Tutorial 10: Tesla Car

Learning Outcomes:

Basic setup for Tesla Car simulation.

X.1 Introduction

Automotive aerodynamics is a significant area of interest for CFD and one which has driven much research in areas such as turbulence modelling. In this tutorial we will examine a simulation of a simplified Tesla racing car. This is provided with an `Allrun` file, which is an OpenFOAM script file (extensively used in the release tutorials) which automates the process of running the case : type `./Allrun` in the case directory and it will automatically execute all the different OpenFOAM applications to complete the simulation. However it is important to understand the different commands and what they do. This tutorial serves as the starting point for your CFD miniproject; however just submitting the results of running `Allrun` will not gain many marks; you are expected to modify the case and use it to undertake a thorough investigation of both the CFD and the results – some suggestions for directions to take with this are provided at the end.

X.2 The Problem

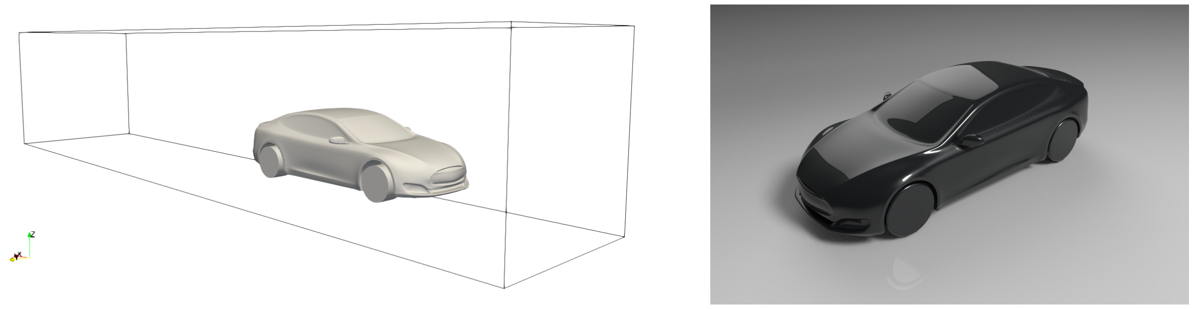

Figure 1: Geometry of Tesla racing car. Note that the geometry is symmetric so only 1/2 the vehicle is actually simulated

The case files provide a basic setup for the Tesla car shown in figure 1. Given the symmetry of the case, in fact only half the vehicle is being simulated, with a symmetry plane down the centre of the car. The car is moving at a speed of 20 m/s, but the simulation is performed in the frame of reference of the car (which is therefore stationary whilst the inlet and ground are moving backwards at this speed). The wheels are at $X = 3.9m$ (back wheel) and $X = 0.93m$ (front wheel) and are rotating at an appropriate angular velocity not to slip on the road (this is defined in the `U` variable file). The side and top are also defined as symmetry planes as this implements a zero gradient condition at these places, which is a good way to specify an "open" domain for an outdoor vehicle.

Car		Wheels	
Length (X)	4.97 m	Diameter	0.64 m
Height (Z)	1.28 m	Front Wheel	X = 0.93 m
Width (Y)	2.17 m	Back Wheel	X = 3.9 m
Wheelbase	2.97 m		

X.3 Allrun

In this case the commands are

```
#!/bin/sh
cd $\{0%/*\} || exit 1    # run from this directory

# Source tutorial run functions
. $WM_PROJECT_DIR/bin/tools/RunFunctions

runApplication surfaceFeatureExtract

runApplication blockMesh

[ ! -d 0 ] && cp -r 0.orig 0

runApplication snappyHexMesh -overwrite

runApplication decomposePar

runParallel potentialFoam

runParallel $(getApplication)

runApplication reconstructParMesh -constant

runApplication reconstructPar -latestTime
```

The main commands can be run separately (probably worth doing this at least once to understand how this works) – `runApplication` and `runParallel` are OpenFOAM scripts which run the commands singly or in parallel, respectively. The actual commands are as follows :

```
surfaceFeatureExtract
```

Extracts the edges of the object from the .stl file – this can be used in `snappyHexMesh` to improve the accuracy of the meshing around the edges.

```
blockMesh
```

Uses `blockMesh` to define a base rectangular block of cells which will be truncated by `snappyHexMesh` to create the mesh.

```
[ !  -d 0 ] && cp -r 0.orig 0
```

Copies the zero timestep directory into place (some Linux wizardry here).

```
snappyHexMesh -overwrite
```

Runs `snappyHexMesh` (based on the preprepared `system/snappyHexMeshDict` dictionary) to do the meshing. The `-overwrite` flag indicates that all necessary information is written back to the `0` timestep directory rather than creating separate directories for each step of the process – so the files can be used instantly for the next stage.

```
decomposePar
```

The aim is to run the simulation in parallel on 4 processors – this command divides the mesh and data files into 4 separate chunks (domain decomposition) as defined in `system/decomposeParDict`, using a simple method called "Scotch".

```
mpirun -np 4 potentialFoam -parallel

mpirun -np 4 simpleFoam -parallel
```

These commands run first `potentialFoam` (solves the velocity potential equation, which is quite a good way of initialising an external aerodynamics solution) then `simpleFoam`, both in parallel on 4 processors (possibly 4 cores, depending on your architecture). The `mpirun` command is being dealt with in the `Allrun` file through the `runParallel` script, but you need it here if you want to run the command on its own.

```
reconstructParMesh -constant

reconstructPar -latestTime
```

Finally, to postprocess the entire domain, we need to pull the individual mesh sections back together – these commands reconstruct the mesh as a single entity and reassemble all the datafields for the latest time step in the simulation (1000, here)

Running the `Allrun` script you will see that the output from each separate OpenFOAM app gets placed in a log file `log.XXX`, where `XXX` is the name of the app. The log file for `simpleFoam` can be analysed using the command

```
foamLog log.simpleFoam
```

which generates a directory `logs` containing files for all the separate residuals, which can then be plotted.

As indicated, this is intended as a starting point for your own investigation of the CFD and outputs for this case. Some suggestions for further investigation :

- What is the effect of changing (refining?) the mesh, on the solution? Ideally you should look for mesh independence, where the solution does not change with finer meshes – this may not be possible for this case, but you should experiment with different meshing strategies; maybe even use `Pointwise` or another meshing package.
- If you have a large mesh, you might like to try a parallelisation study – how does the calculation time change when run on different numbers of processors?
- Has the calculation converged? Is it actually steady or would a transient solution (`pisoFoam`) be better?
- Is the domain large enough/too large? How would you change it to be representative of a wind tunnel (say?)
- How can you validate the solution? What typical drag coefficient would you expect for a Tesla car, and how does this compare? We do not have experimental data for this case, but you might try a different case, such as the Ahmed body.
- Try different vehicle speeds – how do you expect the drag force to change? Don't forget to change the rotation speed of the wheels! How are the turbulence quantities calculated at the inlet, and what happens if you change these?
- Is $k-\epsilon$ the best turbulence model to use? Check y^+ values on the vehicle surfaces to ensure the wall model is adequate.
- What information can you extract from the simulation, and what is the best way to present it? Since you will be doing a poster/powerpoint or YouTube video, you might want to put some effort into producing some good visualisations.