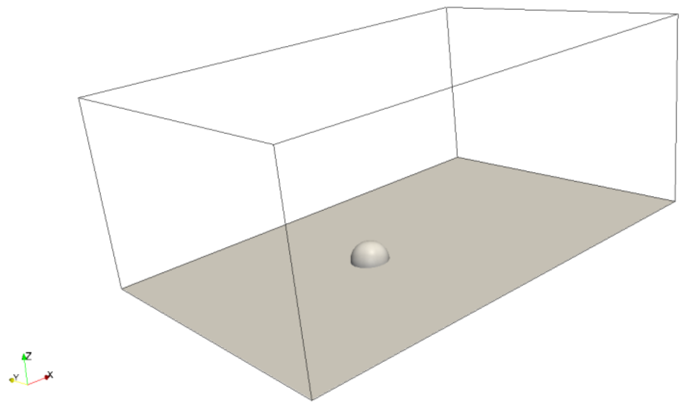
Hairpin vortices

Flow around a hemisphere – Hairpin vortices – Re = 800
Incompressible flow

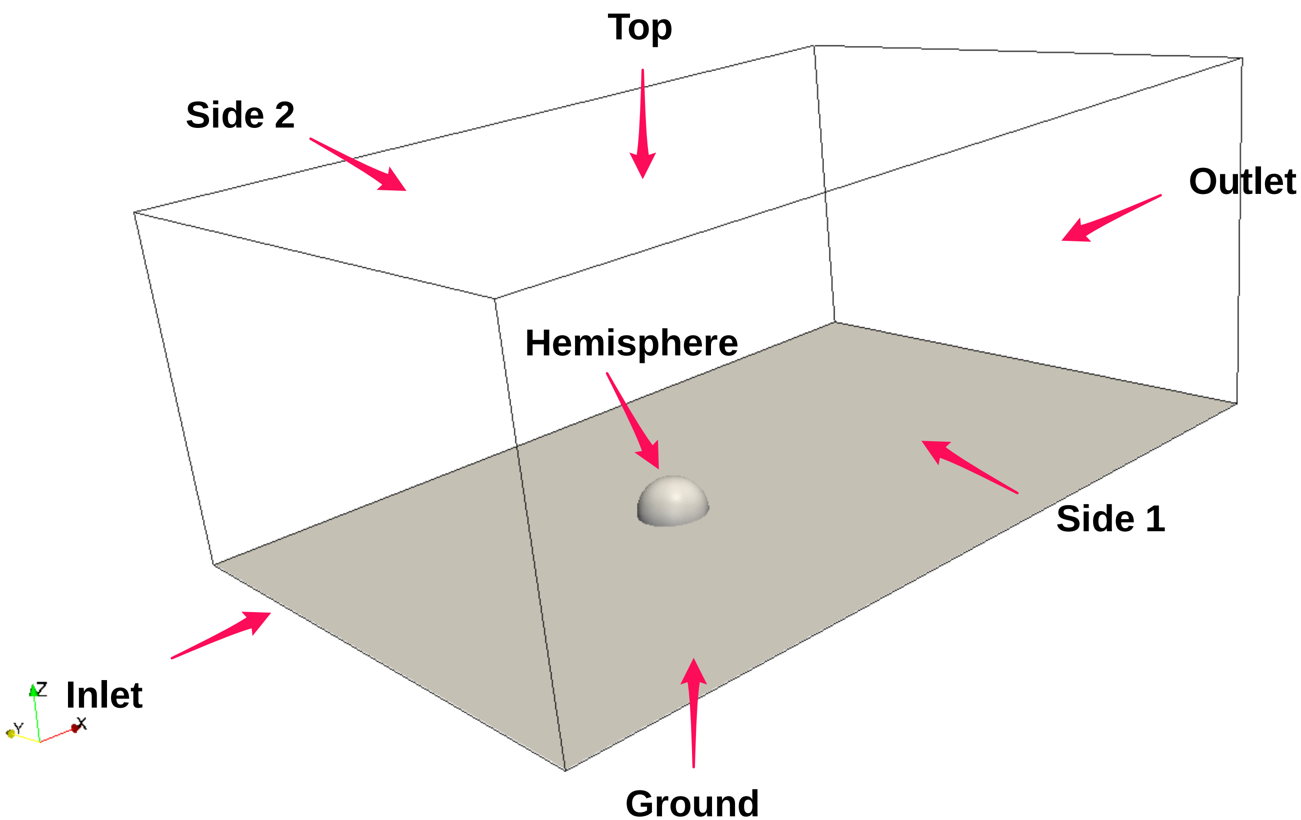


Physical and numerical side of the problem:

- The governing equations of the problem are the incompressible laminar Navier-Stokes equations.
- In this case we are going to solve the flow around a hemisphere.
- For the desired Reynolds number (Re = 800), the flow is fully unsteady, there is a horseshoe vortex in front of the hemisphere, and hairpin vortices develop in the wake of the hemisphere.
- The hairpin vortices forms an interlacing pattern in the wake of the hemisphere and lift away from the wall. The vortices are stretched by the shearing action of the boundary layer since the tails remain in the low-speed (near-wall) region of the flow while the heads are entrained in the high-speed region.
- The vortices are identified using the Q-criterion.

Hairpin vortices

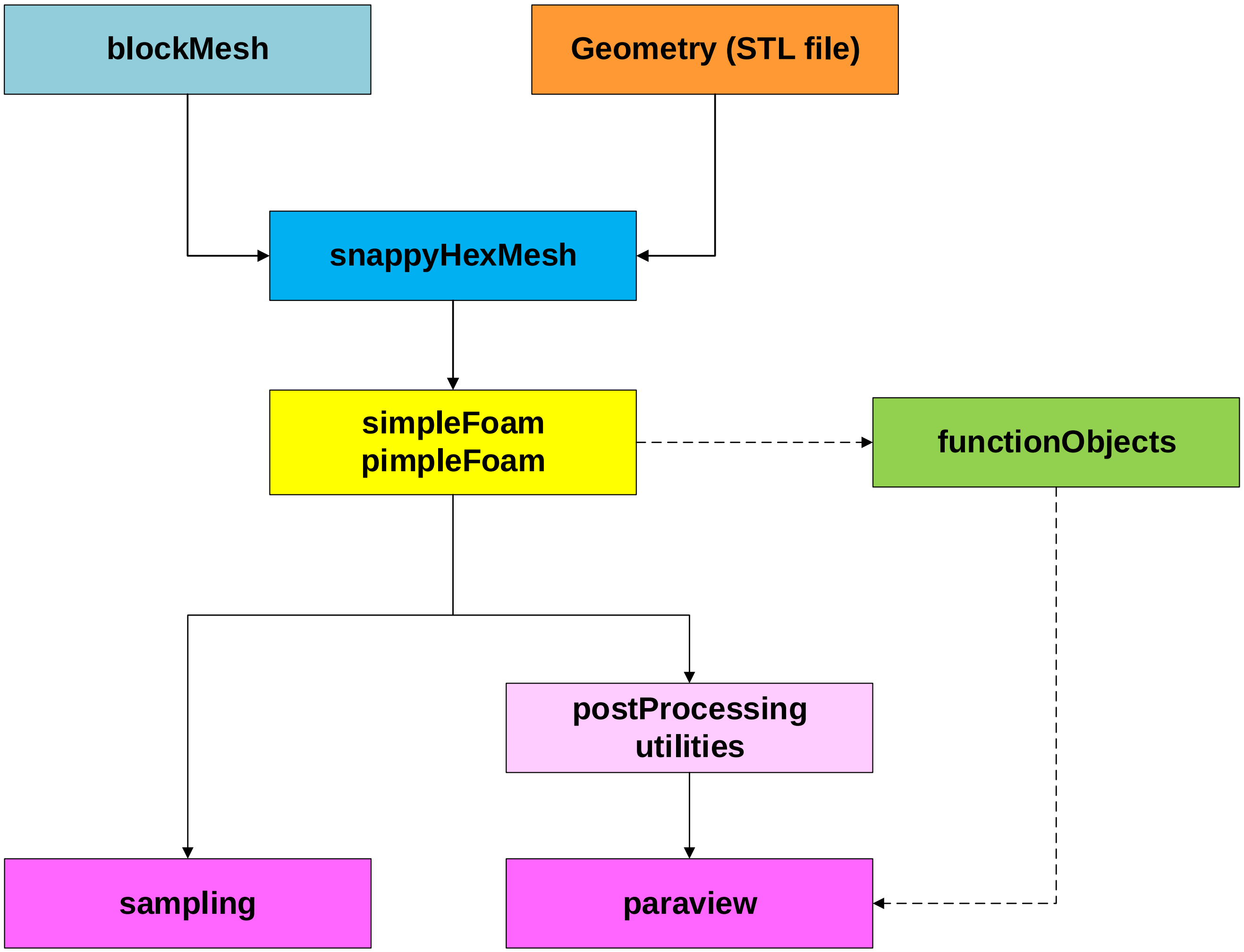
Flow around an hemisphere – Hairpin vortices – Re = 800
Incompressible flow



Boundary patches

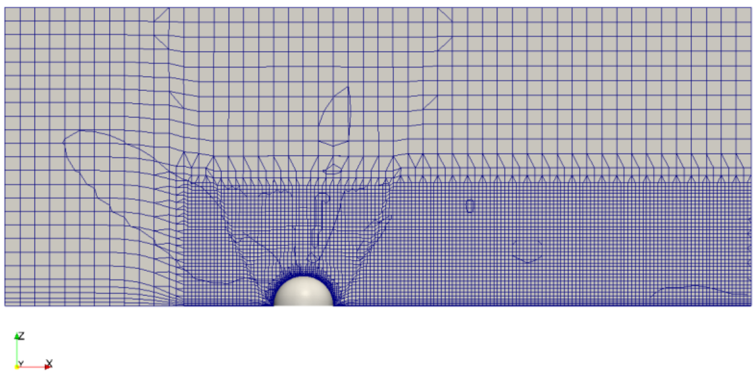
Hairpin vortices

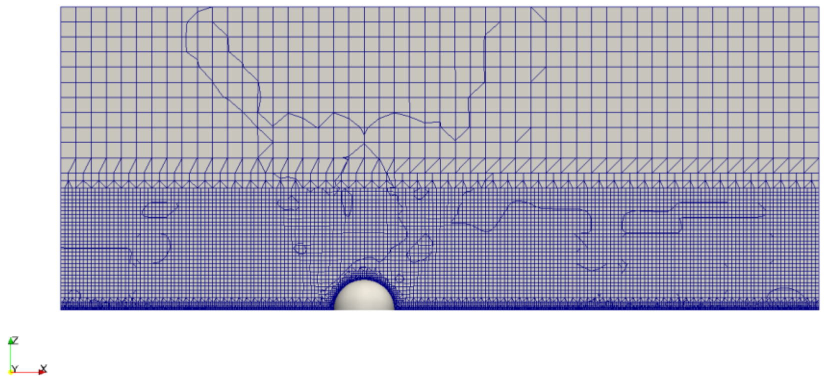
Workflow of the case

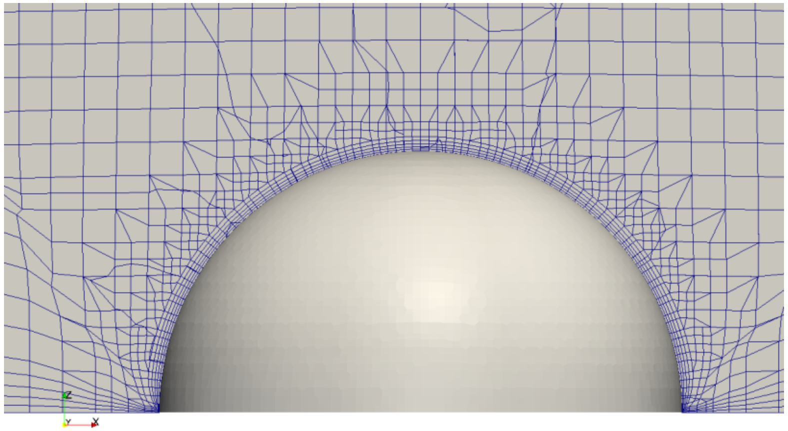


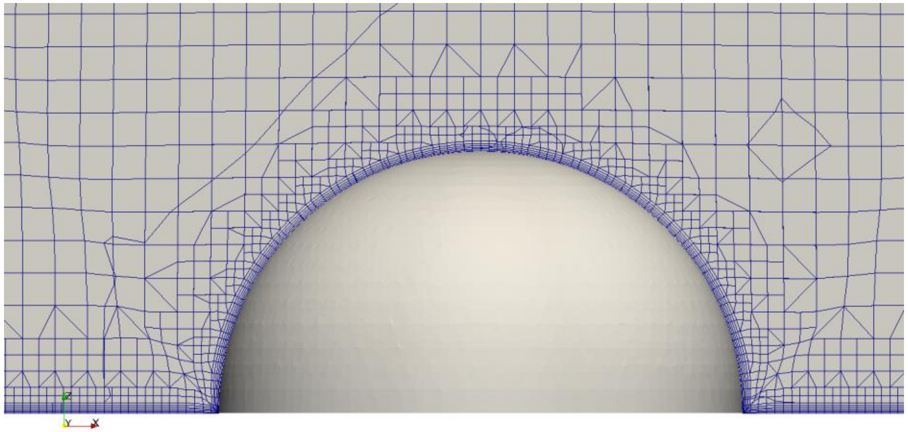
Hairpin vortices

At the end of the day you should get something like this









Coarse mesh

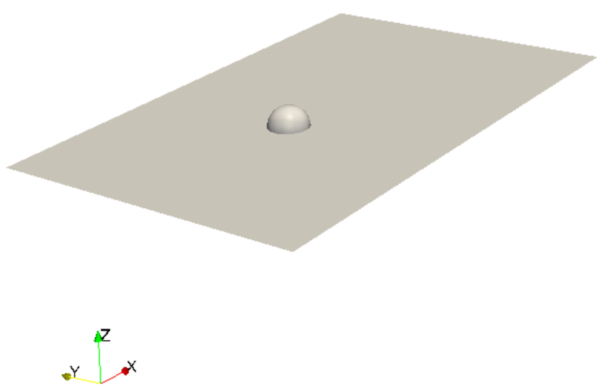
Fine mesh

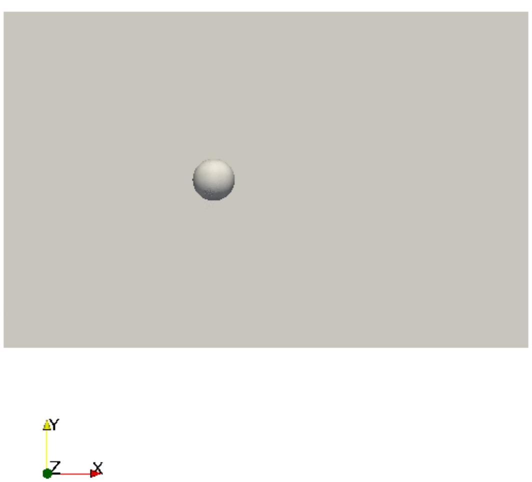
Hairpin vortices

At the end of the day you should get something like this

Unsteady or steady solver?

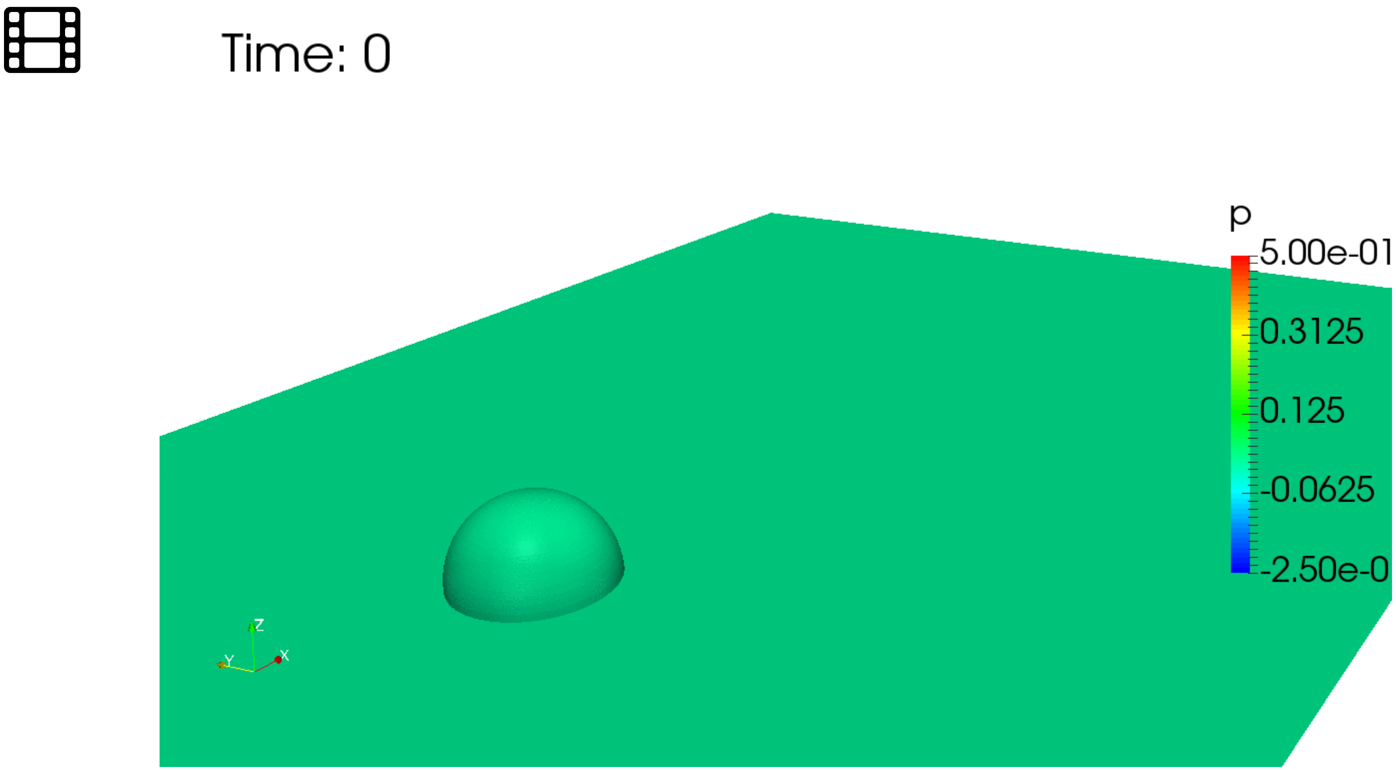
www.wolfdynamics.com/wiki/hairpin_vortices/ste/ani2.gif





Hairpin vortices

At the end of the day you should get something like this

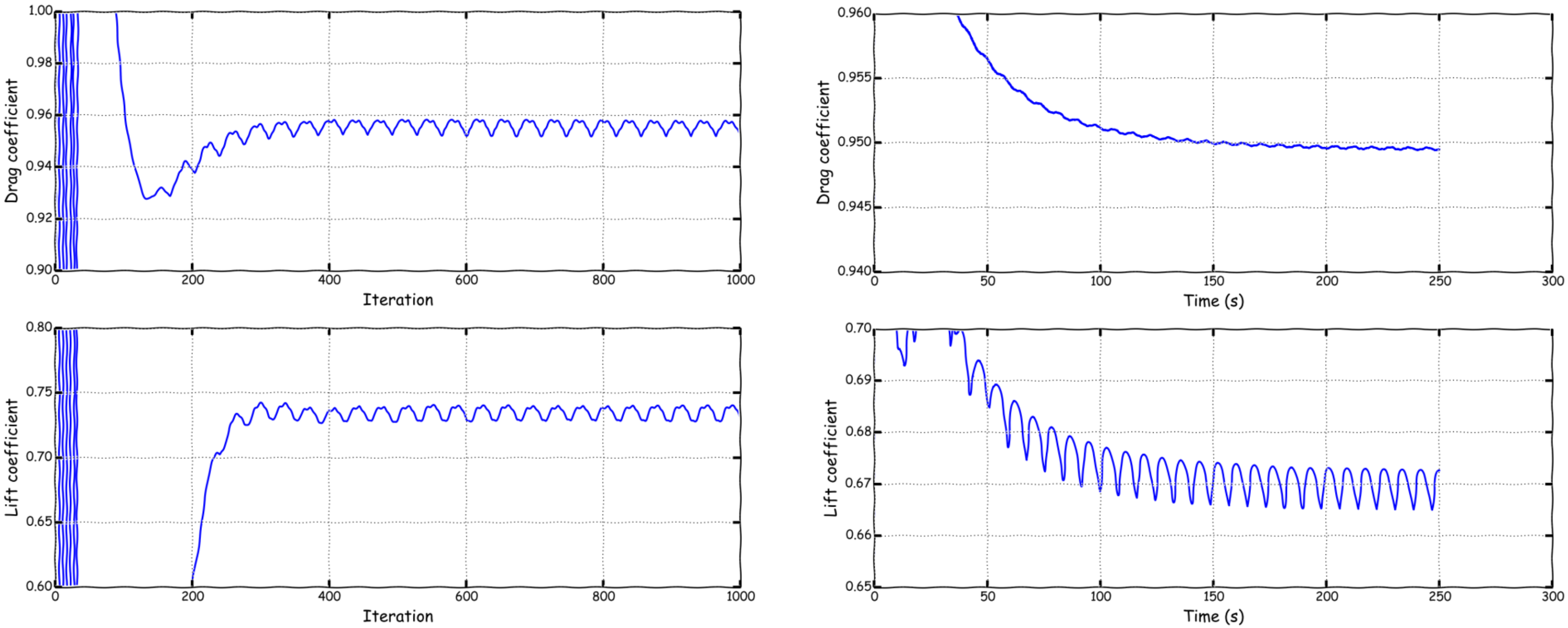


Hairpin vortices – Re = 800 – Unsteady simulation – Pressure field (relative pressure) and vortices visualization using Q-criterion

www.wolfdynamics.com/wiki/hairpin_vortices/uns1/ani1.gif

Hairpin vortices

At the end of the day you should get something like this



Steady simulation

- Steady simulations are not time accurate, hence we can not use them to compute temporal statistics or compute the shedding frequency.
- Generally speaking and in the absence of highly unsteady phenomena, steady simulations should give a result that is close to the mean solution of an unsteady simulation.

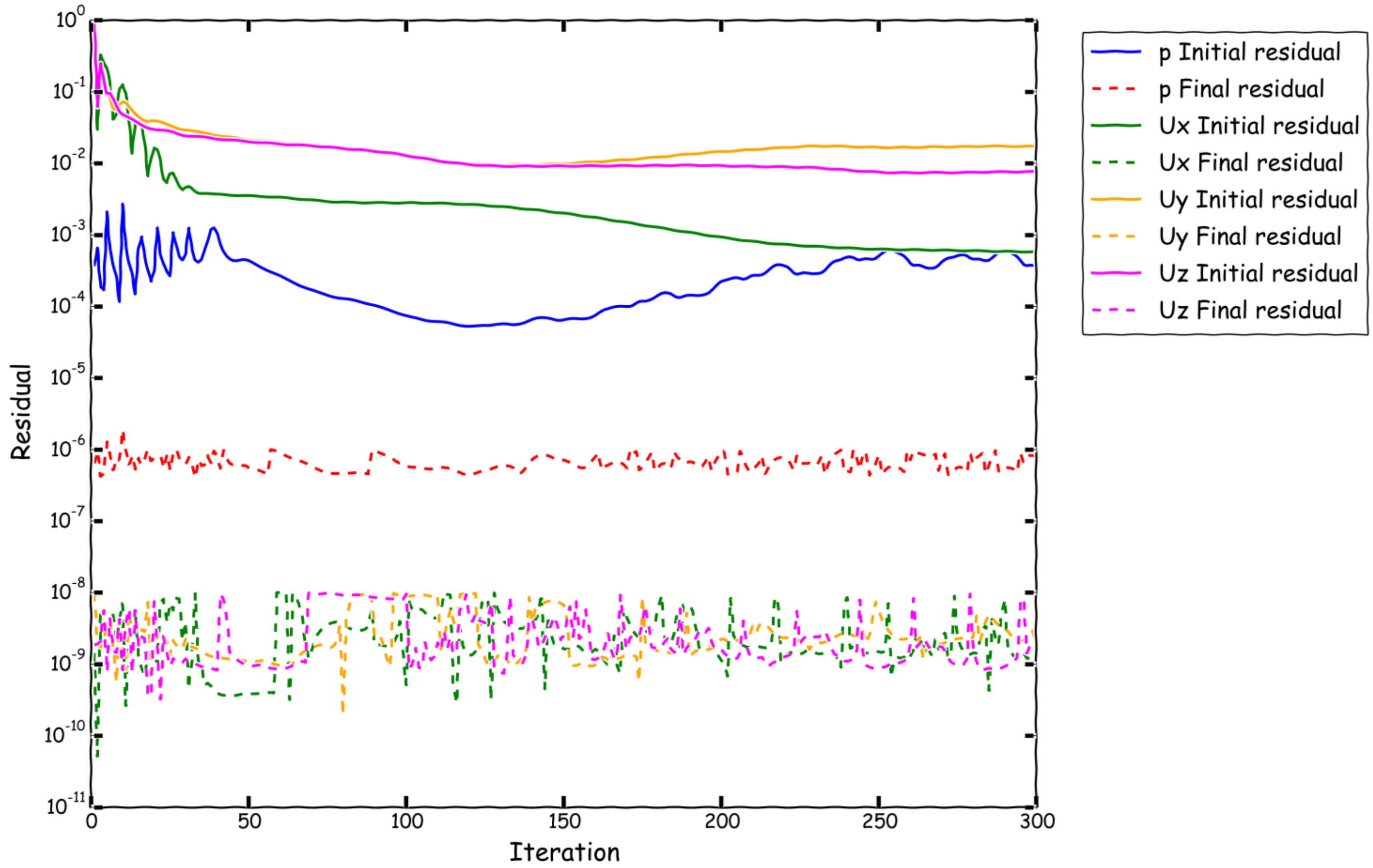
Unsteady simulation

- Unsteady simulations are time-accurate.
- They capture the unsteadiness of the flow (temporal scales).
- You can use these simulations to compute shedding frequency, but remember, you need to define an adequate saving frequency and time-step.
- Numerical diffusion can give you the impression that you have arrived to an steady state.

Drag and lift coefficient signals on the coarse mesh

Hairpin vortices

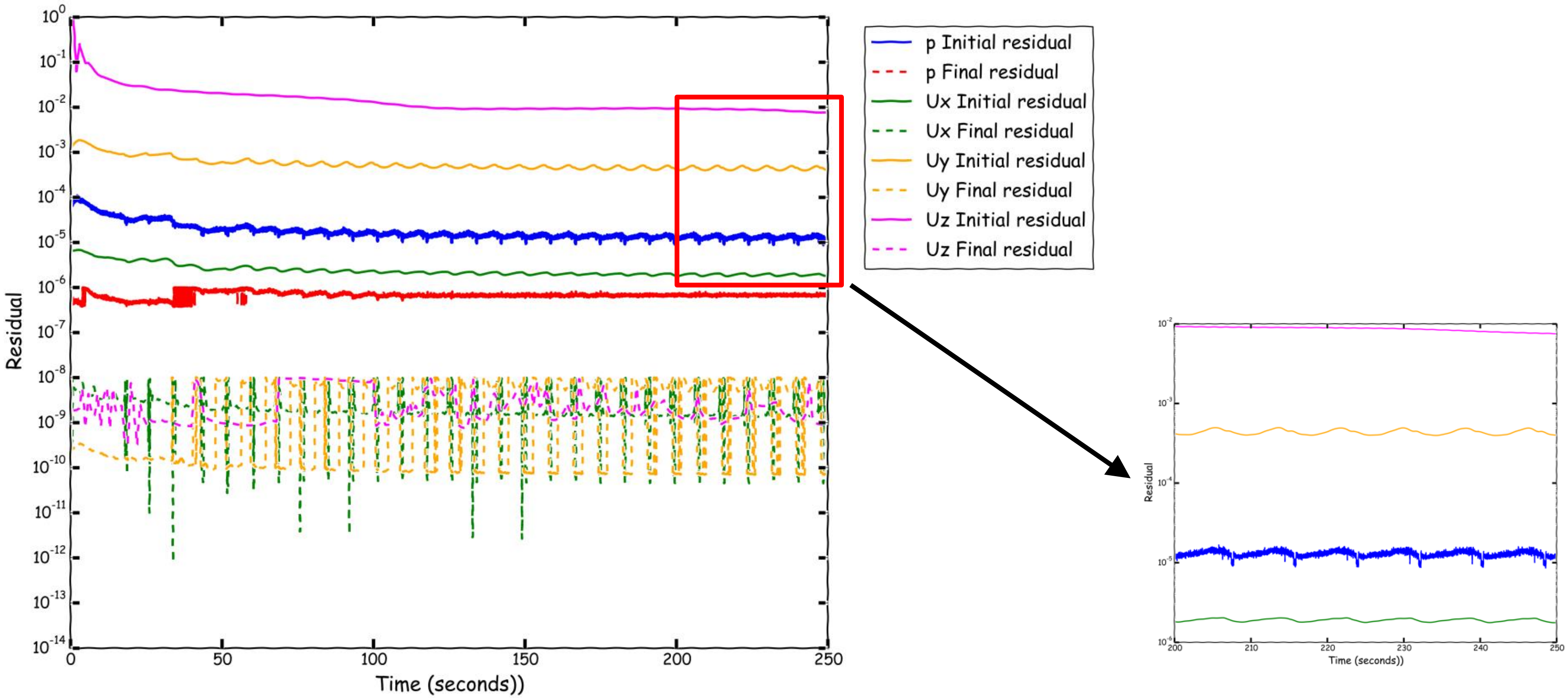
At the end of the day you should get something like this



Steady simulation residuals

Hairpin vortices

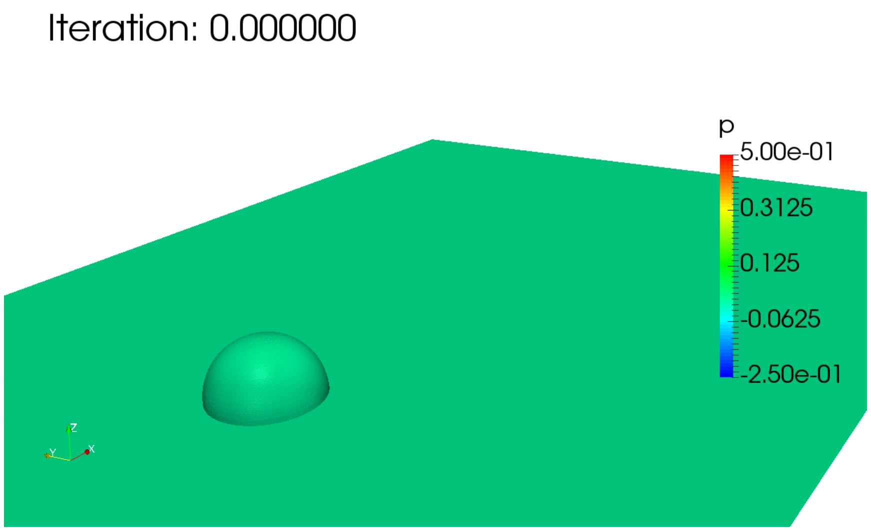
At the end of the day you should get something like this



Unsteady simulation residuals

Hairpin vortices

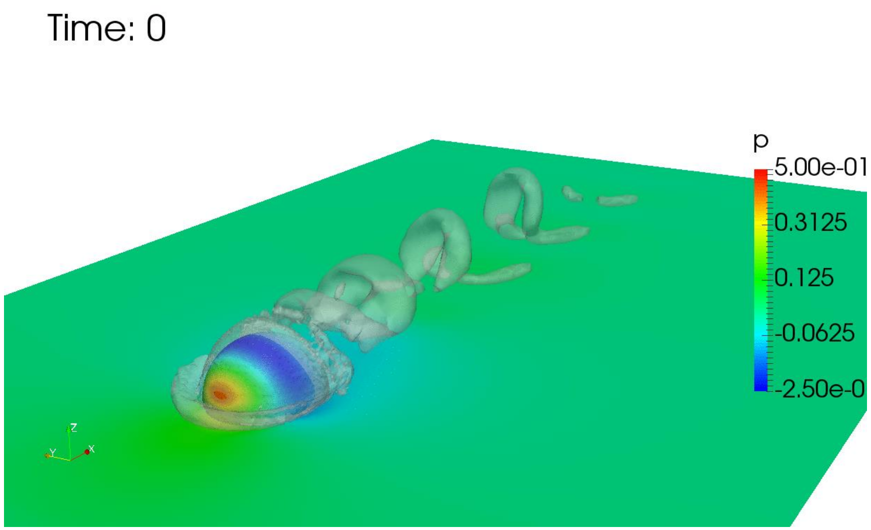
At the end of the day you should get something like this



Steady simulation

www.wolfdynamics.com/wiki/hairpin_vortices/ste/ani1.gif

- Steady simulations are not time accurate, hence we can not use them to compute temporal statistics or compute the shedding frequency.
- Generally speaking and in the absence of highly unsteady flows, steady simulations should give a result that is close to the mean solution of an unsteady simulation.
- Be careful when post-processing steady simulations, the animations you obtain does not represent temporal scales, they only show you how the solution change from iteration to iteration.
- When post-processing steady simulations, you should use the last saved iteration.
- You can also compute the average of a series of snapshots.



Unsteady simulation

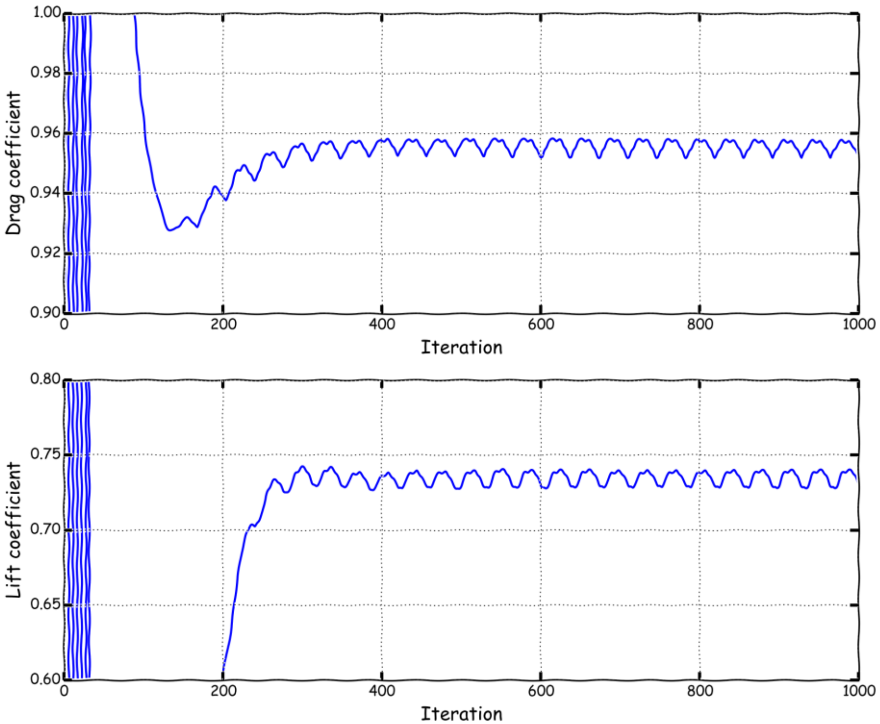
www.wolfdynamics.com/wiki/hairpin_vortices/uns0/ani_smallcfl.gif

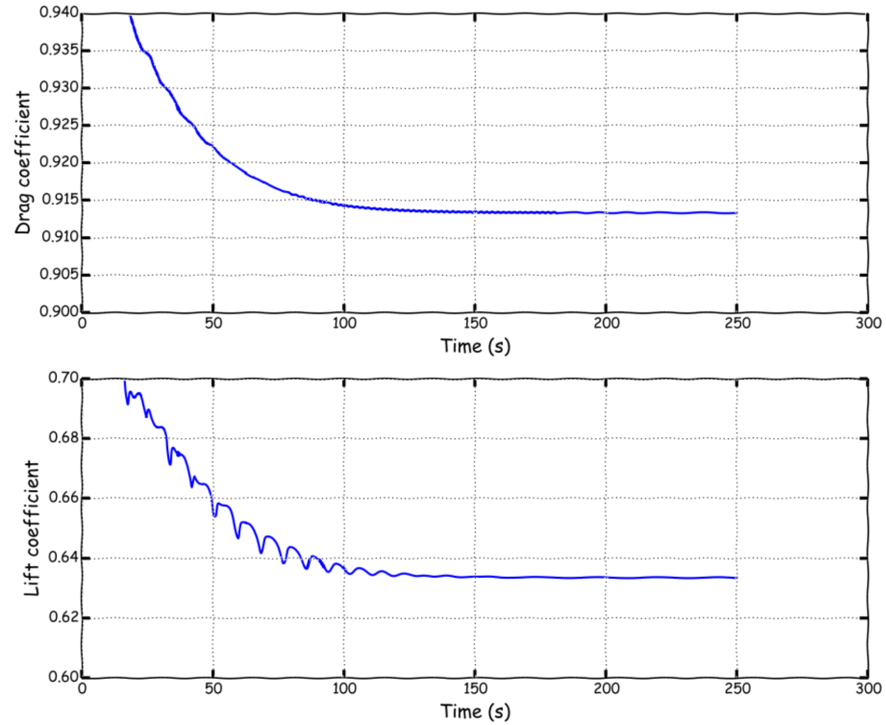
- Unsteady simulation are time-accurate.
- They capture the unsteadiness of the flow (temporal scales).
- You can use these simulations to compute shedding frequency.
- Post-processing unsteady simulations can be difficult and time-consuming.
- When you post-process unsteady simulations, you access all the time-steps saved.
- You can also compute the average of a series of time-steps.
- Remember, you need to define an adequate saving frequency and time-step.
- You can use steady simulations to initialize unsteady simulations.

Drag and lift coefficient signals on the coarse mesh

Hairpin vortices

At the end of the day you should get something like this





Steady simulation

- Steady simulations are not time accurate, hence we can not use them to compute temporal statistics or compute the shedding frequency.
- Generally speaking and in the absence of highly unsteady phenomena, steady simulations should give a result that is close to the mean solution of an unsteady simulation.

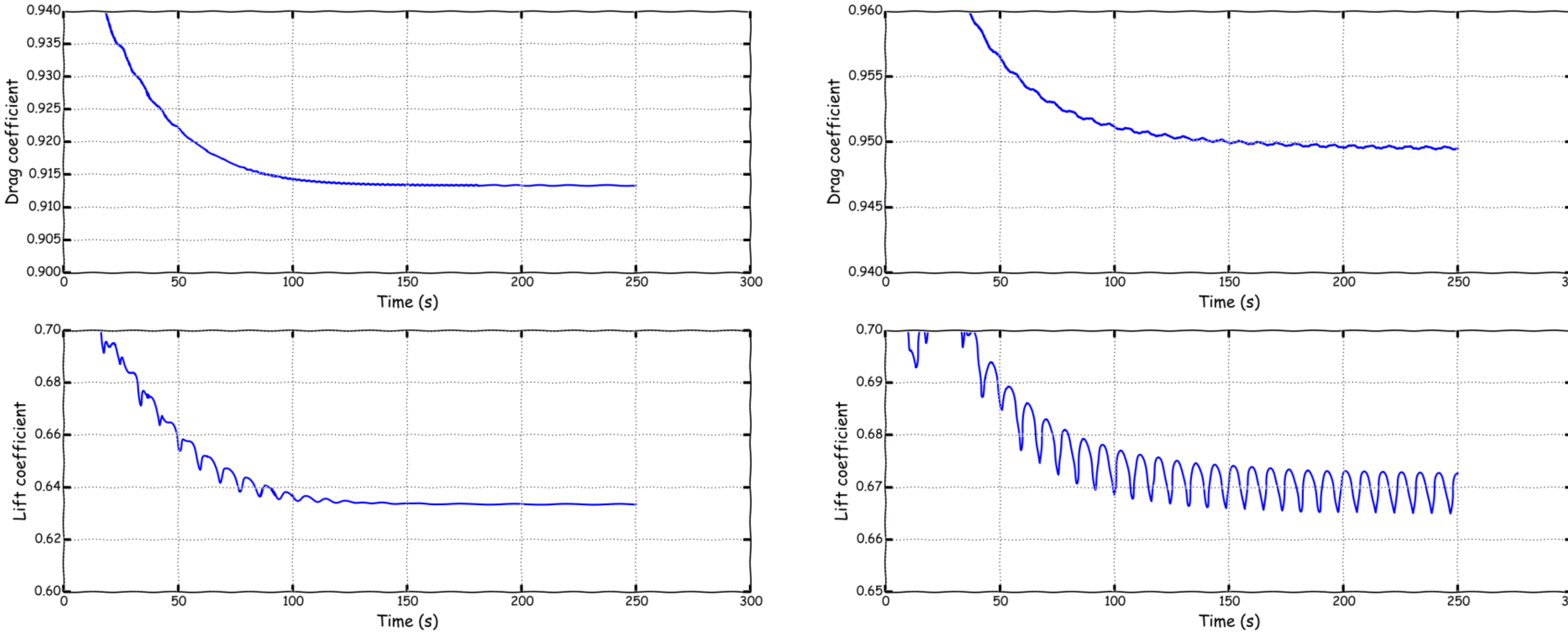
Unsteady simulation

- Unsteady simulations are time-accurate.
- They capture the unsteadiness of the flow (temporal scales).
- You can use these simulations to compute shedding frequency, but remember, you need to define an adequate saving frequency and time-step.
- Numerical diffusion can give you the impression that you have arrived to an steady state.

Drag and lift coefficient signals on the coarse mesh

Hairpin vortices

At the end of the day you should get something like this



Coarse mesh

- The coarse mesh does not capture small spatial scales, hence, they add numerical diffusion to the solution.
- You will have the impression that you have arrived to an steady state.

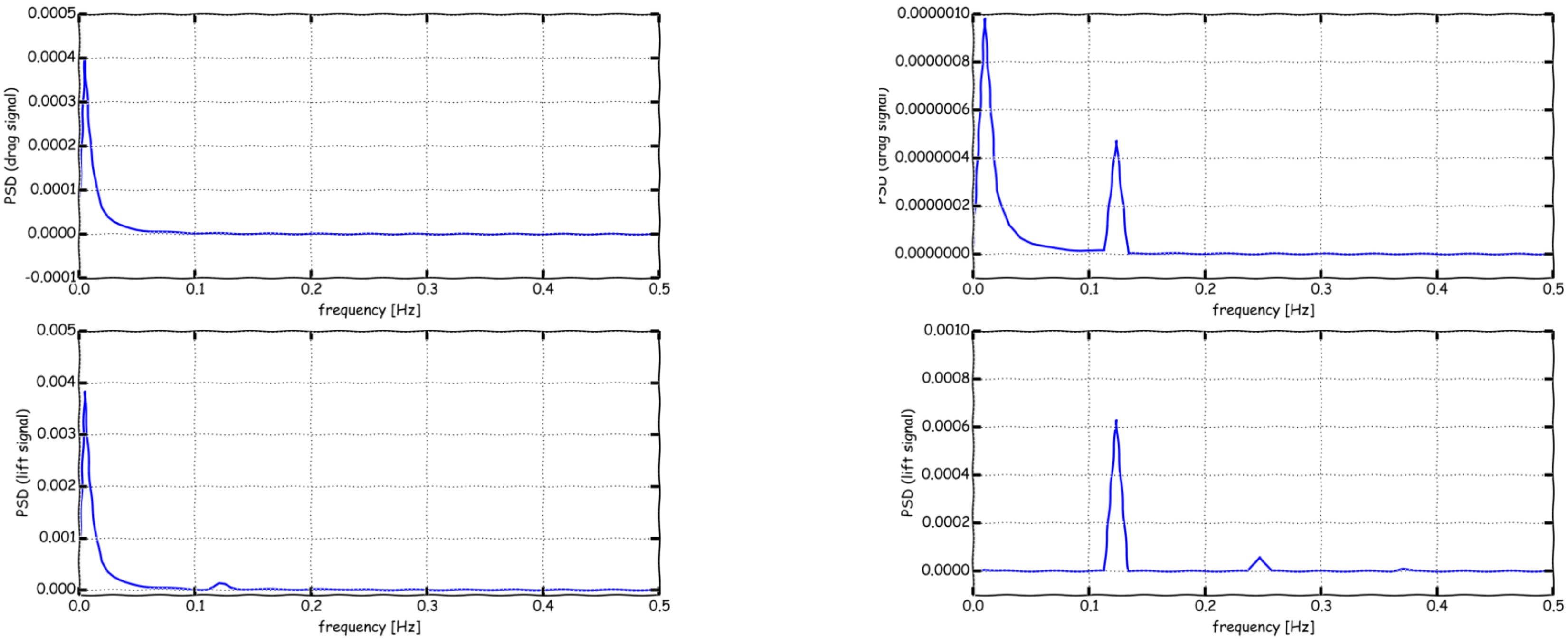
Fine mesh

- The fine mesh captures the small spatial scales that the coarse mesh does not manage to resolve.

Drag and lift coefficient signals – Unsteady simulations

Hairpin vortices

At the end of the day you should get something like this



Coarse mesh

- Due to numerical diffusion (under-resolve temporal and/or spatial scales), it is not possible to use this solution to conduct a temporal analysis of the solution.

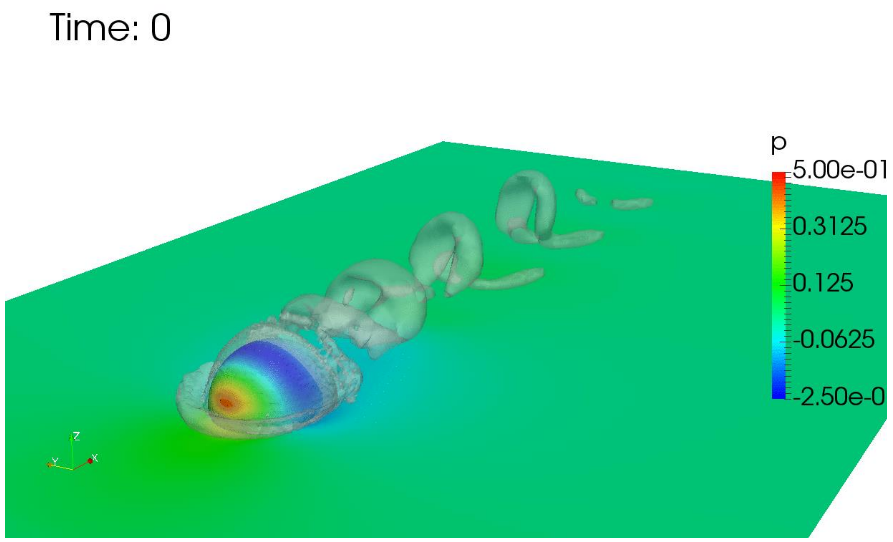
Fine mesh

- As the accuracy is better in the fine mesh, it manages to capture the shedding frequency.

Power spectral density (PSD) of drag and lift coefficient signals

Hairpin vortices

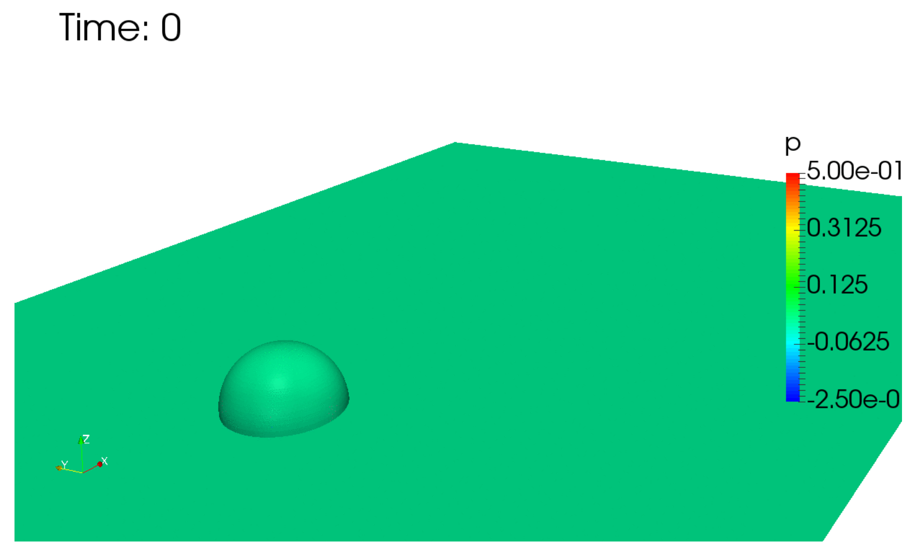
At the end of the day you should get something like this



Coarse mesh

www.wolfdynamics.com/wiki/hairpin_vortices/uns0/ani_smallcfl.gif

- The vortices are dissipated due to numerical diffusion (low mesh resolution)



Fine mesh

www.wolfdynamics.com/wiki/hairpin_vortices/uns1/ani1.gif

- The fine mesh captures the small spatial scales that the coarse mesh does not manage to resolve.

Vortices visualized using Q criterion.

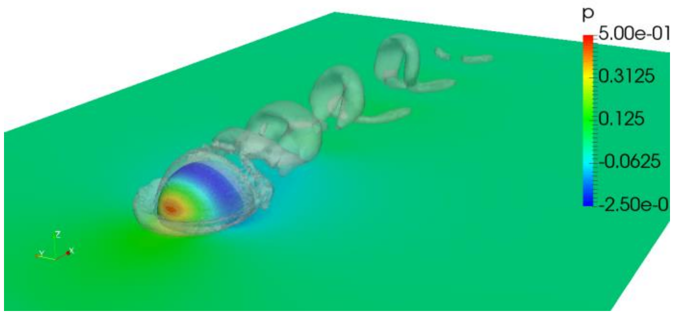
Hairpin vortices

At the end of the day you should get something like this

CFL = 2.0 – Large time-step

www.wolfdynamics.com/wiki/hairpin_vortices/uns0/ani_largecfl.gif

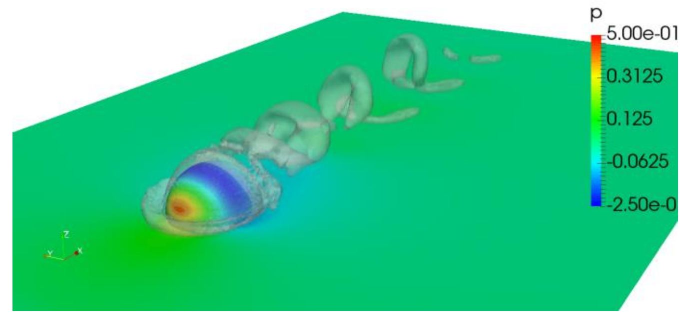
Time: 0

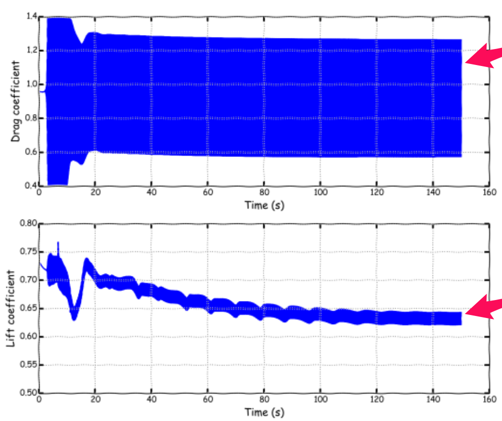


CFL = 0.5 – Small time-step

www.wolfdynamics.com/wiki/hairpin_vortices/uns0/ani_smallcfl.gif

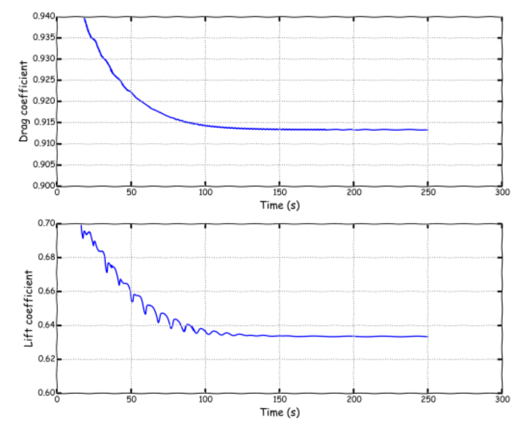
Time: 0





Unphysical oscillations due to large CFL number

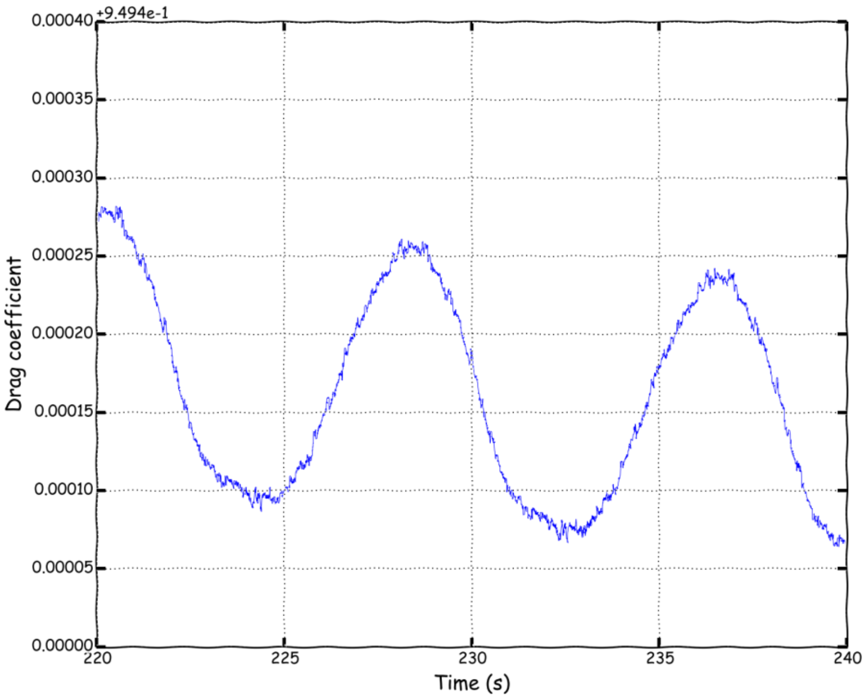
Unphysical oscillations due to large CFL number

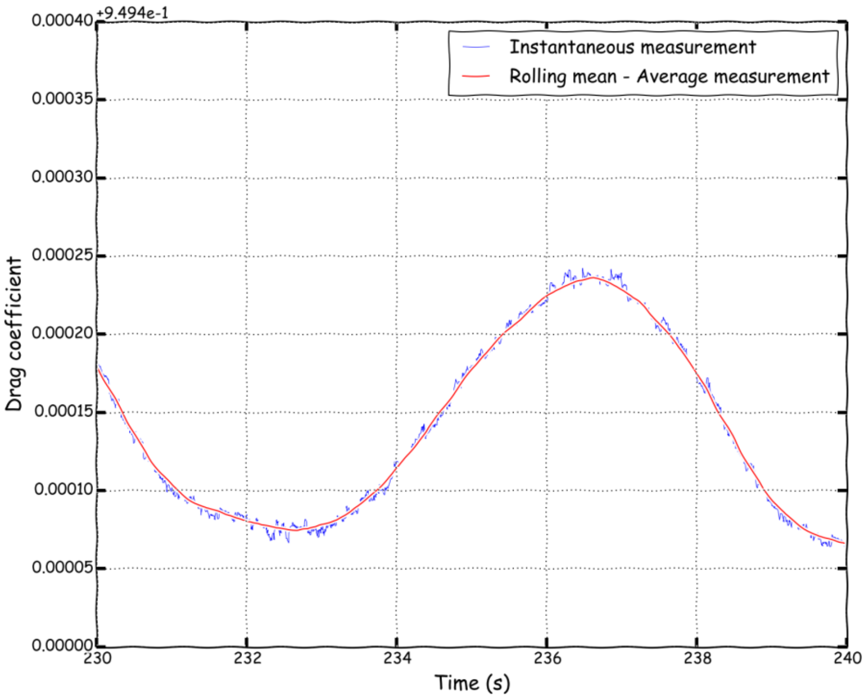


Drag and lift coefficient signals for different CFL numbers on the coarse mesh – Unsteady simulation
Vortices visualized using Q criterion.

Hairpin vortices

At the end of the day you should get something like this

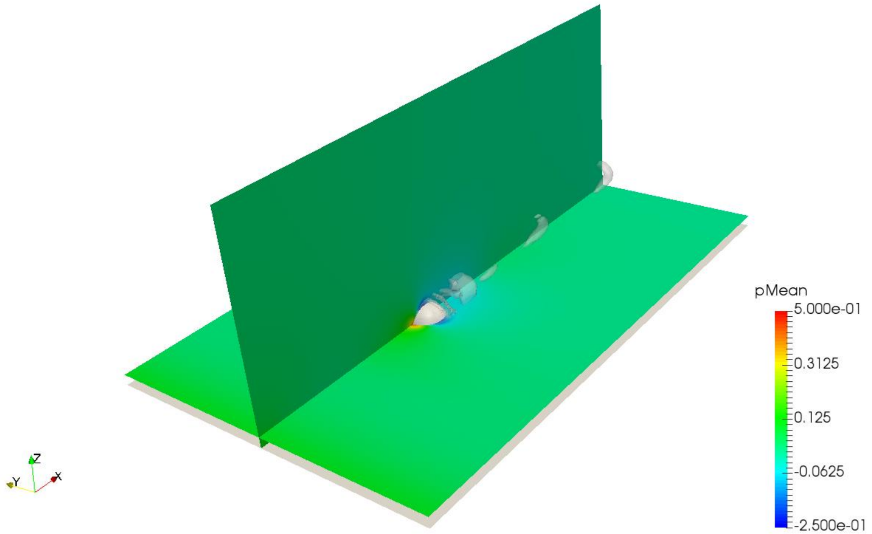


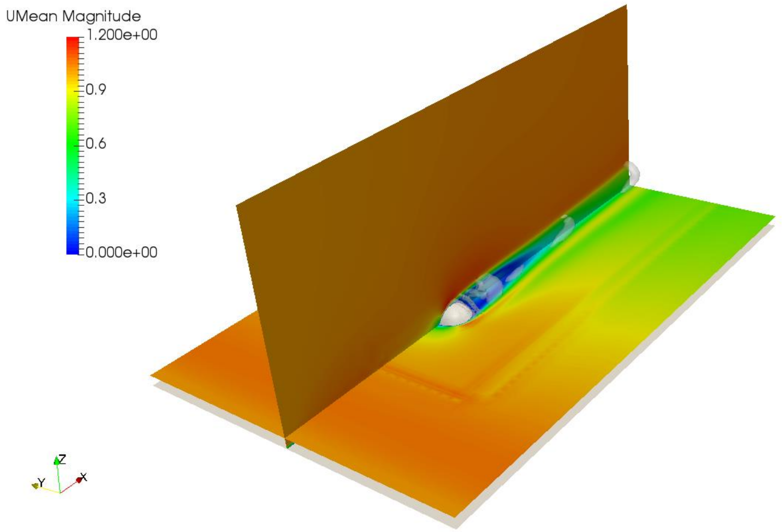


Drag coefficient signal on the fine mesh – Instantaneous value and average value (rolling mean)

Hairpin vortices

At the end of the day you should get something like this



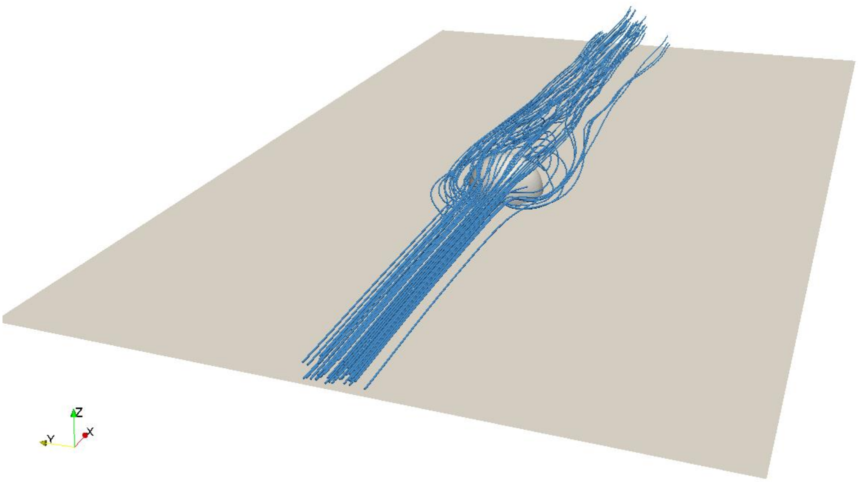


Cutplanes colored by pressure mean value contours and iso-surfaces of Q-criterion

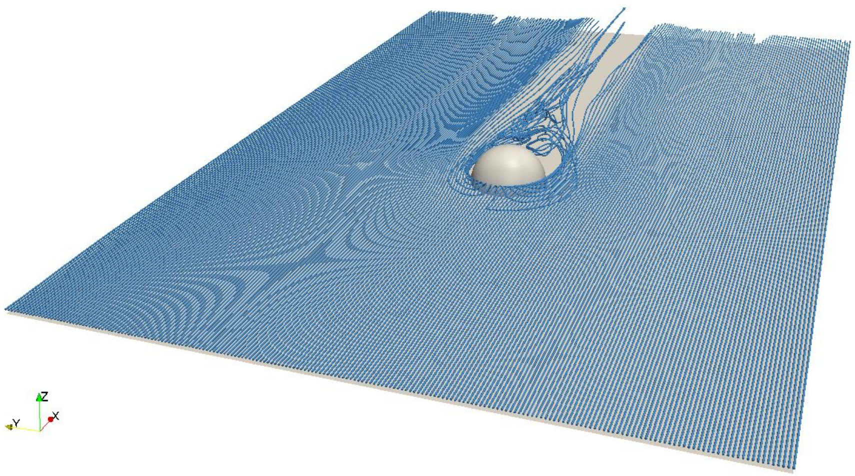
Cutplanes colored by velocity magnitude mean value contours and iso-surfaces of Q-criterion

Hairpin vortices

At the end of the day you should get something like this



Streamlines released from a point source



Streamlines released from a line source

Hairpin vortices

- Let us run our first case. Go to the directory:

```
$PTOFC/hairpin_vortices
```

- $PTOFC is pointing to the directory where you extracted the training material.
- 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 you to open the `README.FIRST` file 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®, use the automatic scripts to run the cases.

Hairpin vortices

What are we going to do?

- In this tutorial, we use `simpleFoam` and `pimpleFoam` in a 3D domain.
- The solver `simpleFoam` is formulated for steady simulations and the solver `pimpleFoam` is formulated for unsteady simulations.
- Running a 3D simulation is not different from the previous 2D simulations. The only difference is that we need to define the boundary conditions in the third dimension, and the simulation requires more computational resources.
- We will generate the mesh using snappyHexhMesh.
- We will use the steady solution as the starting point for a unsteady solution
- We will map the solution from a coarse mesh to a finer mesh.
- We will visualize unsteady data.

Hairpin vortices

Running the case
Let's run a steady simulation

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/ste`**
- Let's first generate the mesh. To generate the mesh will use snappyHexMesh (sHM), do not worry we will talk about sHM tomorrow.
- In the terminal window type:

1. `$> foamCleanTutorials`
2. `$> blockMesh`
3. `$> surfaceFeatureExtract`
4. `$> snappyHexMesh -overwrite`
5. `$> checkMesh`
6. `$> paraFoam`

Hairpin vortices

Running the case
Let's run a steady simulation

- In step 2 we use `blockMesh` to generate the background mesh for `snappyHexMesh`.
- In step 3 we use the utility `surfaceFeatureExtract` to extract geometry features (edges) for `snappyHexMesh`. This utility reads the dictionary *`system/surfaceFeatureExtractDict`*.
- In step 4 we use `snappyHexMesh` to generate the 3D mesh. It will read the dictionary *`system/snappyHexMeshDict`*.
- In step 5 we check the topology and mesh quality.
- Finally, in step 6 we visualize the mesh.

Hairpin vortices

Running the case
Let's run a steady simulation

- As usual, remember to take a look at the file `boundary` and adapt it to your needs. In this case we already assigned the right **names** and **base type** to all the boundary patches.
- At this point, we are all familiar with the dictionaries.
- But it will not hurt you to take a look at them again. Feel free to do any modification.
- Reminder:
  - The diameter of the hemi-sphere is 2.0 m.
  - And we are targeting for a Re = 800.

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

Hairpin vortices

Running the case
Let's run a steady simulation

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/ste`**
- Run this case just 400 iterations.
- Let's run the simulation using `simpleFoam`. In the terminal window type:

1. `$> renumberMesh -overwrite`
2. `$> simpleFoam > log.simplefoam &`
3. `$> pyFoamPlotWatcher.py log.simplefoam`
4. `$> Q`
5. `$> paraFoam`

Hairpin vortices

Running the case
Let's run a steady simulation

- In step 1 we use the utility `renumberMesh` to make the linear system more diagonal dominant, this will speed-up the linear solvers.
- In step 2 we run the simulation and save the log file. Notice that we are sending the job to background.
- In step 3 we use `pyFoamPlotWatcher.py` to plot the residuals on-the-fly. As the job is running in background, we can launch this utility in the same terminal tab.
- In step 4 we compute the Q-Criterion (for vortex visualization).
- Finally, in step 5 we visualize the solution.

Hairpin vortices

Running the case
Let's run a steady simulation

- By the way, to visualize the results you do not need to wait until the simulation is over. You can launch `paraFoam` at anytime. Remember, you will need to open a new terminal.
- Also, we do not need to run this case until the **endTime** (1000 iterations). Just run a few iterations (about 400 iterations), monitor the solution and try to do some post-processing.
- Do not erase the solution, as we are going to use it as initial conditions for the transient simulation.
- And as the meshes are the same, we only need to copy the last saved solution of the steady simulation into the directory containing the initial conditions of the transient simulation.
- If you are in a hurry, in the compressed files *sol_constant.tar.gz* and *sol_1000.tar.gz* you will find the mesh and the solution respectively. To uncompress the files, type on the terminal:

```
1. $> tar -xzvf sol_constant.tar.gz
2. $> tar -xzvf sol_1000.tar.gz
```

Hairpin vortices

Running the case

Let's run an unsteady simulation using the solution from the steady case

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/uns0`**
- Let's copy to the current case directory the last saved solution from the steady simulation. We will use this solution as boundary and initial conditions for the unsteady simulation.
- In the terminal window type:

1. `$> cd $PTOFC/101OF/hairpin_vortices/`
2. `$> cd uns0`
3. `$> mkdir 0`
4. `$> cp ../ste/1000/U ./0`
   Assuming that you run 1000 iterations. If you run more or less iterations, just copy the last saved solution.
5. `$> cp ../ste/1000/p ./0`
   Assuming that you run 1000 iterations. If you run more or less iterations, just copy the last saved solution.

- At this point, if you want to change the boundary conditions, just open the field dictionaries and modify them. Remember, as it is a big file you better use vi or emacs.

Hairpin vortices

Running the case

Let’s run an unsteady simulation using the solution from the steady case

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/uns0`**
- Remember, to run a simulation we need a mesh.
- As the mesh is the same as the one we used in the case `../`**`ste`**, we do not need to redo it. Just copy the mesh from the steady case or unpack the file *`sol_constant.tar.gz`*
- Let’s copy the mesh from the steady case. In the terminal window type:

1.
```
$> cp -r ../ste/constant/polyMesh/ constant/
```

- Alternatively:

1.
```
$> tar -xzvf sol_constant.tar.gz
```

Hairpin vortices

Running the case

Let’s run an unsteady simulation using the solution from the steady case

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/uns0`**
- Now we are ready to run the simulation. In the terminal window type:

```
1. $> renumberMesh -overwrite
2. $> pimpleFoam > log.pimplefoam &
3. $> pyFoamPlotWatcher.py log.pimplefoam
4. $> Q
5. $> paraFoam
```

Hairpin vortices

Running the case

Let's run an unsteady simulation using the solution from the steady case

- By the way, to visualize the results you do not need to wait until the simulation is over. You can launch `paraFoam` at anytime. Remember, you will need to open a new terminal.
- Also, we do not need to run this case until the **endTime** (250 seconds). Just run a few seconds of simulation time (about 10 seconds), monitor the solution and try to do some post-processing.
- Remember, unsteady solvers are much much time consuming than steady solvers.
- If you have time you can run the simulation until the **endTime**. At the end of the simulation, compare the force coefficients of the steady with the force coefficients of the unsteady simulations.
- Do not erase the solution, as we are going to use it as initial conditions for the transient simulation using a finer mesh.

Hairpin vortices

Running the case

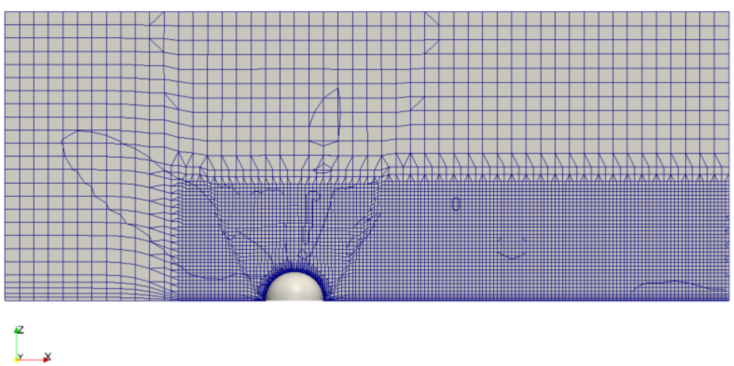
Let's run an unsteady simulation using the solution from the steady case

- In step 1 we use the utility `renumberMesh` to make the linear system more diagonal dominant, this will speed-up the linear solvers.
- In step 2 we run the simulation and save the log file. Notice that we are sending the job to background.
- In step 3 we use `pyFoamPlotWatcher.py` to plot the residuals on-the-fly. As the job is running in background, we can launch this utility in the same terminal tab.
- In step 4 we compute the Q Criterion (for vortex visualization).
- Finally, in step 5 we visualize the solution.

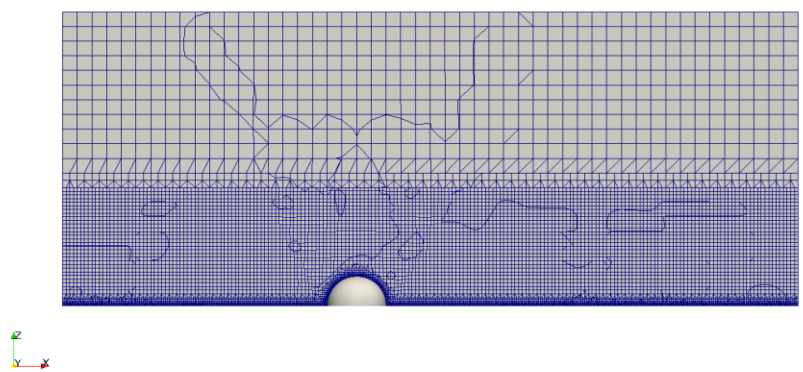
Hairpin vortices

Running the case
Let's run the same case using a finer mesh

- In the directory **`uns1`**, you will find the same case setup as in **`uns0`**. The only difference is that we are using a finer mesh.
- In other words, we are conducting a mesh refinement study.
- The new mesh will resolve better the wake behind the hemisphere and the boundary layer at the ground.
- It will also resolve better the boundary layer of the hemisphere (it will predict better the forces).
- Qualitative and quantitative speaking the results will be better.



Coarse mesh



Fine mesh

Hairpin vortices

Running the case
Let's run the same case using a finer mesh

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/uns1`**
- Let's first generate the mesh. Generating the mesh in this case is a little bit time consuming, therefore let's use a pre-generated mesh.
- In the terminal window type:

1. `$> foamCleanTutorials`
2. `$> foamCleanPolyMesh`
3. `$> tar -xzvf sol_constant.tar.gz`
   To uncompress the pre-generated mesh

- This case is computationally intensive. The mesh is about 3.8 millions cells. So you need to have at least 4 Gigs of memory and a lot of patience if you are running in serial.

Hairpin vortices

Running the case
Let’s run the same case using a finer mesh

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/uns1`**
- If you want to generate the mesh, in the terminal window type:

1. `$> foamCleanTutorials`
2. `$> blockMesh`
3. `$> surfaceFeatureExtract`
4. `$> snappyHexMesh -overwrite`
5. `$> checkMesh`

Generating this mesh can be time consuming

Hairpin vortices

Running the case
Let’s run the same case using a finer mesh

- `checkMesh` output

```
Create time

Create polyMesh for time = 0

Time = 0

Mesh stats
    points:           4222485
    faces:            11827716
    internal faces:   11407628
    cells:            3803904
    faces per cell:   6.1082887
    boundary patches: 7
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     3655280
    prisms:        1042
    wedges:        8
    pyramids:      0
    tet wedges:    8
    tetrahedra:    0
    polyhedra:     147566
    Breakdown of polyhedra by number of faces:
        faces   number of cells
            4   1224
            5   818
            6   1580
            7   8812
            8   3934
            9   129680
           10   18
           12   1016
           15   484
```

Total number of cells

Cells breakdown by element type

Breakdown of polyhedra.
Polyhedra with many faces can give problems.

Hairpin vortices

Running the case
Let’s run the same case using a finer mesh

- `checkMesh` output

```
Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
                   Patch    Faces   Points                  Surface topology
                    minx     7092     7627   ok (non-closed singly connected)
                    maxx     7092     7627   ok (non-closed singly connected)
                    miny     9100     9934   ok (non-closed singly connected)
                    maxy     9100     9934   ok (non-closed singly connected)
                    minz   380984   382316   ok (non-closed singly connected)
                    maxz     1500     1581   ok (non-closed singly connected)
             semi_sphere     5220     6641   ok (non-closed singly connected)

Checking geometry...
    Overall domain bounding box (-10 -8 -2.4459601e-16) (15 8 10)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (-4.651149e-15 -1.1428046e-15 1.1290504e-13) OK.
    Max cell openness = 6.6352707e-16 OK.
    Max aspect ratio = 14.34437 OK.
    Minimum face area = 3.6373532e-05. Maximum face area = 0.27049071.  Face area magnitudes OK.
    Min volume = 6.2267806e-07. Max volume = 0.13497113.  Total volume = 3997.9127.  Cell volumes OK.
    Mesh non-orthogonality Max: 63.660466 average: 5.7030887
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 3.5522648 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

Boundary patches

Aspect ratio OK

Non-orthogonality OK

Skewness OK

Hairpin vortices

Running the case
Let's run the same case using a finer mesh

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/uns1`**
- Let's map the solution from the steady case or the unsteady case **`uns0`** (is up to you).
- Hereafter we will use the solution form the steady case. In the terminal window type:

1. `$> rm -rf 0 > /dev/null 2>&1`
2. `$> cp -r 0_org 0`
3. `$> mapfields ../ste -consistent -noFunctionObjects -mapMethod cellPointInterpolate -sourceTime 'latestTime'`
4. `$> paraFoam`

Hairpin vortices

Running the case
Let's run the same case using a finer mesh

- In step 3 we run the utility `mapFields` with the following options:
  - We copy the solution from the directory `../ste`
  - The options `-consistent` is used when the domains and BCs are the same.
  - The option `-noFunctionObjects` is used to avoid conflicts with the **functionObjects**.
  - The option `-mapMethod cellPointInterpolate` defines the interpolation method.
  - The option `-sourceTime 'latestTime'` defines the time from which we want to interpolate the solution.

- This step will give you a lot warnings, just ignore them.
- Remember, `mapFields` will interpolate everything it finds in the source directory. Just keep the files you need in the target directory. It is also a good idea to have a backup of the original BC/IC.

Hairpin vortices

Running the case
Let's run the same case using a finer mesh

- If you try to open this mesh using `paraFoam`, it will take about 50 seconds (at least on my workstation with a good video card).
- An alternative to `paraFoam` is to use `paraview` (a native installation). In our workstation, it takes about 15 seconds to open the same mesh.
- Remember, to open the case with `paraview`, you will need to create the file *`case_name.foam`* manually and then open it in paraview. In the terminal type:

1. `$> touch case_name.foam`
2. `$> paraview5`

   This is an alias that points to a native paraview 5 installation

Hairpin vortices

Running the case
Let's run the same case using a finer mesh

- You will find this tutorial in the directory **`$PTOFC/hairpin_vortices/uns1`**
- Now we are ready to run the simulation. In the terminal window type:

1. `$> renumberMesh -overwrite`
2. `$> pimpleFoam > log.pimplefoam &`
3. `$> pyFoamPlotWatcher.py log.pimplefoam`
4. `$> Q`
5. `$> paraFoam`

Hairpin vortices

Running the case
Let's run the same case using a finer mesh

- In step 1 we use the utility `renumberMesh` to make the linear system more diagonal dominant, this will speed-up the linear solvers.
- In step 2 we run the simulation and save the log file. Notice that we are sending the job to background.
- In step 3 we use `pyFoamPlotWatcher.py` to plot the residuals on-the-fly. As the job is running in background, we can launch this utility in the same terminal tab.
- In step 4 we compute the Q Criterion (for vortex visualization).
- Finally, in step 5 we visualize the solution.

Hairpin vortices

Running the case
Let’s run the same case using a finer mesh

- We just run the simulation using `pimpleFoam`, if you want you can use `pisoFoam`.
- The main differences are:
  - No outer loops in `pisoFoam`.
  - No adjustable time stepping in `pisoFoam`.

- FYI, in the *`fvSolution`* dictionary you will need to add the related entry for the **PISO** pressure-velocity coupling method (same entry as in the `icoFoam` solver).

```
PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 1;
}
```

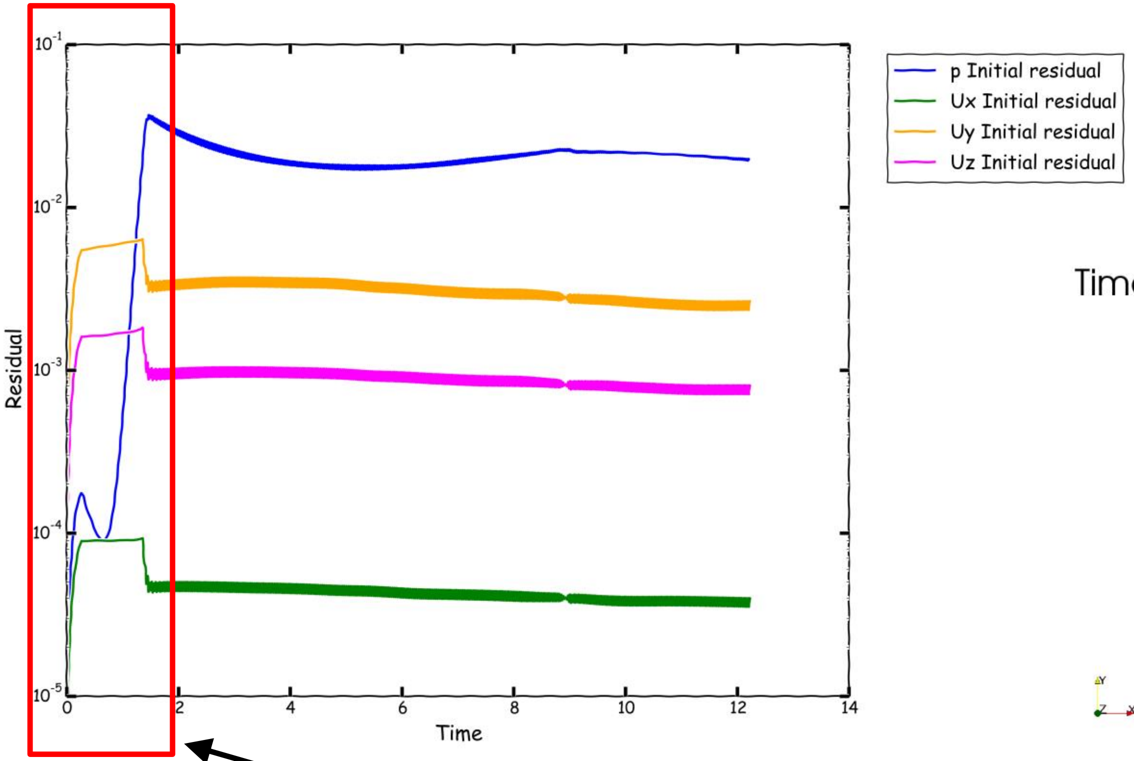
Hairpin vortices

Running the case
Let's run the same case using a finer mesh

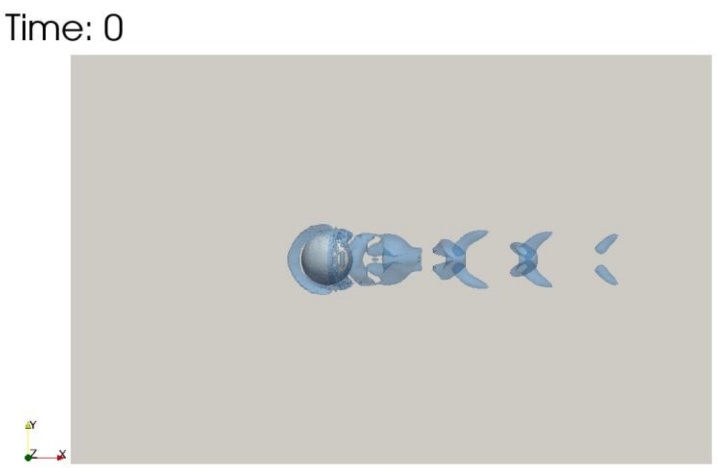
- By the way, to visualize the results you do not need to wait until the simulation is over. You can launch `paraFoam` at anytime. Remember, you will need to open a new terminal.
- Also, we do not need to run this case until the **endTime** (250 seconds). Just run a few seconds of simulation time (about 2 seconds), monitor the solution and try to do some post-processing.
- This case is computationally intensive. The mesh is about 3.8 millions cells. So you need to have at least 4 Gigs of memory and a lot of patience if you are running in serial.
- Running in parallel will speed-up everything. Tomorrow we are going to address how to run in parallel.
- If you have time you can run the simulation until the **endTime**. At the end of the simulation, compare the force coefficients of the steady with the force coefficients of the unsteady simulations.

Running the case

Let's run the same case using a finer mesh



Initial transient



www.wolfdynamics.com/wiki/hairpin_vortices/uns0/ani1.gif