

# Online Training – Advanced session

## July 2021

---

# Turbulence modeling in OpenFOAM®: Guided Tutorials



# Guided tutorials

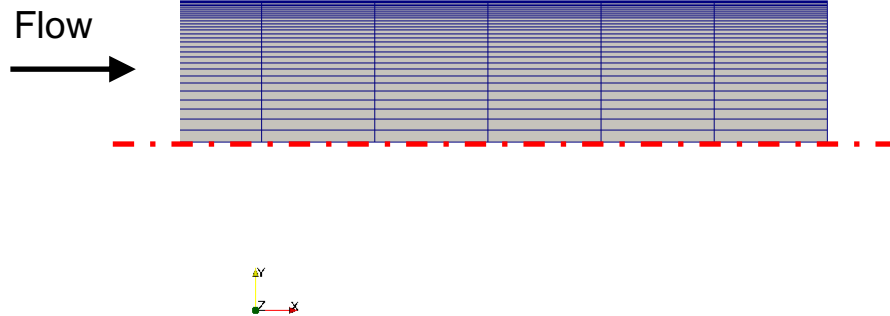
- **Guided tutorial 1.** Laminar-Turbulent pipe.
- This case is ready to run.
- The case is located in the directory:

```
$TM/turbulence/GT1/
```

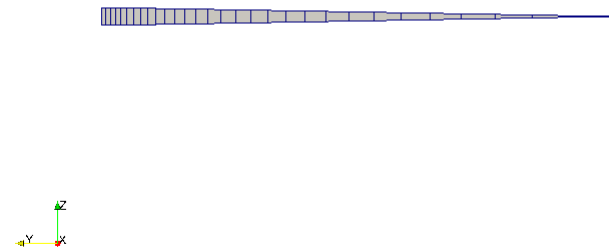
- In the case directory, you will find the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.
- You will also find a few additional files (or scripts) with the extension `.sh`, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.
- We highly recommend opening the `README.FIRST`, file and typing 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, use the automatic scripts to run the cases.
- From this point on, please follow me. We are all going to work at the same pace.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe



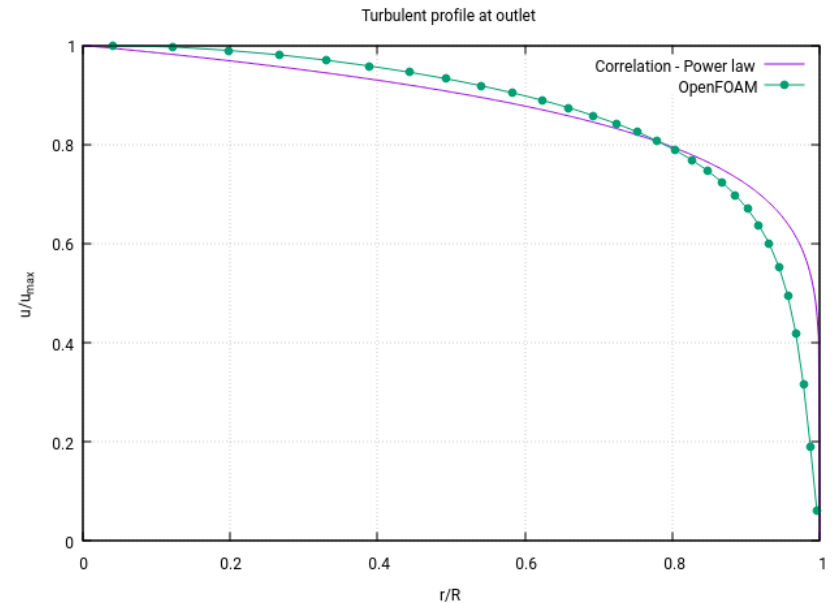
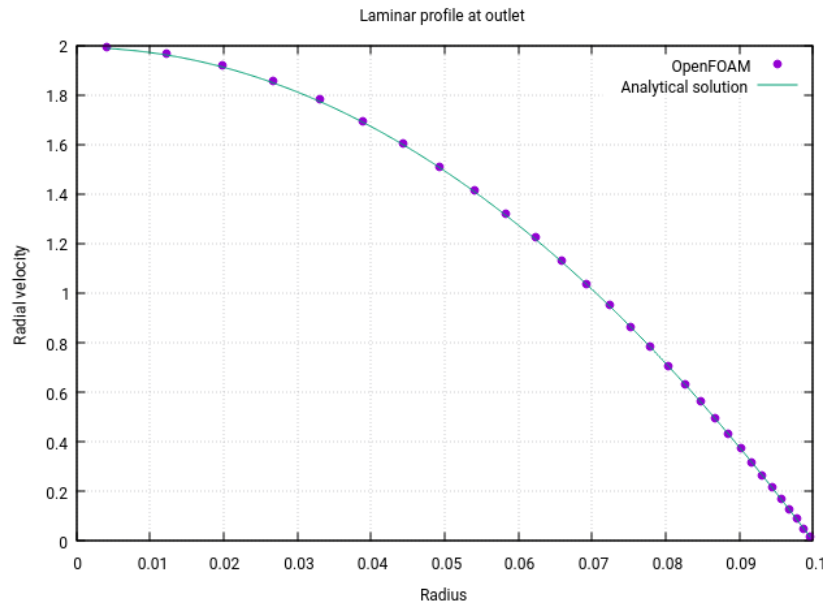
Circular wedge



- In this guided tutorial we will simulate a laminar and a turbulent flow in a circular pipe.
- We will use axial symmetry.
- This problem has plenty of experimental data and analytical correlations for validation.

# Guided tutorials

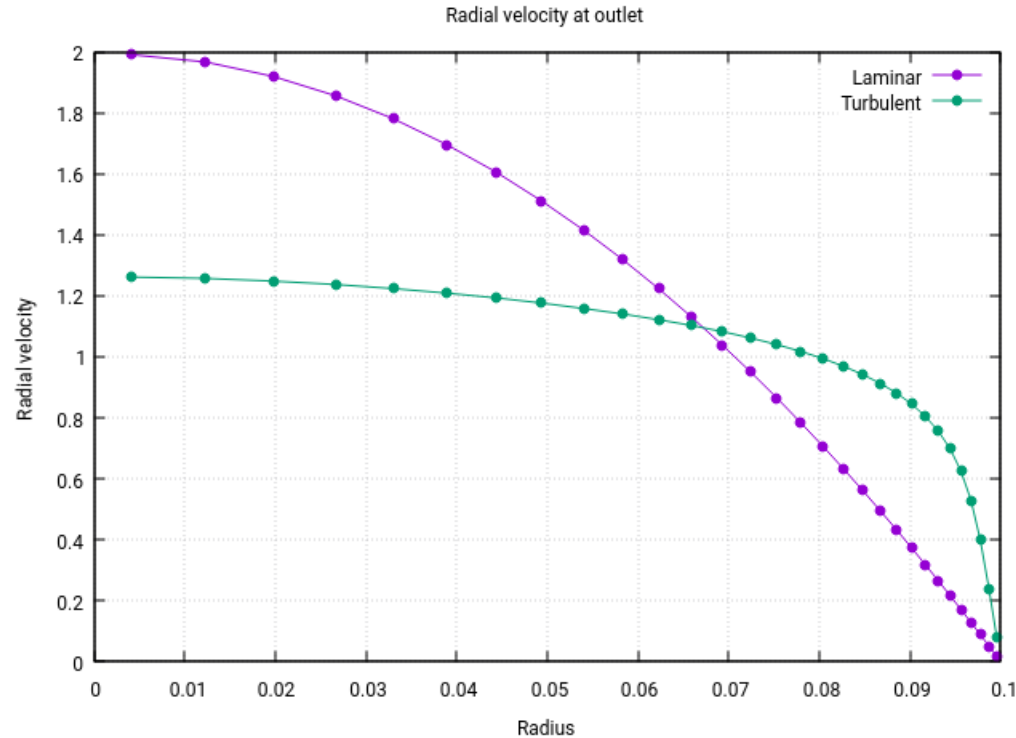
## Guided tutorial 1 – Laminar-Turbulent pipe



- In this guided tutorial we will simulate a laminar and a turbulent flow in a circular pipe.
- We will use axial symmetry.
- This problem has plenty of experimental data and analytical correlations for validation.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe



- In this guided tutorial we will simulate a laminar and a turbulent flow in a circular pipe.
- We will use axial symmetry.
- This problem has plenty of experimental data and analytical correlations for validation.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- To set a turbulent model in OpenFOAM, you will need to modify the dictionary *momentumTransport*. This dictionary is located in the directory **constant**.

```
simulationType  RAS;           ← RANS type simulation
RAS             ← RANS sub-dictionary
{
  RASModel      kOmegaSST;     ← RANS model to use
  turbulence    on;           ← Turn on/off turbulence. Runtime modifiable
  printCoeffs   on;           ← Print model coefficients
}
```

- In this dictionary you choose what kind of approach you want to use (keyword **simulationType**): laminar, RAS, or LES.
- In you choose a RAS or LES approach, you will need to define a sub-dictionary with the options specific to the selected model.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- To option **printCoeffs** in the dictionary *momentumTransport* will print all the constants/coefficients used in the model (you can also read the source code to get this information).
- In this case, you will get the following information in the output screen,

```
RASModel      kOmegaSST;  
turbulence    on;  
printCoeffs   on;  
alphaK2       1;  
alphaK1       0.85;  
alphaOmega1   0.5;  
alphaOmega2   0.856;  
gamma1        0.5555555555556;  
gamma2        0.44;  
beta1         0.075;  
beta2         0.0828;  
betaStar      0.09;  
a1            0.31;  
b1            1;  
c1            10;  
F3            false;
```

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- So, if you want to change one or more of these constants/coefficients just add the new entry in the dictionary *momentumTransport*, in the sub-dictionary RAS (after the keyword **printCoeffs**), as follows,

```
simulationType  RAS;  
  
RAS  
{  
    RASModel      kOmegaSST;  
    turbulence    on;  
    printCoeffs   on;  
  
    alphaK1       8.31445;  
    alphaK2       3.14159; ← New model coefficients  
  
}
```

- This works for all RANS and LES models.
- No need to say that you need to know what are you doing.



# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- When dealing with turbulence, setting boundary and initial conditions is of paramount importance.
- Our best advice is to get familiar with the theory.
- In the NASA Turbulence Modeling Resource site, you will find plenty of information:  
<https://turbmodels.larc.nasa.gov/>
- If you are feeling lazy and you do not have good estimates, you can use our calculator for the estimation of turbulence properties (use with caution):  
<http://www.wolfdynamics.com/tools.html?id=110>
- If you want to estimate the  $y^+$  value or get an idea of where to put the first mesh vertex or cell center normal to the wall, you can use our calculator:  
<http://www.wolfdynamics.com/tools.html?id=2>
- Remember to choose the near the wall treatment.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- The initial value for the turbulent kinetic energy  $k$  can be found as follows

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

- The initial value for the specific kinetic energy  $\omega$  can be found as follows

$$\omega_{farfield} = \frac{\rho k}{\mu} \left( \frac{\mu_t}{\mu} \right)^{-1}$$

- If you need to, the initial value for the turbulent dissipation  $\epsilon$  can be found as follows

$$\epsilon_{farfield} = \frac{C_\mu k^2}{\nu \cdot (\mu_t/\mu)} \quad \text{where} \quad C_\mu = 0.09$$

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- If you are totally lost, you can use the following 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

- Where  $\mu_t/\mu$  is the viscosity ratio and  $I = u'/\bar{u}$  is the turbulence intensity.
- By the way, use these guidelines for external aerodynamics only.
- Use these values only if you do not have better estimates.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- When it comes to near-wall treatment, you have three options: use wall functions ( $30 < y^+ < 300$ ), use  $y^+$  insensitive wall functions ( $1 < y^+ < 300$ ), and resolve the boundary layer ( $y^+ < 6$ ).
- Hereafter you will find a list of typical wall functions boundary conditions:

Field	Wall functions – High RE	Resolved BL – Low RE
nut	nut(-)WallFunction* or nutUSpaldingWallFunction** (with 0 or a small number)	nutUSpaldingWallFunction**, nutkWallFunction**, nutUWallFunction**, nutLowReWallFunction or fixedValue (with 0 or a small number)
k, q, R	kqRWallFunction $k_{wall} = k$	kqRWallFunction or kLowReWallFunction (with inlet value, 0, or a small number)
epsilon	epsilonWallFunction $\epsilon_{wall} = \epsilon$	epsilonWallFunction (with inlet value) or zeroGradient or fixedValue (with 0 or a small number)
omega	omegaWallFunction*** $\omega_{wall} = 10 \frac{6\nu}{\beta y^2}$	omegaWallFunction** or fixedValue (both with a large number)
nuTilda	–	fixedValue (one to ten times the molecular viscosity, a small number, or 0)

\* Usually, zero or a small number at the wall

\*\* For  $y^+$  insensitive wall functions (also known as continuous or scalable wall functions)

\*\*\*  $\beta = 0.075$

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- Typical **nut** wall functions boundary conditions are:
  - **nutkRoughWallFunction** (high-Re)
  - **nutkWallFunction** (high-Re and low-Re -  $y^+$  insensitive)
  - **nutLowReWallFunction** (low-Re)
  - **nutUWallFunction** (high-Re and low-Re -  $y^+$  insensitive)
  - **nutUSpaldingWallFunction** (high-Re and low-Re -  $y^+$  insensitive using Spalding continuous function)
- A low-Re turbulence model solves the boundary layer ( $y^+ < 6$ ).
- A high-Re turbulence model uses wall functions ( $30 < y^+ < 300$ ).
- Low-Re means that the local Re is low (as in the boundary layer), and high-Re means the local Re is high. These definitions are not related to the global Re.
- Remember, in order to use wall functions, you need to define the patch type as a **wall** in the dictionary *constant/polyMesh/boundary*.
- Also, if you are dealing with moving walls, do not forget to use boundary conditions that take into account the grid/patch velocity.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- The boundary and initial conditions are set in the dictionary files located in the 0 folder.
- As we are using the  $k - \omega$  model, we will need the following dictionary files:  $k$ ,  $\omega$ , and  $\nu_t$ .
- $k$  and  $\omega$  are the additional closure equations of the turbulence model, and  $\nu_t$  is the turbulent viscosity.
- The initial conditions can be calculated using the recommended formulas, but if you have better estimates feel free to use them.
- In this case we will use the following boundary conditions at the walls: for  $k$  we use the boundary condition **kqRWallFunction**, for  $\omega$  we use **omegaWallFunction**, and for  $\nu_t$  we use **nutUSpaldingWallFunction**.
- The **calculated** boundary condition used in the  $\nu_t$  dictionary, means that its value is computed from  $k$  and  $\omega$ .
- This case setup will use  $y^+$  insensitive wall functions.
- This is a pretty standard setup for the  $k - \omega$  model.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- Reminder:
  - The flow is incompressible.
  - The pipe radius and length are 0.1 m and 8.0 m, respectively.
  - And we are targeting for a  $Re = 100$  (laminar flow), and  $Re = 10000$  (turbulent flow).

$$Re = \frac{\rho \times U \times D}{\mu} = \frac{U \times D}{\nu}$$

$$\nu = \frac{\mu}{\rho}$$

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- Reminder:
  - To compute the wall distance units  $y^+$ , you can use the following equation

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

- Where  $y$  is the distance to the first cell center normal to the wall, and  $U_\tau$  is the friction velocity and is equal to

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

where  $\tau_w$  is the wall shear stresses.



# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- We are going to use the following solver: `simpleFoam`.
- This case is rather simple, but we will use it to explain many features used in OpenFOAM when dealing with turbulence.
- We will compare the numerical solution with the analytical solution and empirical correlations.
- We will compare the velocity profile of the laminar flow and the turbulent flow.
- After finding the numerical solution, we will do some sampling and data manipulation.
- Then we will do some plotting (using gnuplot or Python) and scientific visualization.
- 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.
- You will find the instructions of how to run this case in the file `README.FIRST` located in the case directory.

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- At this point, we are ready to run the simulation.
- To run the turbulent case, go to the `GT1/turbulent` directory.
- You can use the script `run_all.sh` to run the case automatically.
- Type in the terminal:
  - `$> sh run_all.sh`
- Feel free to open the file `run_all.sh` to see all the commands used.
- Or if you prefer, you can insert the commands manually in the terminal window (refer to the next slide).

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- To run the turbulent case typing one command at a time, go to the `GT1/turbulent` directory and type in the terminal:

```
1. $> foamCleanTutorials
```

```
2. $> blockMesh
```

```
3. $> extrudeMesh
```

```
4. $> createPatch -overwrite
```

```
5. $> simpleFoam | tee log
```

```
6. $> simpleFoam -postprocess -func wallShearStress -latestTime
```

```
7. $> simpleFoam -postprocess -func yPlus
```

```
8. $> postprocess -func sampleDict0 -latestTime
```

```
9. $> gnuplot gnuplot/gnuplot_script
```

```
10.$> paraFoam
```

# Guided tutorials

## Guided tutorial 1 – Laminar-Turbulent pipe

- In steps 1 we clean the case directory.
- Be careful, the command `foamCleanTutorials` will erase the mesh.
- If you want to erase the solution and keep the mesh, type in the terminal:
  - `$> foamListTimes -rm`
- In step 2, 3 and 4 we generate the mesh and rename and change the type of the patches.
- In step 5, we run the simulation using the solver `simpleFoam`.
- In step 6, we use the utility `postprocess` to compute the wall shear stresses.
- In step 7, we use the utility `postprocess` to compute the  $y^+$  value at the walls.
- Steps 6 and 7 are very important when post processing turbulence simulations.
- In step 8 we do some sampling.
- In step 9, we plot the data sampled in step 8. For plotting, we use the program `gnuplot`.

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

# Guided tutorials

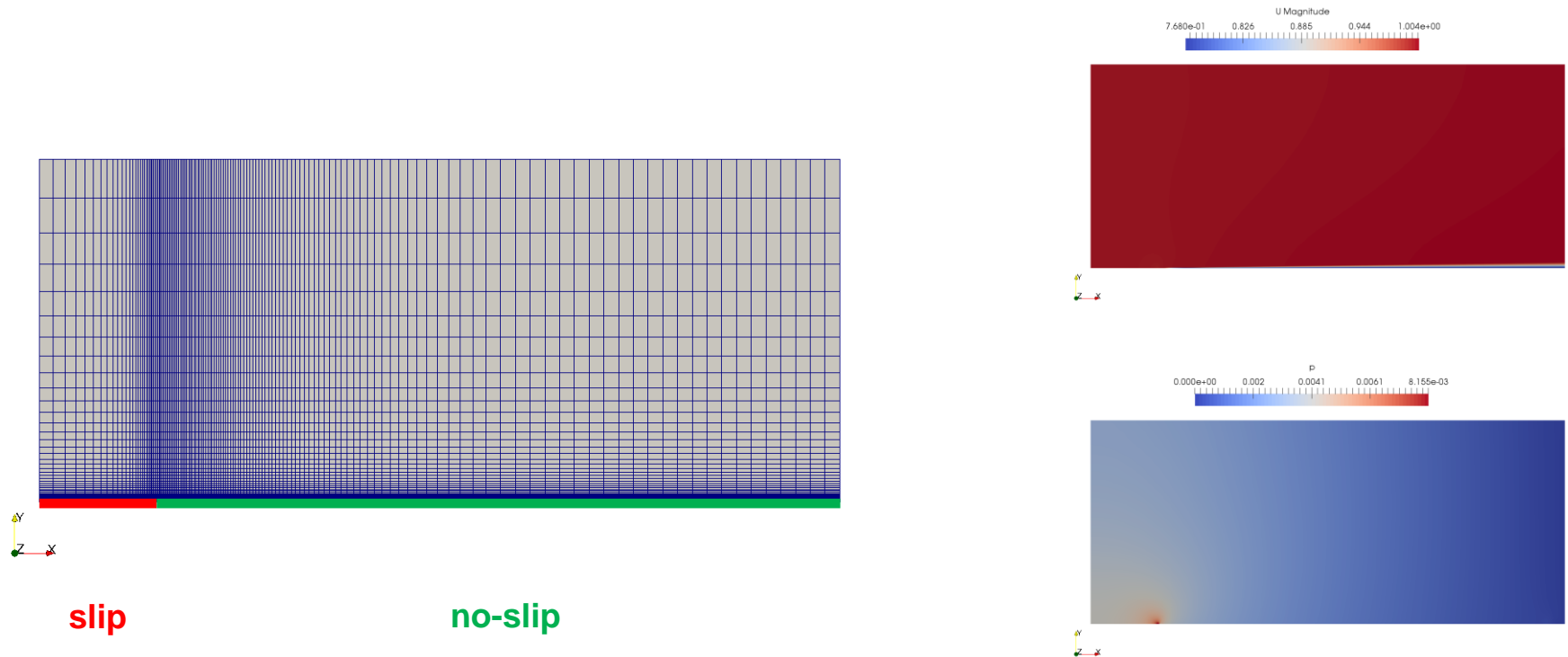
- **Guided tutorial 2.** Laminar-Transitional-Turbulent flat plate.
- This case is ready to run.
- The case is located in the directory:

```
$TM/turbulence/GT2/
```

- In the case directory, you will find the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.
- You will also find a few additional files (or scripts) with the extension `.sh`, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.
- We highly recommend opening the `README.FIRST`, file and typing 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, use the automatic scripts to run the cases.
- From this point on, please follow me. We are all going to work at the same pace.

# Guided tutorials

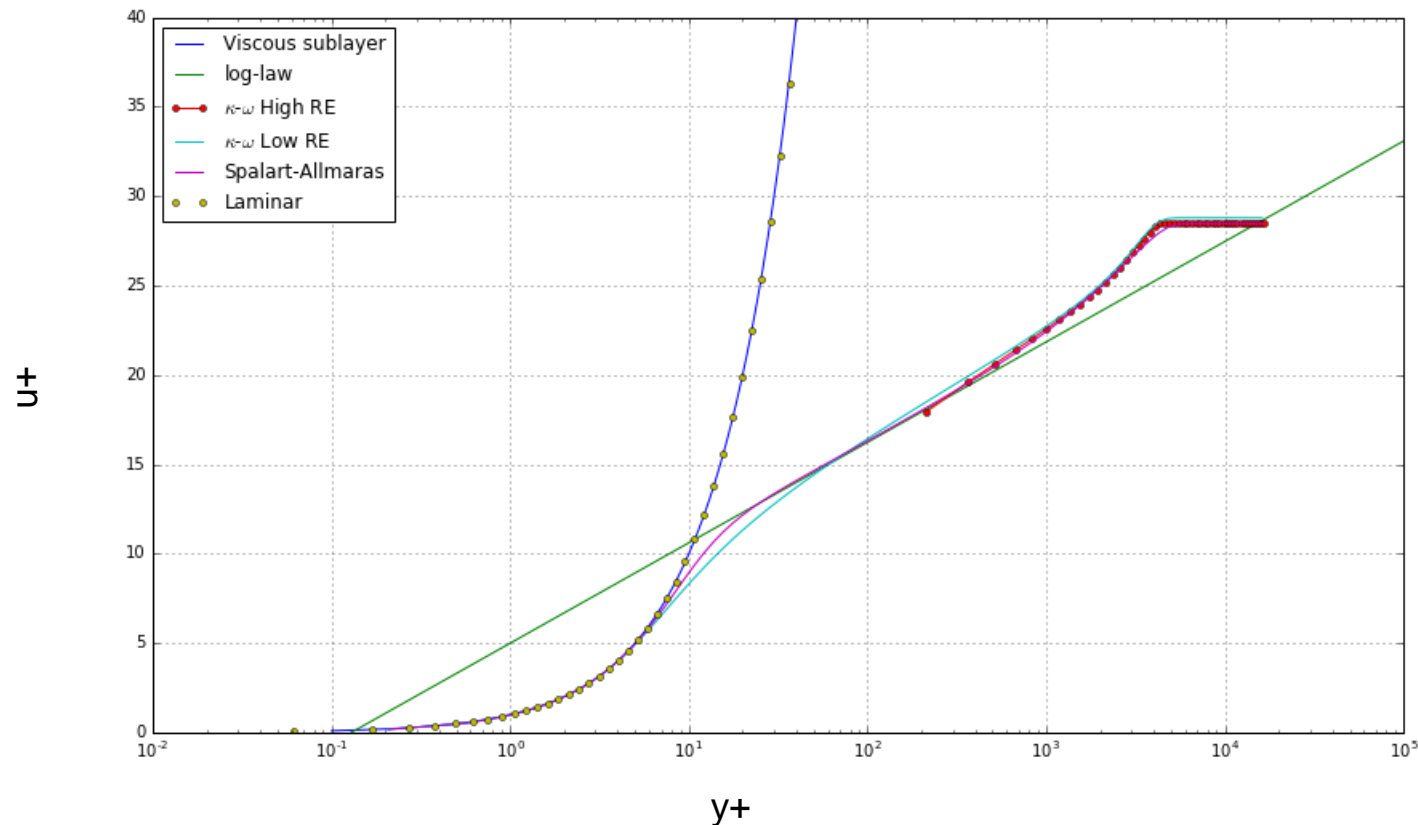
## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate



- In this guided tutorial we will simulate a laminar, a transitional, and a turbulent flow.
- We will use the classical 2D Zero Pressure Gradient Flat Plate validation case.
- This problem has plenty of experimental data, numerical data, and analytical correlations for validation.

# Guided tutorials

## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate

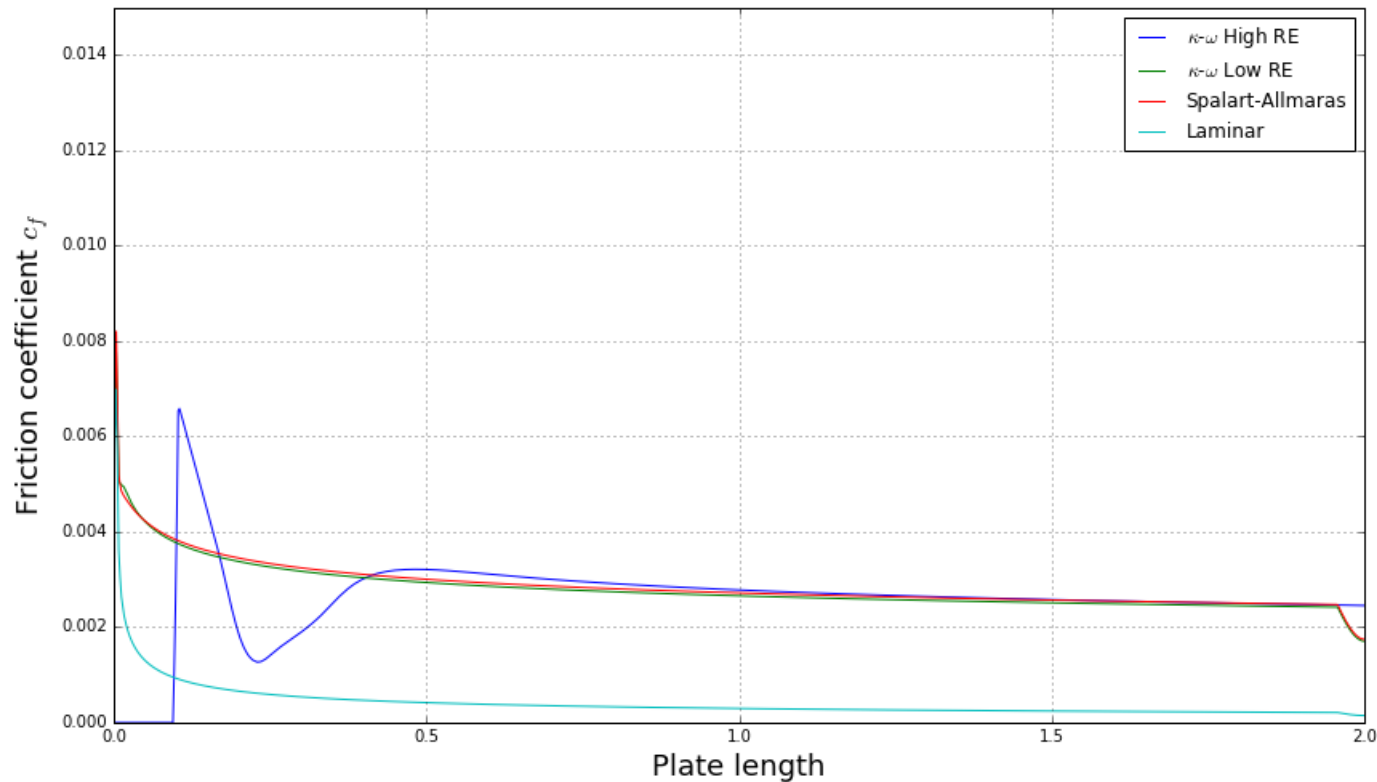


- The best way to understand the near the wall treatment and the effect of turbulence near the walls, is by reproducing the law of the wall.
- By plotting the velocity in terms of the non-dimensional variables  $u^+$  and  $y^+$ , we can compare the profiles obtained from the simulations with the theoretical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



# Guided tutorials

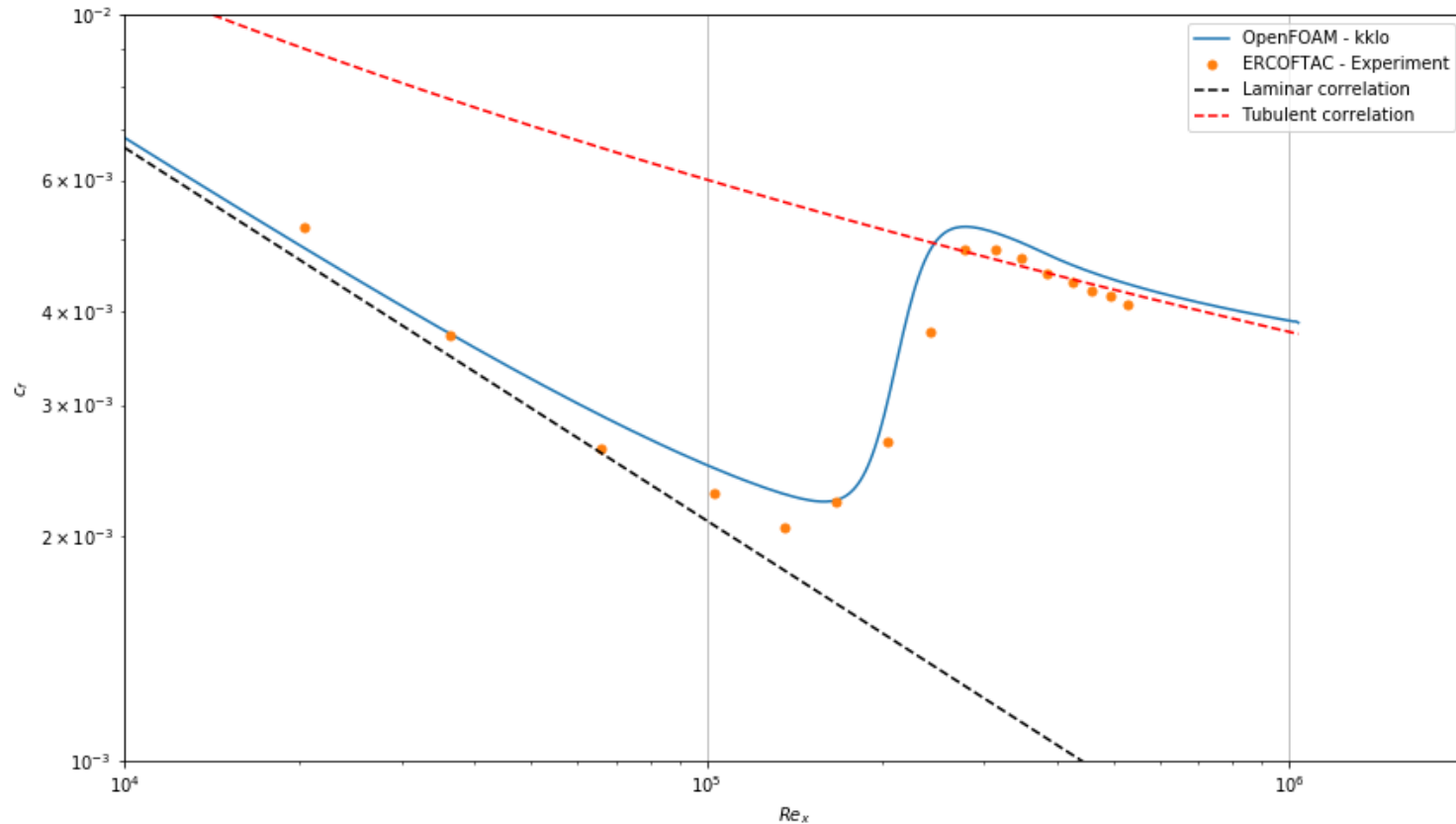
## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate



- We can also compute the friction coefficient and compare it with other results and empirical correlations.

# Guided tutorials

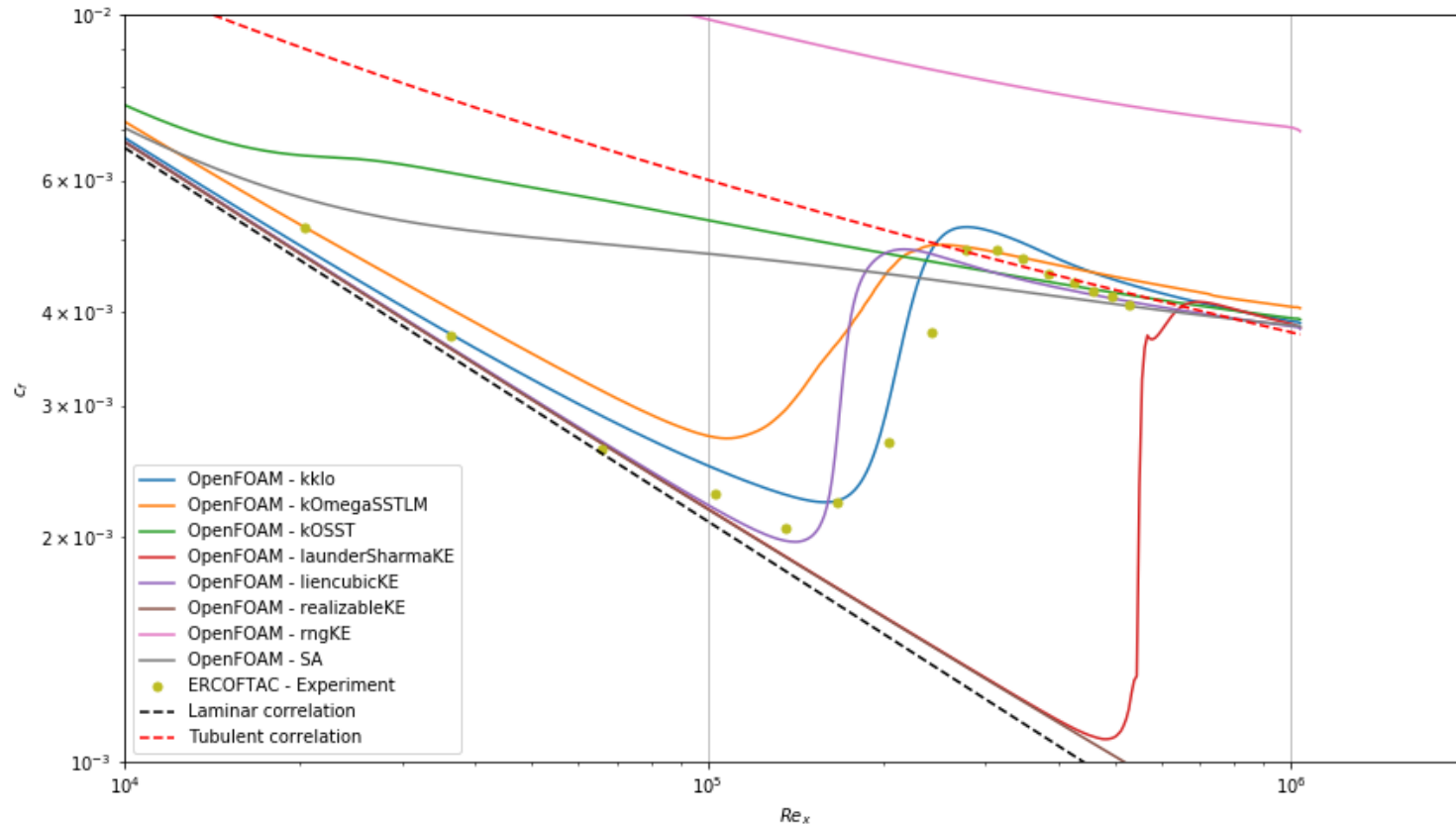
## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate



- Finally, we will briefly address a case related to transition to turbulence (ERCOFTAC T3A 3% test-case).
- Maybe, this is the most challenging part in turbulence modelling as nobody knows what is happening in the transition region.

# Guided tutorials

## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate



- Finally, we will briefly address a case related to transition to turbulence (ERCOFTAC T3A 3% test-case).
- Maybe, this is the most challenging part in turbulence modelling as nobody knows what is happening in the transition region.

# Guided tutorials

## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate

- In the NASA Turbulence Modeling Resource site <https://turbmodels.larc.nasa.gov/>, you will find the complete details of the 2D Zero Pressure Gradient Flat Plate Validation Case. You can download the mesh and numerical results in this site.
- In the following links,  
[http://www.ercoftac.org/special\\_interest\\_groups/15\\_turbulence\\_modelling/sig\\_15\\_test\\_case\\_database/](http://www.ercoftac.org/special_interest_groups/15_turbulence_modelling/sig_15_test_case_database/)  
[http://cfm.mace.manchester.ac.uk/cgi-bin/cfddb/prpage.cgi?20&EXP&&database/cases/case20&cas20\\_head.html&cas20\\_desc.html&cas20\\_meth.html&cas20\\_data.html&cas20\\_refs.html&cas20\\_sol.html&0&1&0&1&1&unknown](http://cfm.mace.manchester.ac.uk/cgi-bin/cfddb/prpage.cgi?20&EXP&&database/cases/case20&cas20_head.html&cas20_desc.html&cas20_meth.html&cas20_data.html&cas20_refs.html&cas20_sol.html&0&1&0&1&1&unknown)

You will find more information about the ERCOFTAC T3A 3% transition test case.

# Guided tutorials

## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate

- In the directory `$TM/turbulence/GT2/laminar-turbulent/samplings` you will find a jupyter notebook that you can use to plot the non-dimensional  $u^+$  and  $y^+$  profiles of the precomputed solutions.
- In the directory `python` of each case, you will a jupyter notebook (a web-browser based python script), and a python script that you can use to plot the non-dimensional  $u^+$  and  $y^+$  profiles of each 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.
- Then, you can compute the shear velocity, friction coefficient, and  $u^+$  and  $y^+$  values.

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

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

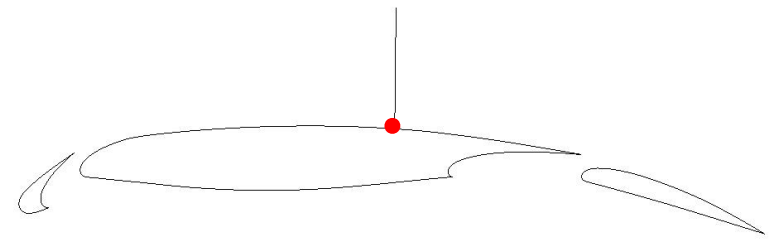
$$U_\tau = \sqrt{\frac{\tau_w}{\rho}}$$

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

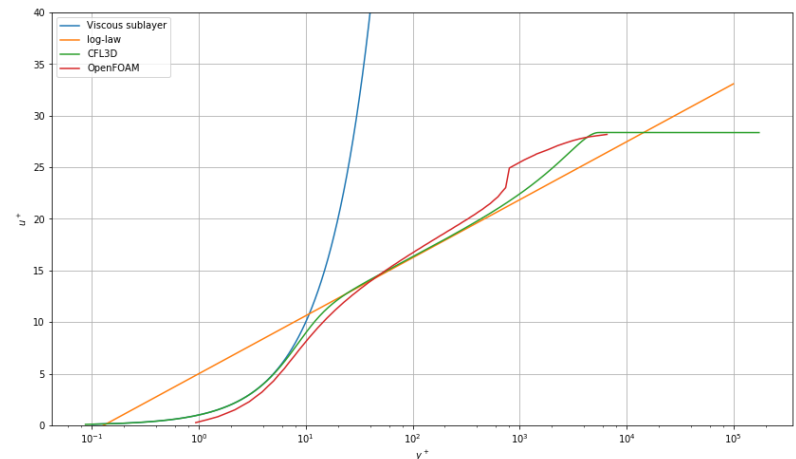
# Guided tutorials

## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate

- Let us explain again how to plot the velocity in terms of the non-dimensional variables  $u^+$  and  $y^+$ .
- First, pick a location where to do the sampling (avoid recirculation areas or regions with flow separation), and sample the velocity normal to the wall.
- At the same location sample the wall shear stresses and compute the magnitude.
- With the sampled values, you can compute the shear velocity,  $u^+$  and  $y^+$ .
- Remember, when computing  $y^+$  you need to use the distance normal to wall  $y$ .
  - Disregarding of the location in the domain, this distance should start from zero.



Sampling location



# Guided tutorials

## Guided tutorial 2 – Laminar-Transitional-Turbulent flat plate

- We are going to use the following solvers: `simpleFoam` (for RANS) and `pimpleFoam` (for LES).
- 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.
- When dealing with transitional models, we do not use wall functions.
- We will also show how to do the post-processing in order to reproduce the law of the wall. For this, we will use a jupyter notebook (or a python script) and paraview. Both approaches are valid.
- 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.
- You will find the instructions of how to run this case in the file `README.FIRST` located in the case directory.

# Guided tutorials

- **Guided tutorial 3.** Turbulence flow over a backward facing step.
- This case is ready to run.
- The case is located in the directory:

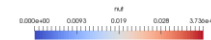
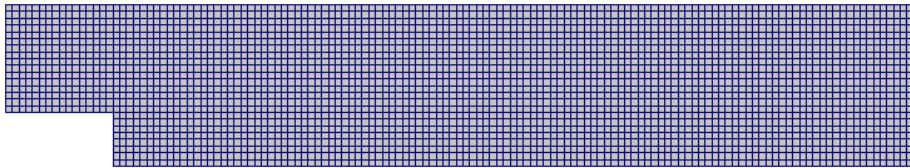
```
$TM/turbulence/GT3/
```

- In the case directory, you will find the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.
- You will also find a few additional files (or scripts) with the extension `.sh`, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.
- We highly recommend opening the `README.FIRST`, file and typing 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, use the automatic scripts to run the cases.
- From this point on, please follow me. We are all going to work at the same pace.



# Guided tutorials

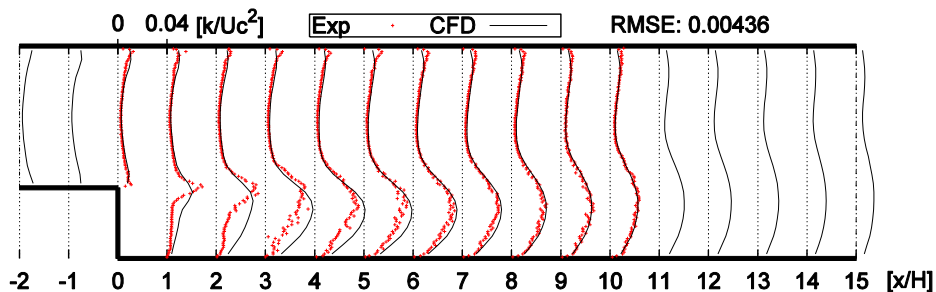
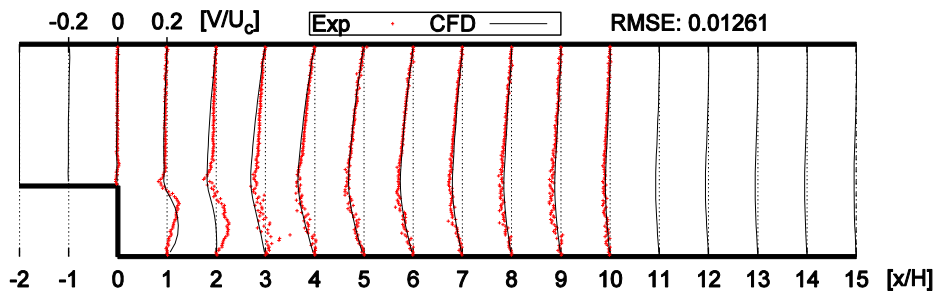
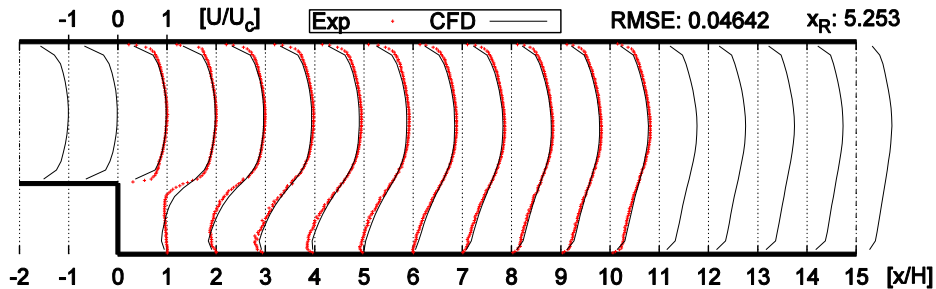
## Guided tutorial 3 – Turbulence flow over a backward facing step



- In this guided tutorial we will simulate a turbulent flow using RANS.
- This is the classical backward facing step widely used for validation. This case has plenty of experimental and numerical data for validation.
- We will see the importance of choosing the right turbulence model and the right near the wall treatment

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step



- We will compare the numerical solution with the experimental results at different locations.
- As inlet boundary conditions we will use the experimental profiles.
- As this is a fast case, we can also play around with the numerical schemes and linear solvers to see their influence on the accuracy of the solution.
- We can also study the influence of mesh refinement on the accuracy and stability of the solution.

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- To interpolate values from a set of supplied points in space and time, you can use the **timeVaryingMappedFixedValue** boundary conditions.
- For a vector field:

```
inlet
{
    type                timeVaryingMappedFixedValue;
    offset              (0 0 0);
    setAverage          off;
}
```

- For a scalar field:

```
inlet
{
    type                timeVaryingMappedFixedValue;
    offset              0;
    setAverage          off;
}
```

- Refer to the online documentation for more information of how to use this BC.

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- The **timeVaryingMappedFixedValue** boundary condition interpolates the values from a set of supplied points in space and time.
- The data to interpolate should be specified in the directory **constant/boundaryData/<patchName>** .
- Inside this directory you should create the file *points*, which contains the coordinates of the points.
- The values to interpolate should be in the directory **constant/boundaryData/<patchName>/0** (or the desired time directory).
- So for example, if you want to interpolate some values (let us say **U** and **p**), in the patch named **inlet**, at time **0**, you should have the following files and directory structure:
  - **constant/boundaryData/inlet/**
    - *constant/boundaryData/inlet/points*
    - **constant/boundaryData/inlet/0**
      - *constant/boundaryData/inlet/0/U*
      - *constant/boundaryData/inlet/0/p*

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- The file `constant/boundaryData/inlet/points` is formatted as follows:

```
FoamFile
{
    version      2.0;
    format       ascii;
    class        vectorField;
    object       points;
}

(
    (-2 1.00 0)
    (-2 1.01 0)
    (-2 1.02 0)
    ...
    ...
    ...
)
```

Points coordinates.  
They do not need to coincide with the  
cell centers or nodes.

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- The file `constant/boundaryData/inlet/0/U` is formatted as follows:

```
FoamFile
{
  version      2.0;
  format       ascii;
  class        vectorAverageField;
  object       values;
}

(0 0 0)
100
(
  (0.0 0 0)
  (0.1 0 0)
  (0.2 0 0)
  ...
  ...
  ...
)
```

For vectors

Average

Number of points. Must be same as in the file points

Values to interpolate.  
Each value corresponds to point in the file points.

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- The file `constant/boundaryData/inlet/0/p` is formatted as follows:

```
FoamFile
{
  version      2.0;
  format       ascii;
  class        scalarAverageField; ← For scalars
  object       values;
}

0 ← Average
100 ← Number of points. Must be same as in the file points
(
  0.2 ← Values to interpolate.
  0.6 ← Each value corresponds to point in the file points.
  1.0
  ...
  ...
  ...
)
```

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- We will also compute integral length scales and ratio of integral length scales to grid length scales. We will do this using paraview.
- Recall that the integral length scale  $l_0$  is equal to,

$$l_0 = \frac{k^{1.5}}{\epsilon} \quad \text{or} \quad l_0 = \frac{k^{0.5}}{C_\mu \omega} \quad \text{where} \quad C_\mu = 0.09$$

- The ratio of integral length scale to grid length scale  $R_l$  can be computed as follows,

$$R_l = \frac{l_0}{\Delta} \quad \text{where } \Delta \text{ can be approximated as follows } \Delta \approx \sqrt[3]{\text{cell volume}}$$

This approximation is accurate if the aspect ratios are modest (less than 1.2)

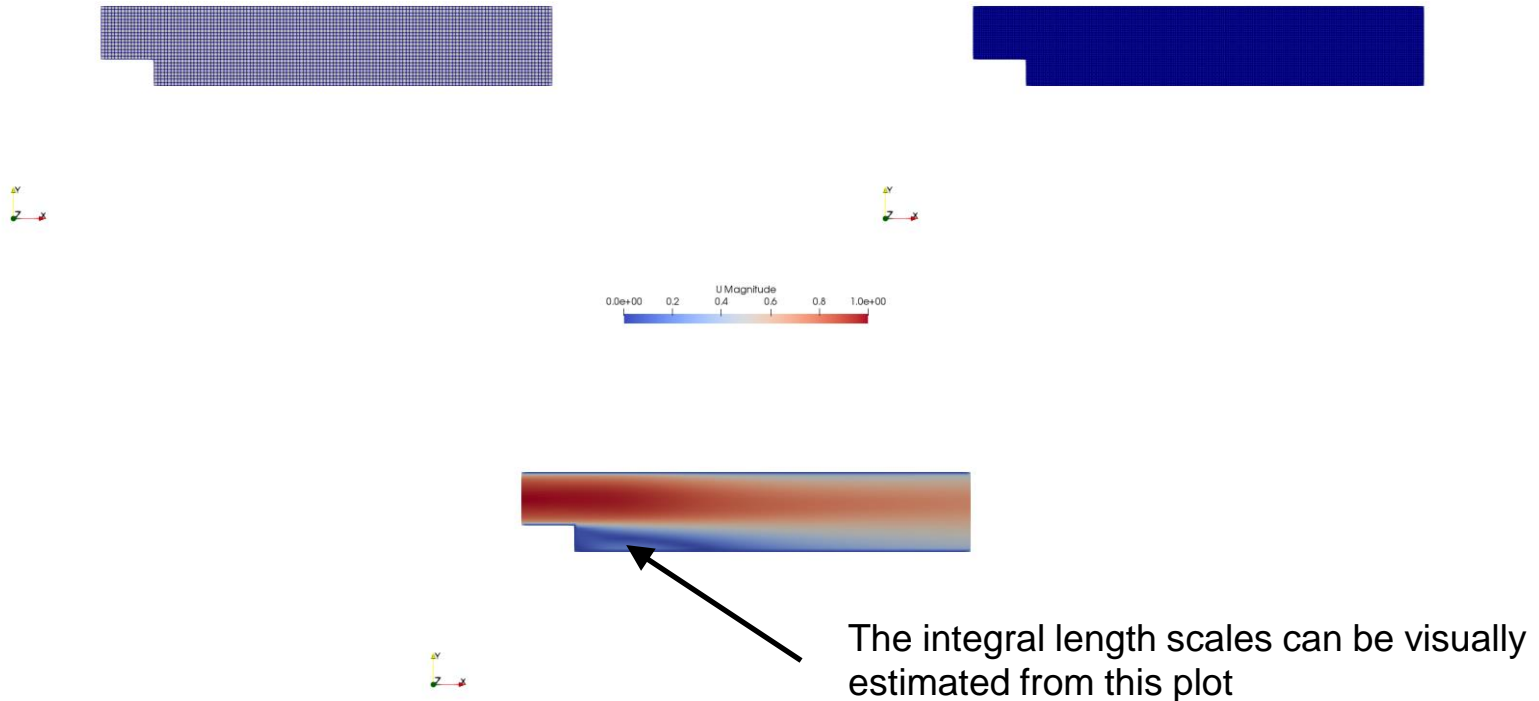
- If you are running a 2D case, it is recommended to set the width of the domain to one.
- The value of  $R_l$  should be  $R_l > 5 - 10$
- To compute the cell volume, you can use the command,
  - `$> postprocess -func writeCellVolumes`



# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

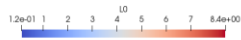
- We will use two meshes.
- And three turbulence models,  $k - \epsilon$ ,  $k - \omega$ , and  $k - \omega$  SST.



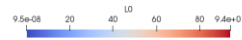
# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

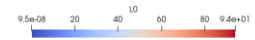
- Integral length scales computed using the coarse mesh for the three turbulence models.



$k - \epsilon$



$k - \omega$

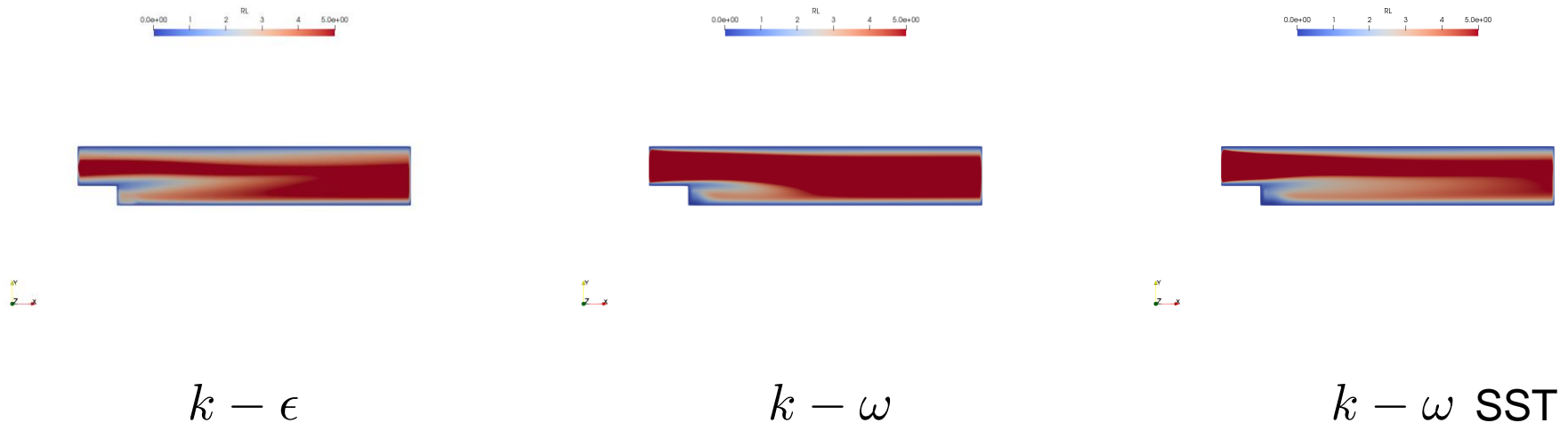


$k - \omega$  SST

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- Ratio of integral length scale to grid length scale computed using the coarse mesh for the three turbulence models.



- Ideally, the value of  $R_l$  should be  $R_l > 5 - 10$
- Regions that do not satisfy this requirement need to be refined.
- Near-wall regions always pose challenges. In these areas is better to quantify the  $y^+$  value.

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- Ratio of integral length scale to grid length scale computed using the  $k - \omega$  SST turbulence model,



Coarse mesh



Fine mesh

# Guided tutorials

## Guided tutorial 3 – Turbulence flow over a backward facing step

- We are going to use the following solver: `simpleFoam`.
- This case is rather simple, but we will use it to explain many features used in OpenFOAM when dealing with turbulence.
- We will see the importance of choosing the right turbulence model for the right mesh.
- After finding the numerical solution, we will do some sampling and data manipulation.
- Then we will do some plotting (using gnuplot or Python) and scientific visualization.
- 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.
- You will find the instructions of how to run this case in the file `README.FIRST` located in the case directory.

# Guided tutorials

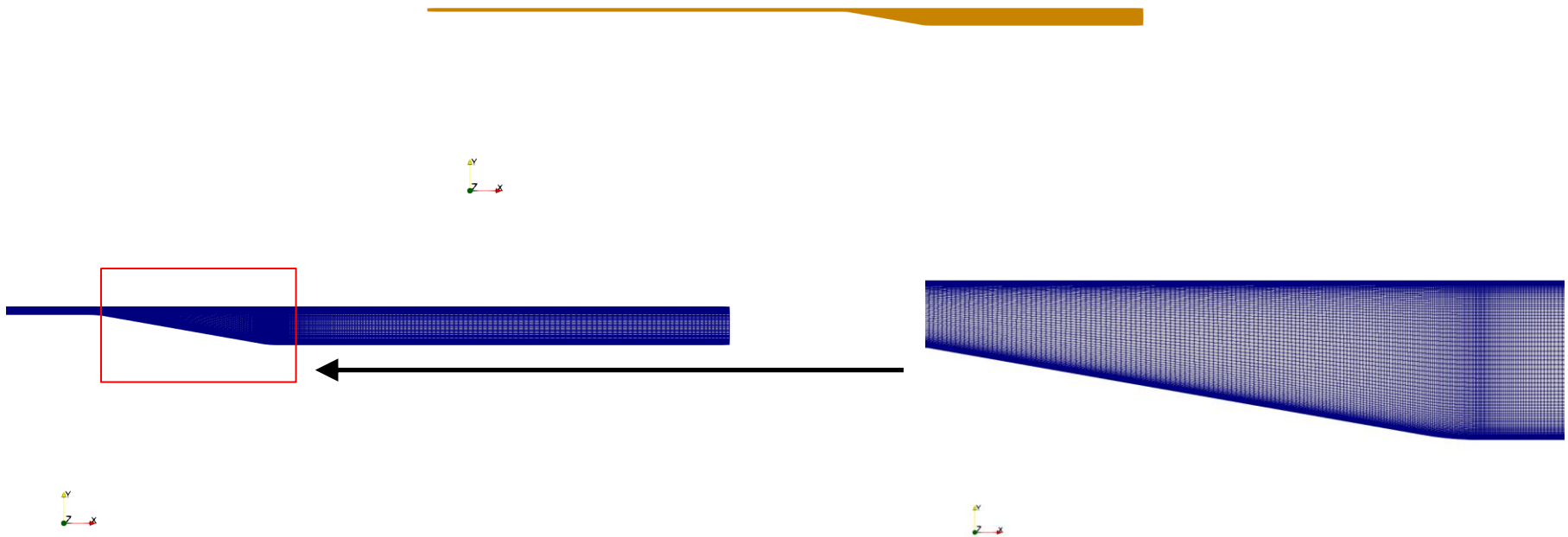
- **Guided tutorial 4.** Buice-Eaton 2D Diffuser.
- This case is ready to run.
- The case is located in the directory:

```
$TM/turbulence/GT4/
```

- In the case directory, you will find the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.
- You will also find a few additional files (or scripts) with the extension `.sh`, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.
- We highly recommend opening the `README.FIRST`, file and typing 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, use the automatic scripts to run the cases.
- From this point on, please follow me. We are all going to work at the same pace.

# Guided tutorials

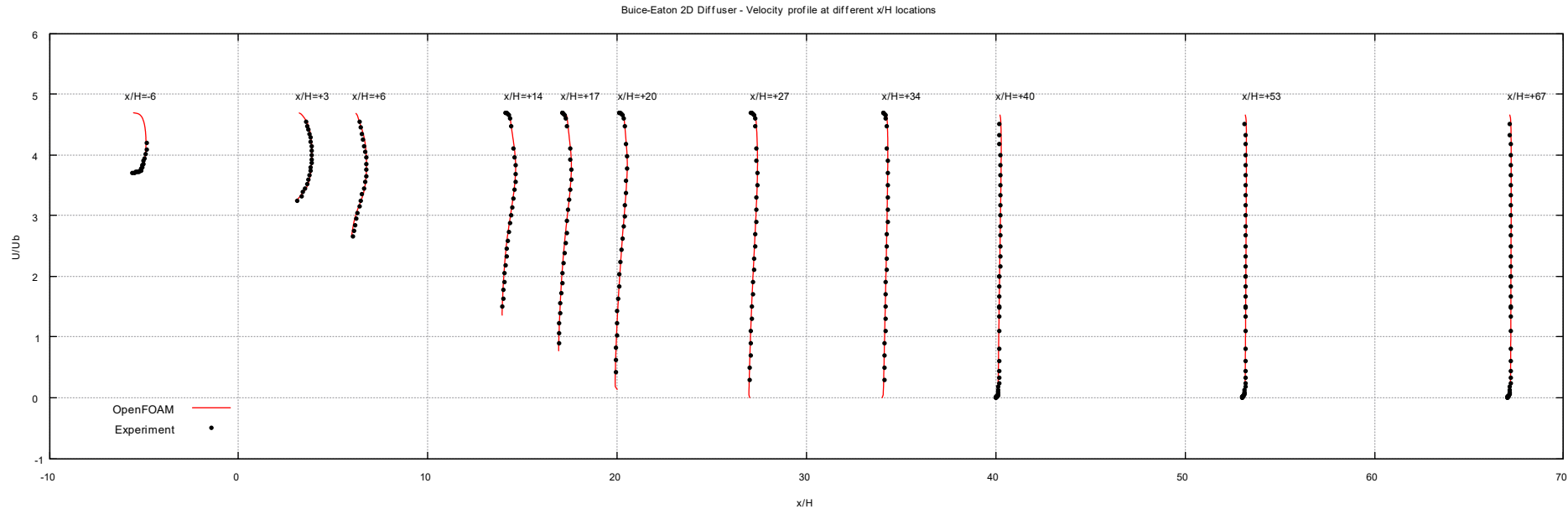
## Guided tutorial 4 – Buice-Eaton 2D Diffuser



- In this guided tutorial we will simulate the separated flow through a 2D asymmetric diffuser (internal flow), also known as the Buice-Eaton 2D diffuser [1].
- We will use different turbulence models.
- This case has plenty of experimental and numerical data for validation.
- We will use a mesh for low-Re methods (we will resolve the boundary layer).

# Guided tutorials

## Guided tutorial 4 – Buice-Eaton 2D Diffuser

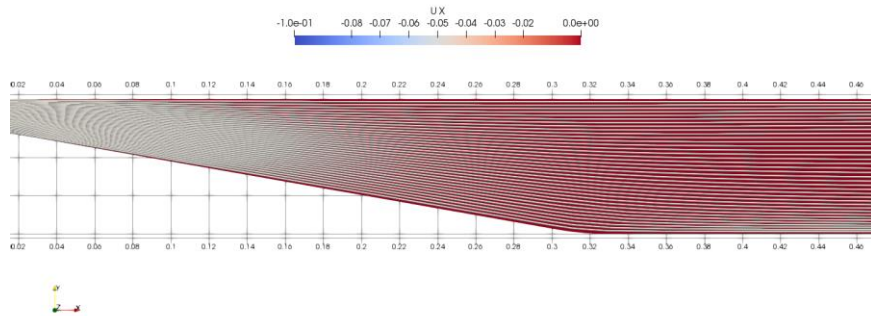


- We will compare the numerical solution with the experimental results at different locations.
- We will play a little bit with the `postProcess` utility to do the sampling at the desired locations.
- We will use `gnuplot` to do some plotting.

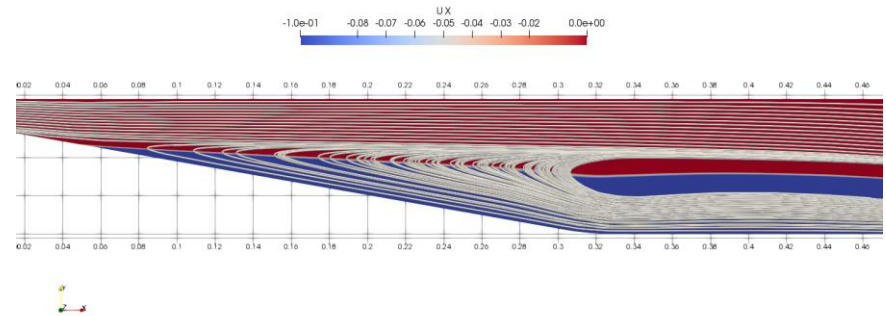


# Guided tutorials

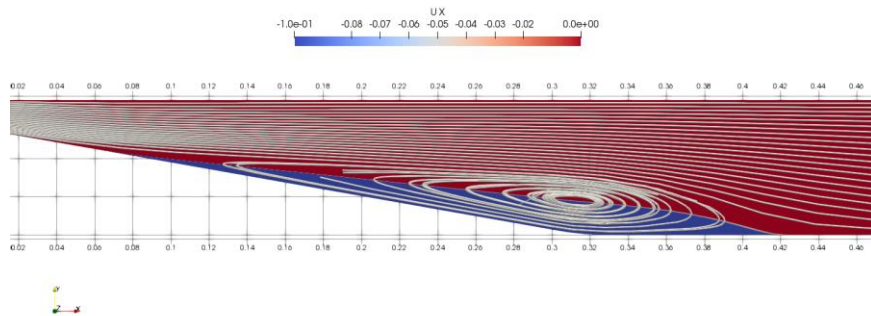
## Guided tutorial 4 – Buice-Eaton 2D Diffuser



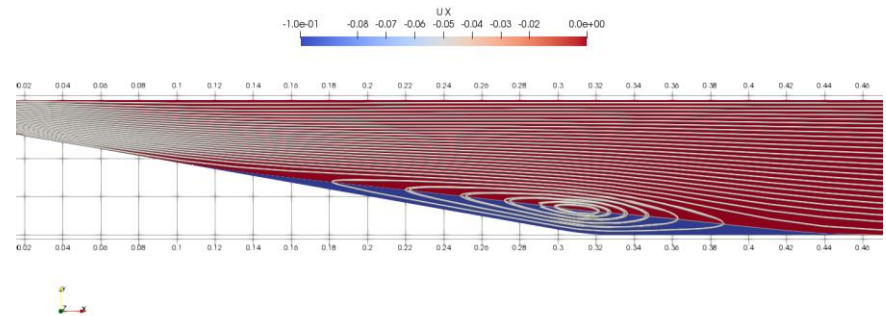
k-epsilon



k-epsilon realizable



k-omega SST



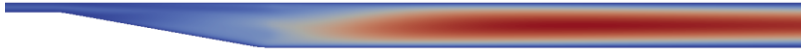
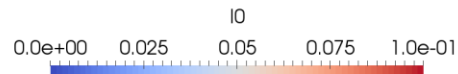
Spalart-Allmaras

- We will also compare the ability of different turbulence model to capture separation/attachment points and recirculation regions.

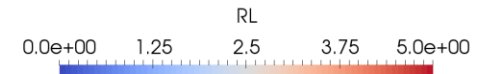
# Guided tutorials

## Guided tutorial 4 – Buice-Eaton 2D Diffuser

- We will also compute the integral length scales and grid length scales using paraview.

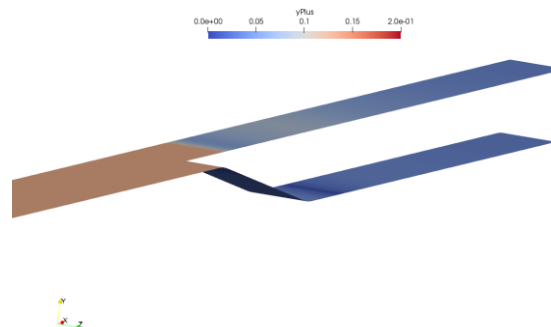


Integral length scales



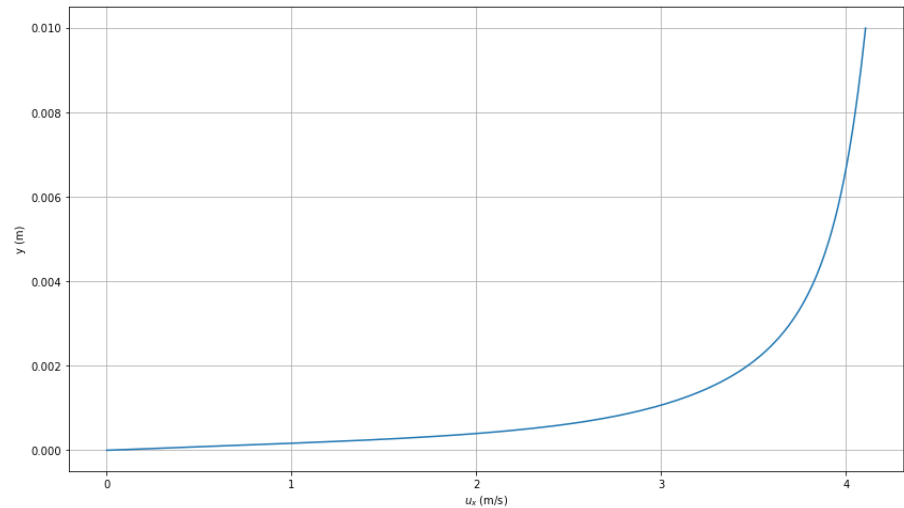
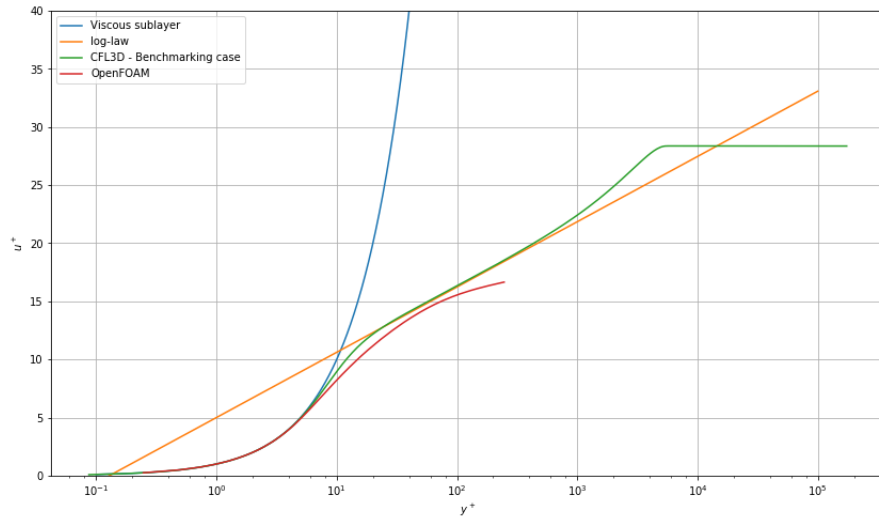
Ratio of integral length scales to grid length scales

- And we will plot the  $y^+$  value at the walls.



# Guided tutorials

## Guided tutorial 4 – Buice-Eaton 2D Diffuser

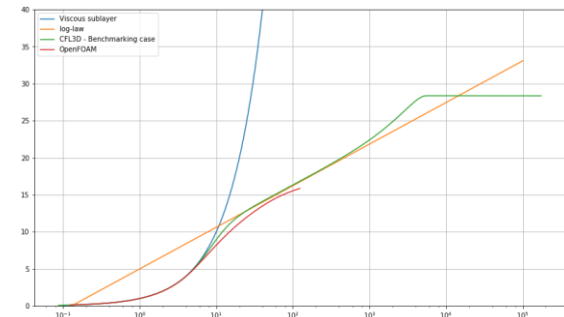
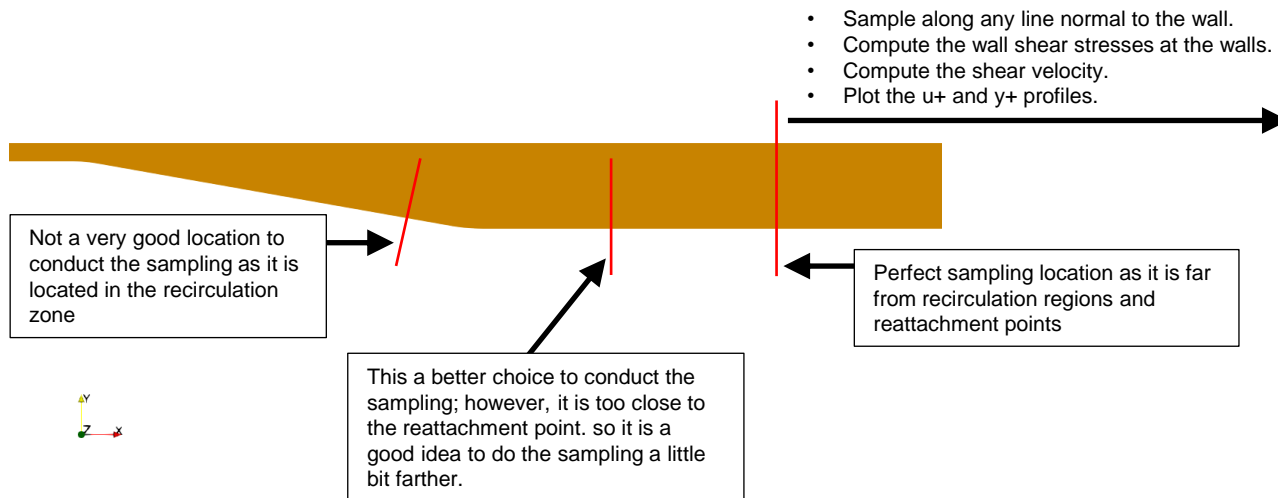


- And we will do more validation.
- We will plot the velocity in terms of the non-dimensional variables  $u^+$  and  $y^+$ .
- Remember, this profile is kind of universal (if you are fine using word universal). So, wherever you sample your data, you should be able to reproduce the profile.
- Have in mind that you should do the sampling where the boundary layer is fully developed. That is, if you do sampling in regions where there is recirculation or flow separation, you will not be able to reproduce the profile.

# Guided tutorials

## Guided tutorial 4 – Buice-Eaton 2D Diffuser

- Let us explain again how to plot the velocity in terms of the non-dimensional variables  $u^+$  and  $y^+$ .
- First, pick a location where to do the sampling (avoid recirculation areas or regions with flow separation), and sample the velocity normal to the wall.
- At the same location sample the wall shear stresses and compute the magnitude.
- With the sampled values, you can compute the shear velocity,  $u^+$  and  $y^+$ .
- Remember, when computing  $y^+$  you need to use the distance normal to wall  $y$ .
  - Disregarding of the location in the domain, this distance should start from zero.
- In the directory `$TM/turbulence/GT4/python`, you will find a python script to plot the velocity profile.
- To sample the wall shear stresses you will need to use the utility `wallShearStress`.



# Guided tutorials

## Guided tutorial 4 – Buice-Eaton 2D Diffuser

- We are going to use the following solver: `simpleFoam`.
- After finding the numerical solution, we will do some sampling and data manipulation.
- Then we will do some plotting (using gnuplot or Python) and scientific visualization with paraview.
- 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.
- You will find the instructions of how to run this case in the file `README.FIRST` located in the case directory.
- In the NASA NPARC Alliance Verification and Validation Archive site <https://www.grc.nasa.gov/WWW/wind/valid/buice/buice01/buice01.html>, you will find the complete details of the Buice-Eaton 2D diffuser Validation Case. You can download the mesh and numerical results in this site.

# Guided tutorials

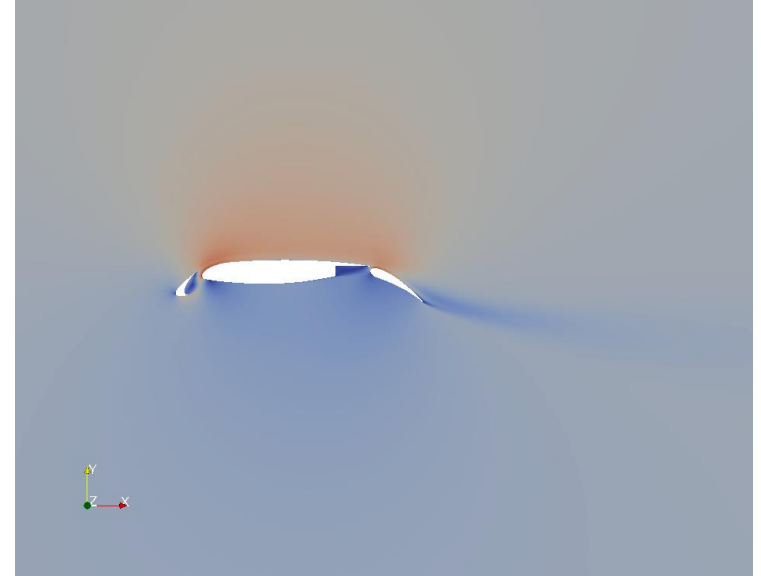
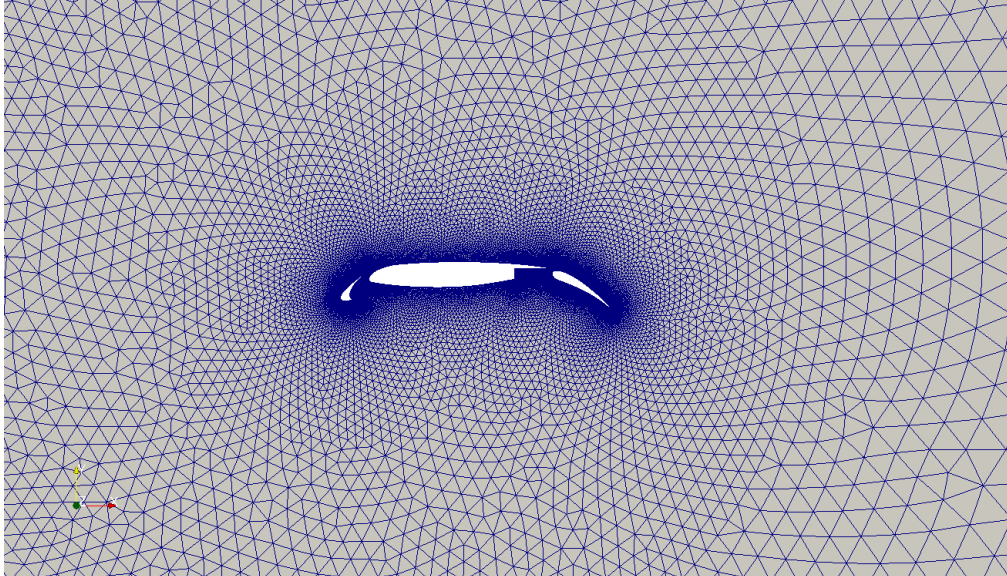
- **Guided tutorial 5.** RANS multielement airfoil.
- This case is ready to run.
- The case is located in the directory:

```
$TM/turbulence/GT5/
```

- In the case directory, you will find the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.
- You will also find a few additional files (or scripts) with the extension `.sh`, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.
- We highly recommend opening the `README.FIRST`, file and typing 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, use the automatic scripts to run the cases.
- From this point on, please follow me. We are all going to work at the same pace.

# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil

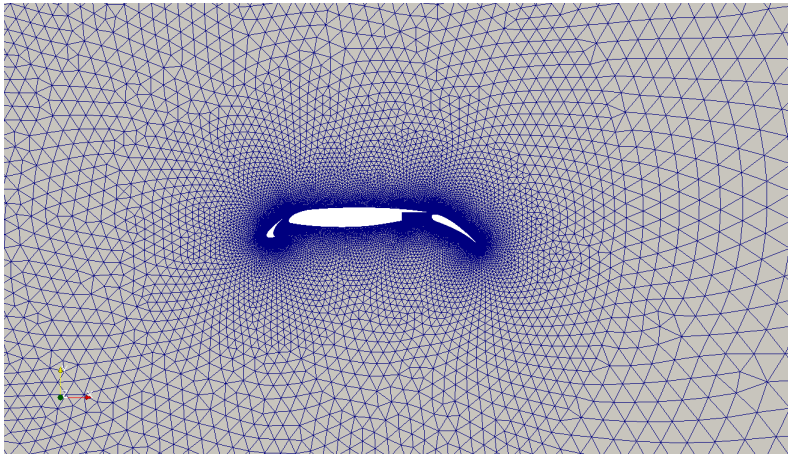


- In this guided tutorial we will simulate a turbulent flow using RANS.
- We will simulate the flow pass a multi-element airfoil. This case has plenty of experimental and numerical data for validation.
- We will study the dependence of the turbulence model on the element type and near the wall mesh.

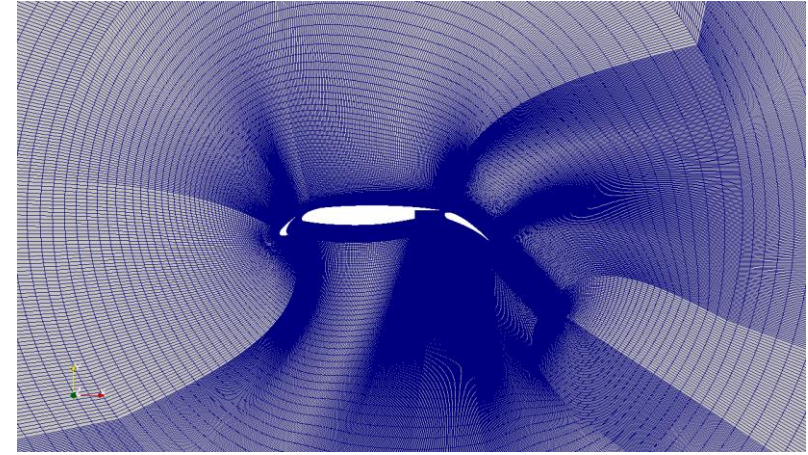


# Guided tutorials

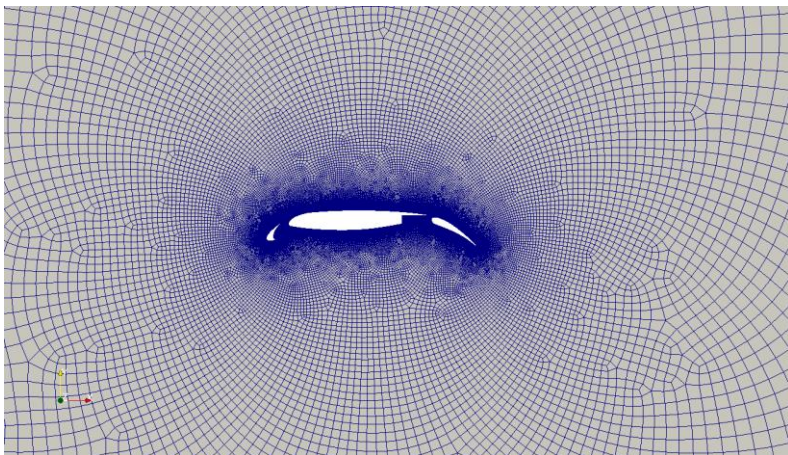
## Guided tutorial 5 – RANS multielement airfoil



Mesh 1. Triangles in the far field, stretched quads near the wall.



Mesh 2. Multiblock structured mesh.



Mesh 3. Quads in the far field, stretched quads near the wall.

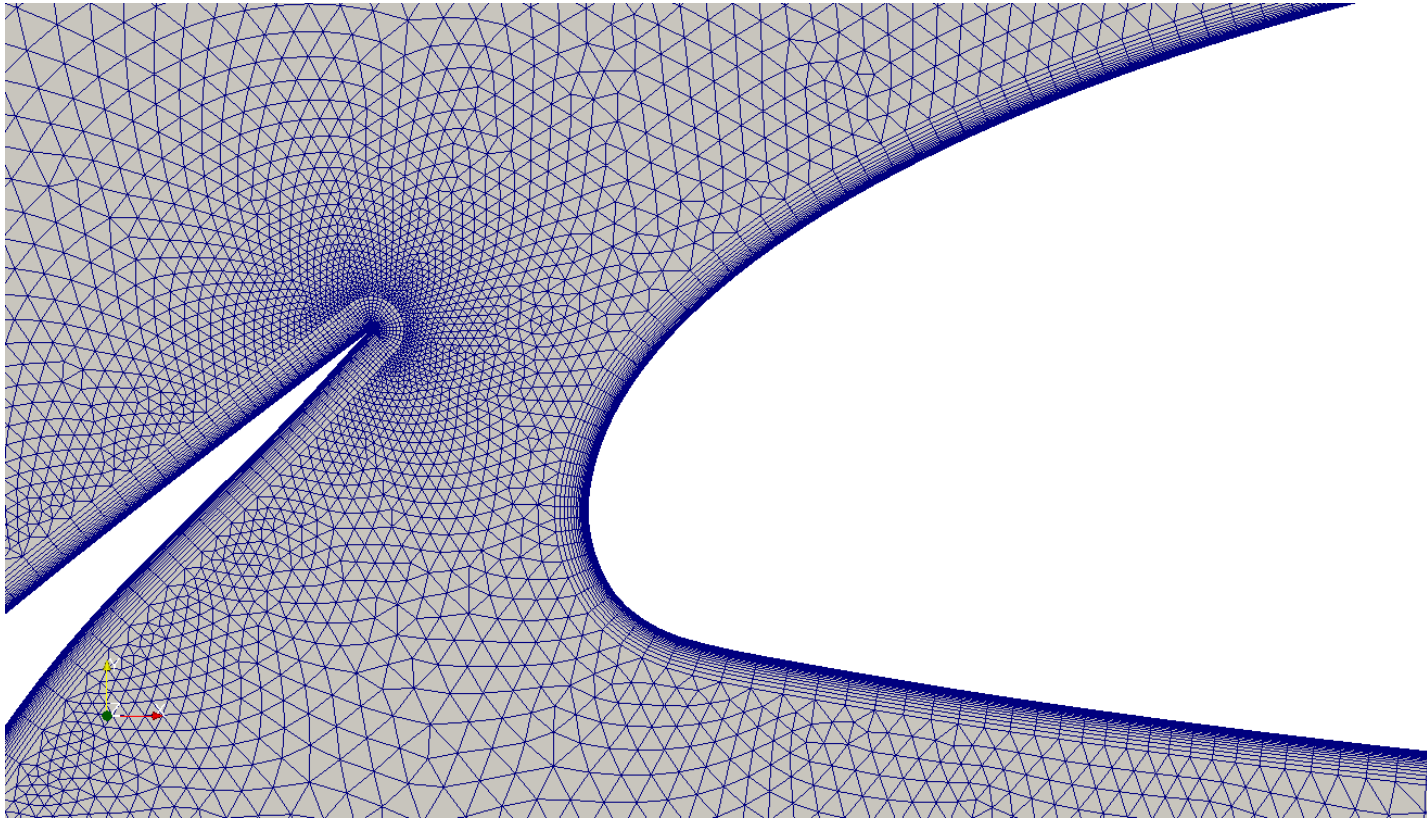
### Note

In 3D the triangles will become pyramids and tetrahedrons, the quads will become hexahedrons, and the stretched quads will become prisms.



# Guided tutorials

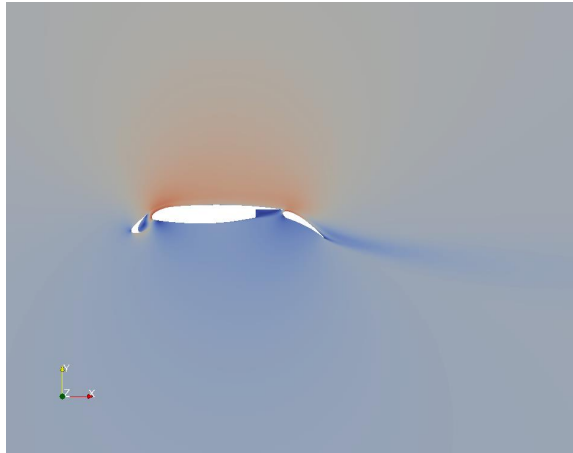
## Guided tutorial 5 – RANS multielement airfoil



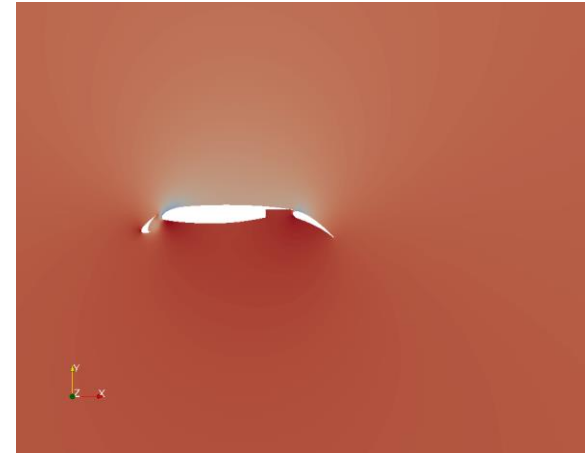
Near the wall mesh, inflation layer or boundary layer mesh.  
The mesh is done in such a way that the average  $y^+$  is approximately 0.2

# Guided tutorials

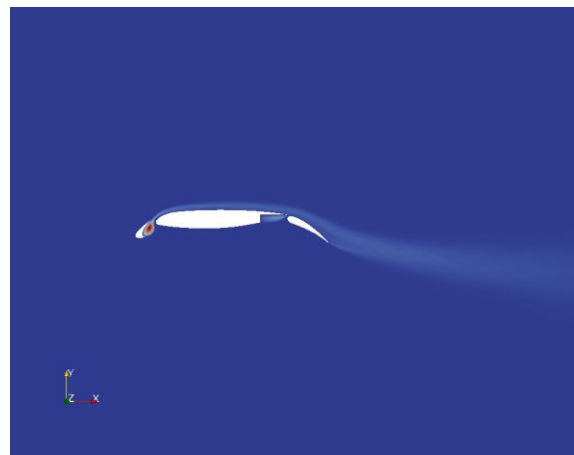
## Guided tutorial 5 – RANS multielement airfoil



Velocity field



Pressure field



Turbulent viscosity field

# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil



The following results need to be updated.  
They were obtained using OpenFOAM 7

QOI	Mesh 1	Mesh 2	Mesh 3	Ref. value
$c_d$	0.0385	0.03808	0.0375	0.043
$c_l$	2.237	2.2401	2.2547	2.18
$\overline{y^+}$	0.2	0.2	0.2	Not applicable
Iterations	537	2000 (did not converged)	858	Not applicable

**Note:** 0° AOA – Spalart-Allmaras

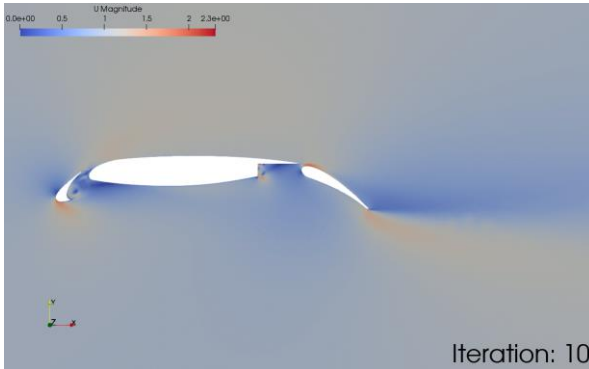
### References:

[1] Daryl L. Bonhaus, W. Kyle Anderson, Dimitri J. Mavriplis. Numerical Study To Assess Sulfur Hexafluoride as a Medium for Testing Multielement Airfoils. NASA Technical Paper 3496.

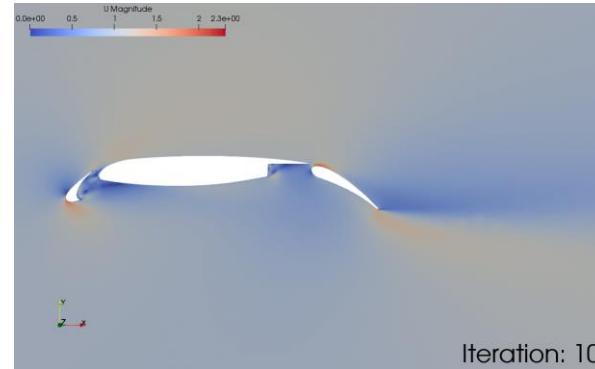
[2] Steven M. Klausmeyer, John C. Lin. Comparative Results From a CFD Challenge Over a 2D Three-Element High-Lift Airfoil. NASA Technical Memorandum 112858.

# Guided tutorials

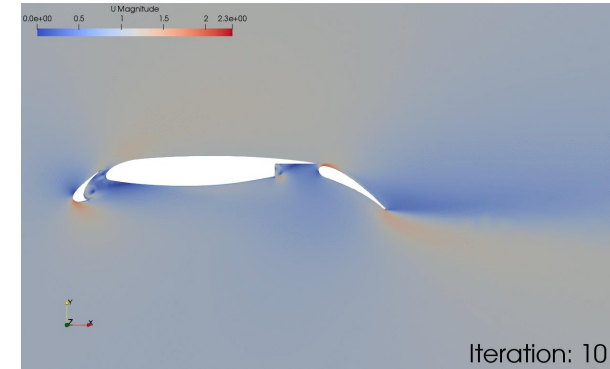
## Guided tutorial 5 – RANS multielement airfoil



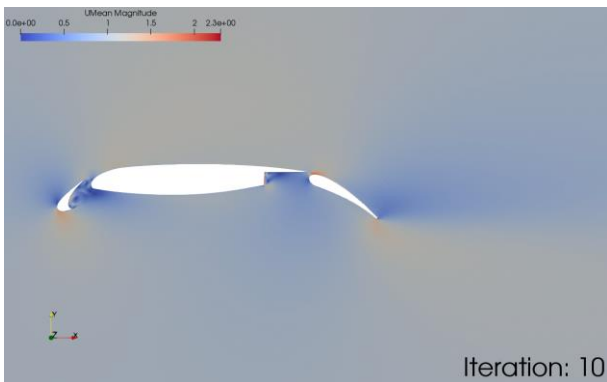
Laminar  
Instantaneous U field



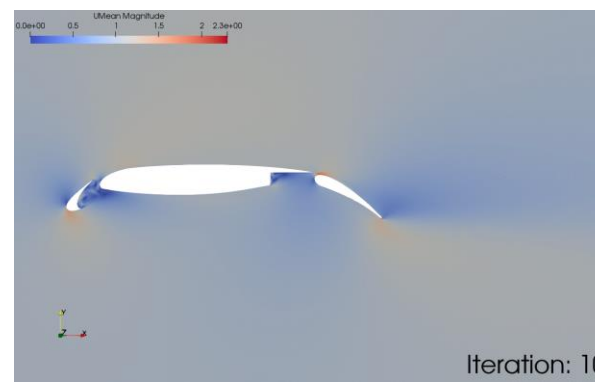
Spalart-Allmaras  
Instantaneous U field



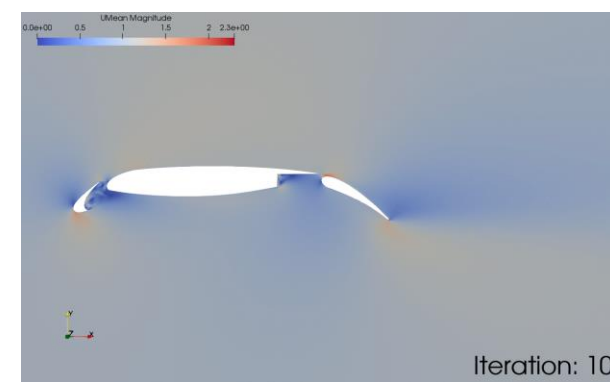
k-omega SST  
Instantaneous U field



Laminar  
Mean U field



Spalart-Allmaras  
Mean U field

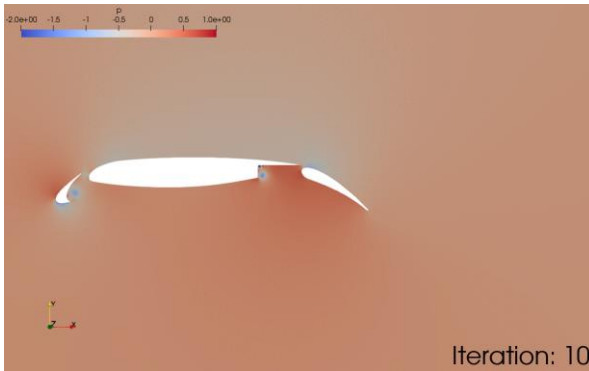


k-omega SST  
Mean U field

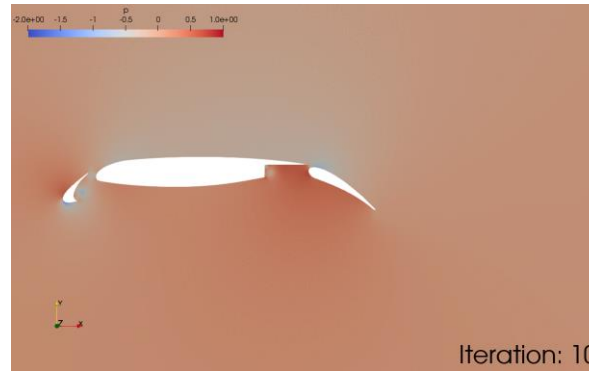
**0° AOA – Incompressible**

# Guided tutorials

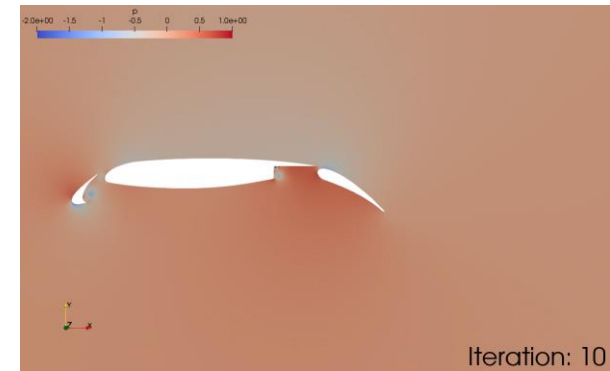
## Guided tutorial 5 – RANS multielement airfoil



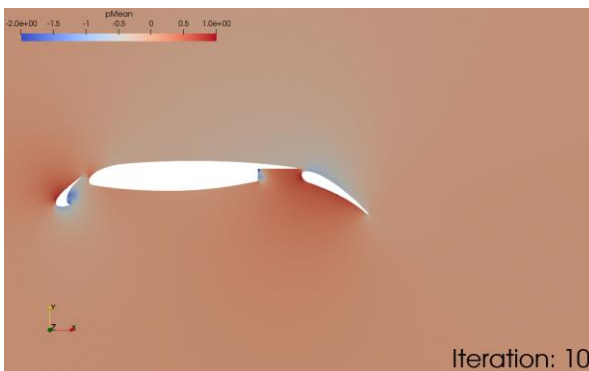
Laminar  
Instantaneous p field



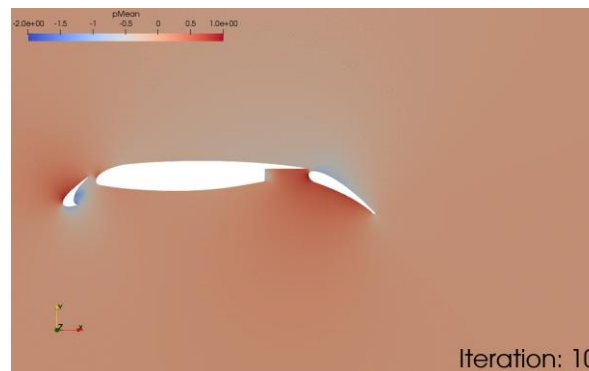
Spalart-Allmaras  
Instantaneous p field



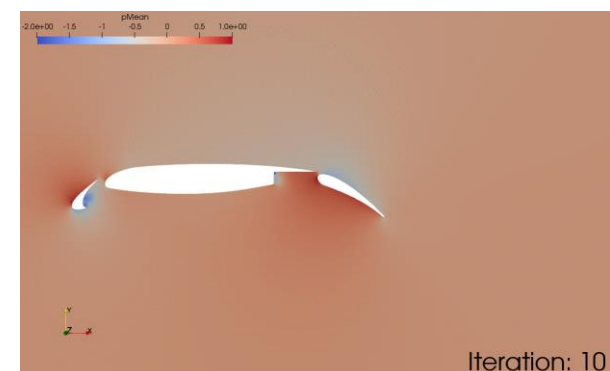
k-omega SST  
Instantaneous p field



Laminar  
Mean p field



Spalart-Allmaras  
Mean p field

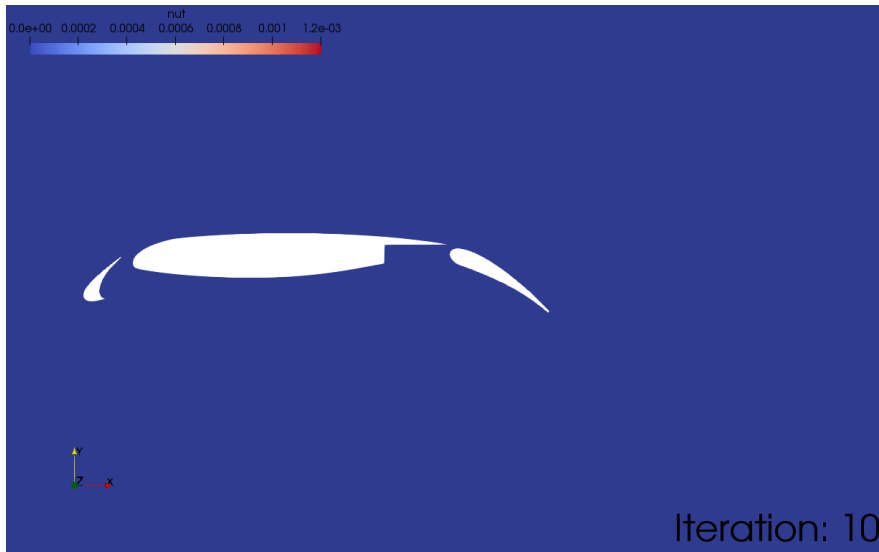


k-omega SST  
Mean p field

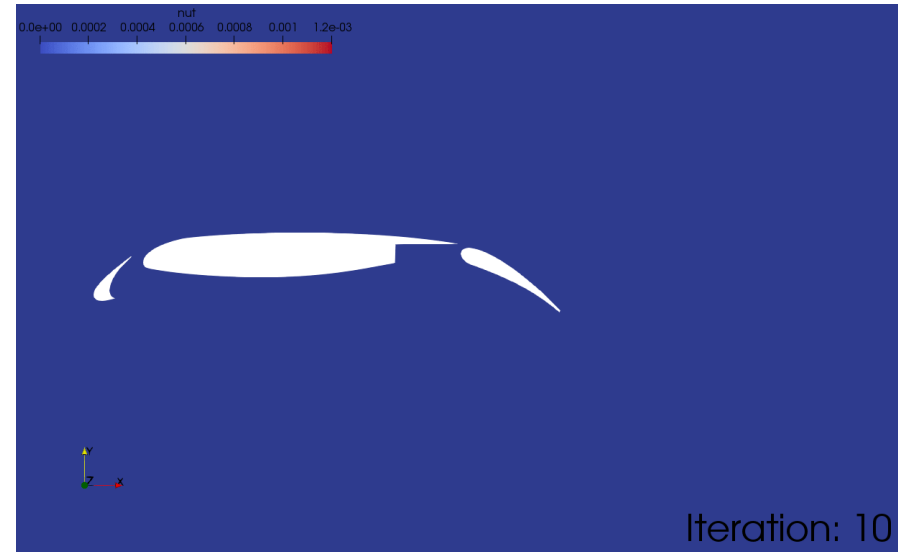
**0° AOA – Incompressible**

# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil



Spalart-Allmaras  
Instantaneous nut field

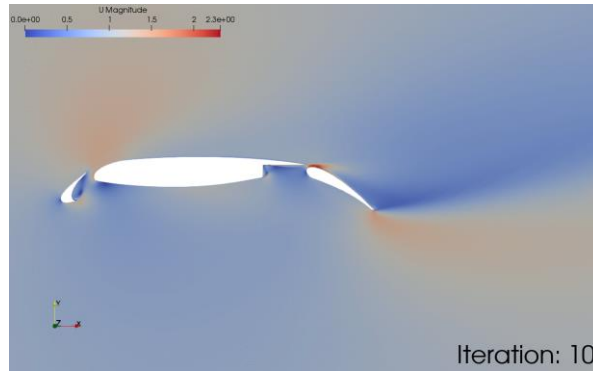


k-omega SST  
Instantaneous nut field

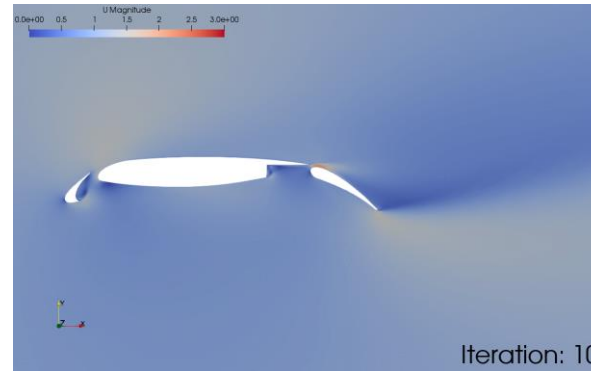
**0° AOA – Incompressible**

# Guided tutorials

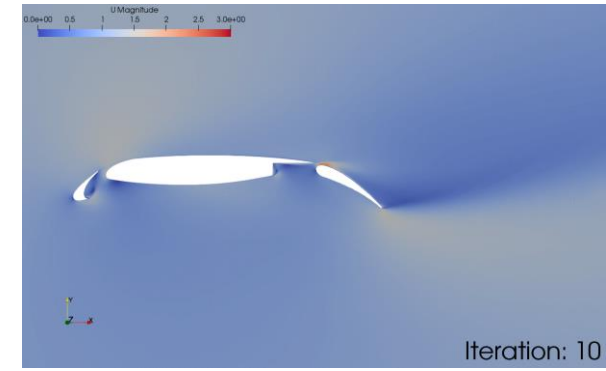
## Guided tutorial 5 – RANS multielement airfoil



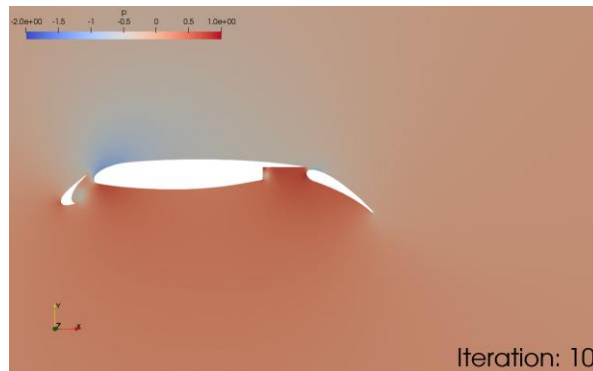
Laminar  
Instantaneous U field



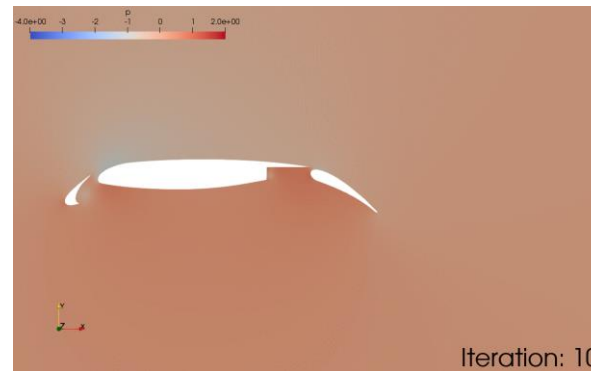
Spalart-Allmaras  
Instantaneous U field



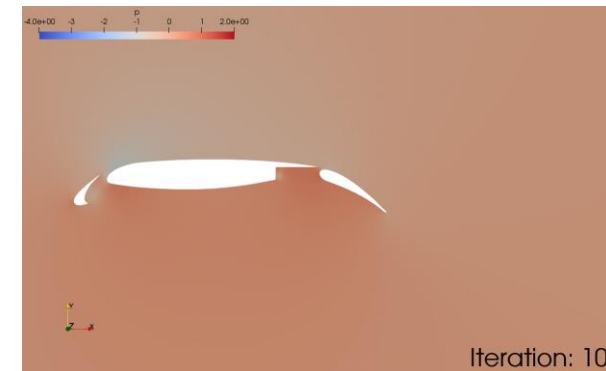
K-omega SST  
Instantaneous U field



Laminar  
Instantaneous p field



Spalart-Allmaras  
Instantaneous p field

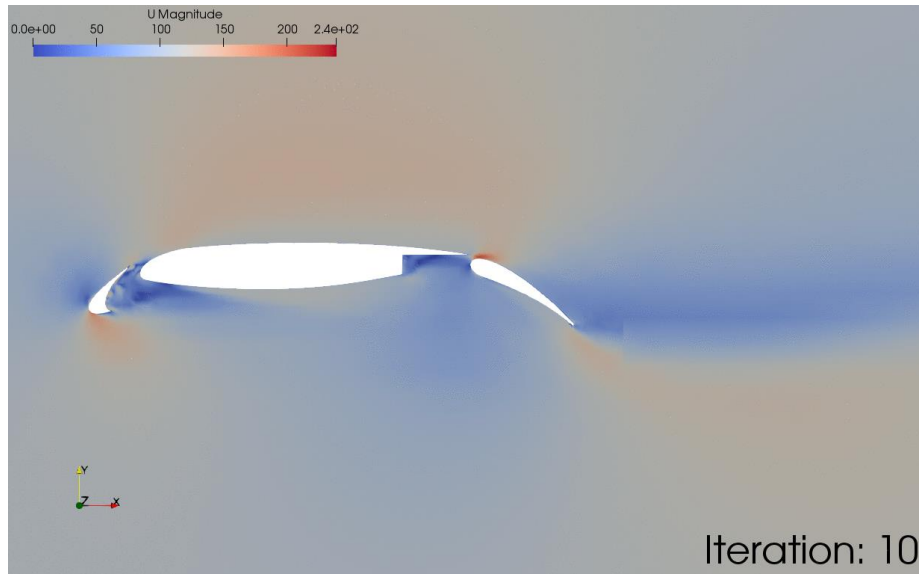


K-omega SST  
Instantaneous p field

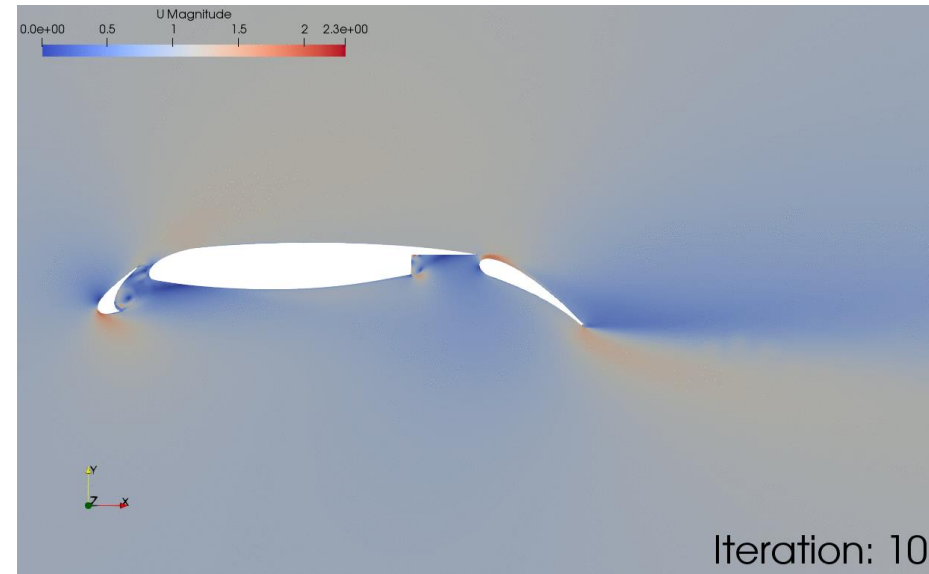
**16° AOA – Incompressible**

# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil



K-omega SST  
Instantaneous U field  
**Compressible solver**  
High-Re approach  
 $c_l = 2.24$   
 $c_d = 0.0492$



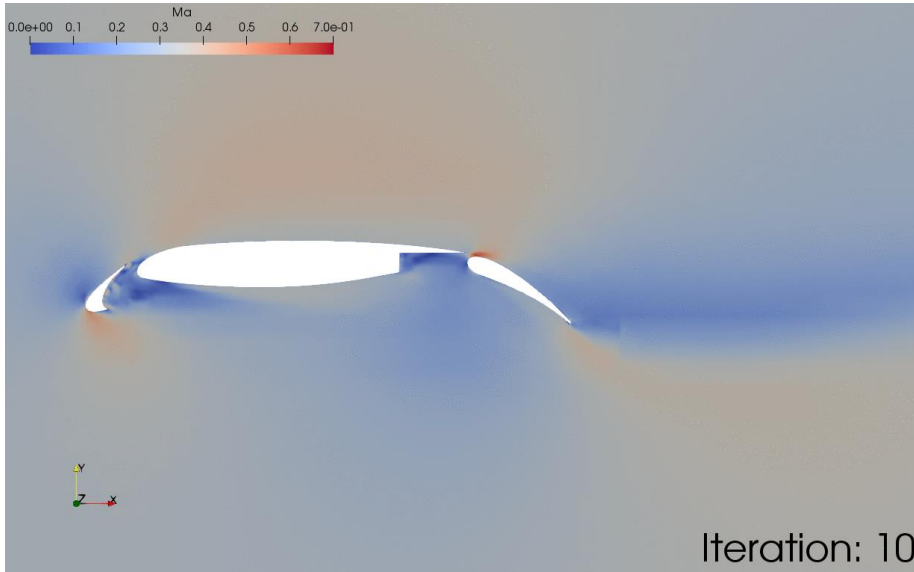
K-omega SST  
Instantaneous U field  
**Incompressible solver**  
Low-Re approach  
 $c_l = 2.23$   
 $c_d = 0.0385$

### 0° AOA – Compressible vs. Incompressible

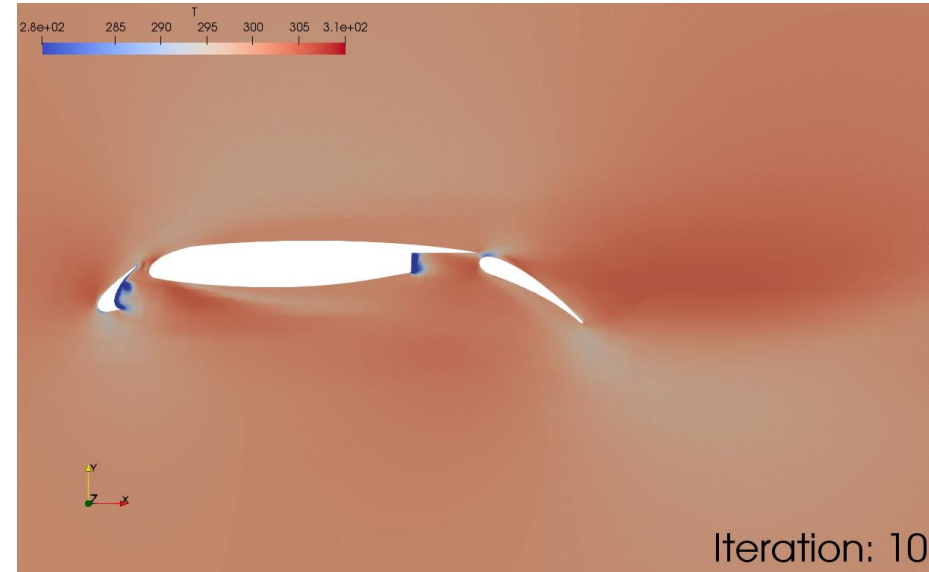


# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil



K-omega SST  
Instantaneous Mach number field



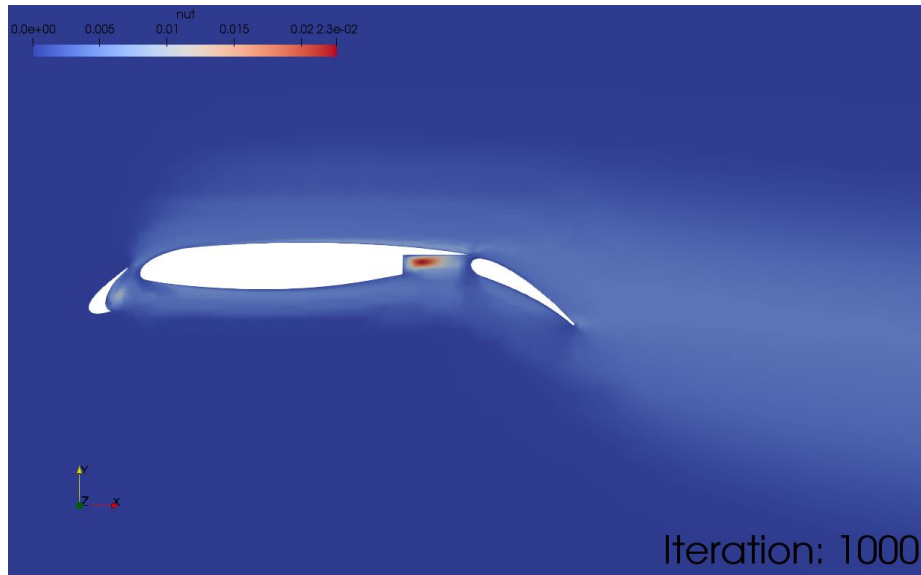
K-omega SST  
Instantaneous temperature field

- Remember, temperature will cause variation in density, viscosity and pressure.
- In compressible flows the equation of state is important.

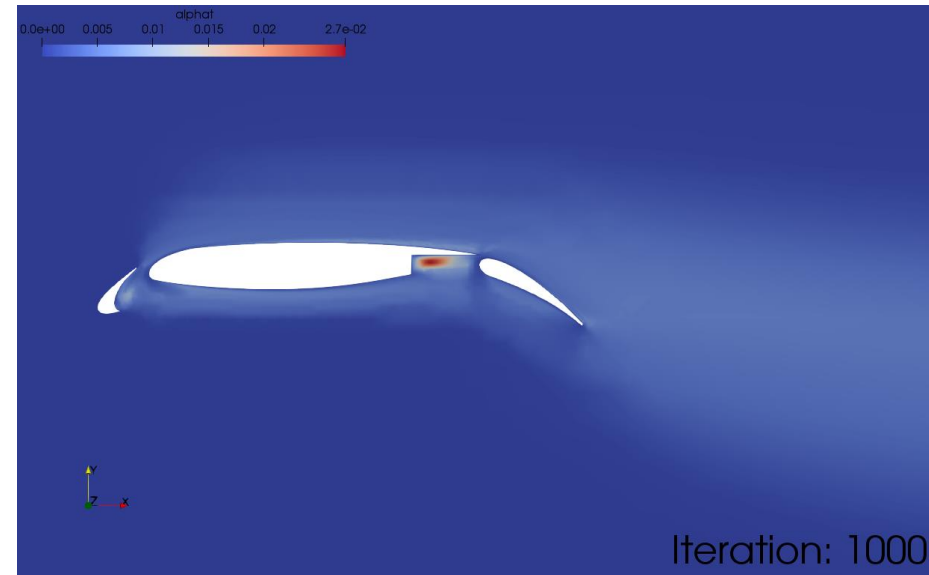
**0° AOA – Compressible**

# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil



K-omega SST  
Turbulent viscosity

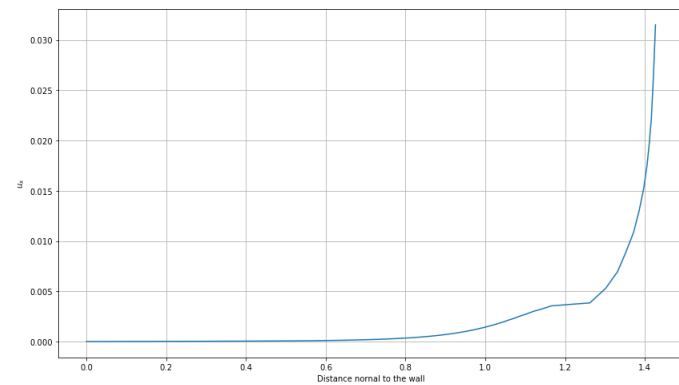
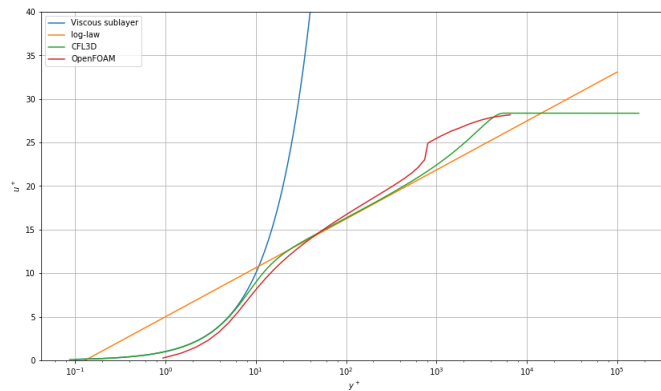
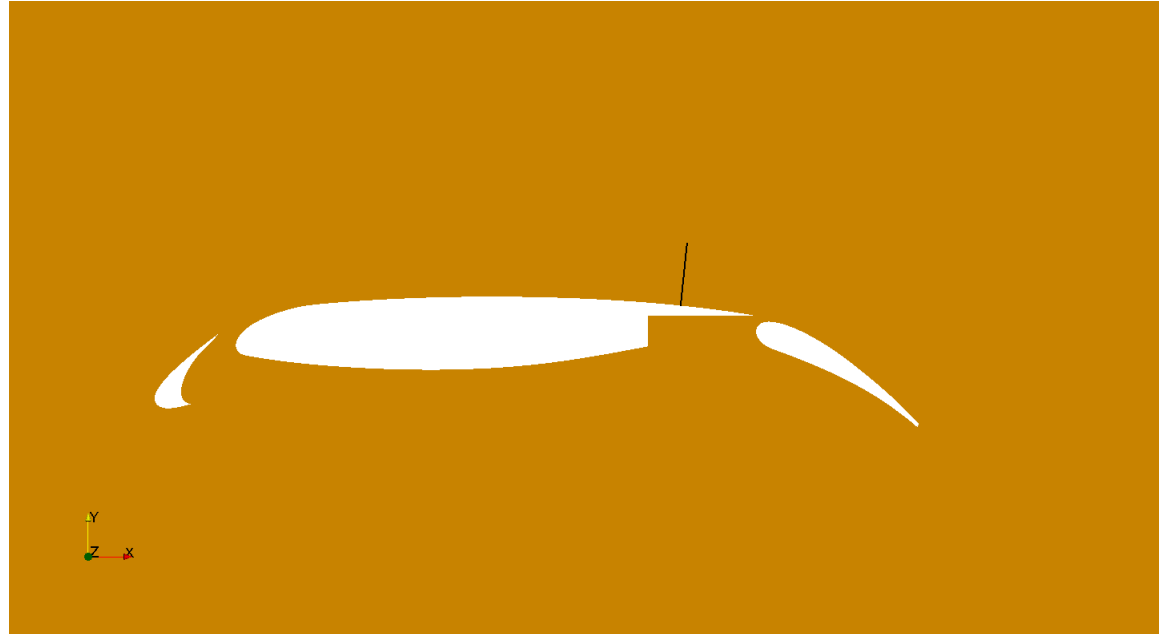


K-omega SST  
Thermal diffusivity

**0° AOA – Compressible**

# Guided tutorials

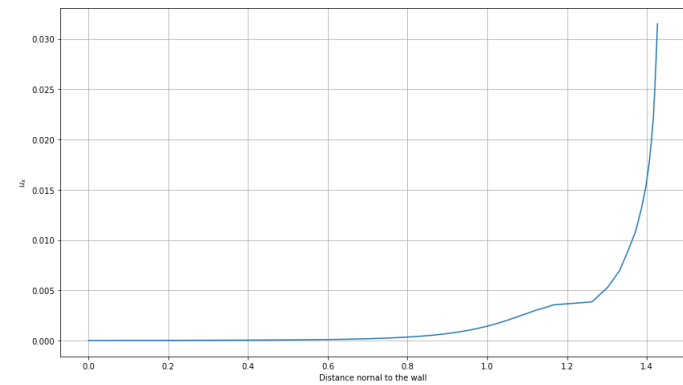
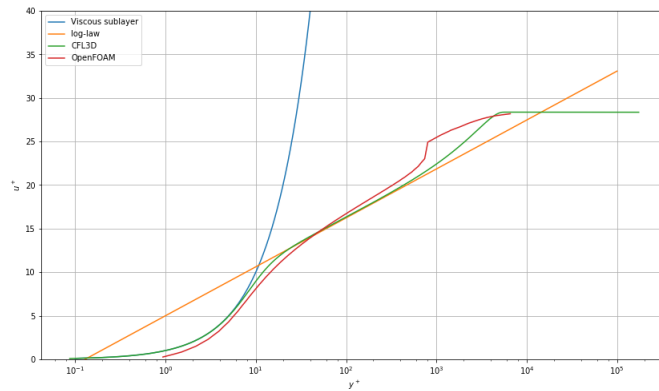
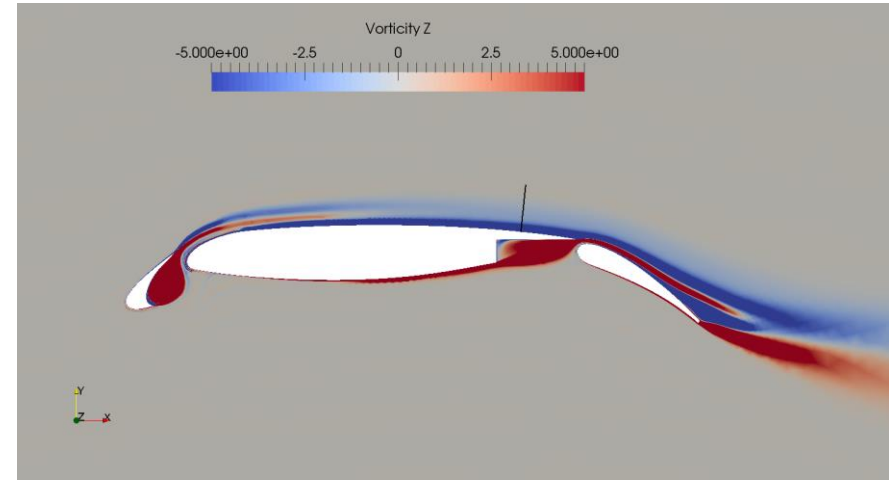
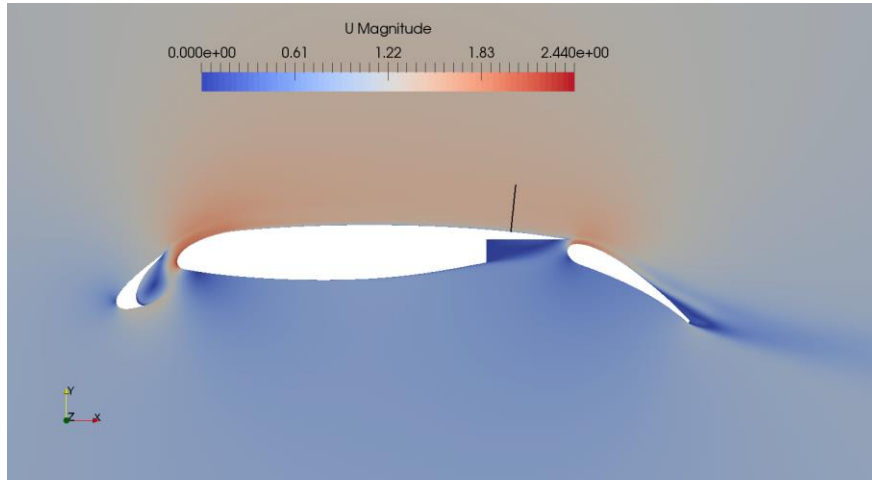
## Guided tutorial 5 – RANS multielement airfoil



- And we should be able to recover the velocity profiles

# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil



- And we should be able to recover the velocity profiles

# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil

- We are going to use the following solvers: `simpleFoam` (for incompressible RANS) and `rhoPimpleFoam` (for compressible URANS).
- We will study the dependence of the turbulence model on the element type and near the wall mesh.
- After finding the numerical solution, we will do some sampling and data manipulation.
- Then we will do some plotting (using gnuplot or Python) and scientific visualization.
- 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.
- You will find the instructions of how to run this case in the file `README.FIRST` located in the case directory.

# Guided tutorials

## Guided tutorial 5 – RANS multielement airfoil

- If you are dealing with compressible flows or heat transfer, you will need to set the initial conditions and boundary conditions for the thermal boundary layer (turbulent thermal diffusivity).
- You also need to set the thermo-physical properties of the working fluid in the dictionary *constant/thermophysicalProperties*. Do not forget to choose how to treat the viscosity, you can choose between constant or the Sutherland model.
- As usual, you need to set the numerical schemes and linear solvers to use for the new field variables (**T** and **alphat**).
- If you plan to use wall functions, you should set wall functions for **alphat** (turbulent thermal diffusivity). These wall functions are valid for lowRE and highRE approaches.
- The rest of the case setup is exactly the same as for incompressible flows.
- We have found that it is a little bit tricky to get a stable solution using a low-RE approach when dealing with high-speed compressible flows and steady solvers. In this case, it is better to use local-time stepping (LTS).
- In the directory `$TM/turbulence/GT4/mesh1/case0_compressible`, you will find a setup of a compressible case using a high-RE approach.

# Guided tutorials

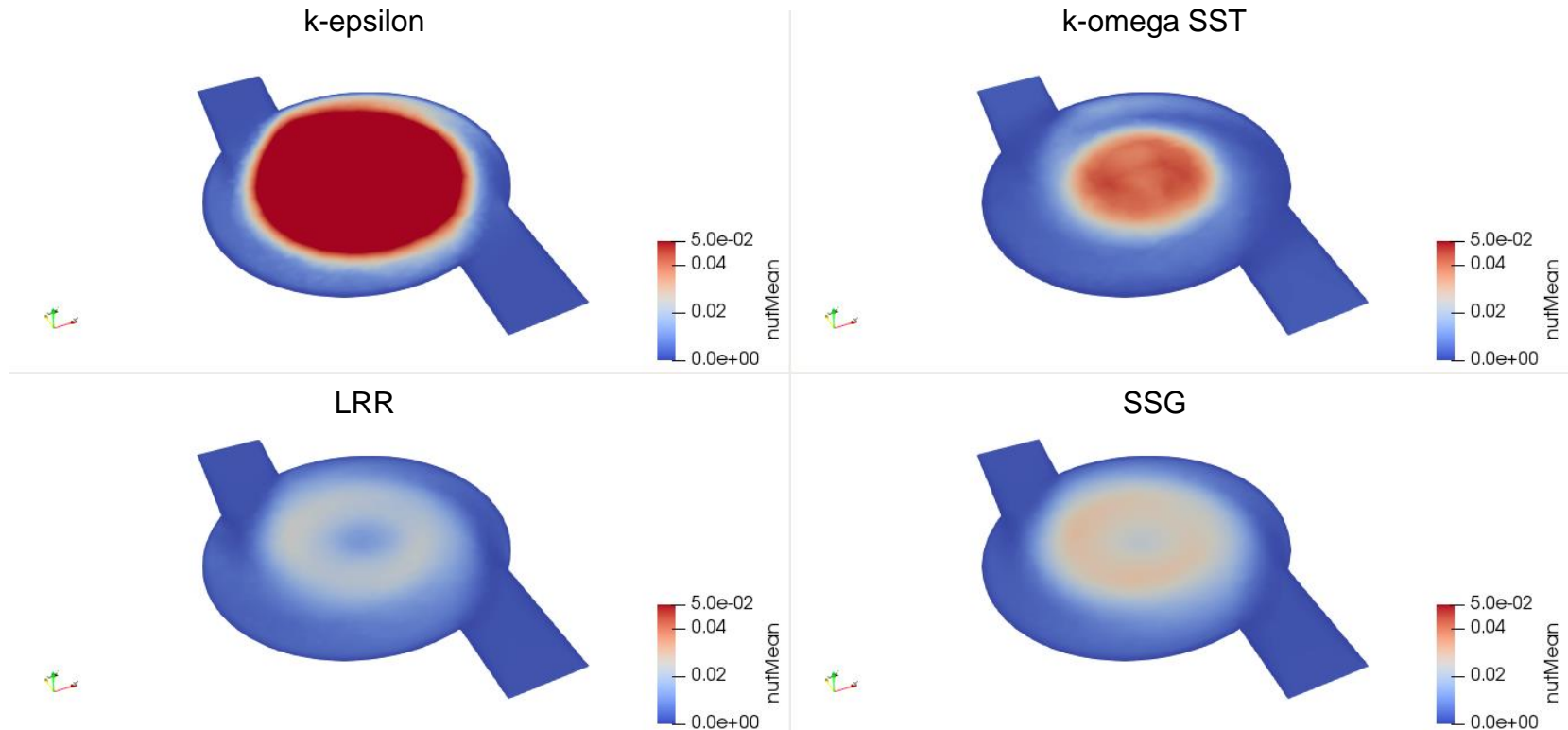
- **Guided tutorial 6.** Static mixer – Reynolds Stress Models (RSM).
- This case is ready to run.
- The case is located in the directory:

```
$TM/turbulence/GT6/
```

- In the case directory, you will find the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.
- You will also find a few additional files (or scripts) with the extension `.sh`, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.
- We highly recommend opening the `README.FIRST`, file and typing 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, use the automatic scripts to run the cases.
- From this point on, please follow me. We are all going to work at the same pace.

# Guided tutorials

## Guided tutorial 6 – Static mixer – Reynolds stress model (RSM)

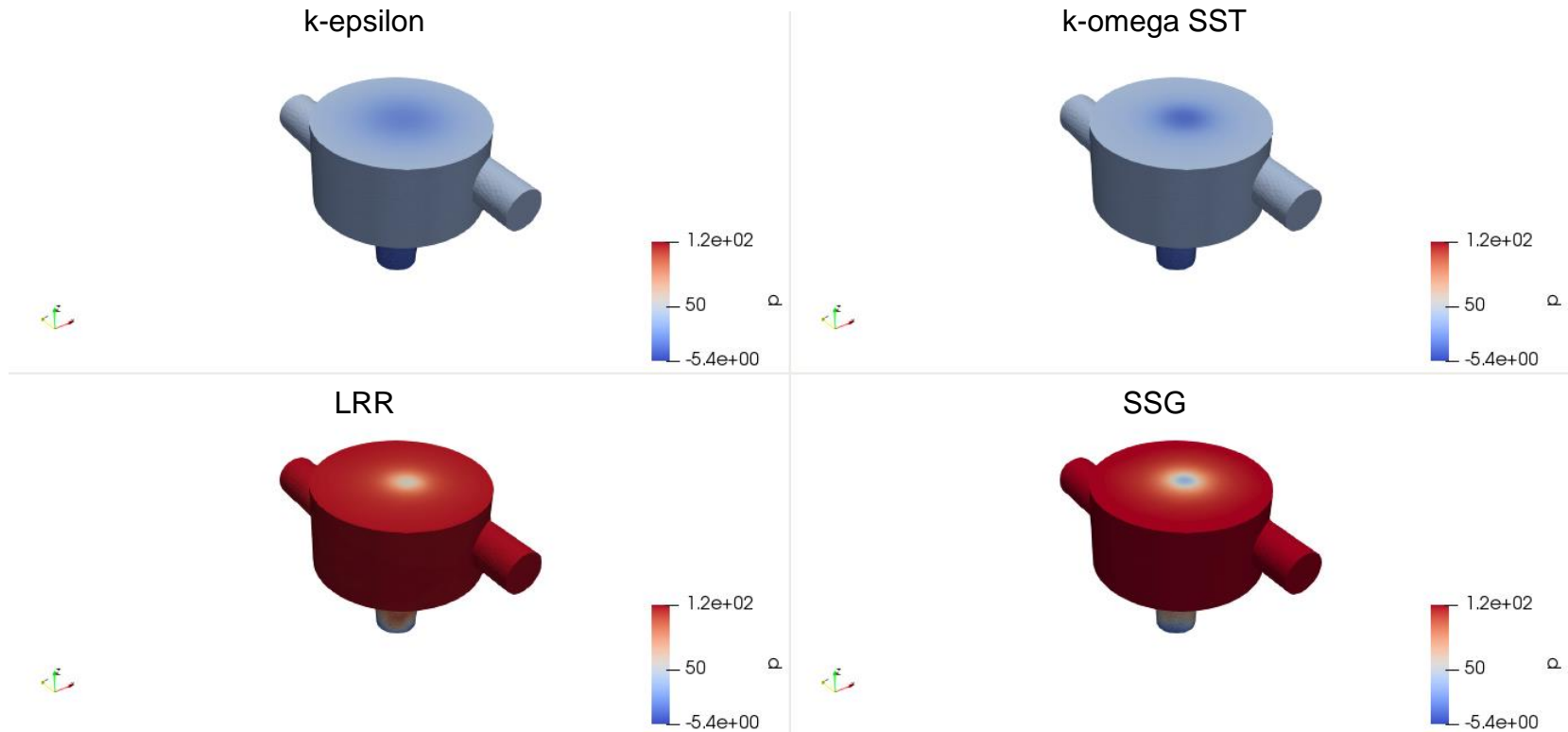


- In this guided tutorial we will simulate a turbulent flow using RSM.
- We will simulate the flow inside a static mixer.
- This problem is characterized with strong system rotation.
- With highly swirling flows and wall curvature, RANS models tend to overpredict turbulent viscosity and fail at resolving the velocity profiles.



# Guided tutorials

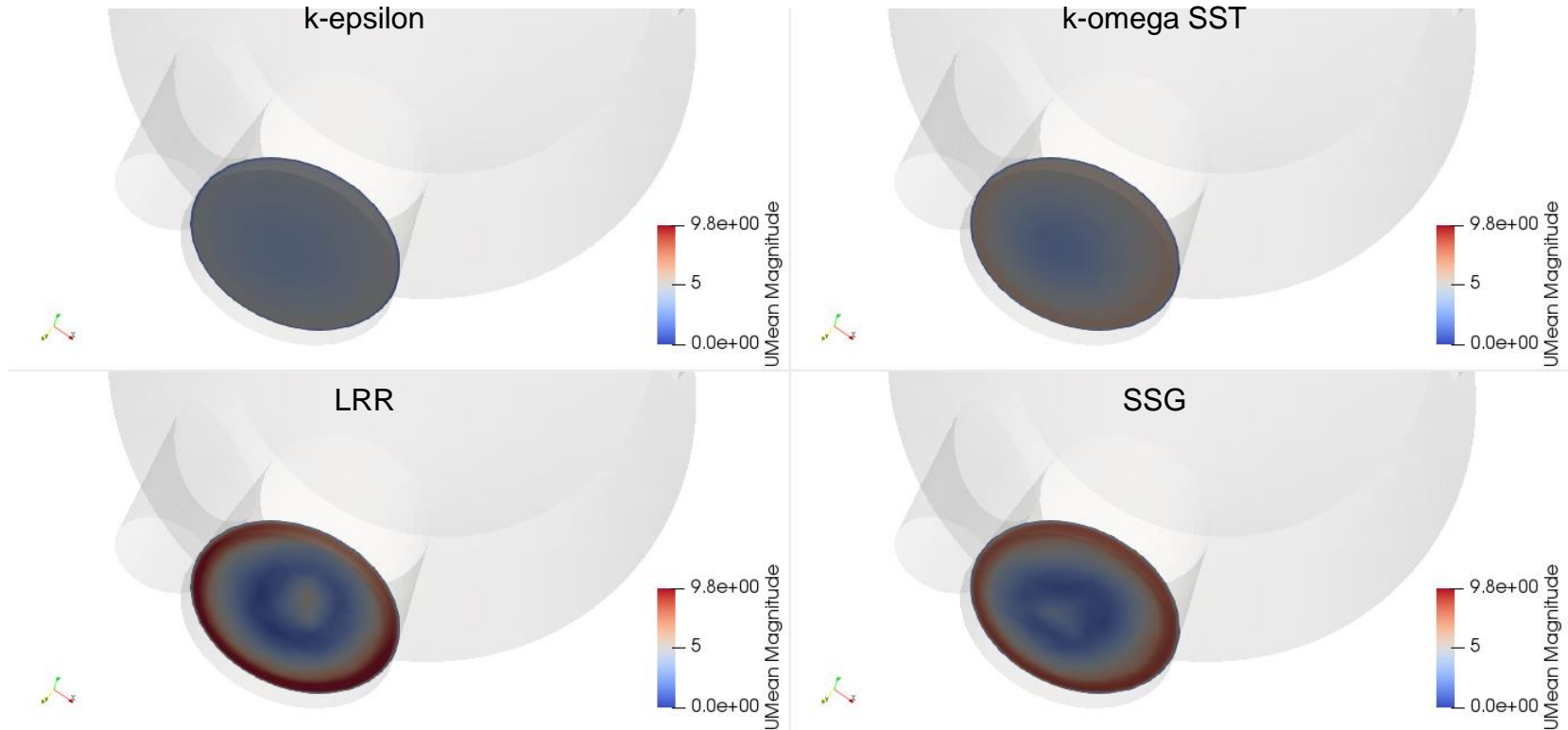
## Guided tutorial 6 – Static mixer – Reynolds stress model (RSM)



- In this guided tutorial we will simulate a turbulent flow using RSM.
- We will simulate the flow inside a static mixer.
- This problem is characterized with strong system rotation.
- With highly swirling flows and wall curvature, RANS models tend to overpredict turbulent viscosity and fail at resolving the velocity profiles.

# Guided tutorials

## Guided tutorial 6 – Static mixer – Reynolds stress model (RSM)



- In this guided tutorial we will simulate a turbulent flow using RSM.
- We will simulate the flow inside a static mixer.
- This problem is characterized with strong system rotation.
- With highly swirling flows and wall curvature, RANS models tend to overpredict turbulent viscosity and fail at resolving the velocity profiles.

# Guided tutorials

## Guided tutorial 6 – Static mixer – Reynolds stress model

- Probably, the RSM is the most physically sound RANS/URANS approach as it avoids the use of hypothesis/assumptions to model the Reynolds stress tensor.
- However, it is computationally expensive, and less robust than eddy viscosity models (EVM).
  - It can be unstable if proper boundary conditions and initial conditions are not used.
  - And it is heavily modeled.
- RSM models perform better in situations where the EVM models have poor performance,
  - Flows with strong curvature or swirl (cyclone separators and flows with concentrated vortices).
  - Flows in corners with secondary motions.
  - Very complex 3D interacting flows.
  - Highly anisotropic flows.
- The RSM closes the RANS equations by solving transport equations for the Reynolds stresses (six additional equations in 3D), together with an equation for the turbulent dissipation rate.
- Properly speaking, RSM can be considered scale-resolving simulations (SRS).
  - Proceed with care if you are doing steady simulations or 2D simulations.

# Guided tutorials

## Guided tutorial 6 – Static mixer – Reynolds stress model

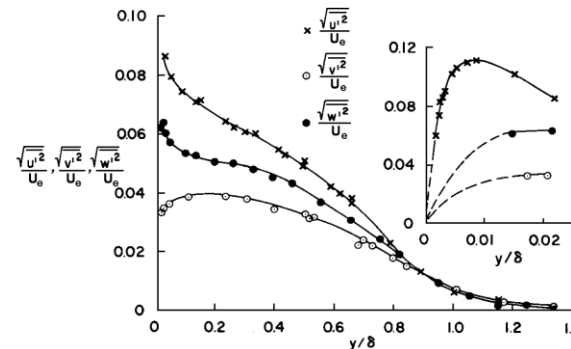
- The RSM model can be used with wall functions.
- If you are using a wall resolving approach, all Reynolds stresses must approach in an asymptotic way to zero at the wall.
- The freestream values can be computed as follows,

$$\overline{u_1'^2} = k$$

$$\overline{u_2'^2} = \overline{u_3'^2} = \frac{1}{2}k$$

$$\overline{u_i' u_j'} = 0 \quad (i \neq j)$$

$$\overline{u'^2} : \overline{v'^2} : \overline{w'^2} \approx 4 : 2 : 3$$



- The boundary condition for turbulent dissipation rate is determined in the same manner as for the two-equations turbulence models.

# Guided tutorials

## Guided tutorial 6 – Static mixer – Reynolds stress model

- In this case, we will first run a simulation using a two-equation model and then we will use this outcome as initial conditions for the RSM simulation.
  - RSM models are very sensitive to initial conditions.
  - It is highly recommended to have a well converged solution.
- In OpenFOAM, you will find the LRR [1] and the SSG [2] RSM models.
- In addition to the six equations of the Reynolds stress tensor, these models solve the turbulent dissipation rate transport equation.
- Therefore, you need to define the boundary conditions, initial conditions, and the proper numerics for the turbulent dissipation rate (epsilon).
- The turbulent kinetic energy is obtained by taking the trace of the Reynolds stress tensor.
- In our experience, the LRR model is more stable.
- If you have problems getting convergence, start the simulation using initial conditions from scratch, as described in the previous slide.
  - You just need to define the values of the normal Reynolds stresses.
  - The rest of the stresses are equal to zero.

[1] B. E. Launder, G. J. Reece, W. Rodi. Progress in the Development of a Reynolds-Stress Turbulence Closure. 1975.

[2] C. G. Speziale, S. Sarkar, T. B. Gatski. Modelling the Pressure-Strain Correlation of Turbulence: An Invariant Dynamical Systems Approach. 1991.

# Guided tutorials

## Guided tutorial 6 – Static mixer – Reynolds stress model

- At the walls, you can use the following boundary conditions (wall functions),

Field	Wall functions – High RE	Resolved BL – Low RE
nut	nutUSpaldingWallFunction nutUWallFunction nutkWallFunction	nutUSpaldingWallFunction nutUWallFunction nutkWallFunction
R	kqRWallFunction	kqRWallFunction
epsilon	epsilonWallFunction	epsilonWallFunction

# Guided tutorials

## Guided tutorial 6 – Static mixer – Reynolds stress model

### Additional notes on the *fvSchemes* dictionary

- Convective terms discretization of the Reynolds stress tensor can be set as follows:

```
divSchemes
{
    div(R)                Gauss linear;
    div(phi,R)            bounded Gauss limitedLinear 1;
    div((nu*dev2(T(grad(U)))) Gauss linear;
    div((nuEff*dev2(T(grad(U)))) Gauss linear;
}
```

Use the keyword `bounded` only in steady simulations

- The previous schemes are second order accurate.
- If you are having stability problems, try to use upwind for the term `div(phi,R)`.
- If you are conducting steady simulations, use under relaxation factors in the order of 0.7 (or even lower) for the Reynolds stresses (R).
- It is also recommended to use gradient limiters with the components of Reynolds stress tensor. The different components are defined as follows:
  - `grad(R.component(0))`, `grad(R.component(1))`, `grad(R.component(2))`, `grad(R.component(3))`, `grad(R.component(4))`, `grad(R.component(5))`

# Guided tutorials

- **Guided tutorial 7.** Vortex shedding past square cylinder – LES, DES, SRS.
- This case is ready to run.
- The case is located in the directory:

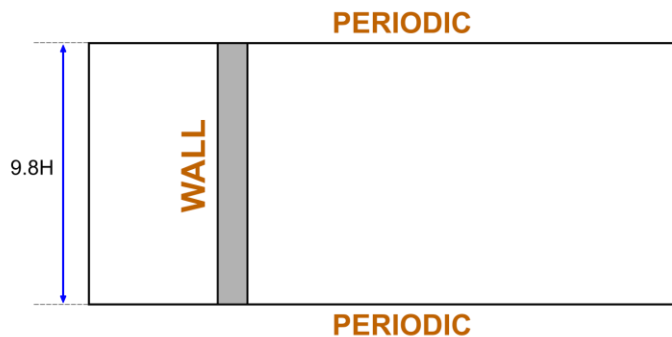
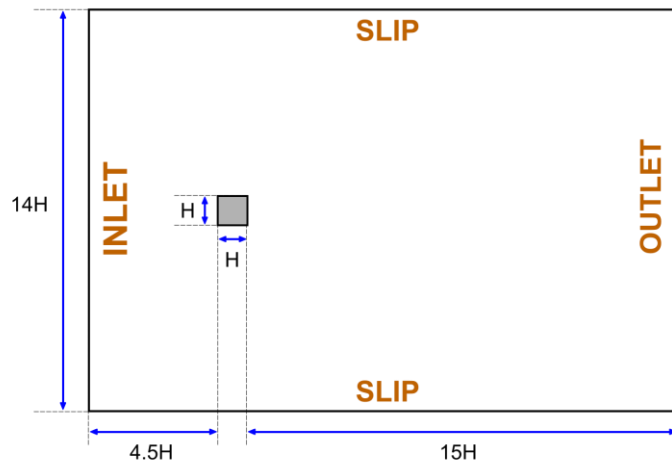
```
$TM/turbulence/GT7/
```

- In the case directory, you will find the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.
- You will also find a few additional files (or scripts) with the extension `.sh`, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.
- We highly recommend opening the `README.FIRST` file and typing 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, use the automatic scripts to run the cases.
- From this point on, please follow me. We are all going to work at the same pace.



# Guided tutorials

## Guided tutorial 7 - 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.

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

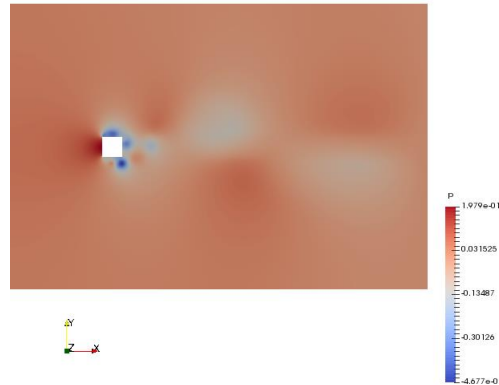
$$H = 0.04 \text{ m}$$

$$Re = 21400$$

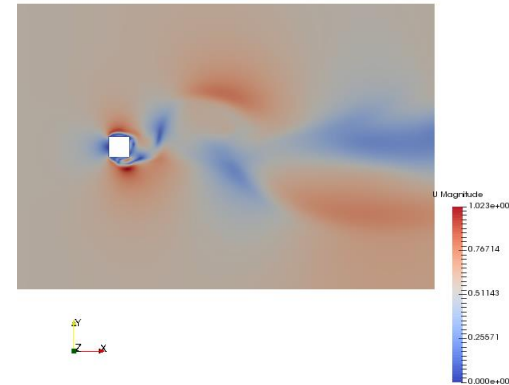
Working fluid: Water

# Guided tutorials

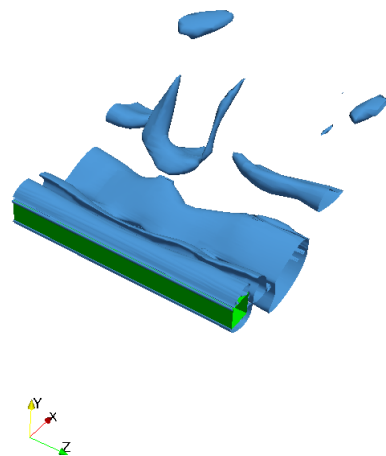
## Guided tutorial 7 - Vortex shedding past square cylinder



RANS (K-Omega SST) – Pressure field



RANS (K-Omega SST) – Velocity field

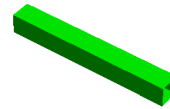


RANS (K-Omega SST) – Vortices visualized by Q-criterion



# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder



**URANS (K-Omega SST with no wall functions)**  
– Vortices visualized by Q-criterion

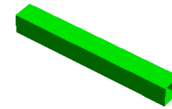
[www.wolfdynamics.com/wiki/squarecil/urans2.gif](http://www.wolfdynamics.com/wiki/squarecil/urans2.gif)

**LES (Smagorinsky) – Vortices visualized by Q-criterion**

[www.wolfdynamics.com/wiki/squarecil/les.gif](http://www.wolfdynamics.com/wiki/squarecil/les.gif)

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder



**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)

# Guided tutorials

## Guided tutorial 7 - 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
SAS	2.40	0.164	57690
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	-

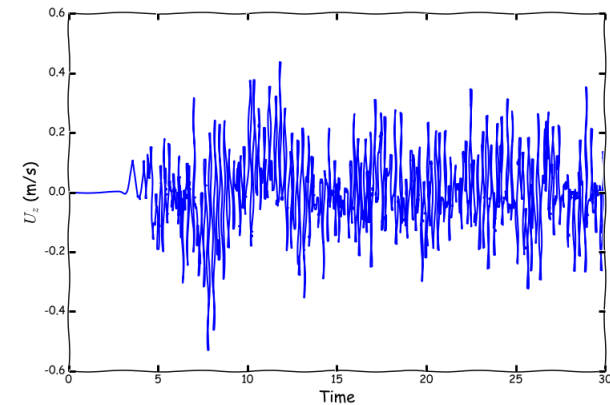
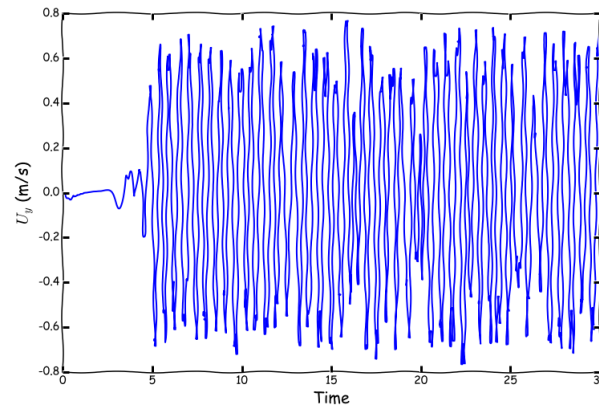
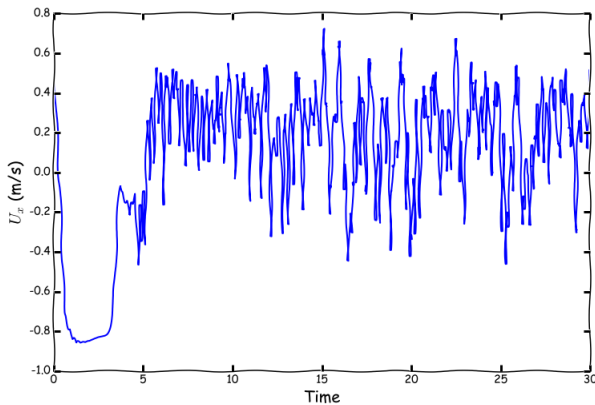
**Note:** all simulations were run using 4 cores.

### 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.*

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

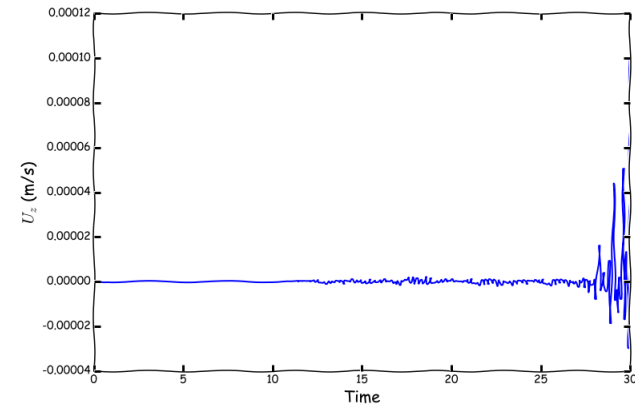
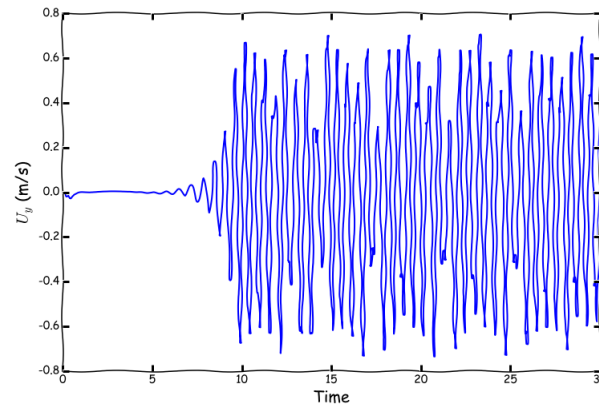
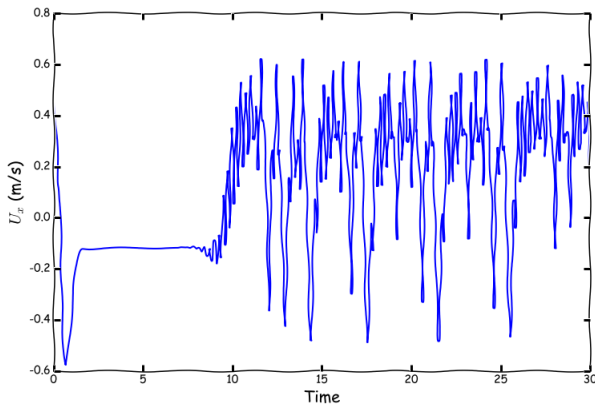


LES (Smagorinsky) –  $U_x$ ,  $U_y$ , and  $U_z$

- Velocity sample on the wake of the cylinder at seven different locations.
- This sampling corresponds to a probe located at 0.075 m from the center of the cylinder.

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

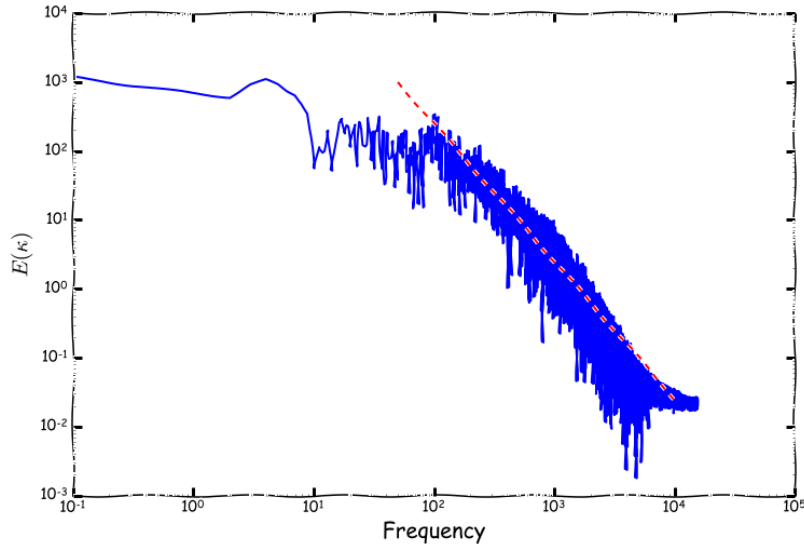


URANS (K-Omega SST with wall functions) –  $U_x$ ,  $U_y$ , and  $U_z$

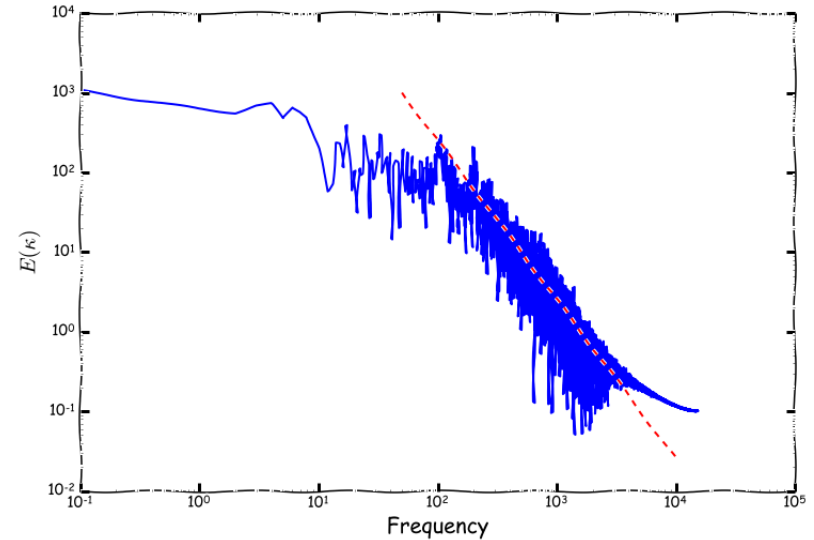
- Velocity sample on the wake of the cylinder at seven different locations.
- This sampling corresponds to a probe located at 0.075 m from the center of the cylinder.

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder



Laminar



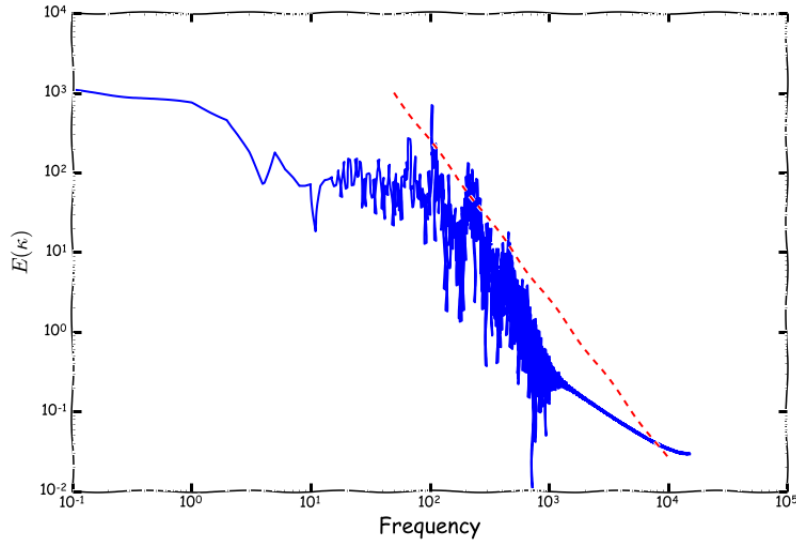
LES (Smagorinsky)

- Velocity sample on the wake of the cylinder at seven different locations.
- This sampling corresponds to a probe located at 0.075 m from the center of the cylinder.
- Notice that this kind of graph is local. It will be different for each and every point in the domain.

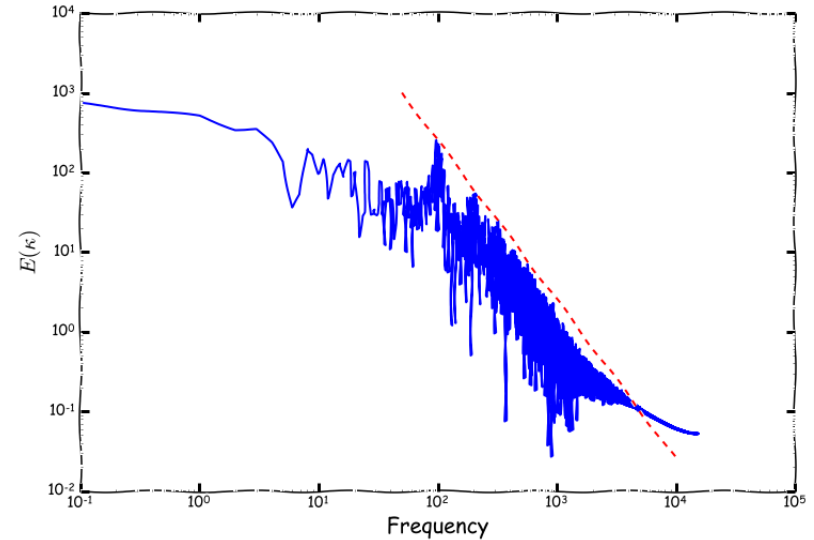


# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder



URANS (K-Omega SST with wall functions)

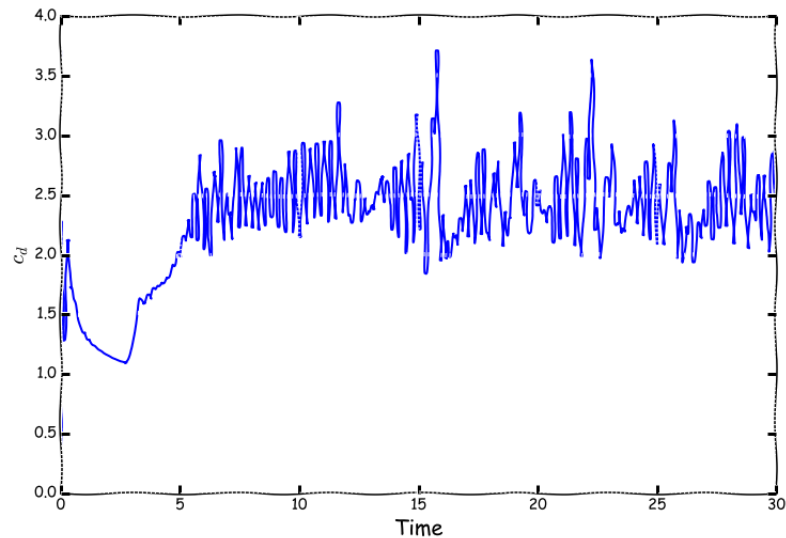


DES (SpalartAllmarasDDES)

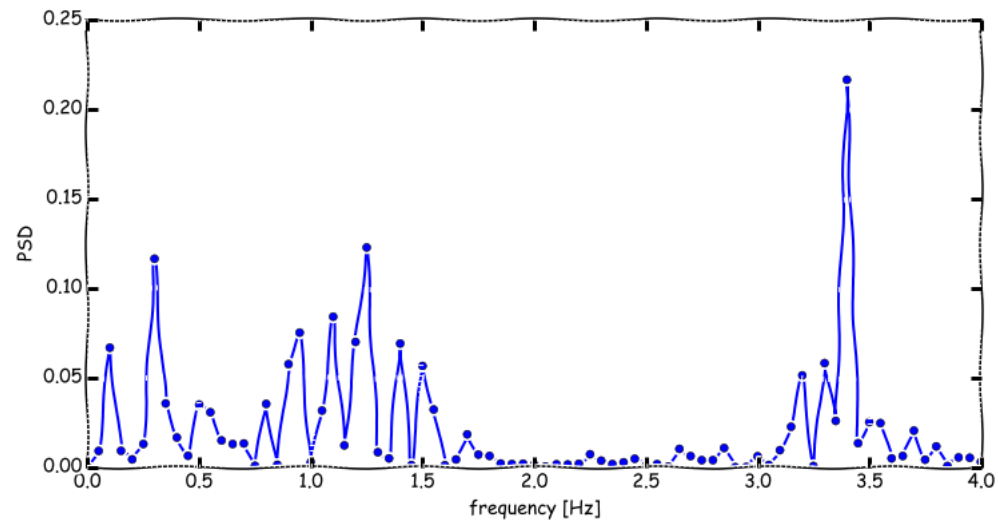
- Velocity sample on the wake of the cylinder at seven different locations.
- This sampling corresponds to a probe located at 0.075 m from the center of the cylinder.
- Notice that this kind of graph is local. It will be different for each and every point in the domain.

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder



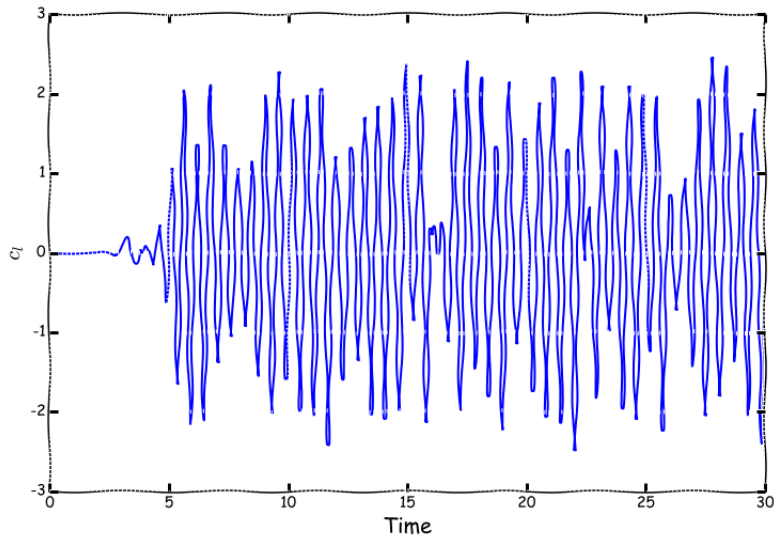
LES (Smagorinsky)  
Drag coefficient signal



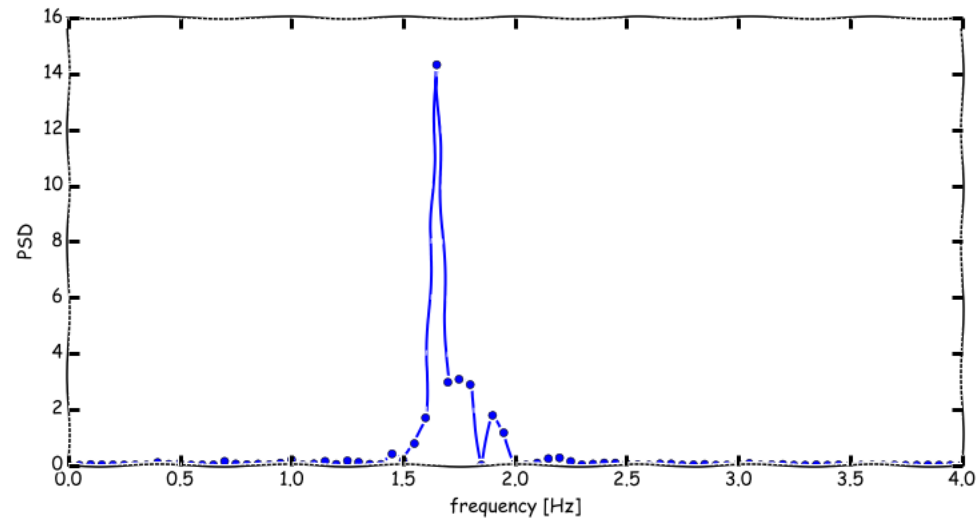
LES (Smagorinsky)  
Power spectrum

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder



LES (Smagorinsky)  
Lift coefficient signal



LES (Smagorinsky)  
Power spectrum

# Guided tutorials

## Guided tutorial 7 - 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 K-Omega SST turbulence model,

```
simulationType  RAS;           ← RANS type simulation
RAS
{
  RASModel      kOmegaSST;     ← RANS model to use
  turbulence     on;           ← Turn on/off turbulence. Runtime modifiable
  printCoeffs   on;           ← Print coefficients at the beginning
}
```

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

# Guided tutorials

## Guided tutorial 7 - 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 a LES turbulence model (Smagorinsky in this case),

```
simulationType  LES;  ← LES type simulation

LES  ← LES sub-dictionary
{
  LESModel      Smagorinsky;  ← LES model to use

  turbulence    on;  ← Turn on/off turbulence. Runtime modifiable

  printCoeffs   on;  ← Print coefficients at the beginning

  delta         cubeRootVol;  }
  cubeRootVolCoeffs
  {
    deltaCoeff   1;  }
}

← LES filter width – Implicit filtering
If you use a dynamic model you need to provide the test filter
```

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

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- When it comes to setting boundary conditions, and initial conditions, LES is much easier than RANS or RSM.
- Depending on the LES model, you only need to define the additional field variable  $\nu_{t}$  (sub-grid scale turbulent viscosity).
  - In this file, you only need to set the near the wall treatment, which for LES is **nutUSpaldingWallFunction** or **nutUWallFunction**.
  - You can use **nutkWallFunction** only if  $k$  is computed by the turbulence model.
- For example, the Smagorinsky and WALE LES models only require the additional turbulent variable  $\nu_{t}$  (sub-grid scale turbulent viscosity).
- The one equation LES models, besides the additional turbulent variable  $\nu_{t}$ , they require the definition of turbulent kinetic energy.
  - The treatment of the turbulent kinetic energy is exactly the same as for the RANS models.
- If you are using DES models, you will need to set the turbulent variables ( $k$ ,  $\omega$ ,  $\tilde{\nu}$ ,  $\nu_{t}$ ) in the same way as for RANS.
- Remember, when conducting LES simulations, it is very important to use good quality meshes, low dissipative and accurate discretization schemes, and keep the CFL number below 1.

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- To select the wall functions in LES and DES, follow these guidelines,

Field	Wall functions – High RE	Resolved BL – Low RE
nut	nutUSpaldingWallFunction nutUWallFunction nutkWallFunction*	nutUSpaldingWallFunction nutUWallFunction nutkWallFunction*
k	kqRWallFunction	kqRWallFunction kLowReWallFunction
omega	omegaWallFunction	omegaWallFunction

- In this tutorial,
  - Use high-Re and low-Re for RANS.
  - Use high-Re for URANS.
  - Use high-Re for LES (use **nutUSpaldingWallFunction**).
  - Use high-Re and low-Re for DES.

\* As this wall function is based on  $k$ , it can only be used if the turbulent model computes  $k$ .

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

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

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

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

$$\omega_{farfield} = \frac{\rho k}{\mu} \left( \frac{\mu_t}{\mu} \right)^{-1}$$

- Use the following initial estimates,

$$I = 5\% \quad \frac{\mu_t}{\mu} = 10$$



# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

### Additional notes on the *fvSolution* dictionary

- For the *fvSolution* dictionary:

momentumPredictor

yes;

Set to yes for high Reynolds flows, where convection dominates (default value is yes)

nOuterCorrectors

1;

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

nCorrector

3;

Recommended to use at least 3 correctors. It improves accuracy and stability. Use 4 or more for highly transient flows or strongly coupled problems..

nNonOrthogonalCorrectors

1;

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

turbOnFinalIterOnly

false;

Flag to indicate whether to solve the turbulence on the final PIMPLE iteration only. For SRS simulations the recommended value is false (the default value is true).

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

### Additional notes on the *fvSchemes* dictionary

- Remember, if you are conducting SRS simulations (LES/DES) it is extremely important to use low numerical dissipation methods.
- Gradient terms discretization can be set as follows:

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

```
gradSchemes
{
    default leastSquares;
}
```

- If you are getting unbounded quantities, you can add a slope limiter (not too aggressive), as follows,

```
gradSchemes
{
    default cellLimited Gauss linear 0.333;
}
```

```
gradSchemes
{
    default cellLimited leastSquares 0.333;
}
```

- Remember, the discretization can be set in a per-term basis.

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

### Additional notes on the *fvSchemes* dictionary

- Remember, if you are conducting SRS simulations (LES/DES) it is extremely important to use low numerical dissipation methods.
- Convective terms discretization can be set as follows:

```
divSchemes
{
    default                none;
    div(phi,U)             Gauss limitedLinearV 1;
    div(phi,k)             Gauss limitedLinear 1;
    div(phi,omega)        Gauss limitedLinear 1;
    div(phi,nuTilda)      Gauss limitedLinear 1;
    div((nuEff*dev2(T(grad(U)))) Gauss linear;
}
```

```
divSchemes
{
    default                none;
    div(phi,U)             Gauss linear;
    div(phi,k)             Gauss linear;
    div(phi,omega)        Gauss linear;
    div(phi,nuTilda)      Gauss linear;
    div((nuEff*dev2(T(grad(U)))) Gauss linear;
}
```

- Both of the previous schemes are second order accurate. However, the scheme limitedLinear is more stable.
- You can also use linearUpwind, however, it may be too diffusive for bad quality meshes.
- Remember, the discretization can be set in a per-term basis.

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

### Additional notes on the *fvSchemes* dictionary

- Remember, if you are conducting SRS simulations (LES/DES) it is extremely important to use low numerical dissipation methods.
- Time terms discretization can be set as follows:

```
ddtSchemes
{
    default backward;
}
```

```
ddtSchemes
{
    default CrankNicolson 0.7;
}
```

- Both of the previous schemes are second order accurate.
- You can also use the Euler method; however, it is first order accurate. For good accuracy you will need to use a low CFL number (in the order of 0.5).
- Disregarding of the method used, it is always recommended to use a CFL number of less than 1 (for low numerical diffusion).
- Remember, the discretization can be set in a per-term basis.

# Guided tutorials

## Guided tutorial 7 - 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.
- 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.
- We will use Python to compute the power spectrum of the signal, as well as the dominant frequency. We will compute basic statistics as well.
- Have in mind that the unsteady case will generate a lot of data.

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- Choose what case you want to run.
- You can choose between RANS, URANS, DES and LES.
- It is recommended to run first a RANS simulation and use the solution to initialize a LES simulation.
- You can also use the precursor RANS simulation to estimate the integral length scales and grid length scales.
- If you run a LES using the same mesh as in the precursor RANS simulation, just copy the solution from the latest time directory (or desired time directory) into the LES case (time directory 0).
- Remember, in the LES simulation you will need to create the dictionary file for *nut* (sub-grid scale turbulent viscosity). In this file, you only need to set the near the wall treatment.
- If you use different meshes between RANS and LES, you will need to interpolate the solution from the RANS simulation, as follows,
  - ```
$> mapFields ../source_directory -consistent -mapMethod  
cellPointInterpolate -sourceTime 'latestTime'
```

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- A few guidelines to consider when running LES simulations:
  - The Smagorinsky model is the cheapest one, but it does not perform well with wall bounded flows. The model coefficients/constants can depend on the flow conditions.
  - The WALE model is a little bit more expensive than the Smagorinsky model, but it works well with wall bounded flows and overcomes many of the limitations of the Smagorinsky model. The model coefficients/constants can depend on the flow conditions.
  - The dynamic models compute automatically the coefficients/constants, but this computation requires additional computational power. The model predicts accurately the wall behavior, transition, and allow energy backscatter. Do not use uniform initial conditions to run these models.
  - If you want to retain the history of the modeled kinetic energy, use one equation models.
  - Use wall function when running LES simulations. In OpenFOAM, you will find the Spalding's law which is essentially a fit of the laminar, buffer and logarithmic regions of the boundary layer (**nutUSpaldingWallFunction**).

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- At this point, we are ready to run.
- But before running, remember to setup the right numerics in the dictionary files *fvSolution* and *fvSchemes*.
- If you forget something, OpenFOAM will let you know.
- Also, for the LES simulation try to keep the CFL number below 0.9
- Finally, do not forget to setup the **functionObjects** to compute the forces, do the sampling, and compute  $y^+$  on-the-fly.



# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- To run the tutorial using `simpleFoam` and will wall refinement, type in the terminal:

```
1.  $> foamCleanTutorials
2.  $> blockMesh
3.  $> checkMesh
4.  $> refineWallLayer '(square)' 0.7 -overwrite
5.  $> refineWallLayer '(square)' 0.6 -overwrite
6.  $> refineWallLayer '(square)' 0.5 -overwrite
7.  $> checkMesh
8.  $> rm -r 0
9.  $> cp -r 0_org 0
```

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- To run the tutorial using `simpleFoam`, type in the terminal:

10. `$> decomposePar`

11. `$> mpirun -np 4 renumberMesh -overwrite -parallel`

12. `$> mpirun -np 4 simpleFoam -parallel > log | tail -f log`

13. `$> mpirun -np 4 postprocess -func Q -parallel`

14. `$> reconstructPar`

15. `$> paraFoam`

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- In steps 1, we clean the case directory.
- In steps 2 and 3 we generate the mesh and check its quality.
- In steps 4-6 we use the utility `refineWallLayer` to refine the mesh close to the walls (**square** in this case). This is helpful if you do not want to regenerate the mesh.
- In step 7 we check the mesh quality after refining the mesh close to the walls.
- In steps 8-9, we transfer the backup or original files to the directory 0. It is highly advisable to always keep backup files.
- In step 10 we decompose the mesh. At this point we are preparing to run in parallel.
- In step 11 we use `renumberMesh` to reduce the bandwidth of the coefficient matrices.
- In step 12 we run in parallel and save the standard output in the file log.
- In step 13 we compute the Q-criterion using the utility `Q`.
- In step 14 we reconstruct the solution.
- In step 15 we use `paraFoam` to visualize the solution.

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- To run the tutorial using `pimpleFoam`, type in the terminal:

1. `$> foamCleanTutorials`
2. `$> blockMesh`
3. `$> checkMesh`
4. `$> rm -r 0`
5. `$> cp -r 0_org 0`
6. `$> decomposePar`
7. `$> mpirun -np 4 renumberMesh -overwrite -parallel`
8. `$> mpirun -np 4 pimpleFoam -parallel > log | tail -f log`
9. `$> mpirun -np 4 Q -parallel`
10. `$> reconstructPar`
11. `$> paraFoam`

# Guided tutorials

## Guided tutorial 7 - Vortex shedding past square cylinder

- In steps 1, we clean the case directory.
- In steps 2 and 3 we generate the mesh and check its quality.
- In steps 4-5, we transfer the backup or original files to the directory 0. It is highly advisable to always keep backup files.
- In step 6 we decompose the mesh. At this point we are preparing to run in parallel.
- In step 7 we use `renumberMesh` to reduce the bandwidth of the coefficient matrices.
- In step 8 we run in parallel and save the standard output in the file `log`.
- In step 9 we compute the Q-criterion using the utility `Q`.
- In step 10 we reconstruct the solution.
- In step 11 we use `paraFoam` to visualize the solution.

# Guided tutorials

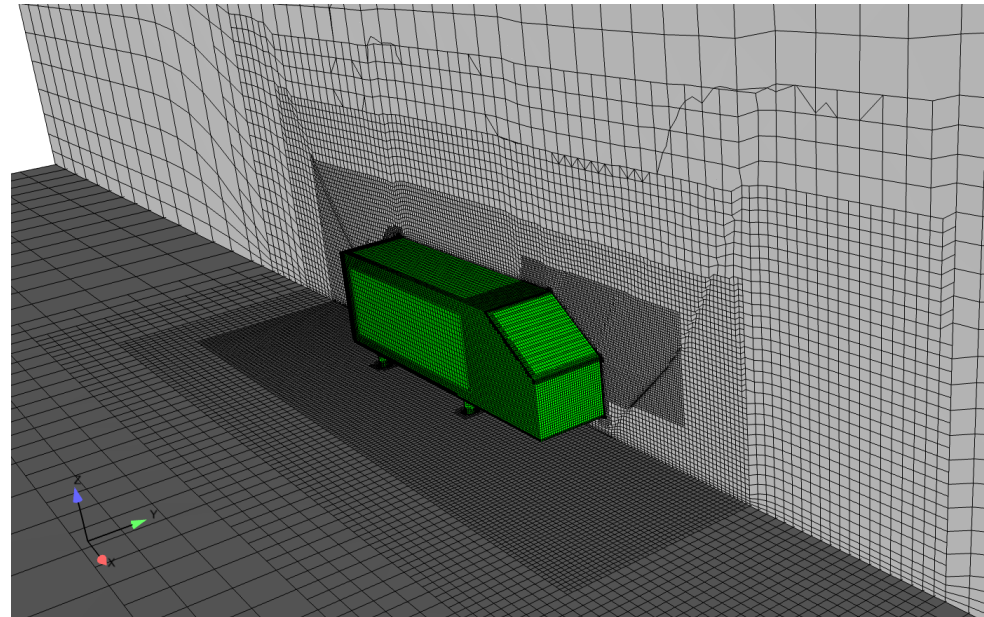
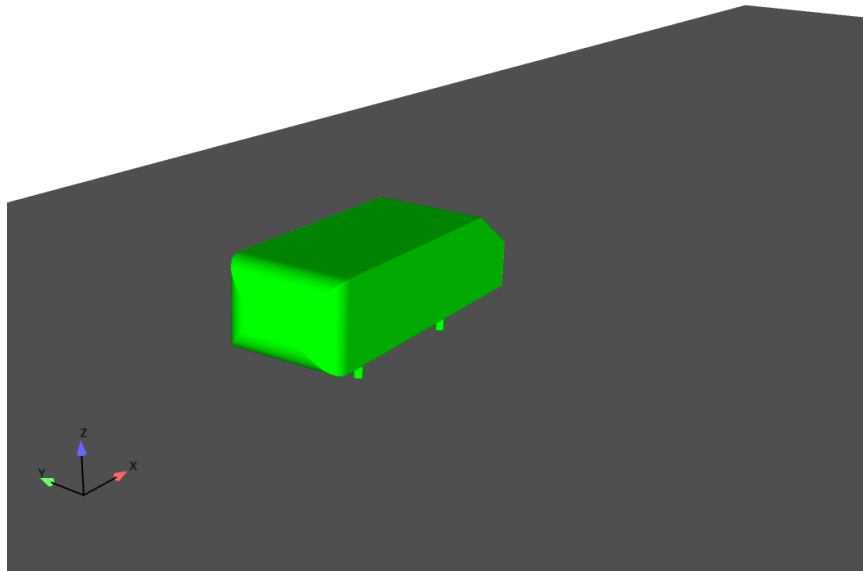
- **Guided tutorial 7.** Bonus case – Ahmed body.
- This case is ready to run.
- The case is located in the directory:

```
$TM/turbulence/GT8/
```

- In the case directory, you will find the `README.FIRST` file. In this file, you will find the general instructions of how to run the case. In this file, you might also find some additional comments.
- You will also find a few additional files (or scripts) with the extension `.sh`, namely, `run_all.sh`, `run_mesh.sh`, `run_sampling.sh`, `run_solver.sh`, and so on. These files can be used to run the case automatically by typing in the terminal, for example, `sh run_solver`.
- We highly recommend opening the `README.FIRST`, file and typing 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, use the automatic scripts to run the cases.
- From this point on, please follow me. We are all going to work at the same pace.

# Guided tutorials

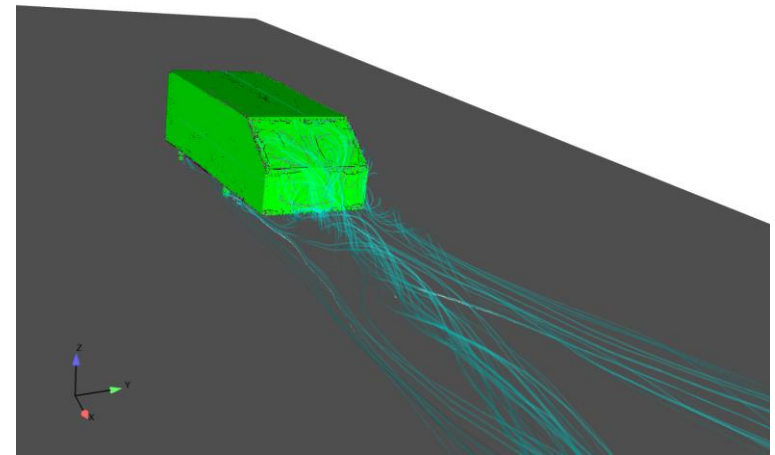
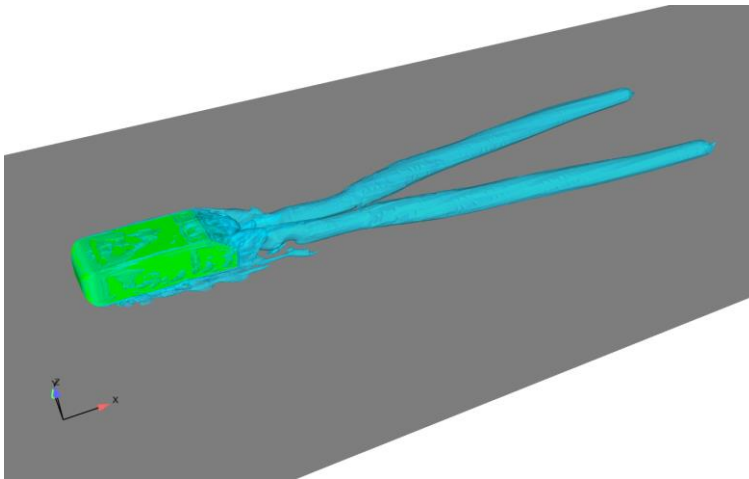
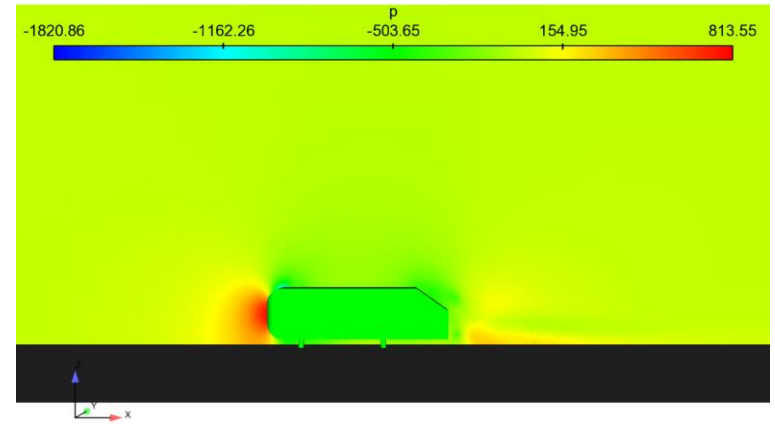
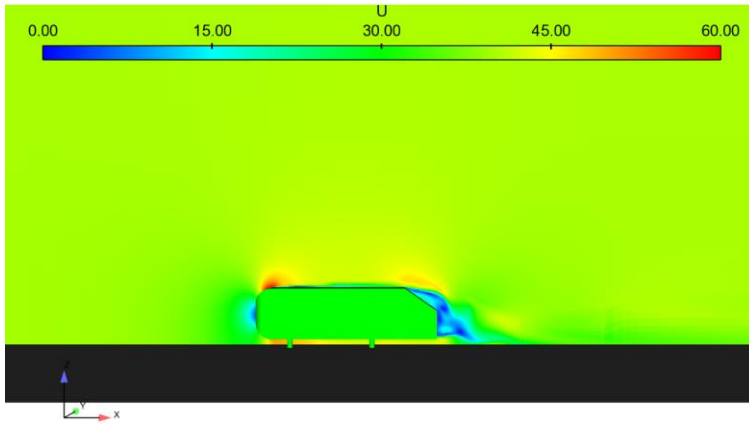
## Guided tutorial 8 – Ahmed body



- This is it; this is our last tutorial.
- We hope that at this point you have a better idea on turbulence modeling.
- So, let us go free styling and let us play around with this case.
- Feel free to use any turbulence model and play around with the mesh.
- Try to use a velocity around 30-40 m/s.
- If you have any question, we are here to help you.

# Guided tutorials

## Guided tutorial 8 – Ahmed body

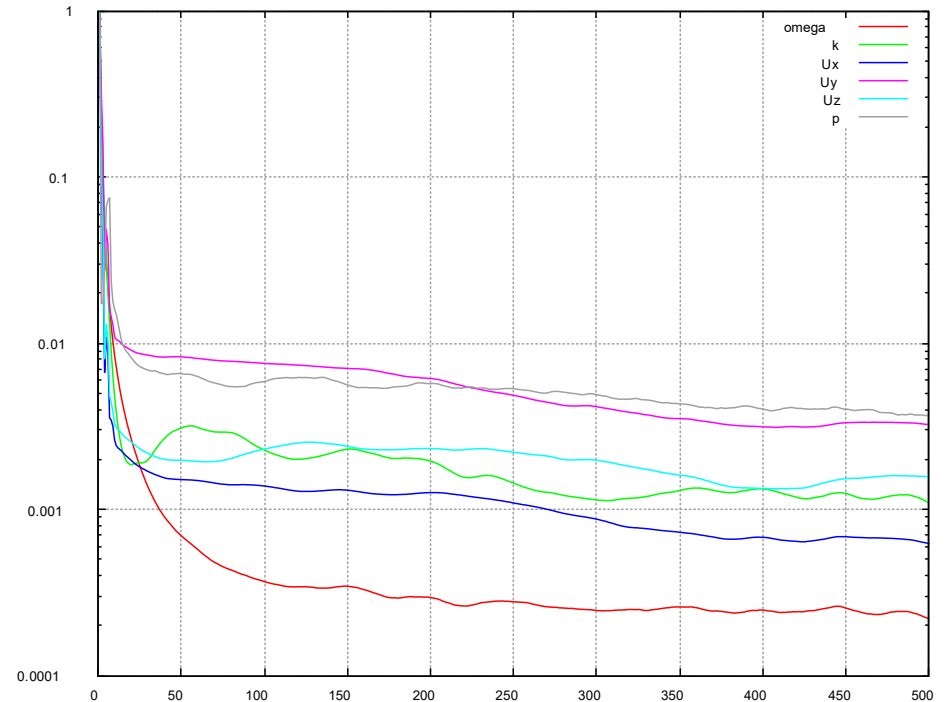
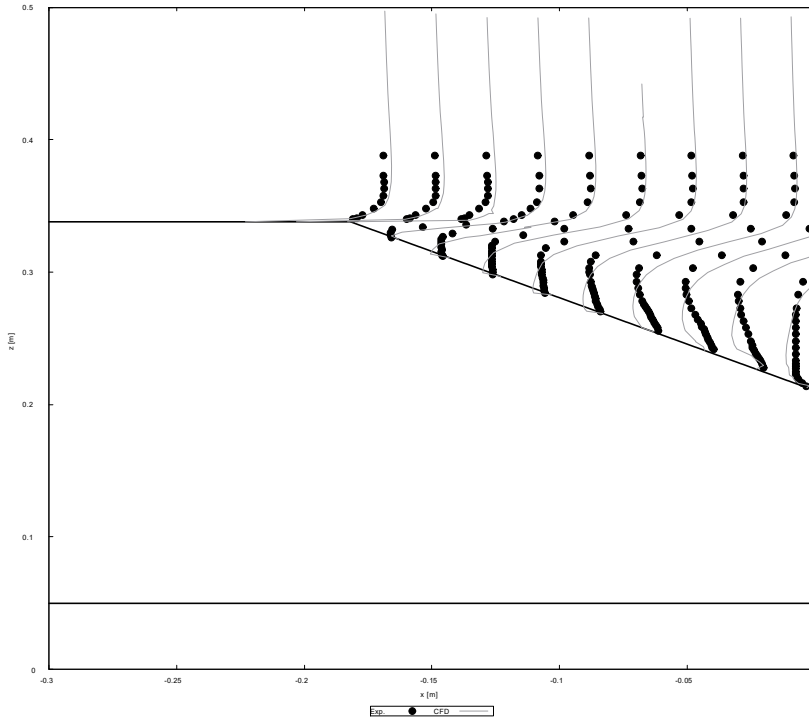


- Some colorful fluid dynamics.
- As an exercise, try to capture the vortices on the rear part of the body.



# Guided tutorials

## Guided tutorial 8 – Ahmed body



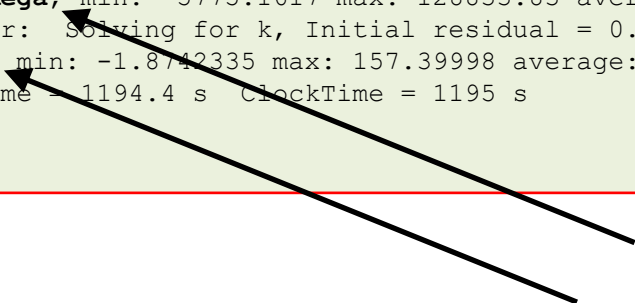
- If you want to go for the extra mile, try to validate your solution (left figure).
- Remember to monitor your solution (right figure), because you may get unbounded quantities that can make the solver go bananas.

# Guided tutorials

## Guided tutorial 8 – Ahmed body

- One final advise, from time to time you may get unbounded quantities.
- Be careful with this, because one single unbounded turbulent quantity can give you very different results or can cause divergence.
- If you get a message similar to the one below, adjust your numerical scheme in order to remove the oscillations.
- That means, use an aggressive gradient limiter, upwind for the convective terms, and decrease the CFL number.

```
smoothSolver: Solving for Ux, Initial residual = 0.00075011, Final residual = 6.488869e-05, No Iterations 4
smoothSolver: Solving for Uy, Initial residual = 0.0030806253, Final residual = 0.00026669783, No Iterations 4
smoothSolver: Solving for Uz, Initial residual = 0.0025587795, Final residual = 0.0002536501, No Iterations 4
GAMG: Solving for p, Initial residual = 0.0048591327, Final residual = 4.2489925e-05, No Iterations 3
time step continuity errors : sum local = 0.00039023267, global = 2.6967204e-05, cumulative = -0.004857943
smoothSolver: Solving for omega, Initial residual = 0.00010231634, Final residual = 4.291796e-06, No Iterations 3
bounding omega, min: -5775.1617 max: 128833.83 average: 2127.9019
smoothSolver: Solving for k, Initial residual = 0.00079768137, Final residual = 3.2814585e-05, No Iterations 3
bounding k, min: -1.8742335 max: 157.39998 average: 6.024669
ExecutionTime = 1194.4 s  ClockTime = 1195 s
```



# Guided tutorials

## Guided tutorial 8 – Ahmed body

- The case is located in the directory:  
`$TM/turbulence/GT8/`
- In this directory you will find the basic setup but try to do something different and play around with the concepts we just studied.
- In the file *README.FIRST* you will find some basic instructions on how to run this case.
- At this point, you are on your own. So please do not break the solver.