Drivaer validation case

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

```
$PTOFC/drivaer
```

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

Drivaer validation case

Drivaer model – External aerodynamics

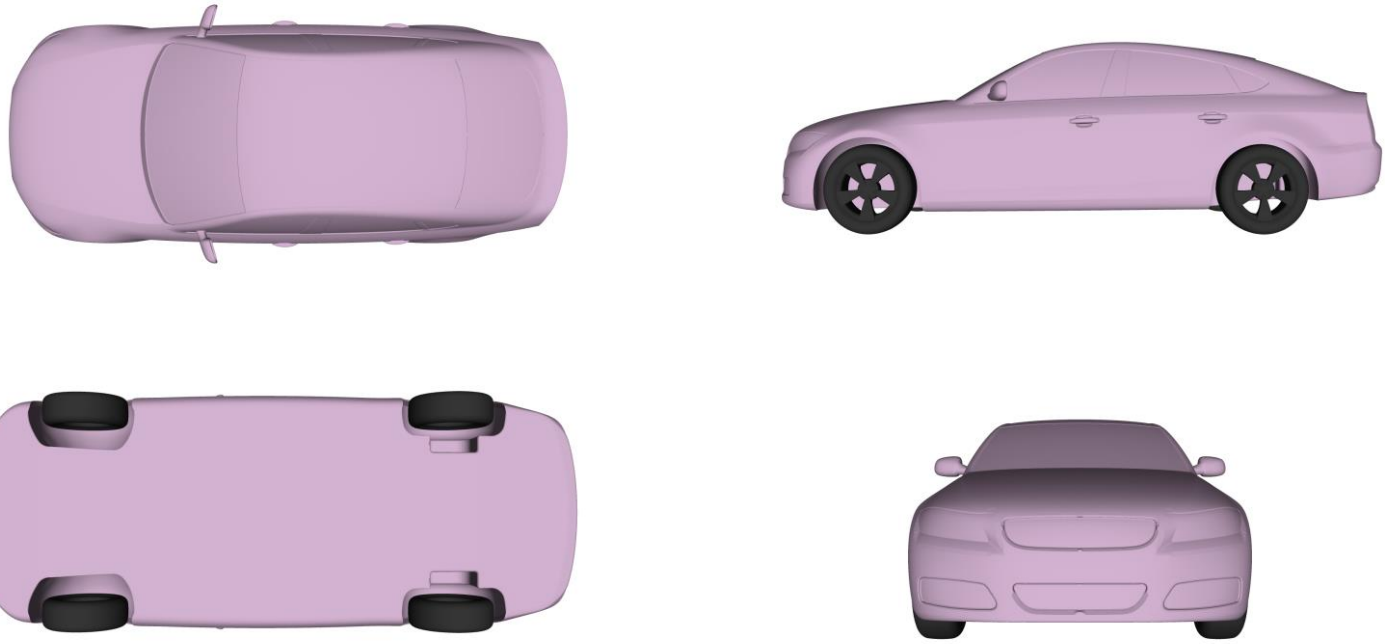
Physical and numerical side of the problem:

- In this case, we are going to solve the flow past the Drivaer model [1].
- This is a turbulent 3D case.
- We will use a RANS turbulence model, and to simplify the setup we will model only half of the model.
- The free-stream velocity is 30 m/s.
- The ground and wheels are moving with a translational and rotational velocity, respectively.
- This problem has a lot of experimental data (with a lot variance in the values reported).

[1] https://www.epc.ed.tum.de/en/aer/research-groups/automotive/drivaer/

Drivaer validation case

Drivaer model – External aerodynamics



Note:
The geometry corresponds to the fastback and smooth underbody model

Physical and numerical side of the problem:

- In this case, we are going to solve the flow past the Drivaer model [1].
- This is a turbulent 3D case.
- We will use a RANS turbulence model, and to simplify the setup we will model only half of the model.
- The free-stream velocity is 30 m/s.
- The ground and wheels are moving with a translational and rotational velocity, respectively.
- This problem has a lot of experimental data (with a lot variance in the values reported).

[1] https://www.epc.ed.tum.de/en/aer/research-groups/automotive/drivaer/

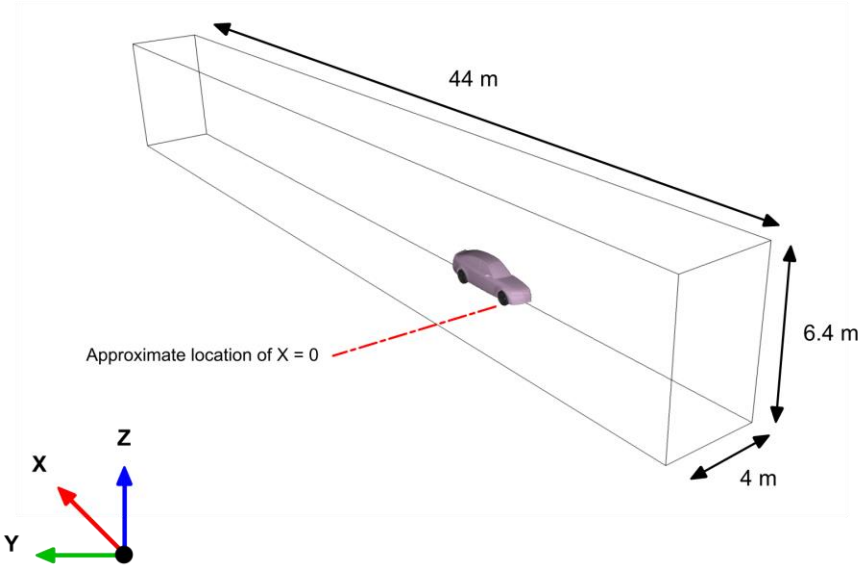
Drivaer validation case

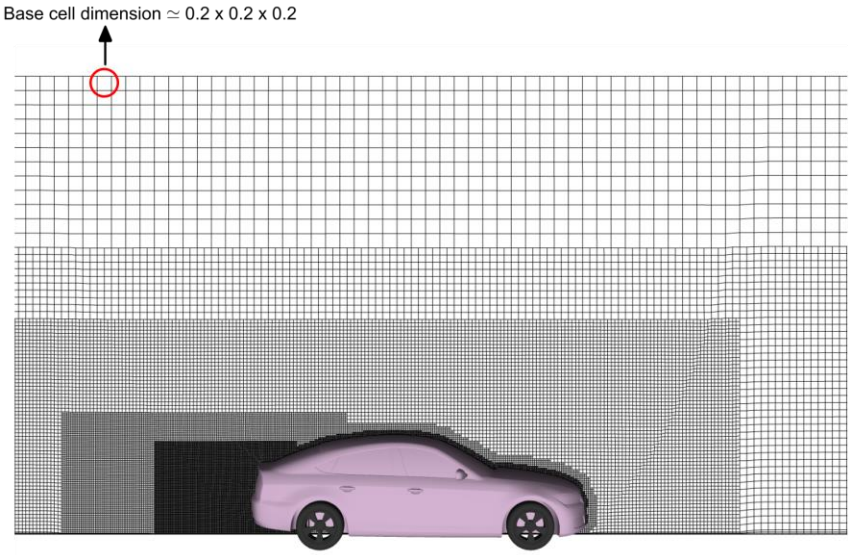
What are we going to do?

- We will use this case to learn how to setup an automotive simulation, including setting the moving ground and rotating wheels.
- We will compare the numerical solution with the experimental values.
- We will run the case with a robust numerics, but you are invited to try different setups and compare the different outcomes.
- You are also invited to try different setups, that is, rotating wheels against fixed wheels, moving ground against fixed ground, and so on.
- To find the numerical solution we will use the solver `simpleFoam`.
- `simpleFoam` is a steady-state solver for incompressible, turbulent flow, using the **SIMPLE** algorithm.
- To speed up the convergence rate, we will use `potentialFoam` to initialize the flow (pressure and velocity fields).
- To avoid velocity spikes during the beginning of the simulation, we will add a source term to limit the maximum velocity.
- This source term will be enabled during the first 200 iterations.

Drivaer validation case

Domain dimensions and meshing parameters – Fine mesh



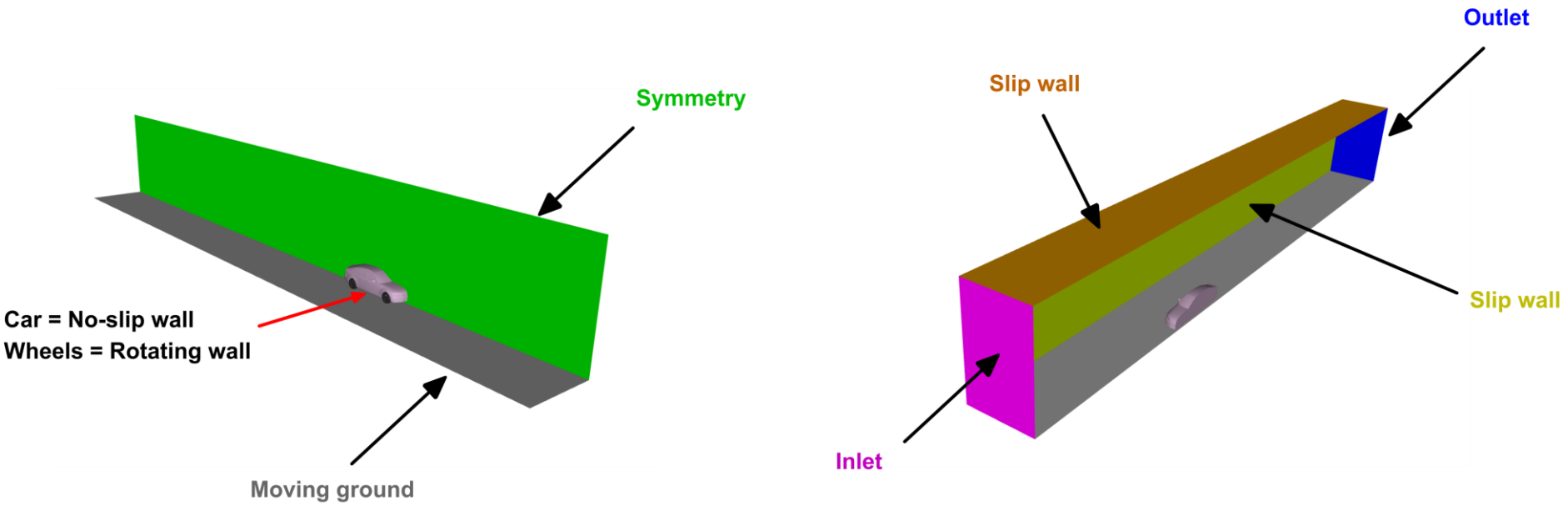


Surface refinement (car and wheels) = 4
Features refinement (only on wheels) = 5
Inflation layers (car, wheels, and gorund) = 3

- The domain dimensions, reference cell dimension, and surface refinement parameters are illustrated in the figure above.
- To get a better mesh, just increase the surface refinement and add more inflation layers.

Drivaer validation case

Boundary conditions

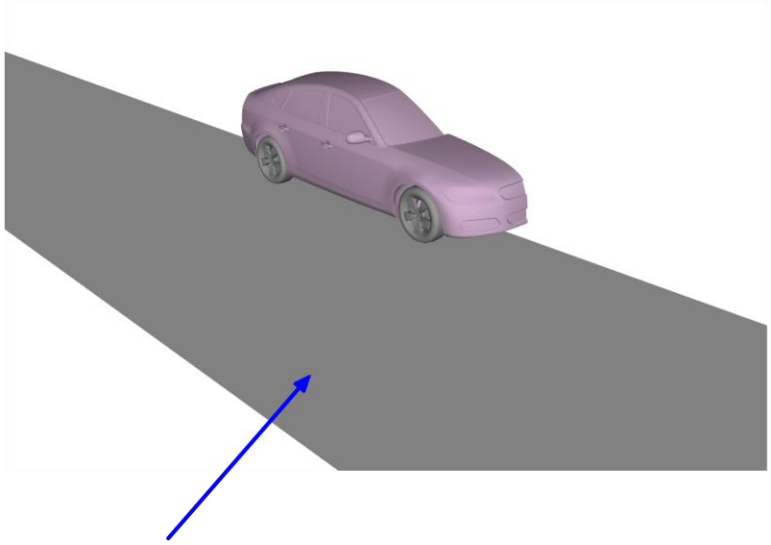


- The boundary conditions are illustrated in the figure above.
- The inlet velocity is equal to 30 m/s.
- Backflow conditions are used at the outlet.
- The turbulence quantities (free-stream and wall values) were computed using the recommendations found in references [1, 2].

[1] https://turbmodels.larc.nasa.gov/
[2] http://www.wolfdynamics.com/tools.html?id=110

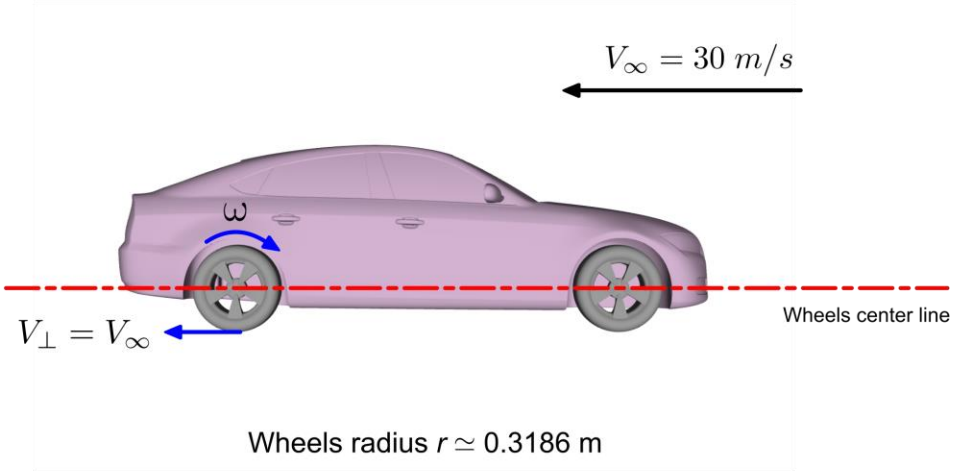
Drivaer validation case

Moving wall boundary conditions



Ground is equivalent to a moving wall with translational velocity

$$V_{ground} = V_\infty = 30\ m/s$$



$$\omega = \frac{V_\perp}{r} \approx 94\ rad/s$$

- The moving wall boundary conditions are illustrated in the figure above.
- A moving wall translational boundary condition is imposed at the ground.
- At the wheels, a moving wall rotational boundary condition is used.

Drivaer validation case

Selecting turbulence model

- In the dictionary *turbulenceProperties* we select the turbulence model. This dictionary is located in the directory **constant**.
- In this case we are using the **kOmegaSST** turbulence model with wall functions.
- The mesh is generated for a target y^+ value between $30 < y^+ < 300$.

```
simulationType RAS;
RAS
{
    RASModel kOmegaSST;
    turbulence on;
    printCoeffs on;
}
```

- We are using the default model coefficients.

Drivaer validation case

Selecting the working fluid physical properties

- The physical properties are set in the dictionary *transportProperties*.
- This dictionary is located in the directory **constant**.
- The working fluid is air at sea level and 25 degrees Celsius.

```
transportModel Newtonian;

nu [0 2 -1 0 0 0 0 ] 1.58E-5;
```

- The transport model is the standard Newtonian model.
- The reference density 1.205, this value is important for normalizing the force coefficients.

Drivaer validation case

Selecting the discretization schemes

```
ddtSchemes
{
    default steadyState;
}


gradSchemes
{
    default cellLimited Gauss linear 1;
    grad(U) cellLimited Gauss linear 1;
    grad(k) cellLimited Gauss linear 1;
    grad(omega) cellLimited Gauss linear 1;
    grad(p) cellLimited Gauss linear 0.333;
}


divSchemes
{
    default none;

    div(phi,U)     bounded Gauss linearUpwind grad(U);

    div(phi,k)             bounded Gauss upwind;
    div(phi,omega)         bounded Gauss upwind;



    div((nuEff*dev(grad(U).T()))) Gauss linear;
}



interpolationSchemes
{
    default linear;
}
```

- **Steady state discretization**

- **Discretization of the gradient terms**
- **Notice that aggressive limiters are only used for the velocity and turbulent quantities.**

- **Discretization of the velocity convective term.**
- **For good accuracy, it is important to use a method at least second order accurate.**

- **Discretization of the turbulence quantities convective terms.**
- **First order discretization of these terms is acceptable.**

- **Discretization of the Reynolds stresses**

- **Interpolation scheme.**
- **Most of the times (if not all the times) linear interpolation is fine.**

Drivaer validation case

Selecting the discretization schemes

```
laplacianSchemes
{
    default Gauss linear limited 0.5;
}


snGradSchemes
{
    default limited 0.5;
}


wallDist
{
    method meshWave;
}
```

- Discretization of the diffusive terms.
- The choice of the method and blending factor is related to the mesh quality.
- For industrial meshes or meshes with non-orthogonality more than 70, using limited 0.5 is recommended

- Method to compute the distance-to-wall.
- This is only used in turbulence models that requires this information (e.g., k-Omega family of turbulence models)
- For meshes with good quality at the walls, the meshWave method is the recommended one.
- If the quality at the walls is low, is better to use the Poisson method.

Drivaer validation case

Selecting the solution method and linear solvers

```
solvers
{
    p
    {
        solver           GAMG;
        tolerance        1e-6;
        relTol           0.01;
        smoother         GaussSeidel;
        nPreSweeps       0;
        nPostSweeps      2;
        cacheAgglomeration on;
        agglomerator     faceAreaPair;
        nCellsInCoarsestLevel 100;
        mergeLevels      1;
       minIter             3;
    }

    U
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0.001;
         minIter 3;
    }

    "(k|omega)"
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0.001;
         minIter 3;
    }
    ...
    ...
    ...
}
```

- Pressure (p) linear solver.
- Most of times GAMG method is fine for pressure.
- If at any point you see this method taking more that 100 iterations to convergence, switch to a Newton-Krylov method (e.g., PCG).
- It is also recommended to set the minimum number of iterations to 3.
- The tolerances used in this case are recommended for most of the cases.

- Velocity (U) and turbulence quantities (k and omega) linear solvers.
- The PBiCGStab is recommended for most of the applications.
- It is also recommended to set the minimum number of iterations to 2-3.
- The tolerances used in this case are recommended for most of the cases.

Drivaer validation case

Selecting the solution method and linear solvers

```
    solvers
    {

        ...
        ...
        ...

        Phi
        {
            solver GAMG;
            agglomerator faceAreaPair;
            mergeLevels 1;
            cacheAgglomeration true;
            nCellsInCoarsestLevel 200;
            tolerance 1e-8;
            relTol 0.001;
            smoother GaussSeidel;
            nPreSweeps 0;
            nPostSweeps 2;
            nFinestSweeps 2;
            minIter 3;
        }

    }


potentialFlow
{
    nNonOrthogonalCorrectors 10;
}
```

- This is the linear solver used with the potential solver (potentialFoam).
- potentialFoam solves for the velocity potential to provide velocity and incompressible flux fields, typically used to initialize viscous calculations.
- Using the method GAMG is recommended.
- It is also recommended to set the minimum number of iterations to 3.
- The tolerances used in this case are recommended for most of the cases.

- Number of non-orthogonal corrections to perform when using potentialFoam.
- It is recommended to perform at least 10 iterations.
- If at any point you see that the residuals get stalled, it is an indication that you have mesh quality problems, or the mesh is very sensitive to gradient computations.

Drivaer validation case

Selecting the solution method and linear solvers

```
SIMPLE
{
    nNonOrthogonalCorrectors 2;

    pRefCell        0;
    pRefValue       0;



    consistent yes;

    residualControl
    {
        p               1e-4;
        U               1e-4;
        nuTilda         1e-4;
        k               1e-4;
        omega           1e-4;
    }
}

relaxationFactors
{
    fields
    {
        p          0.9;
    }

    equations
    {
        p          0.9;
        U          0.7;
        k          0.7;
        omega      0.7;
    }

}
```

SIMPLE method options

Non-orthogonal corrections.
It is recommended to o at least one correction (even in good quality meshes).

Enable consistent formulation of the SIMPLE method

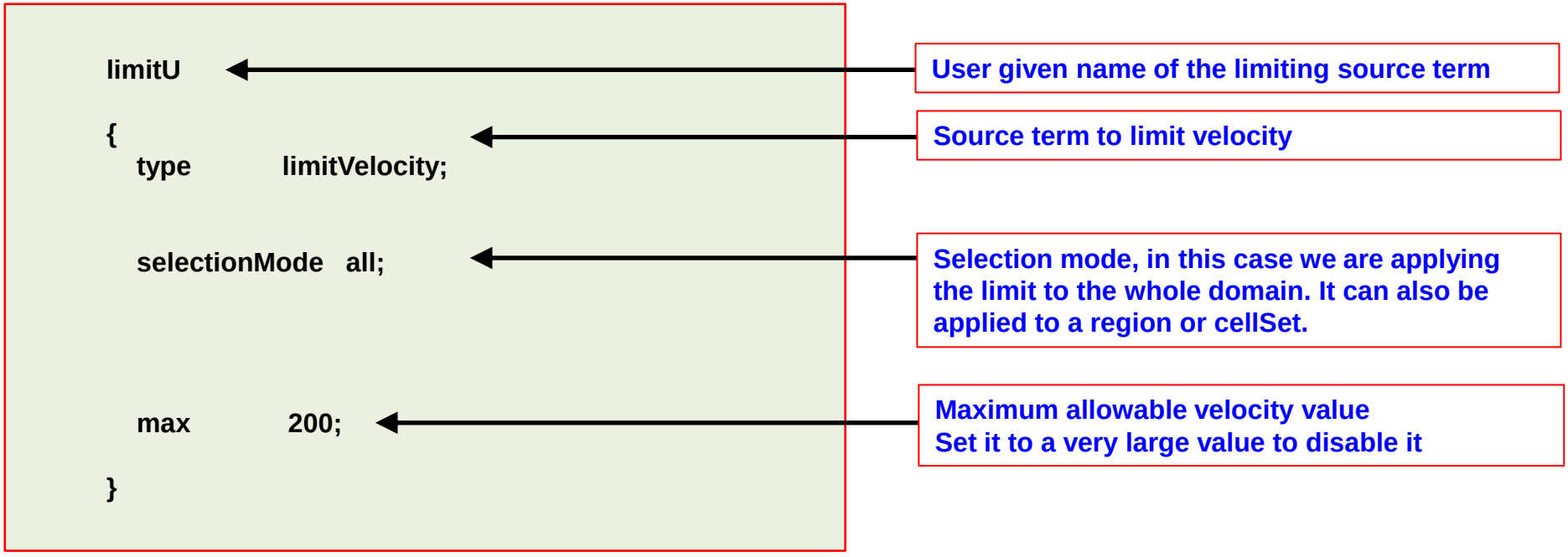
Field quantities residuals.
If the residuals are not reached, the solver will iterative until the maximum number of iterations.

Under-relaxation factors (URF).
Remember, URF are problem dependent.
Most of the times, the values used in this case are the recommended ones with the SIMPLE consistent method.

Drivaer validation case

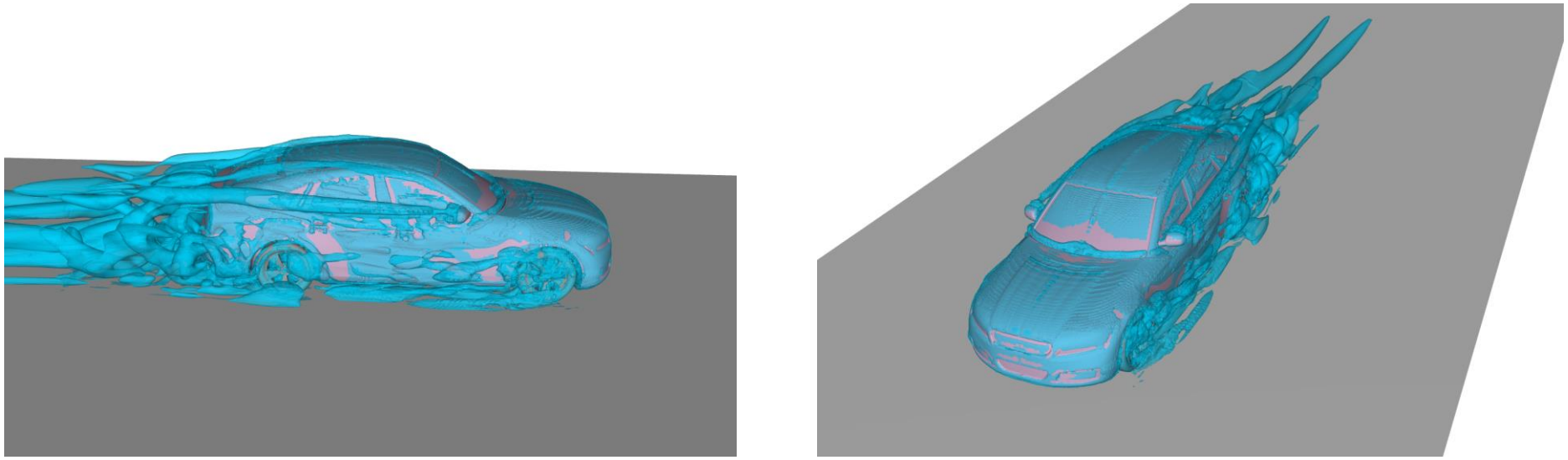
fvConstraints – Limiting the maximum velocity magnitude value

- To avoid velocity spikes during the beginning of the simulation, we can add a source term to limit the maximum velocity magnitude value.
- This limiting of the maximum value is not compulsory. However, it is recommended to follow this practice to avoid reaching unrealistic values and eventually divergence.
- This can be done by adding a limit in the form of a source term in the `fvConstraints` dictionary (located in the directory **`system`**).
- Also, it is recommended to use this form of limiter during the initial iterations and until the solution has stabilized.



Drivaer validation case

Quantitative and qualitative post-processing



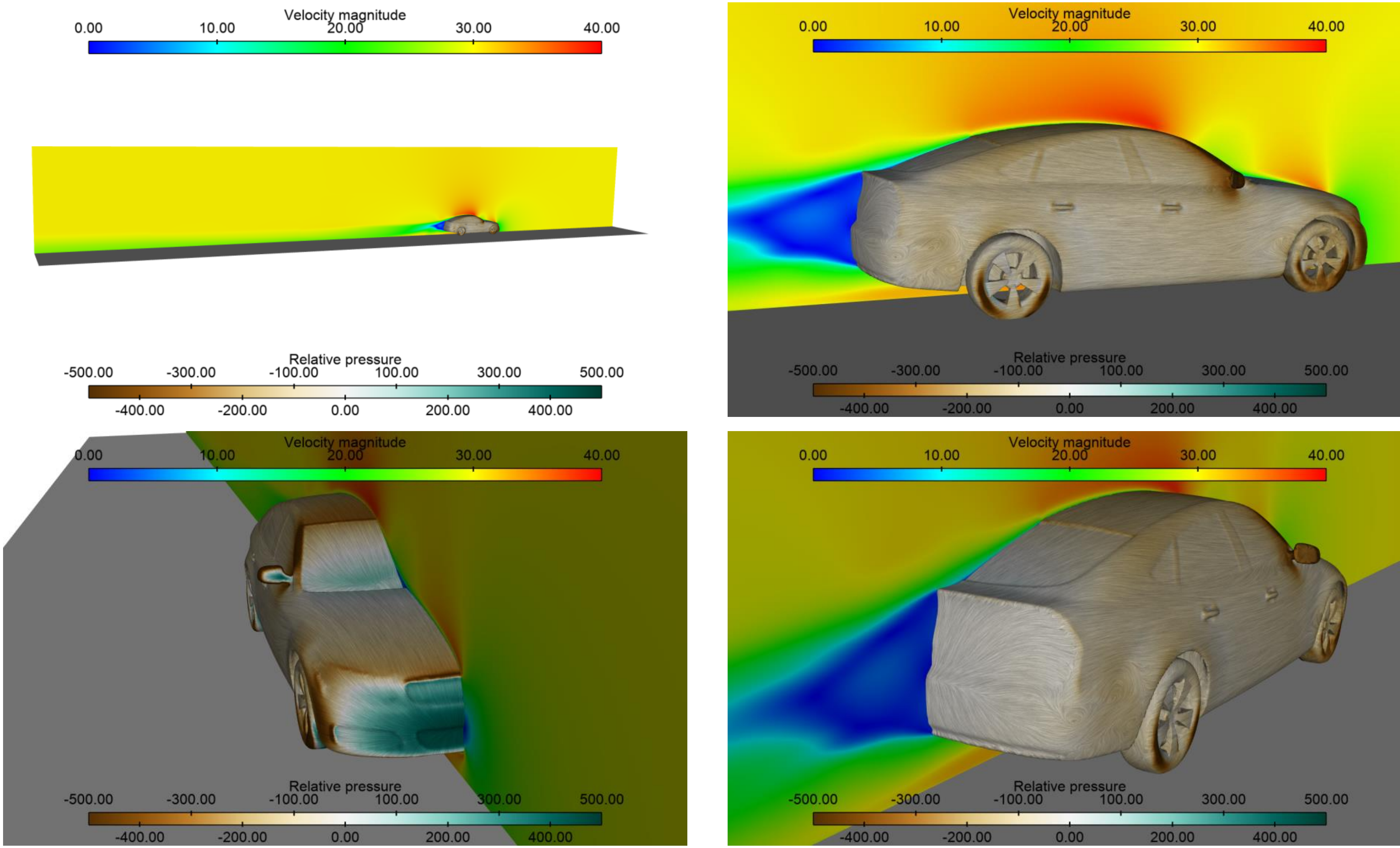
Visualization of vortical structures using Q-criterion

Note:

All the results reported correspond to the fastback and smooth underbody model

Drivaer validation case

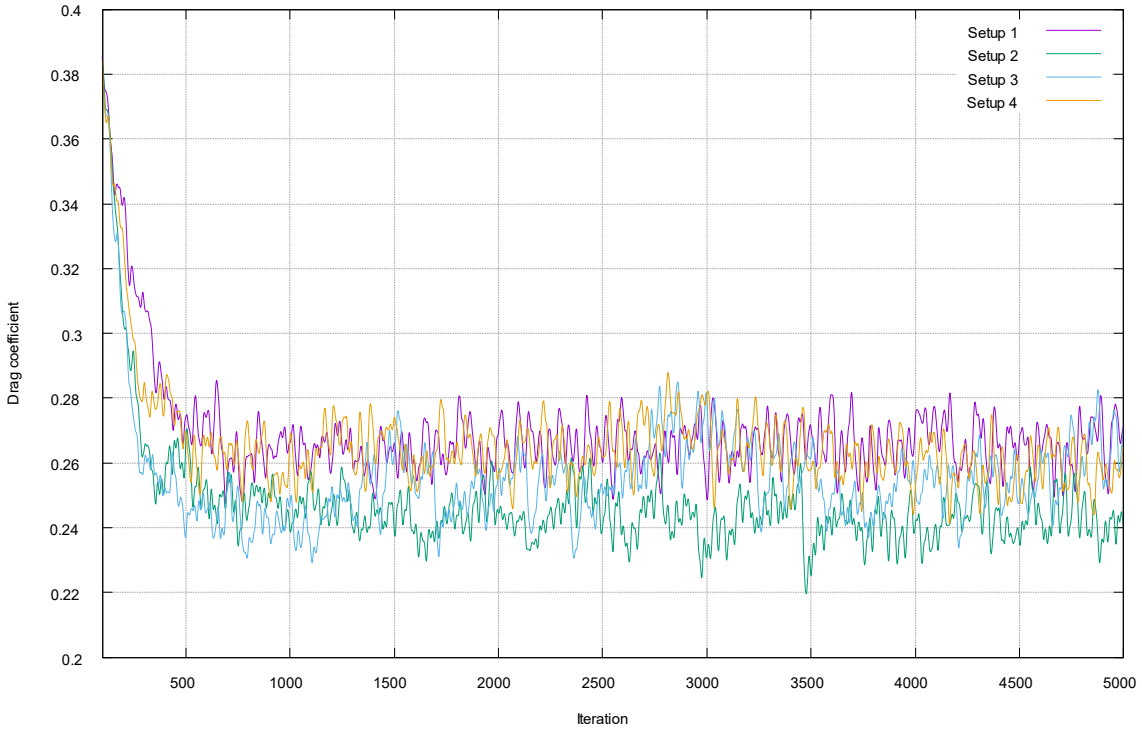
Quantitative and qualitative post-processing



Note: All the results reported correspond to the fastback and smooth underbody model

Drivaer validation case

Quantitative and qualitative post-processing



Case number	Description	Mean drag coefficient (OpenFOAM)
Setup 1	No rotating wheels – Moving ground	0.2660
Setup 2	Rotating wheels – Moving ground	0.2426
Setup 3	Rotating wheels – No moving ground	0.2569
Setup 4	No rotating wheels – No moving ground	0.2630

Note:

All the results reported correspond to the fastback and smooth underbody model

Drivaer validation case

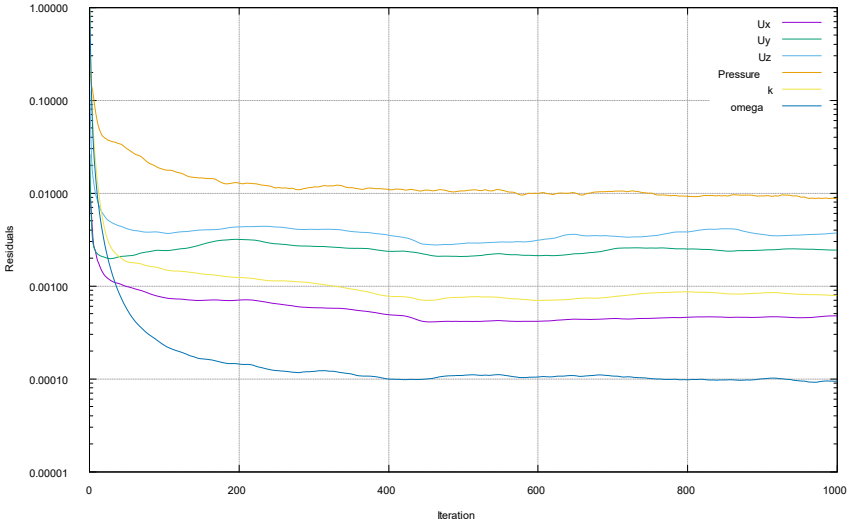
Quantitative and qualitative post-processing

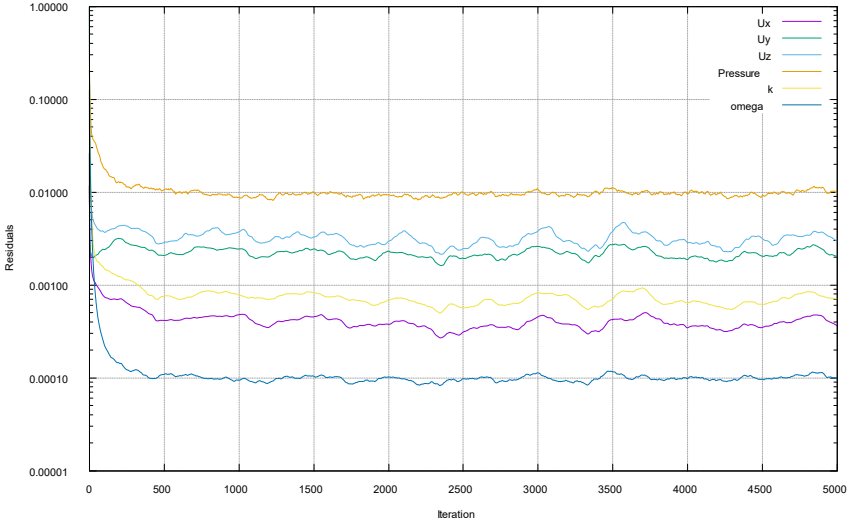
Case number	Mean drag coefficient (OpenFOAM)
Setup 1	0.2660
Setup 2	0.2426
Setup 3	0.2569

Reference	Reported mean drag coefficient
Ref. [1] – EXP PVT UC	0.243
Ref. [1] – EXP PVT	0.228
Ref. [1] – EXP TUM ASME	0.247
Ref. [1] – EXP TUM SA	0.243
Ref. [1] – NUM US:200M	0.235
Ref. [1] – NUM SS−EWT:200M	0.214
Ref. [2]	0.258

Note:

All the results reported correspond to the fastback and smooth underbody model





[1] R. Yazdani. Steady and Unsteady Numerical Analysis of the DrivAer Model. Chalmers University of Technology, Master Thesis, 2015.

[2] Experimental Comparison of the Aerodynamic Behavior of Fastback and Notchback DrivAer Models. SAE 2014-01-0613.

Drivaer validation case

Running the case

- This case is distributed with two meshes,
    - A coarse mesh with approximately 660000 cells
    - A fine mesh with approximately 4048000 cells.
- If you do not want to generate the mesh from scratch, you can use the pre-generated meshes.
- In the terminal type

1. ```
   $> sh run_mesh_fluent.sh
   ```

- Use the coarse mesh to obtain fast outcomes.
- While the fine mesh is not optimal, it is good enough to obtain an accurate and stable solution.
   ```

Drivaer validation case

Running the case

- To automatically generate the mesh using `snappyHexMesh`, type in the terminal,

```
$> sh run_mesh_shm.sh
```

- This script will automatically generate the mesh using the predefined parameters.
- Use the coarse mesh to obtain fast outcomes.
- While the fine mesh is not optimal, it is good enough to obtain an accurate and stable solution.
- For better accuracy, the user is invited to try with a finer mesh, for this, you will need to change surface refinement and inflation layers the parameters in the dictionary *`snappyHexMeshDict`*.

Drivaer validation case

Running the case

- The `run_mesh_shm.sh` script, will execute the following steps (not all of them are listed):

1. `$> foamCleanTutorials`
2. `$> rm -rf 0 > /dev/null 2>&1`
3. `$> surfaceFeatures`
4. `$> blockMesh`
5. `$> decomposePar`
6. `$> mpirun -np 4 snappyHexMesh -parallel -overwrite`
7. `$> mpirun -np 4 checkMesh -parallel`
8. `$> reconstructParMesh -constant`

- These are the basic steps to generate the mesh.
- Notice that the meshing is done in parallel with 4 processors.

Drivaer validation case

Running the case

- It is extremely recommended to use the pre-generated mesh as the mesh generation can be time consuming.
- To use the pre-generated mesh (coarse of fine), type in the terminal,

```
$> sh run_mesh_fluent.sh
```

- To run the case, using the pre-generated mesh, type in the terminal,

```
$> sh run_solver_fluent.sh
```

- The scripts are setup to run in parallel with 4 processors.

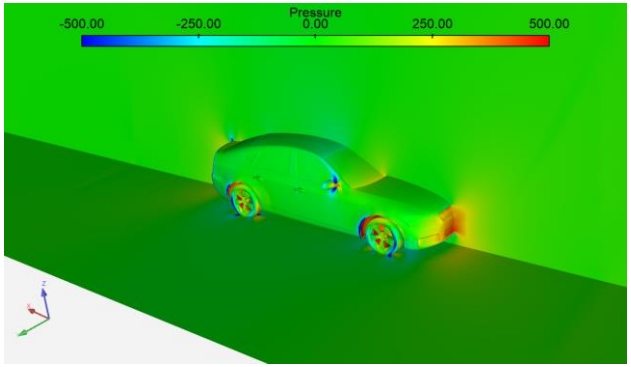
Drivaer validation case

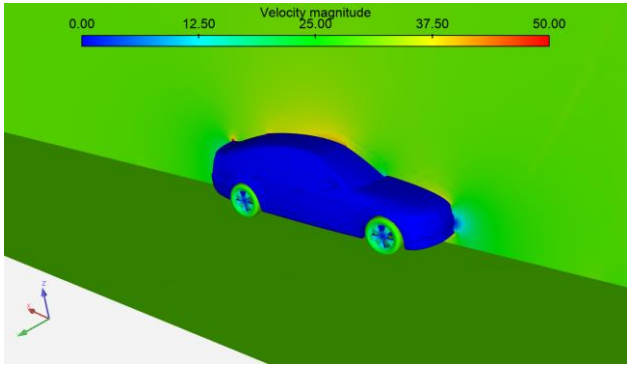
Running the case – Non-uniform initialization

- The non-uniform initialization is done by typing the following command in the terminal,

```
1. $> mpirun -np 4 potentialFoam -parallel -writep
```

- This will initialize the velocity and pressure fields using the inlet velocity.
- This initialization is done in the script `run_solver_fluent.sh`, before running the simulation using the solver `simpleFoam`.
- If you are working with external aerodynamics, it is strongly recommended to use this kind of initialization as it speeds the computations and gives more stability (specially during the initial iterations).
- This kind of initialization does not work in closed domains.





Drivaer validation case

Acknowledgements

- We would like to thanks Marco Giachi for the enlightening discussions, valuable suggestions, and sharing his setup as a starting point for developing this validation case.

- If you want to know more about Marco’s work, look for his book in Amazon:
  - https://www.amazon.com/Capire-Formula-segreti-della-evoluzione/dp/8833241068/ref=sr_1_1?keywords=marco+giachi&qid=1567298134&s=gateway&sr=8-1
  - https://www.amazon.it/Capire-Formula-segreti-evoluzione-ampliata/dp/8833241068/ref=sr_1_1?__mk_it_IT=%C3%85M%C3%85%C5%BD%C3%95%C3%91&keywords=marco+giachi&qid=1567298167&s=gateway&sr=8-1