

# **Module 8**

**Advanced physics**

**Turbulence modeling – Multiphase flows –  
Compressible flows – Source terms –  
Passive scalars – Moving bodies**

- In this module, we will deal with advanced modeling capabilities.
- Advanced modeling capabilities rely a lot in physical models, such as, turbulence, multiphase flows, porous media, combustion, radiation, heat transfer, phase change, acoustics, cavitation, and so on.
- Therefore, it is extremely important to get familiar with the theory behind these models.

“Essentially, all models are wrong,  
but some are useful”

G. E. P. Box



**George Edward Pelham Box**

18 October 1919 – 28 March 2013. Statistician, who worked in the areas of quality control, time-series analysis, design of experiments, and Bayesian inference. He has been called “*one of the great statistical minds of the 20th century*”.

# Roadmap

**A crash introduction to:**

- 1. Turbulence modeling in OpenFOAM®**
- 2. Multiphase flows modeling in OpenFOAM®**
- 3. Compressible flows in OpenFOAM®**
- 4. Source terms in OpenFOAM®**
- 5. Scalar transport pluggable solver**
- 6. Moving bodies in OpenFOAM®**

# Roadmap

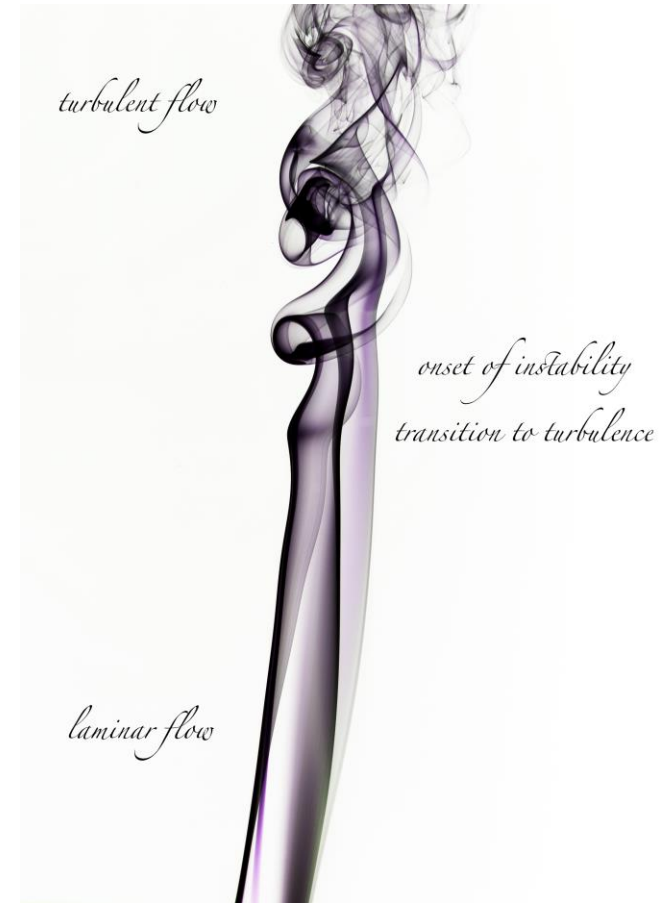
**A crash introduction to:**

- 1. Turbulence modeling in OpenFOAM®**
2. Multiphase flows modeling in OpenFOAM®
3. Compressible flows in OpenFOAM®
4. Moving bodies in OpenFOAM®
5. Source terms in OpenFOAM®
6. Scalar transport pluggable solver

# A crash introduction to turbulence modeling in OpenFOAM®

## What is turbulence?

- For the purpose of this training, let us state the following:
  - Turbulence is an unsteady, aperiodic motion in which all three velocity components fluctuate in space and time.
  - Every transported quantity shows similar fluctuations (pressure, temperature, species, concentration, and so on)
  - Turbulent flows contains a wide range of eddy sizes (scales):
    - Large eddies derives their energy from the mean flow. The size and velocity of large eddies are on the order of the mean flow.
    - Large eddies are unstable and they break-up into smaller eddies.
    - The smallest eddies convert kinetic energy into thermal energy via viscous dissipation.
    - The behavior of small eddies is more universal in nature.



**Buoyant plume of smoke rising from a stick of incense**  
Photo credit: <https://www.flickr.com/photos/jlhopgood/>  
This work is licensed under a Creative Commons License  
(CC BY-NC-ND 2.0)

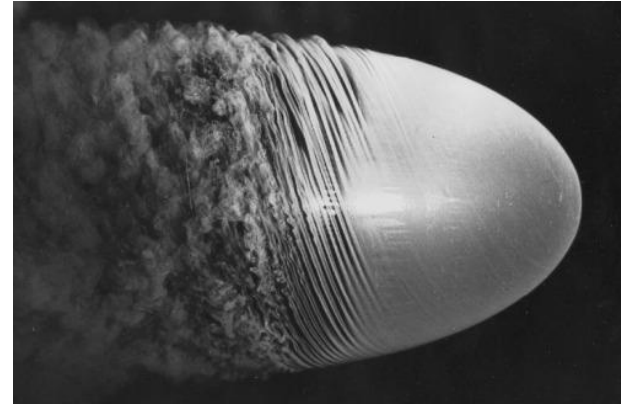
# A crash introduction to turbulence modeling in OpenFOAM®



## Wake turbulence behind individual wind turbines

Photo credit: NREL's wind energy research group.

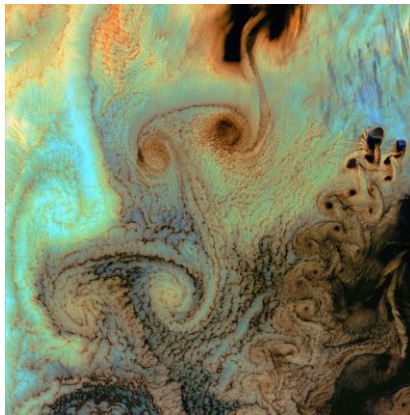
Copyright on the images is held by the contributors. Apart from Fair Use, permission must be sought for any other purpose.



## Flow visualization over a spinning spheroid

Photo credit: Y. Kohama.

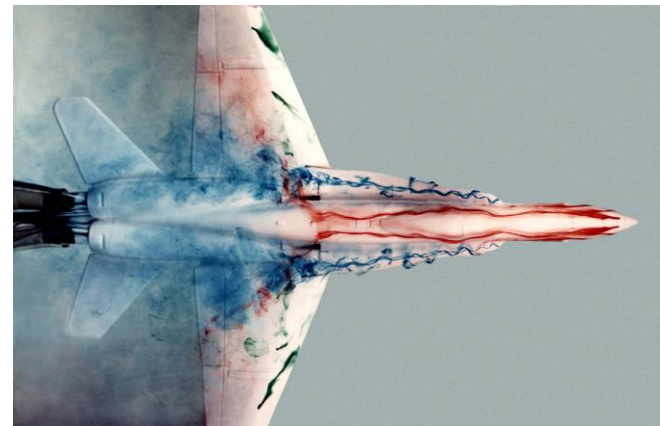
Copyright on the images is held by the contributors. Apart from Fair Use, permission must be sought for any other purpose.



## Von Karman vortices created when prevailing winds sweeping east across the northern Pacific Ocean encountered Alaska's Aleutian Islands

Photo credit: USGS EROS Data Center Satellite Systems Branch.

Copyright on the images is held by the contributors. Apart from Fair Use, permission must be sought for any other purpose.



## Vortices on a 1/48-scale model of an F/A-18 aircraft inside a Water Tunnel

Photo credit: NASA Dryden Flow Visualization Facility.

Copyright on the images is held by the contributors. Apart from Fair Use, permission must be sought for any other purpose.

# A crash introduction to turbulence modeling in OpenFOAM®

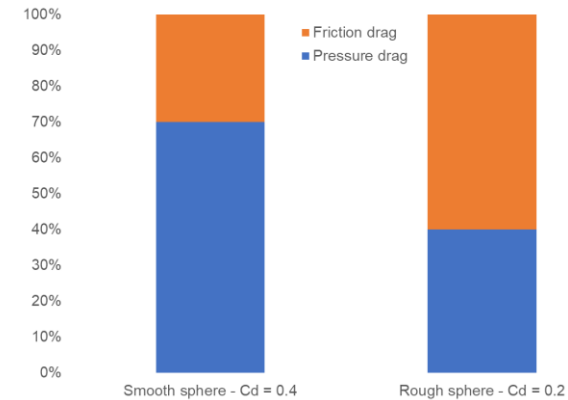
## Turbulence, does it matter?



(a)  $C_D \approx 0.4$



(b)  $C_D \approx 0.2$



**Abstract representation of the drag decomposition**

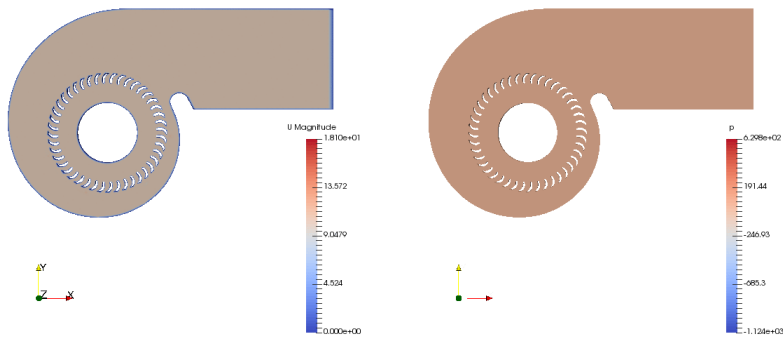
# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence, does it matter?

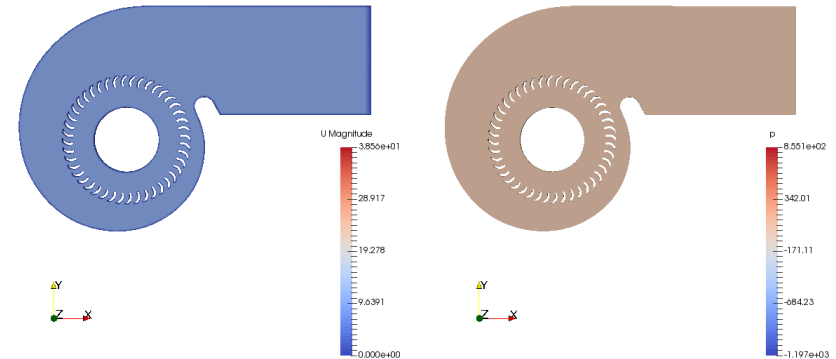
Blower simulation using sliding grids



Time: 0.000000



Time: 0.000000



**No turbulence model used (laminar, no turbulence modeling, DNS, unresolved DNS, name it as you want)**

<http://www.wolfdynamics.com/training/turbulence/image1.gif>

**K-epsilon turbulence model**

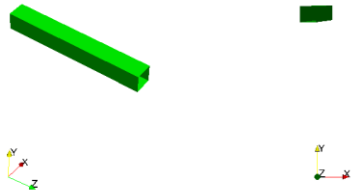
<http://www.wolfdynamics.com/training/turbulence/image2.gif>



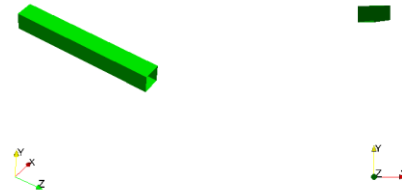
# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence, does it matter?

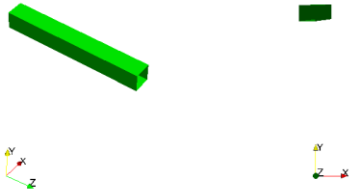
Vortex shedding past square cylinder



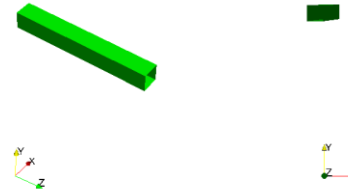
**URANS (K-Omega SST with no wall functions) – Vortices visualized by Q-criterion**  
[www.wolfdynamics.com/wiki/squarecil/urans2.gif](http://www.wolfdynamics.com/wiki/squarecil/urans2.gif)



**LES (Smagorinsky) – Vortices visualized by Q-criterion**  
[www.wolfdynamics.com/wiki/squarecil/les.gif](http://www.wolfdynamics.com/wiki/squarecil/les.gif)



**Laminar (no turbulence model) – Vortices visualized by Q-criterion**  
[www.wolfdynamics.com/wiki/squarecil/laminar.gif](http://www.wolfdynamics.com/wiki/squarecil/laminar.gif)



**DES (SpalartAllmarasDDES) – Vortices visualized by Q-criterion**  
[www.wolfdynamics.com/wiki/squarecil/des.gif](http://www.wolfdynamics.com/wiki/squarecil/des.gif)

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence, does it matter?

Vortex shedding past square cylinder

Turbulence model	Drag coefficient	Strouhal number	Computing time (s)
Laminar	2.81	0.179	93489
LES	2.32	0.124	77465
DES	2.08	0.124	70754
URANS (WF)	2.31	0.130	67830
URANS (No WF)	2.28	0.135	64492
RANS	2.20	-	28246 (10000 iter)
Experimental values	2.05-2.25	0.132	-

**Note:** all simulations were run using 4 cores.

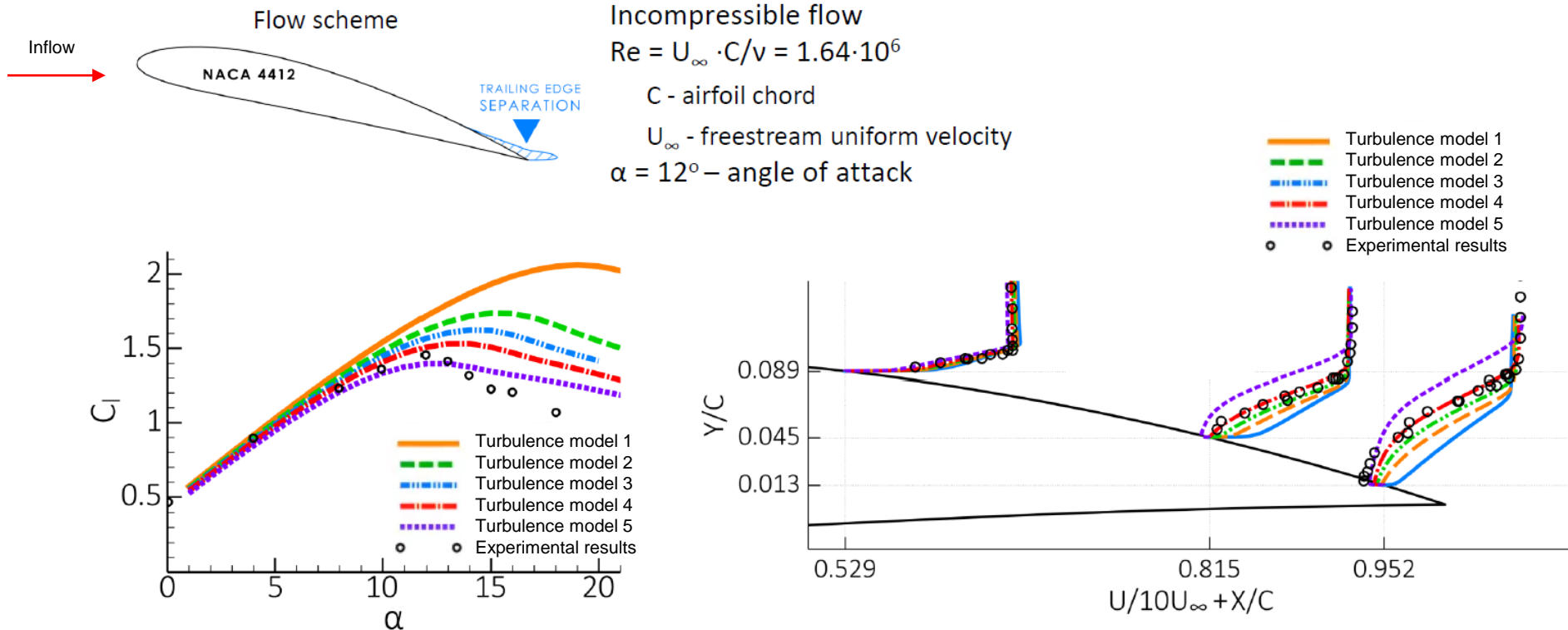
### References:

Lyn, D.A. and Rodi, W., The flapping shear layer formed by flow separation from the forward corner of a square cylinder. *J. Fluid Mech.*, 267, 353, 1994.  
Lyn, D.A., Einav, S., Rodi, W. and Park, J.H., A laser-Doppler velocimetry study of ensemble-averaged characteristics of the turbulent near wake of a square cylinder. *Report. SFB 210 /E/100*.

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence, does it matter?

### Separated flow around a NACA-4412 airfoil



- CFD has been around since the late 1970s, and after all these years is not that easy to compute the flow around 2D airfoils.
- In particular, predicting the maximum lift and stall characteristics is not trivial.

#### References:

- F. Menter. "A New Generalized k-omega model. Putting flexibility into Turbulence models (GEKO)", Ansys Germany  
A. J. Wadcock. "Investigation of Low-Speed Turbulent Separated Flow Around Airfoils", NASA Contractor Report 177450

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence modeling in engineering

- Most natural and engineering flows are turbulent, hence the necessity of modeling turbulence.
- The goal of turbulence modeling is to develop equations that predict the time averaged velocity, pressure, temperature fields without calculating the complete turbulent flow pattern as a function of time.
- There is no universal turbulence model, hence you need to know the capabilities and limitations of the turbulence models.
- Simulating turbulent flows in any general CFD solver requires selecting a turbulence model, providing initial conditions and boundary conditions for the closure equations of the turbulent model, selecting a near-wall modeling, and choosing runtime parameters and numerics.

## Why modeling turbulent flows is challenging?

- Unsteady aperiodic motion.
- All fluid properties and transported quantities exhibit random spatial and temporal variations.
- They are intrinsically three-dimensional due to vortex stretching.
- Strong dependence from initial conditions.
- Contains a wide range of scales (eddies).
- Therefore, in order to accurately model/resolve turbulent flows, the simulations must be three-dimensional, time-accurate, and with fine enough meshes such that all spatial scales are resolved.

# A crash introduction to turbulence modeling in OpenFOAM®

## Reynolds number and Rayleigh number

- It is well known that the Reynolds number characterizes if the flow is laminar or turbulent.
- So before doing a simulation or experiment, check if the flow is turbulent.
- The Reynolds number is defined as follows,

$$\text{Convective effects} \xrightarrow{\quad} Re_L = \frac{\rho U L}{\mu} \xleftarrow{\quad} \text{Viscous effects} \quad \text{where} \quad L = x, d, d_h, \text{ etc}$$

- Where  $U$  and  $L$  are representative velocity and length scales.
- If you are dealing with natural convection, you can use the Rayleigh number, Grashof number, and Prandtl number to characterize the flow.

$$\begin{array}{l} \text{Buoyancy effects} \xrightarrow{\quad} Ra = \frac{g \beta L^3 \Delta T}{\nu \alpha} = \frac{\rho^2 \overset{\text{Specific heat}}{c_p} \overset{\text{Thermal expansion coefficient}}{\beta} g L^3 \Delta T}{\mu \overset{\text{Thermal conductivity}}{k}} = Gr \times Pr \\ \text{Viscous effects} \xrightarrow{\quad} \end{array}$$

$$\begin{array}{l} \text{Momentum diffusivity} \xrightarrow{\quad} Pr = \frac{\nu}{\alpha} = \frac{\mu c_p}{k} \\ \text{Thermal diffusivity} \xrightarrow{\quad} \end{array} \quad Gr = \frac{g \beta (T_S - T_\infty) L^3}{\nu^2}$$

## Reynolds number and Rayleigh number

- Turbulent flow occurs at large Reynolds number.

- For external flows,

$$Re_x \geq 500000 \quad \text{Around slender/streamlined bodies (surfaces)}$$

$$Re_d \geq 20000 \quad \text{Around an obstacle (bluff body)}$$

- For internal flows,

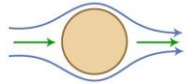
$$Re_{d_h} \geq 2300$$

- Notice that other factors such as free-stream turbulence, surface conditions, blowing, suction, roughness and other disturbances, may cause transition to turbulence at lower Reynolds number.
- If you are dealing with natural convection and buoyancy, turbulent flows occurs when

$$\frac{Ra}{Pr} \geq 10^9$$

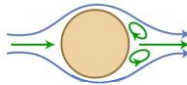
# A crash introduction to turbulence modeling in OpenFOAM®

## What happens when we increase the Reynolds number?



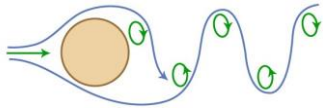
**Creeping flow (no separation)**  
Steady flow

$$Re < 5$$



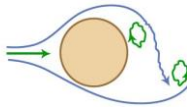
**A pair of stable vortices  
in the wake**  
Steady flow

$$5 < Re < 40 - 46$$



**Laminar vortex street  
(Von Karman street)**  
Unsteady flow

$$40 - 46 < Re < 150$$

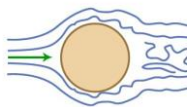


**Laminar boundary layer up to  
the separation point, turbulent  
wake**  
Unsteady flow

$$150 < Re < 300$$

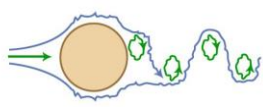
Transition to turbulence

$$300 < Re < 3 \times 10^5$$



**Boundary layer transition to  
turbulent**  
Unsteady flow

$$3 \times 10^5 < Re < 3 \times 10^6$$



**Turbulent vortex street, but the  
wake is narrower than in the  
laminar case**  
Unsteady flow

$$3 \times 10^6 > Re$$

- Easy to simulate
- Steady

- Relatively easy to simulate.
- It becomes more challenging when the boundary layer transition to turbulent
- Unsteady

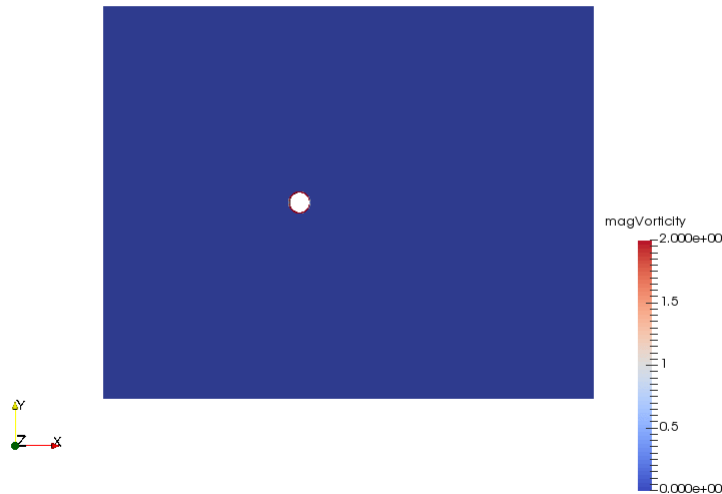
- Challenging to simulate
- Unsteady

Vortex shedding behind a cylinder and Reynolds number

# A crash introduction to turbulence modeling in OpenFOAM®

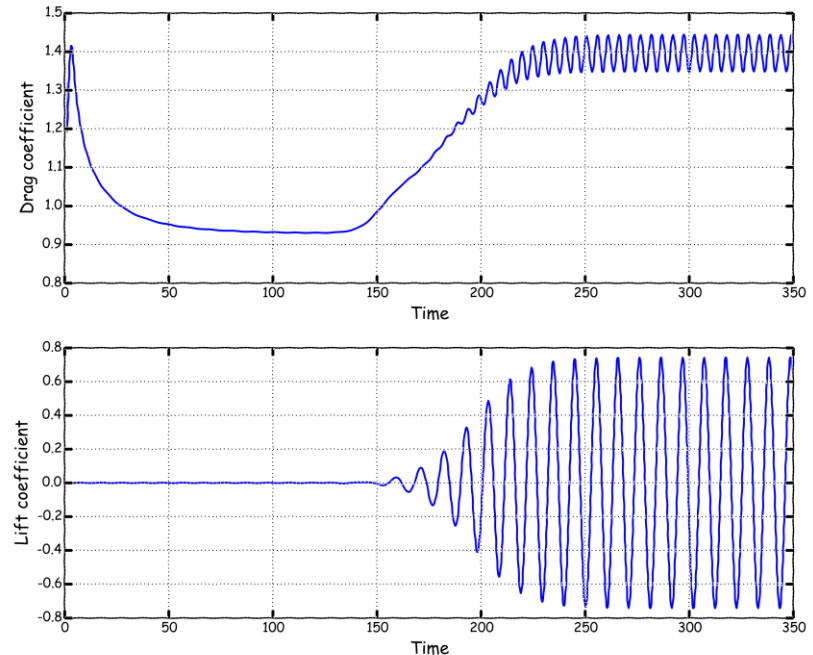
## Vorticity does not always mean turbulence

Time: 0.000000



### Instantaneous vorticity magnitude field

[www.wolfodynamics.com/wiki/cylinder\\_vortex\\_shedding/movvort.gif](http://www.wolfodynamics.com/wiki/cylinder_vortex_shedding/movvort.gif)

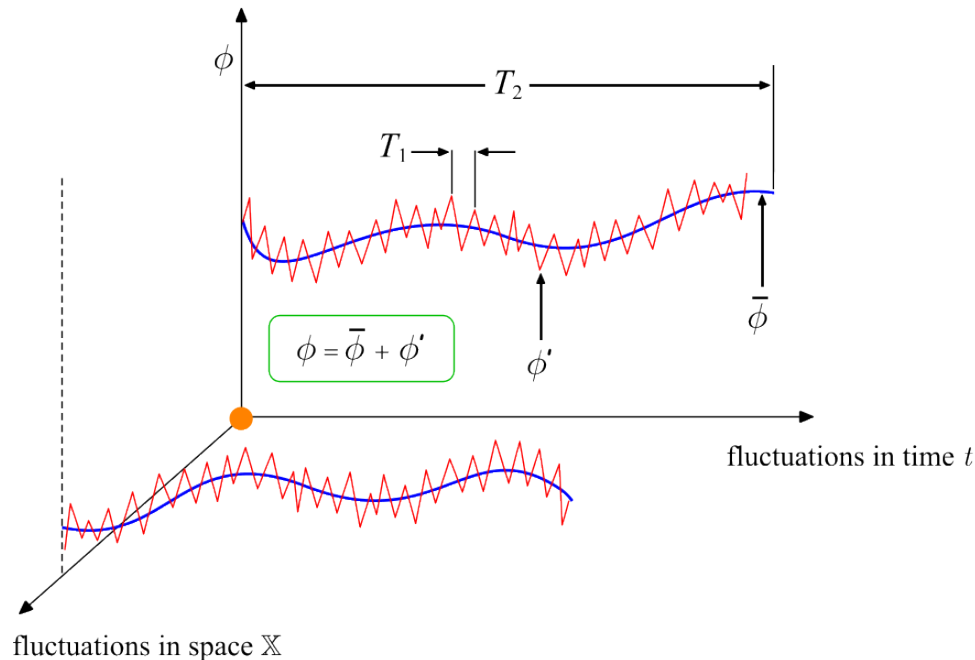


- The Reynold number in this case is 100, for these conditions the flow still is laminar.
- We are in the presence of the Von Karman vortex street, which is the periodic shedding of vortices caused by the unsteady separation of the fluid around blunt bodies.
- Vorticity is not a direct indication of turbulence.
- However turbulent flows are rotational, they exhibit vortical structures.



# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence modeling – Fluctuations of transported quantities



$$\phi(\mathbf{x}, t) = \bar{\phi}(\mathbf{x}, t) + \phi'(\mathbf{x}, t)$$

### In RANS

- The overbar denotes the mean value.
- The prime denotes the fluctuating value.

### In LES

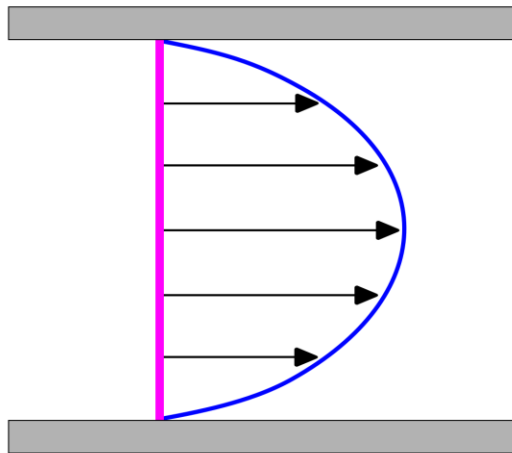
- The overbar denotes the filtered value.
- The prime denotes the modeled value or residual.

$$\bar{\phi}(\mathbf{x}, t) = \frac{1}{T} \int_t^{t+T} \phi(\mathbf{x}, t) dt, \quad T_1 \ll T \ll T_2$$

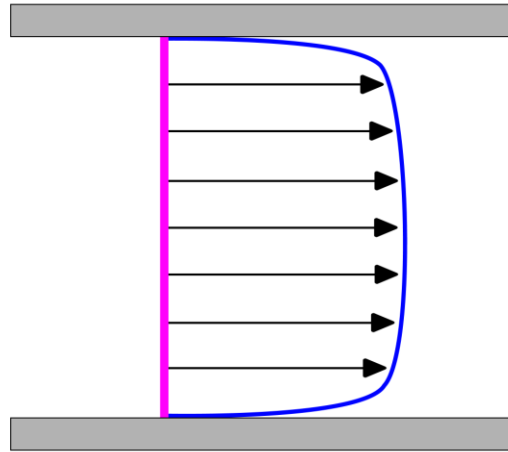
- We have defined turbulence as an unsteady, aperiodic motion in which velocity components and every transported quantity fluctuate in space and time.
- For most engineering application it is impractical to account for all these instantaneous fluctuations.
- Therefore, we need to somehow remove those small scales by using models.
- To remove or filter the instantaneous fluctuations or small scales, two methods can be used: Reynolds averaging and Filtering
- Both methods introduce additional terms that must be modeled for closure.
- We are going to talk about closure methods later.

# A crash introduction to turbulence modeling in OpenFOAM®

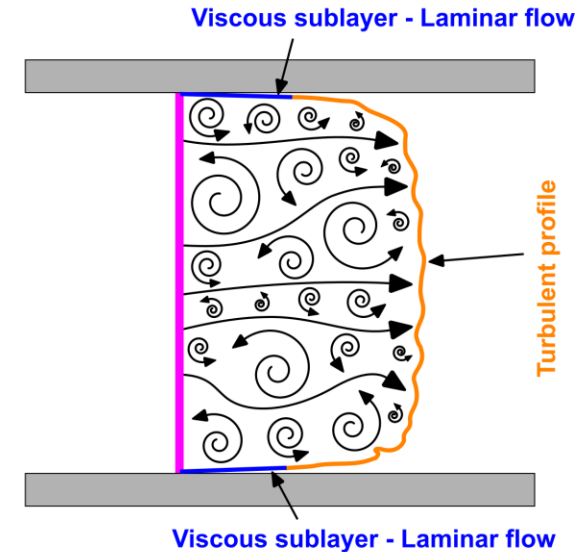
## Turbulence modeling – Velocity profile



Laminar flow profile



Averaged turbulent flow profile

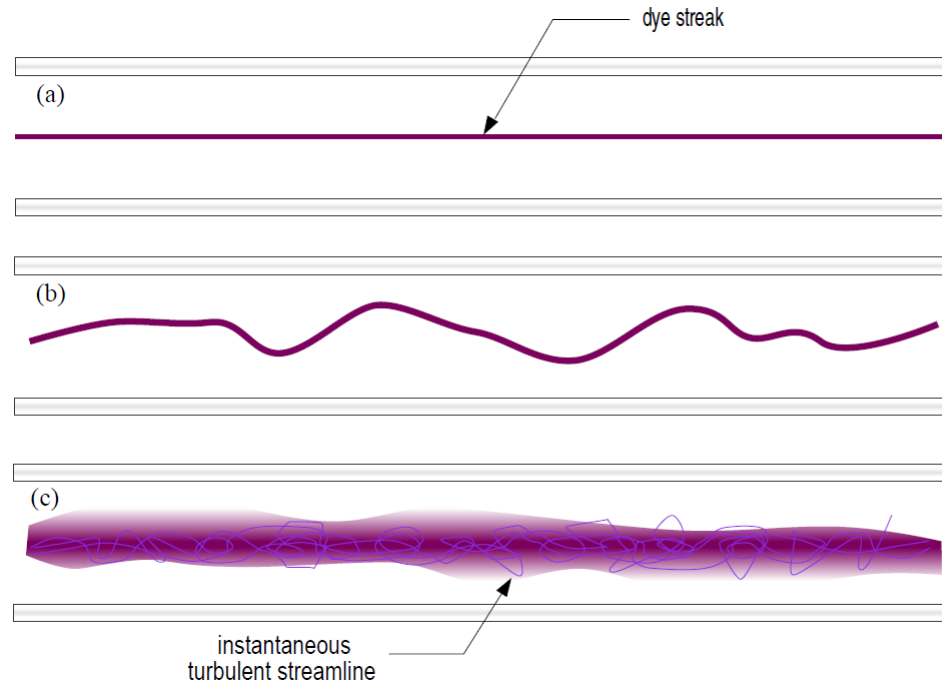


Instantaneous turbulent flow profile

- In the laminar flow case, the velocity gradients close to the walls are low and the velocity profile is parabolic.
- Turbulence has a direct effect on the velocity profiles and mixing of transported quantities.
- The turbulent case shows two regions. One thin region close to the walls with very large velocity gradients, and a region far from the wall where the velocity profile is nearly uniform.
- The thin region close to the walls is laminar.
- Far from the flows, the flow becomes turbulent.

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence modeling – Mixing of transported quantities



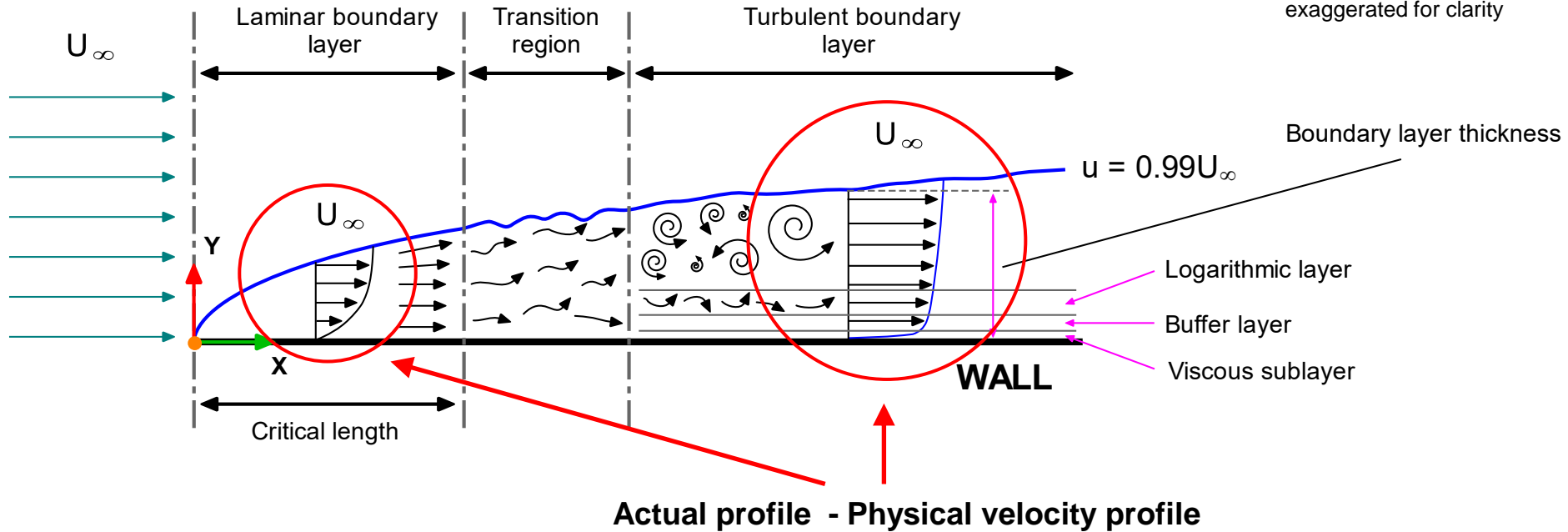
Flow in a pipe. (a) Laminar, (b) Transitional, (c) Turbulent

- Turbulence has a direct effect on the velocity profiles and mixing of transported quantities.
- Case (a) correspond to a laminar flow, where the dye can mix with the main flow only via molecular diffusion, this kind of mixing can take very long times.
- Case (b) shows a transitional state where the dye streak becomes wavy, but the main flow still is laminar.
- Case (c) shows the turbulent state, where the dye streak changes direction erratically, and the dye has mixed significantly with the main flow due to the velocity fluctuations.

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence near the wall – Boundary layer

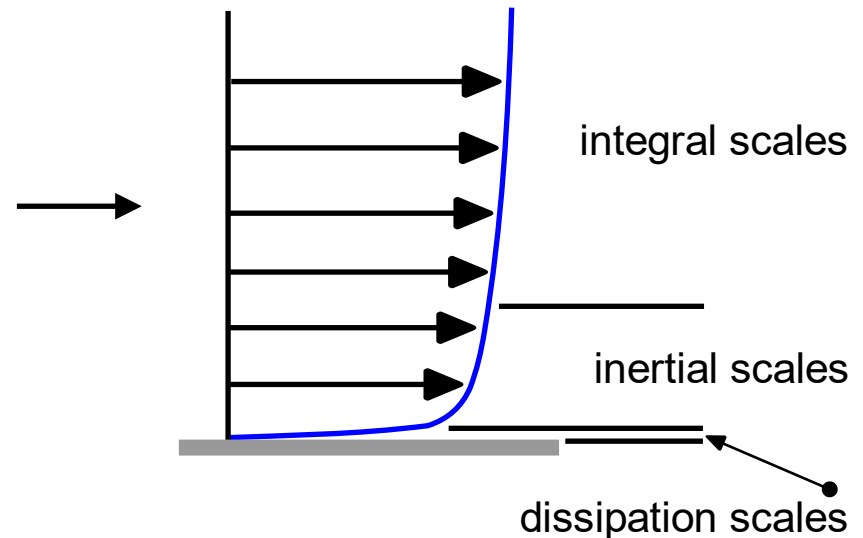
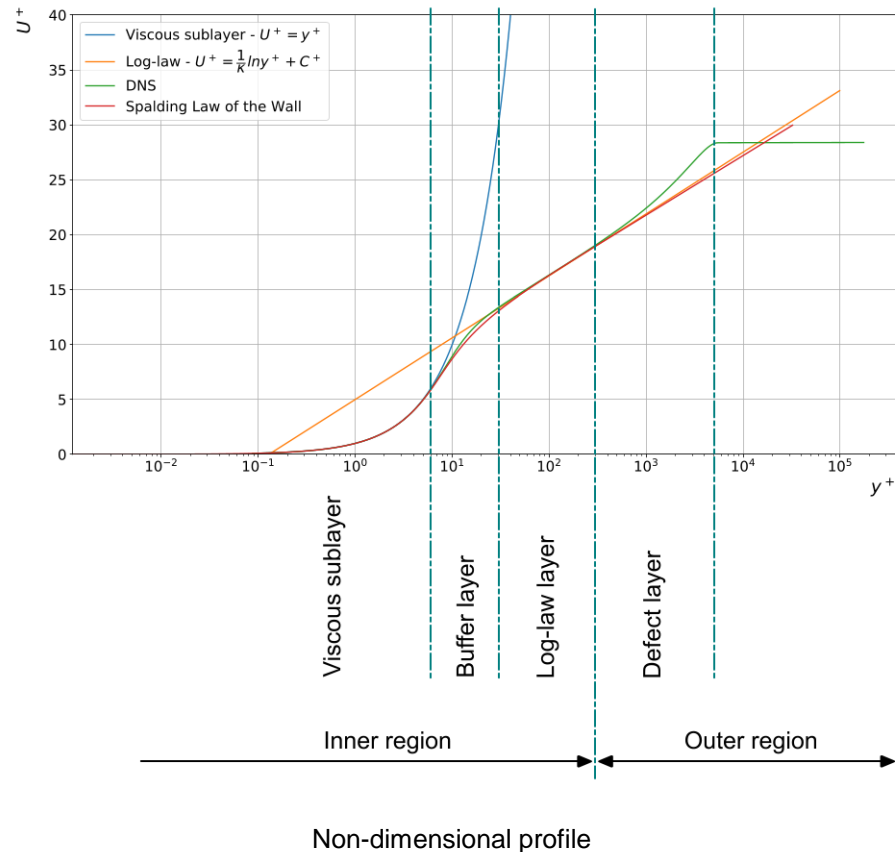
**Note:** The scales are exaggerated for clarity



- Near walls, in the boundary layer, the velocity changes rapidly.
- A laminar boundary layer starts to form at the leading edge. As the flow proceeds further downstream, large shear stresses and velocity gradient develop within the boundary layer. At one point, the flow becomes turbulent.
- In CFD, we try to avoid the transition region and the buffer layer. What is happening in this region is not well understood. The flow can become laminar again or can become turbulent.
- The velocity profiles in the laminar and turbulent regions are different.
- Turbulence models require different considerations depending on whether you solve the viscous sublayer, model the log-law layer, or solve the whole boundary layer.

# A crash introduction to turbulence modeling in OpenFOAM®

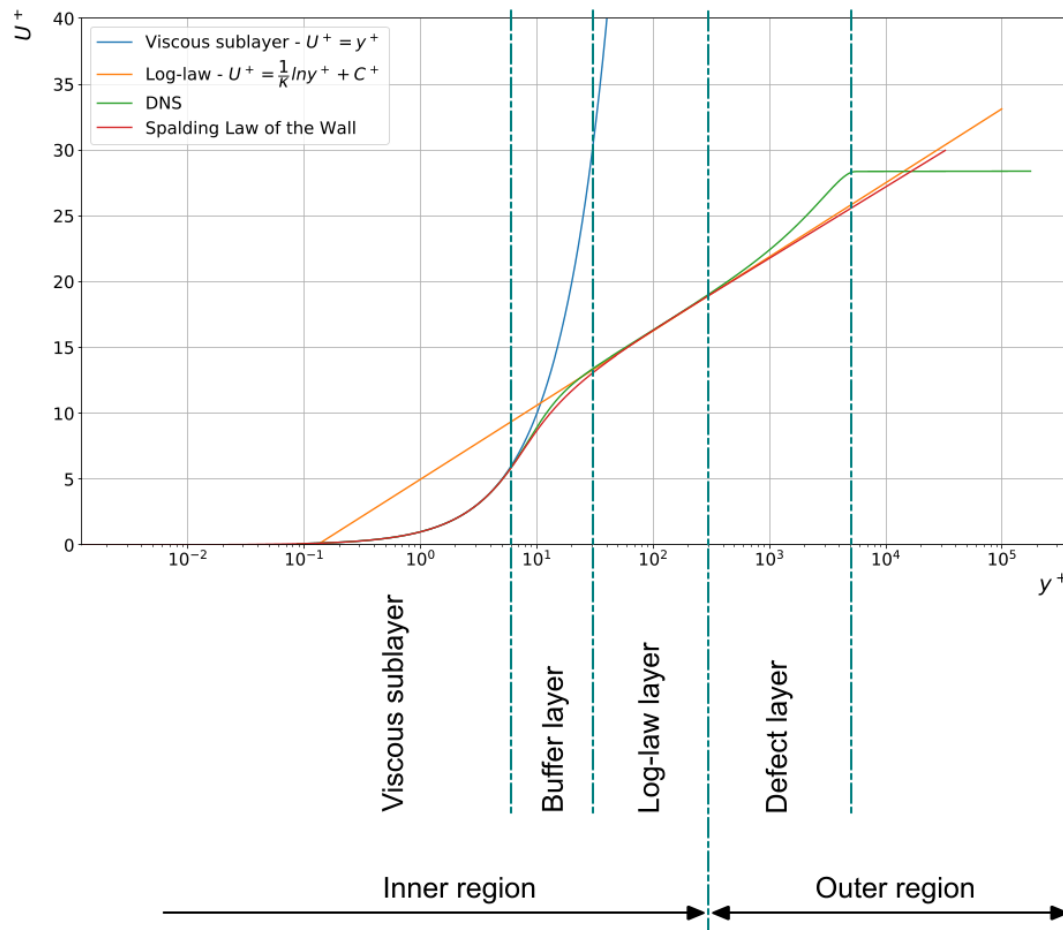
## Turbulence near the wall - Law of the wall



- The use of the non-dimensional velocity  $u^+$  and non-dimensional distance from the wall  $y^+$ , results in a predictable boundary layer profile for a wide range of flows.
- Turbulence models require different considerations depending on whether you solve the viscous sublayer or model the log-law layer.

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence near the wall - Definition of $y^+$ and $u^+$



$$y^+ = \frac{\rho \times U_\tau \times y}{\mu} = \frac{U_\tau \times y}{\nu}$$

$$U_\tau = \sqrt{\frac{\tau_w}{\rho}}$$

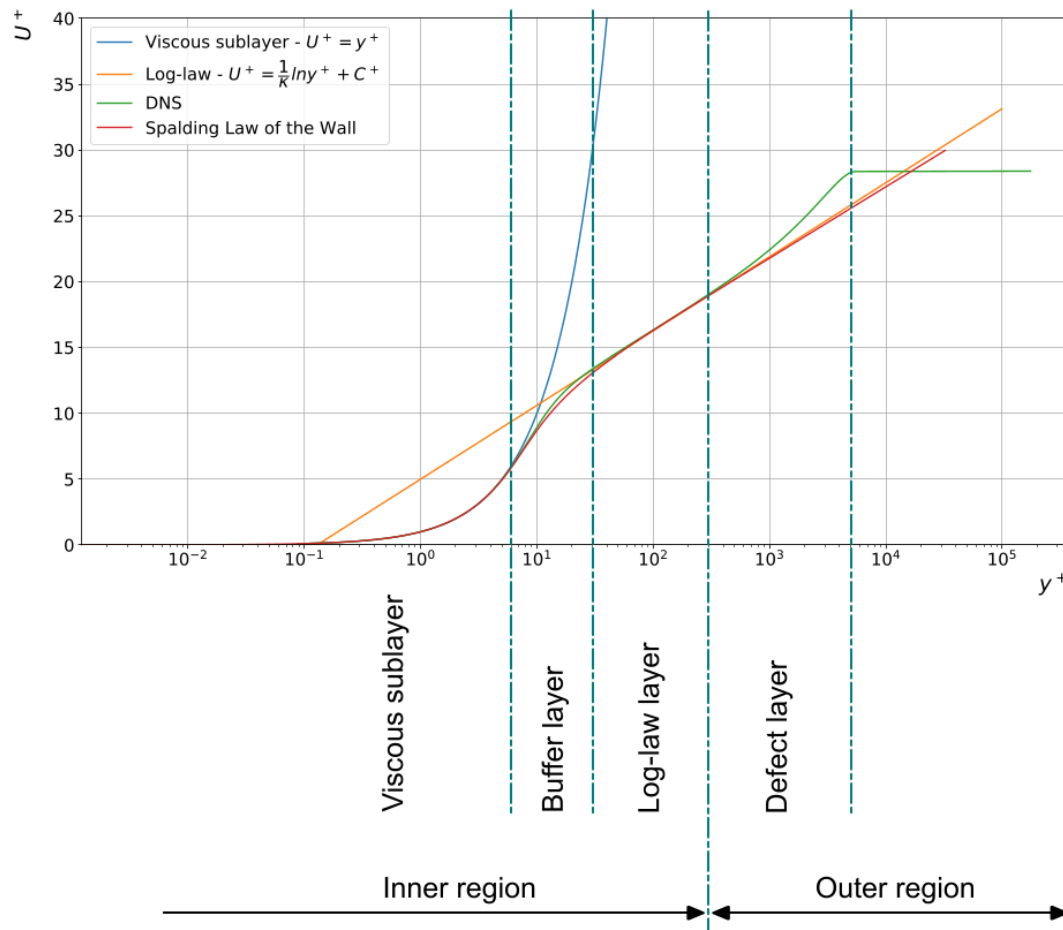
$$u^+ = \frac{U}{U_\tau}$$

Where  $y$  is the distance normal to the wall,  $U_\tau$  is the shear velocity, and  $u^+$  relates the mean velocity to the shear velocity

- $y^+$  or wall distance units is a very important concept when dealing with turbulence modeling.
- Remember this definition as we are going to use it a lot.

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence near the wall - Relations according to $y^+$ value



Logarithmic layer (log-law)

$$30 < y^+ < 300$$

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

$$\kappa \approx 0.41 \quad C^+ \approx 5.0$$

Viscous sublayer

$$y^+ < 6$$

$$u^+ = y^+$$

Buffer layer

$$6 < y^+ < 30$$

$$u^+ \neq y^+$$

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

**Note 1:** the range of  $y^+$  values might change from reference to reference but roughly speaking they are all close to these values.

**Note 2:** the  $y^+$  upper limit of the buffer layer depends on the Reynolds number. Large Re will have higher  $y^+$  upper limit

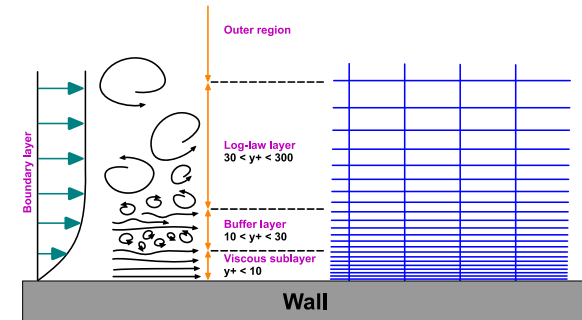
# A crash introduction to turbulence modeling in OpenFOAM®

## Near-wall treatment and wall functions

- When dealing with wall turbulence, we need to choose a near-wall treatment.

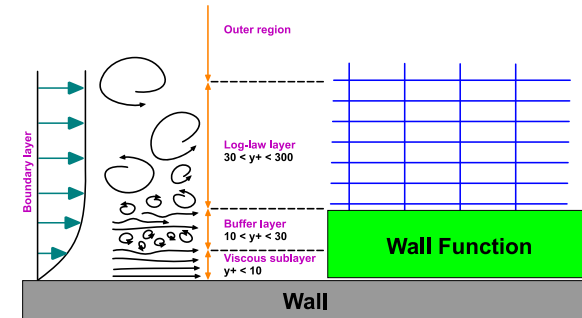
### Wall resolving approach

- If you want to resolve the boundary layer up to the viscous sub-layer you need very fine meshes close to the wall.
- In terms of  $y^+$ , you need to cluster at least 8-10 layers at  $y^+ < 6-10$ .
- But for good accuracy, usually you will use 15 to 30 layers, with a low growth rate.
- You need to properly resolve the velocity profile.
- This is the most accurate approach, but it is computationally expensive.



### Wall modeling approach

- If you are not interested in resolving boundary layer up to the viscous sub-layer, you can use wall functions.
- In terms of  $y^+$ , wall functions will model everything below  $y^+ < 30$  or your lower  $y^+$  limit.
- You will need to cluster at least 5 to 10 layers to resolve the profiles ( $U$  and  $k$ ).
- This approach uses coarser meshes, but you should be aware of the limitations of the wall functions.



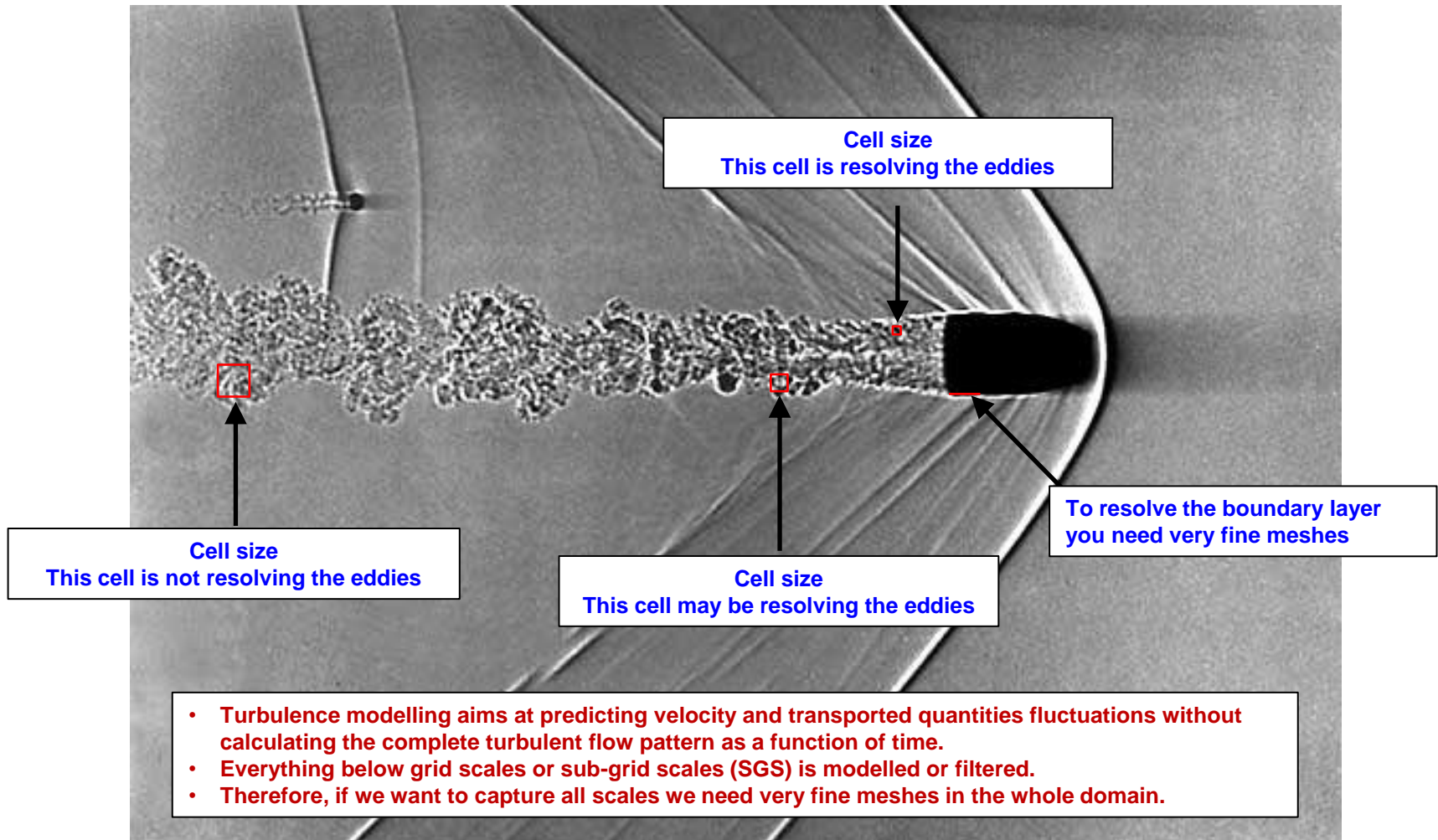
### $y^+$ insensitive

- You can also use the  $y^+$  insensitive wall treatment (sometimes known as continuous wall functions or scalable wall functions).
- This near-wall treatment is valid in the whole boundary layer.
- In terms of  $y^+$ , you can use this approach for values between  $1 < y^+ < 300$ .
- This approach is very flexible as it is independent of the  $y^+$  value but is not available in all turbulence models.
- You should also be aware of its limitations.



# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence modeling – Grid scales



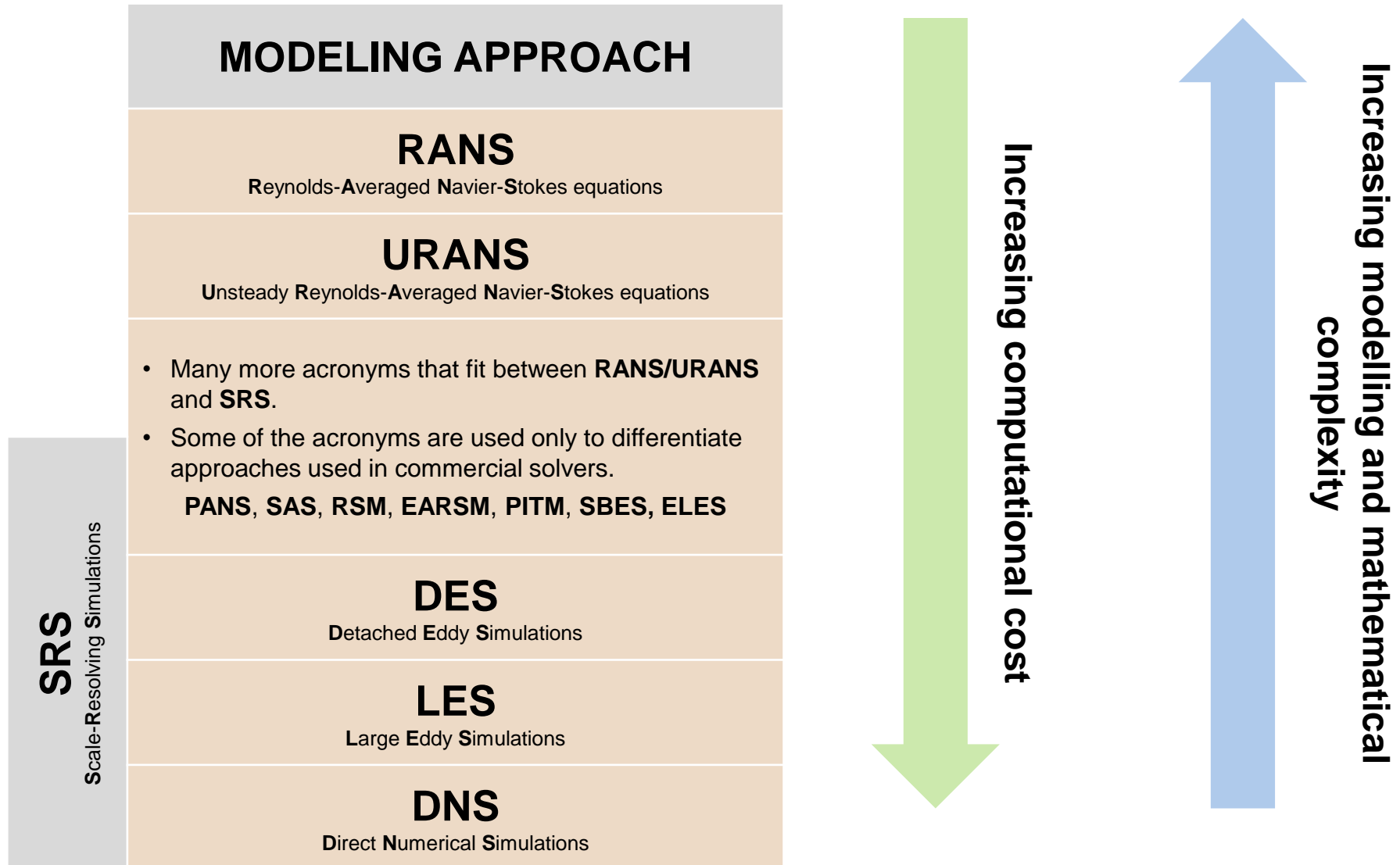
**Bullet at Mach 1.5**

Photo credit: Andrew Davidhazy. Rochester Institute of Technology.

Copyright on the images is held by the contributors. Apart from Fair Use, permission must be sought for any other purpose.

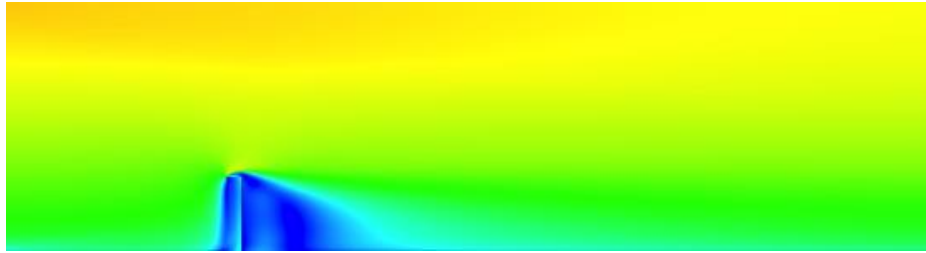
# A crash introduction to turbulence modeling in OpenFOAM®

## Overview of turbulence modeling approaches

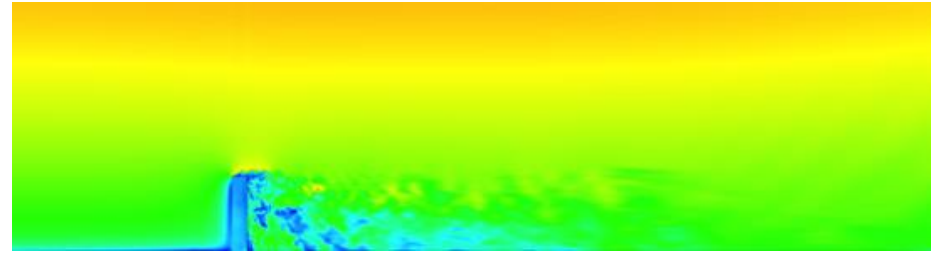


# A crash introduction to turbulence modeling in OpenFOAM®

## Overview of turbulence modeling approaches



**RANS**



**LES (Instantaneous field)**

RANS/URANS	DES/LES	DNS
<ul style="list-style-type: none"><li>• Solve the time-average NSE.</li><li>• All turbulent spatial scales are modeled.</li><li>• Many models are available. One equation models, two equation models, Reynolds stress models, transition models, and so on.</li><li>• This is the most widely approach for industrial flows.</li><li>• Unsteady RANS (URANS), use the same equations as the RANS but with the transient term retained.</li><li>• It can be used in 2D and 3D cases.</li></ul>	<ul style="list-style-type: none"><li>• Solve the filtered unsteady NSE.</li><li>• Sub-grid scales (SGS) are filtered, grid scales (GS) are resolved.</li><li>• Aim at resolving the temporal scales, hence requires small time-steps.</li><li>• For most industrial applications, it is computational expensive. However, thanks to the current advances in parallel and scientific computing it is becoming affordable.</li><li>• Many models are available.</li><li>• It is intrinsically 3D and asymmetric.</li></ul>	<ul style="list-style-type: none"><li>• Solves the unsteady laminar NSE.</li><li>• Solves all spatial and temporal scales; hence, requires extremely fine meshes and small time-steps.</li><li>• No modeling is required.</li><li>• It is extremely computational expensive.</li><li>• Not practical for industrial flows.</li><li>• It is intrinsically 3D and asymmetric.</li></ul>

# A crash introduction to turbulence modeling in OpenFOAM®

## Short description of some RANS turbulence models

Model	Short description
<b>Spalart-Allmaras</b>	Suitable for external aerodynamics, tubomachinery and high-speed flows. Good for mildly complex external/internal flows and boundary layer flows under pressure gradient (e.g. airfoils, wings, airplane fuselages, ship hulls). Performs poorly for free shear flows and flows with strong separation.
<b>Standard k-epsilon</b>	Robust. Widely used despite the known limitations of the model. Performs poorly for complex flows involving severe pressure gradient, separation, strong streamline curvature. Suitable for initial iterations, initial screening of alternative designs, and parametric studies.
<b>Realizable k-epsilon</b>	Suitable for complex shear flows involving rapid strain, moderate swirl, vortices, and locally transitional flows (e.g., boundary layer separation, massive separation, and vortex shedding behind bluff bodies, stall in wide-angle diffusers, room ventilation). It overcome the limitations of the standard k-epsilon model.
<b>Standard k-omega</b>	Superior performance for wall-bounded boundary layer, free shear, and low Reynolds number flows compared to models from the k-epsilon family. Suitable for complex boundary layer flows under adverse pressure gradient and separation (external aerodynamics and turbomachinery).
<b>SST k-omega</b>	Offers similar benefits as standard k-omega. Not overly sensitive to inlet boundary conditions like the standard k-omega. Provides more accurate prediction of flow separation than other RANS models. Probably the most widely used RANS model.

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence modeling – Starting equations

Exact NSE

$$\left\{ \begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \mathbf{S}_u \\ \frac{\partial (\rho e_t)}{\partial t} + \nabla \cdot (\rho e_t \mathbf{u}) &= -\nabla \cdot q - \nabla \cdot (p \mathbf{u}) + \boldsymbol{\tau} : \nabla \mathbf{u} + \mathbf{S}_{e_t} \end{aligned} \right.$$

Source terms  
↓

+

Additional equations to close the system (thermodynamic variables)

Additionally, relationships to relate the transport properties

**Additional closure equations for the turbulence models**

- Turbulence models equations cannot be derived from fundamental principles.
- All turbulence models contain some sort of empiricism.
- Some calibration to observed physical solutions is contained in the turbulence models.
- Also, some intelligent guessing is used.
- A lot of uncertainty is involved!

# A crash introduction to turbulence modeling in OpenFOAM®

## Incompressible RANS equations

- The RANS equations are very similar to the starting equations.

$$\nabla \cdot (\bar{\mathbf{u}}) = 0$$
$$\frac{\partial \bar{\mathbf{u}}}{\partial t} + \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) = -\frac{1}{\rho} (\nabla p) + \nu \nabla^2 \bar{\mathbf{u}} + \frac{1}{\rho} \nabla \cdot \boldsymbol{\tau}^R$$

↑

If we retain this term, we talk about URANS equations and if we drop it we talk about RANS equations

RANS/URANS equations

$$\nabla \cdot (\mathbf{u}) = 0$$
$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\frac{\nabla p}{\rho} + \nu \nabla^2 \mathbf{u}$$

NSE with no turbulence models (DNS)

- The differences are that all quantities have been averaged (the overbar over the primitive variables).
- And the appearance of the Reynolds stress tensor  $\boldsymbol{\tau}^R$ .
- Notice that the Reynolds stress tensor is not actually a stress, it must be multiplied by density in order to have dimensions corresponding to stresses,

$$\boldsymbol{\tau}^R = -\rho (\overline{\mathbf{u}'\mathbf{u}'})$$

- To derive the RANS equations we used Reynolds decomposition and a few averaging rules (a lot of algebra is involved),

$$\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}) + \mathbf{u}'(\mathbf{x}, t), \quad p(\mathbf{x}, t) = \bar{p}(\mathbf{x}) + p'(\mathbf{x}, t)$$

← Reynolds decomposition

## Incompressible RANS equations

- The RANS approach to turbulence modeling requires the Reynolds stresses to be appropriately modeled.

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

- The Reynolds stress tensor can be modeled using the Boussinesq hypothesis, Reynolds stress models, non-linear eddy viscosity models or algebraic models.
- Let us address the Boussinesq hypothesis which is the most widely used approach to model the Reynolds stress tensor.
- By using this hypothesis, we can relate the Reynolds stress tensor to the mean velocity gradient such that,

$$\tau^R = -\rho (\overline{\mathbf{u}'\mathbf{u}'} ) = 2\mu_T \bar{\mathbf{D}}^R - \frac{2}{3}\rho\kappa\mathbf{I} = \mu_T \left[ \nabla \bar{\mathbf{u}} + (\nabla \bar{\mathbf{u}})^T \right] - \frac{2}{3}\rho\kappa\mathbf{I},$$

- In the previous equation,  $\bar{\mathbf{D}}^R = \frac{1}{2}(\nabla \bar{\mathbf{u}} + \nabla \bar{\mathbf{u}}^T)$  denotes the strain-rate tensor.
- $\mathbf{I}$  is the identity matrix.
- $\mu_T$  is the turbulent eddy viscosity.
- $\kappa = \frac{1}{2}\overline{\mathbf{u}' \cdot \mathbf{u}'}$  is the turbulent eddy viscosity.
- At the end of the day we want to determine the turbulent eddy viscosity.
- The turbulent eddy viscosity is not a fluid property, it is a property needed by the turbulence model.
- Each turbulence model will compute the turbulent eddy viscosity in a different way.

# A crash introduction to turbulence modeling in OpenFOAM®

## Incompressible RANS equations

- After introducing the Boussinesq approximation and doing some algebra, we obtain the following form of the governing equations,

$$\nabla \cdot (\bar{\mathbf{u}}) = 0$$
$$\frac{\partial \bar{\mathbf{u}}}{\partial t} + \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) = -\frac{1}{\rho} \left( \nabla \bar{p} + \frac{2}{3} \rho \nabla k \right) + \nabla \cdot \left[ \frac{1}{\rho} (\mu + \mu_t) \nabla \bar{\mathbf{u}} \right]$$

Effective viscosity

Turbulent viscosity

Normal stresses arising from the Boussinesq approximation

- Notice that by introducing the Boussinesq approximation the fluctuating quantities (the prime in the equations) do not appear in the final equations.
- The new equations are expressed entirely in terms of mean values (overbar), which can be computed.
- The problem now reduces to computing the turbulent eddy viscosity  $\mu_t$  in the momentum equation.
- This is done by adding closure models (one-equation, two-equations, algebraic, transition, Reynolds stress, and so on).



# A crash introduction to turbulence modeling in OpenFOAM®

## Additional remarks

- We just outlined the incompressible RANS.
- The compressible RANS equations are similar, but when we derive them, we use Favre average (which can be seen as a mass-weighted averaging), instead of Reynolds average.
- Besides RANS, there is also LES and DES turbulence models.
- The idea behind LES/DES models is very similar to RANS, but instead of using averaging we filter the equations in space, and we solve the temporal scales
- At the end of the day, in LES/DES it is also required to model a stress tensor, usually called the SGS stress tensor.
- This stress tensor is related to the scales that cannot be resolved with the mesh; therefore, need to be modelled.
- LES/DES models are intrinsically unsteady and three-dimensional.
- Let us take a look at the governing equations of the  $k - \omega$  RANS model (Wilcox 1998 revision).
- Remember, the main goal of the RANS turbulence models is to model the Reynolds stress tensor by computing the turbulent eddy viscosity.

# A crash introduction to turbulence modeling in OpenFOAM®

## $\mathcal{K} - \omega$ Turbulence model overview (Wilcox 1998 revision)

- It is called  $\mathcal{K} - \omega$  because it solves two additional equations for modeling the turbulent eddy viscosity, namely, the turbulent kinetic energy  $\mathcal{K}$  and the specific dissipation rate  $\omega$ .

$$\begin{aligned}\nabla_t \rho k + \nabla \cdot (\rho \bar{\mathbf{u}} k) &= \overbrace{\tau^R : \nabla \bar{\mathbf{u}}}^{\text{Production}} - \overbrace{\beta^* \rho \omega k}^{\text{Dissipation}} + \overbrace{\nabla \cdot [(\mu + \sigma_k \mu_t) \nabla k]}^{\text{Diffusion}} \\ \nabla_t \rho \omega + \nabla \cdot (\rho \bar{\mathbf{u}} \omega) &= \underbrace{\alpha \frac{\omega}{k} \tau^R : \nabla \bar{\mathbf{u}}}_{\text{Production}} - \underbrace{\beta \rho \omega^2}_{\text{Dissipation}} + \underbrace{\nabla \cdot [(\mu + \sigma_\omega \mu_t) \nabla \omega]}_{\text{Diffusion}}\end{aligned}$$

- These are the closure equations of the turbulence problem using the  $\mathcal{K} - \omega$  RANS model.
- These are not physical quantities. They kind of represent the generation and destruction of turbulence.
- In the  $\mathcal{K} - \omega$  model, the turbulent eddy viscosity can be computed as follows,

$$\mu_T = \frac{\rho k}{\omega}$$

- The model has many closure coefficient that we do not show here. These coefficients are calibrated using experimental data, DNS simulations, analytical solutions, or empirical data.
- Note that all quantities are computed in function of mean values. The Reynolds stresses are modeled using the Boussinesq hypothesis. By proceeding in this way, we remove any dependence on the fluctuations.
- It is worth mentioning that different turbulence models will have different ways of computing the turbulent eddy viscosity.

# A crash introduction to turbulence modeling in OpenFOAM®

## $\kappa - \omega$ Turbulence model free stream initial conditions

- The freestream and initial value for the turbulent kinetic energy  $\kappa$  can be computed as follows,

$$\kappa = \frac{3}{2}(UI)^2$$

- The freestream and initial value for the specific dissipation rate  $\omega$  can be computed as follows,

$$\omega = \frac{\rho \kappa}{\mu} \frac{\mu_t}{\mu}^{-1}$$

- Where  $\mu_t/\mu$  is the viscosity ratio and  $I = u'/\bar{u}$  is the turbulence intensity.
- If you are totally lost, you can use these reference values. They work most of the times, but it is a good idea to have some experimental data or a better initial estimate.

	Low	Medium	High
$I$	1.0 %	5.0 %	10.0 %
$\mu_t/\mu$	1	10	100

- By the way, use these guidelines for external aerodynamics only.

# A crash introduction to turbulence modeling in OpenFOAM®

## $\kappa - \omega$ Turbulence model free stream initial conditions

- In the literature, you will find many relations to estimate the freestream values and initial conditions.
- It does not matter what relation you use; at the end of the day, they are all similar.
- However, you should assign physically realistic values.
- Remember, different turbulence models will have different relations to compute the freestream values and initial conditions.
- At the following link,
  - <https://turbmodels.larc.nasa.gov/>
- You will find a description of many turbulence models.
- Among the information given, you will find the recommended boundary conditions and initial conditions for several turbulence models.

## $\kappa - \omega$ Turbulence model boundary conditions at the walls

- Use the following guidelines to find the boundary conditions for the near-wall treatment.
- We highly recommend you read the source code and find the references used to implement the model.
- As for the free-stream boundary conditions, you need to give the boundary conditions for the near-wall treatment.
- When it comes to near-wall treatment, you have three options:

- Use wall functions:

$$30 \leq y^+ \leq 300$$

- $y^+$  insensitive wall functions, this only applies to the  $\kappa - \omega$  family of model:

$$1 \leq y^+ \leq 300$$

- Resolve the boundary layer (no wall functions):

$$y^+ \leq 6$$

# A crash introduction to turbulence modeling in OpenFOAM®

## $k - \omega$ Turbulence model boundary conditions at the walls

- Remember, you can only use wall functions if the primitive patch (the patch type defined in the *boundary* dictionary), is of type **wall**.



Field	Wall functions – High RE	Resolved BL – Low RE
nut	<b>nut(-)WallFunction*</b> or <b>nutUSpaldingWallFunction**</b> (with 0 or a small number)	<b>nutUSpaldingWallFunction**</b> or <b>nutLowReWallFunction</b> or <b>fixedValue</b> (with 0 or a small number)
k, q, R	<b>kqRWallFunction</b> $k_{wall} = k$ or $k_{wall} = 1e - 10$	<b>kqRWallFunction</b> or <b>kLowReWallFunction</b> $k_{wall} = k$ or $k_{wall} = 1e - 10$
epsilon	<b>epsilonWallFunction</b> (with inlet value) $\epsilon_{wall} = \epsilon$	<b>epsilonWallFunction</b> (with inlet value) or <b>zeroGradient</b> or <b>fixedValue</b> (with 0 or a small number)
omega	<b>omegaWallFunction</b> $\omega_{wall} = 10 \frac{6\nu}{\beta y^2}$ $\beta = 0.075$	<b>omegaWallFunction**</b> or <b>fixedValue</b> $\omega_{wall} = 10 \frac{6\nu}{\beta y^2}$ $\beta = 0.075$
nuTilda	–	<b>fixedValue</b> (with 0 or a small number)

\* \$WM\_PROJECT\_DIR/src/TurbulenceModels/turbulenceModels/derivedFvPatchFields/wallFunctions/nutWallFunctions

\*\* For  $y^+$  insensitive wall functions (continuous wall functions)

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence models available in OpenFOAM®

- There are many turbulence models implemented in OpenFOAM®, from RANS to LES.
- You can also implement yours!
- You can find the turbulence models in the following directories:
  - `$WM_PROJECT_DIR/src/MomentumTransferModels`
- The wall functions are in the following directories:
  - `$WM_PROJECT_DIR/src/MomentumTransferModels/momentumTransferModels/derivedFvPatchFields`
- If you have absolutely no idea of what model to use, we highly recommend you the k-omega family models or the realizable k-epsilon model.
- Remember, when a turbulent flow enters a domain, turbulent boundary conditions and initial conditions must be specified.
- Also, if you are dealing with wall bounded turbulence you will need to choose the near-wall treatment.
- You can choose to solve the viscous sub-layer (low-Re approach) or use wall functions (high-Re approach).
- You will need to give the appropriate boundary conditions to the near-wall treatment.
- **Our task is to choose the less wrong model !**

# A crash introduction to turbulence modeling in OpenFOAM®

## $y^+$ wall distance units normal to the wall

- We never know a priori the  $y^+$  value (because we do not know the friction velocity).
- What we usually do is to run the simulation for a few time-steps or iterations, and then we get an estimate of the  $y^+$  value.
- After determining where we are in the boundary layer (viscous sub-layer, buffer layer or log-law layer), we take the mesh as a good one or we modify it if is deemed necessary.
- It is an iterative process, and it can be very time consuming, as it might require remeshing and rerunning the simulation.
- Have in mind that it is quite difficult to get a uniform  $y^+$  value at the walls.
- Try to get a  $y^+$  mean value as close as possible to your target.
- Also, check that you do not get very high maximum values of  $y^+$  (more than a 1000)
- Values up to 300 are fine. Values larger than 300 and up to a 1000 are acceptable if they do not cover a large surface (no more than 10% of the total wall area), or they are not located in critical zones.
- Remember, the upper limit of  $y^+$  also depends on the Reynolds number.
- Use common sense when accessing  $y^+$  value.



# A crash introduction to turbulence modeling in OpenFOAM®

## Estimating normal wall distance

- To get an initial estimate of the distance from the wall to the first cell center  $y^+$ , without recurring to a precursor simulation, you can proceed as follows,

1. 
$$Re = \frac{\rho \times U \times L}{\mu}$$

2. 
$$C_f = 0.058 \times Re^{-0.2} \longrightarrow$$
 (Skin friction coefficient of a flat plate, there are similar correlations for pipes)

3. 
$$\tau_w = \frac{1}{2} \times C_f \times \rho \times U_\infty^2$$

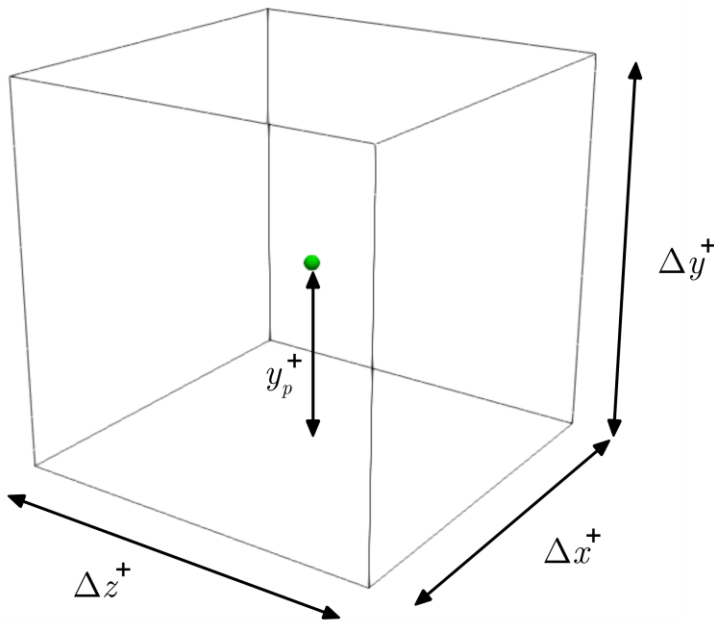
4. 
$$U_\tau = \sqrt{\frac{\tau_w}{\rho}}$$

5. 
$$y = \frac{\mu \times y^+}{\rho \times U_\tau} \longleftarrow \text{Your desired value}$$

- You will find a simple calculator for the wall distance estimation in the following link:  
<http://www.wolfdynamics.com/tools.html?id=2>

# A crash introduction to turbulence modeling in OpenFOAM®

## Wall distance units $x^+ - y^+ - z^+$



- Similar to  $y^+$ , the wall distance units can be computed in the stream-wise ( $\Delta x^+$ ) and span-wise ( $\Delta z^+$ ) directions.
- The wall distance units in the stream-wise and span-wise directions can be computed as follows:

$$\Delta x^+ = \frac{U_\tau \Delta x}{\nu} \qquad \Delta z^+ = \frac{U_\tau \Delta z}{\nu}$$

- And recall that  $y^+$  is computed at the cell center, therefore:

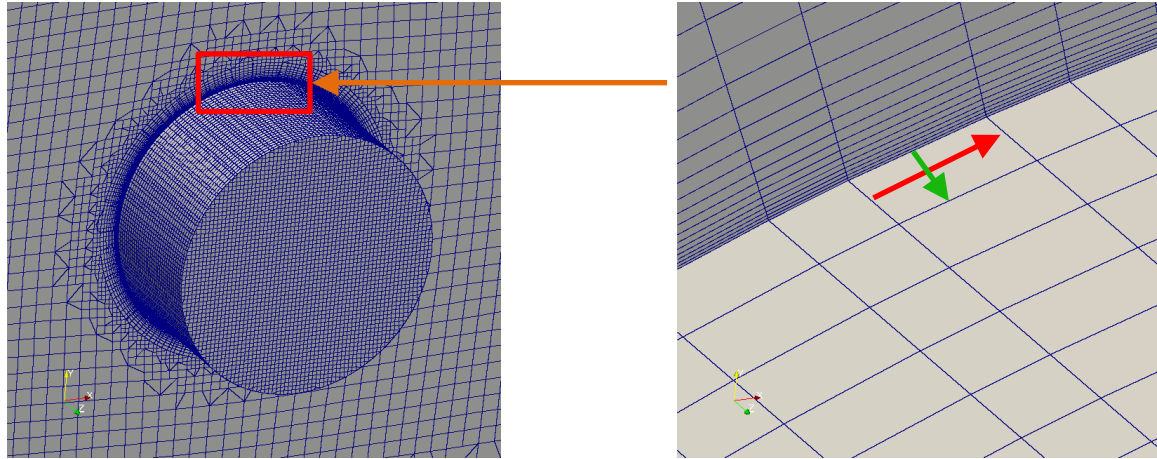
$$\Delta y^+ = 2 \times y^+$$

$$(\Delta x^+, \Delta y^+, \Delta z^+) = \left( \frac{x}{l_\tau}, \frac{y}{l_\tau}, \frac{z}{l_\tau} \right)$$

where  $l_\tau = \frac{\nu}{U_\tau}$   
Viscous length

# A crash introduction to turbulence modeling in OpenFOAM®

## Wall distance units and some rough estimates



- Similar to  $y^+$ , the wall distance units can be computed in the stream-wise ( $\Delta x^+$ ) and span-wise ( $\Delta z^+$ ) directions.
- DES and RANS simulations do not have stream-wise and span-wise wall distance units requirements as in LES simulations. Therefore, they are more affordable.
- Typical requirements for LES are (these are approximations based on different references):

$$\Delta x^+ < 50, \Delta z^+ < 50 \quad \text{for} \quad y^+ < 6$$

Wall resolving

$$\Delta x^+ < 4\Delta y^+, \Delta z^+ < 4\Delta y^+ \quad \text{for} \quad 30 \leq y^+ \leq 300$$

Wall modeling

# A crash introduction to turbulence modeling in OpenFOAM®

## Turbulence modeling guidelines and tips

- Compute Reynolds number and determine whether the flow is turbulent.
- Try to avoid the use of turbulent models with laminar flows.
- Choose the near-wall treatment and estimate  $y$  before generating the mesh.
- Run the simulation for a few time steps and get a better prediction of  $y$  and correct your initial prediction of  $y^+$ .
- The realizable  $\kappa - \epsilon$  or SST  $\kappa - \omega$  models are good choices for general applications.
- The standard  $\kappa - \epsilon$  model is very reliable, you can use it to get initial values. Have in mind that this model use wall functions.
- If you are interested in resolving the large eddies and modeling the smallest eddies, DES or LES are the right choice.
- If you do not have any restriction in the near wall treatment method, use wall functions (even with LES/DES models).
- Be aware of the limitations of the turbulence model chosen, find and read the original references used to implement the model in OpenFOAM®.
- Set reasonable boundary and initial conditions for the turbulence model variables.

## Turbulence modeling guidelines and tips

- Always monitor the turbulent variables, some of them are positive bounded.
- Avoid strong oscillations of the turbulent variables.
- If you are doing LES, remember that these models are intrinsically 3D and unsteady. You should choose your time-step in such a way to get a CFL of less than 1 and preferably of about 0.5.
- If you are doing RANS with wall functions, it is perfectly fine to use upwind to discretize the turbulence closure equations. After all, turbulence is a dissipative process. However, some authors may disagree with this, make your own conclusions.
- On the other hand, if you are using a wall resolved approach, it is better to use a high-order discretization scheme to discretize the turbulence closure equations.
- If you are doing unsteady simulations, always remember to compute the average values (ensemble average).
- If you are dealing with external aerodynamics and detached flows, DES simulations are really affordable.
- The work-horse of turbulence modeling in CFD, **RANS**

# Turbulence modeling hands-on tutorials

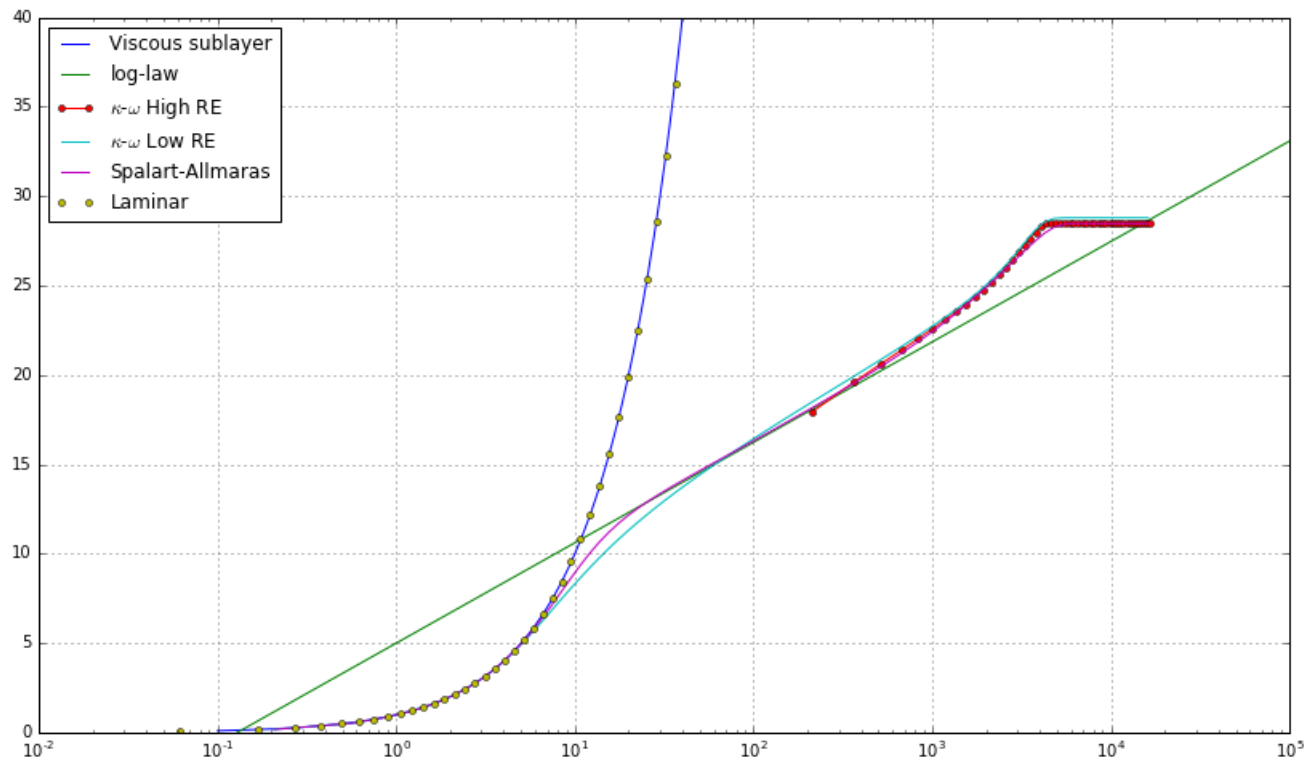
- Laminar-Turbulent flat plate
- Let us run this case. Go to the directory:

```
$PTOFC/advanced_physics/turbulence/flatPlate
```

- 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.

# Turbulence modeling hands-on tutorials

## Laminar-Turbulent flat plate



- The best way to understand the near the wall treatment and the effect of turbulence near the walls, is by reproducing the law of the wall.
- By plotting the velocity in terms of the non-dimensional variables  $u^+$  and  $y^+$ , we can compare the profiles obtained from the simulations with the theoretical profiles.

# Turbulence modeling hands-on tutorials

## Laminar-Turbulent flat plate

- In the directory **python** of each case, you will find a jupyter notebook (a python script), that you can use to plot the non-dimensional  $u^+$  and  $y^+$  profiles.
- The notebook uses some precomputed results, but you can adjust it to any case.
- Remember, the  $u^+$  vs.  $y^+$  plot is kind of a universal plot.
- It does not matter your geometry or flow conditions, if you are resolving well the turbulent flow, you should be able to recover this profile.
- To compute this plot, you must sample the wall shear stresses and the velocity along a line normal to the wall.
- Then, you can compute the shear velocity, friction coefficient, and  $u^+$  and  $y^+$  values.

$$y^+ = \frac{\rho \times U_\tau \times y}{\mu} = \frac{U_\tau \times y}{\nu}$$

$$U^+ = \frac{U}{U_\tau}$$

$$U_\tau = \sqrt{\frac{\tau_w}{\rho}}$$

$$c_f = \frac{\tau_w}{0.5\rho U_\infty^2}$$



# Turbulence modeling hands-on tutorials

## Laminar-Turbulent flat plate

- We are going to use the following solver: `simpleFoam` (for RANS).
- This case is rather simple, but we will use it to explain many features used in OpenFOAM® when dealing with turbulence, especially when dealing with near the wall treatment.
- We will also show you how to do the post-processing in order to reproduce the law of the wall. For this, we will use a jupyter notebook (a python script).
- Remember, as we are introducing new closure equations for the turbulence problem, we need to define initial and boundary conditions for the new variables.
- We also need to define the discretization schemes and linear solvers to use to solve the new variables.
- It is also a good idea to setup a few functionObjects, such as:  $y^+$ , minimum and maximum values, forces, time average, and online sampling.

# Turbulence modeling hands-on tutorials

## Laminar-Turbulent flat plate

- We select the turbulence model in the *momentumTransport* dictionary file.
- This dictionary file is located in the directory **constant**.
- To select the K-Omega SST turbulence model,

```
17  simulationType  RAS; ← RANS type simulation
18
19  RAS ← RANS sub-dictionary
20  {
21      RASModel      kOmegaSST; ← RANS model to use
22
23      turbulence     on; ← Turn on/off turbulence. Runtime modifiable
24
25      printCoeffs    on; ← Print coefficients at the beginning
26  }
```

- Remember, you need to assign boundary and initial conditions to the new variables (**k**, **omega**, and **nut**).

# Turbulence modeling hands-on tutorials

## Vortex shedding past square cylinder

- To define the wall functions, follow this table,

Field	Wall functions – High RE	Resolved BL – Low RE
<b>nut</b>	<b>nutUSpaldingWallFunction</b>	<b>fixedValue 0</b> or a small number
<b>k</b>	<b>kqRWallFunction</b>	<b>fixedValue 0</b> or a small number
<b>omega</b>	<b>omegaWallFunction</b> $\omega_{wall} = 10 \frac{6\nu}{\beta y^2} \quad \beta = 0.075$	<b>omegaWallFunction</b> $\omega_{wall} = 10 \frac{6\nu}{\beta y^2} \quad \beta = 0.075$

- Run using high-RE and low-RE approaches.
- Compute the initial values of the turbulent quantities using a turbulent intensity value equal to 1% and an eddy viscosity ratio equal to 1.
- After computing the solution with  $\kappa - \omega$  model, try to setup the case using the standard  $\kappa - \epsilon$  model and the realizable  $\kappa - \epsilon$ .

# Turbulence modeling hands-on tutorials

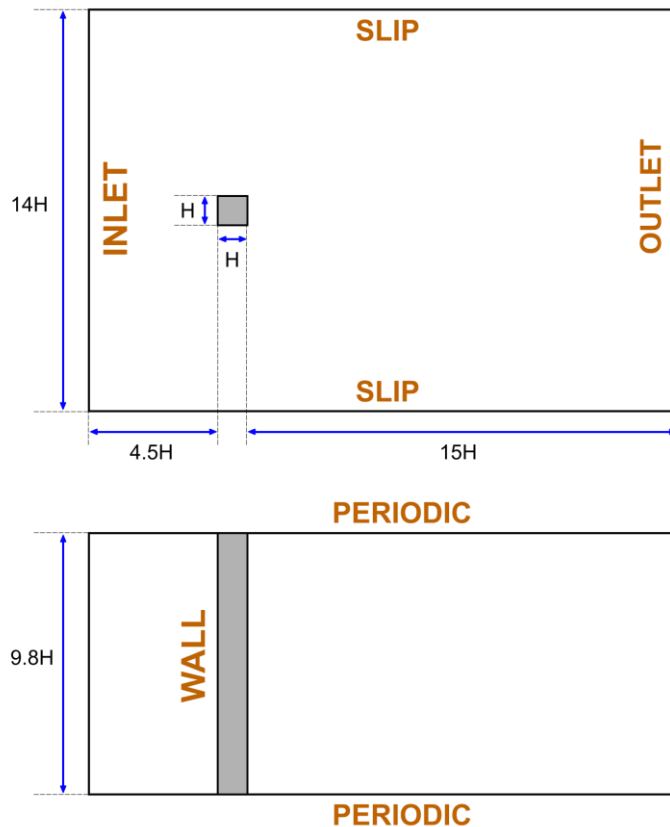
- Vortex shedding past square cylinder
- Let us run this case. Go to the directory:

```
$PTOFC/advanced_physics/turbulence/squarecil
```

- 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.

# Turbulence modeling hands-on tutorials

## Vortex shedding past square cylinder



### Physical and numerical side of the problem:

- The governing equations of the problem are the incompressible Navier-Stokes equations.
- To model the turbulence, we will use two approaches, LES and RANS.
- We are going to work in a 3D domain with periodic boundary conditions.
- This problem has plenty of experimental data for validation.

$$U_{in} = 0.535 \text{ m/s}$$

$$H = 0.04 \text{ m}$$

$$Re = 21400$$

Working fluid: Water

# Turbulence modeling hands-on tutorials

## Vortex shedding past square cylinder

Turbulence model	Drag coefficient	Strouhal number	Computing time (s)
Laminar	2.81	0.179	93489
LES	2.36	0.124	77465
DES	2.08	0.124	70754
SAS	2.40	0.164	57690
URANS (WF)	2.31	0.130	67830
URANS (No WF)	2.29	0.135	64492
RANS	2.30	-	28246 (10000 iter)
Experimental values	2.05-2.25	0.132	-

### References:

Lyn, D.A. and Rodi, W., The flapping shear layer formed by flow separation from the forward corner of a square cylinder. *J. Fluid Mech.*, 267, 353, 1994.  
Lyn, D.A., Einav, S., Rodi, W. and Park, J.H., A laser-Doppler velocimetry study of ensemble-averaged characteristics of the turbulent near wake of a square cylinder. *Report. SFB 210 /E/100*.

# Turbulence modeling hands-on tutorials

## Vortex shedding past square cylinder

- We will use this case to learn how to setup a turbulent case (RANS and LES).
- To run this case, we will use the solvers *simpleFoam* (steady solver) and *pimpleFoam* (unsteady solver).
- To get fast outcomes, we will use a coarse mesh. But feel free to refine the mesh, especially close to the walls.
- Remember, as we are introducing new closure equations for the turbulence problem, we need to define initial and boundary conditions for the new variables.
- We will use a few **functionObjects** to compute some additional quantities, such as, Q criterion,  $y^+$ , minimum and maximum values, forces, time average, and online sampling.
- After finding the solution, we will visualize the results.
- We will also compare the numerical solution with the experimental results.
- At the end, we will do some plotting and advanced post-processing using gnuplot and Python.
- Have in mind that the unsteady case will generate a lot of data.

# Turbulence modeling hands-on tutorials

## Vortex shedding past square cylinder

- We select the turbulence model in the *momentumTransport* dictionary file.
- This dictionary file is located in the directory **constant**.
- To select the LES (Smagorinsky) turbulence model,

```
17  simulationType  LES; ← LES type simulation
18
19  LES ← LES sub-dictionary
20  {
21      LESModel      Smagorinsky; ← LES model to use
22
23
24      turbulence    on; ← Turn on/off turbulence. Runtime modifiable
25
26      printCoeffs   on; ← Print coefficients at the beginning
27
28      delta          cubeRootVol;
29      cubeRootVolCoeffs
30      {
31          deltaCoeff  1;
32      }
33
34  }
100 }
```

Diagram illustrating the configuration of the LES (Smagorinsky) turbulence model in the *momentumTransport* dictionary file. The configuration is shown with line numbers and annotations:

- Line 17: `simulationType LES;` (LES type simulation)
- Line 19: `LES` (LES sub-dictionary)
- Line 21: `LESModel Smagorinsky;` (LES model to use)
- Line 24: `turbulence on;` (Turn on/off turbulence. Runtime modifiable)
- Line 25: `printCoeffs on;` (Print coefficients at the beginning)
- Lines 27-34: `delta cubeRootVol;` and `cubeRootVolCoeffs { deltaCoeff 1; }` (LES filter)

- Remember, you need to assign boundary and initial conditions to the new variables (**nut**).



# Turbulence modeling hands-on tutorials

## Vortex shedding past square cylinder

- To define the wall functions, follow this table,

Field	Wall functions – High RE	Resolved BL – Low RE
<b>nut</b>	<b>nutUSpaldingWallFunction</b>	<b>fixedValue 0</b> or a small number
<b>k</b>	<b>kqRWallFunction</b>	<b>fixedValue 0</b> or a small number
<b>omega</b>	<b>omegaWallFunction</b> $\omega_{wall} = 10 \frac{6\nu}{\beta y^2} \quad \beta = 0.075$	<b>omegaWallFunction</b> $\omega_{wall} = 10 \frac{6\nu}{\beta y^2} \quad \beta = 0.075$

- Run using the following combinations of wall functions and compare the outcome.
  - Use High RE for RANS.
  - Use High RE and Low RE for URANS.
  - Use High RE and Low RE for LES.

# Turbulence modeling hands-on tutorials

## Vortex shedding past square cylinder

- The initial value for the turbulent kinetic energy  $\kappa$  can be found as follows,

$$\kappa = \frac{3}{2}(UI)^2$$

- The initial value for the specific dissipation rate  $\omega$  can be found as follows,

$$\omega = \frac{\rho \kappa}{\mu} \frac{\mu_t}{\mu}^{-1}$$

- Use the following initial estimates,  $I = 5\%$  and  $\frac{\mu_t}{\mu} = 10$
- At this point, we are ready to run. But before running, remember to setup the right numerics in the dictionary files *fvSolution* and *fvSchemes*.
- For the LES simulation, try to keep the CFL number below 0.9. For the URANS simulation, you can go as high as 10.
- Finally, do not forget to setup the **functionObjects** to compute the forces, average values, do the sampling, and compute  $y^+$  on-the-fly.

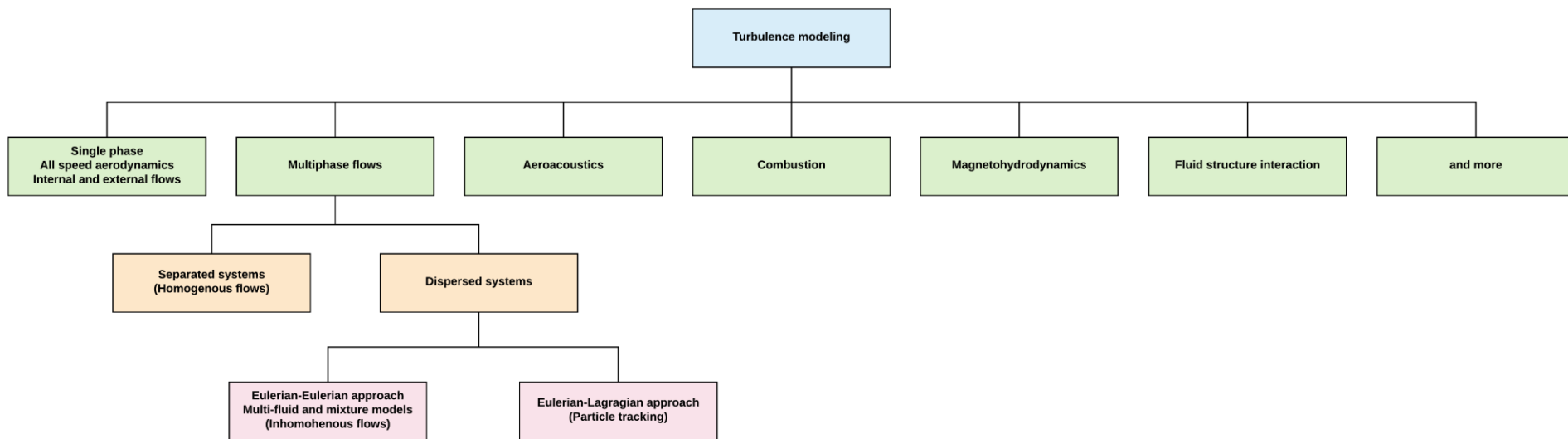
# Roadmap

**A crash introduction to:**

- ~~1. Turbulence modeling in OpenFOAM®~~
- 2. Multiphase flows modeling in OpenFOAM®**
- ~~3. Compressible flows in OpenFOAM®~~
- ~~4. Source terms in OpenFOAM®~~
- ~~5. Scalar transport pluggable solver~~
- ~~6. Moving bodies in OpenFOAM®~~

# A crash introduction to multiphase flows modeling OpenFOAM®

- Turbulence modeling is on top of every type of physics to be simulated, including multiphase flows.
- As in turbulence modeling, when dealing with multiphase flows we can take an approach that resolves all scales, but it is too expensive.
- Therefore, the need of using models.
- And depending on the multiphase system, there are models very specific for the multiphase physics involved.
- Multiphase flows are heavily modeled.



# A crash introduction to multiphase flows modeling OpenFOAM®

## What is a multiphase flow?

- A multiphase flow is a fluid flow consisting of more than one phase component and have some level of phase separation above molecular level.
- Multiphase flows exist in many different forms. For example, two phase flows can be classified according to the state of the different phases:
  - Gas-Liquid mixture.
  - Gas-Solid mixture.
  - Liquid-Solid mixture.
  - Immiscible liquid-liquid.
- Multiphase flows are present in many industrial processes and natural systems.
- Hence the importance of understanding, modeling, and simulating multiphase flows.



# A crash introduction to multiphase flows modeling OpenFOAM®

## Examples of multiphase flows



**Municipal and industrial water treatment**

<http://www.asiapacific.basf.com/apex/AP/en/upload/Press2010/BASF-Water-Chem-2010-Paper-Chem-2010-Intex-Shanghai>



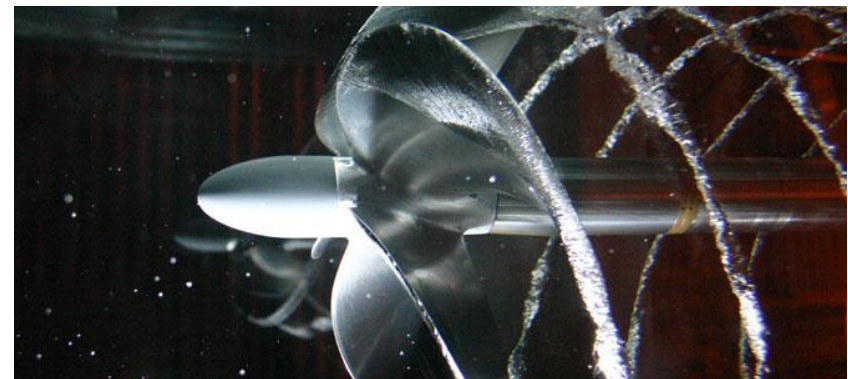
**Cargo ship wake**

<http://developeconomies.com/development-economics/how-to-get-america-back-on-track-free-trade-edition/>



**Siltation & Sedimentation**

<http://blackwarriorriver.org/siltation-sedimentation/>



**Propeller cavitation**

<http://www.veempropellers.com/features/cavitationresistance>



# A crash introduction to multiphase flows modeling OpenFOAM®

## Examples of multiphase flows



**Cooling Towers**

<https://whatiswatertreatment.wordpress.com/what-are-the-systems-associated-with-water-treatment-and-how-are-they-treated/103-2/>



**Volcano eruption**

<http://americanpreppersnetwork.com/2014/08/preparing-volcano-eruption.html>



**Fermentation of beer and spirits**

<http://www.distillingliquor.com/2015/02/05/how-to-make-alcohol-and-spirits/>



**Chemical reactor for the pharmaceutical and biotechnology industry**

<http://www.total-mechanical.com/Industrial/CaseStudies.aspx>

## Why simulating multiphase flows is challenging?

- Simulating multiphase flows is not an easy task.
- The complex nature of multiphase flows is due to:
  - More than one working fluid.
  - The transient nature of the flows.
  - The existence of dynamically changing interfaces.
  - Significant discontinuities (fluid properties and fluid separation).
  - Complicated flow field near the interface.
  - Interaction of small-scale structures (bubbles and particles).
  - Different spatial-temporal scales.
  - Dispersed phases and particle-particle interactions.
  - Mass transfer and phase change.
  - Turbulence.
  - Many models involved (drag, lift, heat transfer, turbulence dispersion, frictional stresses, collisions, kinetic theory, and so on).



## Classifying multiphase flows according to phase morphology

- **Disperse system:** the phase is dispersed as non-contiguous isolated regions within the other phase (the continuous phase) . When we work with a disperse phase, we say that the system is dispersed: disperse-continuous flow.
- **Separated system:** the phase is contiguous throughout the domain and there is one well defined interphase with the other phase. When we work with continuous phases, we say that the system is separated: continuous-continuous flow.



Dispersed system

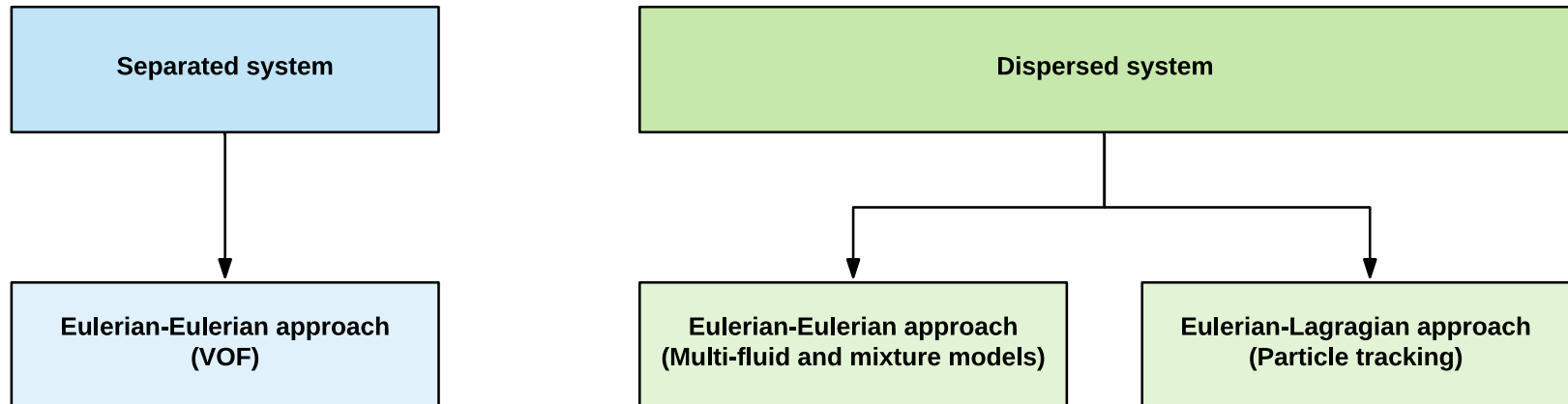
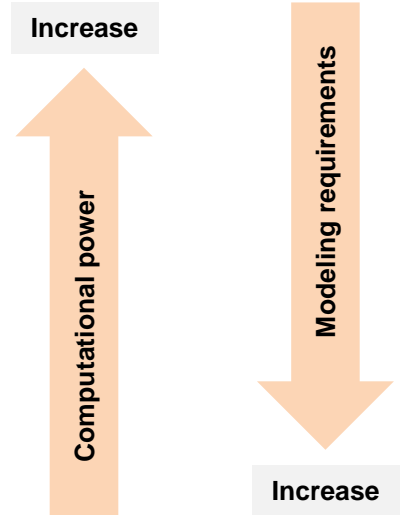


Separated system

# A crash introduction to multiphase flows modeling OpenFOAM®

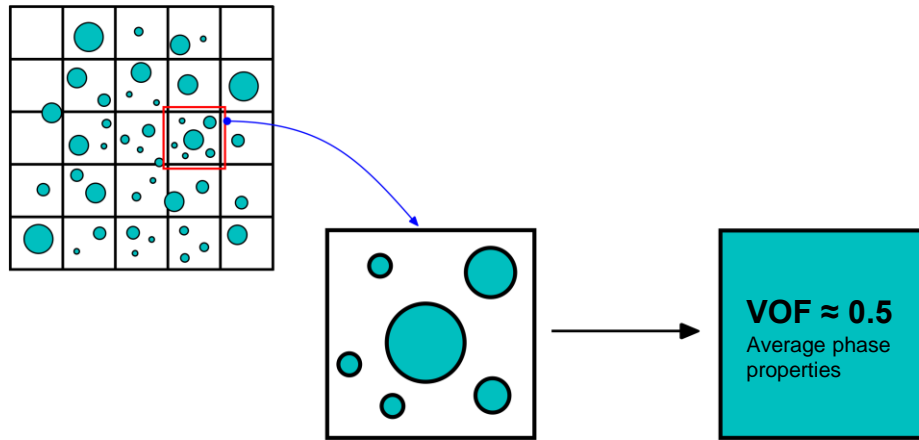
## How to treat the wide range of behaviors in multiphase flows

- **Fully resolved:** solves complete physics. All spatial and temporal scales are resolved. Equivalent to DNS in turbulence modelling.
- **Eulerian-Lagrangian:** solves idealized isolated particles that are transported with the flow. One- or two-way coupling is possible. It can account for turbulence, momentum transfer, and mass transfer.
- **Eulerian-eulerian:** solves two or more co-existing fluids. The system can be dispersed or separated, and can account for turbulence, momentum transfer, and mass transfer.

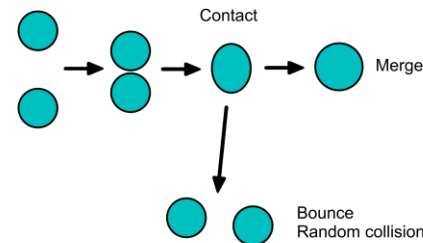


# A crash introduction to multiphase flows modeling OpenFOAM®

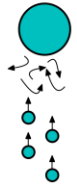
## How to treat the wide range of behaviors in multiphase flows



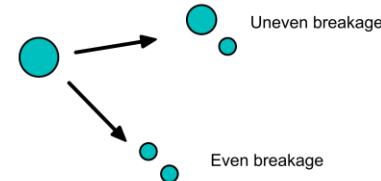
Coalescence mechanism



Wake entrainment



Break-up mechanism



- Dispersed phase in a continuous phase.
- In this case, the VOF method is not able to handle bubbles smaller than grid scales.
- Multi-fluid and mixture models are able to model bubbles smaller than grid scales by averaging the phase properties in the discrete domain.

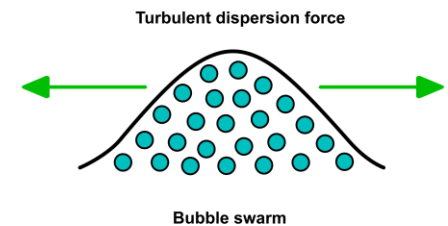
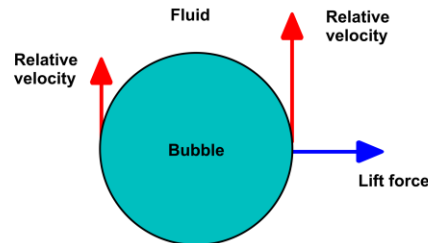
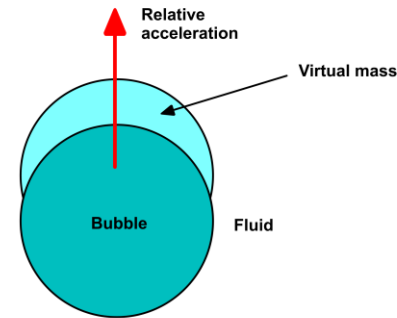
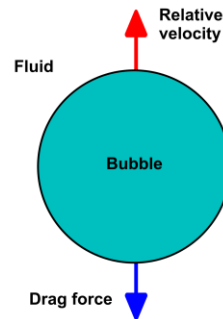
- Multi-fluid and mixture approaches can model bubble coalescence, bubble break-up and wake entrainment in dispersed systems.

In theory, the VOF method can resolve the smallest bubbles/droplets but the mesh requirements are too prohibitive (equivalent to DNS). In multiphase flows, this is called fully resolved approach.

# A crash introduction to multiphase flows modeling OpenFOAM®

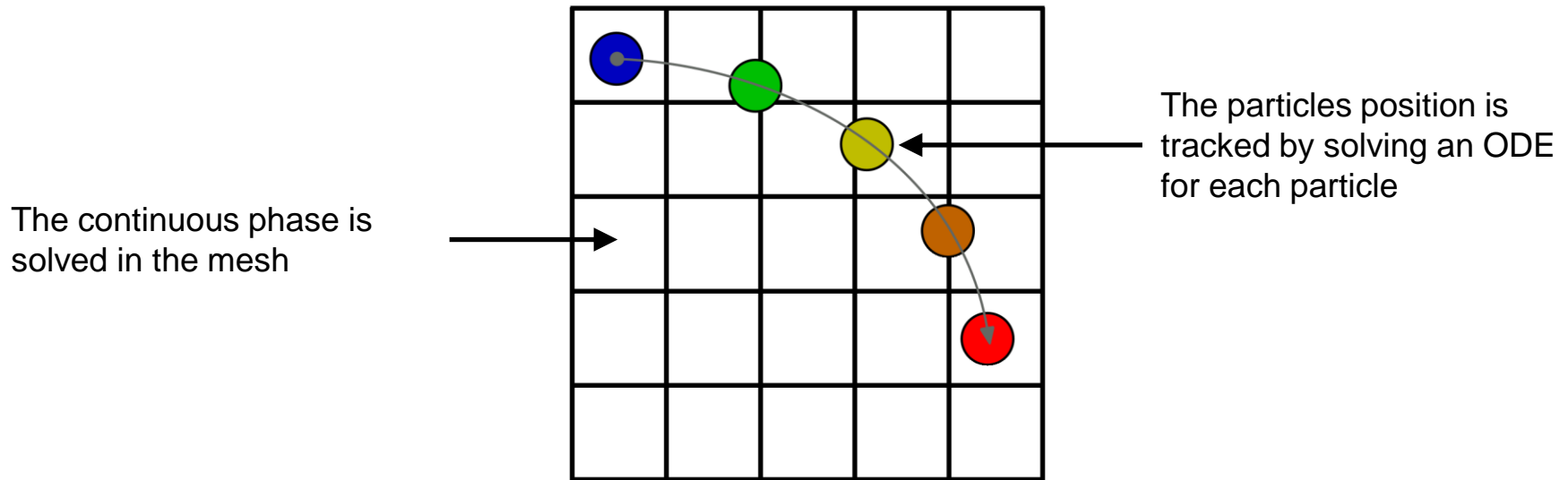
## How to treat the wide range of behaviors in multiphase flows

- When using multi-fluid and mixture approaches, interfacial momentum transfer models must be taken into account in order to model mass transfer and phases interaction.
- As for turbulence modeling, there is no universal model.
- It is up to you to choose the model that best fit the problem you are solving.
- Depending on the physics involved, you will find different models and formulations
- You need to know the applicability and limitations of each model, for this, refer to the literature.



# A crash introduction to multiphase flows modeling OpenFOAM®

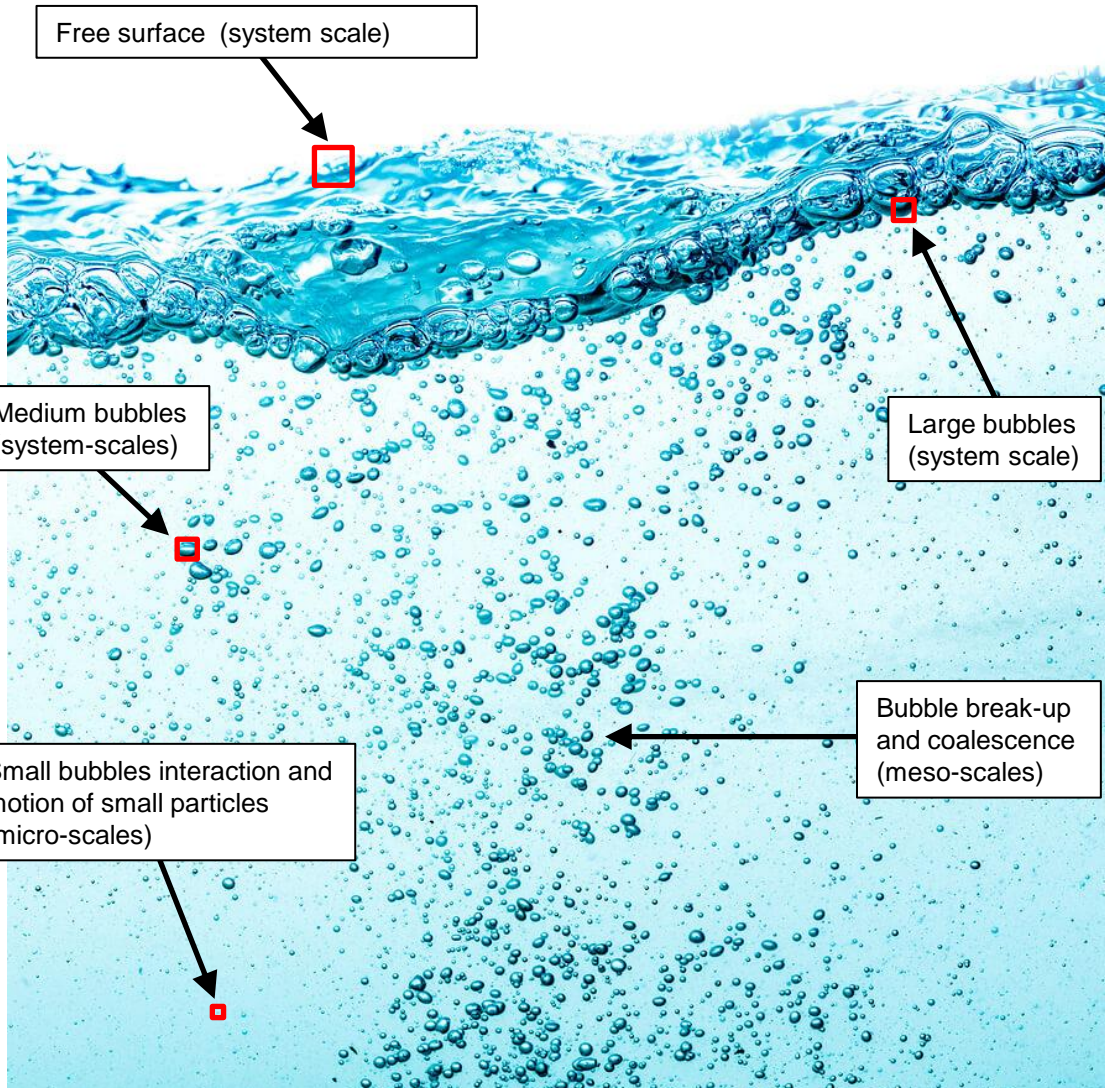
## How to treat the wide range of behaviors in multiphase flows



- In the Eulerian-Lagrangian framework, the continuous phase is solved in an Eulerian reference system and the particles or dispersed phase is solved in a Lagrangian reference system.
- The particles can be smaller or larger than the grid size.
- The particles can be transported passively, or they can be coupled with the fluid governing equations.
- It accounts for particle interaction and mass transfer.
- The particles can interact with the boundaries and have a fate.

# A crash introduction to multiphase flows modeling OpenFOAM®

## Multiphase flows – Grid scales



- Applicability of the VOF method to separated systems (non-interpenetrating continua).
- In the figure, the free surface and large bubbles can be track/resolve by the mesh.
- The smaller the features we want to track/resolve, the smaller the cells should be.
- Bubbles, droplets and/or particles larger than grid scales (GS), can be resolved using VOF.
- To resolve a bubble, you will need at least two cells in every direction
- Bubbles, droplets and/or particles smaller than grid scales (sub-grid scales or SGS), can not be resolve using the VOF method.
- In such a case, we need to use models.
- Also, bubble break-up, coalescence and entrainment must be modeled, unless the mesh is fine enough so it captures the dynamics and solves the smallest scales.

**Multiphase flows are transient and multiscale**

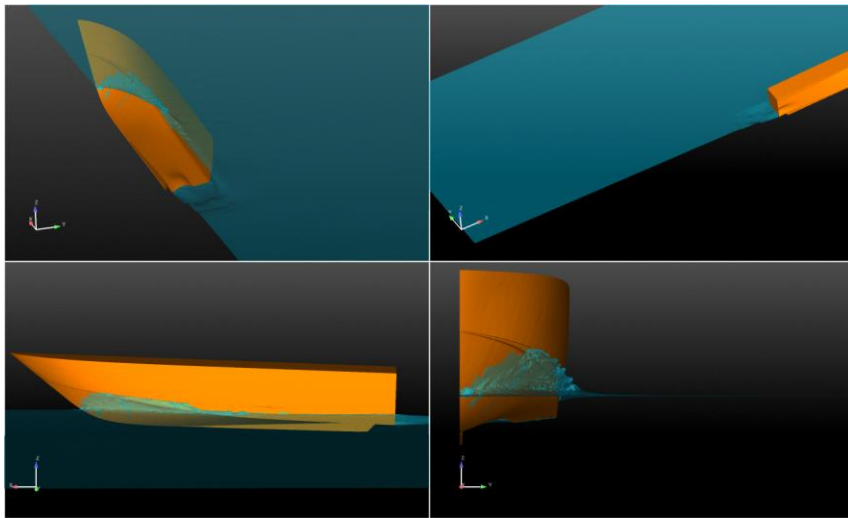


## Numerical approaches for multiphase flows

<b>Eulerian-Eulerian (VOF for separated systems)</b>	<b>Eulerian-Eulerian (Dispersed systems)</b>	<b>Eulerian-Lagrangian</b>
<ul style="list-style-type: none"><li>• Non-interpenetrating continua.</li><li>• Continuous phases: Eulerian.</li><li>• Fluid properties are written on either side of the interface (no averaging).</li><li>• Solves one single set of PDEs: mass, momentum, energy.</li></ul>	<ul style="list-style-type: none"><li>• Interpenetrating continua.</li><li>• Continuous phase: Eulerian.</li><li>• Dispersed phase: Eulerian.</li><li>• Phase-weighted averages.</li><li>• Solves PDEs for all phases (including interphase transfer terms): mass, momentum, energy.</li><li>• It can deal with gas-liquid, gas-solid, and liquid-solid interactions.</li></ul>	<ul style="list-style-type: none"><li>• Continuous phase: Eulerian.</li><li>• Dispersed phase: Lagrangian.</li><li>• Solves ODEs for particle tracking (for every single particle).</li><li>• Solves a set of PDEs for the continuous phase: mass, momentum, energy.</li><li>• Phase interaction terms (including interphase transfer terms).</li></ul>

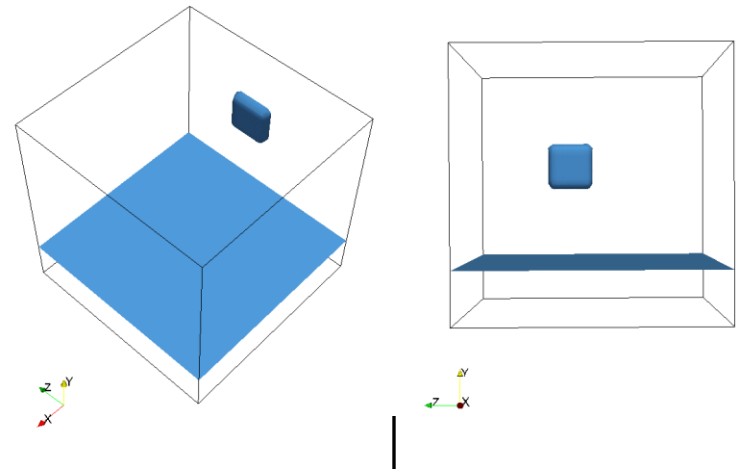
# A crash introduction to multiphase flows modeling OpenFOAM®

## Numerical approaches for multiphase flows



[www.wolfdynamics.com/training/mphase/image10.gif](http://www.wolfdynamics.com/training/mphase/image10.gif)

Time: 0.050000



<http://www.wolfdynamics.com/training/mphase/image16.gif>

- Simulations showing free surface tracking using the VOF approach
- The left image corresponds to a simulation with rigid body motion and accurate surface tracking using the VOF method.

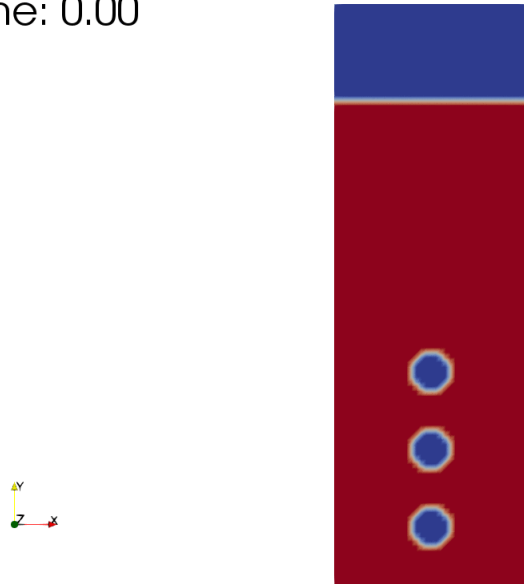


# A crash introduction to multiphase flows modeling OpenFOAM®

## Numerical approaches for multiphase flows

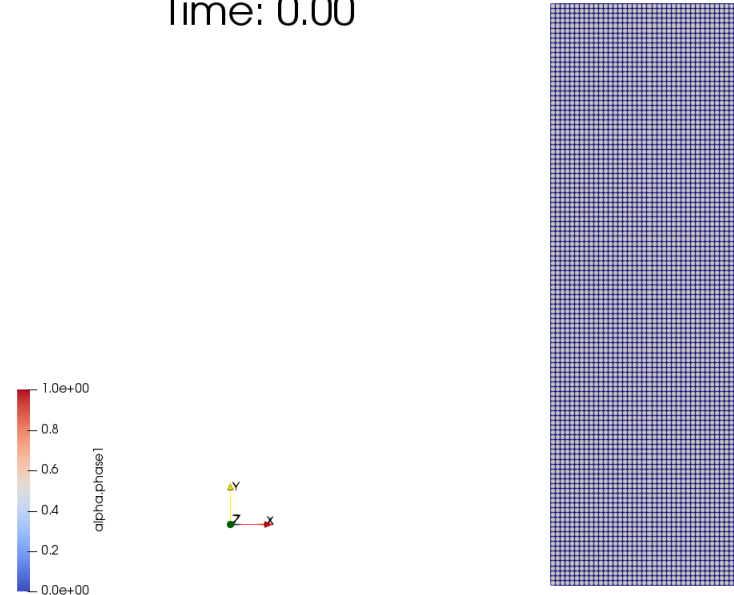


Time: 0.00



<http://www.wolfdynamics.com/training/mphase/image2.gif>

Time: 0.00



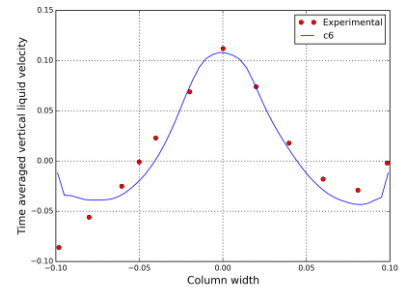
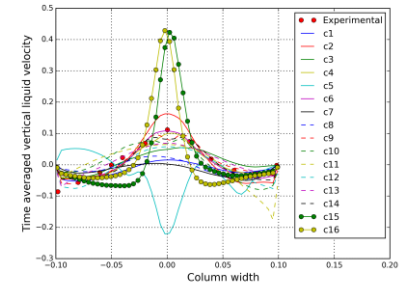
<http://www.wolfdynamics.com/training/mphase/image3.gif>

- Simulation showing free surface tracking, bubble tracking, bubble coalescence, bubble break-up and wake entrainment using the VOF method.
- In this simulation the free surface and bubbles are capture by using AMR.
- However, the smallest bubble that can be resolved is at the smallest grid size.

# A crash introduction to multiphase flows modeling OpenFOAM®

## Numerical approaches for multiphase flows

Time: 0.5



<http://www.wolfdynamics.com/training/mphase/image18.gif>

- Eulerian-Eulerian simulation (gas-liquid).
- The bubbles are not being solved, instead, the interaction between phase is being averaged.

### 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

# A crash introduction to multiphase flows modeling OpenFOAM®

## Numerical approaches for multiphase flows

Time: 0.00



**twoPhaseEulerFoam**  
**Air volume fraction**  
**Turbulent case**

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

Time: 0.00



**twoPhaseEulerFoam**  
**Air volume fraction**  
**Laminar case**

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

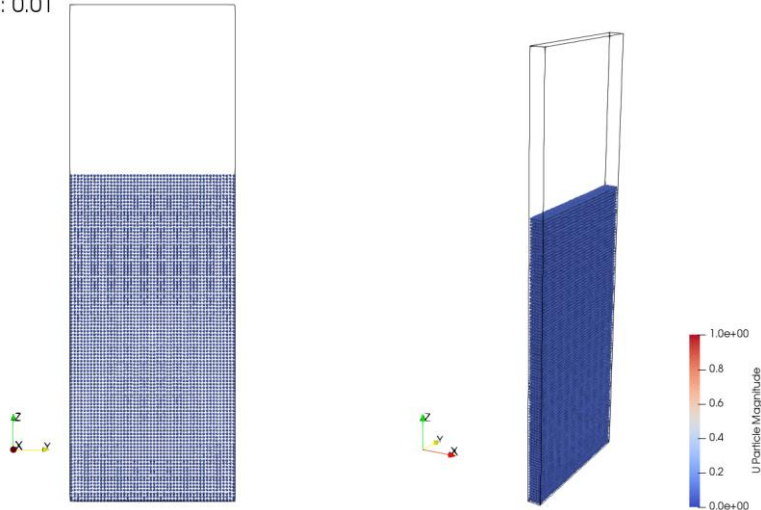


- Eulerian-Eulerian simulations using the Eulerian-Granular KTGF approach (solid-gas).
- The granular phase is simulated as continuous phase.
- In these simulations we can observe the influence of turbulence modeling in the solution.

# A crash introduction to multiphase flows modeling OpenFOAM®

## Numerical approaches for multiphase flows

Time: 0.01



**DPMFoam**

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

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

Time: 0.00



**twoPhaseEulerFoam**

**Air volume fraction  
Turbulent case**

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

- Comparison of an Eulerian-Lagrangian simulation and an Eulerian-Eulerian simulation (gas-solid).
- In the Eulerian-Lagrangian approach we track the position of every single particle. We also solve the fate and interaction of all particles.
- In the Eulerian-Eulerian approach we solve the granular phase as a continuous phase.
- The computational requirements of the Eulerian-Eulerian simulation are much lower than those for the Eulerian-Lagrangian simulation.

# A crash introduction to multiphase flows modeling OpenFOAM®

## Volume-of-Fluid (VOF) governing equations for separated systems

- The incompressible, isothermal governing equations can be written as follows,

$$\nabla \cdot \mathbf{U} = 0$$

Surface tension - Continuum surface force (CSF)

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot (\rho \mathbf{U} \mathbf{U}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{g} + f_{\sigma} + \rho S$$

Source terms:

- Porous media
- Coriolis forces
- Centrifugal forces
- Mass transfer
- and so on ...

$$\frac{\partial \gamma}{\partial t} + \nabla \cdot \mathbf{U} \gamma + \nabla \cdot (\mathbf{U}_r \gamma (1 - \gamma)) = 0 \longrightarrow$$

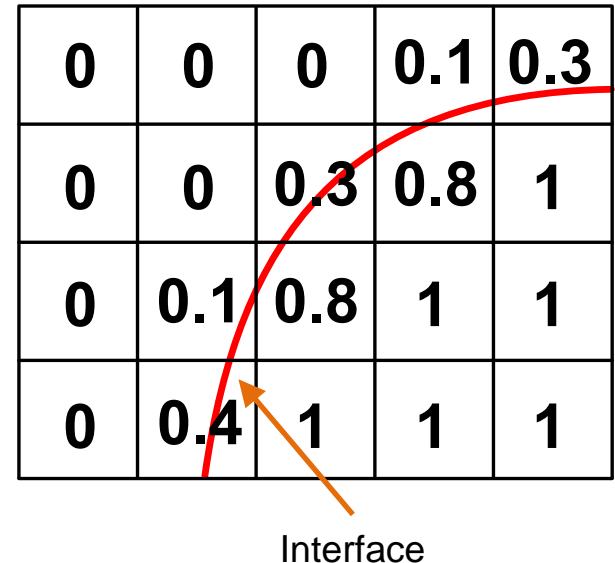
Phase transport equation and interface tracking with surface compression

$$0 < \gamma < 1 \longrightarrow \text{Volume fraction (bounded quantity)}$$

- You can see the volume fraction  $\gamma$  as a pointer that indicates what phase (with the corresponding physical properties), is inside each cell of the computational domain.

## Volume-of-Fluid (VOF) governing equations for separated systems

- For example, in the case of two phases where phase 1 is represented by  $\gamma = 1$  and phase 2 is represented by  $\gamma = 0$ ; a volume fraction value of 1 indicates that the cell is fill with phase 1; a volume fraction of 0.8 indicates that the cell contains 80% of a phase 1; and a volume fraction of 0, indicates that the cell is fill with phase 2.
- The values between 0 and 1 can be seen as the interface between the phases.



- The fluid properties can be written on either side of the interface as follows,

$$\rho = \gamma_1 \rho_1 + (1 - \gamma_1) \rho_2$$

$$\mu = \gamma_1 \mu_1 + (1 - \gamma_1) \mu_2$$

# A crash introduction to multiphase flows modeling OpenFOAM®

## Eulerian-Eulerian governing equations for dispersed systems

- The Eulerian-Eulerian approach solves the governing equations for each phase, it treats the phases as interpenetrating continua.
- The incompressible, isothermal governing equations with interface tracking can be written as follows,

$$\frac{\partial \alpha_k \rho_k}{\partial t} + \nabla \cdot (\alpha_k \rho_k \mathbf{U}_k) = 0$$

$$\frac{\partial (\alpha_k \rho_k \mathbf{U}_k)}{\partial t} + \nabla \cdot (\alpha_k \rho_k \mathbf{U}_k \mathbf{U}_k) = -\nabla \cdot (\alpha_k \tau_k) - \alpha_k \nabla p + \alpha_k \rho_k \mathbf{g} + \mathbf{M}_k + f_\sigma + \mathbf{S}_k$$

$$\frac{\partial \alpha_k \rho_k}{\partial t} + \nabla \cdot \mathbf{U}_k \alpha_k \rho_k + \nabla \cdot (\mathbf{U}_r \alpha_k \rho_k (1 - \alpha_k)) = 0$$

$$\sum_k \alpha_k = 1.0 \quad \rho_m = \sum_k \alpha_k \rho_k \quad \mathbf{U}_m = \frac{\sum_k \alpha_k \rho_k \mathbf{U}_k}{\rho_m}$$

Surface tension - Continuum surface force (CSF)

Interface forces or momentum transfer.  
Bubbles interaction models

Source terms:

- Porous media
- Coriolis forces
- Centrifugal forces
- Mass transfer
- and so on ...

## Eulerian-Lagrangian governing equations

- In the Eulerian-Lagrangian framework, the continuous phase is solved in an Eulerian reference system and the particles or dispersed phase is solved in a Lagrangian reference system.
- The particles can be transported passively, or they can be coupled with the fluid governing equations (they can modify the fluid field).
- The particles motion is calculated by solving an ODE for every single particle (Newton-Euler equation of motion).
- The particles can interact with the boundaries, they can escape, bounce, stick, or form a wall film.
- This formulation accounts for particle interaction and mass transfer.
- The governing equations can be written as follows,

$$m \frac{d\mathbf{U}}{dt} = \mathbf{F}_{drag} + \mathbf{F}_{pressure} + \mathbf{F}_{virtual\ mass} + \mathbf{F}_{other}$$
$$+$$

Any of the Eulerian formulations (single or multi-phase)



# A crash introduction to multiphase flows modeling OpenFOAM®

## Multiphase solvers in OpenFOAM®

- In OpenFOAM®, there are many interfacial momentum transfer models implemented.
- There are also many models for Eulerian-Lagrangian methods.
- No need to say that turbulence also applies to multiphase flows.
- There is no universal model, it is up to you to choose the model that best fit the problem you are solving.
- You need to know the applicability and limitations of each model, for this, refer to the literature.
- When dealing with multiphase flows in OpenFOAM®, you can use VOF, Eulerian-Eulerian, Eulerian-Eulerian with VOF, and Eulerian-Lagrangian methods.
- The solution methods can account for turbulence models, interface momentum transfer models, mass transfer models, particle interaction models and so on.
- It is also possible to add source terms, deal with moving bodies or use adaptive mesh refinement.
- You will find the source code of all the multiphase solvers in the directory:
  - **OpenFOAM-9/applications/solvers/multiphase**
- You will find the source code all the particle tracking solvers in the directory:
  - **OpenFOAM-9/applications/solvers/lagrangian**

# A crash introduction to multiphase flows modeling OpenFOAM®

## Multiphase solvers in OpenFOAM®

- These are the multiphase solvers that you will use most of the time in OpenFOAM®.
- The VOF approach:
  - interFoam family solvers
- The Eulerian-Eulerian approach:
  - twoPhaseEulerFoam, multiphaseEulerFoam
- The Eulerian-Granular KTGF (kinetic theory of granular flows) approach.
  - twoPhaseEulerFoam
- The Eulerian-Lagrangian framework,
  - DPMFoam, MPPICFoam

# Multiphase flows hands-on tutorials

- Free surface – Ship resistance simulation
- Let us run this case. Go to the directory:

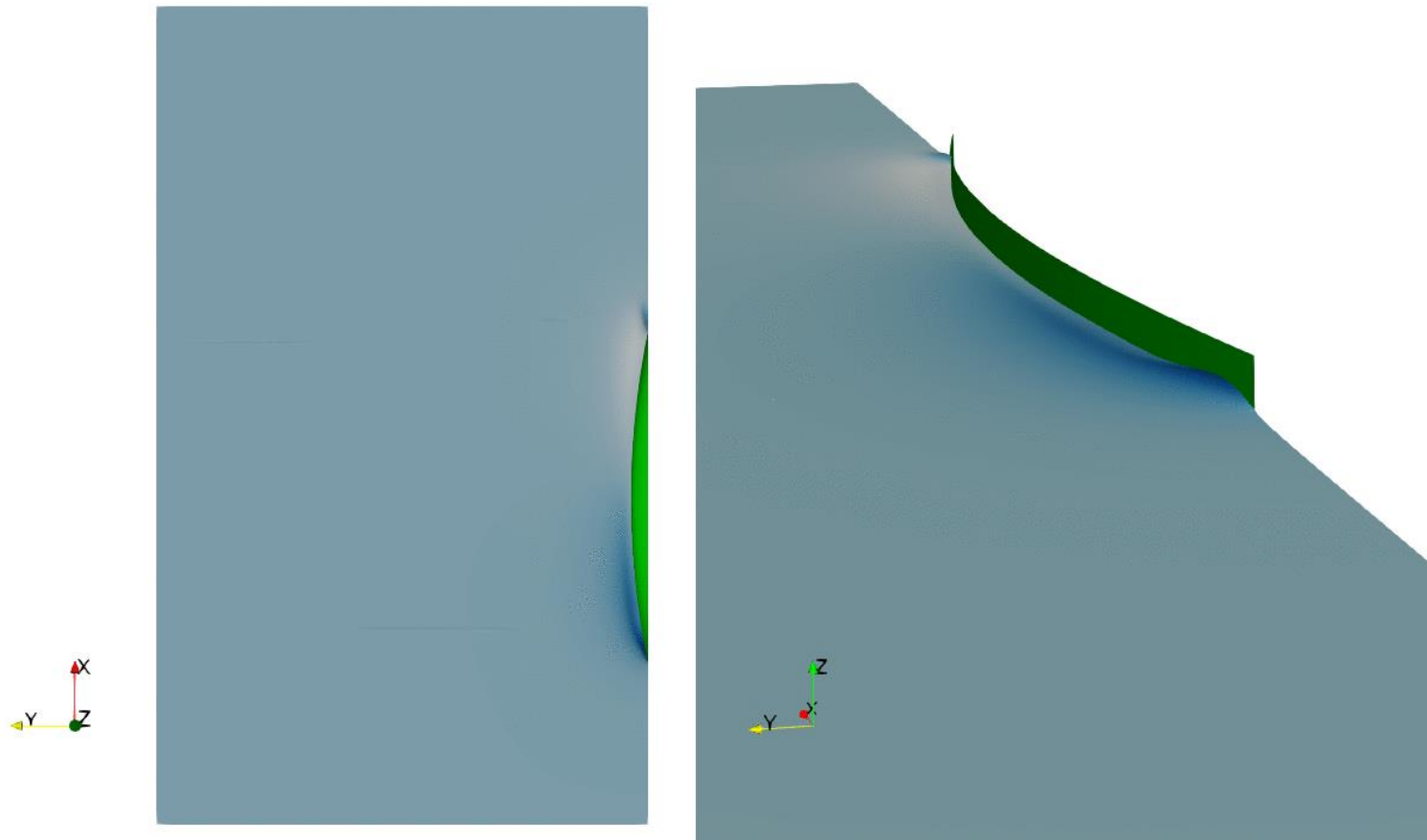
```
$PTOFC/advanced_physics/multiphase/wigleyHull
```

- 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.

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

Time: 0.25

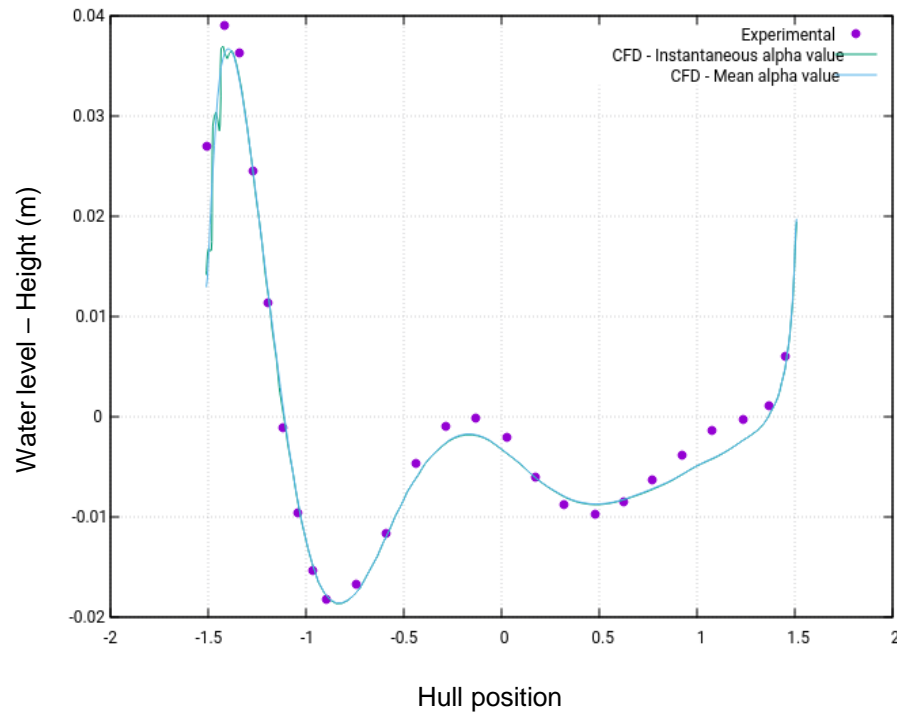


Free surface colored by height

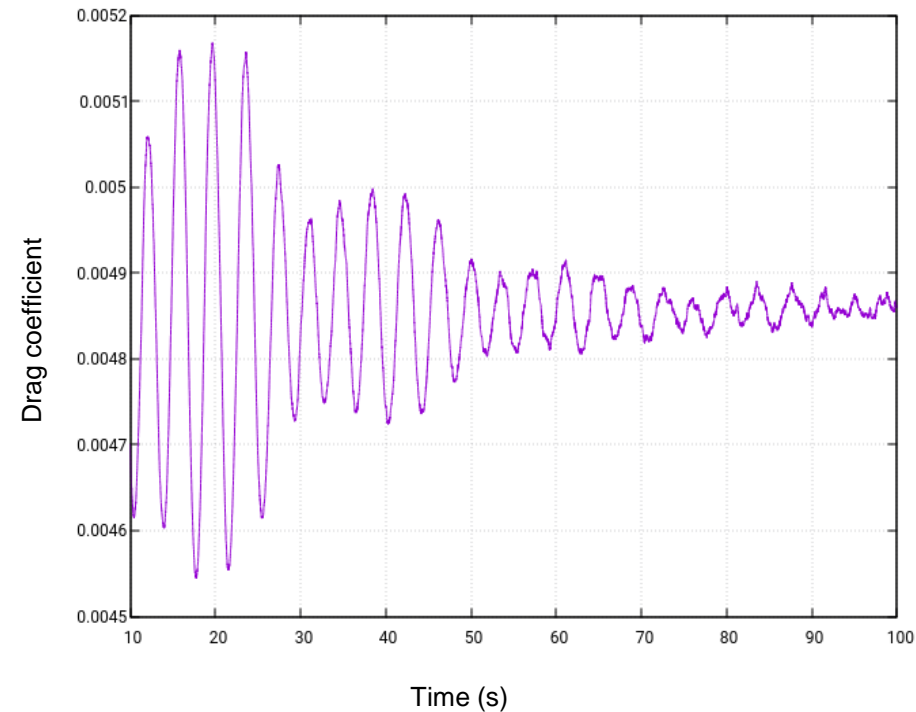
<http://www.wolfdynamics.com/training/mphase/image45.gif>

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation



Comparison of water level on hull surface

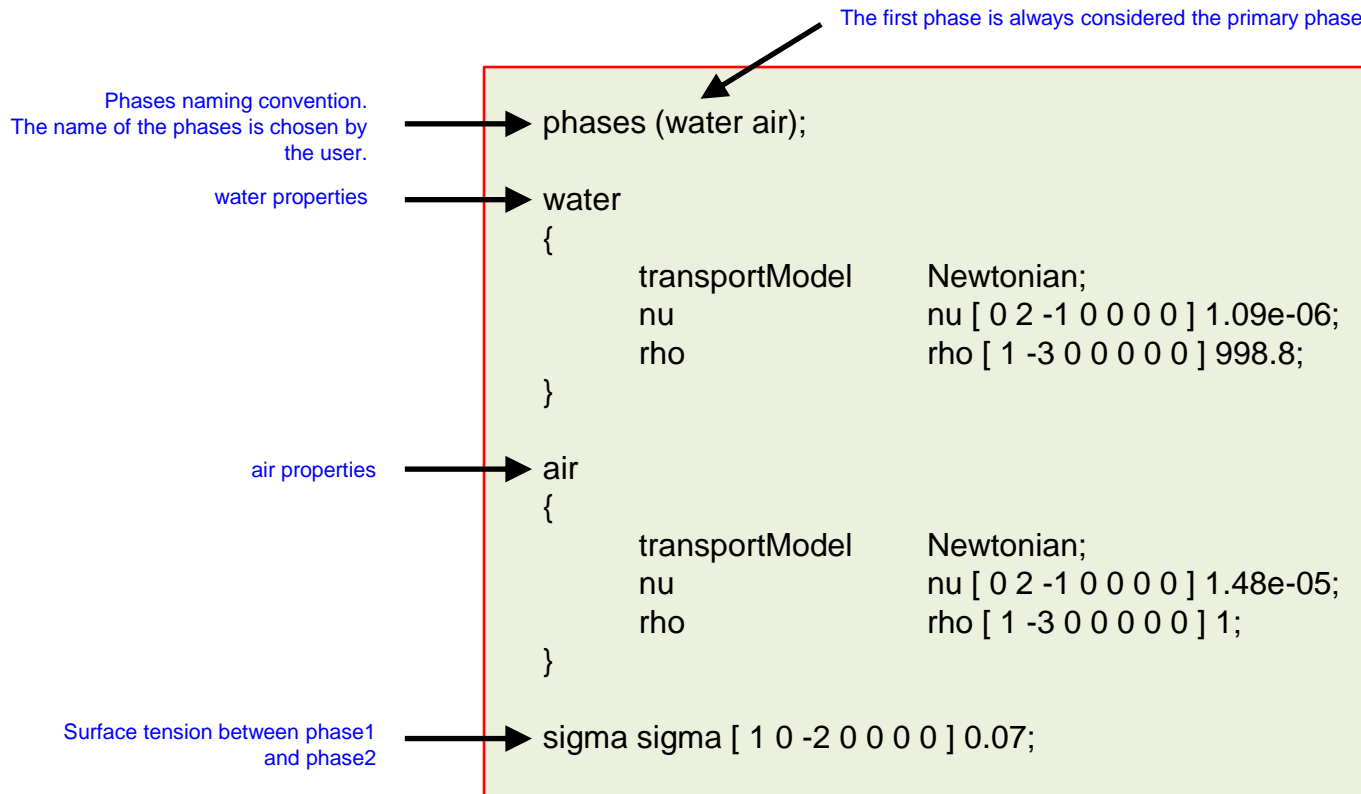


Drag coefficient monitor

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- 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*.



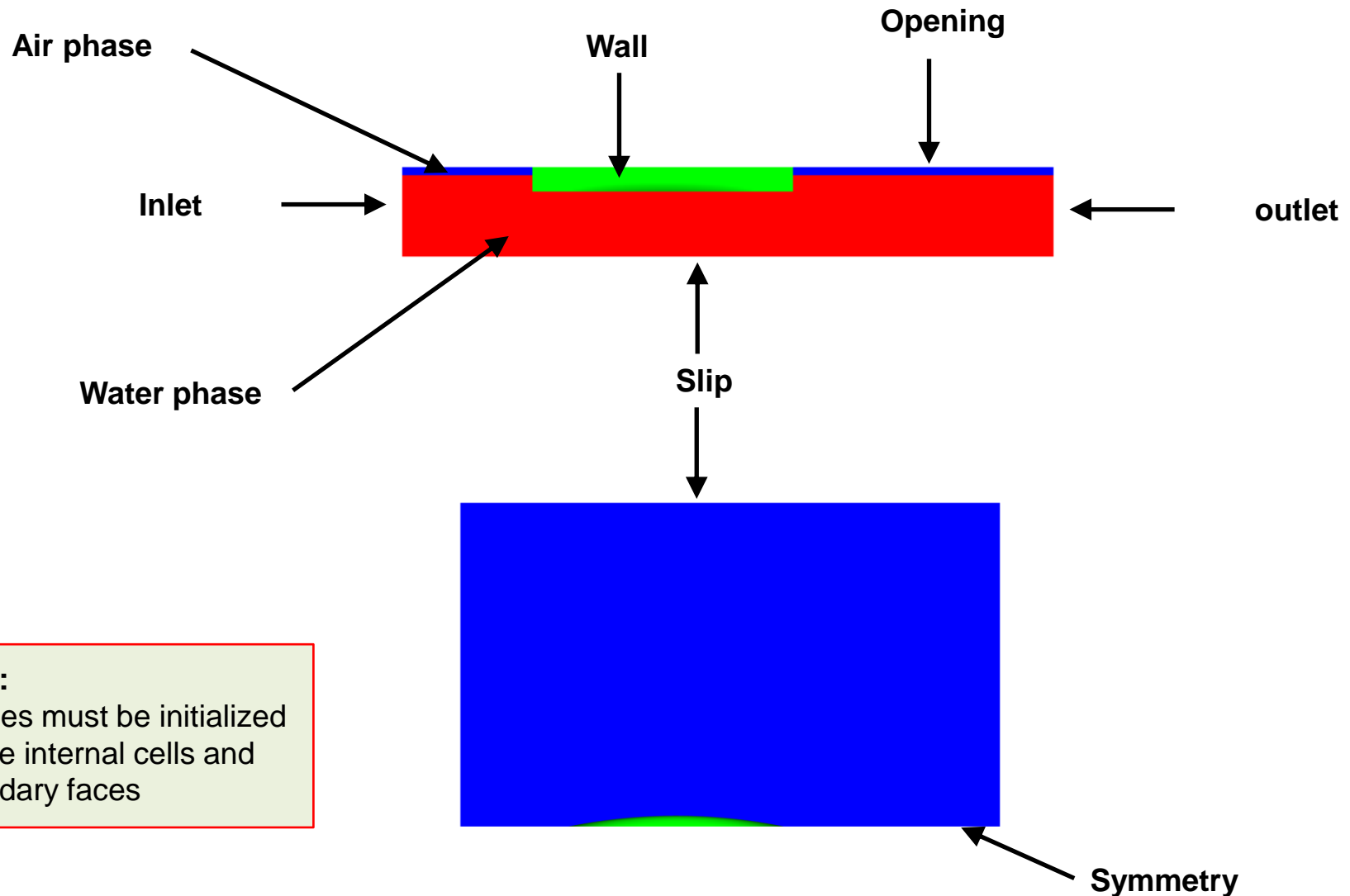
# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- 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).
- If you are using a different solver (e.g., `twoPhaseEulerFoam`), you will need to use additional dictionaries where you define the interfacial models and so on.
- Remember, you should always conduct production runs using a second order discretization scheme

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation





# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

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

Typical setup of boundary conditions for numerical towing tank simulations

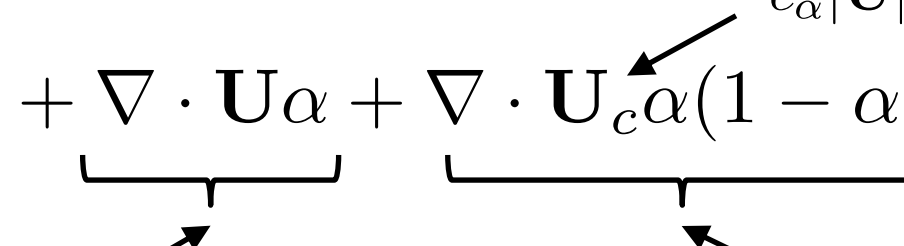
# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

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

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

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



(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 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.

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- 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.

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- Additional notes on the *fvSolution* dictionary.

momentumPredictor      yes;

nOuterCorrectors      1;

nCorrector      3;

nNonOrthogonalCorrectors      2;

← Set to yes for high Reynolds flows, where convection dominates

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

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

← Recommend to use at least 1 corrector. Increase the value for bad quality meshes.

- If you are planning to use large time-steps (CFL number larger than 1), it is recommended to do at least 2 nOuterCorrectors and 3 nCorrector.


# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- 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:

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

This term is related to the volume fraction equation



- Notice that we are using a high-resolution scheme for the surface tracking (div(phi,alpha)).

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- 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).

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- 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 (**PISO** **ITA**).

# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- 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;

}

Maximum flow Courant

Local time step  
smoothing

Limit the maximum  
time-step size

Maximum interface  
Courant

Local time step  
damping



# Multiphase flows hands-on tutorials

## Free surface – Ship resistance simulation

- 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`).

# Roadmap

## A crash introduction to:

- ~~1. Turbulence modeling in OpenFOAM®~~
- ~~2. Multiphase flows modeling in OpenFOAM®~~
- 3. Compressible flows in OpenFOAM®**
- ~~4. Source terms in OpenFOAM®~~
- ~~5. Scalar transport pluggable solver~~
- ~~6. Moving bodies in OpenFOAM®~~

## What are compressible flows?

- In few words, compressible flows are flows where the density change (significant changes).
- The changes in density can be due to velocity, pressure, or temperature variations.
- Compressible flows can happen at low speed (subsonic) or at high speed (transonic, supersonic, hypersonic and so on).
- Buoyancy-driven flows are also considered compressible flows. After all, the buoyancy effect is due to temperature gradients.
- In compressible flows, the viscosity also changes with temperature.
- The thermodynamical variables (pressure, temperature, and density) are related via an equation of state (e.g., ideal gas law).
- Also, the physical properties (viscosity, specific heats, thermal conductivity, and so on), have a dependence on the temperature or pressure.
  - This dependence is defined using models, such as, Sutherland model, kinematic theory, power law, polynomial models, and so on.
- In principle, all flows are compressible.
- Usually, compressibility effects start to become significant when the Mach number is higher than 0.3.

## A few compressible flows applications

- The following applications fall within the compressible flows classification:
  - External and internal aerodynamics (high speed).
  - Heat transfer and conjugate heat transfer.
  - Fire dynamics.
  - Buoyancy driven flows
  - Heating, ventilation, and air conditioning (HVAC).
  - Thermal comfort.
  - Turbomachinery.
  - Combustion.
  - Chemical reactions.
  - Condensation, evaporation, and melting.
  - Cavitation.
  - And many more.
- As you can see, the range of applicability is very wide.

# A crash introduction to compressible flows modeling OpenFOAM®



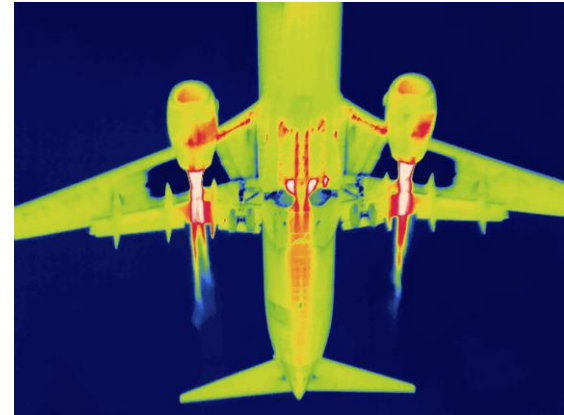
**Large Natural Convection Plume, as effect of combustion of excess non-useable gases behind oilfield.**  
[https://en.wikipedia.org/wiki/Plume\\_\(fluid\\_dynamics\)#/media/File:Natural\\_convectionplume.JPG](https://en.wikipedia.org/wiki/Plume_(fluid_dynamics)#/media/File:Natural_convectionplume.JPG)



**Iron melting**  
[https://commons.wikimedia.org/wiki/File:Iron\\_-melting.JPG](https://commons.wikimedia.org/wiki/File:Iron_-melting.JPG)

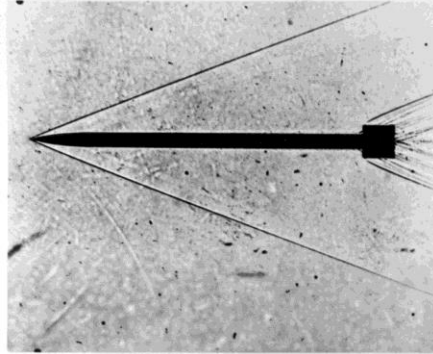


**Rayleigh–Bénard convection cells**  
[https://en.wikipedia.org/wiki/File:B%C3%A9nard\\_cells\\_convection.ogg](https://en.wikipedia.org/wiki/File:B%C3%A9nard_cells_convection.ogg)

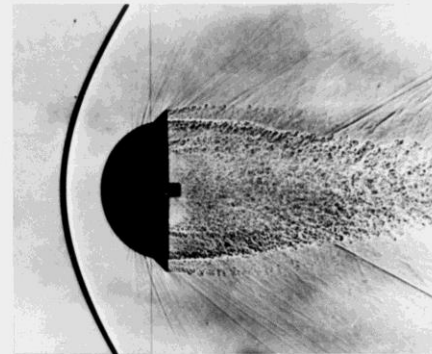


**Airplane thermal image**  
<http://www.blackroc.com/wp-content/uploads/2016/03/thermal-image.jpg>

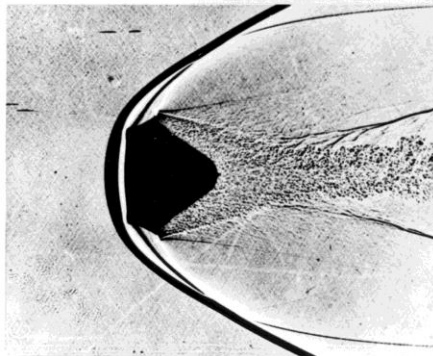
## RESEARCH CONTRIBUTING TO PROJECT MERCURY



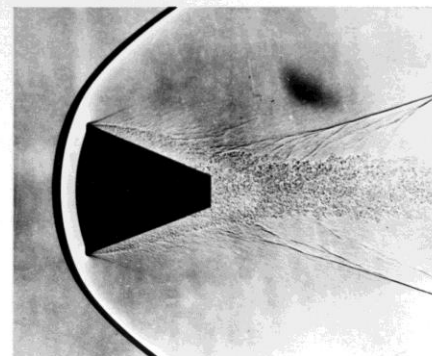
**INITIAL CONCEPT**



**BLUNT BODY CONCEPT 1953**



**MISSILE NOSE CONES 1953-1957**



**MANNED CAPSULE CONCEPT 1957**

### Shadowgraph Images of Re-entry Vehicles

Photo credit: NASA on the Commons.

<https://www.flickr.com/photos/nasacommons/>

## Compressible flows – Starting equations

$$\text{Exact NSE} \left\{ \begin{array}{l} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \mathbf{S}_u \\ \frac{\partial (\rho e_t)}{\partial t} + \nabla \cdot (\rho e_t \mathbf{u}) = -\nabla \cdot \mathbf{q} - \nabla \cdot (p \mathbf{u}) + \boldsymbol{\tau} : \nabla \mathbf{u} + \mathbf{S}_{e_t} \end{array} \right.$$

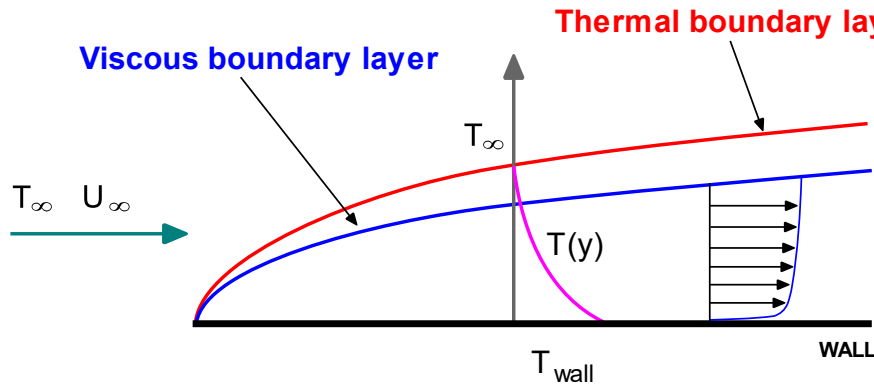
+

Equation of state and thermodynamics relations (thermophysical models)

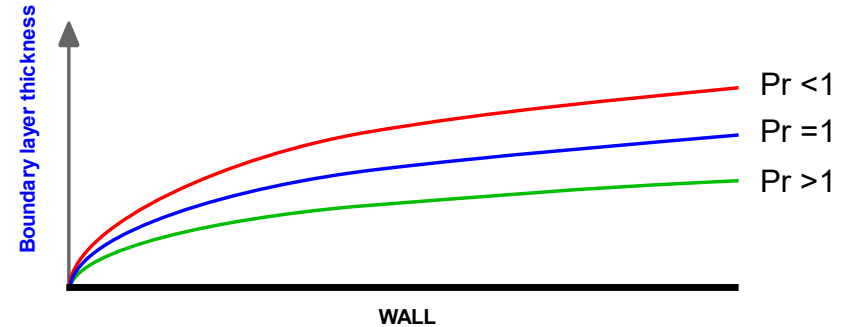
+

Additional closure equations for turbulence models, multiphase models, combustion, particles, source terms, and so on

## Compressible flows – Boundary layer



Thermal boundary layer vs. Viscous boundary layer  
Forced convection



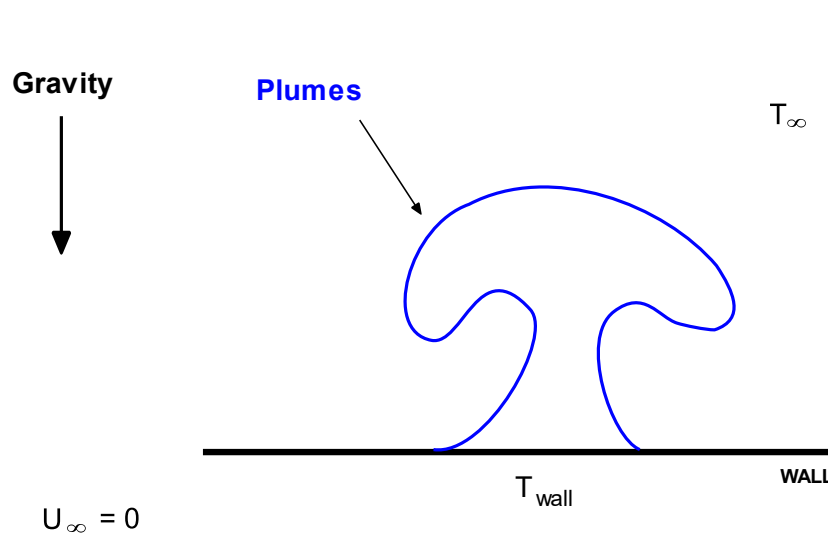
Thermal boundary layer in function of Prandtl number (Pr)

## Momentum and thermal boundary layer

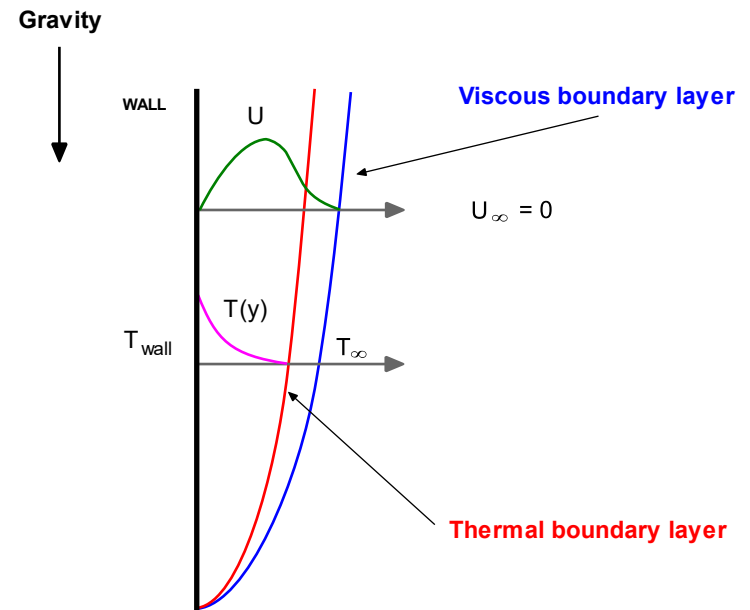
- Just as there is a viscous (or momentum) boundary layer in the velocity distribution, there is also a thermal boundary layer.
- Thermal boundary layer thickness is different from the thickness of the viscous sublayer (or momentum) and is fluid dependent.
- The thickness of the thermal sublayer for a high Prandtl number fluid (e.g., water) is much less than the momentum sublayer thickness.
- For fluids of low Prandtl numbers (e.g., air), it is much larger than the momentum sublayer thickness.
- For Prandtl number equal 1, the thermal boundary layer is equal to the momentum boundary layer.



## Compressible flows – Boundary layer



Horizontal heated plate immersed in a quiescent fluid.  
Natural convection



Vertical heated plate immersed in a quiescent fluid.  
Natural convection.

## Natural convection in a heated plate

- As the fluid is warmed by the plate, its density decreases, and a buoyant force arises which induces flow motion in the vertical or horizontal direction.
- The force is proportional to  $(\rho - \rho_\infty) \times g$ , therefore gravity must be considered.

## Compressible solvers in OpenFOAM®

- Dealing with compressible flows in OpenFOAM® is not so different from what we have done so far.
- The new steps are:
  - Define the thermophysical variables.
  - Define the boundary conditions and initial conditions for temperature.
  - If you are dealing with turbulence (most of the times), you will need to define the boundary conditions and initial conditions for the turbulent thermal diffusivity.
    - Do not forget to choose the near-wall treatment.
  - Depending on the thermophysical model and physics involved, you will need to define discretization schemes and linear solvers for the new variables and equations, that is, **T**, **h**, **e** and so on.
  - Define solver parameters for the new variables, that is, under-relaxation factors, **SIMPLE/PISO/PIMPLE** corrections, maximum and minimum allowable pressure or density values, high-speed corrections, and so on.
  - Remember, as for pressure, mesh non-orthogonality and skewness also introduces secondary gradients in the energy equation,  $f(\nabla T)$ .
    - In general, every equation that have a Laplacian on it, will show this dependence on mesh quality.

## Compressible solvers in OpenFOAM®

- Additionally, the numerics of compressible solvers is a little bit more delicate.
  - Temperature is a bounded quantity, so we need to use accurate and stable methods (preferably TVD).
  - If you are in the presence of shock waves or strong discontinuities, you need to use TVD methods and gradient limiters.
  - The solvers are very sensitive to overshoots and undershoots of the gradients, so you need to use aggressive limiters.
  - If you are dealing with chemicals reactions or combustion, you need to use accurate and stable methods (preferably TVD).
  - TVD methods requires good meshes and CFL number below 1 for good accuracy and stability.
  - Using steady solvers requires tuning of the under-relaxation factors. Usually, the default values do not work well.
  - The use of local time stepping to reach steady state can improve the convergence rate.
- We have found that it is tricky to achieve good convergence using a low-RE approach with steady solvers in high-speed compressible flows.

## Compressible solvers in OpenFOAM®

- OpenFOAM® comes with many solvers and models that can address a wide physics.
- Compressibility can be introduced in all the modeling capabilities we have seen so far (turbulence modeling and multiphase flows).
- It is also possible to add source terms, deal with moving bodies, or use adaptive mesh refinement.
- You will find the source code of all the compressible solvers in the directories:
  - `OpenFOAM-9/applications/solvers/compressible`
  - `OpenFOAM-9/applications/solvers/combustion`
  - `OpenFOAM-9/applications/solvers/heatTransfer`
  - `OpenFOAM-9/applications/solvers/lagrangian`
  - `OpenFOAM-9/applications/solvers/multiphases`
- You will find the source code of the thermophysical models in the directory:
  - `OpenFOAM-9/src/thermophysicalModels`
  - `OpenFOAM-9/src/ThermophysicalTransportModels`

## Compressible solvers in OpenFOAM®

- These are the compressible solvers that you will use most of the time in OpenFOAM®.
  - HVAC and low-speed aerodynamics:
    - rhoSimpleFoam, rhoPimpleFoam
  - High-speed aerodynamics:
    - rhoSimpleFoam, rhoPimpleFoam
    - rhoCentralFoam (this solver is explicit)
  - Buoyancy driven flows (including Boussinesq approximation):
    - buoyantSimpleFoam, buoyantPimpleFoam
  - Conjugate heat transfer
    - chtMultiRegionFoam

## Selecting thermophysical properties

```
1 thermoType
2 {
3     type          hePsiThermo;
4     mixture       pureMixture;
5     transport      const;
6     thermo         hConst;
7     equationOfState perfectGas;
8     specie         specie;
9     energy         sensibleEnthalpy;
10 }
11
12 mixture
13 {
14     specie
15     {
16         molWeight 28.9;
17     }
18     thermodynamics
19     {
20         Cp        1005;
21         Hf        0;
22     }
23     transport
24     {
25         mu        0;
26         Pr        0.713;
27     }
28 }
```

- The thermophysical properties are set in the dictionary *thermophysicalProperties*.
- This dictionary file is located in the directory **constant**.
- Thermophysical models are concerned with energy equation formulation and physical properties.
- In the sub-dictionary **thermoType** (lines 1-10), we define the thermophysical models.
- The entries in lines 3-4, are determined by the choice of the solver (they are hardwired to the solver).
- The **transport** keyword (line 5). concerns evaluating dynamic viscosity. In this case the viscosity is constant.
- The thermodynamic models (**thermo** keyword) are concerned with evaluating the specific heat Cp (line 6). In this case Cp is constant.
- The **equationOfState** keyword (line 7) concerns to the equation of state of the working fluid. In this case,

$$\rho = \frac{p}{RT}$$

- Line 8 is a fixed option (hardwired to the solver).

## Selecting thermophysical properties

```
1 thermoType
2 {
3     type            hePsiThermo;
4     mixture         pureMixture;
5     transport        const;
6     thermo           hConst;
7     equationOfState perfectGas;
8     specie           specie;
9     energy           sensibleEnthalpy;
10 }
11
12 mixture
13 {
14     specie
15     {
16         molWeight    28.9;
17     }
18     thermodynamics
19     {
20         Cp           1005;
21         Hf           0;
22     }
23     transport
24     {
25         mu           0;
26         Pr           0.713;
27     }
28 }
```

- The form of the energy equation to be used is specified in line 9 (**energy**).
- In this case we are using enthalpy formulation (**sensibleEnthalpy**).
- In this formulation, the following equation is solved,

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) - \frac{\partial p}{\partial t} = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

- Alternatively, we can use the **sensibleInternalEnergy** formulation, where the following equation is solved for the internal energy,

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho \mathbf{u} e) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) + \nabla \cdot (\mathbf{u} p) = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

- In the previous equations, the effective thermal diffusivity is equal to,

$$\alpha_{eff} = \alpha_{turbulent} + \alpha_{laminar} = \frac{\rho \nu_t}{Pr_t} + \frac{\mu}{Pr} = \frac{\rho \nu_t}{Pr_t} + \frac{k}{c_p}$$

- And  $K \equiv |\mathbf{u}|^2/2$  is the kinetic energy per unit mass.

## Selecting thermophysical properties

```
1 thermoType
2 {
3     type          hePsiThermo;
4     mixture       pureMixture;
5     transport      const;
6     thermo         hConst;
7     equationOfState perfectGas;
8     specie         specie;
9     energy         sensibleEnthalpy; ←
10 }
11
12 mixture
13 {
14     specie
15     {
16         molWeight 28.9;
17     }
18     thermodynamics
19     {
20         Cp        1005;
21         Hf        0; ←
22     }
23     transport
24     {
25         mu        0;
26         Pr        0.713;
27     }
28 }
```

- When we use the sensible formulation (**sensibleEnthalpy** or **sensibleInternalEnergy**), the heat of formation is not included in the energy equation.
- If you want to include the heat of formation, you must use the **absoluteEnthalpy** or **absoluteInternalEnergy** formulations.



## Selecting thermophysical properties

```
1 thermoType
2 {
3     type          hePsiThermo;
4     mixture        pureMixture;
5     transport      const; ←
6     thermo          hConst;
7     equationOfState perfectGas;
8     specie          specie;
9     energy          sensibleEnthalpy;
10 }
11
12 mixture
13 {
14     specie
15     {
16         molWeight 28.9;
17     }
18     thermodynamics
19     {
20         Cp         1005;
21         Hf         0;
22     }
23     transport
24     {
25         mu         0;
26         Pr         0.713;
27     }
28 }
```

- In the sub-dictionary **mixture** (lines 12-28), we define the thermophysical properties of the working fluid.
- In line 16, we define the molecular weight.
- In line 20, we define the specific heat.
- The heat of formation is defined in line 21 (not used in this case).
- In this case, we are defining the properties for air at 20° Celsius and at a sea level.
- As we are using the transport model **const**, we need to define the dynamic viscosity and Prandtl number (lines 25 and 26).
- If you set the viscosity to zero, you solve the Euler equations.
- Remember, transport modeling (line 5), concerns evaluating dynamic viscosity, thermal conductivity and thermal diffusivity.
  - In this case, all these properties are constant.

## Selecting thermophysical properties

```
1  thermoType
2  {
3      type          hePsiThermo;
4      mixture       pureMixture;
5      transport      sutherland; ←
6      thermo         hConst;
7      equationOfState perfectGas;
8      specie         specie;
9      energy         sensibleEnthalpy;
10 }
11
12 mixture
13 {
14     specie
15     {
16         molWeight  28.9;
17     }
18     thermodynamics
19     {
20         Cp         1005;
21         Hf         0;
22     }
23     transport
24     {
25         As         1.4792e-06;
26         Ts         116;
27     }
28 }
```

- If you use the transport model **sutherland** (line 5), you will need to define the coefficients of the Sutherland model.
- The Sutherland model is defined as follows (OpenFOAM® uses the 2 coefficients formulation):

$$\mu = \frac{A_s \sqrt{T}}{1 + T_s/T}$$

- The Sutherland coefficients are defined in lines 25-26.

- $A_s$       1.4792e-06;
- $T_s$       116;

- Remember, you can use the banana method to know all the options available.



## Adjusting the numerical method

- If you choose the **sensibleEnthalpy** formulation, you need to define the convective discretization schemes and linear solvers of the energy equation (enthalpy formulation).

*fvSchemes*

### divSchemes

```
{  
    div(phi,K) Gauss linearUpwind default;  
    div(phi,h) Gauss linearUpwind default;  
    div(phid,p) Gauss linearUpwind default;  
    div(phi,(p|rho)) Gauss linearUpwind default;  
    ...  
    ...  
    ...  
}
```

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) - \frac{\partial p}{\partial t} = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

*fvSolution*

### “(h|rho)”

```
{  
    solver          PBiCGStab;  
    preconditioner  DILU;  
    tolerance       1e-8;  
    relTol          0.01;  
    ...  
    ...  
    ...  
}
```

- Remember, temperature is a bounded quantity, so you need to use non-oscillatory methods.
- For low-speed flows, the kinetic energy **K** and the enthalpy **h** can be discretized using the linear method. For high-speed flows, is better to use bounded methods.
- Remember to use gradient limiters.
- If you are using a steady solver, remember to set the under-relaxation factors for **h** and **rho**.

## Adjusting the numerical method

- If you choose the **sensibleInternalEnergy** formulation, you need to define the convective discretization schemes and linear solvers of the energy equation (internal energy formulation).

*fvSchemes*

**divSchemes**

```
{  
    div(phi,K) Gauss linearUpwind default;  
    div(phi,e) Gauss linearUpwind default;  
    div(phid,p) Gauss linearUpwind default;  
    div(phi,(p|rho)) Gauss linearUpwind default;  
    ...  
    ...  
    ...  
}
```

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho \mathbf{u} e) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) + \nabla \cdot (\mathbf{u} p) = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

*fvSolution*

**“(e|rho)”**

```
{  
    solver          PBiCGStab;  
    preconditioner  DILU;  
    tolerance       1e-8;  
    relTol          0.01;  
    ...  
    ...  
    ...  
}
```

- Remember, temperature is a bounded quantity, so you need to use non-oscillatory methods.
- For low-speed flows, the kinetic energy **K** and the internal energy **e** can be discretized using the linear method. For high-speed flows, is better to use bounded methods.
- Remember to use gradient limiters.
- If you are using a steady solver, remember to set the under-relaxation factors for **e** and **rho**.

## Final remarks

- When solving the enthalpy formulation of the energy equation,

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{u} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{u} K) - \frac{\partial p}{\partial t} = \nabla \cdot (\alpha_{eff} \nabla e) + \rho \mathbf{g} \cdot \mathbf{u} + S$$

the pressure work term  $\partial p / \partial t$  can be excluded from the solution.

- This has a stabilizing effect on the solution, specially if you are using steady solvers.
- To turn off the pressure work term  $\partial p / \partial t$ , set the option `dpdt` to `no` ( **dpdt no;** ) in the *thermophysicalProperties* dictionary.
- By default, the term `dpdt` is enabled (hardwired in the solver).
- Finally, when you work with compressible solvers you use absolute pressure, and the working units are in Pascals.



# Compressible flows hands-on tutorials

- 2D supersonic cylinder – Shock waves
- Let us run this case. Go to the directory:

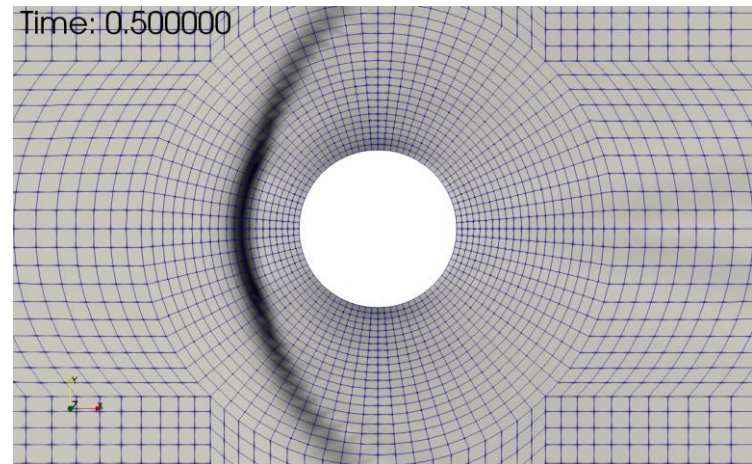
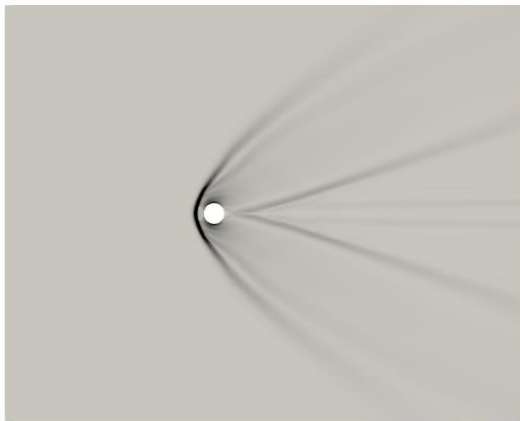
```
$PTOFC/advanced_physics/compressible/supersonic_cyl
```

- 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.

# Compressible flows hands-on tutorials

## 2D supersonic cylinder – Shock waves

Time: 0.500000



Shock wave visualization using numerical Schlieren (density gradient)

- Shock waves are strong discontinuities that need to be treated using high resolution schemes.
- Additionally, the non-orthogonality add extra complications to this problem.

# Compressible flows hands-on tutorials

## 2D supersonic cylinder – Shock waves

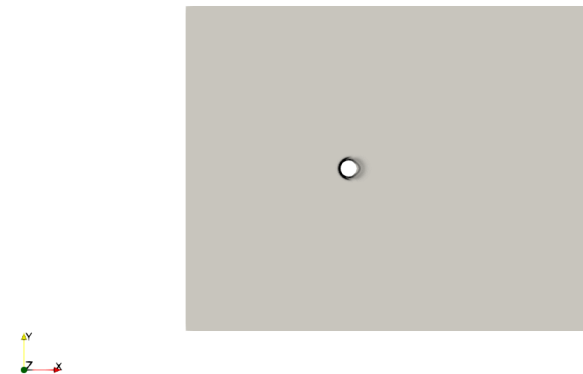
Time: 0.001000

Time: 0.001000



Mach number contours

<http://www.wolfdynamics.com/training/compressible/image3.gif>



Schlieren contours

<http://www.wolfdynamics.com/training/compressible/image4.gif>

- The numerical Schlieren can be computed by taking the gradient of the density,

$$\nabla(\rho)$$

- The numerical shadowgraph can be computed by taking the Laplacian of the density (or the divergence of the gradient of the density)

$$\Delta(\rho) = \nabla \cdot \nabla(\rho)$$



# Compressible flows hands-on tutorials

## 2D supersonic cylinder – Shock waves

- In this case we will use the solver `rhoPimpleFoam` with transonic corrections.
- By enabling transonic correction, we can use this solver to tackle trans-sonic/supersonic flows.
- Transonic corrections are enabled in the **PIMPLE** block of the dictionary `fvSolution`,
  - **transonic yes;**
- `rhoPimpleFoam` is an unsteady solver, but if you are interested in a steady solution, you can use local time stepping.
- As the flow is compressible, we need to define the thermodynamical properties of the working fluid.
- This is done in the dictionary `constant/thermophysicalProperties`.
- We also need to define the boundary conditions and initial conditions for the temperature field.
- Additionally, if you are using a turbulence model, you will need to define wall functions for the thermal diffusivity.
  - The rest of the turbulent variables are defined as in incompressible flows.
- Finally, adjust the numerics according to your physics.

# Compressible flows hands-on tutorials

## 2D supersonic cylinder – Shock waves

```
18     limitp
19     {
20         type limitPressure;

22         minFactor 0.1;
23         maxFactor 10;
27     }

30     limitT
31     {
32         type limitTemperature;

34         min 100;
35         max 1000;

37         selectionMode all;
38     }

40     limitU
41     {
42         type limitVelocity;

44         selectionMode all;

47         max 1000;
48     }
```

- In the directory **system**, you will find the input file *fvConstraints*.
- This input file is optional and run-time modifiable.
- In this dictionary, we can define constraints to pressure, temperature, and velocity magnitude, among many options.
- These constraints are used to avoid exceeding some predefined values (usually a minimum and a maximum value).
- Constraints are often useful to ensure numerical stability, particularly during the initial start-up period of a simulation.
- But be careful, if you use values too conservative or unrealistic, you may be forcing the solution to an unrealistic behavior.
- You can use constraints with any solver.
- You can find the source code in the directory,
  - **OpenFOAM-9/src/fvConstraints**

# Compressible flows hands-on tutorials

## 2D supersonic cylinder – Shock waves

```
18  limitp
19  {
20      type limitPressure;

22      minFactor 0.1;
23      maxFactor 10;
27  }

30  limitT
31  {
32      type limitTemperature;

34      min 100;
35      max 1000;

37      selectionMode all;
38  }

40  limitU
41  {
42      type limitVelocity;

44      selectionMode all;

47      max 1000;
48  }
```

- In lines 18-27, we are defining a constraint for the absolute pressure.
  - In this case, the minimum and maximum values in the domain must be bounded between 0.1 and 10 times the minimum and maximum values of the absolute pressure, such as,

$$p_{min} = \max(p, p \times pMinFactor)$$

$$p_{max} = \min(p, p \times pMaxFactor)$$

- You can also define the minimum (min) and maximum (max) values of the absolute pressure instead of using minFactor and maxFactor as follows,
  - min = 10000;
  - max = 500000;

# Compressible flows hands-on tutorials

## 2D supersonic cylinder – Shock waves

```
18     limitp
19     {
20         type limitPressure;

22         minFactor 0.1;
23         maxFactor 10;
27     }

30     limitT
31     {
32         type limitTemperature;

34         min 100;
35         max 1000;

37         selectionMode all;
38     }

40     limitU
41     {
42         type limitVelocity;

44         selectionMode all;

47         max 1000;
48     }
```

- In lines 30-38, we are defining a constraint for the temperature.
  - In this case, the minimum and maximum values in the domain cannot be larger than 100 and 1000, respectively.
  - Notice that you can apply this constraint to a cell selection in the domain.
- In lines 40-48, we are defining a constraint for the velocity magnitude.
  - In this case, the velocity magnitude cannot be larger than 1000.
  - Notice that you can apply this constraint to a cell selection in the domain.

# Roadmap

## A crash introduction to:

- ~~1. Turbulence modeling in OpenFOAM®~~
- ~~2. Multiphase flows modeling in OpenFOAM®~~
- ~~3. Compressible flows in OpenFOAM®~~
- 4. Source terms in OpenFOAM®**
- ~~5. Scalar transport pluggable solver~~
- ~~6. Moving bodies in OpenFOAM®~~

# A crash introduction to source terms OpenFOAM®

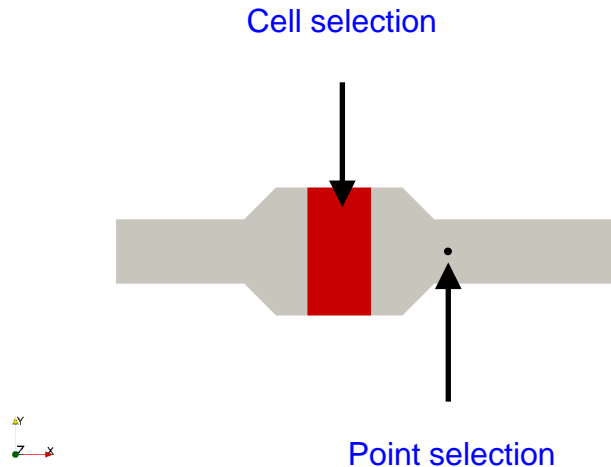
## Source terms in OpenFOAM®

- In addition to all modeling capabilities we have seen so far, you can also add source terms to the governing equations without the need of modifying the original source code.
- This functionality is provided via the dictionary *fvModels*, which is located in the directory **constants**.
- There are many source terms implemented in OpenFOAM®, you can even use **codeStream** to program your own source term without the need of recurring to high level programming.
- The *fvModels* functionality work with most of the solvers that deal with advanced modeling capabilities.
- Remember, the solver *icoFoam* is very basic with no modeling capabilities. Therefore, you can not use the *fvModels* functionality and many other modeling and postprocessing capabilities with it.
- You will find the source code of the source terms in the directory:
  - **OpenFOAM-9/src/fvConstraints**
  - **OpenFOAM-9/src/fvModels**

# A crash introduction to source terms OpenFOAM®

## Source terms in OpenFOAM®

- To use source terms, you must first select where you want to apply it
- You can apply a source term in the whole domain, a set of cells, a cell zone, or a point (or group of points).
- You can create the set of cells at meshing time, or you can use the utility `topoSet` to select a group of cells.
- By the way, boundary conditions are source terms added at the boundary patches, and MRF are source terms added to a cell selection.



# A crash introduction to source terms OpenFOAM®

## Source terms in OpenFOAM®

```
1 user_defined_name
2 {
3     type          name_of_source_term;
4
5     selectionMode points; ←
6     points
7     (
8         (0 0 0) ←
9     );
10
11     volumeMode absolute; ←
12
13     source_term_options
14     {
15         ...
16         ...
17         ...
18     } ←
19 }
```

- The source terms can be selected in the dictionary *fvModels*, and they can be modified on-the-fly.
- This dictionary file is located in the directory **constant**.
- Hereafter we show a generic *fvModels* dictionary.
- According to the selected source term, you will need to give different input parameters in the source term options section (lines 13-16).
  - These options are specific to each source term.
- The input parameters indicated with an arrow can be used with any source term.
- The **volumeMode** keyword (line 11) let you choose between absolute and specific.
  - **absolute**: input values are given as quantity/volume.
  - **specific**: input values are given as quantity.
- Remember, you can use the banana method to know all source terms and options available.
- You can also read the source code.



# Source terms hands-on tutorials

## Momentum source term

*constant/fvModels*

```
17 momentumSource1
18 {
19     type semiImplicitSource;

28     selectionMode points;
30     points
31     (
32         (2.85 0.5 0)
33     );

35     volumeMode absolute;

38     sources
39     {
41         U
42         {
45             explicit
46             {
54                 type scale;
55                 scale squarePulse;
56                 start 1;
57                 duration 6;
59                 value (0.5 0.5 0)
61             }

63             implicit 0;
64         }
82     }
84 }
```

- The source terms can be selected in the dictionary *fvModels*, and they can be modified on-the-fly.
- In this case, we are using the source term **semiImplicitSource** (line 19).
- The source term is used in a point selection (lines 28-33).
- In line 35 we select **absolute** volume more (input values are given as quantity/volume).
- In lines 38-84, we define the input parameters of the model.
  - In line 41 we apply the source term to the field variable **U**.
  - The explicit contribution is defined in lines 45-61, and the implicit contribution is defined in line 63
  - In lines 54-59, we define the start time (line 56) and the duration (line 57) of the explicit contribution (line 59), using a function of type **squarePulse** (lines 54-55).
  - If you do not want to use a ramp function for the source term, just define the source contribution as,
    - explicit (0.5 0.5 0);

# Source terms hands-on tutorials

- Momentum source term
- Let us run this case. Go to the directory:

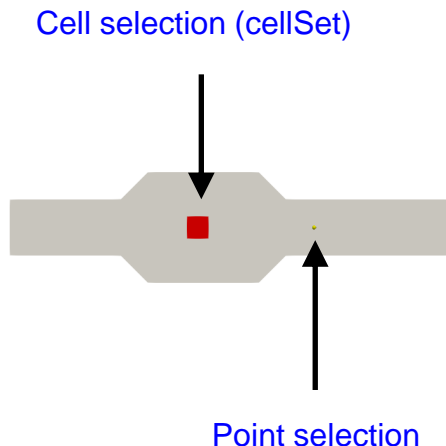
```
$PTOFC/advanced_physics/source_terms_passive_scalars/momentum_source
```

- 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.

# Source terms hands-on tutorials

## Momentum source term

- In this case we are going to use the source term **semImplicitSource**.
- This source term will add a momentum source to the governing equations
- Using this source term, we can apply a momentum source term in the whole mesh or in a selection (cells or points).
- The source terms can be activated after a given time and can have a specific duration.



Time: 0.000000



<http://www.wolfdynamics.com/training/sourceterms/ani1.gif>

# Source terms hands-on tutorials

## Momentum source term

*constant/fvModels*

```
17 momentumSource1
18 {
19     type semiImplicitSource;

28     selectionMode points;
30     points
31     (
32         (2.85 0.5 0)
33     );

35     volumeMode absolute;

38     sources
39     {
41         U
42         {
45             explicit
46             {
54                 type scale;
55                 scale squarePulse;
56                 start 1;
57                 duration 6;
59                 value (0.5 0.5 0)
61             }

63             implicit 0;
64         }
82     }
84 }
```

- The source terms can be selected in the dictionary *fvModels*, and they can be modified on-the-fly.
- In this case, we are using the source term **semiImplicitSource** (line 19).
- The source term is used in a point selection (lines 28-33).
- In line 35 we select **absolute** volume more (input values are given as quantity/volume).
- In lines 38-84, we define the input parameters of the model.
  - In line 41 we apply the source term to the field variable **U**.
  - The explicit contribution is defined in lines 45-61, and the implicit contribution is defined in line 63
  - In lines 54-59, we define the start time (line 56) and the duration (line 57) of the explicit contribution (line 59), using a function of type **squarePulse** (lines 54-55).
  - If you do not want to use a ramp function for the source term, just define the source contribution as,
    - explicit (0.5 0.5 0);

# Source terms hands-on tutorials

## Momentum source term

*constant/fvModels*

```
86 momentumSource2
87 {
88     type semiImplicitSource;

90     selectionMode cellSet;
91     cellSet filter;

93     volumeMode absolute;

96     sources
97     {
98         U
99         {
102             explicit
103             {
104                 type scale;
105                 scale squarePulse;
106                 start 0;
107                 duration 3;
108                 value (-1 0 0)
109             }

111             implicit 0;
112         }
113     }
114 }
```

- The source terms can be selected in the dictionary *fvModels*, and they can be modified on-the-fly.
- In this case, we are using the source term **semiImplicitSource** (line 88).
- The source term is used in a **cellSet** selection (line 90) named **filter** (line 91).
  - This **cellSet** must be created at meshing time or using the utility `topoSet`.
- In line 93 we select **absolute** volume more (input values are given as quantity/volume).
- In lines 96-113 we define the input parameters of the source term.
  - In line 98 we apply the source term to the field variable **U**.
  - In lines 104-108, we define the start time (line 106) and the duration (line 107) of the explicit contribution (line 108), using a function of type **squarePulse** (lines 104-105).
  - If you do not want to use a ramp function for the duration, just define the source contribution as,

# Source terms hands-on tutorials

## Momentum source term

*system/topoSetDict*

```
actions
(
    // filter
    {
        name    filter;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
            box (1.5 -10 -10) (2 10 10);
        }
    }
    ...
    ...
    ...
)
```

- To create the **cellSet** used in *fvModels*, we first create a **cellSet**.
- The set of cells (**cellSet**) is constructed using the utility *topoSet*.
- This utility reads the dictionary *topoSetDict*, which is located in the directory **system**.
- Hereafter we show the dictionary inputs used to create the **cellSet** named **filter**.

Name of the selection

Select a set of cells

Create a new selection

Use a box to select the set of cells

# Source terms hands-on tutorials

## Momentum source term

*system/topoSetDict*

```
actions
(
    ...
    ...
    {
        name    filter;
        type    cellZoneSet;
        action  new;

        source  setToCellZone;
        sourceInfo
        {
            set filter;
        }
    }
)
```

Annotations:

- Name of the selection
- Select a zone set (cellZoneSet)
- Create a new selection
- Convert the cellSet filter to a cellZoneSet named filter

- From the **cellSet**, we can also create a **cellZone**.
- Now that we have the **cellSet** filter, we can convert it to a **cellZoneSet**.
- The name of the new **cellZoneSet** is **filter**.
- Remember, you can apply the source terms in a **cellSet** or in a **cellZone**.

# Source terms hands-on tutorials

## Momentum source term

- At this point, we are ready to run the simulation.
- We will use the solver `pimpleFoam`, which can use source terms.
- Before running the simulation, remember to use the utility `topoSet` to create the filter region used by the source term.
- You can visualize the region using `paraview`.
- Finally, remember to adjust the numerics according to your physics.



# Roadmap

## A crash introduction to:

- ~~1. Turbulence modeling in OpenFOAM®~~
- ~~2. Multiphase flows modeling in OpenFOAM®~~
- ~~3. Compressible flows in OpenFOAM®~~
- ~~4. Source terms in OpenFOAM®~~
- 5. Scalar transport pluggable solver**
- ~~6. Moving bodies in OpenFOAM®~~

# Scalar transport pluggable solver

## The `functionObject` `scalarTransport`

- In addition to all modeling capabilities we have seen so far, you can also add scalar transport equations (or the convection-diffusion equation), without the need of modifying the original source code.
- This functionality is provided via **`functionObjects`**, and it can be seen as a solver that can be plug into another one.
- To setup the scalar transport equation you need to:
  - Define the **`functionObject`** in the dictionary `controlDict`.
  - Add the discretization schemes and linear solvers for the new equations.
  - Define the boundary conditions and initial conditions of the transported scalars.
- Using this **`functionObject`**, you can add multiple scalars.
- You will find the source code of the scalar transport pluggable solver in the directory:
  - `OpenFOAM-9/src/functionObjects/solvers`

# Scalar transport pluggable solver

## The functionObject scalarTransport

- The scalar transport **functionObject** is defined in the dictionary *controlDict*.
- Remember, the input parameters can be modified on-the-fly.

```
scalar1
{
    type scalarTransport;
    functionObjectLibs ("libsolverFunctionObjects.so");

    enabled true;

    writeControl outputTime;

    log yes;

    nCorr 1;

    D 0;
    //alphaD 0;
    //alphaDt 0;

    field s1;

    //schemesField U;
}
```

Select scalarTransport functionObject

Number of corrector iterations.  
It is recommended to do at least one iteration.

Diffusion coefficient.  
If turbulent modeling is in use, you can define the laminar diffusion coefficient alphaD and the turbulent diffusion coefficient alphaDt

Name of the new field. You will need to select the discretization schemes and linear solvers for this field. You will also need to define the boundary conditions and initial conditions for this field.

Option to use the same numerical scheme as the one used for the field U

# Scalar transport pluggable solver

## Boundary conditions

*0/s1*

```
dimensions      [0 0 0 0 0 0 0];
internalField    uniform 0;
boundaryField
{
    walls
    {
        type      zeroGradient;
    }

    inlet
    {
        type      fixedValue;
        value      uniform 1;
    }

    outlet
    {
        type      inletOutlet;
        inletValue uniform 0;
        value      uniform 0;
    }
}
```

- Assuming that you named the new scalar *s1*, you will need to define the boundary conditions and initial conditions for the field *s1*.
- This is done in the dictionary *0/s1*.
- In this case, the scalar is entering in the patch **inlet** with a value of 1 (this is a concentration therefore it has no dimensions).
- The initial concentration of the scalar is zero.

# Scalar transport pluggable solver

## Discretization schemes and linear solvers

- Finally, and assuming that you named the new scalar `s1`, you need to define the discretization schemes and linear solvers of the new equations.
- Remember, this is a bounded quantity, so it is a good idea to use TVD schemes and gradient limiters.

*system/fvSchemes*

```
gradSchemes
{
    default      Gauss linear;
    grad(s1)     cellLimited Gauss linear 1;
}

divSchemes
{
    default      none;
    div(phi,U)   Gauss linearUpwindV default;
    div(phi,s1)  Gauss vanLeer;
}
```

*system/fvSolution*

```
s1
{
    solver       PBiCGStab;
    preconditioner DILU;
    tolerance    1e-08;
    relTol       0;
    minIter      2;
}
```

# Scalar transport pluggable solver hands-on tutorials

- Scalar transport in an elbow – Internal geometry
- Let us run this case. Go to the directory:

```
$PTOFC/advanced_physics/source_terms_passive_scalars/2Delbow_passive_scalar
```

- 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.

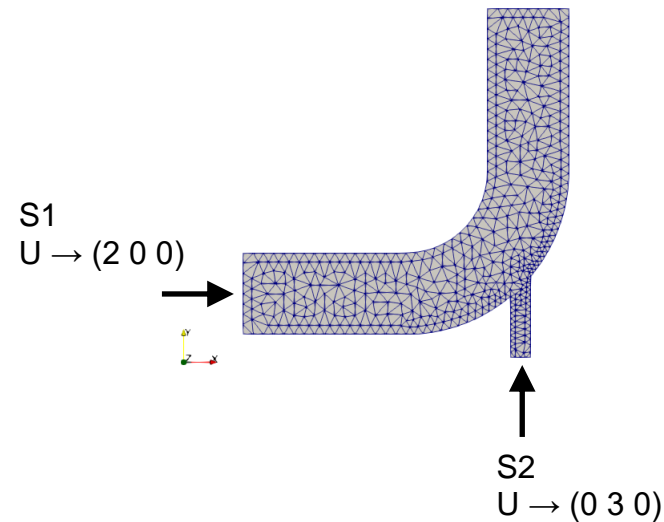
# Scalar transport pluggable solver hands-on tutorials

## Scalar transport in an elbow – Internal geometry

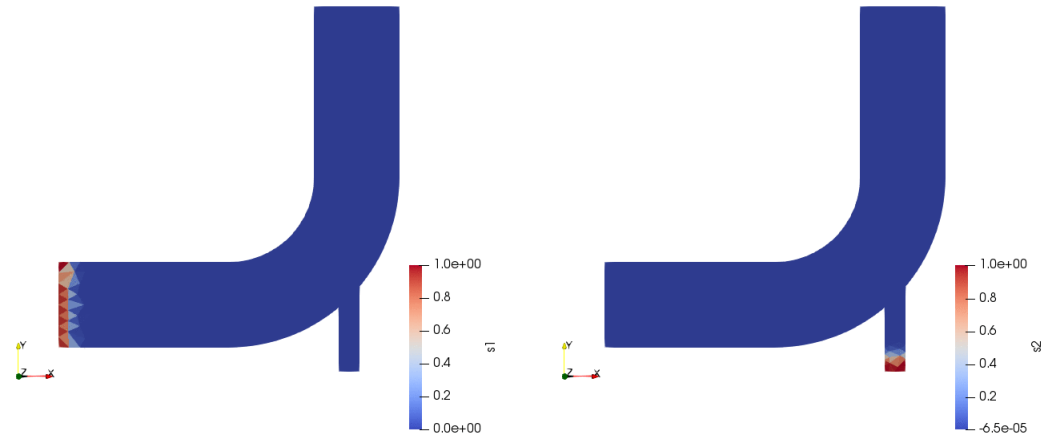
- Notice that we are adding two scalars, s1 and s2.



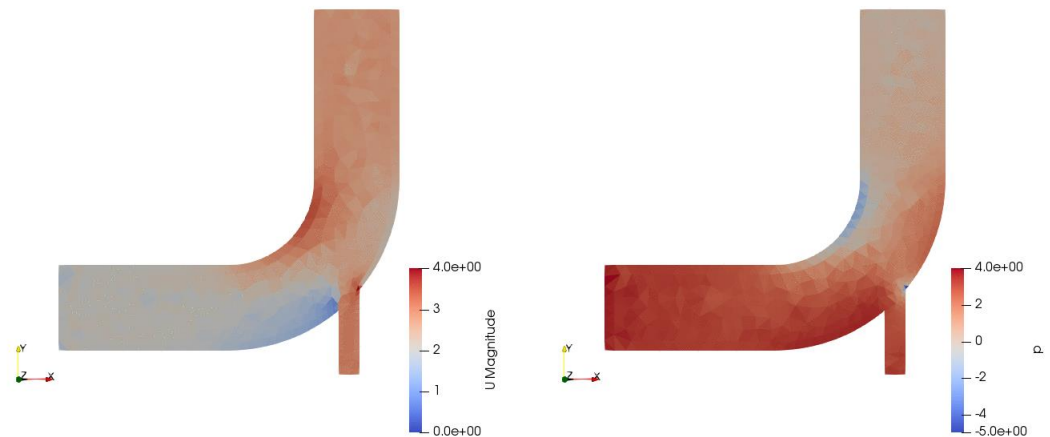
Time: 1.0



Time: 1.0



<http://www.wolfdynamics.com/training/sourceterms/image2.gif>



<http://www.wolfdynamics.com/training/sourceterms/image3.gif>

# Scalar transport pluggable solver hands-on tutorials

## Scalar transport in an elbow – Internal geometry

- At this point, we are ready to run the simulation.
- We will use the solver `pisFoam`.
- Remember to adjust the numerics according to your physics.
- Do not forget to create the boundary conditions and initial conditions of the new field variables.



# Roadmap

## A crash introduction to:

- ~~1. Turbulence modeling in OpenFOAM®~~
- ~~2. Multiphase flows modeling in OpenFOAM®~~
- ~~3. Compressible flows in OpenFOAM®~~
- ~~4. Source terms in OpenFOAM®~~
- ~~5. Scalar transport pluggable solver~~
- 6. Moving bodies in OpenFOAM®**

# A crash introduction to moving bodies OpenFOAM®

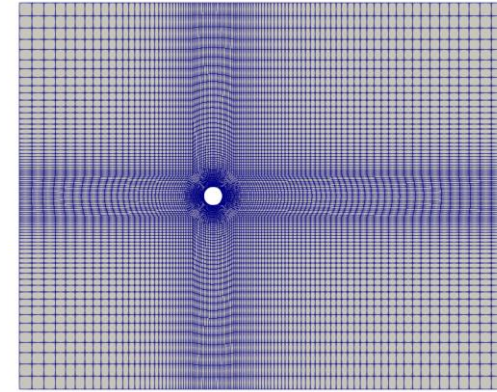
## Moving bodies in OpenFOAM® – A few examples

Time: 0.025



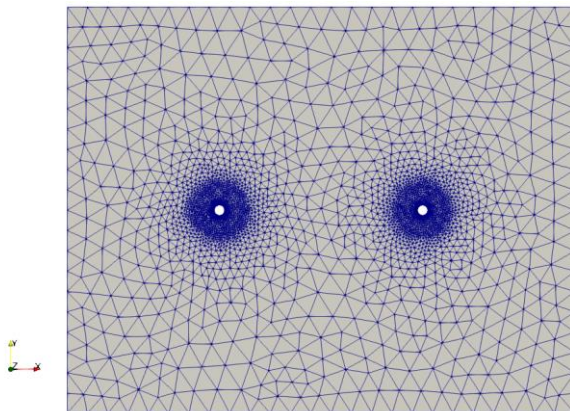
Sloshing tank

<http://www.wolfdynamics.com/training/dynamicMeshes/sloshing1.gif>



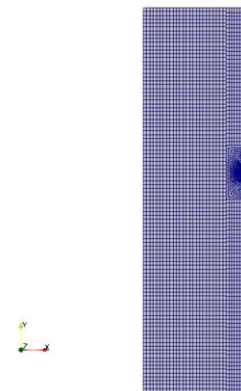
Oscillating cylinder (prescribed motion)

<http://www.wolfdynamics.com/training/dynamicMeshes/meshMotion1>



Prescribed motion with multiple bodies

<http://www.wolfdynamics.com/training/dynamicMeshes/meshMotion1>



Time: 0.001



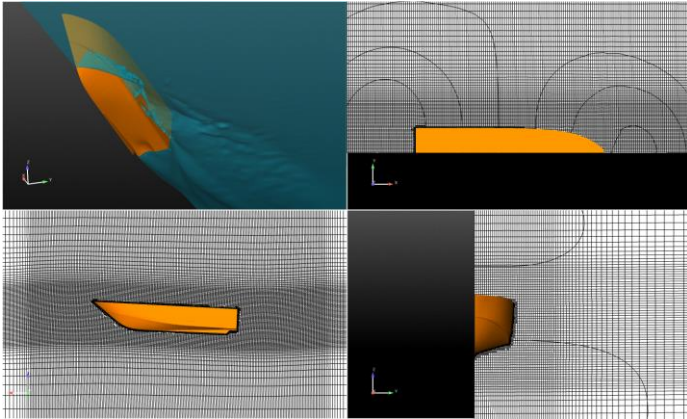
1.0e+00  
0.8  
0.6  
0.4  
0.2  
-6.5e-20  
alpha.water

Layering with mesh zones interface

<http://www.wolfdynamics.com/training/dynamicMeshes/layeringMesh.gif>

# A crash introduction to moving bodies OpenFOAM®

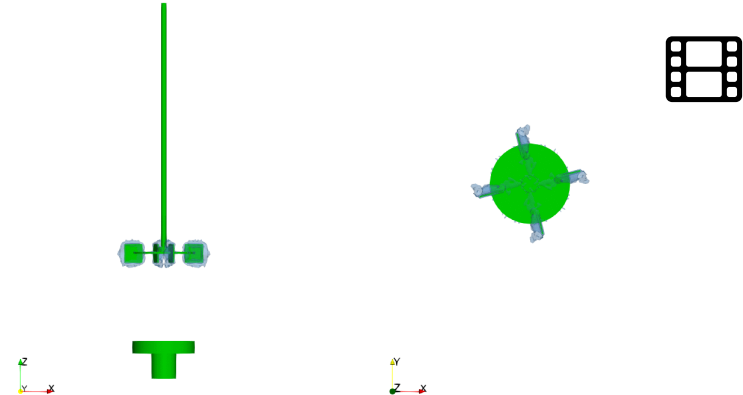
## Moving bodies in OpenFOAM® – A few examples



Sea keeping

<http://www.wolfdynamics.com/training/dynamicMeshes/seakeeping.gif>

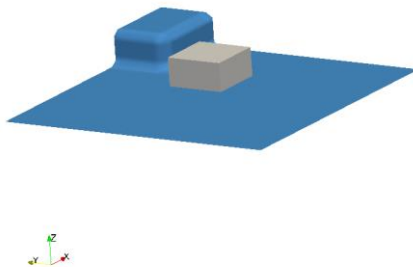
Time: 0.020



Continuous stirring tank reactor (CSTR)

<http://www.wolfdynamics.com/training/movingbodies/image13.gif>

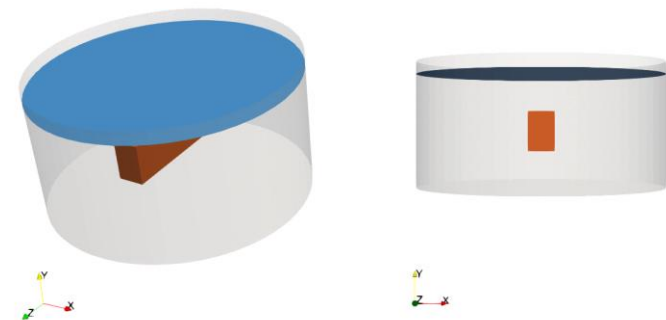
Time: 0.000000



Falling body (6 DoF)

<http://www.wolfdynamics.com/training/movingbodies/image5.gif>

Time: 0.00



VOF with sliding meshes

<http://www.wolfdynamics.com/training/mphase/image33.gif>

# A crash introduction to moving bodies OpenFOAM®

## Moving bodies in OpenFOAM®

- OpenFOAM® comes with many solvers and models that can address a wide physics.
- Moving bodies can be added to all the modeling capabilities we have seen so far (turbulence modeling, multiphase flows, and compressible flows).
- Several class of motions can be simulated in OpenFOAM®:
  - Prescribed motion.
  - Rigid body motion.
  - Sliding meshes.
  - MRF.
- Setting moving bodies simulations is not so different from what we have done so far.
- The main difference is that we must assign a motion type to a surface patch, a cell region, or the whole domain.

# A crash introduction to moving bodies OpenFOAM®

## Moving bodies in OpenFOAM®

- The mesh motion solver is selected in the dictionary *constant/dynamicMeshDict*.
- In the case of prescribed motion of a boundary patch, the motion is assigned in the dictionary *0/pointDisplacement*.
- Also, the boundary type of the moving walls must be **movingWallVelocity**, this is set in the dictionary *0/U*.
- And as usual, you will need to adjust the numerics according to your physics.
- To use the moving bodies capabilities, you will need to use solvers able to deal with dynamic meshes.
- To find which solvers work with dynamic meshes, go to the solvers directory by typing `sol` in the command line interface. Then type in the terminal:
  - `$> grep -r dynamicFvMesh.H`
- The solvers containing this header file support dynamic meshes.
- A few solvers that work with dynamic meshes: `interFoam`, `pimpleFoam`, `rhoPimpleFoam`, `buoyantPimpleFoam`.

# A crash introduction to moving bodies OpenFOAM®

## Moving bodies in OpenFOAM®

- You will find the source code of all the mesh motion libraries in the directories:
  - `OpenFOAM-9/src/dynamicFvMesh`
  - `OpenFOAM-9/src/dynamicMesh`
  - `OpenFOAM-9/src/fvMotionSolver`
  - `OpenFOAM-9/src/rigidBodyDynamics`
  - `OpenFOAM-9/src/rigidBodyMeshMotion`
  - `OpenFOAM-9/src/rigidBodyState`
  - `OpenFOAM-9/src/sixDoFRigidBodyMotion`
  - `OpenFOAM-9/src/sixDoFRigidBodyState`
- You will find the source code of the prescribed patch motion in the directory:
  - `OpenFOAM-9/src/fvMotionSolver/pointPatchFields/derived`
- You will find the source code of the restraints/constraints of rigid body motion solvers in the directory:
  - `OpenFOAM-9/src/rigidBodyDynamics/joints`
  - `OpenFOAM-9/src/rigidBodyDynamics/restraints`
  - `OpenFOAM-9/src/sixDoFRigidBodyMotion/sixDoFRigidBodyMotion`

# Moving bodies hands-on tutorials

- Continuous stirring tank reactor – Sliding meshes and MRF
- Let us run this case. Go to the directory:

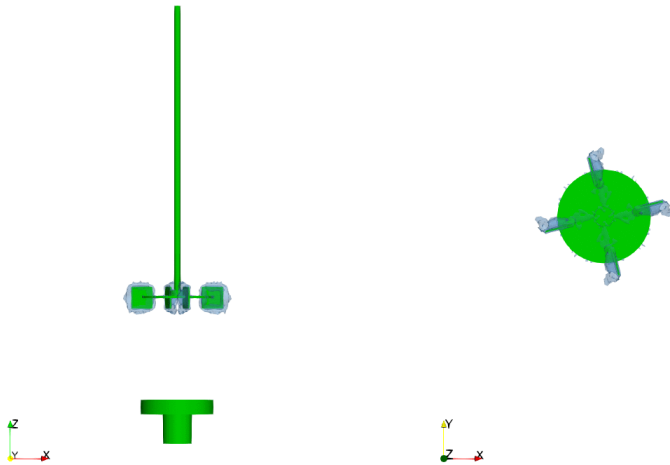
```
$PTOFC/advanced_physics/dynamic_meshes_MRF/sliding_meshes_MRF/CSTR
```

- 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.

# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF

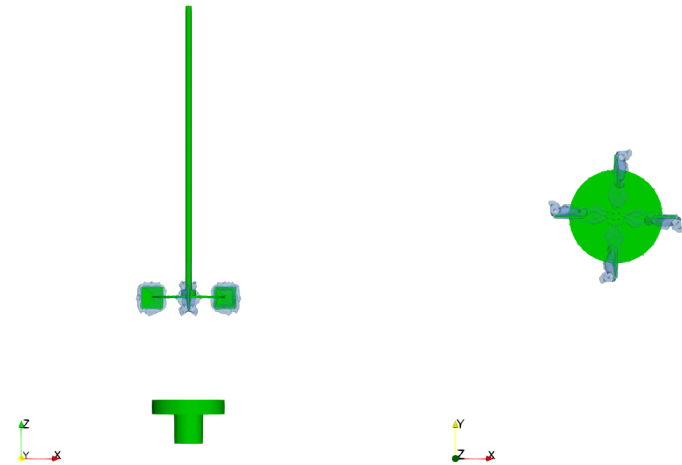
Time: 0.020



Sliding grids – Unsteady solver

<http://www.wolfdynamics.com/training/movingbodies/image13.gif>

Time: 0.020



MRF – Steady solver

<http://www.wolfdynamics.com/training/movingbodies/image14.gif>

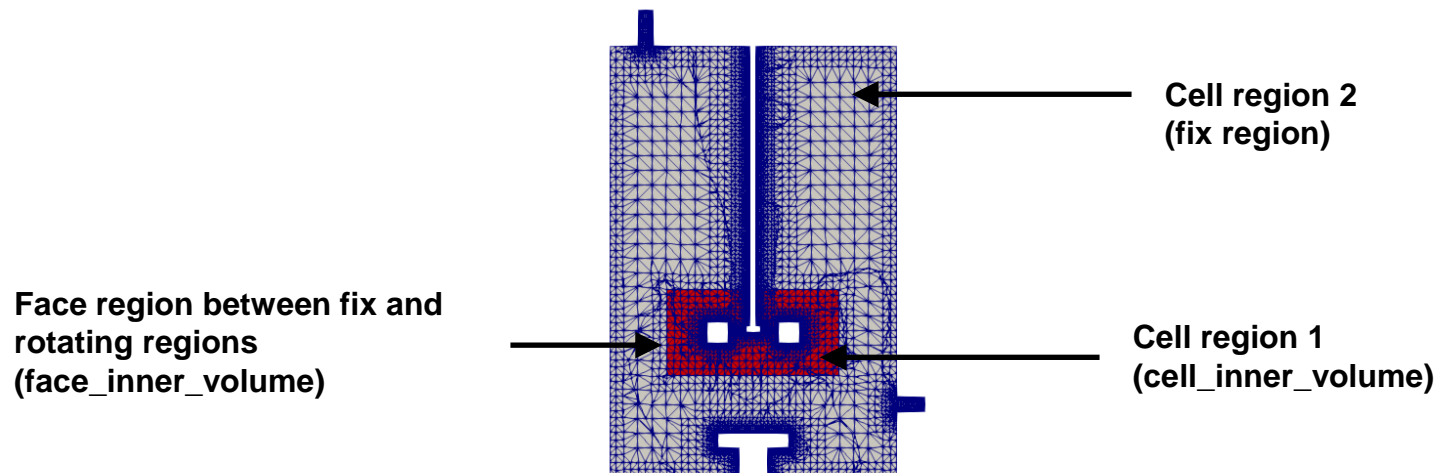




# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF

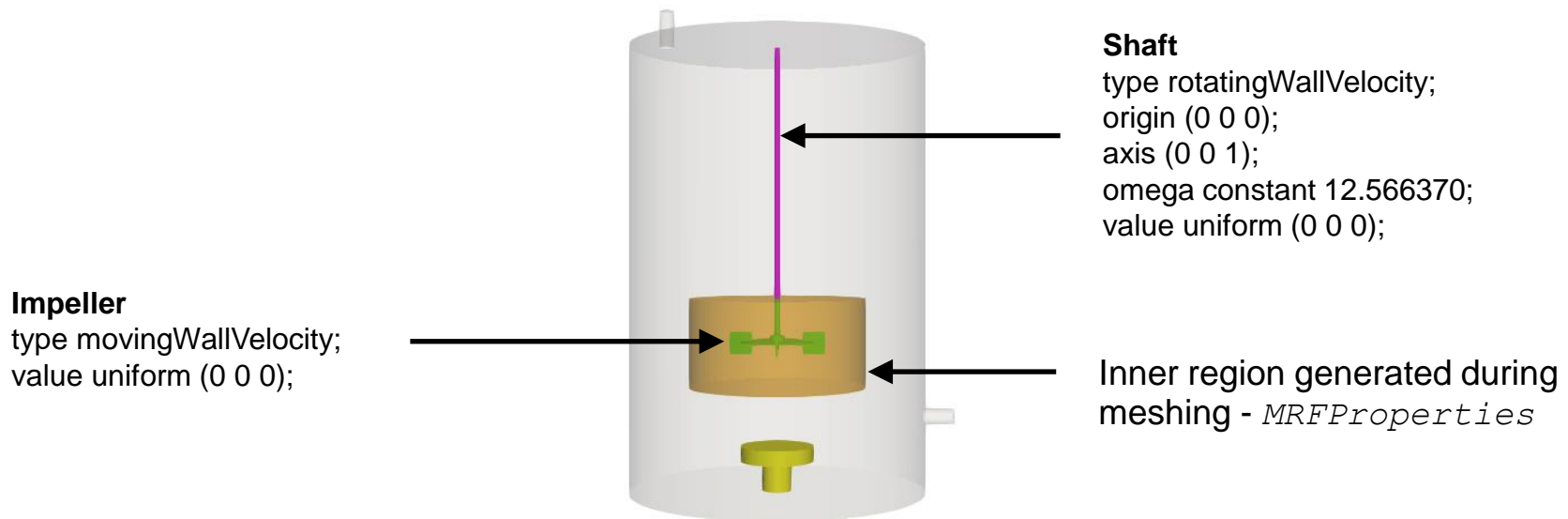
- In supplement 2, you will find a complete description on how to generate this mesh.
- We will address the differences in meshing for an MRF simulation and a sliding mesh simulation.
- In this case, at the end of the meshing stage we should have a **faceZone** and a **cellZone**.
- This is a conforming mesh, that is, the cells in the interface of the inner and outer regions are perfectly matching.
- For MRF simulations, the **cellZone** can be used to assign the MRF properties to the rotating zone.
- For sliding meshes or non-conforming meshes, there is an extra step where we need to split the mesh in two regions and create the interface patches between the fix zone and the rotating zone (the solution will be interpolated in these patches).



# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF

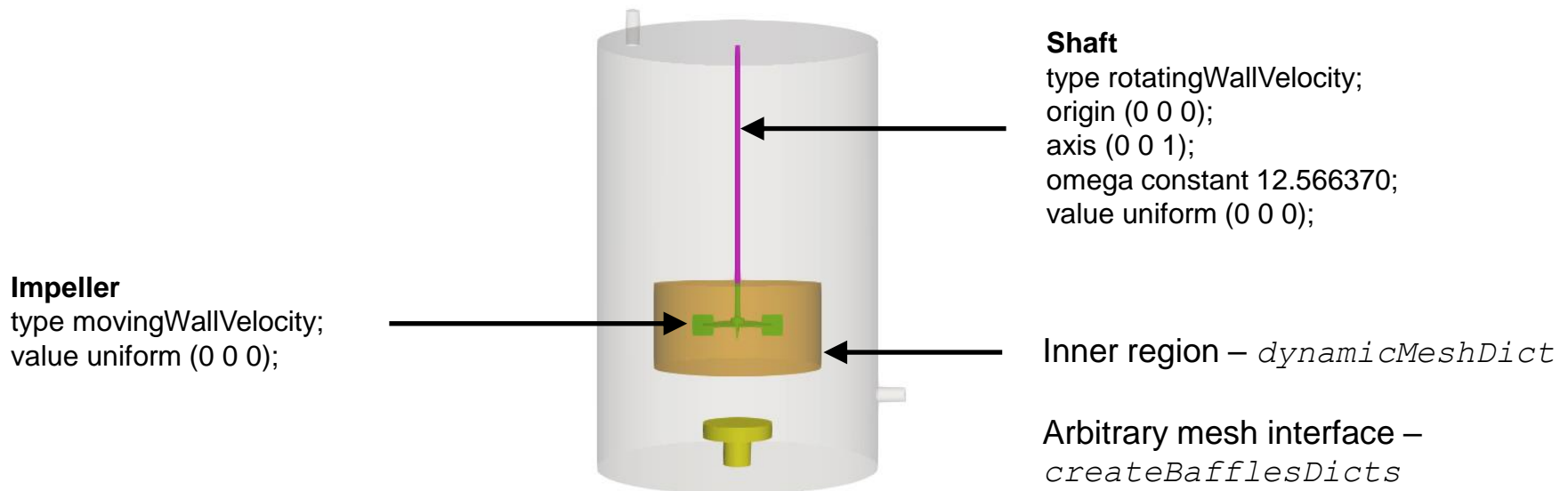
- In the MRF approach, the governing equations are solved in a relative rotating frame in the selected rotating zone.
- Additional source terms that model the rotation effect are taken into account.
- You select the rotating zone and set the rotation properties in the dictionary *constant/MRFProperties*.
- In this case, the mesh is conforming.



# Moving bodies hands-on tutorials

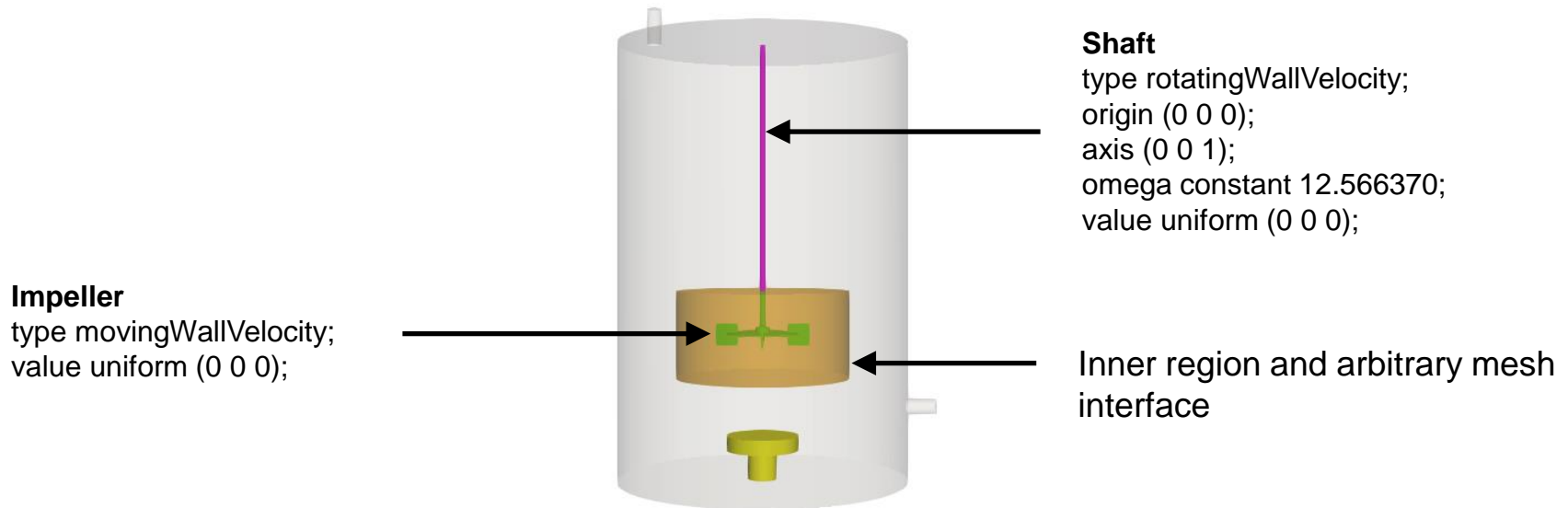
## Continuous stirring tank reactor – Sliding meshes and MRF

- In the sliding meshes approach, the selected rotating region is physically rotating.
- As the meshes are non-conforming, the solution between the rotating region and the fix region must be interpolated using arbitrary mesh interface.
- In the sliding meshes approach, is not enough to only identify the rotating region.
- We also need to create the interface patches between the fix zone and the rotating zone.



# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF



*constant/dynamicMeshDict* – For sliding meshes

```
dynamicFvMesh    dynamicMotionSolverFvMesh;  
motionSolverLibs ( "libfvMotionSolvers.so" );  
motionSolver     solidBody;  
cellZone         cell_inner_volume;  
solidBodyMotionFunction rotatingMotion;  
origin  (0 0 0);  
axis    (0 0 1);  
omega   constant 12.566370;
```

*constant/MRFProperties* – For MRF approach

```
cellZone  cell_inner_volume;  
active    yes;  
  
// Fixed patches (by default they move' with the MRF zone)  
nonRotatingPatches ();  
  
origin  (0 0 0);  
axis    (0 0 1);  
omega   constant 12.566370;
```

# Moving bodies hands-on tutorials

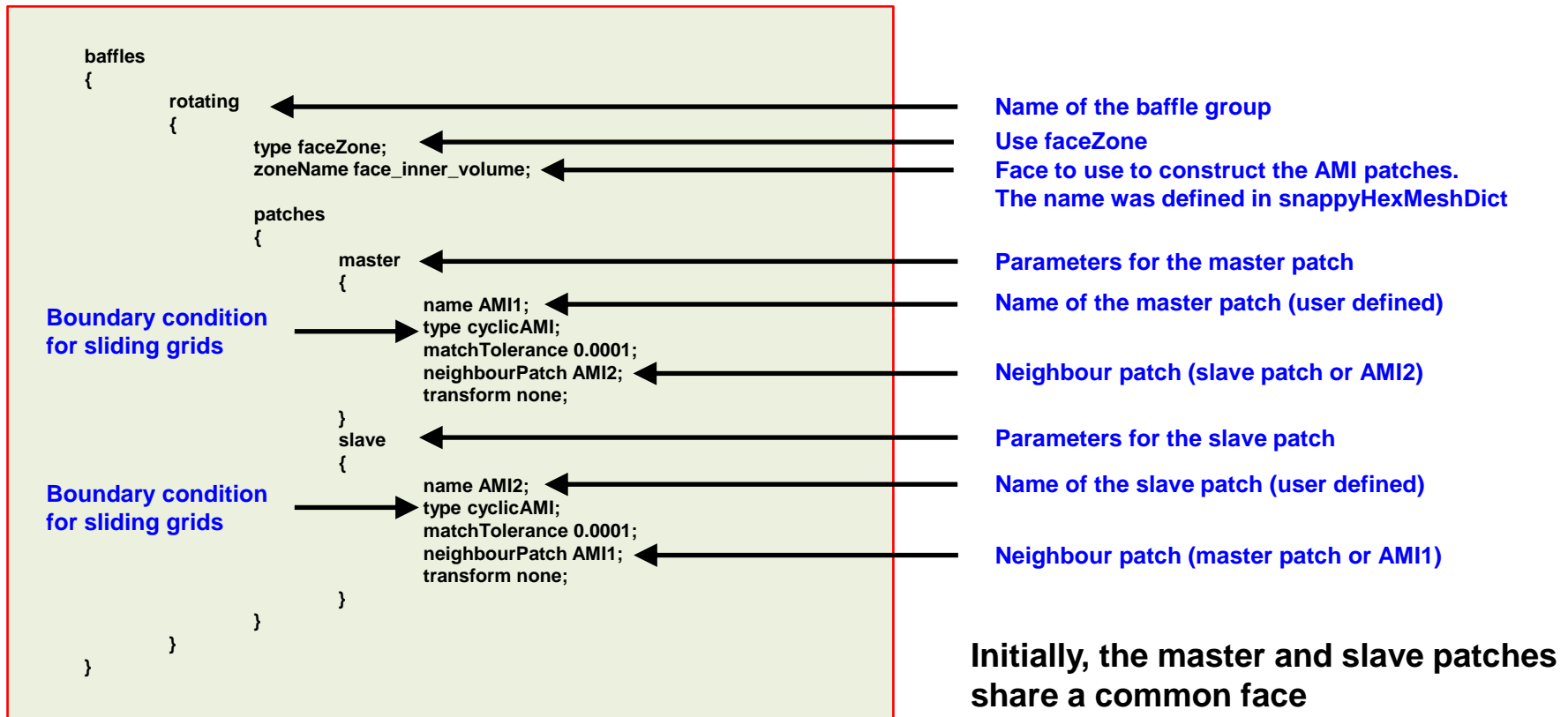
## Continuous stirring tank reactor – Sliding meshes and MRF

- For sliding meshes, we need to create separated regions.
- In this case, to create the two regions we proceed as follows,
  1. `$> createBaffles -overwrite`
  2. `$> splitBaffles -overwrite`
- In step 1, we split the mesh in regions using the baffles (**faceZone**), created during the meshing stage.
- We also create the **cyclicAMI** patches **AMI1** and **AMI2**.
- At this point we have two regions and one zone. However, the two regions are stick together via the patches **AMI1** and **AMI2**.
- In step 2, we topologically split the patches **AMI1** and **AMI2**. As we removed the link between **AMI1** and **AMI2**, the regions are free to move.

# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF

- The utility `createBaffles`, reads the dictionary `createBafflesDict`.
- With this utility we create the interface patches between the fix zone and the rotating zone.



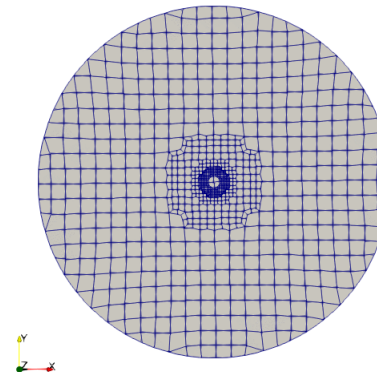
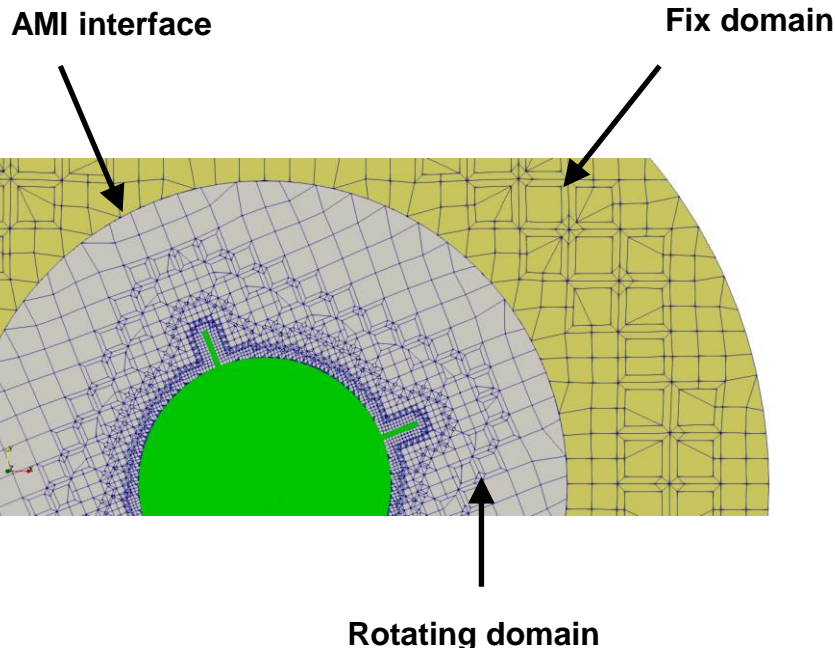
# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF

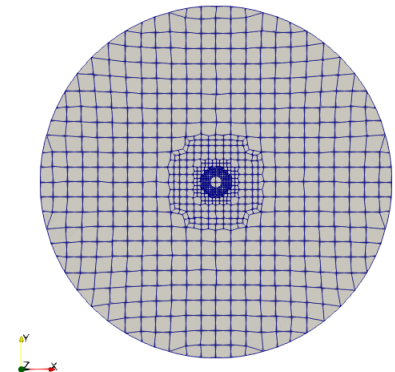
- In sliding mesh simulations, the solution is interpolated back-and-forth between the regions.
- The interpolation is done at the arbitrary mesh interface patches (AMI) .
- To reduce interpolation errors at the AMI patches, the meshes should be similar in the master and slave patches.



<http://www.wolfdynamics.com/training/movingbodies/image8.gif>



Rotating patch  
Master patch



Fix patch  
Slave patch

# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF

- At this point, the mesh is ready to use.
- You can visualize the mesh using `paraFoam`.
- If you use `checkMesh`, it will report that there are two regions.
- In the dictionary `constant/dynamicsMeshDict` we set which region will move and the rotation parameters.
- To preview the region motion, in the terminal type:
  - `$> moveDynamicMesh`
- To preview the region motion and check the quality of the AMI interfaces, in the terminal type:
  - `$> moveDynamicMesh -checkAMI -noFunctionObjects`
- In our YouTube channel you can find a step-by-step video explaining this case.

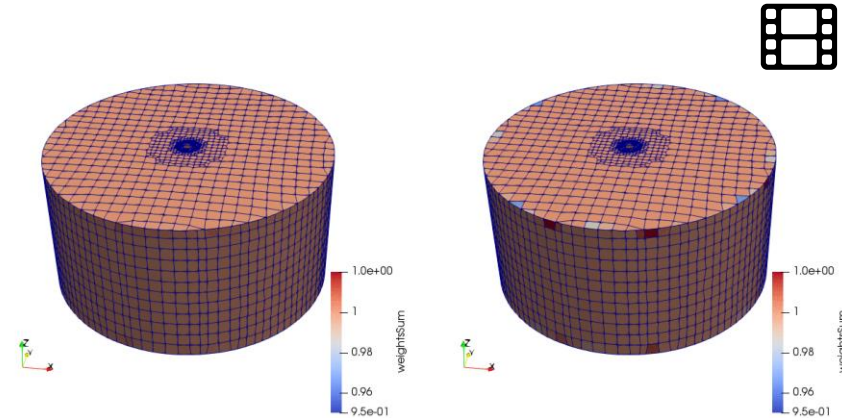




# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF

- The command `moveDynamicMesh -checkAMI` will print on screen the quality of the AMI interfaces for every time step.
- Ideally, you should get the AMI patches weights as close as possible to one.
- Weight values close to one will guarantee a good interpolation between the AMI patches.



<http://www.wolfynamics.com/training/movingbodies/image9.gif>

...  
...  
...

Name of the AMI patch

Name of the AMI patch



**Calculating AMI weights between owner patch: AMI1 and neighbour patch: AMI2**

**AMI: Creating addressing and weights between 2476 source faces and 2476 target faces** ← Number of faces in the AMI patches

**AMI: Patch source sum(weights) min/max/average = 0.94746705, 1.0067199, 0.99994232** ← AMI1 patch weights

**AMI: Patch target sum(weights) min/max/average = 0.94746692, 1.0004497, 0.99980782** ← AMI2 patch weights

...  
...  
...

# Moving bodies hands-on tutorials

## Continuous stirring tank reactor – Sliding meshes and MRF

- At this point, we are ready to run the simulation.
- You can choose between running using sliding meshes (directory `sliding_piso`) or MRF (directory `MRF_simple`).
- You can use the solver `pimpleFoam` for sliding meshes and the solver `simpleFoam` for MRF.
- You can also use `pimpleFoam` (unsteady solution) for the MRF. However, it is not computationally efficient, the idea of MRF is to reach a steady solution fast.
- You can also try `pimpleFoam` (with local time stepping LTS).

# Moving bodies hands-on tutorials

- Oscillating cylinder – Prescribed motion
- Let us run this case. Go to the directory:

```
$PTOFC/advanced_physics/dynamic_meshes/prescribed_motion/oscillatingCylinder
```

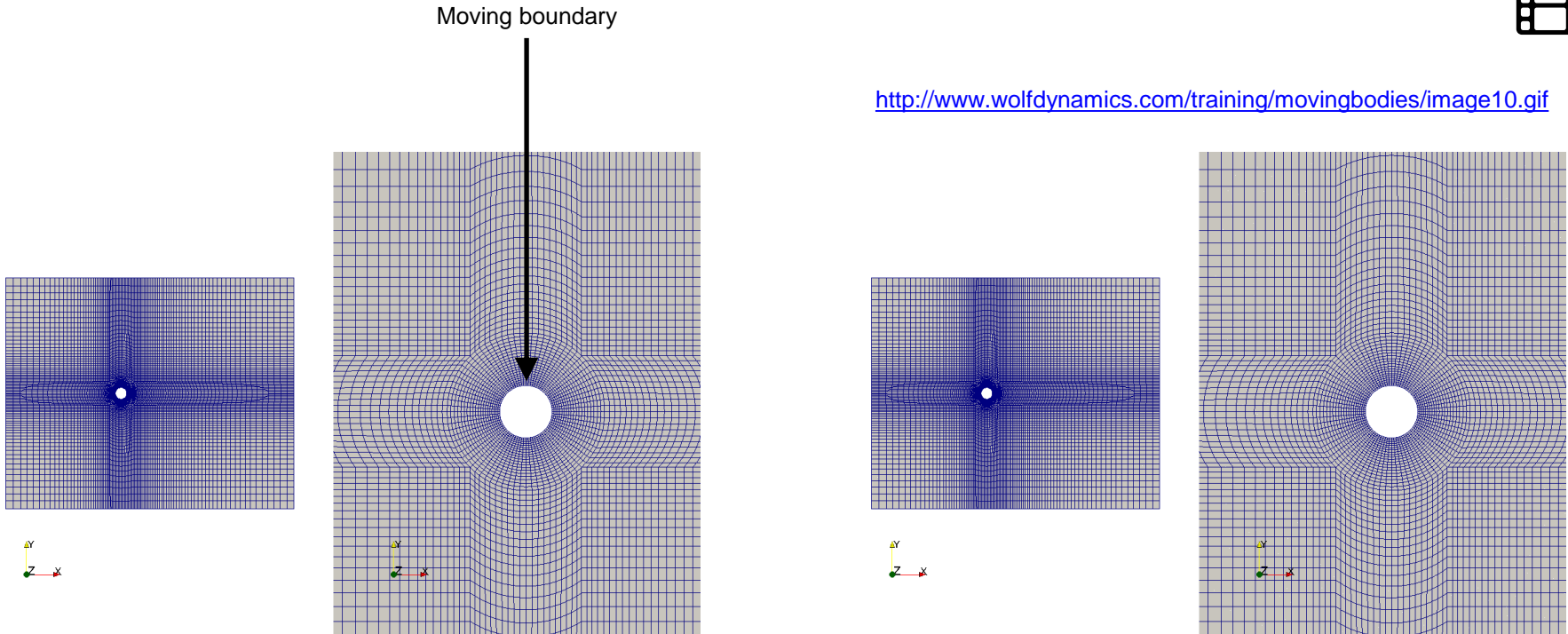
- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are 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.

# Moving bodies hands-on tutorials

## Oscillating cylinder – Prescribed motion



<http://www.wolfdynamics.com/training/movingbodies/image10.gif>



Cylinder prescribed motion – Oscillating motion

# Moving bodies hands-on tutorials

## Oscillating cylinder – Prescribed motion

- In the dictionary *constant/dynamicMeshDict* we select the mesh morphing method and the boundary patch that it is moving (lines 21 and 26, respectively).
- There are many mesh morphing methods implemented in OpenFOAM®.
- Mesh morphing is based in diffusing or propagating the mesh deformation all over the domain.
- You will need to find the best method for your case.
- The setup used in this case works fine most of the times.

The diagram shows a code snippet from the `dynamicMeshDict` file in OpenFOAM. The code is enclosed in a light green box with a red border. Annotations with arrows point to specific parts of the code:

- Method coefficients:** An arrow points to line 23, `displacementLaplacianCoeffs`.
- Mesh motion library for boundary patches:** An arrow points to line 17, `dynamicMotionSolverFvMesh`.
- Motion library:** An arrow points to line 19, `libfvMotionSolvers.so`.
- Solver for mesh motion method:** An arrow points to line 21, `displacementLaplacian`.
- Mesh diffusion method:** An arrow points to line 25, `inverseDistance`.
- Patch name:** An arrow points to line 26, `cylinder`.

```
17  dynamicFvMesh      dynamicMotionSolverFvMesh;
18
19  motionSolverLibs   ("libfvMotionSolvers.so");
20
21  motionSolver       displacementLaplacian;
22
23  displacementLaplacianCoeffs
24  {
25      diffusivity     inverseDistance (cylinder);
26  }
```

# Moving bodies hands-on tutorials

## Oscillating cylinder – Prescribed motion

- In the dictionary *0/pointDisplacement* we select the prescribed body motion.
- In this case we are using **oscillatingDisplacement** (line 44) for the **cylinder** patch.
- Each method has different input values. In this case it is required to define the amplitude (line 45) and the angular velocity (line 46) in rad/s.
- If the patch is not moving, we assign to it a fixedValue boundary conditions (lines 37-41).

```
37  in
38  {
39      type          fixedValue;
40      value          uniform (0 0 0);
41  }
42  cylinder
43  {
44      type          oscillatingDisplacement;
45      amplitude      ( 0 1 0 );
46      omega          6.28318;
47      value          uniform ( 0 0 0 ); ← Dummy value for paraview
48  }
```

# Moving bodies hands-on tutorials

## Oscillating cylinder – Prescribed motion

- You must assign the boundary condition **movingWallVelocity** to all patches that are moving.
- This is done in the dictionary  $O/U$ .

```
41  cylinder
42  {
43      type            movingWallVelocity;
44      value            uniform (0 0 0);
45  }
```

- And as usual, you will need to adjust the numerics according to your physics.
- In this case we need to solve the new fields **cellDisplacement** and **diffusivity**, which are related to the mesh motion and morphing.
- In the dictionary *fvSolution*, you will need to add a linear solver for the field **cellDisplacement**.
- In the dictionary *fvSchemes*, you will need to add the discretization schemes related to the mesh morphing diffusion method, **laplacian(diffusivity, cellDisplacement)**.
- If you are dealing with turbulence modeling the treatment of the wall functions is the same as if you were working with fixed meshes.

# Moving bodies hands-on tutorials

## Oscillating cylinder – Prescribed motion

- At this point, we are ready to run the simulation.
- Before running the simulation, you can check the mesh motion.
- During this check, you can use large time-steps as we are not computing the solution, we are only interested in checking the motion.
- To check the mesh motion, type in the terminal:

```
1. | $> moveDynamicMesh -noFunctionObjects
```



# Moving bodies hands-on tutorials

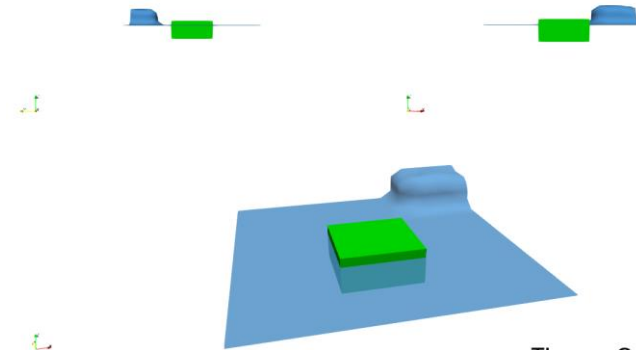
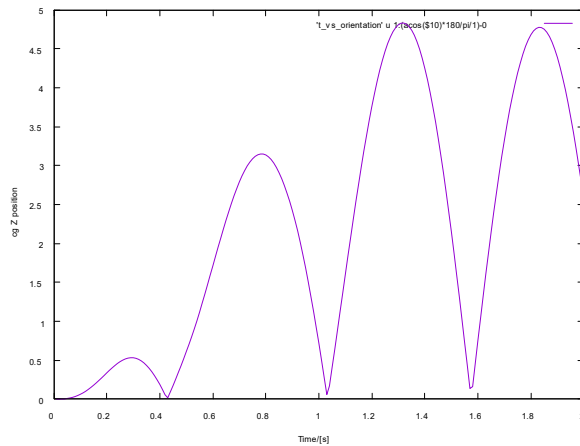
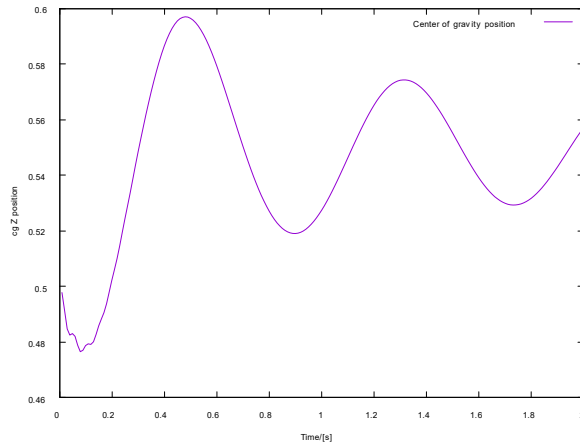
- Floating body – Rigid body motion
- Let us run this case. Go to the directory:

```
$PTOFC/advanced_physics/dynamic_meshes/rigid_body_motion/floatingObject
```

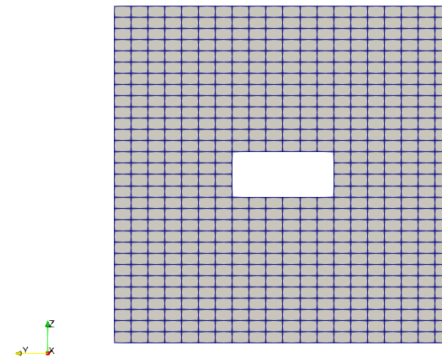
- In the case directory, you will find a few scripts with the extension .sh, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on.
- These scripts can be used to run the case automatically by typing in the terminal, for example,
  - `$> sh run_solver`
- These scripts are 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.

# Moving bodies hands-on tutorials

## Floating body – Rigid body motion



<http://www.wolfdynamics.com/training/movingbodies/image11.gif>



<http://www.wolfdynamics.com/training/movingbodies/image12.gif>

Floating body simulation with VOF and turbulence modeling.

# Moving bodies hands-on tutorials

## Floating body – Rigid body motion

- As for prescribed motion, in rigid body motion the mesh morphing is based in diffusing or propagating the mesh deformation all over the domain.
- In the dictionary *constant/dynamicMeshDict* we select the mesh morphing library and rigid body motion library (lines 17-21).
- The rigid motion solver will compute the response of the body to external forces.
- In lines 23-77, we define all the inputs required by the rigid motion solver.

```
17  dynamicFvMesh      dynamicMotionSolverFvMesh; ← Mesh motion library for
18                                     boundary patches
19  motionSolverLibs    ("libsixDoFRigidBodyMotion.so"); ← Motion library
20
21  motionSolver        sixDoFRigidBodyMotion; ← Solver for mesh motion
22                                     method
23  sixDoFRigidBodyMotionCoeffs
24  {
25      ...
26      ...
27      ...
28  }
```

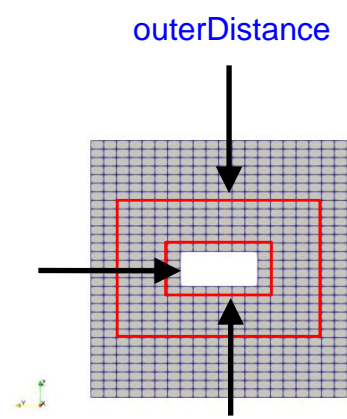
Method coefficients

# Moving bodies hands-on tutorials

## Floating body – Rigid body motion

- The dictionary *constant/dynamicMeshDict* (continuation).

```
23 sixDoFRigidBodyMotionCoeffs
24 {
25     patches          (floatingObject); ← Moving patch
26
27     innerDistance    0.1; } ← Mesh deformation limits.
28     outerDistance    0.4; } ← The mesh will not be deformed in the fringe located within
                               innerDistance and outerDistance (distance normal to the wall)
29
30     centreOfMass     (0.5 0.5 0.5); } ← Physical properties of the body
31     mass             5;
32     momentOfInertia  (0.08 0.08 0.1);
33
34     report           on; ← Report on screen position of the body
35
36     accelerationRelaxation 0.7;
37
38
39
40
41
42
43
44     solver           ← Rigid body motion solver
45     {
46         type Newmark;
47     }
48
49     ...
50     ...
51     ...
```



The diagram illustrates the mesh deformation region around a body patch. A central white rectangle represents the 'Body patch'. It is surrounded by a blue grid representing the mesh. Two concentric red rectangles define the 'innerDistance' and 'outerDistance' from the body patch. Arrows point from the labels 'outerDistance' and 'innerDistance' to their respective red rectangles. An arrow points from the 'Body patch' label to the white rectangle. A small 3D coordinate system is shown at the bottom left of the diagram.

Set it to zero if you do not want to apply mesh morphing to the inner region

# Moving bodies hands-on tutorials

## Floating body – Rigid body motion

- The dictionary *constant/dynamicMeshDict* (continuation).

```
50     constraints
51     {
52         fixedAxis
53         {
54             sixDoFRigidBodyMotionConstraint axis;
55             axis (0 1 0);
56         }
57
58         fixedLine
59         {
60             sixDoFRigidBodyMotionConstraint line;
61             centreOfRotation (0.5 0.5 0.5);
62             direction (0 0 1);
63         }
64     }
65
66     restraints
67     {
68
69     }
70 }
```

Motion constraints  
If you do not give any constraint, the body is free to move in all directions.

Body restraints  
Restraints can be used to damp the acceleration of the body.  
In this case, we are not using restraints

# Moving bodies hands-on tutorials

## Floating body – Rigid body motion

- In the dictionary *0/pointDisplacement* we select the body motion.
- For rigid body motion, the body motion is computed by the solver, therefore, we use the boundary condition calculated.

```
33 floatingObject
34 {
35     type          calculated;
36     value          uniform (0 0 0);
37 }
```

- You must assign the boundary condition **movingWallVelocity** to all patches that are moving. This is done in the dictionary *0/U*.

```
33 floatingObject
34 {
35     type          movingWallVelocity;
36     value          uniform (0 0 0);
37 }
```

- If you are dealing with turbulence modeling the treatment of the wall functions is the same as if you were working with fixed meshes.

# Moving bodies hands-on tutorials

## Floating body – Rigid body motion

- And as usual, you will need to adjust the numerics according to your physics.
- In the case directory, you will find the script *extractData*.
- This script can be used to extract the position of the body during the simulation.
- In order to use the *extractData* script, you will need to save the log file of the simulation.
- At this point, we are ready to run the simulation.
- We will use the solver `interFoam`.