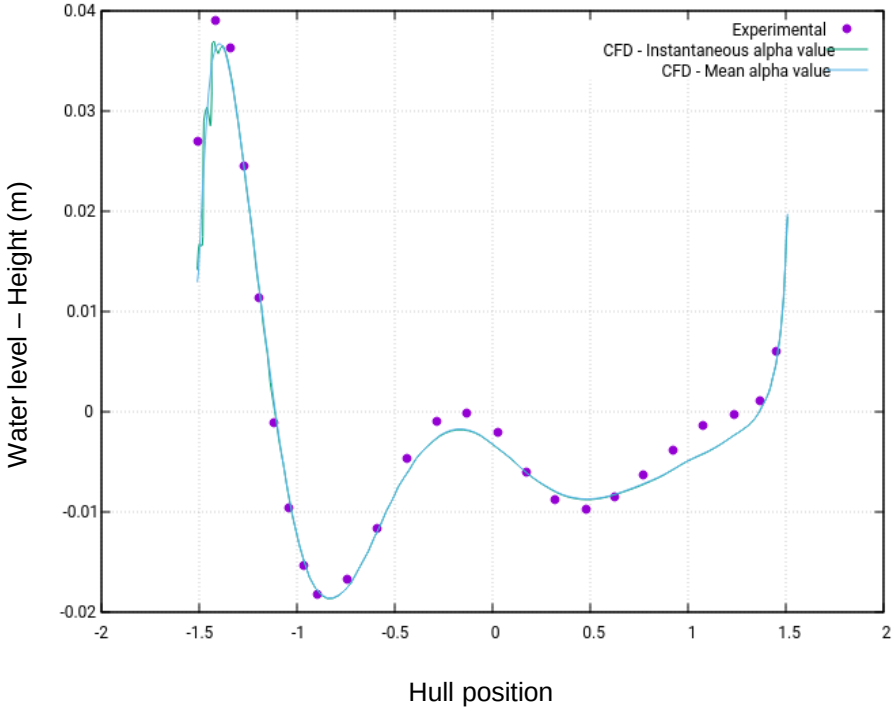


Unsteady simulation
Free surface colored by height – VOF CFL = 1
http://www.wolfdynamics.com/training/mphase/gt2/uns1.gif

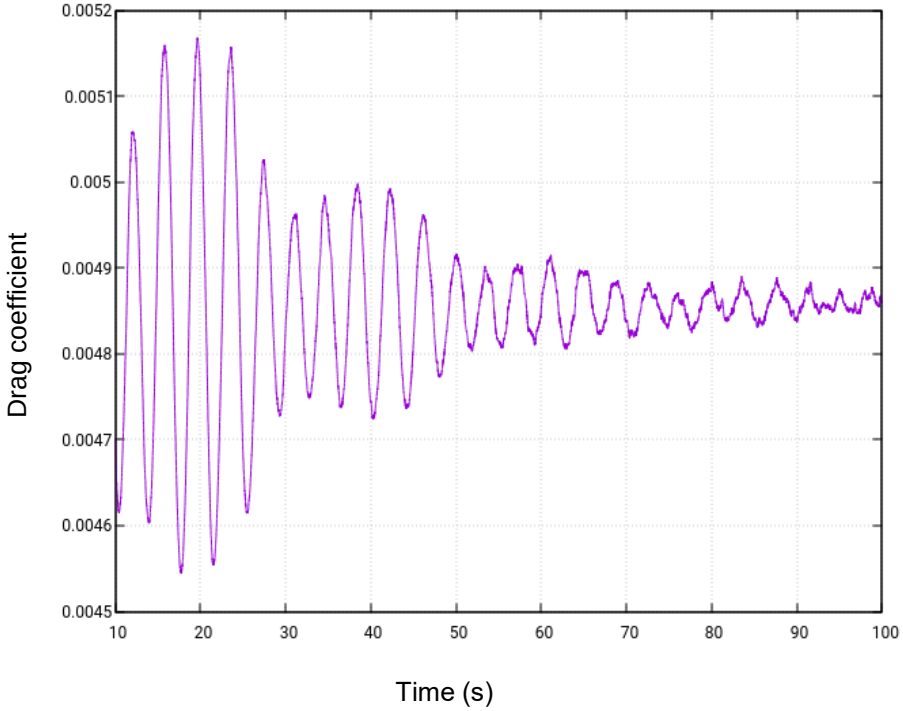


Unsteady simulation
Free surface colored by height – VOF CFL = 10
http://www.wolfdynamics.com/training/mphase/gt2/uns2.gif

Guided tutorial 2 – Wigley Hull – Towing tank



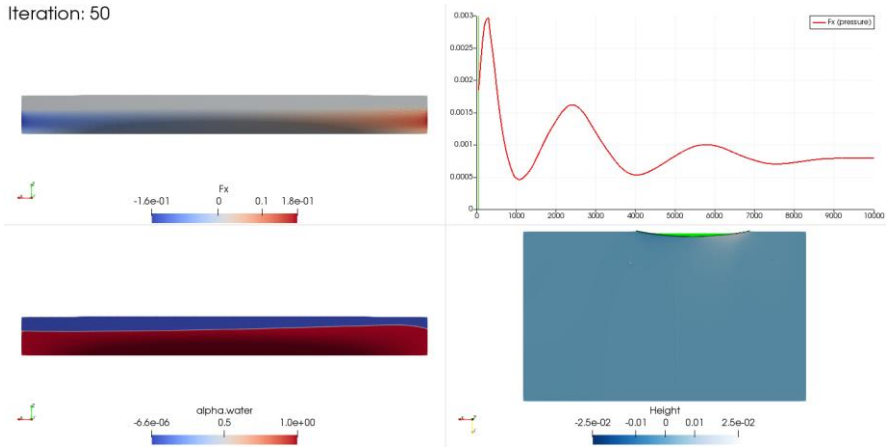
Comparison of water level on hull surface



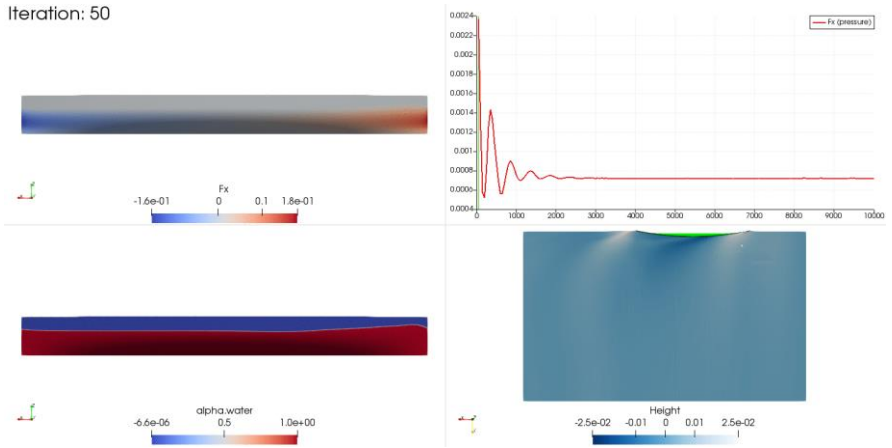
Drag coefficient monitor

Unsteady simulation – VOF CFL =1

Guided tutorial 2 – Wigley Hull – Towing tank

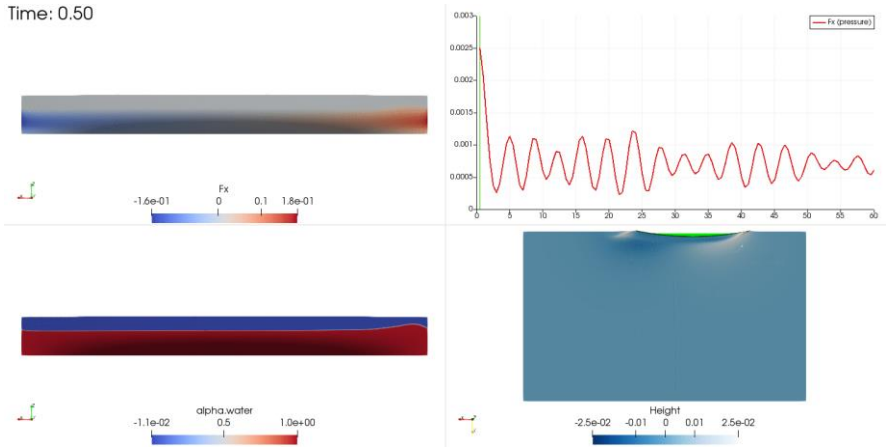


Local-time stepping (LTS) simulation
Free surface colored by height – VOF CFL = 0.9
http://www.wolfdynamics.com/training/mphase/gt2/LTS2.gif

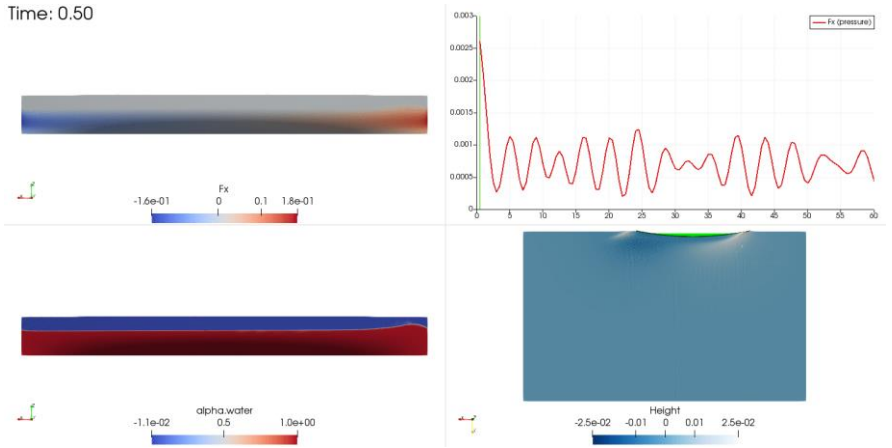


Local-time stepping (LTS) simulation
Free surface colored by height – VOF CFL = 4
http://www.wolfdynamics.com/training/mphase/gt2/LTS1.gif

Guided tutorial 2 – Wigley Hull – Towing tank

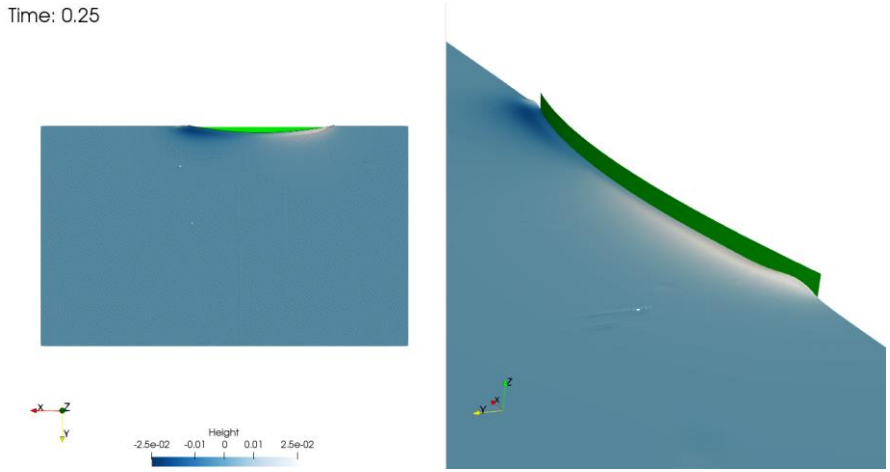


Unsteady simulation – Global time-stepping
Free surface colored by height – VOF CFL = 1
http://www.wolfdynamics.com/training/mphase/gt2/ani_sts.gif

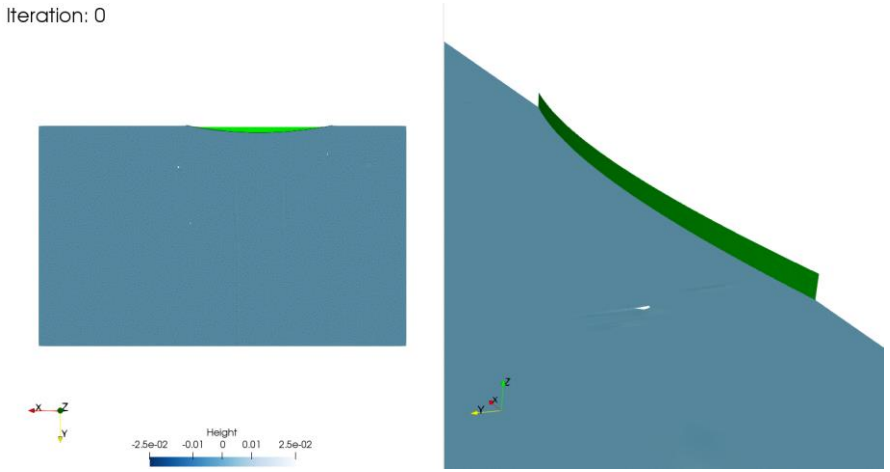


Unsteady simulation – Global time-stepping
Free surface colored by height – VOF CFL = 10
http://www.wolfdynamics.com/training/mphase/gt2/ani_lts.gif

Guided tutorial 2 – Wigley Hull – Towing tank

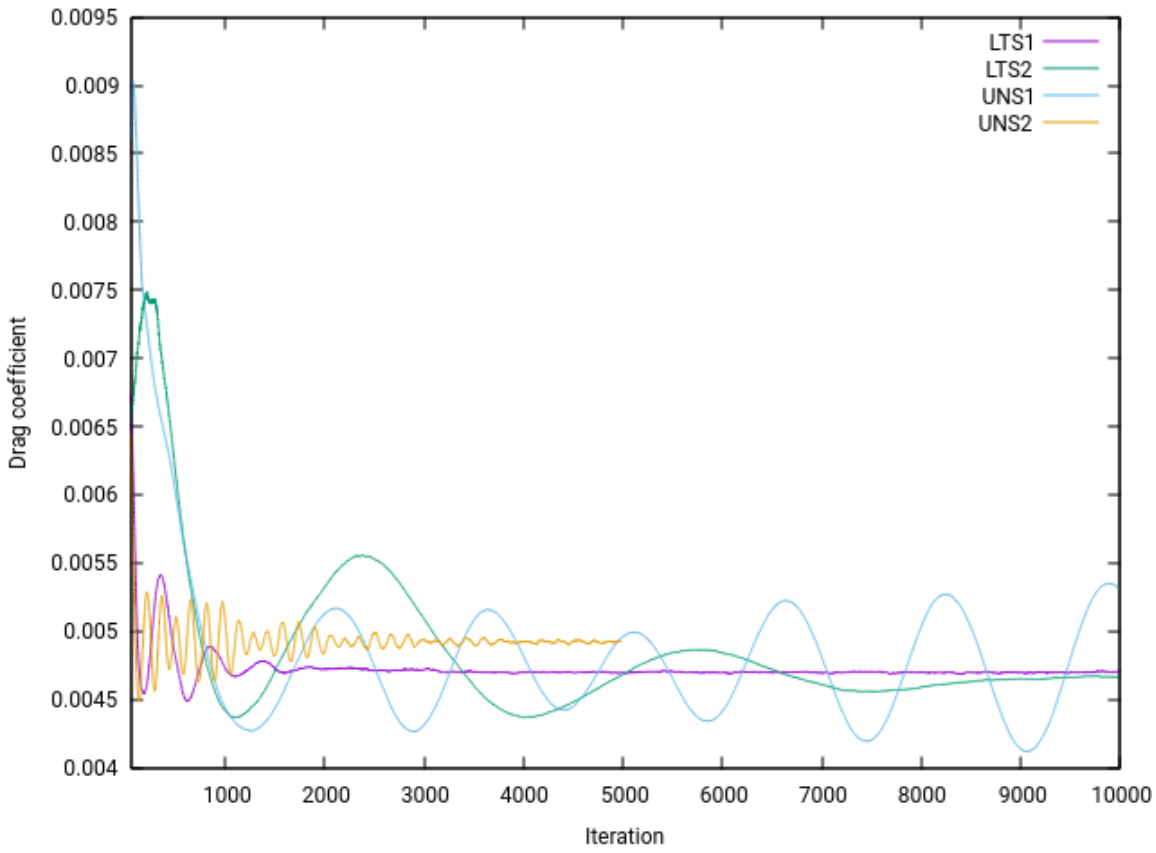


Unsteady simulation
Free surface colored by height – VOF CFL = 1
http://www.wolfdynamics.com/training/mphase/gt2/uns1.gif



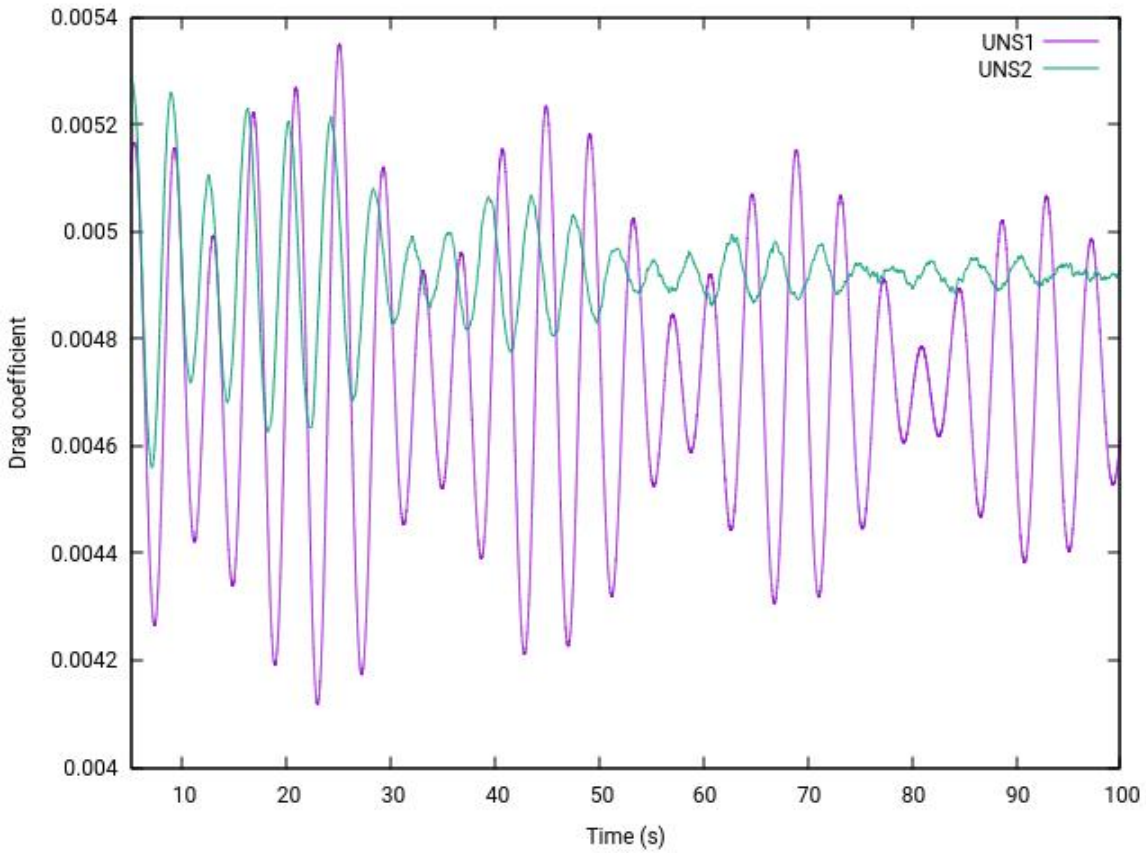
Local-time stepping (LTS) simulation
Free surface colored by height – VOF CFL = 4
http://www.wolfdynamics.com/training/mphase/gt2/LTS1.gif

Guided tutorial 2 – Wigley Hull – Towing tank



Case	Time marching method	Max. VOF CFL	Max. flow CFL	LTS smoothing	Max. time step (s)	Clock time - 4 cores (s)
LTS1	Local time-stepping	4	10	0.1	1	11853
LTS2	Local time-stepping	0.9	0.9	0.1	1	9547
UNS1	Global time-stepping - Unsteady	1	10	-	0.1	51961
UNS2	Global time-stepping - Unsteady	10	20	-	0.1	13161

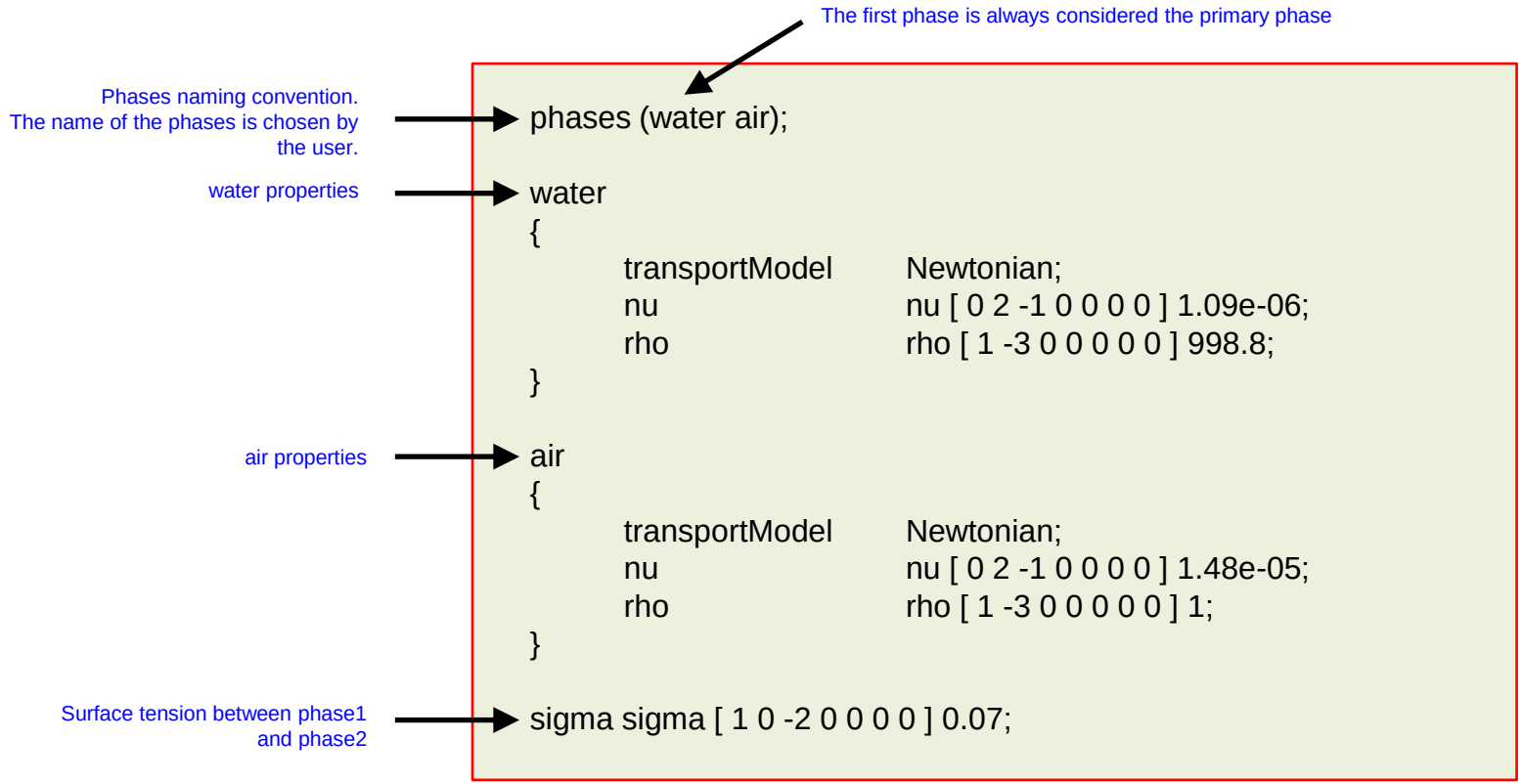
Guided tutorial 2 – Wigley Hull – Towing tank



Case	Time marching method	Max. VOF CFL	Max. flow CFL	Max. time step	Clock time - 4 cores (s)
UNS1	Global time-stepping - Unsteady	1	10	0.1	51961
UNS2	Global time-stepping - Unsteady	10	20	0.1	13161

Guided tutorial 2 – Wigley Hull – Towing tank

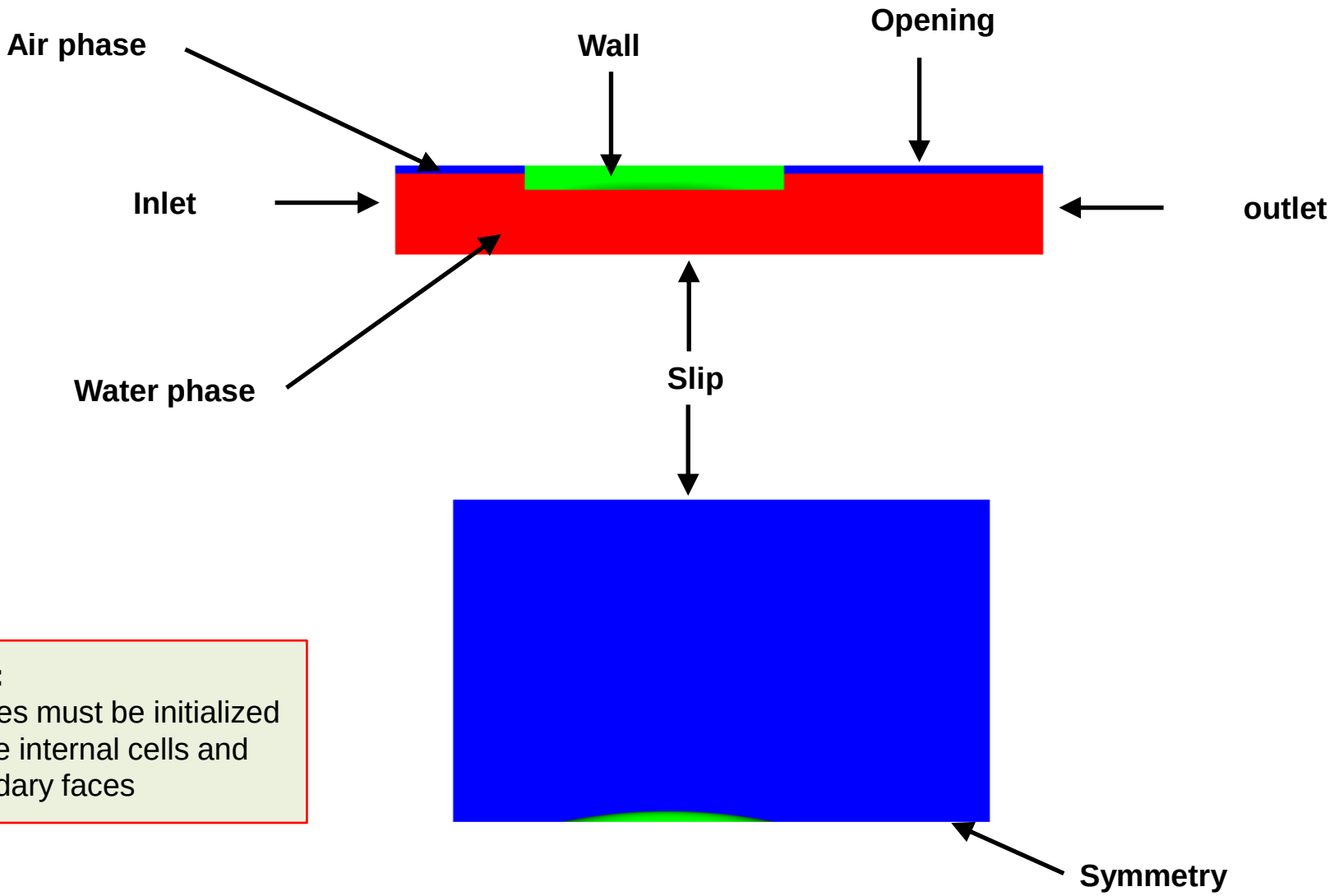
- We are going to use the following solver: **interFoam**
- The first step is to set the physical properties. In the dictionary *constant/transportProperties* we defined the phases.
- Go to the directory **constant** and open the dictionary *transportProperties*.



Guided tutorial 2 – Wigley Hull – Towing tank

- The next step is to set the boundary conditions and initial conditions.
- Therefore, in the directory **0** we define the dictionary *alpha.water* that will take the values of the phase water.
- In this case, you will find the directory **0_org**, here is where we keep a backup of the original files as we are doing field initialization using `setFields`.
- In the directory **0**, you will find the dictionary *p_rgh*, in this dictionary we set the boundary and initial conditions for the pressure field, and the dimensions are in Pascals.
- The turbulence variables values were calculated using an eddy viscosity ratio equal to 1, turbulence intensity equal 5%, and the water properties.
- If you are simulating numerical towing tanks, the setup of the boundary conditions is always the same.
- Feel free to reuse this setup.
- The dictionaries used in this case are standard for the VOF solvers (`interFoam` family solvers).
- Remember, you should always conduct production runs using a second order discretization scheme

Guided tutorial 2 – Wigley Hull – Towing tank



Note:
Phases must be initialized on the internal cells and boundary faces

Physical domain and boundary patches

Guided tutorial 2 – Wigley Hull – Towing tank

Patch name	Pressure	Velocity	Turbulence fields	alpha.water
inflow	fixedFluxPressure	fixedValue	fixedValue calculated (nut)	fixedValue
outflow	inletOutlet or zeroGradient	outletPhaseMeanVelocity	inletOutlet or zeroGradient calculated (nut)	variableHeightFlowRate
bottom	symmetry	symmetry	symmetry	symmetry
midplane	symmetry	symmetry	symmetry	symmetry
side	symmetry	symmetry	symmetry	symmetry
top	totalPressure	pressureInletOutletVelocity	inletOutlet or zeroGradient calculated (nut)	inletOutlet
ship	fixedFluxPressure	fixedValue	kqRWallFunction (k) omegaFunction (omega) nutkWallFunction (nut)	zeroGradient

Typical setup of boundary conditions for numerical towing tank simulations

Guided tutorial 2 – Wigley Hull – Towing tank

- OpenFOAM solves the following modified volume fraction convective equation to track the interface between the phases,

$$\frac{\partial \alpha}{\partial t} + \underbrace{\nabla \cdot \mathbf{U}\alpha}_{} + \underbrace{\nabla \cdot \mathbf{U}_c \alpha(1-\alpha)}_{} = 0$$

$c_\alpha |\mathbf{U}|$

(phi, alpha)
Use a TVD scheme with gradient limiters.
Good choice is the vanLeer scheme.

(phirb, alpha)
Use a high order scheme. The use of interfaceCompression or linear interpolation is fine for this term.

- Where a value of $c_\alpha = 1$ (cAlpha), is recommended to accurately resolve the sharp interface.
- To solve this equation, OpenFOAM uses the semi-implicit MULES* method.
- The MULES options can be controlled in the `fvSolution` dictionary.

* https://openfoam.org/release/2-3-0/multiphase/

Guided tutorial 2 – Wigley Hull – Towing tank

- For interface capturing, OpenFOAM uses,
  - The MULES algorithm (semi-implicit and second order in time) to ensure that the volume fraction (alpha) remains between strict bounds of 0 and 1.
  - The interface compression scheme, based on counter-gradient transport, to maintain sharp interfaces during a simulation.

- At the following link, you can find the release notes related latest developments related to the interface capturing in OpenFOAM 8 (and newer versions),
  - https://cfd.direct/openfoam/free-software/multiphase-interface-capturing/

- Among the latest improvements and developments, was the addition of the Piecewise-linear interface calculation (PLIC) family of interpolation schemes.
  - PLIC represents an interface by surface-cuts which split each cell to match the volume fraction of the phase in that cell.
  - The surface-cuts are oriented according to the point field of the local phase fraction.
  - The phase fraction on each cell face (the interpolated value), is then calculated from the amount submerged below the surface-cut.

Guided tutorial 2 – Wigley Hull – Towing tank

- In addition, the MPLIC or multicut PLIC method was also introduced in OpenFOAM 8.
- Where a single cut is insufficient, MPLIC performs a topological face-edge-face walk to produce multiple splits of a cell.
- If that is still insufficient, MPLIC decomposes the cell into tetrahedrons on which the cuts are applied.
- The extra cutting carries an additional computational cost but requires no fallback to the interface compression method.
- The PLIC and MPLIC methods are more precise than interface compression for meshes with refinement patterns.
- These methods have the potential of removing or reducing the numerical oscillations that appear on the interface due to refinement patterns.
- These oscillations may cause excessive entrainment of air beneath the hull in naval applications, leading to an inaccurate prediction of forces on the hull.
- (M)PLIC reduces these oscillations for a better prediction of the forces.

Guided tutorial 2 – Wigley Hull – Towing tank

- MULES options in the *fvSolution* dictionary.
- The semi-implicit MULES offers significant speed-up and stability over the explicit MULES.

```
"alpha.*"
{
    MULESCorr           yes;
    nAlphaSubCycles     1;

    nAlphaCorr          3;



    nLimiterIter        10;


    alphaApplyPrevCorr  yes;
    ...
}
```

Turn on/off semi-implicit MULES

For semi-implicit MULES use 1. Use 2 or more for explicit MULES.

Number of corrections.
Use 2-3 for slowly varying flows.
Use 3 or more for highly transient, high Reynolds, high CFL number flows.

Number of iterations to calculate the MULES limiter. Use 3-5 if CFL number is less than 3. Use 5-10 if CFL number is more than 3.

Use previous time corrector as initial estimate. Set to yes for slowly varying flows. Set to no for highly transient flows.

Guided tutorial 2 – Wigley Hull – Towing tank

- Additional notes on the *fvSolution* dictionary.

```
PIMPLE
{
    momentumPredictor           yes;

    nOuterCorrectors            1;

    nCorrector                  2;

    nNonOrthogonalCorrectors    1;
    ...
}
```

Set to yes for high Reynolds flows, where convection dominates

Recommended value is 1 if CFL < 1 (equivalent to PISO). Increase to improve the stability of second order time discretization schemes (LES simulations). Increase for highly coupled problems or large CFL.

Recommended to use at least 2 correctors. It improves accuracy and stability.

Recommended to use at least 1 corrector. Increase the value for meshes with high non-orthogonality (mora than 70).

- If you are planning to use large time-steps (CFL number larger than 1), it is recommended to do at least 2 nCorrector and 2 nOuterCorrectors correctors.

Guided tutorial 2 – Wigley Hull – Towing tank

- Additional notes on the *fvSolution* dictionary.

```
PIMPLE
{
    consistent                  yes;
    ...
}

...

relaxationFactors
{
    fields
    {
        ".*" 0.9;
    }
    equations
    {
        ".*" 0.9;
    }
}
```

For extra stability and robustness, it is recommended to use the consistent formulation of the SIMPLE P-V.

- Add implicit (equations) and explicit (fields) under-relaxation factors to get extra stability.
- Do not use too low values as you might lose time accuracy.
- Instead of reducing the URF to values below 0.5, it is better to reduce the time step.

Guided tutorial 2 – Wigley Hull – Towing tank

- Finally, we need to set the discretization schemes
- This is done in the dictionary *fvSchemes*.
- In this dictionary, we set the discretization method for every term appearing in the governing equations.
- Convective terms discretization is set as follows:

This term is related to the volume fraction equation

```
divSchemes
{
    div(rhoPhi,U)                   Gauss linearUpwind grad(U);

    div(phi,alpha)                  Gauss interfaceCompression vanLeer 1;

    div(((rho*nuEff)*dev2(T(grad(U))))      Gauss linear;
}
```

- Notice that we are using a high-resolution scheme for the surface tracking (div(phi,alpha)).
- The new notation if OpenFOAM 8, is meant to improve usability with a simple selection of interpolation scheme in the *fvSchemes* file, among many other things.

Guided tutorial 2 – Wigley Hull – Towing tank

- In OpenFOAM 8, the PLIC method can be enabled as follows,

This term is related to the volume fraction equation

```
divSchemes
{
    div(rhoPhi,U)                   Gauss linearUpwind grad(U);

    div(phi,alpha)                  Gauss PLIC interfaceCompression vanLeer 1;

    div(((rho*nuEff)*dev2(T(grad(U))))      Gauss linear;
}
```

- The basic PLIC method generates a single cut so cannot handle cells in which there are multiple interfaces or where the interface is not fully resolved.
- In those cells, the interpolation reverts to an alternative scheme, typically standard interface compression.
- The PLIC method, with a fallback to interface compression, produces robust solutions and it can run with large time steps.
- The PLIC potentially provides extra accuracy at the cost of an additional computational cost.

Guided tutorial 2 – Wigley Hull – Towing tank

- In addition, the MPLIC or multicut PLIC method was also introduced in OpenFOAM 8.
- The MPLIC method can be enabled as follows,

This term is related to the volume fraction equation

```
divSchemes
{
    div(rhoPhi,U)                   Gauss linearUpwind grad(U);

    div(phi,alpha)                  Gauss MPLIC;

    div(((rho*nuEff)*dev2(T(grad(U))))      Gauss linear;
}
```

- The PLIC and MPLIC methods are more precise than interface compression for meshes with refinement patterns.
- However, the extra cutting carries an additional computational cost.
- Regarding stability, still is a little bit early to give our opinion.

Guided tutorial 2 – Wigley Hull – Towing tank

- In OpenFOAM 7 and earlier, the discretization of the volume fraction convective term was defined as follows,

This term is related to the volume fraction equation

```
divSchemes
{
    div(rhoPhi,U)                   Gauss linearUpwind grad(U);

    div(phi,alpha)                  Gauss vanLeer;
    div(phirb,alpha)                Gauss interfaceCompression;

    div(((rho*nuEff)*dev2(T(grad(U))))      Gauss linear;
}
```

- The new notation if OpenFOAM 8, is meant to improve usability with a simple selection of interpolation scheme in the fvSchemes file, among many other things.

```
div(phi,alpha)      Gauss interfaceCompression vanLeer 1;
```

Guided tutorial 2 – Wigley Hull – Towing tank

- For time discretization we can use an unsteady formulation (Euler in this case).
- This scheme requires setting the time-step, and it should be choosing in such a way that it resolves the mean physics.
- Remember, as the free surface is a strong discontinuity, for stability and good resolution we need to use a CFL less than one for the interface courant.

```
ddtSchemes
{
    default Euler;
}
```

- Hereafter, we are using what is know as global time stepping, that is, the CFL number is limited by the smallest cell.
- The simulation is time-accurate, but it requires a lot of CPU time to reach a steady state (if it reaches one).

Guided tutorial 2 – Wigley Hull – Towing tank

- A way to accelerate the convergence to steady state, is by using local time stepping* (LTS).
- In LTS, the time-step is manipulated for each individual cell in the mesh, making it as high as possible to enable the simulation to reach steady-state quickly.
- When we use LTS, the transient solution is no longer time accurate.
- The stability and accuracy of the method are driven by the local CFL number of each cell.
- To avoid instabilities caused by sudden changes in the time-step of each cell, the local time-step can be smoothed and damped across the domain.
- Try to avoid having local time-steps that differ by several order of magnitudes.
- To enable LTS, we use the localEuler method.

```
ddtSchemes
{
    default localEuler;
}
```

- LTS in OpenFOAM can be used with any solver that supports the **PISO** or **PIMPLE** loop.

* https://openfoam.org/release/2-0-0/steady-state-vof/

Guided tutorial 2 – Wigley Hull – Towing tank

- In the LTS method, the maximum flow CFL number, maximum interface CFL number, and the smoothing and damping of the solution across the cells, can be controlled in the dictionary *fvSolution*, in the sub-dictionary **PIMPLE**.

```
PIMPLE
{
    momentumPredictor           yes;

    nOuterCorrectors            2;
    nCorrector                  3;
    nNonOrthogonalCorrectors    2;

    maxCo                       10;

    maxAlphaCo                  1;

    rDeltaTSmoothingCoeff       0.05;

    rDeltaTDampingCoeff         0.5;

    maxDeltaT                   1;
}
```

NOTE:
Always monitor the solution and avoid having local time-steps that differ by several order of magnitudes (two or more).

Maximum flow Courant

Maximum interface Courant

- Local time step smoothing.
- Bounded between 0 and 1.
- Recommended values are between 0.05 and 0.5.
- Larger values translate into limited smoothing of the local time step.

- Local time step damping.
- Bounded between 0 and 1.
- Recommended value is 0.5
- A value equal to 1 is equivalent to turning it off.

Limit the maximum local time-step size

Guided tutorial 2 – Wigley Hull – Towing tank

- At this point, we are ready to run the simulation.
- Remember to adjust the numerics according to your physics.
- You can choose between running using global time stepping or unsteady (directory **`uns`**) or local time stepping (directory **`LTS`**).
- You will find the instructions of how to run the cases in the file *`README.FIRST`* located in the case directory.

Guided tutorials

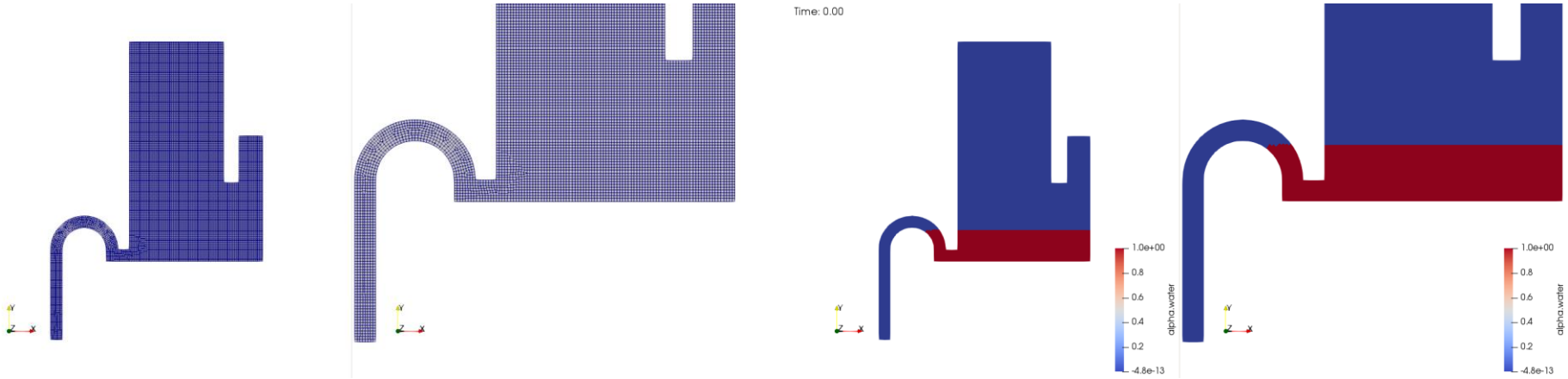
- **Guided tutorial 3.** Syphon effect (Flushing a tank). VOF Small scale.
- This case is ready to run.
- The case is located in the directory:

$TM/multiphase/guided_tutorials/GT3/

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

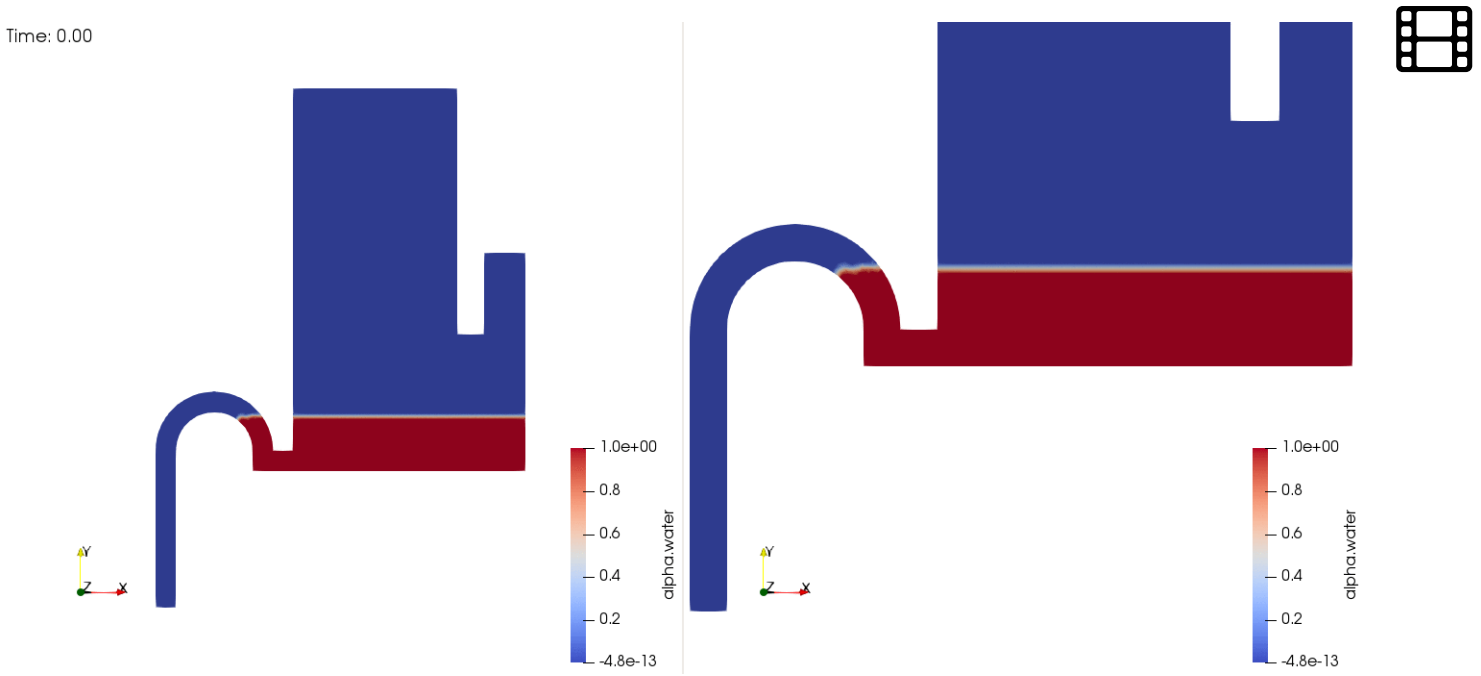
Guided tutorial 3 – Syphon effect (flushing a tank)

What will happen in this case?



- In this guided tutorial, we model a tank emptying due to syphon effect.
- This problem is transient in nature.
- We will use the VOF approach to resolve the interface between the two phases.

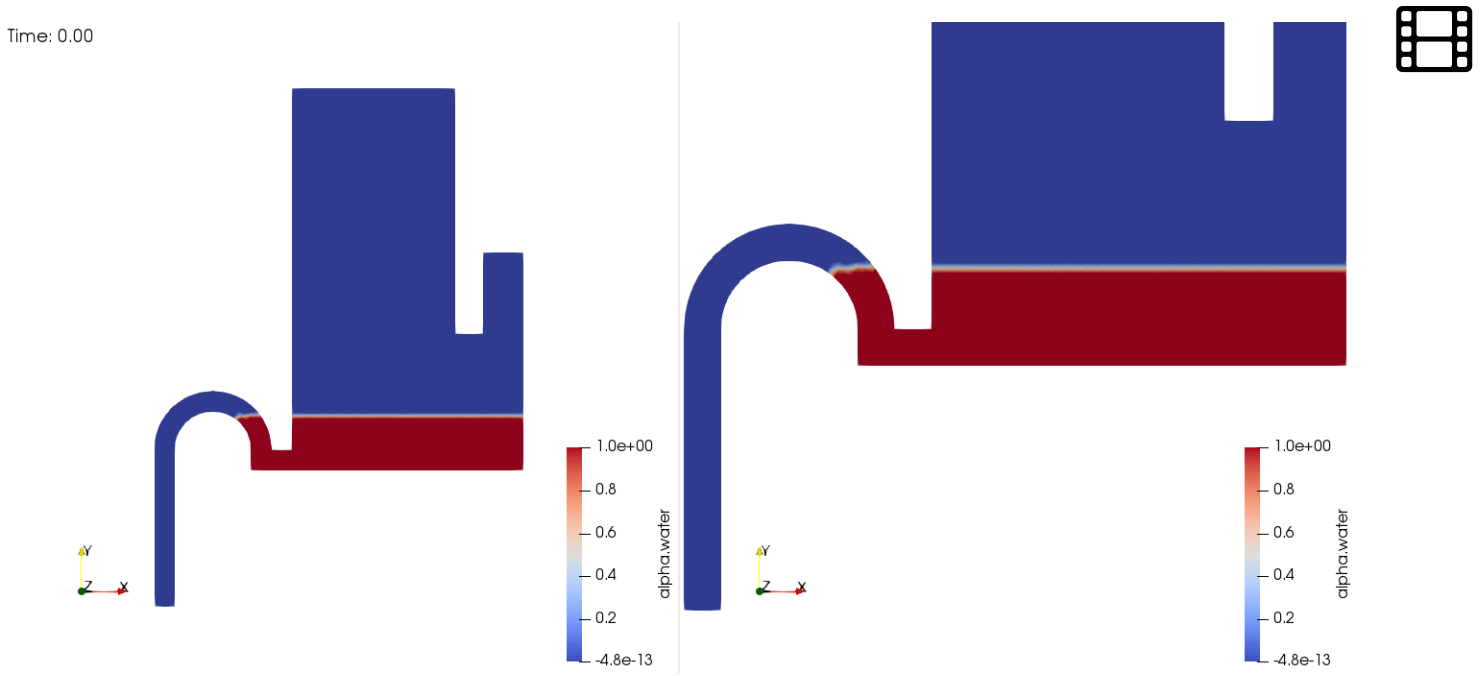
Guided tutorial 3 – Syphon effect (flushing a tank)



http://www.wolfdynamics.com/training/mphase/image34b.gif

- In reality, nothing should happen.
- Maybe some wiggles due to numerical instabilities and mesh alignment.

Guided tutorial 3 – Syphon effect (flushing a tank)

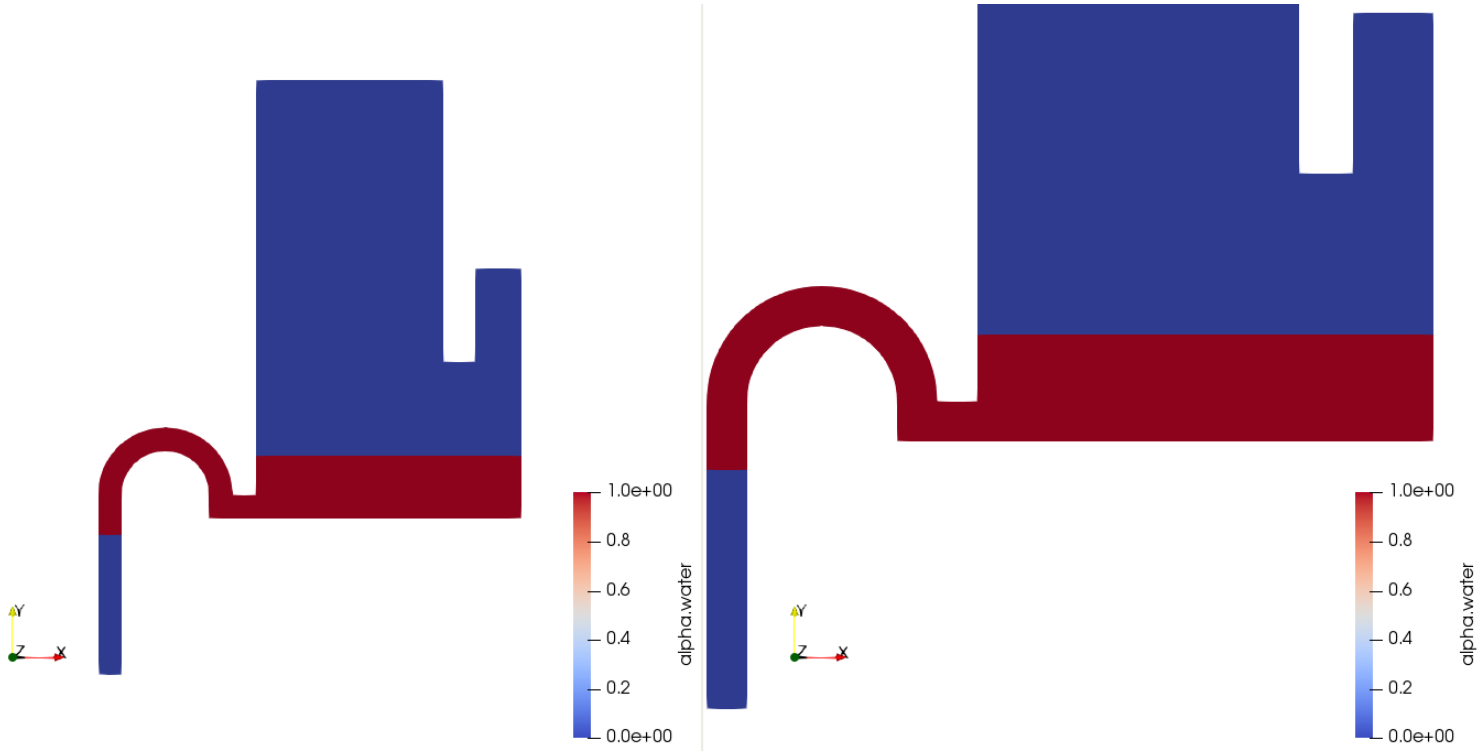


http://www.wolfdynamics.com/training/mphase/image34d.gif

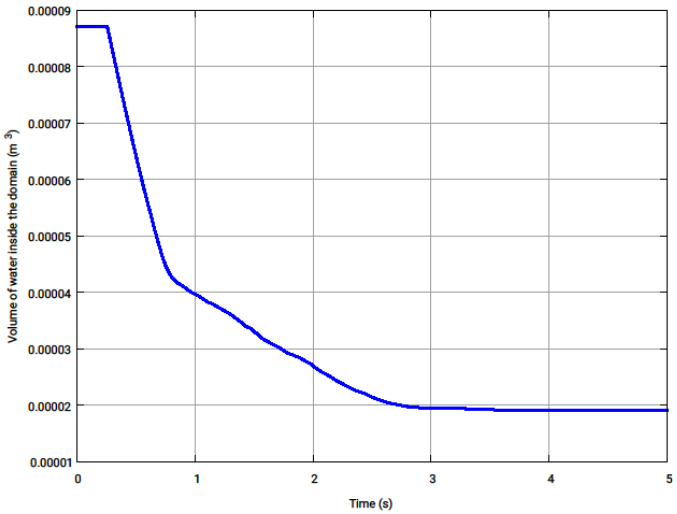
- In this case, there are some small wiggles at the surface.
- These wiggles are due to the effect of the surface tension model and spurious velocity currents at the interface.
- At small scales, these effects can become significant and influence your results.

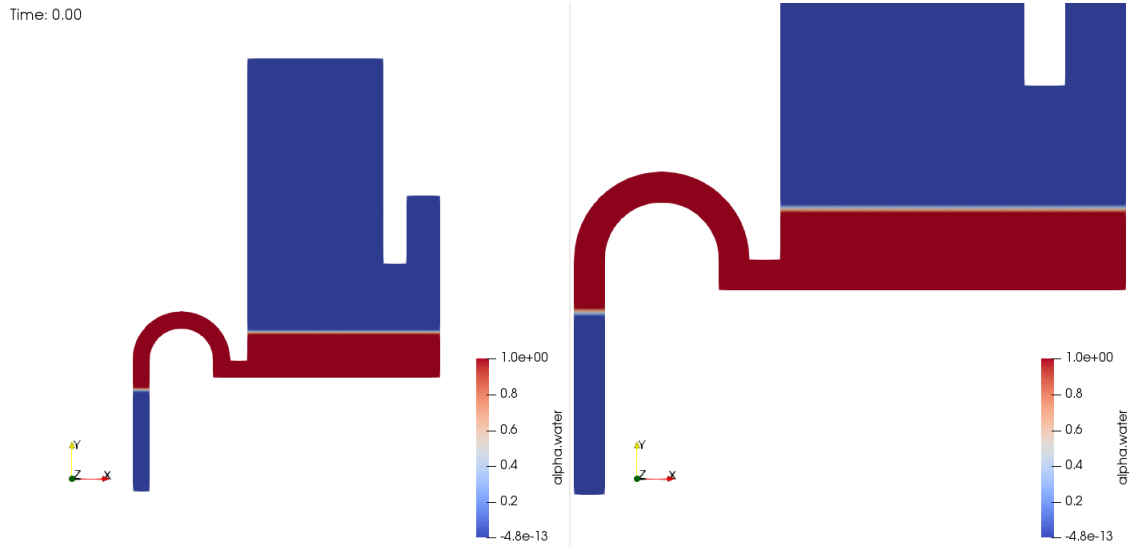
Guided tutorial 3 – Syphon effect (flushing a tank)

What will happen in this case?



Guided tutorial 3 – Syphon effect (flushing a tank)

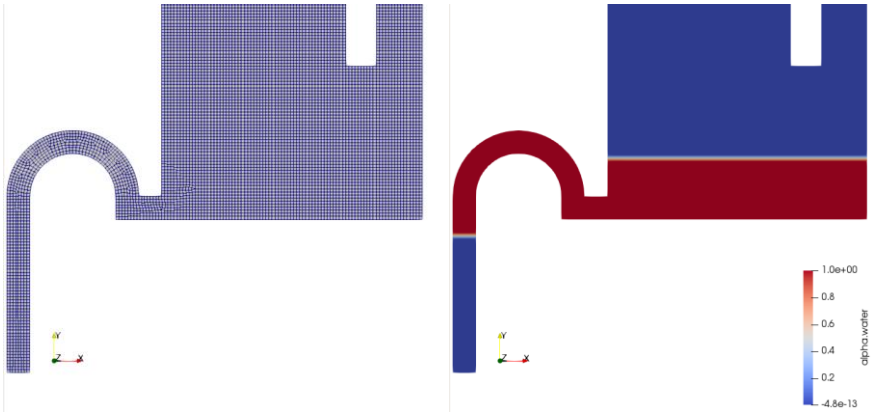




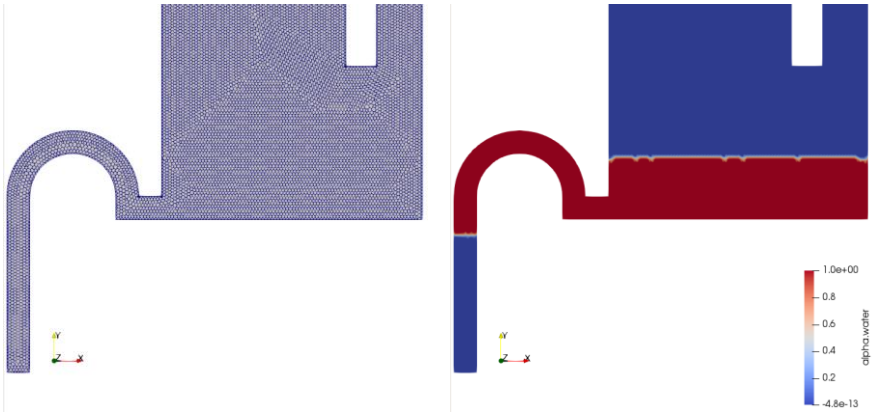
http://www.wolfdynamics.com/training/mphase/image34c.gif

- In this case and due to the initialization used, the tank will be emptied thanks to the syphon effect.
- This problem appears to be easy but there are many things going on.
  - A fast transient, small bubbles, maybe some cavitation, effect of surface tension, turbulence, spurious velocity currents, back flow at the outlet, mesh alignment, and so on.
- All these factors can have a strong effect on the solution stability and accuracy.

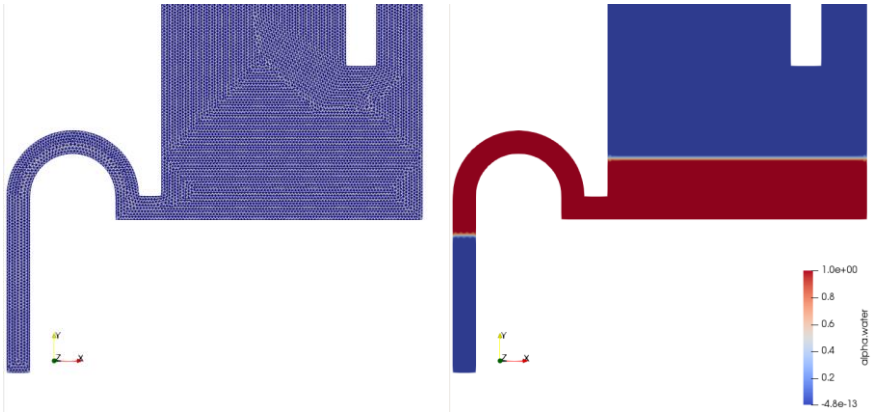
Guided tutorial 3 – Syphon effect (flushing a tank)



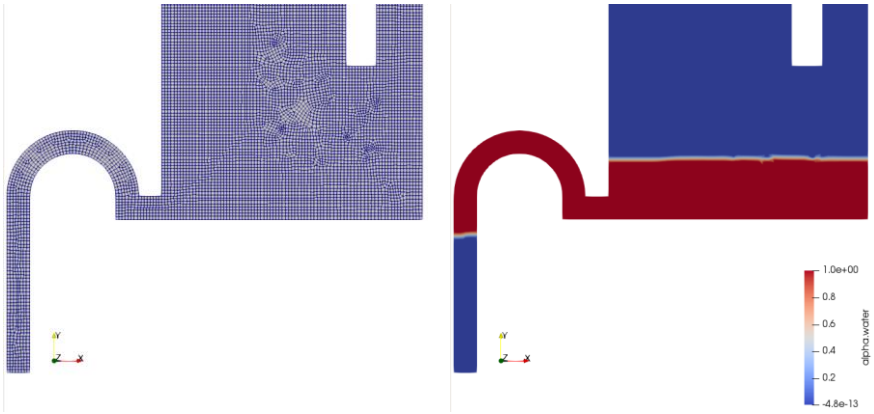
Quad mesh – Free surface aligned with the cells



Polyhedral mesh – Free surface NOT aligned with the cells



Triangular mesh – Free surface aligned with the cells



Quad mesh – Free surface NOT aligned with the cells

- When generating the mesh, try to do your best work so the cells are aligned with the free surface.

Guided tutorial 3 – Syphon effect (flushing a tank)

- We are going to use the following solver: **interFoam**
- Let us explore every dictionary in the case directory.
- The first step is to set the physical properties.
- Go to the directory **`constant`** and open the following dictionaries:
    - `g`: in this dictionary we set the gravity.
    - `transportProperties`: this dictionary contains the material properties for each phase, separated into two blocks (one for water and the other for air).
    - `momentumTransport`: in this dictionary we select the turbulence model to use.
- Remember, you will need to set the boundary conditions, initial conditions, discretization method and solution method for the turbulent variables related to the turbulence model.

Guided tutorial 3 – Syphon effect (flushing a tank)

- The next step is to set the boundary and initial conditions.
- Go to the directory **0** (or **0_org**) and open the following dictionaries:
  - *U*: in this dictionary we set the boundary and initial conditions for the velocity vector field.
  - *p_rgh*: in this dictionary we set the boundary and initial conditions for the pressure scalar field.
  - *alpha.water.org*: in this dictionary we set the boundary and initial conditions for the alpha scalar field (volume of fraction).
- In the dictionary *constant/transportProperties* we defined the phases water and air, hence in the directory **0** we define the dictionary alpha.water.org that will take the values of the phase water.
- The other phase will take the remaining values.
- We set the boundary and initial conditions for the alpha scalar field (volume of fraction) in the dictionary *alpha.water*.
- The dictionary *alpha.water.org* is a backup of the original dictionary as it will be overwritten when we initialize the fields.

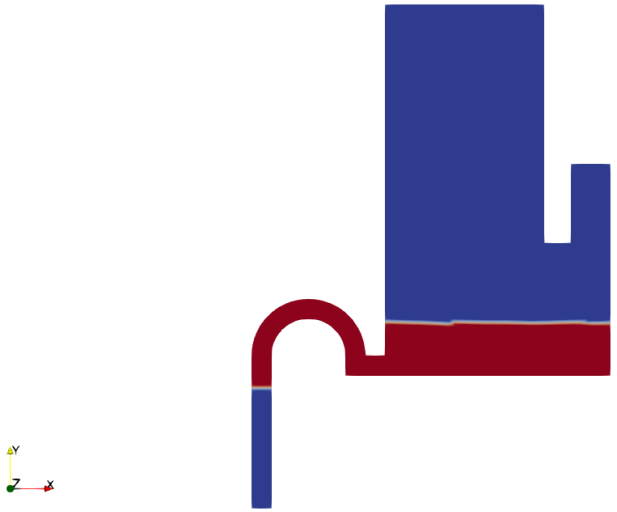
Guided tutorial 3 – Syphon effect (flushing a tank)

- Finally, we set the run-time parameter, numerical method and linear solvers.
- Go to the directory **`system`** and open the following dictionaries:
    - *`controlDict`*: in this dictionary we set general run-time parameters and function objects.
    - *`fvSchemes`*: in this dictionary we set the discretization method.
    - *`fvSolution`*: in this dictionary we set the linear solvers and parameters specific of the pressure-velocity coupling method.

- Do not worry, later on we are going to study in more details the dictionaries *`fvSchemes`* and *`fvSolution`*.

Guided tutorial 3 – Syphon effect (flushing a tank)

- Before proceeding to the simulation stage, we need to do a custom initialization of the fields.
- Go to the directory **`system`** and open the following dictionary:
    - *`setFieldsDict`*: we use this dictionary for custom initialization of the fields



Guided tutorial 3 – Syphon effect (flushing a tank)

- At this point, we are ready to run the simulation.
- To run the tutorial automatically, type in the terminal:

```
1. $> run_all.sh
```

- Feel free to open the file `run_all.sh` to see all the commands used.
- If you prefer to run the case manually, type the commands in the terminal window.

Important note on the VOF method and turbulent flows

- Dealing with turbulent flows using the VOF method is exactly the same as dealing with turbulence in single phase flows.
- When setting boundary and initial conditions for turbulent flows using the VOF method, it is recommended to use the primary phase properties to compute the turbulent quantities.
- If you are simulating a quiescent process, set the turbulent quantities to a small value, e.g., 10e-8.

Guided tutorials

- **Guided tutorial 4.** Surface tension driven flow – Bubble in zero gravity. VOF Small scale.
- This case is ready to run.
- The case is located in the directory:

$TM/multiphase/guided_tutorials/GT4/

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

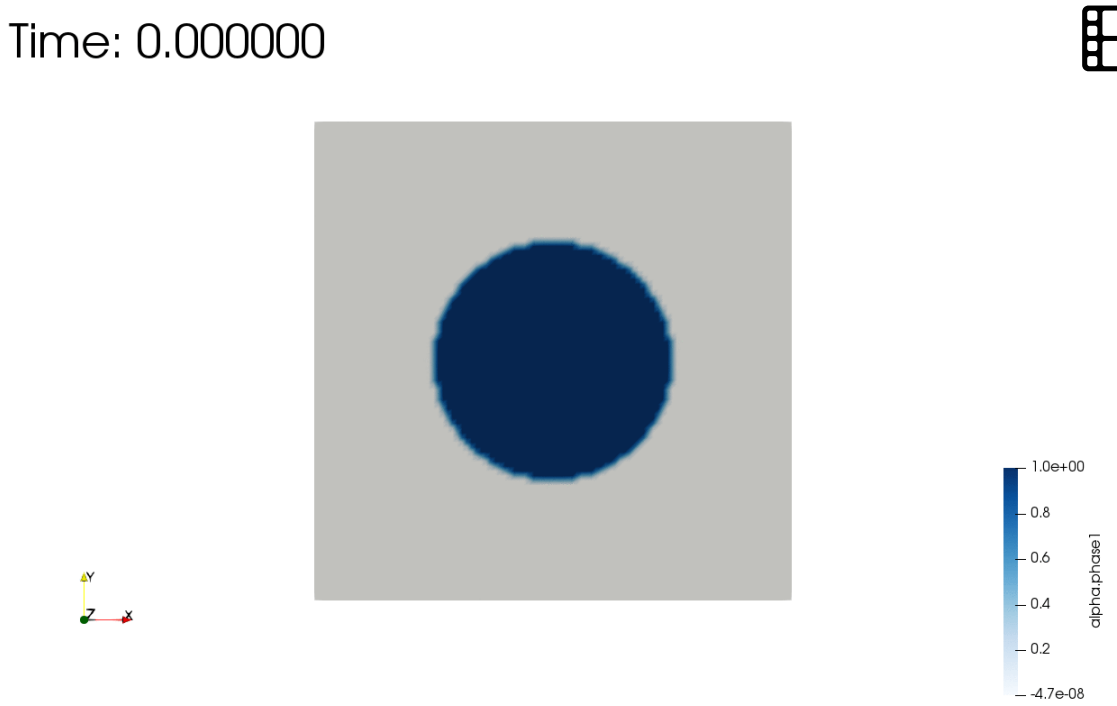
Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- According to you, what will happen to a perfect circular bubble, in perfect equilibrium, in zero gravity, with no perturbations and with given surface tension?

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF



http://www.wolfdynamics.com/training/mphase/image36.gif

- In reality, nothing should happen but due to the mesh, numerical diffusion, and discretization errors the bubble wobbles a little bit.

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- According to you, what will happen to this elliptical bubble in zero gravity, with no perturbations and no surface tension?

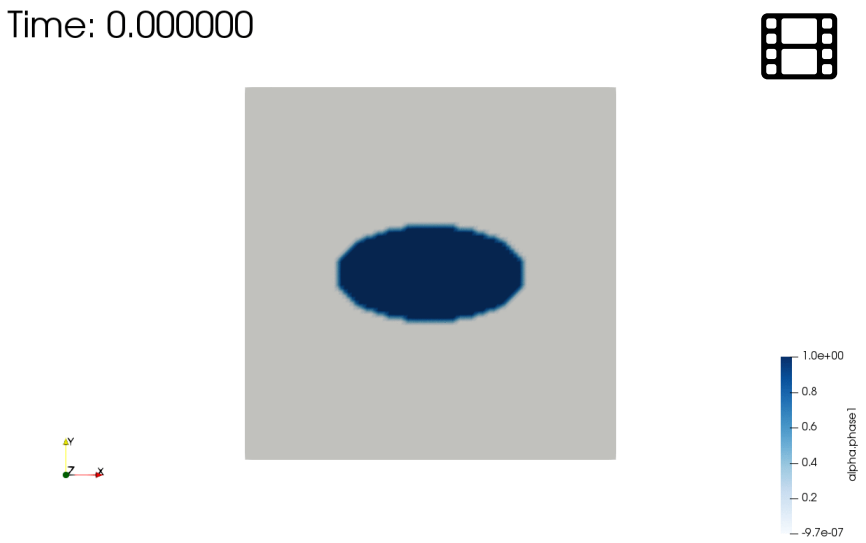
Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- According to you, what will happen to this elliptical bubble in zero gravity, with no perturbations and with given surface tension?

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF



http://www.wolfdynamics.com/training/mphase/image37.gif

- In this guided tutorial we model an elliptical bubble in a zero-gravity field.
- The problem is transient in nature.
- The wobbling is due to surface tension.
- We will use the VOF approach to resolve the interface between the two phases.
- When dealing with surface tension driven flows, numerical accuracy is extremely important as the surface tension forces depend on the curvature of the surface.

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- We are going to use the following solver: **interFoam**
- Let us explore every dictionary in the case directory.
- The first step is to set the physical properties.
- Go to the directory **`constant`** and open the following dictionaries:
  - `g`: in this dictionary we set the gravity.
  - `transportProperties`: this dictionary contains the material properties for each phase, separated into two blocks (one for water and the other for air).
  - `momentumTransport`: in this dictionary we select the turbulence model to use.
- In this case we are not using turbulence model (laminar).
- If you want, you can use a turbulence model. Remember, you will need to set the boundary conditions, initial conditions, discretization method and solution method for the new turbulent variables.

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- The next step is to set the boundary and initial conditions.
- Go to the directory **`0`** (or **`0_org`**) and open the following dictionaries:
  - *`U`*: in this dictionary we set the boundary and initial conditions for the velocity vector field.
  - *`p_rgh`*: in this dictionary we set the boundary and initial conditions for the pressure scalar field.
  - *`alpha.phase1.org`*: in this dictionary we set the boundary and initial conditions for the alpha scalar field (volume of fraction).

- In the dictionary *`constant/transportProperties`* we defined the phases phase1 and phase2, hence in the directory **`0`** we define the dictionary *`alpha.phase1.org`* that will take the values of the phase phase1.
- The other phase will take the remaining values.

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- We set the boundary and initial conditions for the alpha scalar field (volume of fraction) in the dictionary *alpha.water*.
- The dictionary *alpha.water.org* is a backup of the original dictionary as it will be overwritten when we initialize the fields.
- Remember, the name of the **base type** boundary condition (dictionary *constant/polyMesh/boundary*) and the name of the **numerical type** boundary condition (dictionaries located in the directory **0**) need to be the same, if not, OpenFOAM will complain.
- Also, the **base type** and **numerical type** boundary conditions need to be consistent.

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- Finally, we set the run-time parameter, numerical method and linear solvers.
- Go to the directory **`system`** and open the following dictionaries:
    - *`controlDict`*: in this dictionary we set general run-time parameters and function objects.
    - *`fvSchemes`*: in this dictionary we set the discretization method.
    - *`fvSolution`*: in this dictionary we set the linear solvers and parameters specific of the pressure-velocity coupling method.

- Do not worry, later on we are going to study in more details the dictionaries *`fvSchemes`* and *`fvSolution`*.

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- Before proceeding to the simulation stage, we need to do a custom initialization of the fields.
- Go to the directory **`system`** and open the following dictionary:
    - *`setFieldsDict.ellipse`* we use this dictionary for custom initialization of the fields using a STL file

Guided tutorial 4

Surface tension driven flow or bubble in zero gravity – VOF

- At this point, we are ready to run the simulation.
- To run the tutorial automatically, type in the terminal:

```
1. $> run_all.sh
```

- Feel free to open the file `run_all.sh` to see all the commands used.
- If you prefer to run the case manually, type the commands in the terminal window.

Guided tutorials

- **Guided tutorial 5.** Rising bubble. VOF Small scale.
- This case is ready to run.
- The case is located in the directory:

$TM/multiphase/guided_tutorials/GT5/

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

Guided tutorial 5

Rising bubble – VOF

- In this guided tutorial we model a rising bubble.
- The problem is transient in nature.
- The bubble will rise due to buoyancy forces.
- We will use the VOF approach to resolve the interface between the two phases.
- We will use this case to study the numerics.

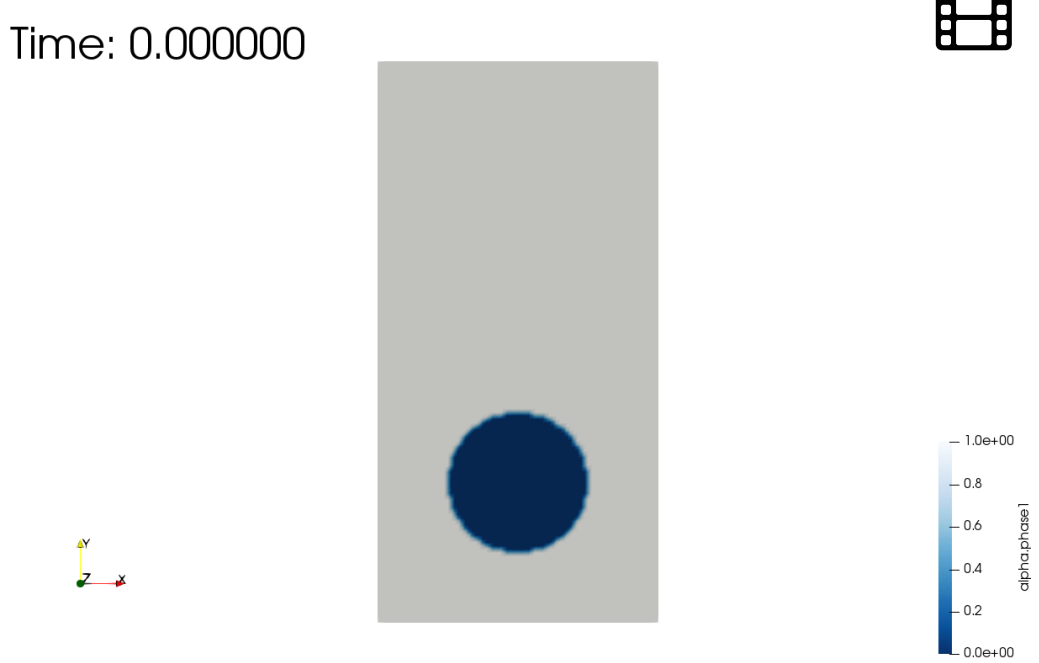
References:

- http://www.featflow.de/en/benchmarks/cfdbenchmarking/bubble.html
- Klostermann, J.; Schaake, K.; Schwarze, R.: Numerical simulation of a single rising bubble by VOF with surface compression, International Journal for Numerical Methods in Fluids, DOI: 10.1002/fld.3692

Guided tutorial 5

Rising bubble – VOF

- In this guided tutorial we model a rising bubble.
- The problem is transient in nature.
- The bubble will rise due to buoyancy forces.
- We will use the VOF approach to resolve the interface between the two phases.
- We will use this case to study the numerics.



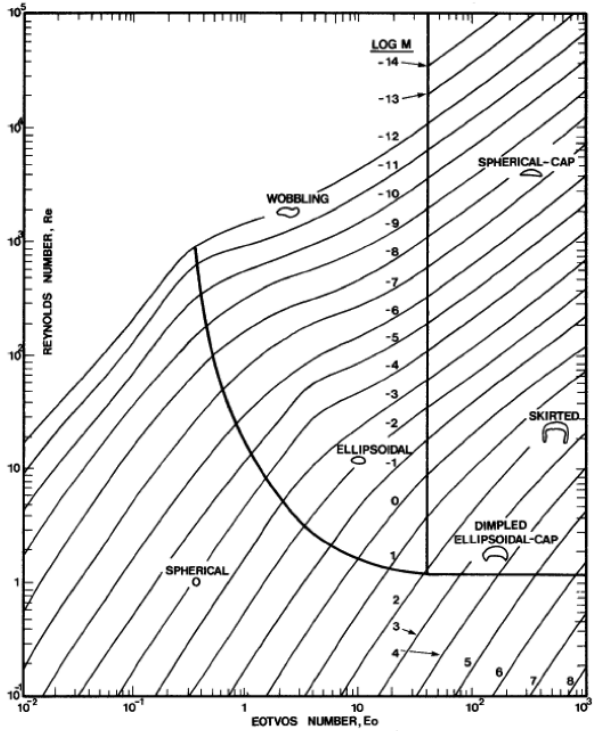
http://www.wolfdynamics.com/training/mphase/image38.gif

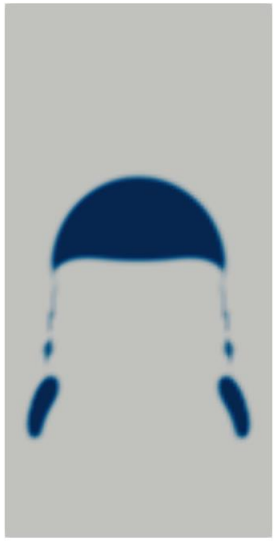
References:

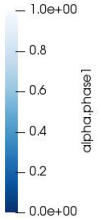
- http://www.featflow.de/en/benchmarks/cfdbenchmarking/bubble.html
- Klostermann, J.; Schaake, K.; Schwarze, R.: Numerical simulation of a single rising bubble by VOF with surface compression, International Journal for Numerical Methods in Fluids, DOI: 10.1002/fld.3692

Guided tutorial 5

Rising bubble – VOF







References:

- http://www.featflow.de/en/benchmarks/cfdbenchmarking/bubble.html
- Klostermann, J.; Schaake, K.; Schwarze, R.: Numerical simulation of a single rising bubble by VOF with surface compression, International Journal for Numerical Methods in Fluids, DOI: 10.1002/fld.3692

Guided tutorial 5

Rising bubble – VOF

- We are going to use the following solver: **interFoam**
- Let us explore the dictionaries in the **`system`** directory.
- Remember:
  - The discretization method is set in the dictionary *`fvSchemes`*.
  - The linear solvers and parameters specific of the pressure-velocity coupling method is set in the dictionary *`fvSolution`*.
  - Runtime parameters such as time-step, CFL number, saving frequency, and so on, are set in the dictionary *`controlDict`*.

- In the folder **`system/default`**, you will find the case default setup.
- All the cases were run in parallel using 4 processors.

Guided tutorial 5

Rising bubble – VOF

- Let us study the dictionary *fvSchemes*.
- In this dictionary we set the discretization method for every term appearing in the governing equations.
- Time discretization is set as follows:

```
ddtSchemes
{
    default CrankNicolson 0;
}
```

- Currently the backward method is not supported for multiphase flows.
- The keyword default means that we use the same method for every requested term.
- Setting the blending factor to 0 is equivalent to the Euler method (first order accurate). Setting the blending factor to 1 is equivalent to use the pure CrankNicolson method (second order accurate).

Guided tutorial 5

Rising bubble – VOF

- Let us study the dictionary *fvSchemes*.
- In this dictionary we set the discretization method for every term appearing in the governing equations.
- You can set the time discretization in a term-by-term basis as follows:

```
ddtSchemes
{
    default none;
    ddt(alpha)      CrankNicolson 0.8;
    ddt(rho,U)      CrankNicolson 0.8;
    ddt(U)          CrankNicolson 0.8;
}
```

Guided tutorial 5

Rising bubble – VOF

- Let us study the dictionary *`fvSchemes`*.
- In this dictionary we set the discretization method for every term appearing in the governing equations.
- Gradient discretization is set as follows:

```
gradSchemes
{
    default Gauss linear;
}
```

Guided tutorial 5

Rising bubble – VOF

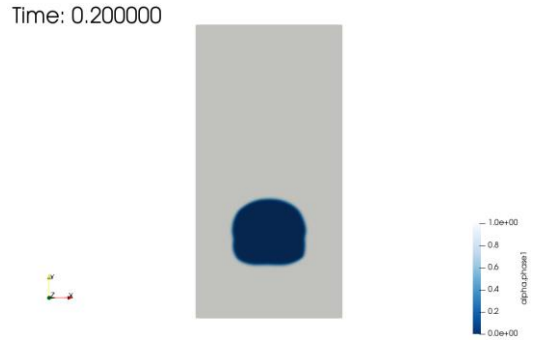
- Let us study the dictionary *fvSchemes*.
- In this dictionary we set the discretization method for every term appearing in the governing equations.
- You can set the gradient discretization in a term-by-term basis as follows:

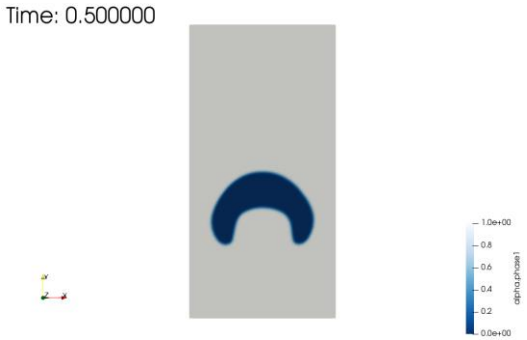
```
gradSchemes
{
    default                 Gauss linear;
    grad(U)                 cellMDLimited Gauss linear 0.5;
    grad(alpha.phase1) cellLimited Gauss linear 1
    grad(alpha.phase2) faceLimited Gauss linear 1
}
```

Guided tutorial 5

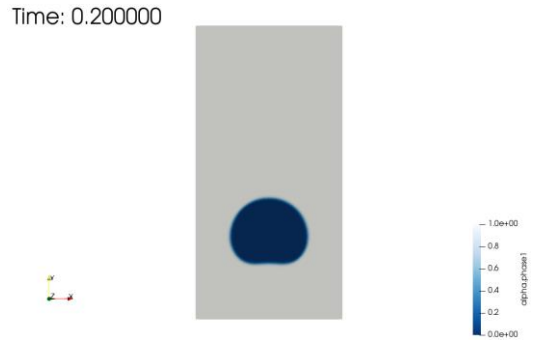
Rising bubble – VOF

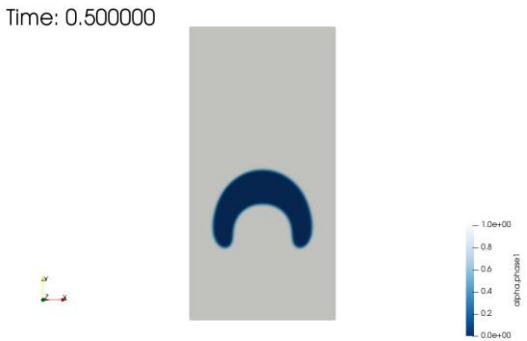
- If you set all the grad terms using the default option, you might get strange interface resolution.
- The difference in the results, is related to the term nHat (face unit interface normal), which should be discretize with no slope limiters.






```
gradSchemes
{
    default cellLimited Gauss linear 1;
}
```



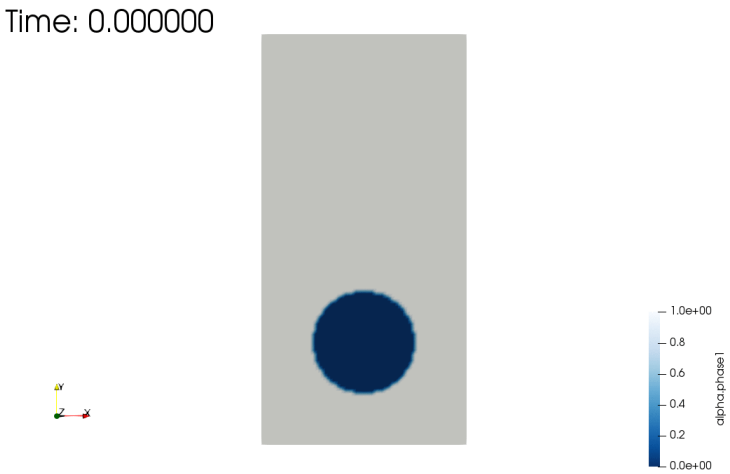



```
gradSchemes
{
    default cellLimited Gauss linear 1;
    nHat Gauss linear;
}
```

Guided tutorial 5

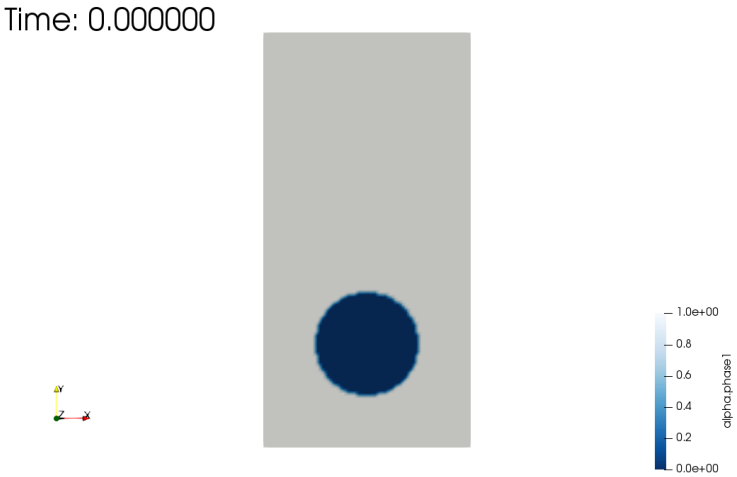
Rising bubble – VOF

- The difference in the results is related to the term nHat (face unit interface normal), which should be discretize with no slope limiters if you want to get a smooth interface.



nHat Gauss linear;

http://www.wolfdynamics.com/training/mphase/nhat1.gif



nHat cellLimited Gauss linear 1;

http://www.wolfdynamics.com/training/mphase/nhat2.gif

Guided tutorial 5

Rising bubble – VOF

- Let us study the dictionary *fvSchemes*.
- In this dictionary we set the discretization method for every term appearing in the governing equations.
- Convective terms discretization is set as follows:

```
divSchemes
{
    div(rhoPhi,U)                   Gauss linearUpwind grad(U);
    div(phi,alpha)                  Gauss interfaceCompression vanLeer 1;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}
```

Guided tutorial 5

Rising bubble – VOF

- Let us study the dictionary *fvSchemes*.
- In this dictionary we set the discretization method for every term appearing in the governing equations.
- Laplacian terms discretization is set as follows:

```
laplacianSchemes
{
    default     Gauss linear limited 1;
}
```

- You can also set the discretization in a term-by-term basis.
- Remember, the choice of the blending factor is related to the quality of the mesh.
- The entry snGradSchemes should use the same method as in laplacianSchemes (limited 1 in this case).

Guided tutorial 5

Rising bubble – VOF

- Comparison of different numerical schemes.

div(phirb,alpha) Gauss linear;

div(phirb,alpha) Gauss vanLeer;

Guided tutorial 5

Rising bubble – VOF

- Comparison of different numerical schemes.

div(phirb,alpha) Gauss upwind;

div(phirb,alpha) Gauss vanLeer;

Guided tutorial 5

Rising bubble – VOF

- Let us study the dictionary *`fvSolution`*.
- In this dictionary we set the linear solvers and parameters specific of the pressure-velocity coupling method.
- The solution of the phasic volume of fraction is set as follows:

```
"alpha.phase1.*"
{
    nAlphaCorr          2;
    nAlphaSubCycles     1;
    cAlpha              1;

    MULESCorr           yes;
    nLimiterIter        3;

    solver              smoothSolver;
    smoother            symGaussSeidel;
    tolerance           1e-8;
    relTol              0;
}
```

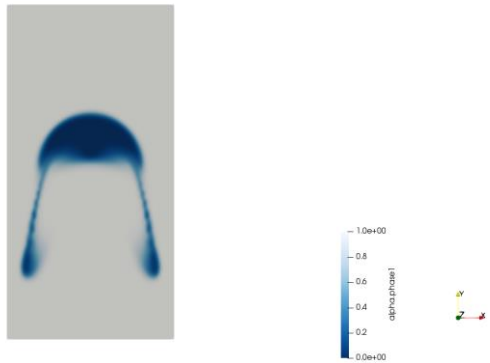
Guided tutorial 5

Rising bubble – VOF

- Comparison of different solution methods for phasic phases.

Note: this parameter controls the sharpness of the interface.

cAlpha 1;



cAlpha 0;

cAlpha 2;

Guided tutorial 5

Rising bubble – VOF

- Comparison of different solution methods for phasic phases.

cAlpha 1;
MULESCorr yes; (semi-implicit formulation)
Execution time = 75 seconds

cAlpha 1;
MULESCorr no; (explicit formulation)
Execution time = 65 seconds

Guided tutorial 5

Rising bubble – VOF

- Let us study the dictionary *fvSolution*.
- In this dictionary we set the linear solvers and parameters specific of the pressure-velocity coupling method.
- The solution of p_rghFinal is set as follows:

```
p_rghFinal
{
    solver          PCG;
    preconditioner  DIC;
    tolerance       1e-08;
    relTol          0.05;
}
```

- The entry p_rghFinal is related to the last iteration of p_rgh. Remember, is a good idea to put more computational effort on this iteration, and advance fast the inner iterations.

Guided tutorial 5

Rising bubble – VOF

- Comparison of different tolerance criterion for p_rgh and p_rghFinal.

Note: not well converged linear solvers can lead to unbounded solutions.

div(phirb,alpha)	Gauss vanLeer;
cAlpha	1;
MULESCorr	yes;
tolerance	1e-08;
relTol	0.5; (p_rgh)
relTol	0.5; (p_rghFinal)
Execution time =	46 seconds

div(phirb,alpha)	Gauss vanLeer;
cAlpha	1;
MULESCorr	yes;
tolerance	1e-08;
relTol	0.01; (p_rgh)
relTol	0; (p_rghFinal)
Execution time =	72 seconds

Guided tutorial 5

Rising bubble – VOF

- Comparison of multigrid vs. Newton-Krylov solvers and preconditioners for p_rgh and p_rghFinal

solver PCG; (p_rgh and p_rghFinal)
preconditioner DIC; (p_rgh and p_rghFinal)
Execution time = 72 seconds

solver GAMG; (p_rgh and p_rghFinal)
Execution time = 73 seconds

Guided tutorial 5

Rising bubble – VOF

- Comparison of multigrid vs. Newton-Krylov solvers and preconditioners for p_rgh and p_rghFinal

solver PCG; (p_rgh and p_rghFinal)
preconditioner DIC; (p_rgh and p_rghFinal)
Execution time = 72 seconds

solver PCG; (p_rgh and p_rghFinal)
preconditioner GAMG; (p_rgh)
preconditioner GAMG; (p_rghFinal)
Execution time = 75 seconds

Guided tutorial 5

Rising bubble – VOF

- Let us study some additional entries that appears in the dictionary files *`fvSolution`*, *`fvSchemes`* *and* *`controlDict`*.
- We will see the influence of many other parameters, such as:
  - momentumPredictor
  - maxCo
  - maxAlphaCo
  - maxDeltaT
  - writeControl
  - adjustTimeStep
  - deltaT

Guided tutorial 5

Rising bubble – VOF

- Parameter → momentumPredictor
- Dictionary file → *fvSolution*

momentumPredictor no;
Execution time = 61 seconds

momentumPredictor yes;
Execution time = 72 seconds

Guided tutorial 5

Rising bubble – VOF

- Parameter → maxCo, maxAlphaCo, maxDeltaT
- Dictionary file → `controlDict`

Note: by increasing the CFL number we get faster outcomes but at the cost of losing accuracy.

maxCo	2;
maxAlphaCo	2;
maxDeltaT	0.1;
MULESCorr	yes;
Execution time =	31 seconds

maxCo	0.1;
maxAlphaCo	0.1;
maxDeltaT	0.1;
MULESCorr	yes;
Execution time =	72 seconds

Guided tutorial 5

Rising bubble – VOF

- Parameter → writeControl
- Dictionary file → *controlDict*

Note: this behavior is stronger at CFL numbers larger than one.

maxCo	2;
maxAlphaCo	2;
maxDeltaT	0.1;
writeControl	adjustableRunTime;
writeInterval	0.01;
MULESCorr	yes;

maxCo	2;
maxAlphaCo	2;
maxDeltaT	0.1;
writeControl	runTime;
writeInterval	0.01;
MULESCorr	yes;

Guided tutorial 5

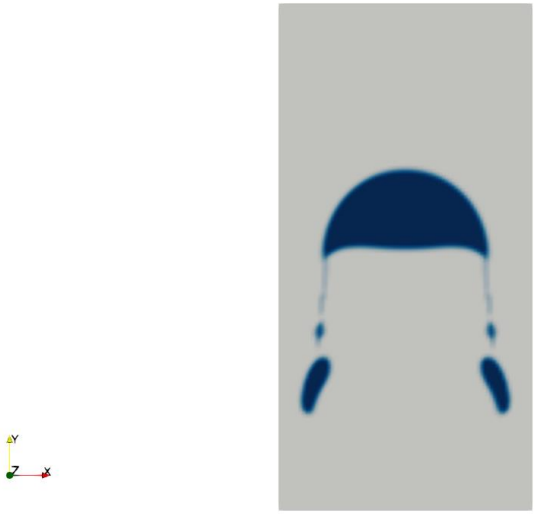
Rising bubble – VOF

- Parameter → writeControl
- Dictionary file → *controlDict*

Note: for some reason, the left case diverges but the main problem here is the use of the explicit scheme.

DIVERGES

- Remember MULESCorr set to no is equivalent to use an explicit scheme.
- Explicit schemes are conditionally stable.



maxCo	2;
maxAlphaCo	2;
maxDeltaT	0.1;
writeControl	runTime;
writeInterval	0.01;
MULESCorr	no;
Execution time =	25 seconds

maxCo	2;
maxAlphaCo	2;
maxDeltaT	0.1;
writeControl	adjustableRunTime;
writeInterval	0.01;
MULESCorr	no;
Execution time =	25 seconds

Guided tutorial 5

Rising bubble – VOF

- Parameter → MULESCorr
- Dictionary file → *fvSolution*

maxCo	2;
maxAlphaCo	2;
maxDeltaT	0.1;
MULESCorr	yes;
Execution time =	31 seconds

maxCo	2;
maxAlphaCo	2;
maxDeltaT	0.1;
MULESCorr	no;
Execution time =	25 seconds

Guided tutorial 5

Rising bubble – VOF

- Parameter → maxCo, maxAlphaCo, maxDeltaT, writeControl
- Dictionary file → *controlDict*

maxCo	0.5;
maxAlphaCo	0.5;
maxDeltaT	0.1;
writeControl	runTime;
writeInterval	0.01;
MULESCorr	no;
Execution time =	20 seconds

maxCo	0.5;
maxAlphaCo	0.5;
maxDeltaT	0.1;
writeControl	adjustableRunTime;
writeInterval	0.01;
MULESCorr	no;
Execution time =	30 seconds

Guided tutorial 5

Rising bubble – VOF

- Parameter → adjustTimeStep
- Dictionary file → *controlDict*

adjustTimeStep no;
deltaT 0.001;
Execution time = 50 seconds

adjustTimeStep yes;
deltaT 0.001;
Execution time = 72 seconds

Guided tutorial 5

Additional notes on the `fvSchemes` and `fvSolution` dictionaries

- The following modified convective equation (alpha) is used to track the interface between the phases,

$$\frac{\partial \alpha}{\partial t} + \underbrace{\nabla \cdot \mathbf{U}\alpha}_{\text{(phi, alpha)}} + \underbrace{\nabla \cdot \mathbf{U}_c \alpha(1-\alpha)}_{\text{(phirb, alpha)}} = 0 \qquad c_\alpha |\mathbf{U}|$$

Use a TVD scheme with gradient limiters.
Good choice is the vanLeer scheme.

(phirb, alpha)
Use a high order scheme. The use of interfaceCompression or linear interpolation is fine for this term.

- Where a value of $c_\alpha = 1$ (cAlpha), is recommended to accurately resolve the sharp interface.

Guided tutorial 5

Additional notes on the `fvSchemes` dictionary

- For interface capturing, OpenFOAM uses,
  - The MULES algorithm (semi-implicit and second order in time) to ensure that the volume fraction (alpha) remains between strict bounds of 0 and 1.
  - The interface compression scheme, based on counter-gradient transport, to maintain sharp interfaces during a simulation.
- At the following link, you can find the release notes related latest developments related to the interface capturing in OpenFOAM 8,
  - https://cfd.direct/openfoam/free-software/multiphase-interface-capturing/
- Among the latest improvements and developments, was the addition of the Piecewise-linear interface calculation (PLIC) family of interpolation schemes.
- The PLIC and MPLIC methods are more precise than interface compression for meshes with refinement patterns.
- These methods carry an additional computational cost.
- Regarding stability, still is a little bit early to give you our opinion

Guided tutorial 5

Additional notes on the *fvSchemes* dictionary

- The new methods can be selected in the *fvSchemes* dictionary as follows,

```
div(phi,alpha)      Gauss PLIC interfaceCompression vanLeer 1;

                    or

div(phi,alpha)      Gauss MPLIC;
```

Guided tutorial 5

Additional notes on the *fvSolution* dictionary

```
"alpha.*"
{
    MULESCorr           yes;
    nAlphaSubCycles     1;

    nAlphaCorr          2;



    nLimiterIter        3;


    alphaApplyPrevCorr  yes;
    …
}
```

- Turn on/off semi-implicit MULES
- For semi-implicit MULES use 1. Use 2 or more for explicit MULES.
- Number of corrections.
  Use 2-3 for slowly varying flows.
  Use 3 or more for highly transient, high Reynolds, high CFL number flows.
- Number of iterations to calculate the MULES limiter. Use 3-5 if CFL number is less than 3. Use 5-10 if CFL number is more than 3.
- Use previous time corrector as initial estimate. Set to yes for slowly varying flows. Set to no for highly transient flows.

- The semi-implicit MULES offers significant speed-up and stability over the explicit MULES

Guided tutorial 5

Additional notes on the *fvSolution* dictionary

```
PIMPLE
{
    momentumPredictor           yes;

    nOuterCorrectors            1;

    nCorrector                  2;

    nNonOrthogonalCorrectors    1;
    ...
}
```

NOTE:
If you are planning to use large time-steps (CFL number larger than 1), it is recommended to do at least 2 nCorrector and 2 nOuterCorrectors correctors.

Set to yes for high Reynolds flows, where convection dominates

Recommended value is 1 if CFL < 1 (equivalent to PISO). Increase to improve the stability of second order time discretization schemes (LES simulations). Increase for highly coupled problems or large CFL.

Recommended to use at least 2 correctors. It improves accuracy and stability.

Recommended to use at least 1 corrector. Increase the value for meshes with high non-orthogonality (mora than 70).

Guided tutorial 5

Additional notes on the *fvSolution* dictionary

```
PIMPLE
{
    consistent                  yes;
    …
}

…

relaxationFactors
{
    fields
    {
        ".*" 0.9;
    }
    equations
    {
        ".*" 0.9;
    }
}
```

For extra stability and robustness, it is recommended to use the consistent formulation of the SIMPLE P-V.

- Add implicit (equations) and explicit (fields) under-relaxation factors to get extra stability.
- Do not use too low values as you might lose time accuracy.
- Instead of reducing the URF to values below 0.5, it is better to reduce the time step.

Guided tutorial 5

Rising bubble – VOF

- At this point, we are ready to run the simulation.
- To run the tutorial automatically, type in the terminal:

```
1. $> run_all.sh
```

- Feel free to open the file `run_all.sh` to see all the commands used.
- If you prefer to run the case manually, type the commands in the terminal window.

Guided tutorials

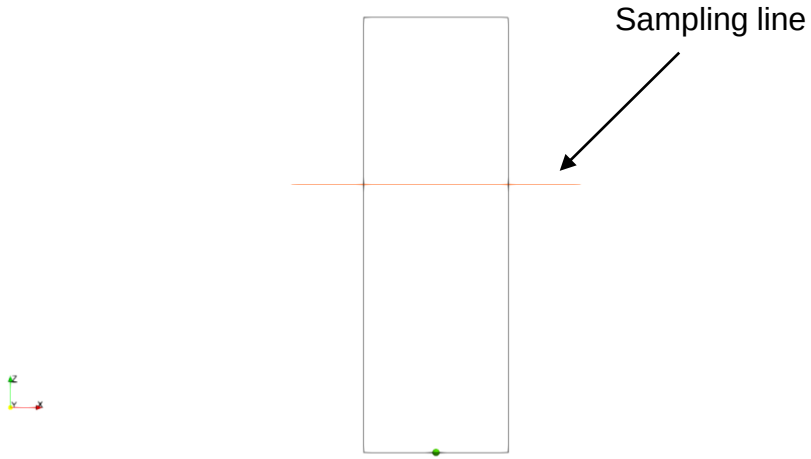
- **Guided tutorial 6.** Bubble column using an Eulerian-Eulerian approach.
- This case is ready to run.
- The case is located in the directory:

$TM/multiphase/guided_tutorials/GT6/

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach



- In this guided tutorial we model a bubble column.
- The problem is transient in nature and is fully 3D.
- Initially, the domain is fill with water and air is injected at the bottom with a given velocity.
- We will use an Eulerian-Eulerian approach to solve the mixture and dynamics of the two phases.
- We are going to use this case to study the different interfacial models implemented in OpenFOAM.
- This is a validation case, we are going to use the same experimental setup as in reference [1].
- To validate the results, we sample the time averaged liquid vertical velocity.

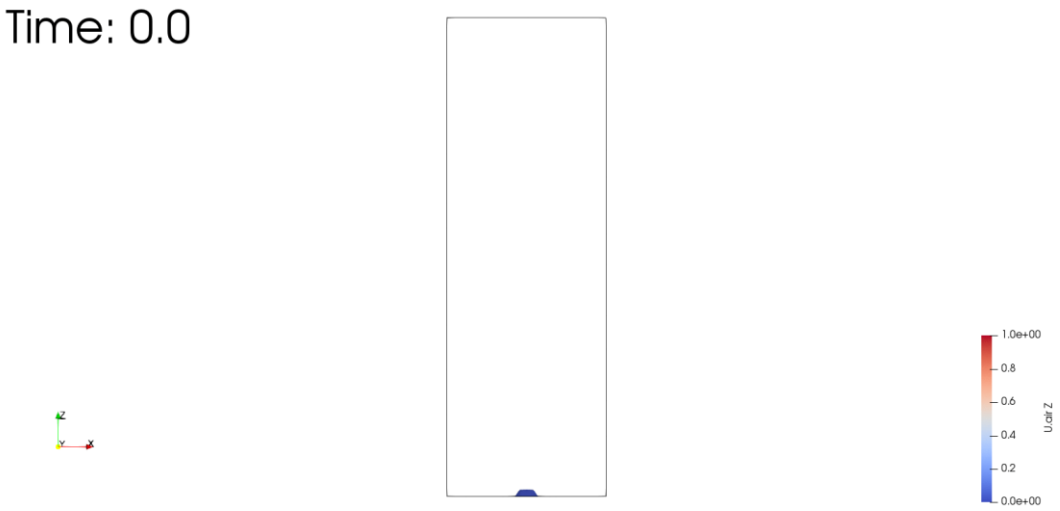
References:

[1] Vivek V. Buwa, Vivek V. Ranade, Dynamics of gas–liquid flow in a rectangular bubble column: experiments and single/multi-group CFD simulations. Chemical Engineering Science 57 (2002) 4715 – 4736

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach

http://www.wolfdynamics.com/wiki/multiphase/ani15.gif



Bubble plume colored by air vertical velocity

- In this guided tutorial we model a bubble column.
- The problem is transient in nature and is fully 3D.
- Initially, the domain is fill with water and air is injected at the bottom with a given velocity.
- We will use an Eulerian-Eulerian approach to solve the mixture and dynamics of the two phases.
- We are going to use this case to study the different interfacial models implemented in OpenFOAM.
- This is a validation case, we are going to use the same experimental setup as in reference [1].
- To validate the results, we sample the time averaged liquid vertical velocity.

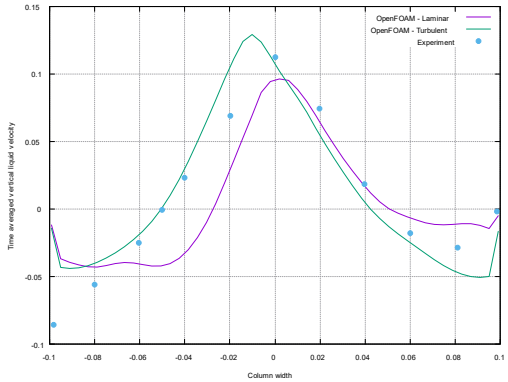
References:

[1] Vivek V. Buwa, Vivek V. Ranade, Dynamics of gas–liquid flow in a rectangular bubble column: experiments and single/multi-group CFD simulations. Chemical Engineering Science 57 (2002) 4715 – 4736

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach

Mean air velocity – Laminar case



Time averaged vertical liquid velocity

Mean air velocity – Turbulent case

- In this guided tutorial we model a bubble column.
- The problem is transient in nature and is fully 3D.
- Initially, the domain is fill with water and air is injected at the bottom with a given velocity.
- We will use an Eulerian-Eulerian approach to solve the mixture and dynamics of the two phases.
- We are going to use this case to study the different interfacial models implemented in OpenFOAM.
- This is a validation case, we are going to use the same experimental setup as in reference [1].
- To validate the results, we sample the time averaged liquid vertical velocity.

References:

[1] Vivek V. Buwa, Vivek V. Ranade, Dynamics of gas–liquid flow in a rectangular bubble column: experiments and single/multi-group CFD simulations. Chemical Engineering Science 57 (2002) 4715 – 4736

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach

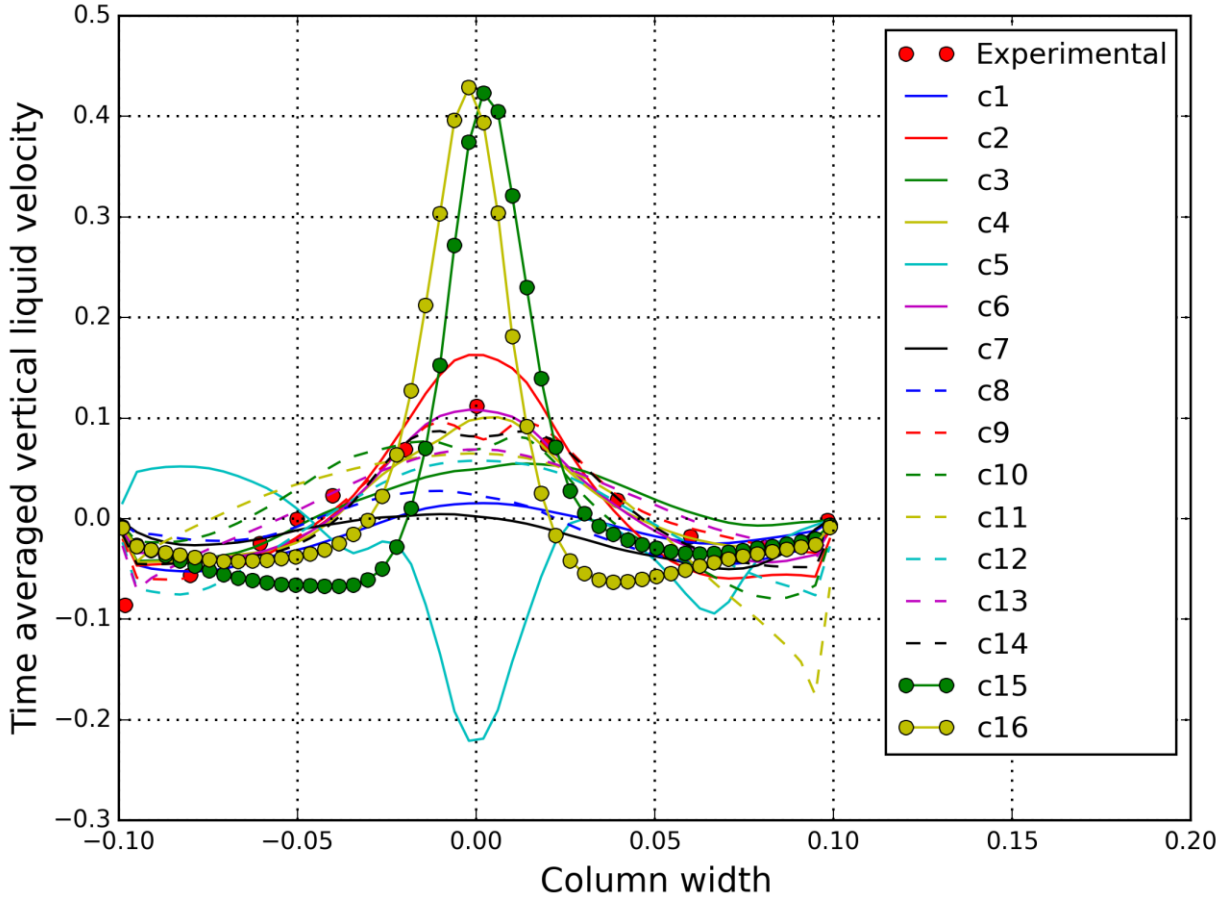
- Cases performed for this validation study:

• C1	DM1-VM-LM1-NFI	NTM	TPF	
• C2	DM1-VM-NFI	NTM	MPF	
• C3	DM1-VM-LM1-WL-NFI	NTM	TPF	
• C4	DM1-VM-NFI	NTM	TPF	
• C5	DM2-VM-NFI	NTM	TPF	
• C6	DM1-VM-FI	NTM	TPF	
• C7	DM1-VM-LM1-FI	NTM	TPF	
• C8	DM1-VM-LM2-FI	NTM	TPF	
• C9	DM1-VM-FI	TM	TPF	RANS1
• C10	DM1-VM-FI-TD1	TM	TPF	RANS1
• C11	DM1-VM-LM3-FI-TD2	TM	TPF	RANS1
• C12	DM1-VM-LM4-FI	NTM	TPF	
• C13	DM3-VM-LM2-FI	NTM	TPF	
• C14	DM1-VM-FI-TD3	TM	TPF	RANS2
• C15	DM1-VM-FI	TM	TPF	LES1
• C16	DM1-VM-FI	TM	TPF	LES2

DM = drag model, VM = virtual mass, LM = lift model, TD = turbulent dispersion, WL = wall lubrication, NFI = no phase inversion, FI = phase inversion, NTM = no turbulence model, TM = turbulence model, TPF = twoPhaseEulerFoam, MPF = multiphaseEulerFoam

Guided tutorial 6

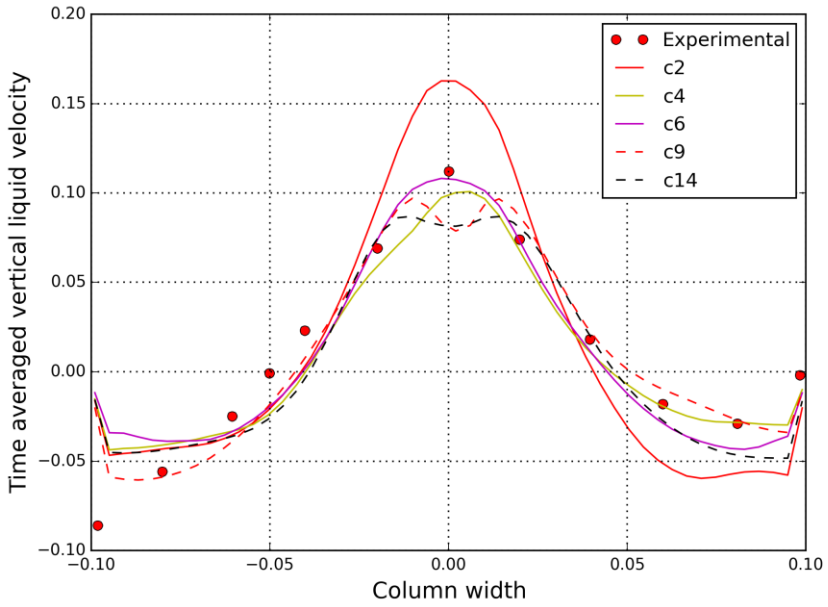
Bubble column – Eulerian-Eulerian approach



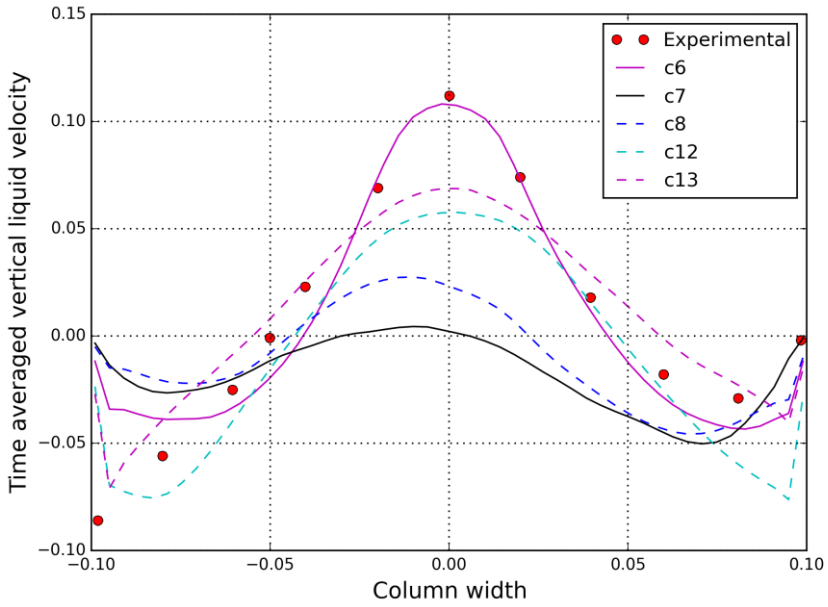
Note: the results presented were obtained using OpenFOAM 3.x

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach



No lift models were used

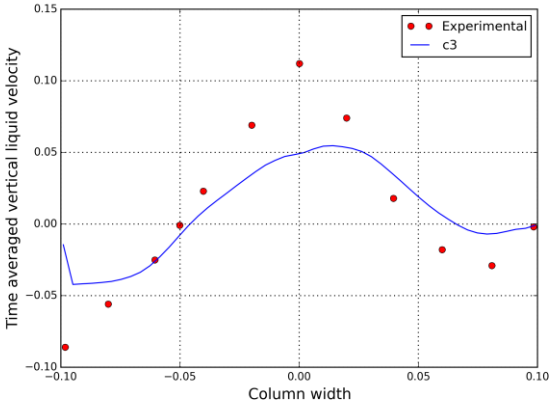


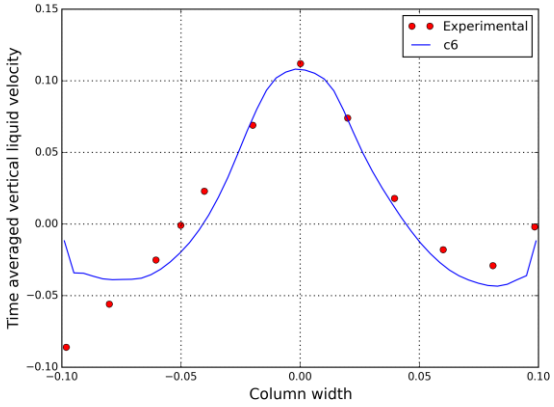
Lift models were used

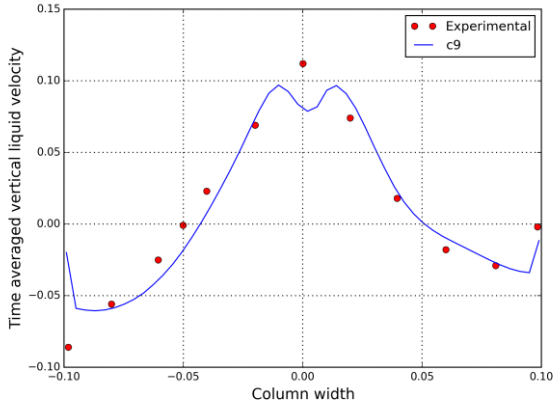
Note: the results presented were obtained using OpenFOAM 3.x

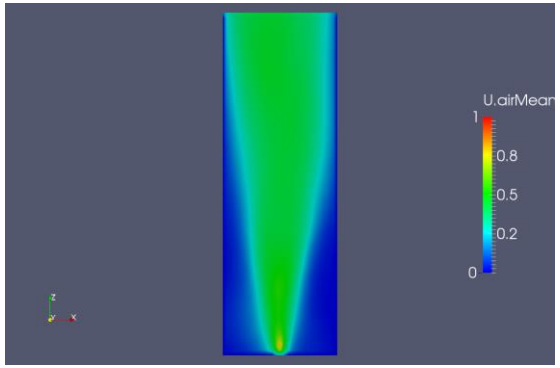
Guided tutorial 6

Bubble column – Eulerian-Eulerian approach

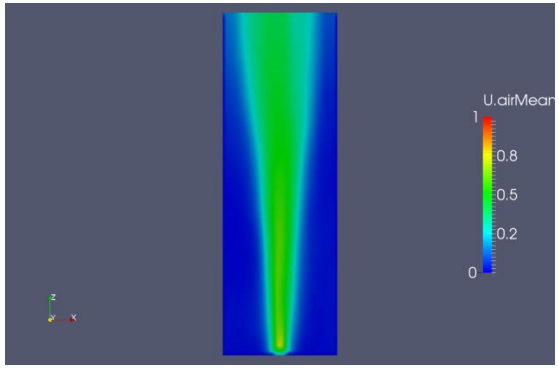




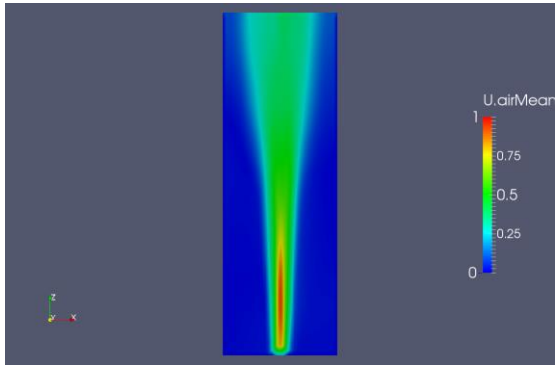




C3
DM1-VM-LM1-WL-NFI-NTM-TPF



C6
DM1-VM-FI-NTM-TPF



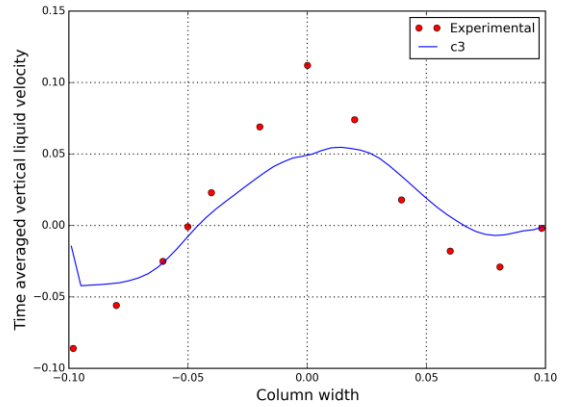
C9
DM1-VM-FI-TM-TPF

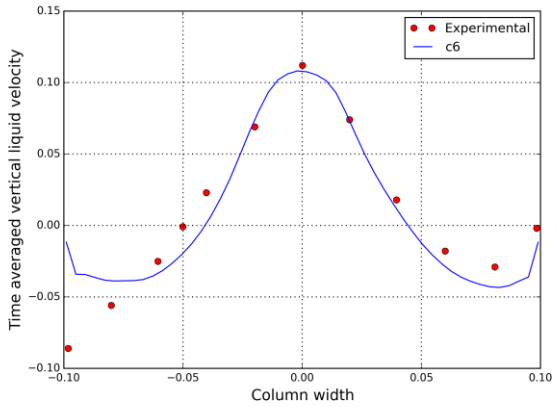
Note: the results presented were obtained using OpenFOAM 3.x

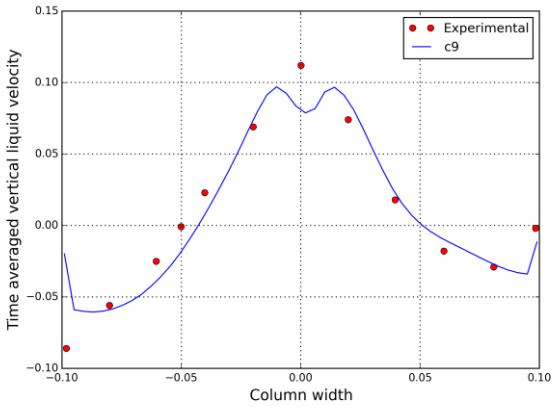
Guided tutorials

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach



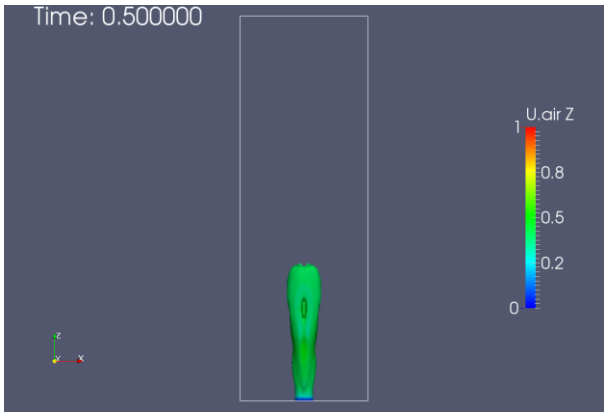


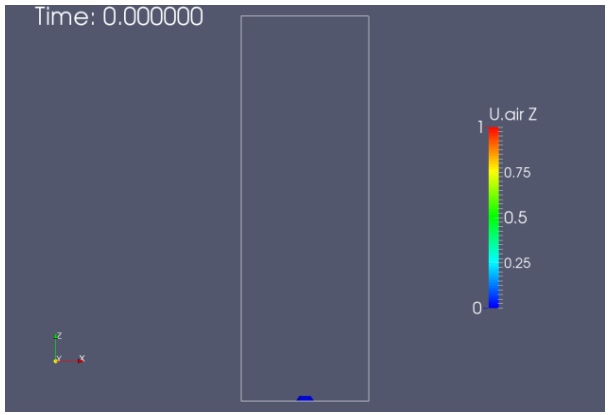


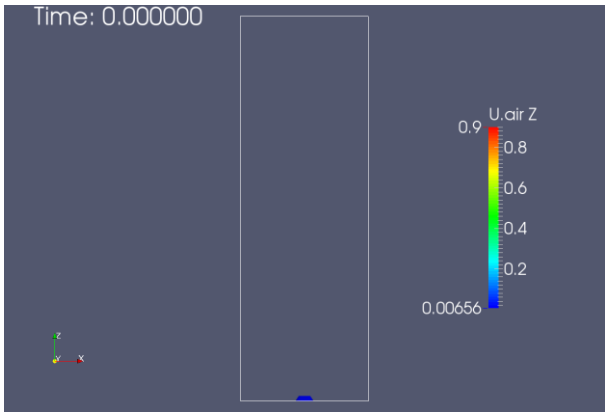
http://www.wolfdynamics.com/wiki/multiphase/ani6.gif

http://www.wolfdynamics.com/wiki/multiphase/ani7.gif

http://www.wolfdynamics.com/wiki/multiphase/ani8.gif







C3
DM1-VM-LM1-WL-NFI-NTM-TPF

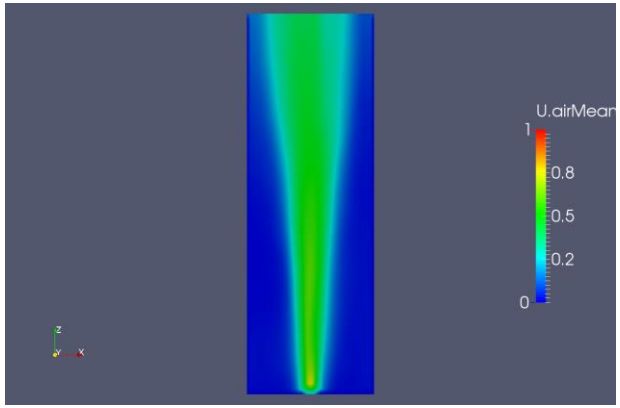
C6
DM1-VM-FI-NTM-TPF

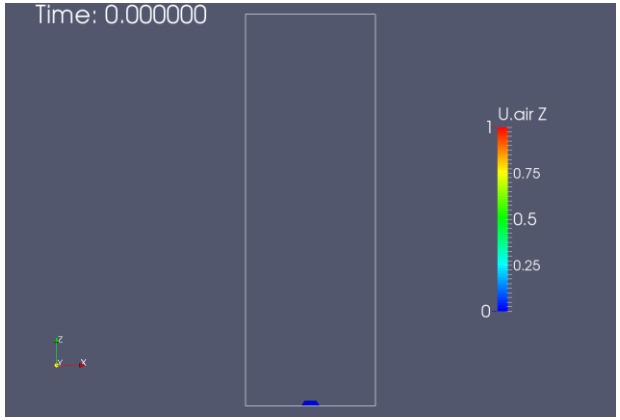
C9
DM1-VM-FI-TM-TPF

Note: the results presented were obtained using OpenFOAM 3.x

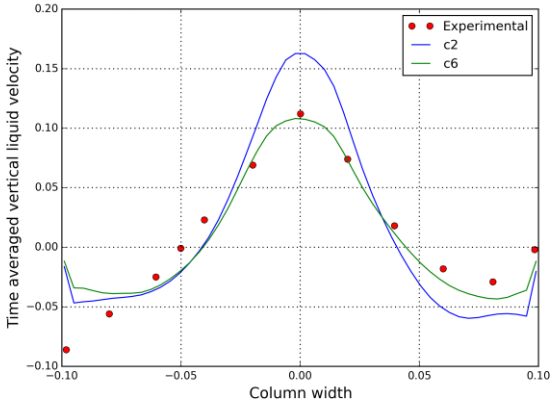
Guided tutorial 6

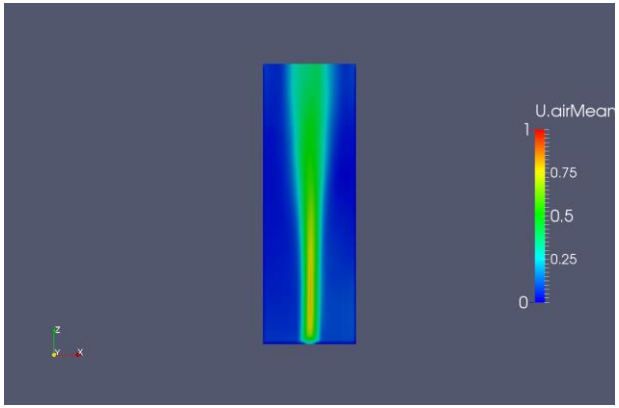
Bubble column – Eulerian-Eulerian approach

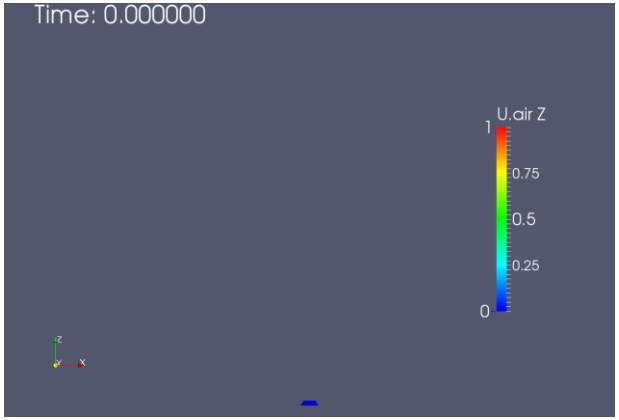




C2
DM1-VM-NFI-NTM-MPF







C6
DM1-VM-FI-NTM-TPF

Note: the results presented were obtained using OpenFOAM 3.x

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach

- This case is ready to run.
- Despite the fact that the mesh is small the cases are time consuming.
- The case is located in the directory:

  `$TM/multiphase/guided_tutorials/GT6/validation_case`

- In the case directory you will find the following sub-directories
  - **`validation_case/laminar`**: laminar bubble column using **multiphaseEulerFoam**
  - **`validation_case/turbulent`**: turbulent bubble column using **multiphaseEulerFoam**

- Hereafter we report the execution times for each case (serial runs):
  - **`validation_case/laminar`** : 77801 seconds
  - **`validation_case/turbulent`** : 84643 seconds

Note: the results presented were obtained using OpenFOAM 3.x

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach

- We are going to use the following solver: **multiphaseEulerFoam**
- Let us explore the dictionaries in the **`constant`** directory.
  - *`g`*: in this dictionary you set the gravity field.
  - *`phaseProperties`*: in this dictionary you set how phases interacts and the physical and interfacial models.
  - *`thermophysicalProperties.air`*: in this dictionary you set the thermo physical properties for the phase air.
  - *`thermophysicalProperties.water`*: in this dictionary you set the thermo physical properties for the phase water.
  - *`momentumTransport.air`*: in this dictionary you set the turbulence model for the phase air.
  - *`momentumTransport.water`*: in this dictionary you set the turbulence model for the phase water.

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach

- We are going to explore as well the dictionaries in the **`system`** directory.
- The discretization method is set in the dictionary *`fvSchemes`*.
- The linear solvers and parameters specific of the pressure-velocity coupling method is set in the dictionary *`fvSolution`*.
- Runtime parameters such as time-step, CFL number, saving frequency, and so on, are set in the dictionary *`controlDict`*.
- In addition, you will find the dictionary *`fvConstraints`* where we define the minimum allowable pressure in the system.
    - In this input file, you can define additional physical limits, e.g., temperature, velocity.
- Remember, as we are solving different equations, the dictionaries will be slightly different from the dictionaries we have used so far.

Guided tutorial 6

Bubble column – Eulerian-Eulerian approach

- At this point, we are ready to run the simulation.
- To run the tutorial automatically, type in the terminal:

```
1. $> run_all.sh
```

- Feel free to open the file `run_all.sh` to see all the commands used.
- If you prefer to run the case manually, type the commands in the terminal window.
- If you are interested, the experimental results (time averaged liquid vertical velocity) are located in the case directory
    - **`$TM/multiphase/guided_tutorials/GT6/validation_case/exp`**

you can plot the experimental results and the simulation results using gnuplot, Python, or your favorite plotting application.

Guided tutorials

- **Guided tutorial 7.** Fluidized bed using kinetic theory of granular flows (KTGF)
- This case is ready to run.
- The case is located in the directory:

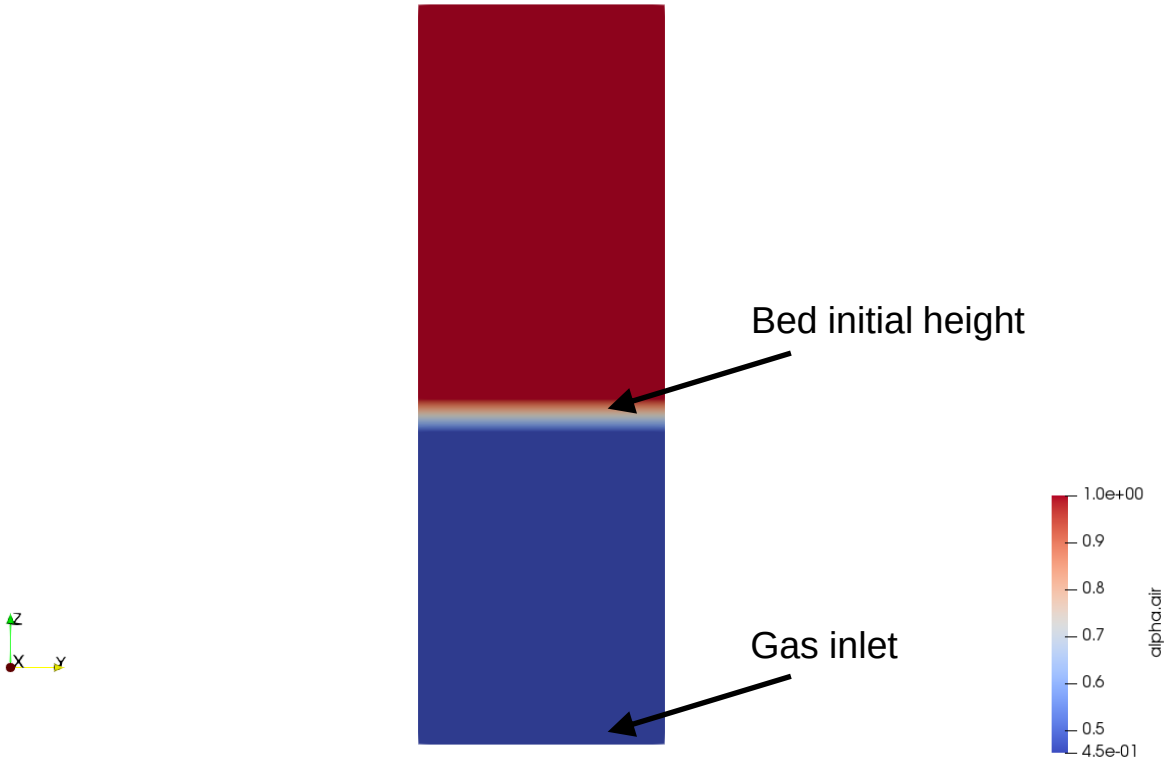
$TM/multiphase/guided_tutorials/GT7/

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

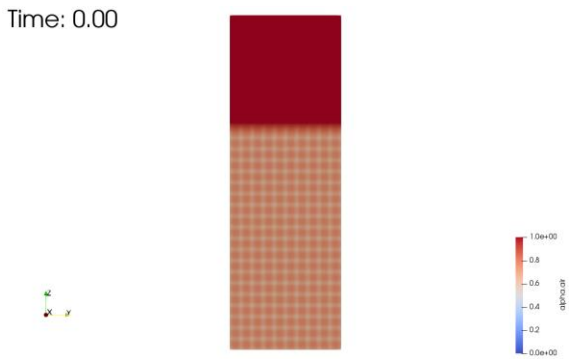
- In this guided tutorial we model a fluidized bed.
- The problem is transient in nature.
- Air is injected at the bottom with a given velocity.
- The particles bed has an initial height.
- We will use an Eulerian-Eulerian approach with kinetic theory of granular flows.
- We are going to use this case to study how to setup a granular flow and the different interfacial models implemented in OpenFOAM.
- This is a validation case, we are going to use an experimental setup similar to the one found in reference [1].
- To validate the results, we compare the results qualitatively



References:
[1] Goldschmidt, M.J.V. and Hoomans, B.P.B. and Kuipers, J.A.M. (2002) *Detailed comparison of Euler-Lagrange and Euler-Euler models for simulation of dense gas fluidised beds.* In: 10th Workshop on Two-phase Flow Predictions, April 9-12, 2002, Merseburg, Germany (pp. pp. 285-299)

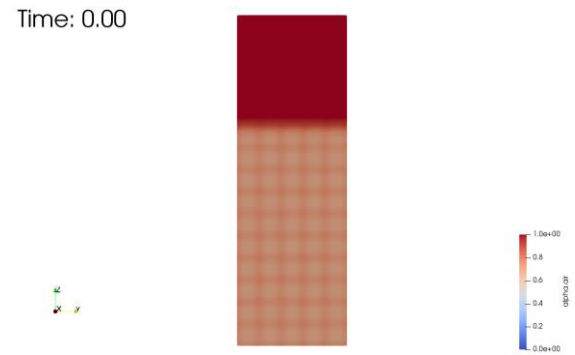
Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows



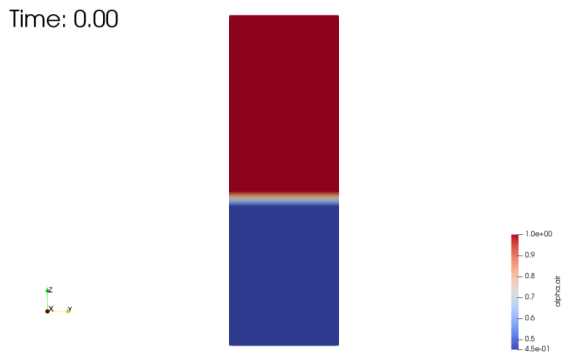
DPM approach
Air volume fraction

http://www.wolfdynamics.com/training/mphase/image39.gif



MPPIC approach
Air volume fraction

http://www.wolfdynamics.com/training/mphase/image40.gif

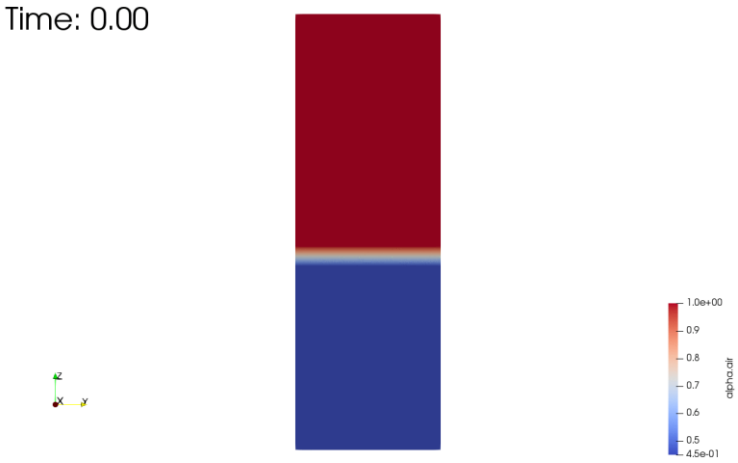


multiphaseEulerFoam
Air volume fraction

http://www.wolfdynamics.com/training/mphase/image41.gif

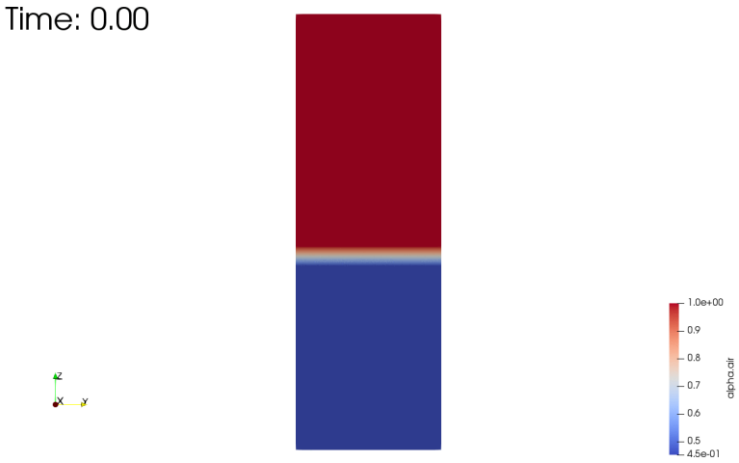
Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows



multiphaseEulerFoam
Air volume fraction
Turbulent case

http://www.wolfdynamics.com/training/mphase/image42.gif

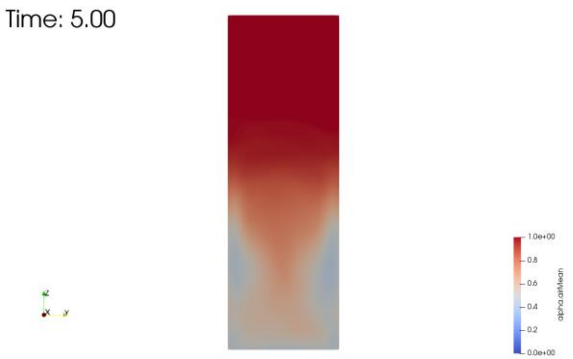


multiphaseEulerFoam
Air volume fraction
Laminar case

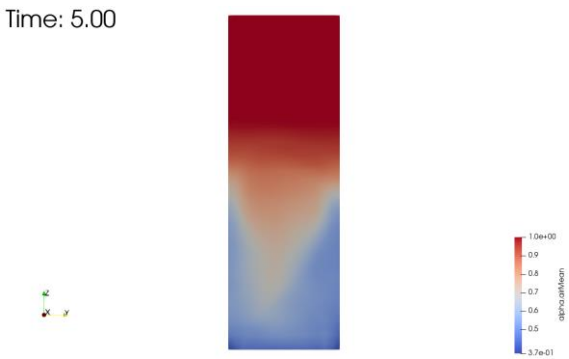
http://www.wolfdynamics.com/training/mphase/image41.gif

Guided tutorial 7

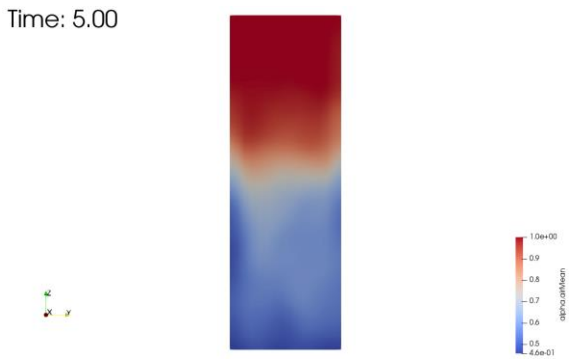
Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows



DPM approach
Air volume fraction mean value



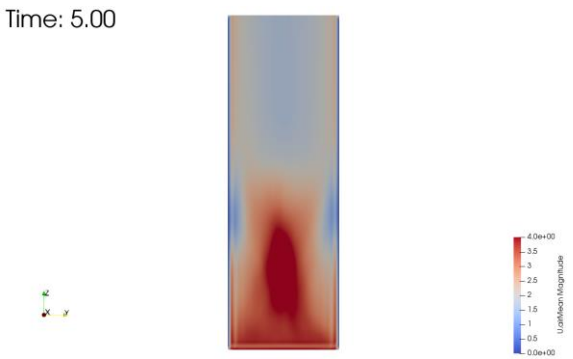
MPPIC approach
Air volume fraction mean value



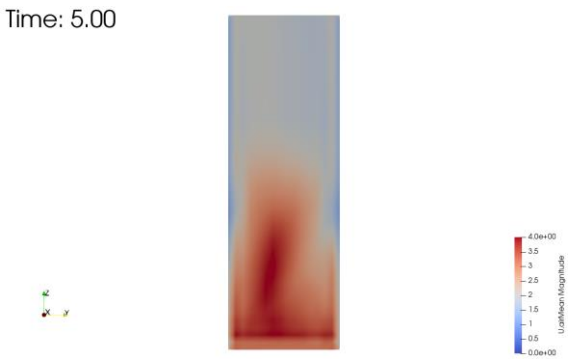
multiphaseEulerFoam
Air volume fraction mean value

Guided tutorial 7

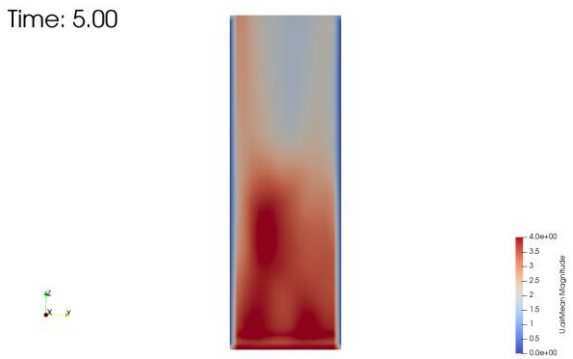
Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows



DPM approach
Air velocity magnitude mean value



MPPIC approach
Air velocity magnitude mean value

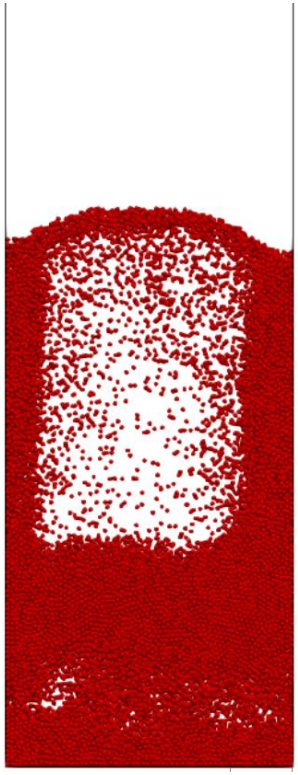


multiphaseEulerFoam
Air velocity magnitude mean value

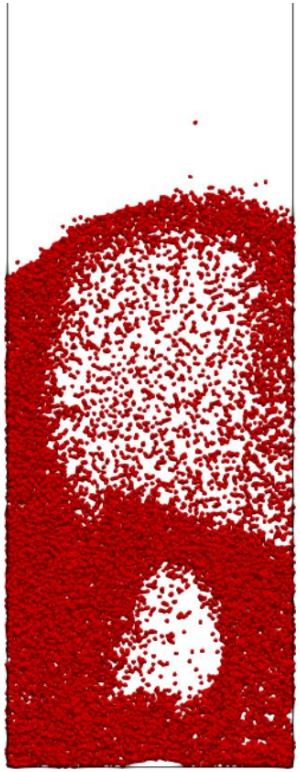
Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

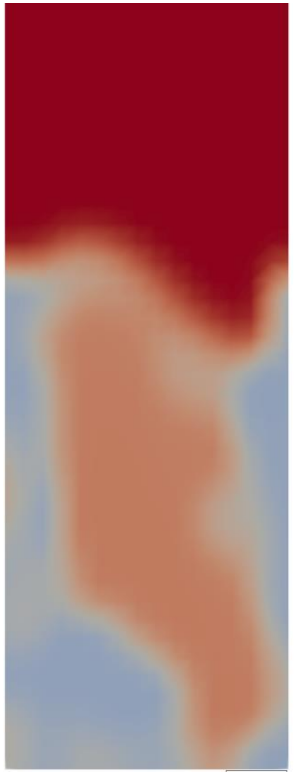
Experiment [1]
Gas velocity: 2.0



DPM approach
Gas velocity: 2.0



MPPIC approach
Gas velocity: 2.0



multiphaseEulerFoam
Gas velocity: 2.0

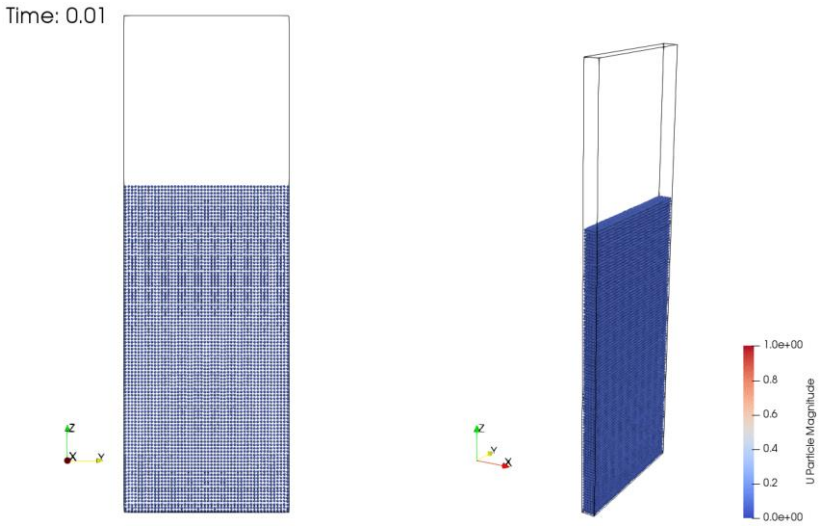
References:
[1] Goldschmidt, M.J.V. and Hoomans, B.P.B. and Kuipers, J.A.M. (2002) *Detailed comparison of Euler-Lagrange and Euler-Euler models for simulation of dense gas fluidised beds.* In: 10th Workshop on Two-phase Flow Predictions, April 9-12, 2002, Merseburg, Germany (pp. pp. 285-299)

Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

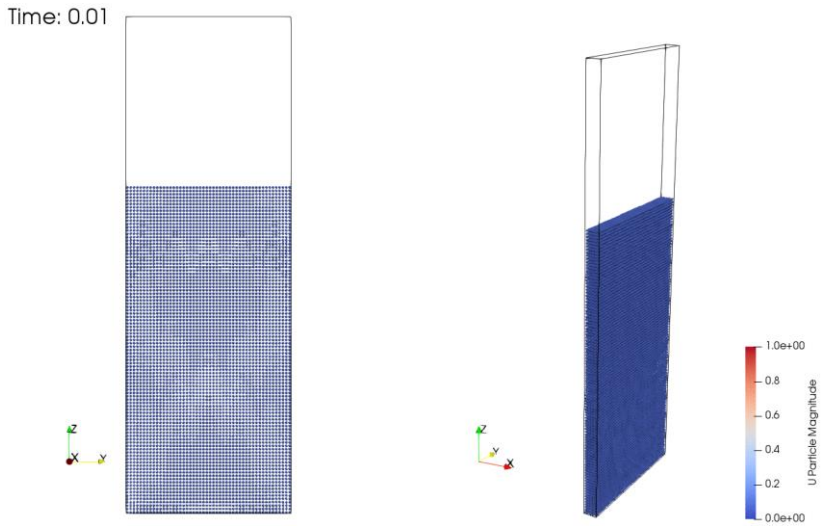
- For completeness, we present the results of particle-particle interactions (lagrangian solvers).
- The eulerian-eulerian solver does not provide this information.

http://www.wolfdynamics.com/training/mphase/image43.gif



DPM approach
Particle-particle interactions colored by velocity magnitude (particles not to scale)

http://www.wolfdynamics.com/training/mphase/image44.gif



MPPIC approach
Particle-particle interactions colored by velocity magnitude (particles not to scale)

Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

- In the case directory **`$TM/multiphase/guided_tutorials/GT7/KTGF`**, you will find the following sub-directories:
    - **`Goldschmidt/C1`**: laminar fluidized bed using **multiphaseEulerFoam** with phasePressure model for the solid phase (particles).
    - **`Goldschmidt/C2`**: turbulent fluidized bed using **multiphaseEulerFoam** with kineticTheory model for the solid phase (particles).
- At this point, choose with case do you want to run.
- The computing time and run instructions are similar.

Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

- In the case directory **`$TM/multiphase/guided_tutorials/GT7`**, you will find the following additional sub-directories:
  - **`DPM`**: fluidized bed using the **DPM** approach.
  - **`MPPIC`**: fluidized bed using the **MPPIC** approach.
- For completeness, hereafter we report the execution times for each solver (serial runs):
  - DPM: 166500 seconds (about two days)
  - MPPIC: 6810 seconds
  - KTGP/Goldschmidt/C2: 250 seconds

Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

- We are going to use the following solver: **multiphaseEulerFoam**
- Let us explore the dictionaries in the **`constant`** directory.
  - *`g`*: in this dictionary you set the gravity field.
  - *`phaseProperties`*: in this dictionary you set how phases interacts and the physical and interfacial models.
  - *`thermophysicalProperties.air`*: in this dictionary you set the thermo physical properties for the phase air.
  - *`thermophysicalProperties.particles`*: in this dictionary you set the thermo physical properties for the particles.
  - *`momentumTransport.air`*: in this dictionary you set the turbulence model for the phase air.
  - *`momentumTransport.particles`*: in this dictionary you set the turbulence model for the particles.

Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

- We are going to explore as well the dictionaries in the **`system`** directory.
- The discretization method is set in the dictionary *`fvSchemes`*.
- The linear solvers and parameters specific of the pressure-velocity coupling method is set in the dictionary *`fvSolution`*.
- Runtime parameters such as time-step, CFL number, saving frequency, and so on, are set in the dictionary *`controlDict`*.
- In addition, you will find the dictionary *`fvConstraints`* where we define the minimum allowable pressure in the system.
    - In this input file, you can define additional physical limits, e.g., temperature, velocity.
- Remember, as we are solving different equations, the dictionaries will be slightly different from the dictionaries we have used so far.

Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

- When setting the continuous phase name using the solvers **denseParticleFoam** or **multiphaseEulerFoam**, remember to use the same phase name when assigning the boundary conditions.
- There are a few boundary conditions that require the name of the continuous phase field, *e.g.*,

```
type            inletOutlet;
phi             phi.air;        <-- Default value is phi
inletValue      uniform 1;
value           uniform 1;
```

```
type            interstitialInletVelocity;
alpha           alpha.air;      <--
inletVelocity   uniform (0 0 1);
value           uniform (0 0 1);
```

```
type            pressureInletOutletVelocity;
phi             phi.air;        <-- Default value is phi
value           uniform (0 0 0);
```

- Check the online documentation or the source code to see what are the default values of the boundary conditions and update them accordingly

Guided tutorial 7

Fluidized bed – Eulerian-Eulerian approach using kinetic theory of granular flows

- At this point, we are ready to run the simulation.
- To run the tutorial automatically, type in the terminal:

```
1. $> run_all.sh
```

- Feel free to open the file `run_all.sh` to see all the commands used.
- If you prefer to run the case manually, type the commands in the terminal window.
- Remember to choose among the DPM, MPPIC and KTGF cases.

Guided tutorials

- **Guided tutorial 8.** Particle injection in a mixing tank – Lagrangian approaches.
- This case is ready to run.
- The case is located in the directory:

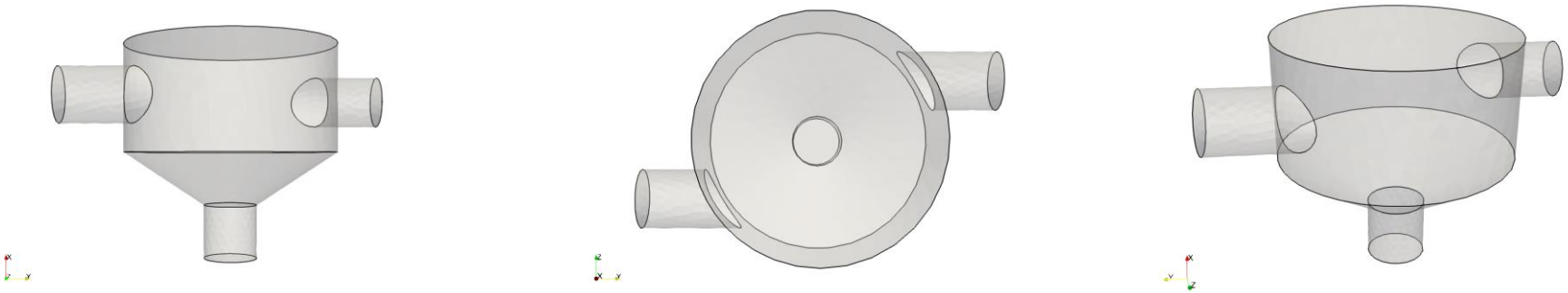
$TM/multiphase/guided_tutorials/GT8/

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

Guided tutorial 8 – Particle injection in a mixing tank

- In this tutorial we will address the tracking of water droplets in a mixing tank.
- This tutorial is divided in two cases:
  - **Case1.** Particles tracking without hydrodynamic coupling using the solver **particleFoam**.
  - **Case2.** Particles tracking tutorial with hydrodynamic coupling using the solver **denseParticleFoam** (MPPIC approach)

- We will use the following mixing tank geometry.



Guided tutorials

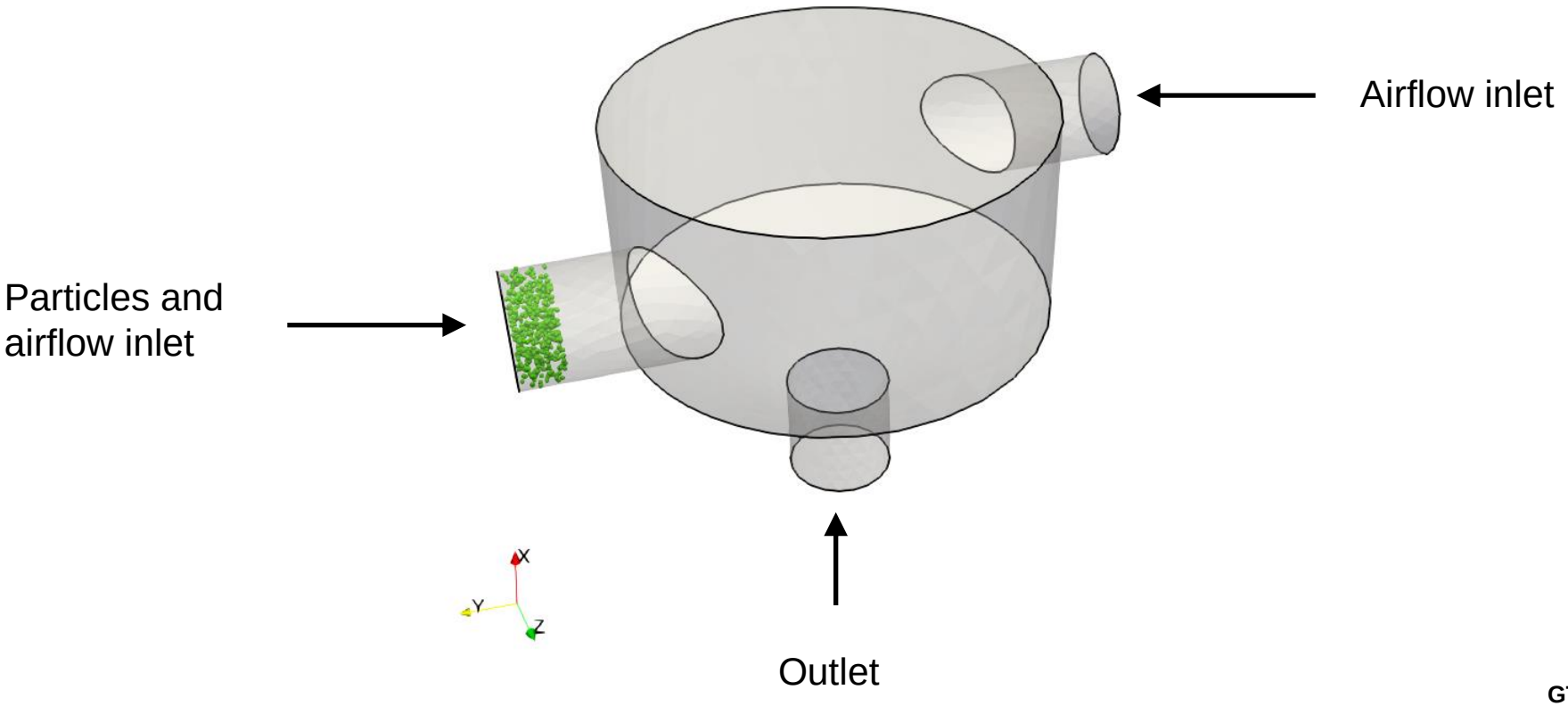
- **Case1.** Particles tracking without hydrodynamic coupling
- This case is ready to run.
- The case is located in the directory:

$TM/multiphase/guided_tutorials/GT8/uncoupled

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

Guided tutorial 8 – Particle injection in a mixing tank

- In this tutorial we will track water droplets in a static mixer by using an uncoupled particle tracking strategy.
- We will consider the gravity and the drag forces.



Guided tutorial 8 – Particle injection in a mixing tank

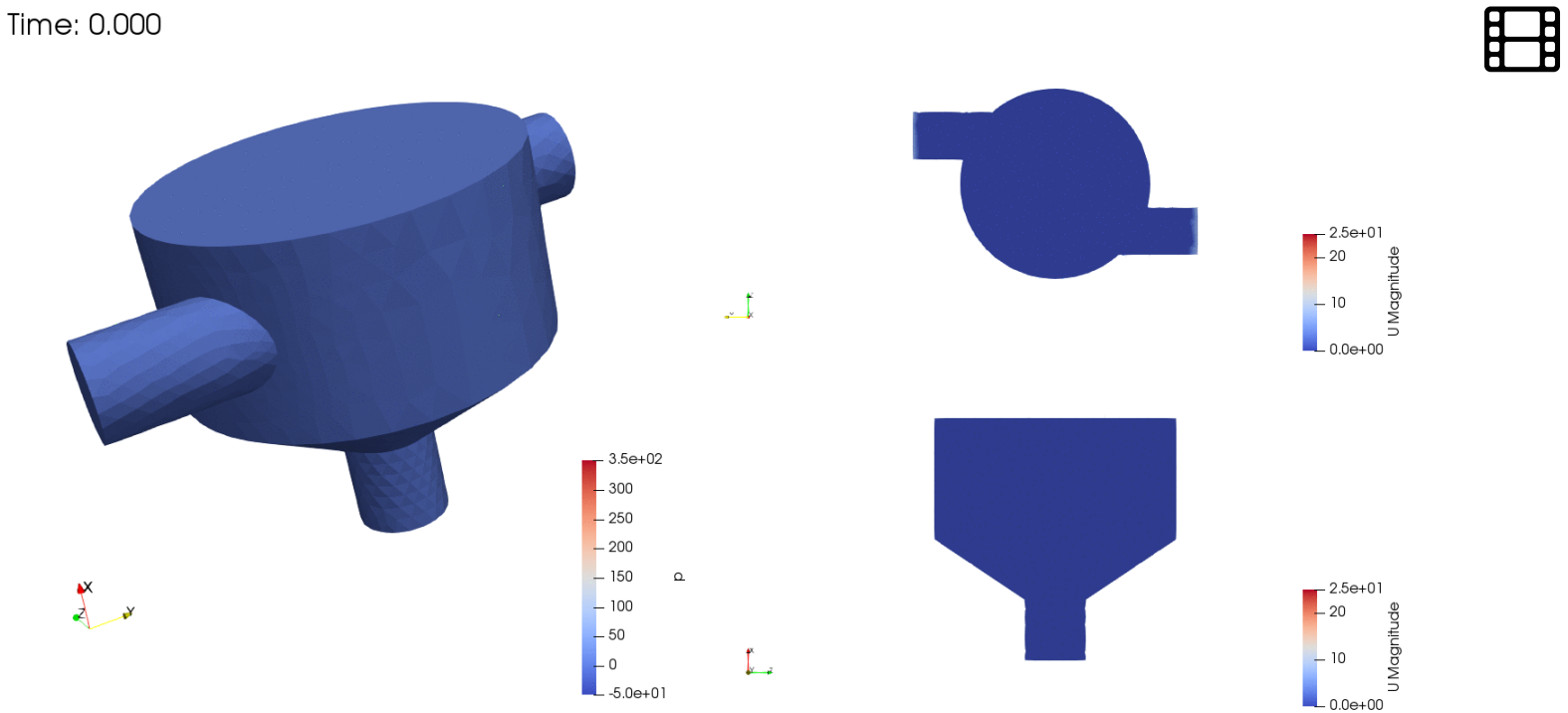
- The uncoupled particle tracking approach is based on the availability of a flow solution computed using any solver (incompressible or compressible).
- The solution is then used to compute the evolution of the particles using a Lagrangian model.
- We stress that this is an uncoupled approach, we are only computing the particle evolution and interaction in a precomputed flow field.
- This tutorial case is divided in two parts.
  - **Part 1:**
    - This tutorial is located in the directory **`GT8/uncoupled/c1_airflow`**
    - In this directory you will find the case setup of the single-phase case (precursor simulation).
    - We will run this case to obtain the flow solution that will be used to run the uncoupled Lagrangian solver.

Guided tutorial 8 – Particle injection in a mixing tank

- The uncoupled particle tracking approach is based on the availability of a flow solution computed using any solver (incompressible or compressible).
- The solution is then used to compute the evolution of the particles using a Lagrangian model.
- We stress that this is an uncoupled approach, we are only computing the particle evolution and interaction in a precomputed flow field.
- This tutorial case is divided in two parts.
  - **Part 2:**
    - This tutorial is located in the directory **`GT8/uncoupled/c2_particles`**
    - In this directory you will find the uncoupled particle tracking case setup.
    - This case uses the solution obtained in the previous step to compute the evolution of the particles..
    - The solution can come from the same mesh or a finer or coarser mesh.
    - In this case we will use the Lagrangian solver **particleFoam.**

Guided tutorial 8 – Particle injection in a mixing tank

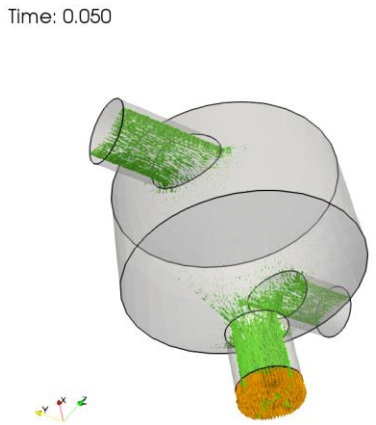
- In the directory **`c1_airflow`** you will find the precursor simulation case setup.
- You will also find a precomputed turbulent solution, so you do not need to run the case from scratch.

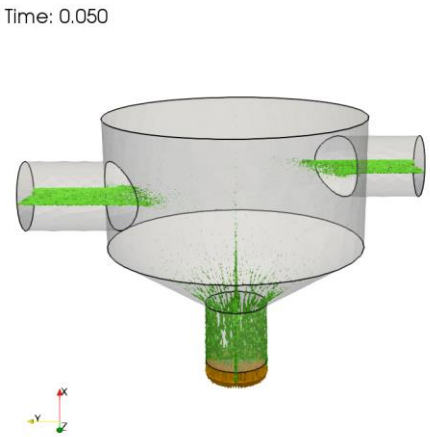


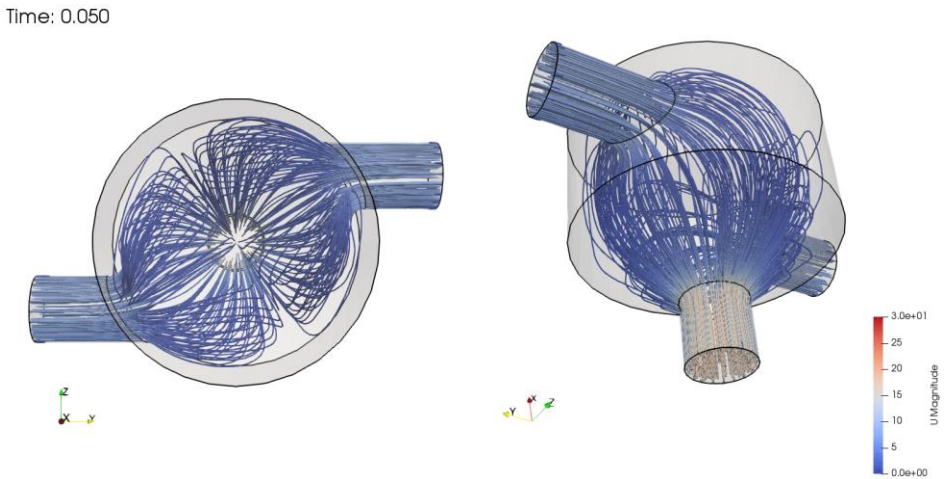
http://www.wolfdynamics.com/training/mphase/image50.gif

Guided tutorial 8 – Particle injection in a mixing tank

- In the directory **`c1_airflow`** you will find the precursor simulation case setup.
- You will also find a precomputed turbulent solution, so you do not need to run the case from scratch.







http://www.wolfdynamics.com/training/mphase/image51.gif

http://www.wolfdynamics.com/training/mphase/image52.gif

Guided tutorial 8 – Particle injection in a mixing tank

Particle injection in a mixing tank

- At this point, we are ready to run the simulation.
- The simulation consist of running the precursor simulation and then obtaining the uncoupled dynamics of the particles using the computed flow field.
- You can start this case from scratch or from the pre-computed solution located in the directory **`c1_airflow`**.
- To run the case, go to the directory **`c2_particles`**, and type in the terminal:

```
1. $> run_all.sh
```

- This script will map the precomputed solution located in the directory **`c1_airflow`** and then will launch the solver `particleFoam`.
- Feel free to open the file `run_all.sh` to see all the commands used.

Guided tutorial 8 – Particle injection in a mixing tank

- The dictionary *cloudProperties* requires special attention, as it is in this dictionary where we set the interaction models and injection models of the particles.
- Remember, this dictionary is located in the directory **constant**.

```
type        cloud;

Solution
{
    active          true;
    coupled         false;
    transient       yes;

    cellValueSourceCorrection off;

    interpolationSchemes
    {
        rho         cell;
        U           cellPoint;
        mu          cell;
    }

    integrationSchemes
    {
        U           Euler;
    }
```

- A basic cloud of solid particles.
- Includes forces,patch interaction, injection, dispersion and stochastic collisions.
- Same as the cloud previously used by rhoParticleFoam (uncoupledKinematicParticleFoam)

Guided tutorial 8 – Particle injection in a mixing tank

- The dictionary *cloudProperties* requires special attention, as it is in this dictionary where we set the interaction models and injection models of the particles.
- Remember, this dictionary is located in the directory **constant**.

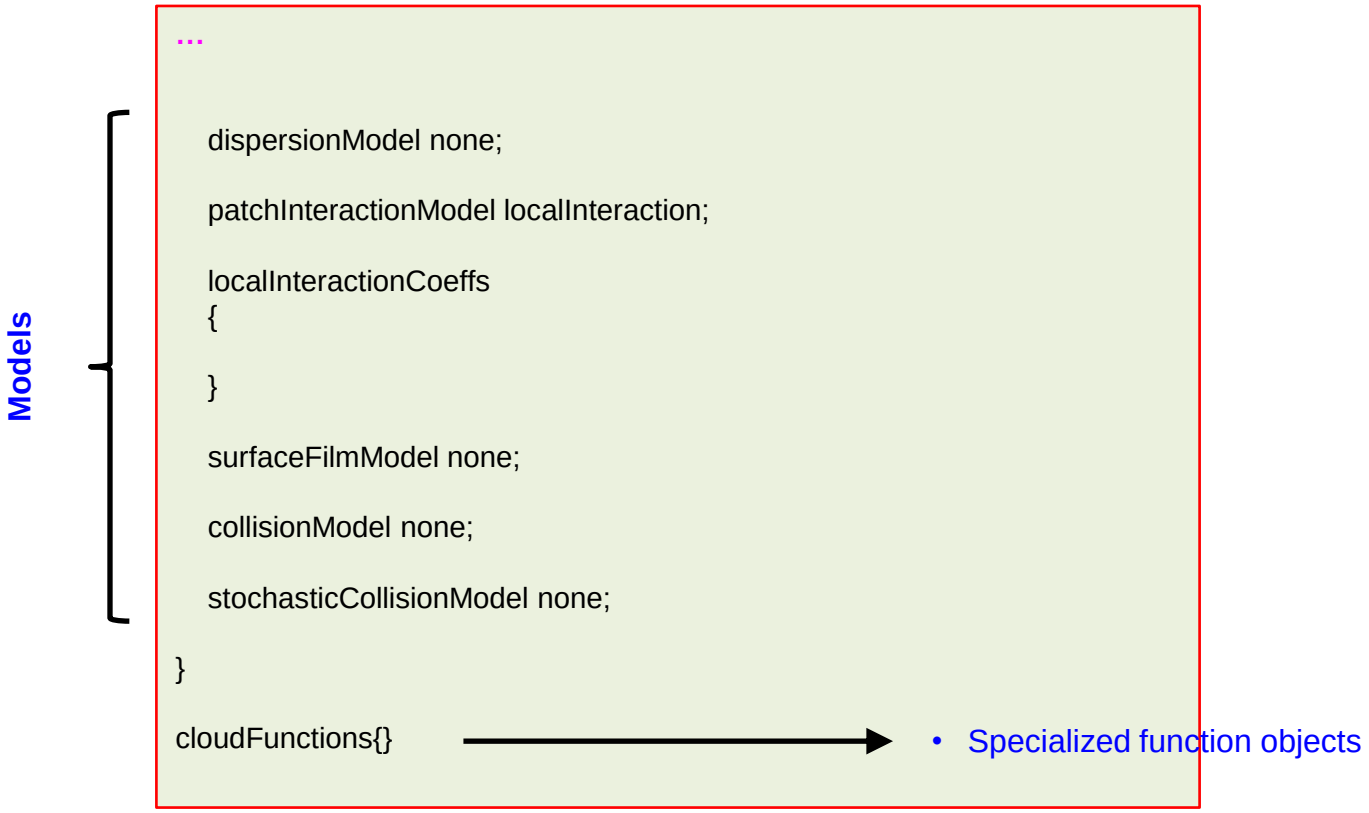
- Physical properties of the particles.

- In the **subModels** section, you define the models to be used with the lagrangian particles.
- Such as the forces applied to the particles, the injection models, the particle-particle treatment, the wall interaction, collision, damping and dispersion models, etc.

```
...

constantProperties
{
    rho0            100;
    alphaMax        0.9;
}

subModels
{

    particleForces
    {
        sphereDrag;
        gravity;
    }

    injectionModels
    {

...
```

Guided tutorial 8 – Particle injection in a mixing tank

- The dictionary *cloudProperties* requires special attention, as it is in this dictionary where we set the interaction models and injection models of the particles.
- Remember, this dictionary is located in the directory **constant**.



Guided tutorial 8 – Particle injection in a mixing tank

- A detailed overview of the dictionary *constant/cloudProperties* is provided in the MPPIC tutorial.
- The particleFoam solver is suitable when a low volume fraction of the discrete phase is expected and the influence of the particles momentum on the continuous phase is not of interest.
- The dictionary *constant/g* contains the specification of the gravity force.

```
dimensions      [0 1 -2 0 0 0 0];

value           ( -9.81 0 0 );
```

Guided tutorial 8 – Particle injection in a mixing tank

- The results of the solver `particleFoam` are saved in each time directory inside the sub-directory **`<time_directory>/lagrangian/cloud`**
- The particle quantities can be stored in ASCII (human readable) or binary format.
- If you save the particle fields in ASCII format, they can be easily parsed using user-defined scripts (Python, Matlab, bash, etc).
- However, it is easier to postprocess the particle fields using paraFoam (in serial mode).
- Each particle field is stored in separates files.
- For example, the age (or resident time) of the particles is saved in the file *`age`*.
- The content of the file *`age`* is as follows,

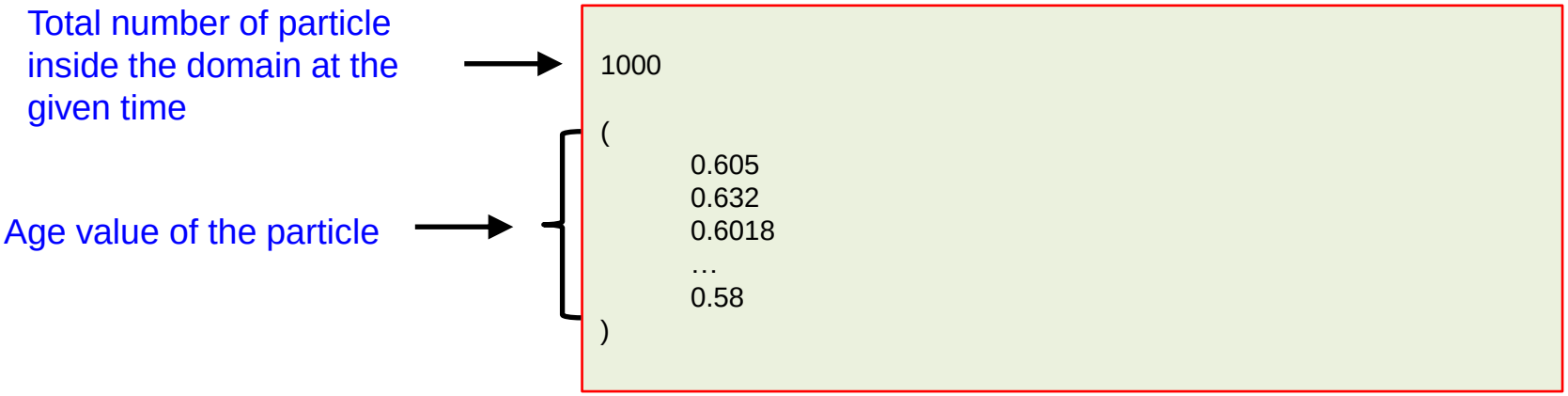
Total number of particle inside the domain at the given time

Age value of the particle

```
1000

(
        0.605
        0.632
        0.6018
        …
        0.58
)
```

Guided tutorial 8 – Particle injection in a mixing tank

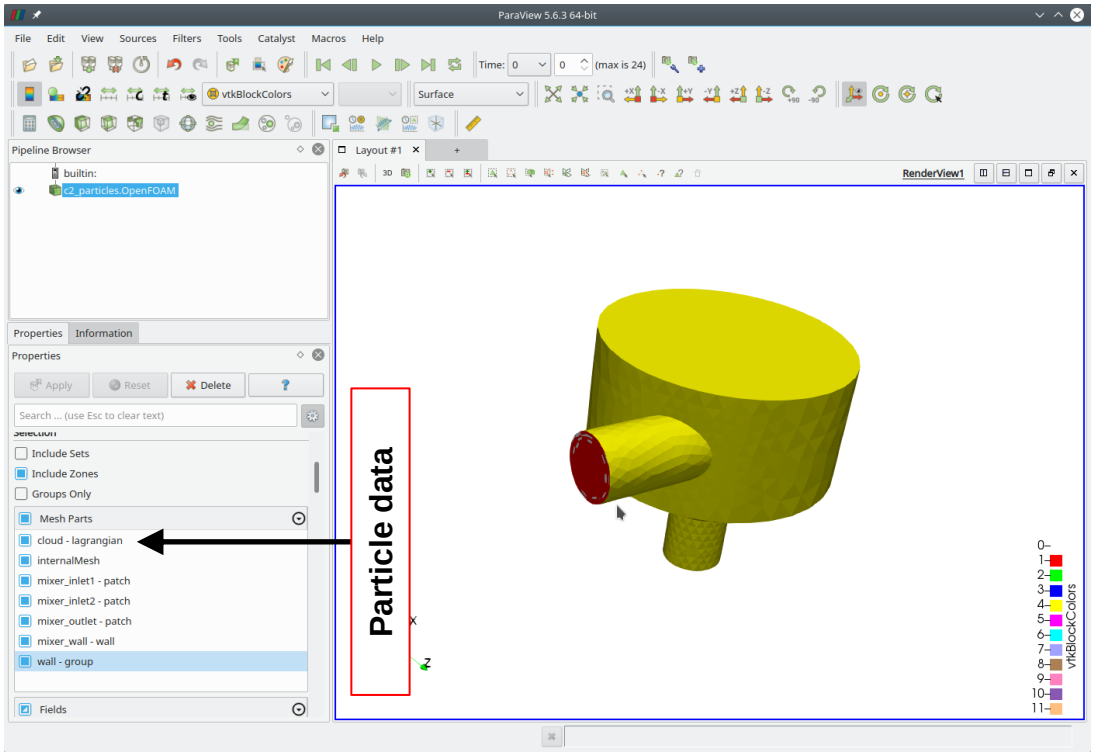
- Each particle is defined by its ID number that can be accessed in the `origId` file.
- At each time, the `origId` list is updated and the particle order may change.
- The user has to pay attention to the current `origID` when parsing the Lagrangian data set of a give time step.
- Common variables stored in the **`lagrangian/cloud`** sub-directory are
  - Particle age, density, velocity, diameter, angular momentum, temperature, position, and so on.



Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

- In paraFoam, you can access and visualize the native Lagrangian fields.
- The Lagrangian fields can be accessed in the **Mesh parts** window, usually under the name **cloud – lagrangian**
- If you use `paraview` or launch `paraFoam` with the option `-builtin`, you will not get access to the Lagrangian fields.
- With `paraFoam`, it is not possible to post-process parallel fields. Therefore, you cannot access the Lagrangian data.
- A way around this, is to convert the solution data to VTK format or any other format compatible with paraview/paraFoam.
- For example, you can convert the solution to VTK format using the following utility (you can run in parallel as well).
    - `$> foamToVTK`
- At this point, you will find the Lagrangian fields in the directory, **`VTK/lagrangian`**

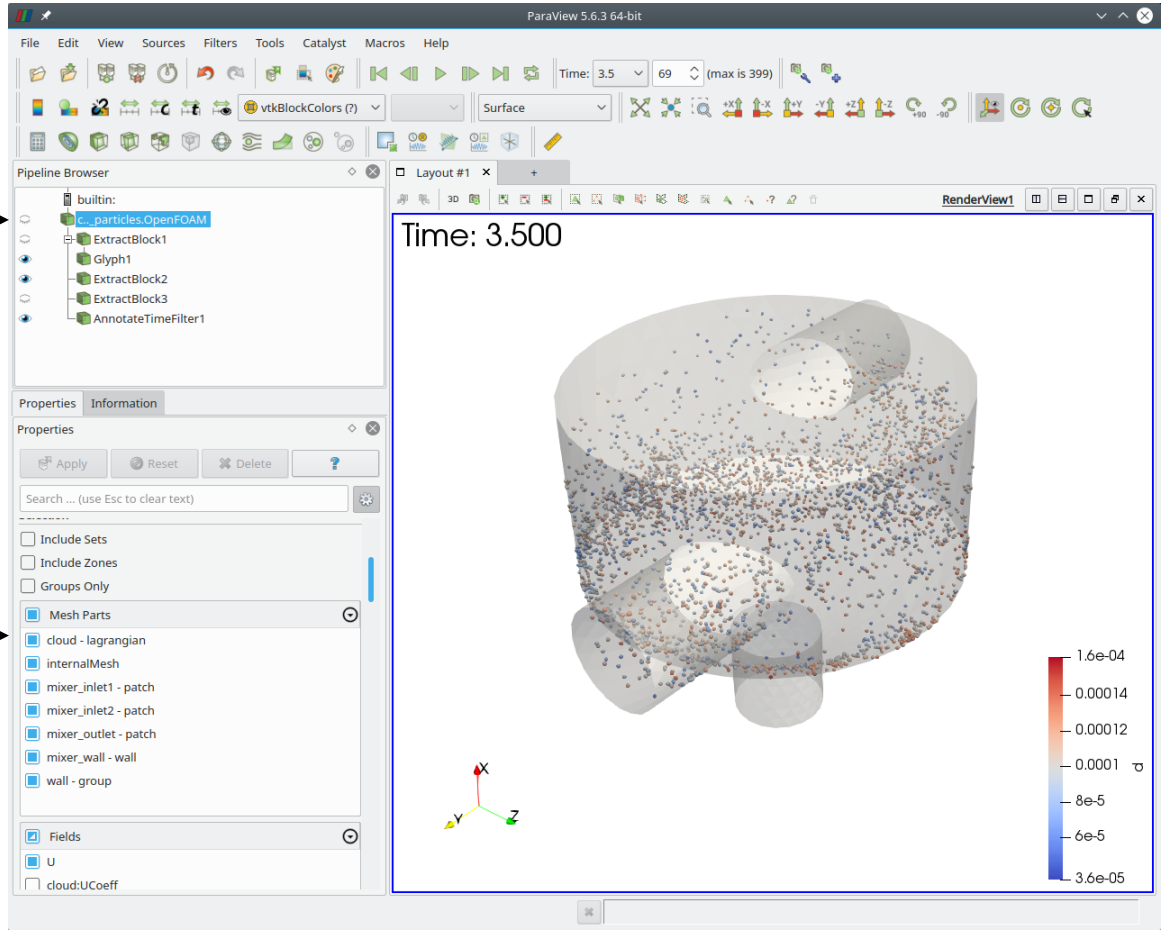


Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

- Select the original data

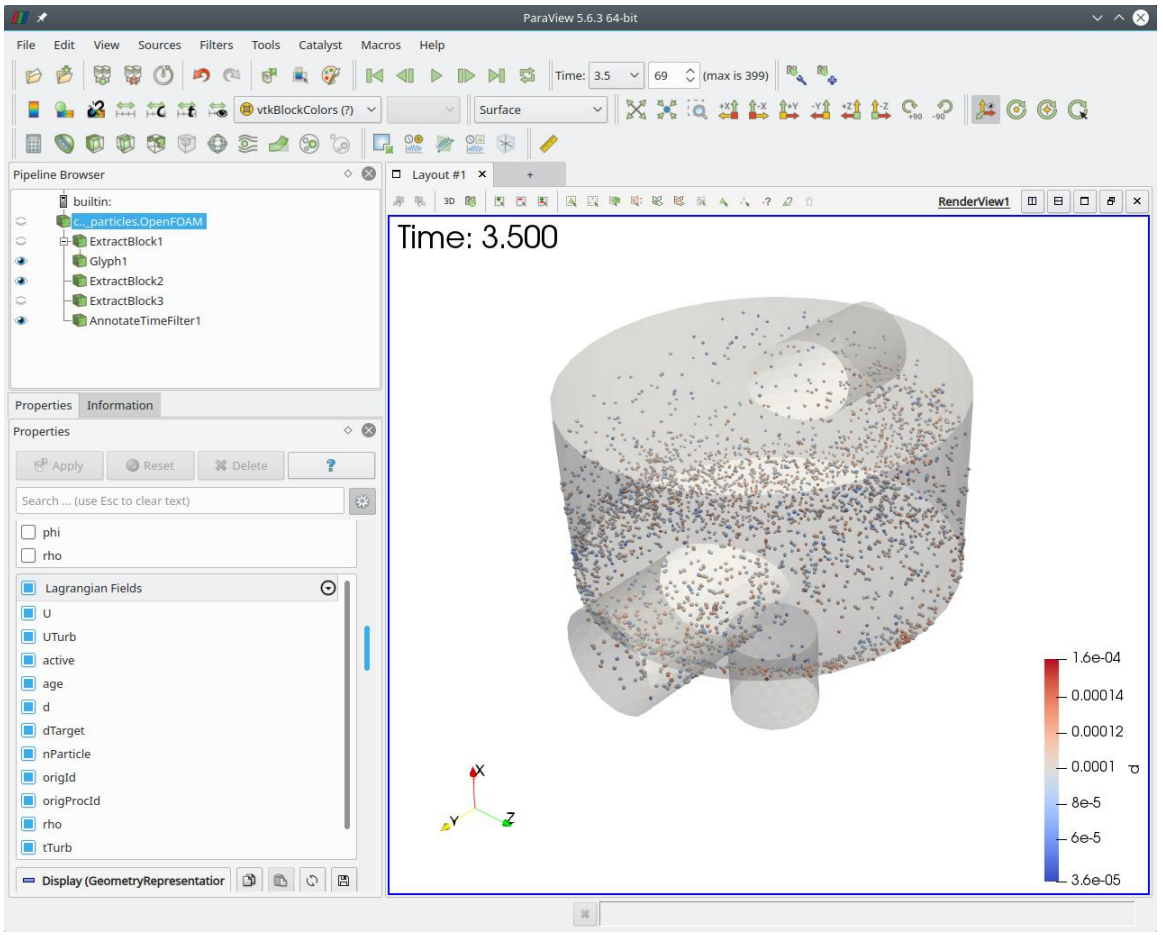
- In the mesh part window select the object **cloud – lagrangian**
- In this case, we selected all the objects.



Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

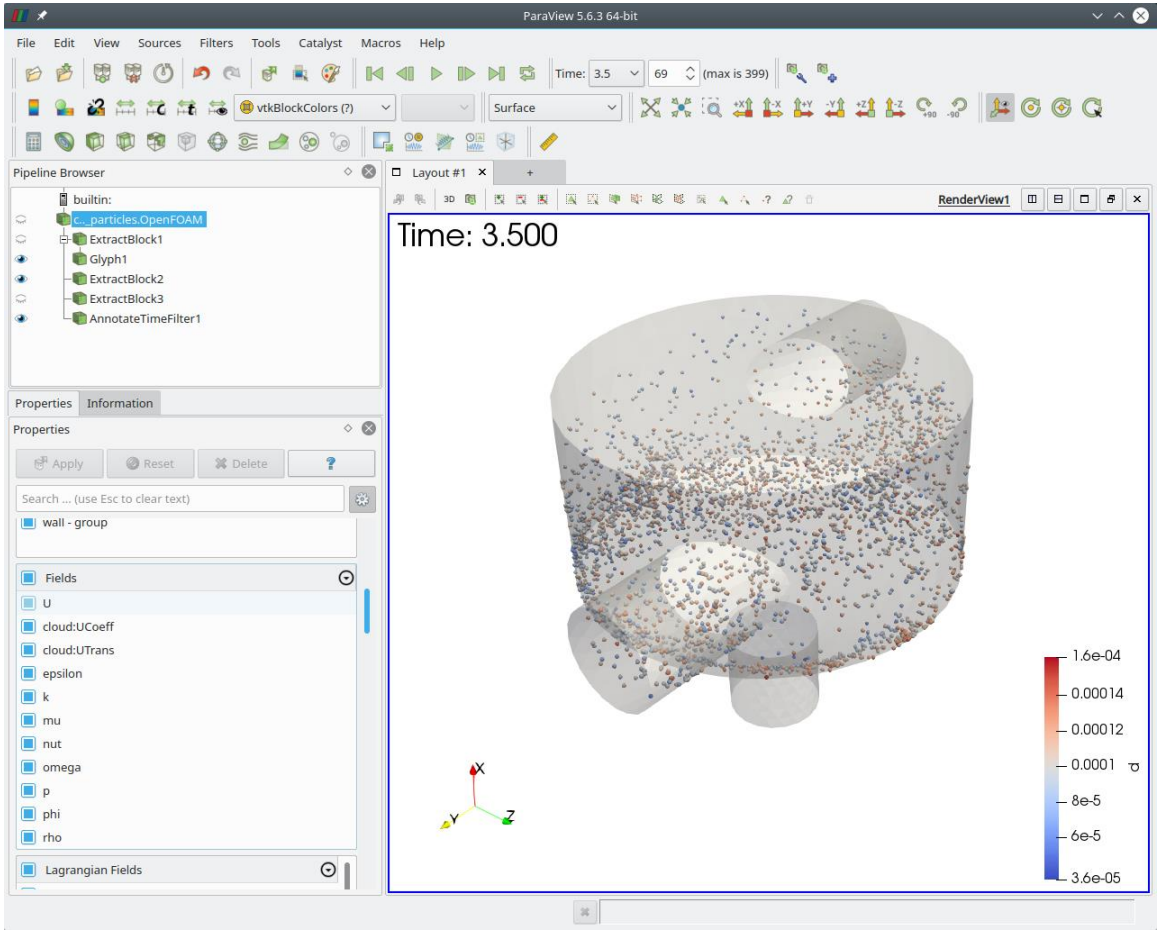
- Select the Lagrangian fields you would like to use.
- In this case, we selected all the objects.



Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

- Select the Eulerian fields you would like to use.
- The Eulerian fields are the fields related to the internal mesh (**internalMesh** object).
- In this case, we selected all the objects.



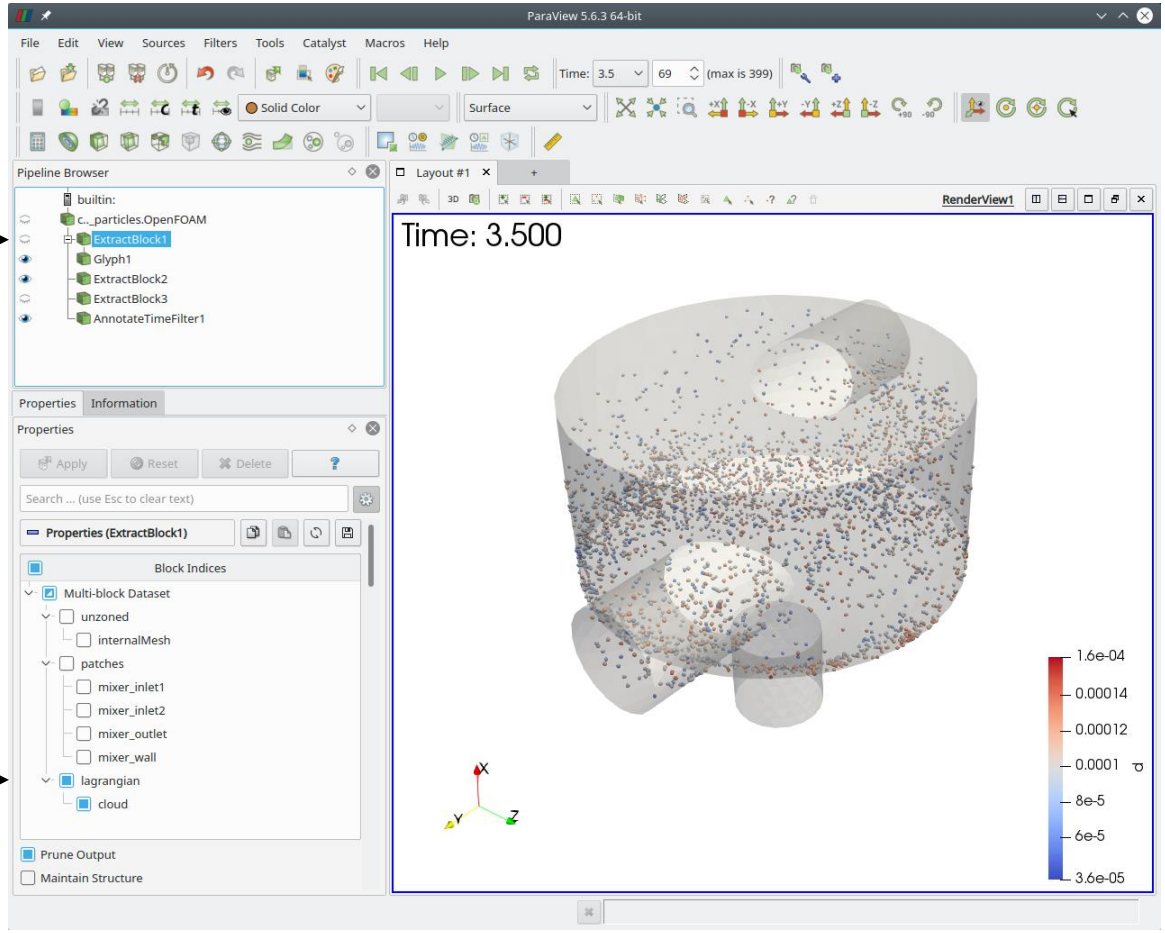
Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

- It is recommended to use the filter **Extract Block** to visualize different objects.
- Namely, the Eulerian mesh and the Lagrangian particles.

- Use the filter **Extract Block** to access different objects in the **Mesh Parts** tab.
- Notice that we are applying the filter **Extract Block** to the original data.
  - That is, the first entry in the pipeline.

- Access only the **lagrangian** object.



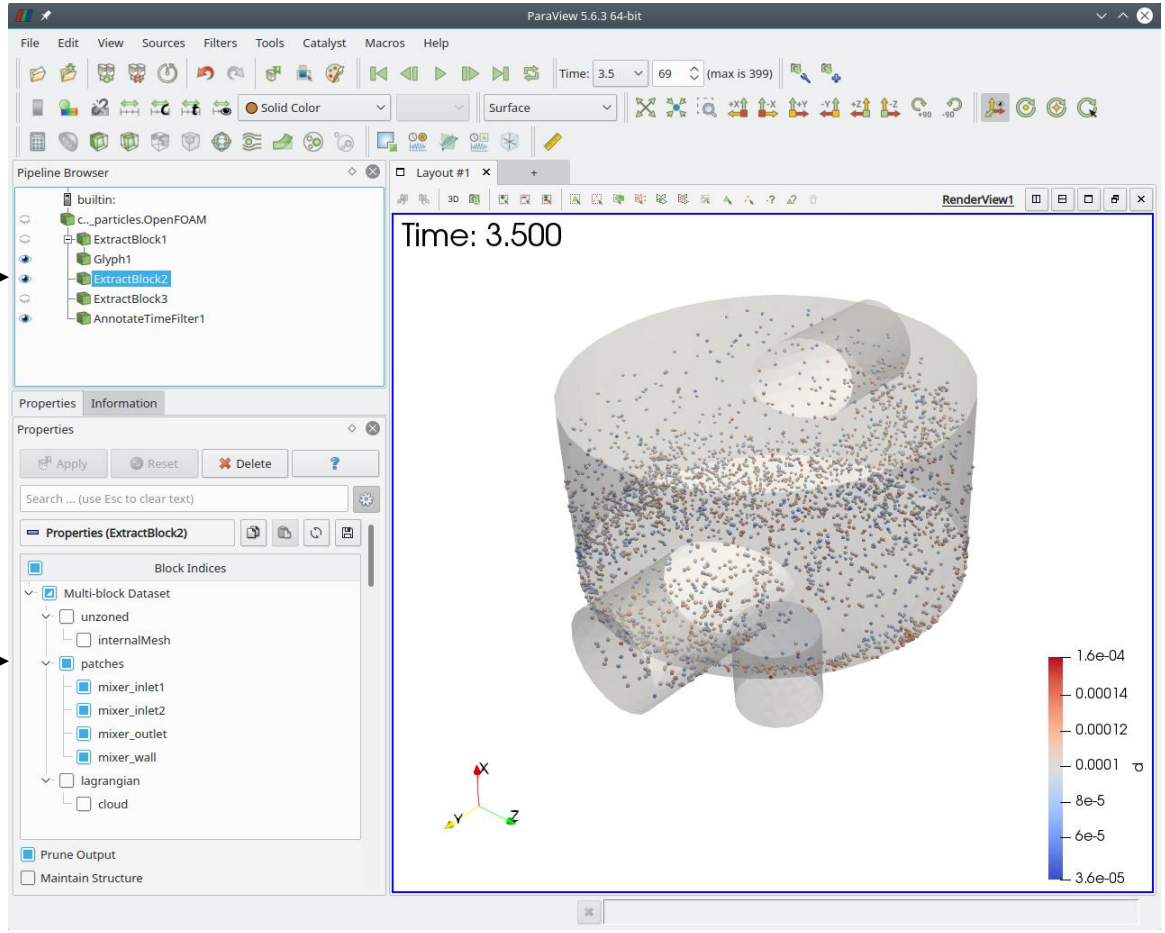
Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

- It is recommended to use the filter **Extract Block** to visualize different objects.
- Namely, the Eulerian mesh and the Lagrangian particles.

- Use the filter **Extract Block** to access different objects in the **Mesh Parts** tab.
- Notice that we are applying the filter **Extract Block** to the original data.
  - That is, the first entry in the pipeline.

- Access only the surface patches data.



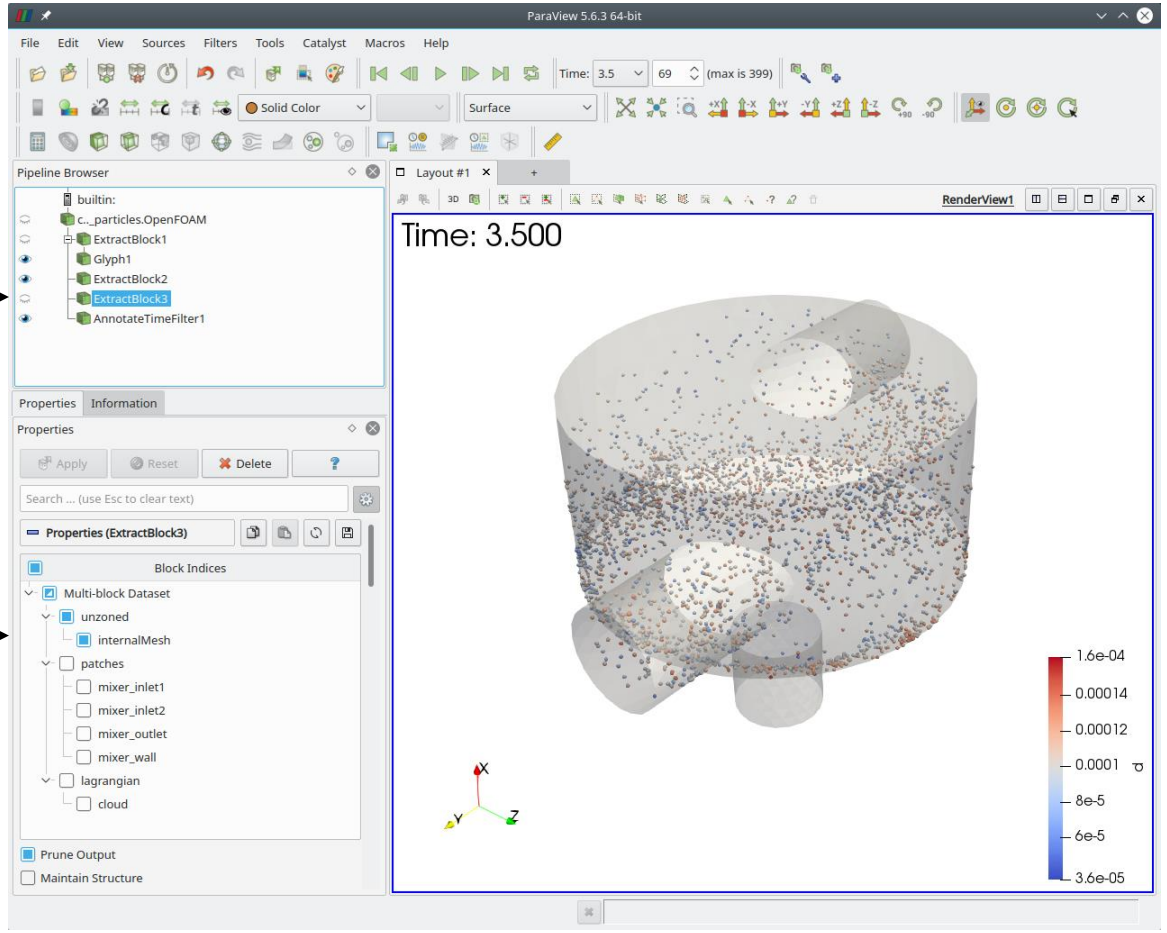
Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

- It is recommended to use the filter **Extract Block** to visualize different objects.
- Namely, the Eulerian mesh and the Lagrangian particles.

- Use the filter **Extract Block** to access different objects in the **Mesh Parts** tab.
- Notice that we are applying the filter **Extract Block** to the original data.
  - That is, the first entry in the pipeline.

- Access only the internal field data.
- Remember, you also need to select the fields that you would like to access.

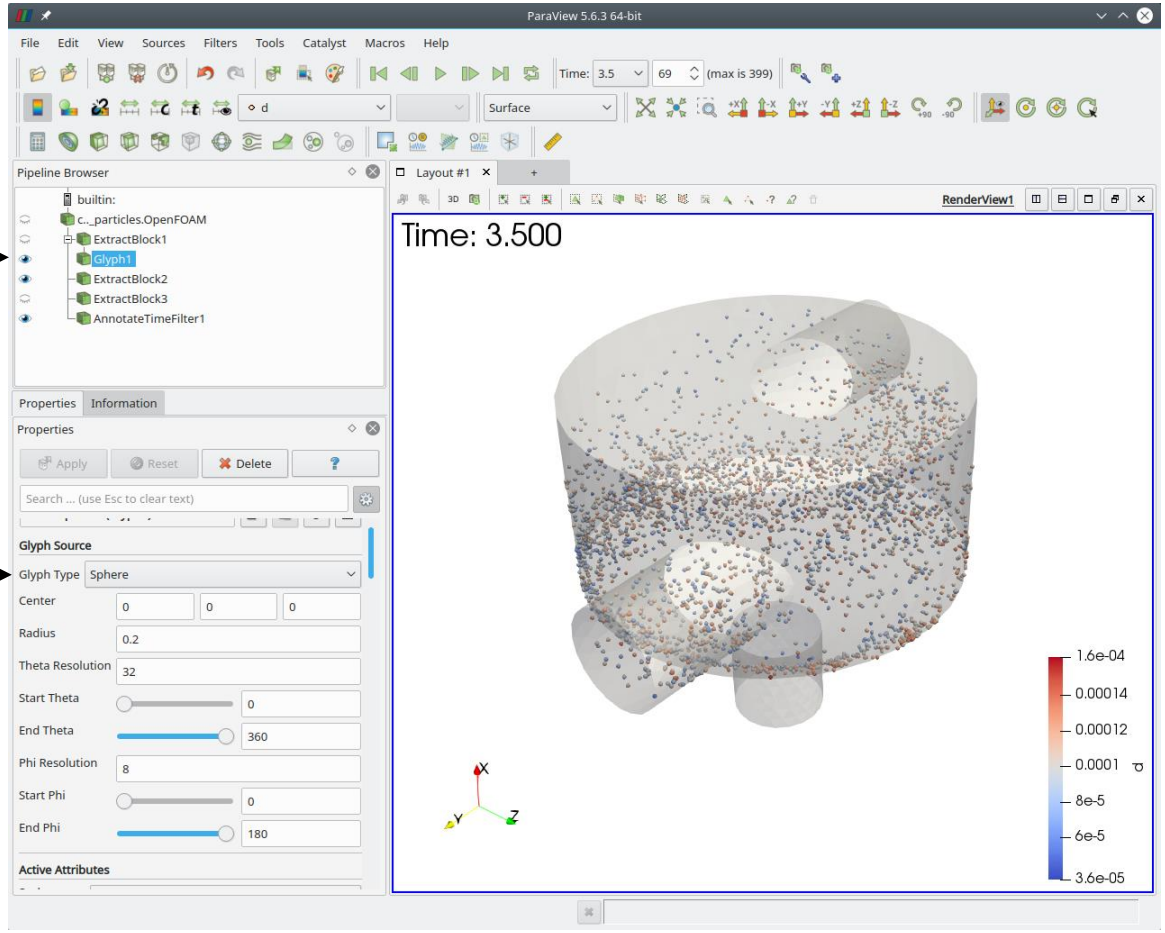


Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

- Apply the **Glyph** filter to the lagrangian field data.

- Use **Spheres** to visualize the **Glyphs**



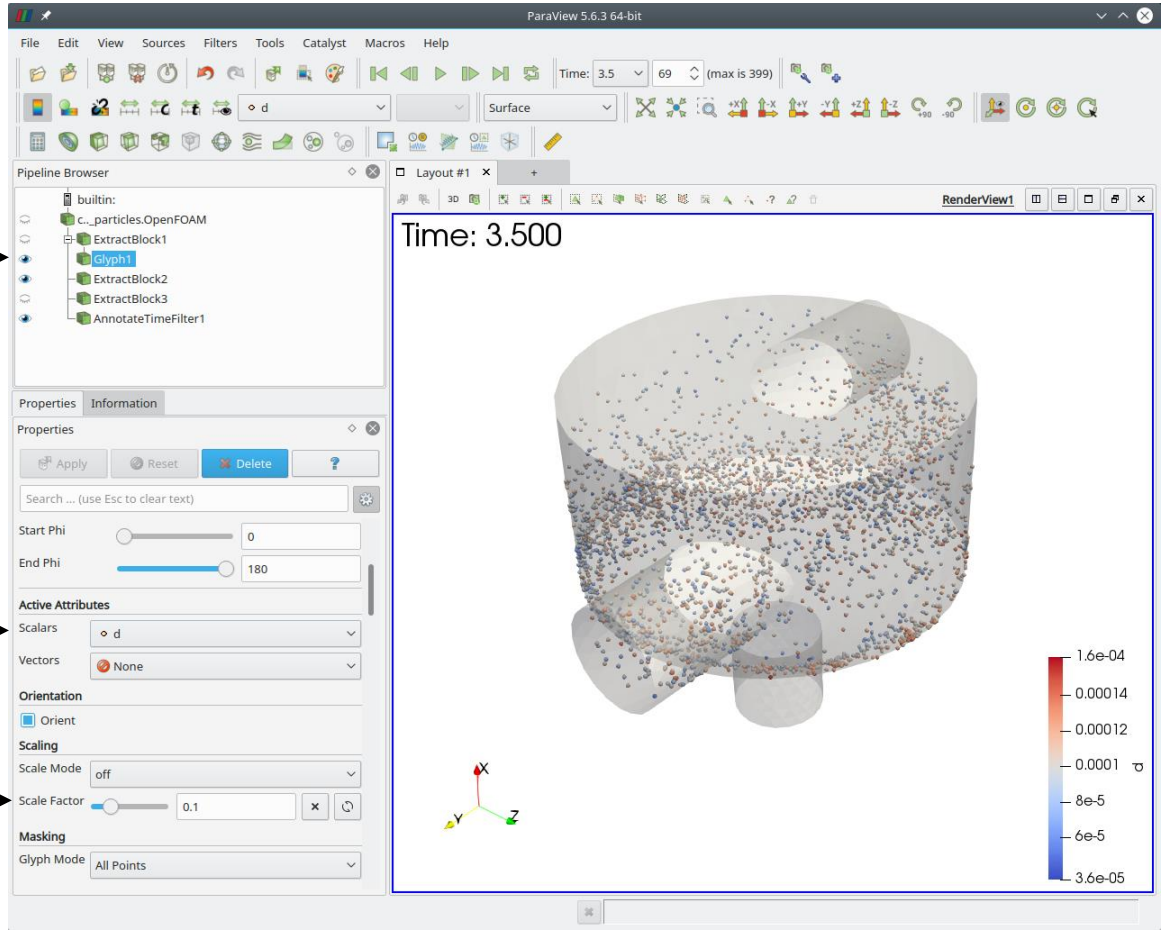
Guided tutorial 8 – Particle injection in a mixing tank

Visualizing Lagrangian fields in paraFoam

- Apply the **Glyph** filter to the lagrangian field data.

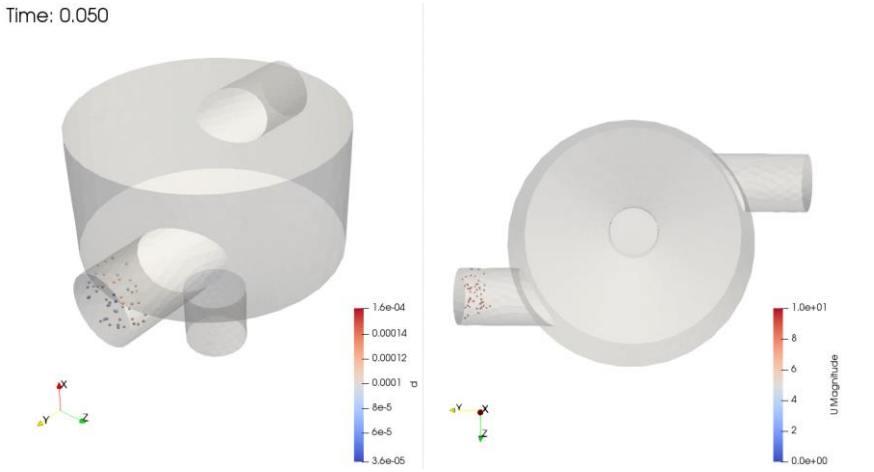
- Apply an scalar or vector attribute to the **Glyph** filer.

- Scale the **Glyph** data.
- In this case, we are scaling the **Glyph** data using a constant value equal to 0.1.
- You can also scale the data using attributes, for example, particle diameter.

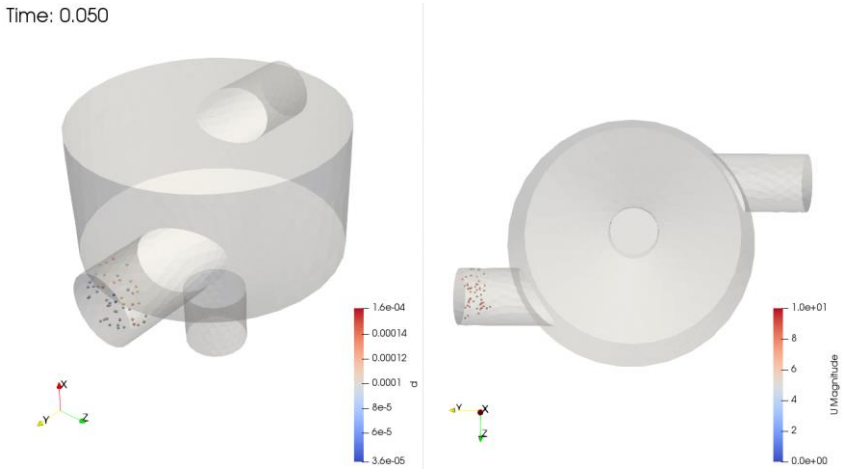


Guided tutorial 8 – Particle injection in a mixing tank

- Visualization of the particles evolution. Note that the injection ends after 10 seconds.
- In the figures, we can see the influence of the gravity vector.
  - Left image: gravity vector in the -Y direction.
  - Right image: gravity vector in the -X direction.
- Note that independent of the gravity vector, particles remain trapped in boundary layer (the particles are not to scale).



http://www.wolfdynamics.com/training/mphase/gv1.gif

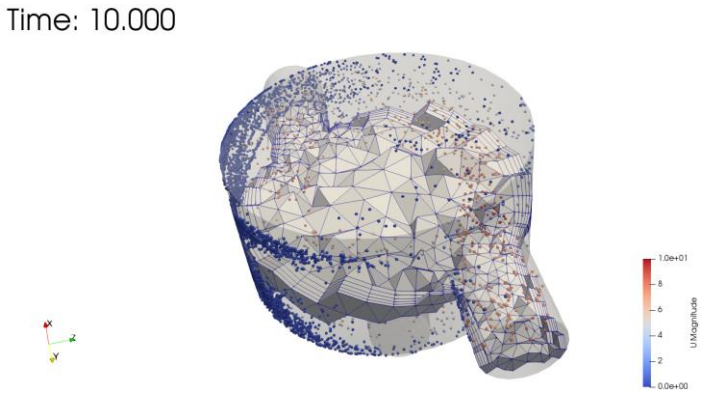


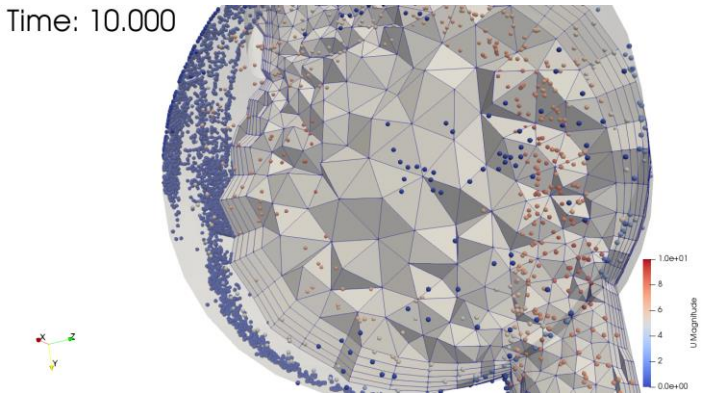
http://www.wolfdynamics.com/training/mphase/gv2.gif

Guided tutorial 8 – Particle injection in a mixing tank

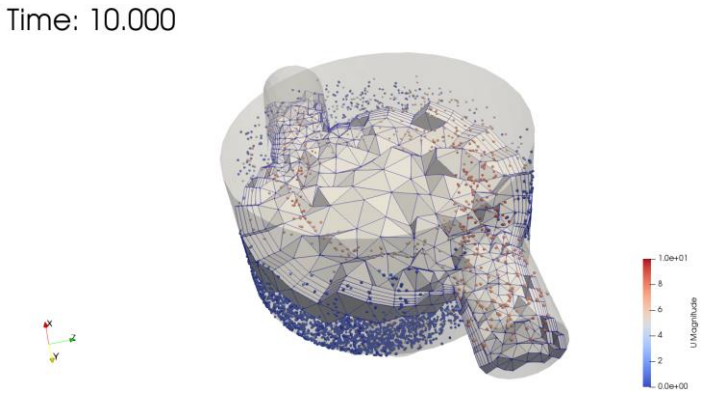
- Note that independent of the gravity vector orientation, particles remain trapped in boundary layer.
- The particles stick at the wall due to large size of the boundary layer mesh, among many other reasons.
- Have in mind that in the figures the particles are not to scale, they have been exaggerated.

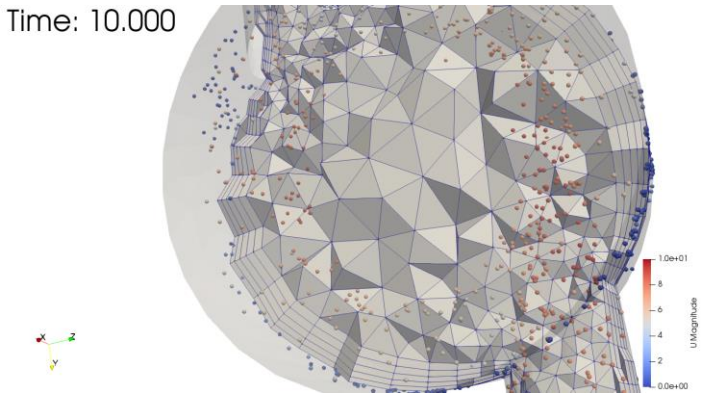
Gravity vector (-X, 0, 0)





Gravity vector (0, 0, -Z)





Guided tutorials

- **Case2.** Particles tracking tutorial with hydrodynamic coupling
- This case is ready to run.
- The case is located in the directory:

$TM/multiphase/guided_tutorials/GT8/MPPIC

- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are human-readable, and we highly recommend you open them, get familiar with the steps, and type the commands in the terminal. In this way, you will get used with the command line interface and OpenFOAM commands.
- If you are already comfortable with OpenFOAM, run the cases automatically using these scripts.
- In the case directory, you will also find the `README.FIRST` file. In this file, you will find some additional comments.

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- The MPPIC approach maps the discrete phase particle properties on the Eulerian grid of the continuous phase.
- Then, after the solution of the continues flow, the quantities of the discrete phase are updated by the model.
- The actual particle size distribution (a probabilistic function) is synthetized into a smaller number of computational particles that are directly solved by the Lagrangian method with a strong reduction of the computational requirements of the model.
- As for the other Lagrangian solvers implemented in OpenFOAM, non-spherical particles can be modelled with he MPPIC approach by choosing a proper drag coefficient and aspect ratio model.

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

Particle injection in a mixing tank

- At this point, we are ready to run the simulation.
- To run the case, type in the terminal:

```
1. $> run_all_parallel.sh
```

- This script will run the simulation in parallel using four cores.
- If you want to run in serial, use the script `run_all.sh`.
- Feel free to open the scripts to see all the commands used.

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *transportProperties* located in the directory **constant**.

The name of the continuous phase (user given). Notice that every dictionary related to the continuous phase will use this word when naming the file.

Density of the continuous phase

Transport model and viscosity of the continuous phase

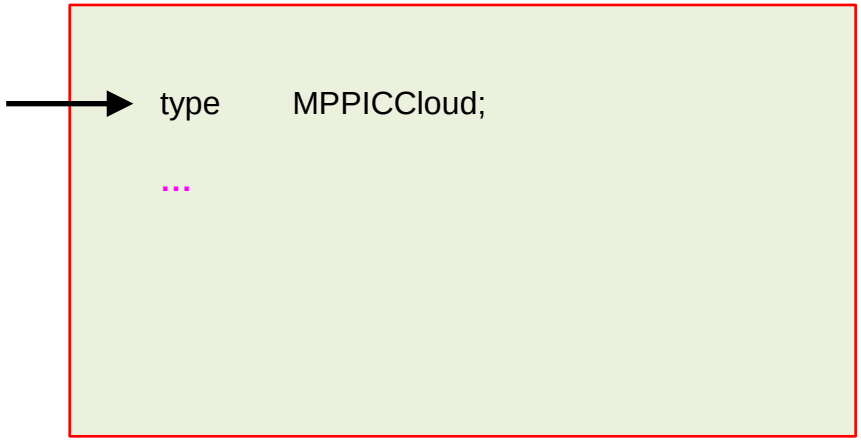
```
continuousPhaseName air;

rho.air    1.2;

transportModel   Newtonian;
nu  1e-05;
```

- The *momentumTransport.air* dictionary contains information on the turbulence model of the continuous phase (the Eulerian mesh).

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *transportProperties* located in the directory **constant**.
- The dictionary *cloudProperties* requires special attention, as it is in this dictionary where we set the interaction models and injection models of the particles.
- If you are using the **MPPIC** approach, the dictionary should look like this one:

- The standard set of Lagrangian clouds are now selectable at run-time.
- This means that a solver that supports Lagrangian modelling can now use any type of cloud (with some restrictions).
- Previously, solvers were hard-coded to use specific cloud modelling. In addition, a cloud-list structure has been added so that solvers may select multiple clouds, rather than just one.
- More information at the following link, https://github.com/OpenFOAM/OpenFOAM-9/commit/43d66b5e7c566e087db187df023f134422ee8a4b



Continues in the next slide (1/9)

Note:

- In the directory OpenFOAM-9/src/lagrangian/parcel/clouds/Templates/MomentumCloud/cloudSolution/ you will find the source code of the momentum cloud where you can find some information about the flags available for the solution of the cloud.
- In the directory **OpenFOAM-9/src/lagrangian/parcel/clouds/Templates** you will find the source code of the different cloud types.

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *cloudProperties* located in the directory **constant**.

- Activation switches.
- The coupled switch turns on/off coupled hydrodynamics.
- The transient switch turns on/off transient coupling.

Flag to correct cell values with latest transfer information during the lagrangian timestep

Interpolation method of cell centered quantities to particle solver.

Interpolation of the average of the lagrangian phase. Important only if you are using a packing model.

```
solution
{

        coupled         true;
        transient       yes;

        maxCo           0.7;

        cellValueSourceCorrection on;

        interpolationSchemes
        {
                rho.air         cell;
                U.air           cellPoint;
                mu.air          cell;
                alpha.air       cell;

        }

        averagingMethod     dual;
...
```

Maximum currant number of the Lagrangian mesh.

Continues in the next slide (2/9)

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *cloudProperties* located in the directory **constant**.

Time integration schemes of particles.

Additional source terms

Physical properties of the particles.

```
...

    integrationSchemes
    {
        U               Euler;
    }

    sourceTerms
    {
        schemes
        {
            U           semiImplicit 1;
        }
    }
}

constantProperties
{
    rho0                1000;
}

...
```

Continues in the next slide (3/9)

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *cloudProperties* located in the directory **constant**.

Selection of sub-models.

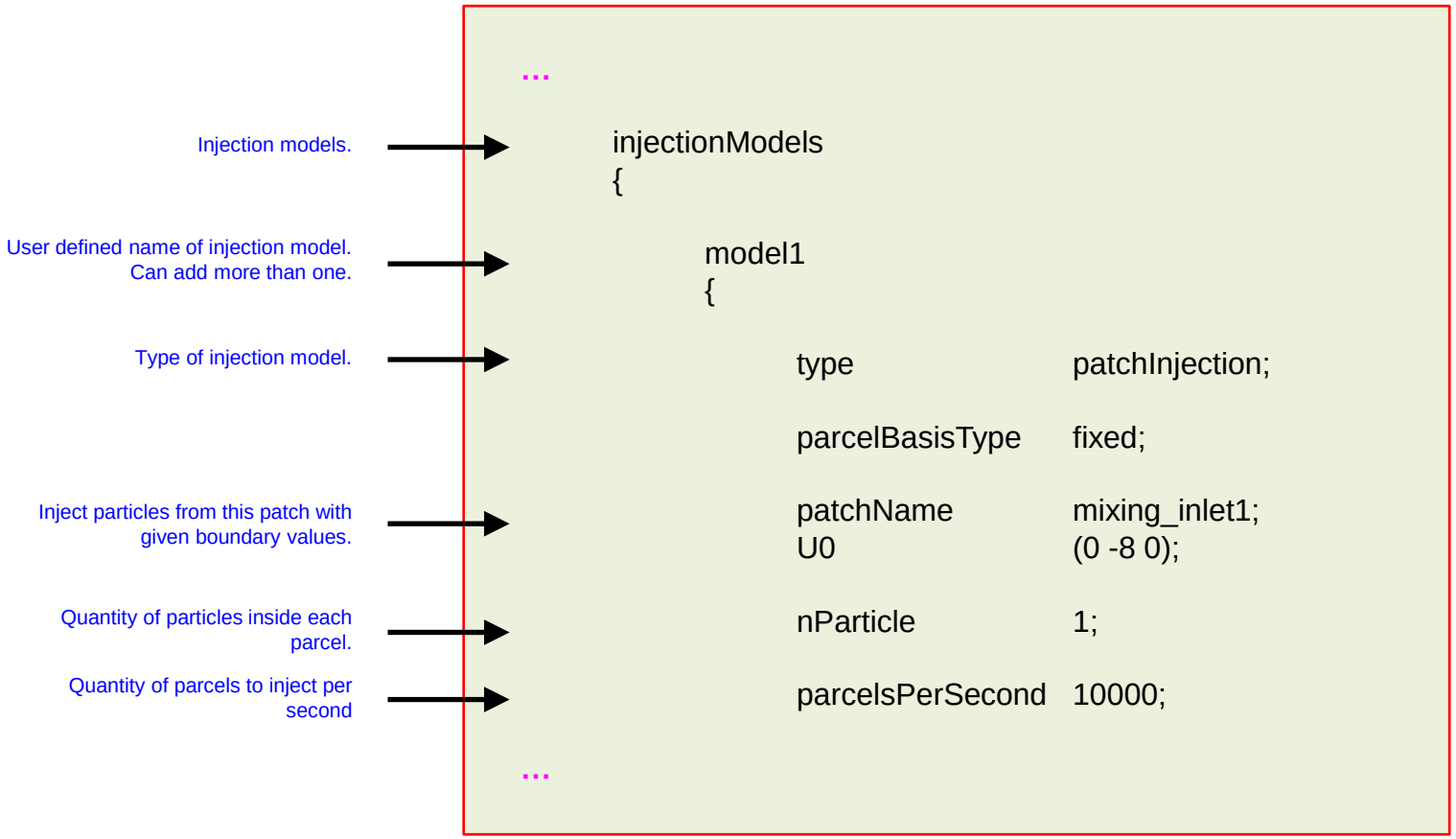
Particle forces models.

```
...

subModels
{


        particleForces
        {
                ErgunWenYuDrag
                {
                        alphac alpha.air;
                }

                gravity;
        }

...
```

Continues in the next slide (4/9)

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *cloudProperties* located in the directory **constant**.



Continues in the next slide (5/9)

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *cloudProperties* located in the directory **constant**.

A valid Function1 type to scale the flow rate profile. In this case, we are using a constant function.

Injection parameters. Total mass to inject into the system, start time of injection and duration of injection.

Model for diameter distribution of particles.

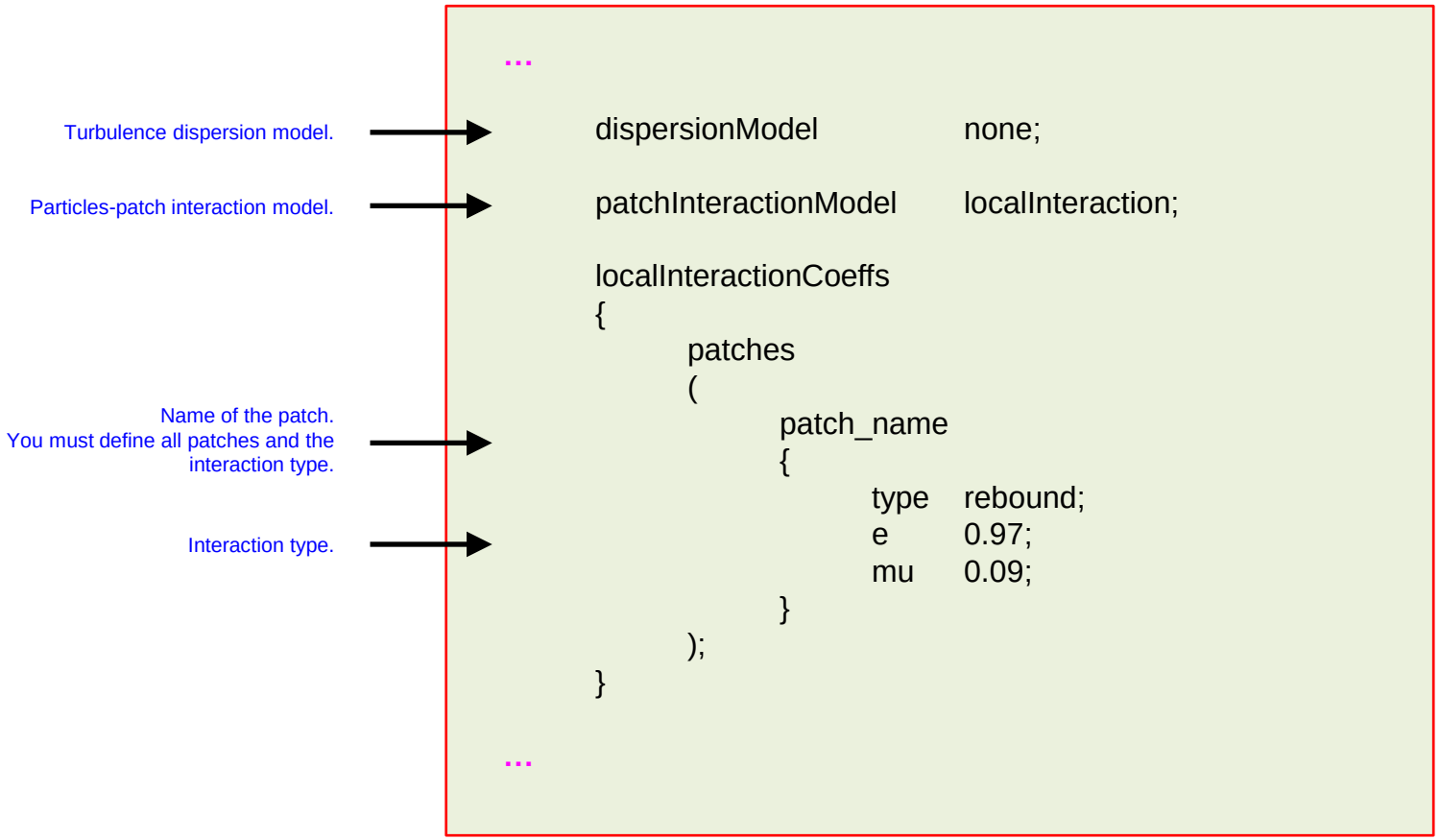
Diameter distribution values.

```
...
        flowRateProfile constant    1;

        massTotal           0;
        SOI                 0;
        duration            5;

        sizeDistribution
        {
            type        normal;
            normalDistribution
            {
                expectation 100e-6;
                variance    25e-6;
                minValue    20e-6;
                maxValue    180e-6;
            }
        }
    }

...
```

Continues in the next slide (6/9)

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *cloudProperties* located in the directory **constant**.



Continues in the next slide (7/9)

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- Let us explore the dictionary *cloudProperties* located in the directory **constant**.

Particles packaging model used to apply non-linear stresses to particle parcels.

Coefficients related to the packing method selected

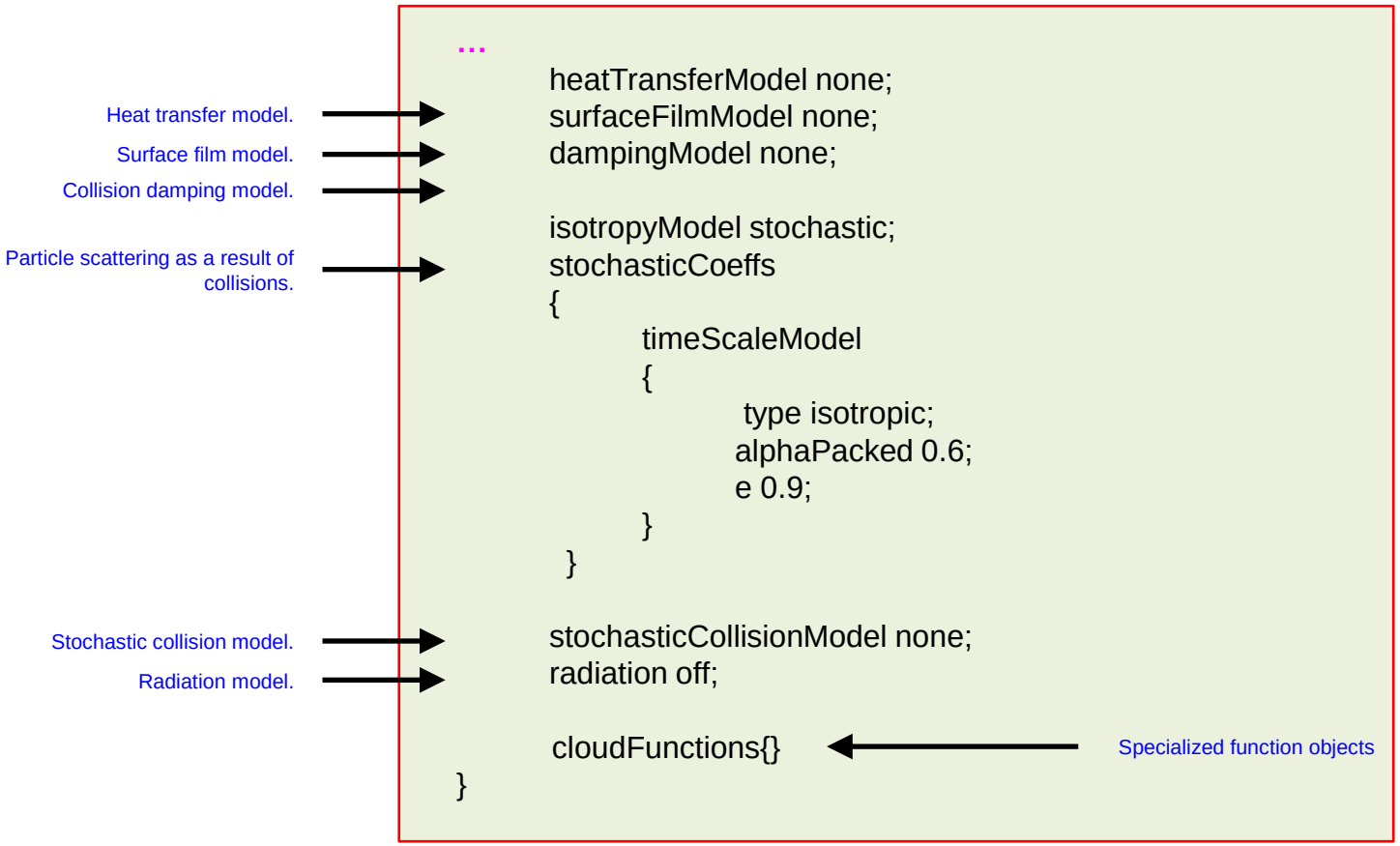
Particle stresses sub-model.

```
...

    //packingModel none;
    //packingModel explicit;
    packingModel implicit;

    implicitCoeffs
    {
        alphaMin        0.0001;
        rhoMin          1.0;
        applyGravity    false;
        particleStressModel

        {
            HarrisCrighton;
            alphaPacked     0.6;
            pSolid          5.0;
            beta            2.0;
            eps             1.0e-2;
        }
    }
```

Continues in the next slide (8/9)

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

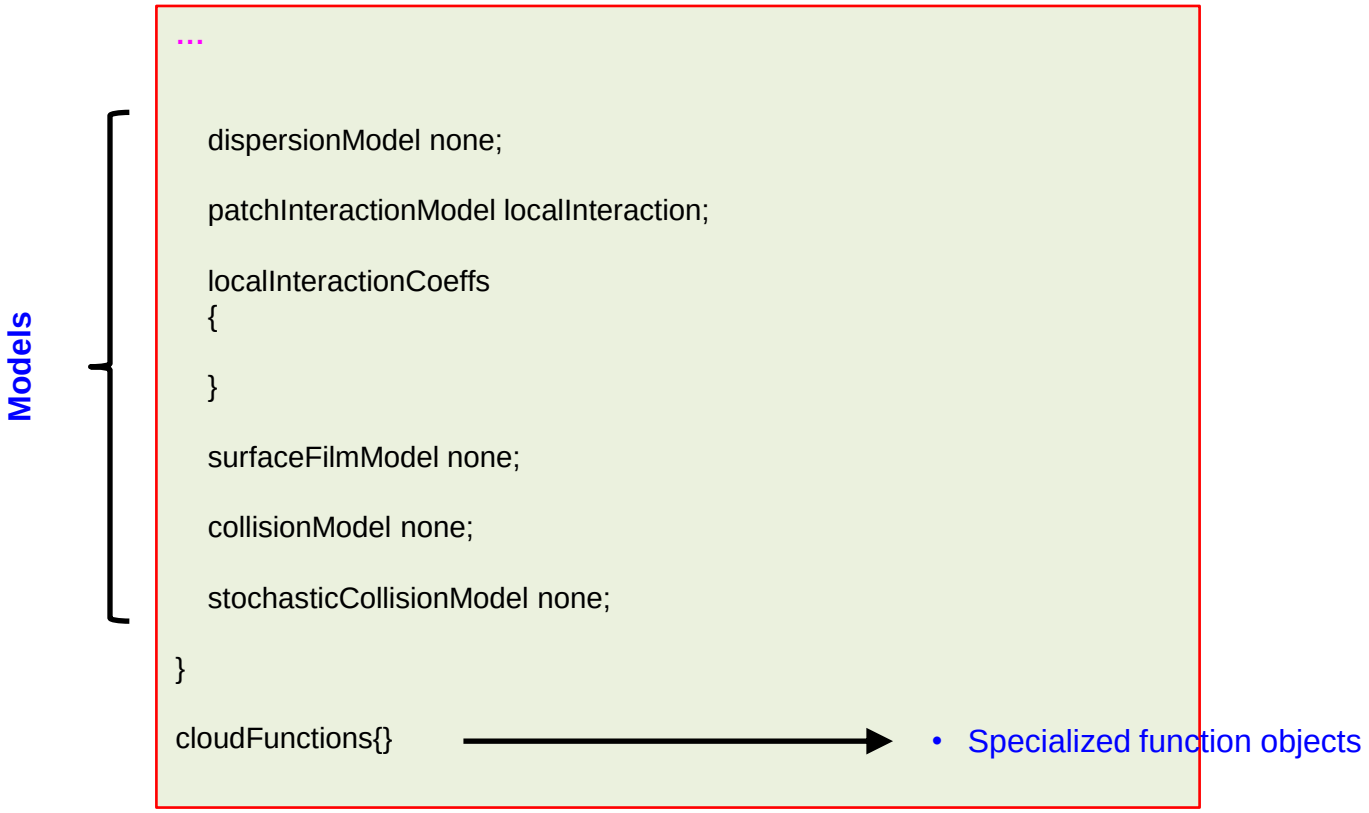
- Let us explore the dictionary *cloudProperties* located in the directory **constant**.



Continues in the next slide (9/9)

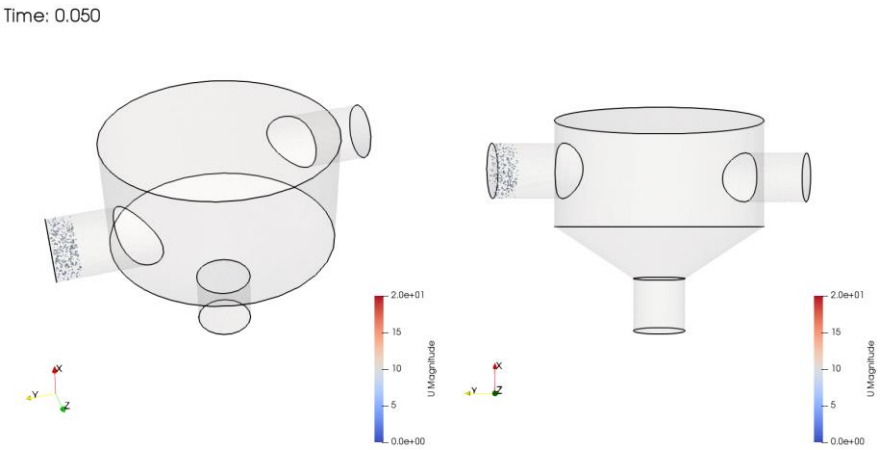
Guided tutorial 8 – Particle injection in a mixing tank

- The dictionary *cloudProperties* requires special attention, as it is in this dictionary where we set the interaction models and injection models of the particles.
- Remember, this dictionary is located in the directory **constant**.



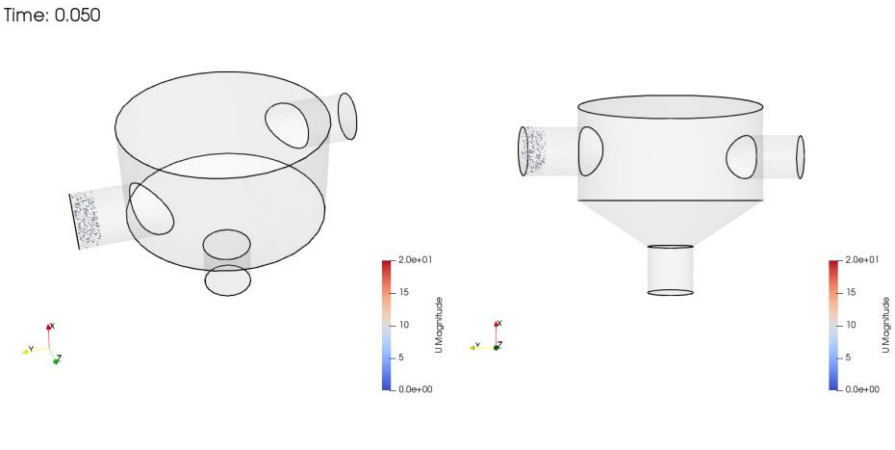
Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- At the end of the day, you should get something like this:



Laminar solution – No packing model

http://www.wolfdynamics.com/training/mphase/image54.gif



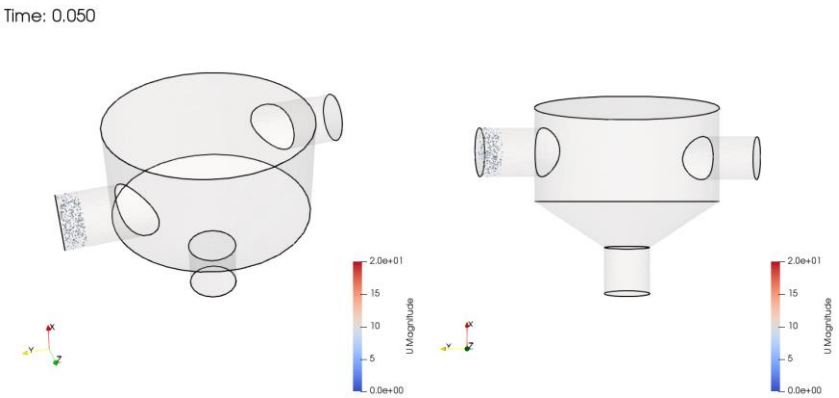
Turbulent solution – No packing model

http://www.wolfdynamics.com/training/mphase/image55.gif

Note: particles scaled by a factor of 200 times.

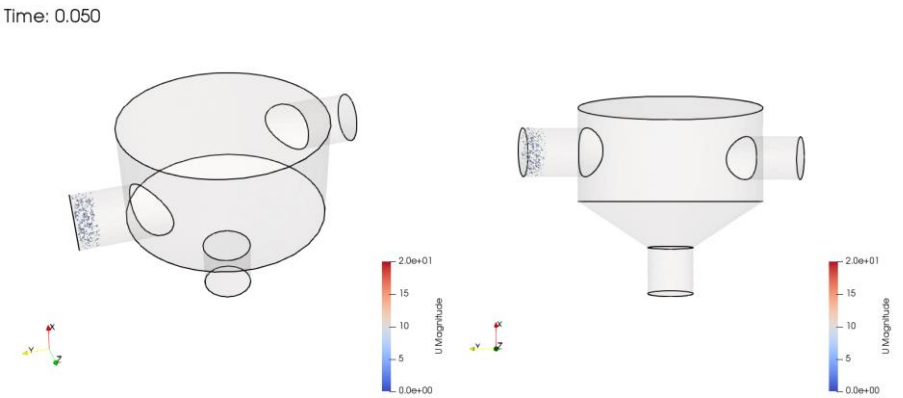
Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- At the end of the day, you should get something like this:



Turbulent solution – Explicit packing model

http://www.wolfdynamics.com/training/mphase/image56a.gif



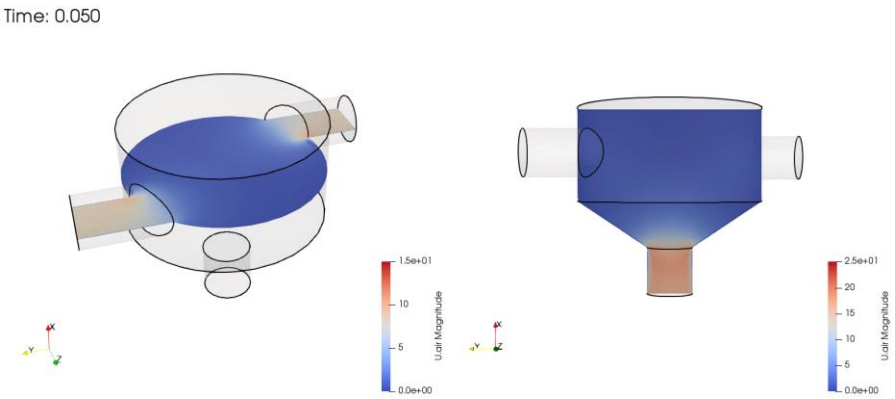
Turbulent solution – Implicit packing model

http://www.wolfdynamics.com/training/mphase/image56.gif

Note: particles scaled by a factor of 200 times.

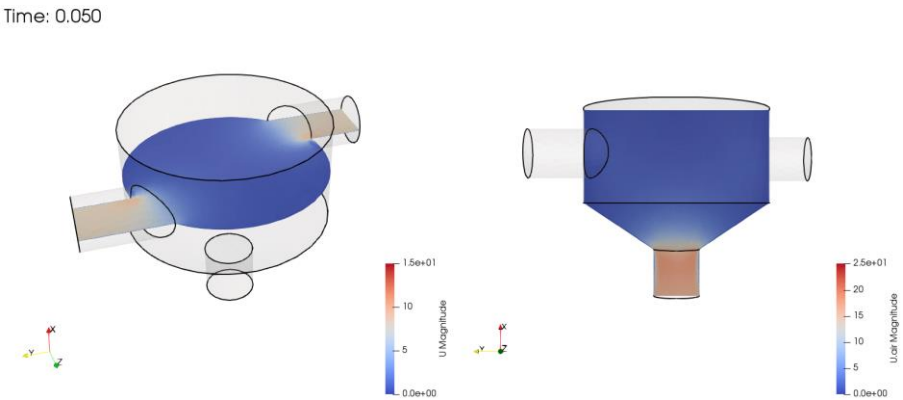
Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- At the end of the day, you should get something like this:



Turbulent solution – No packing model

http://www.wolfdynamics.com/training/mphase/image57.gif



Turbulent solution – Implicit packing model

http://www.wolfdynamics.com/training/mphase/image58.gif

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- While running the simulation, the lagrangian solver reports the following information.

```
Courant Number mean: 0.00703507842106 max: 0.258287904711
deltaT = 0.000246788667208
Time = 0.000889055

Solving 3-D cloud delta

Cloud: cloud injector: model1
    Added 3 new parcels

Cloud: cloud
    Current number of parcels       = 9
    Current mass in system          = 4.82585484211e-09
    Linear momentum                 = (-6.33534842005e-11 -3.83183263286e-08 7.7080707659e-12)
   |Linear momentum|                = 3.83183794765e-08
    Linear kinetic energy           = 1.52136853593e-07
    model1:
        number of parcels added     = 9
        mass introduced             = 4.82585484211e-09
    Parcel fate (number, mass)      : patch mixer_wall
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_inlet1
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_inlet2
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_outlet
      - escape                      = 0, 0
      - stick                       = 0, 0
    Min cell volume fraction        = 0
    Max cell volume fraction        = 4.24152704156e-09
    Min dense number of parcels     = 1
```

- Lagrangian information related to the number of particles, linear momentum, fate of the particles, and so on.
- The information printed may change depending on the models used.

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- While running the simulation, the lagrangian solver reports the following information.

```
Cloud: cloud
    Current number of parcels       = 9
    Current mass in system          = 4.82585484211e-09

    Linear momentum                 = (-6.33534842005e-11 -3.83183263286e-08 7.7080707659e-12)
   |Linear momentum|                 = 3.83183794765e-08
    Linear kinetic energy           = 1.52136853593e-07

    model1:
        number of parcels added     = 9
        mass introduced             = 4.82585484211e-09
    Parcel fate (number, mass)      : patch mixer_wall
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_inlet1
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_inlet2
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_outlet
      - escape                      = 0, 0
      - stick                       = 0, 0

    Min cell volume fraction        = 0
    Max cell volume fraction        = 4.24152704156e-09

    Min dense number of parcels     = 1
```

- Number and mass of particles in the system
- Linear and angular momentum
- Number and mass of the particle stick/escaped at a given patch
- Min/Max alpha of the disperse phase.
- Number of particles in the smallest cell

Guided tutorial 8 – Particle injection in a mixing tank with hydrodynamic coupling

- While running the simulation, the lagrangian solver reports the following information.

```
Courant Number mean: 0.151742046621 max: 4.62296776031
deltaT = 0.00416666666667
Time = 0.1125

Solving 3-D cloud cloud

Cloud: cloud injector: model1
    Added 42 new parcels

GAMG:  Solving for cloud:alpha, Initial residual = 0.000609328928101, Final residual = 1.9643710767e-05, No Iterations 1000
Cloud: cloud
    Current number of parcels       = 1123
    Current mass in system          = 6.3447511763e-07
    Linear momentum                 = (-2.25890764653e-08 -4.38682007539e-06 3.74741765096e-07)
   |Linear momentum|                = 4.40285498634e-06
    Linear kinetic energy           = 1.63024837905e-05
    model1:
        number of parcels added     = 1123
        mass introduced             = 6.3447511763e-07
    Parcel fate (number, mass)      : patch mixer_wall
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_inlet1
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_inlet2
      - escape                      = 0, 0
      - stick                       = 0, 0
    Parcel fate (number, mass)      : patch mixer_outlet
      - escape                      = 0, 0
      - stick                       = 0, 0
    Min cell volume fraction        = 0
    Max cell volume fraction        = 1.89459145677e-08
    Min dense number of parcels     = 2.85371747396
```

- Some Lagrangian models need to solve a linear system.
- For example, when using **implicit packing**.
- These iterations are fast. But be careful of unconverged iterations.
- In this case we have reached the maximum number of iterationx, so this iteration is considered unconverged.
- Even if the initial residual are relatively low.