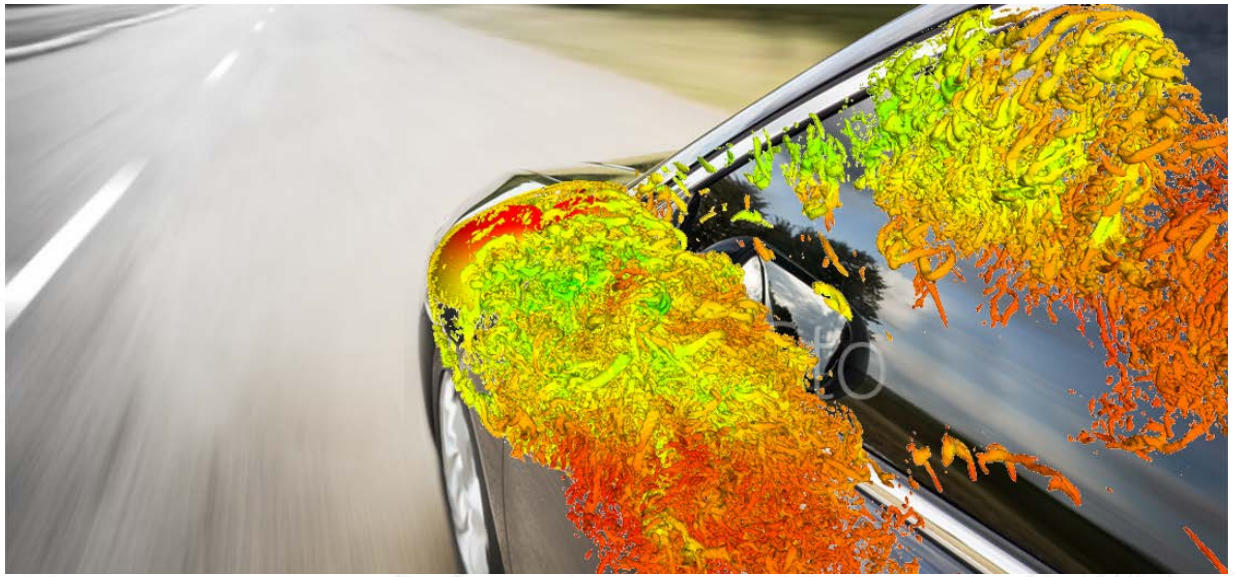
Latest meshing development in OpenFOAM

GOFUN 2019



Dr. Sebastien Vilfayeau
February 27th.

get it right®

www.esi-group.com

Overview of content

- Directional refinement + stretching 1806 1812
- Excluding mesh generation in geometric region 1812
- Generic leak detection (as a post processing tool) 1812
- Improvement in run time performance of castellation 1806
- Improvement in layer addition (single cell between walls) 1812
- checkCase in snappyHexMesh 1906

Directional refinement

Why?

- Cube of 1m side with 100 cells in each direction = 1M cells
- Same cube with 100 cells in one direction while rest with 50 = 0.25M => 1/4th size!

- **Aerodynamics/ aeroacoustics**
  - Wake of vehicle, mirror consume more cells
  - Can have influence of 2:1 refinement transition

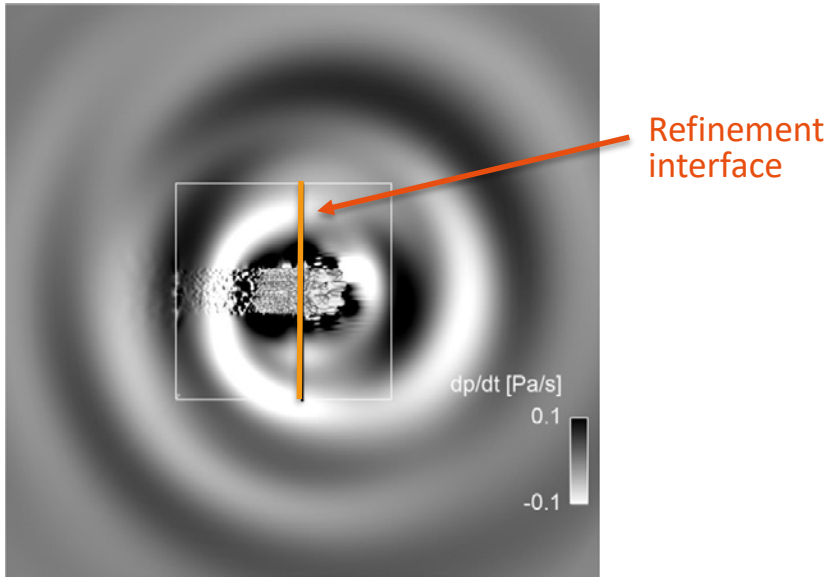
- **Advantages of directional refinement**
  - Reduction in cell count
  - Control of cell size in and across flow direction
  - Smooth cell size transition (uniform delta between cells, instead of 2:1)

- **Limitations**
  - the refinement is only in regions - there is no surface-based equivalent
  - the starting mesh has to be coordinate axis aligned (you can move it afterwards using e.g. transformPoints)
  - the resulting mesh will be incompatible with dynamic refinement/unrefinement.

Directional stretching

Why?

- **Aeroacoustics application:**
  - Refinement interface introduces spurious noise



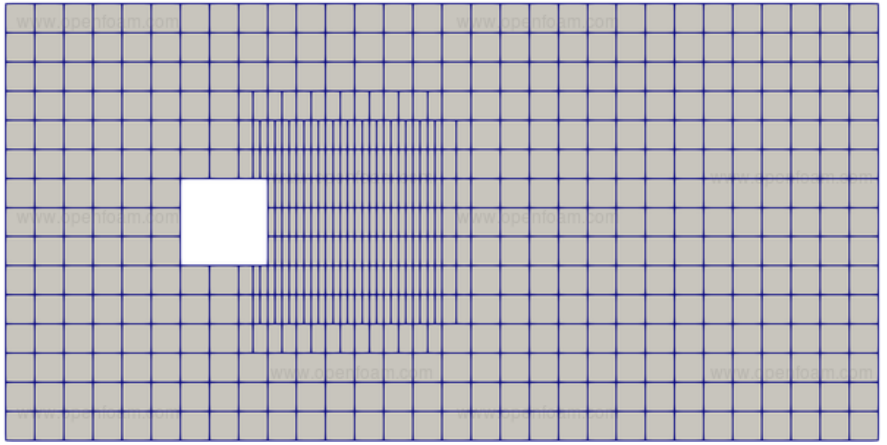
"A.H. Dawi, R.A.D. Akkermans, *Spurious noise in direct noise computation with a finite volume method for automotive applications, IJfHFF, vol. 72 (2018).*"

Directional refinement + stretching

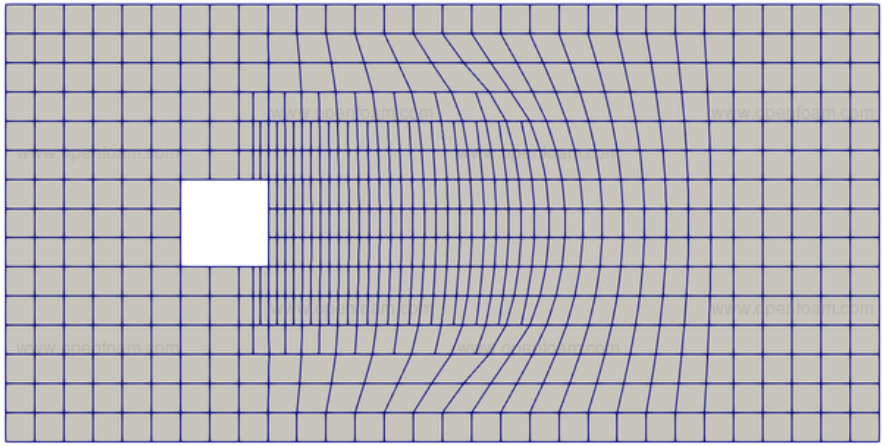
How?

```
refinementRegions
{
    wake
    {
        mode        inside;

        // Dummy uniform refinement
        levels      ((10000 0));

        // Directional refinement
        //  - for all cells with uniform level 0-1
        //  - do one split in y and z direction.
        levelIncrement  (0 1 (0 1 1));

        // Directional expansion-ratio smoothing
        smoothDirection     (1 0 0);
        // Smoothing of expansion ratio
        nSmoothExpansion    100;
        // Smoothing of positions
        nSmoothPosition     100;
    }
}
```

Directional refinement



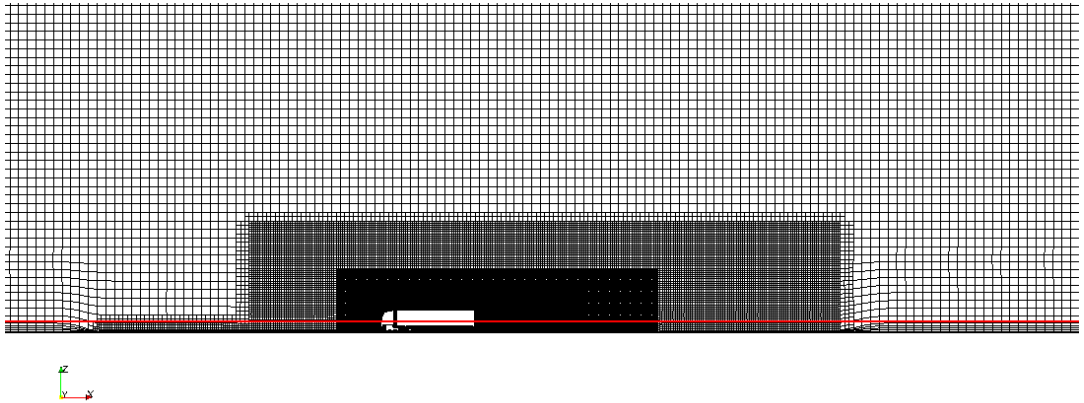
Directional refinement + stretching



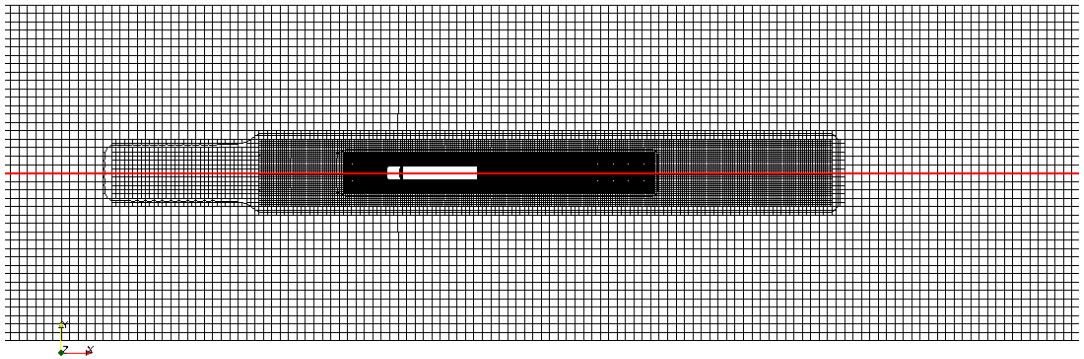
- Example:
  - $FOAM_TUTORIALS/mesh/snappyHexMesh/aerofoilNACA0012_directionalRefinement

Directional refinement + stretching

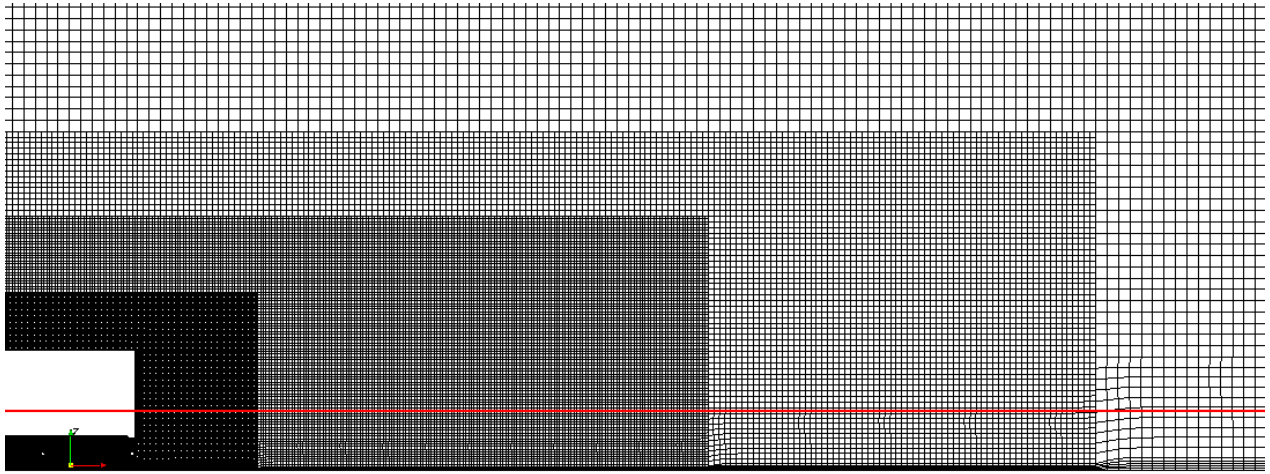
Example - Conventional approach



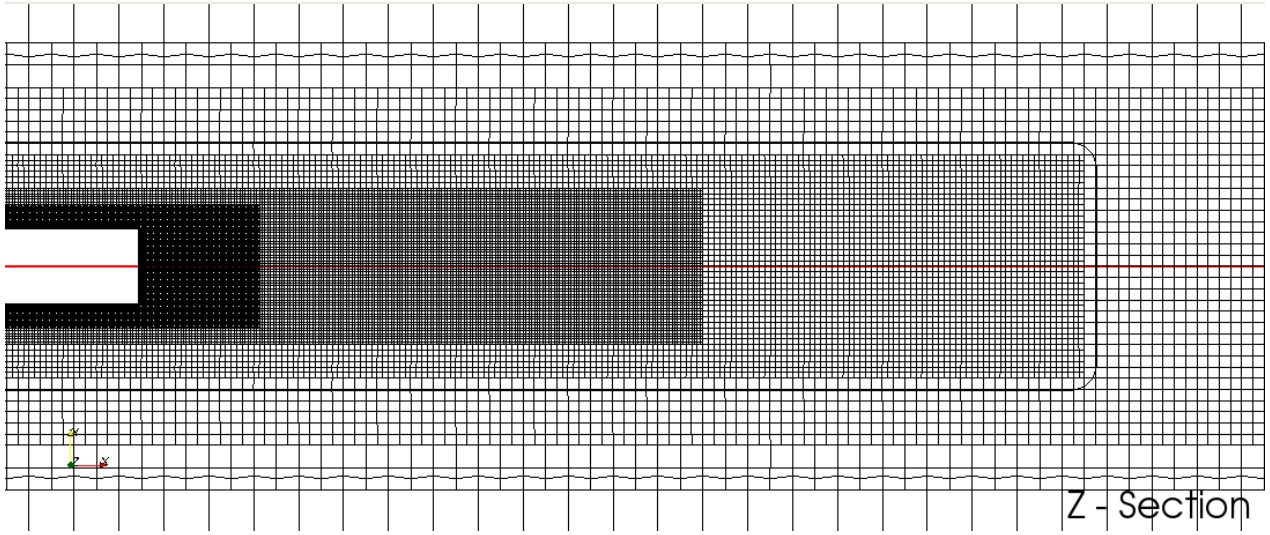
Y - Section



Z - Section



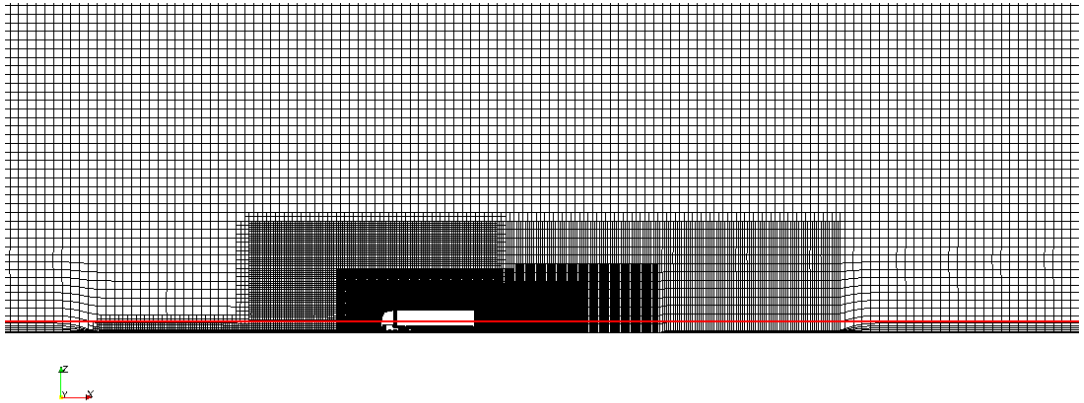
Y - Section



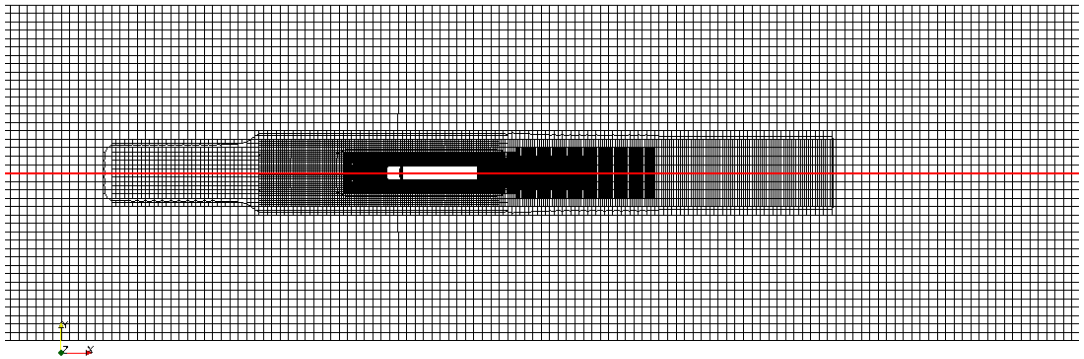
Z - Section

Directional refinement + stretching

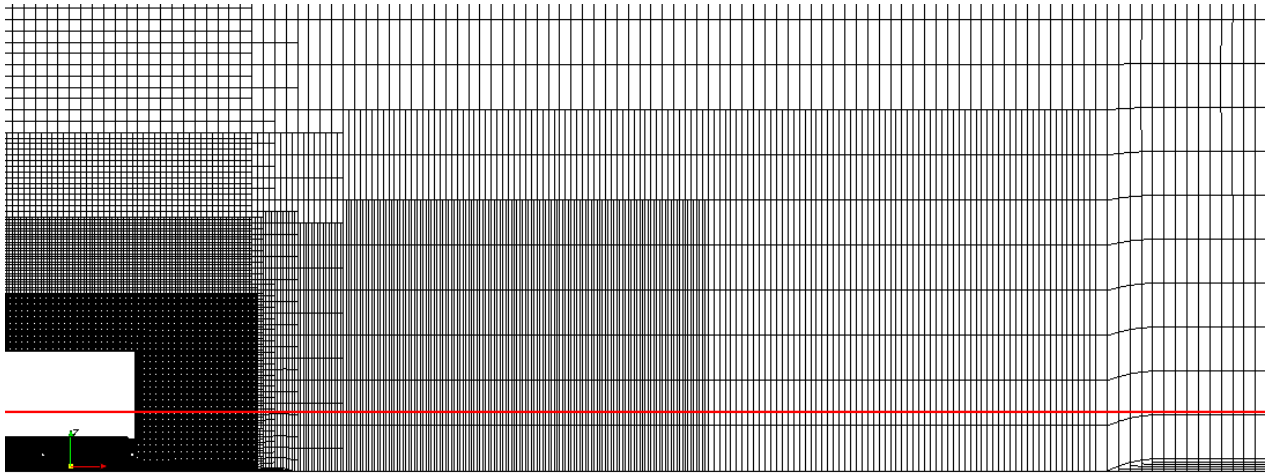
Directional refinement approach



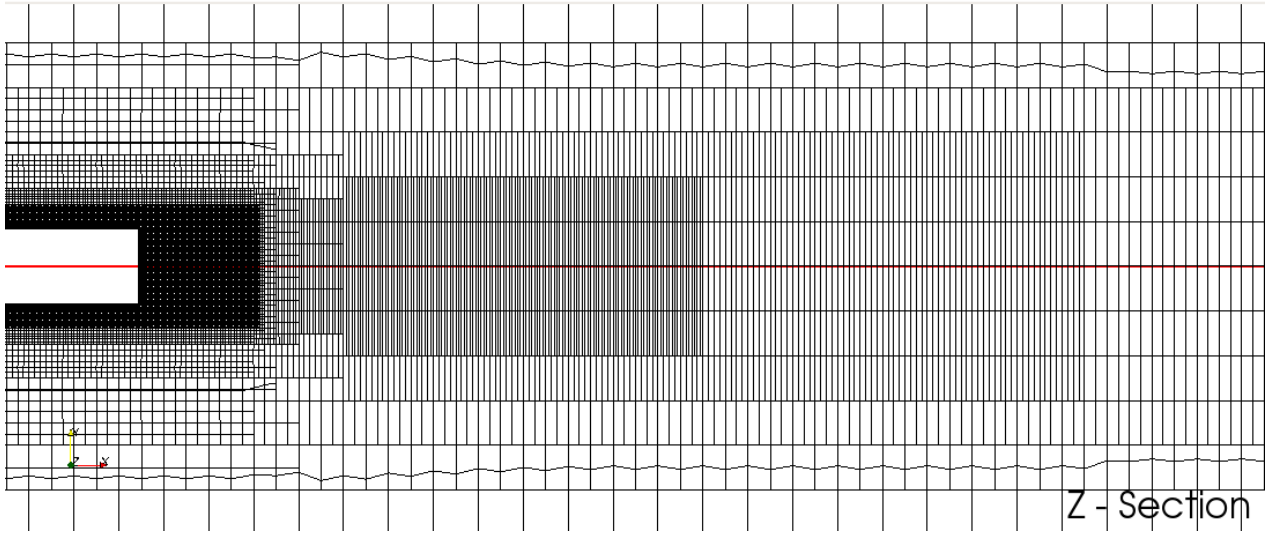
Y - Section



Z - Section



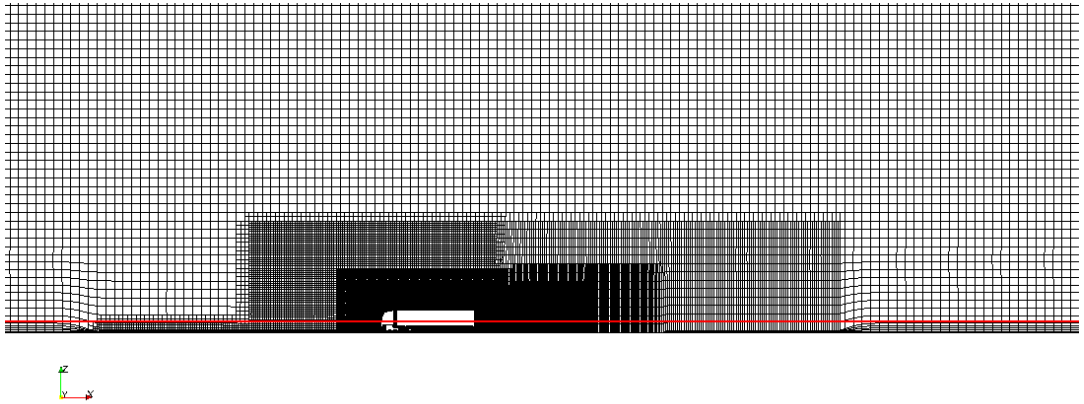
Y - Section



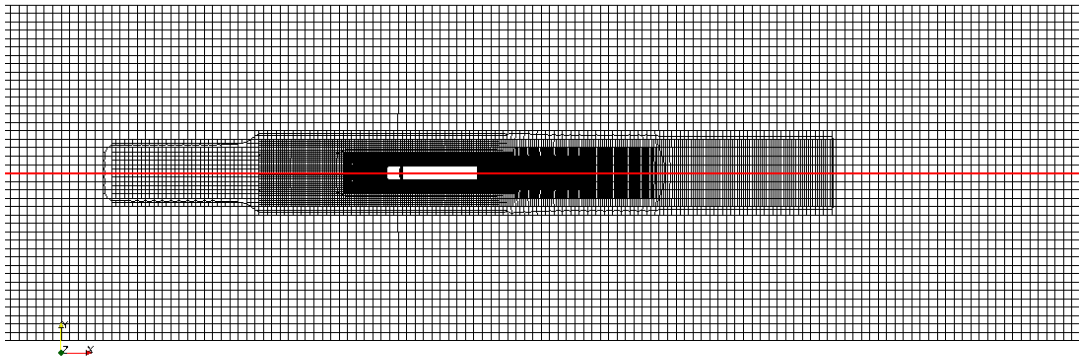
Z - Section

Directional refinement + stretching

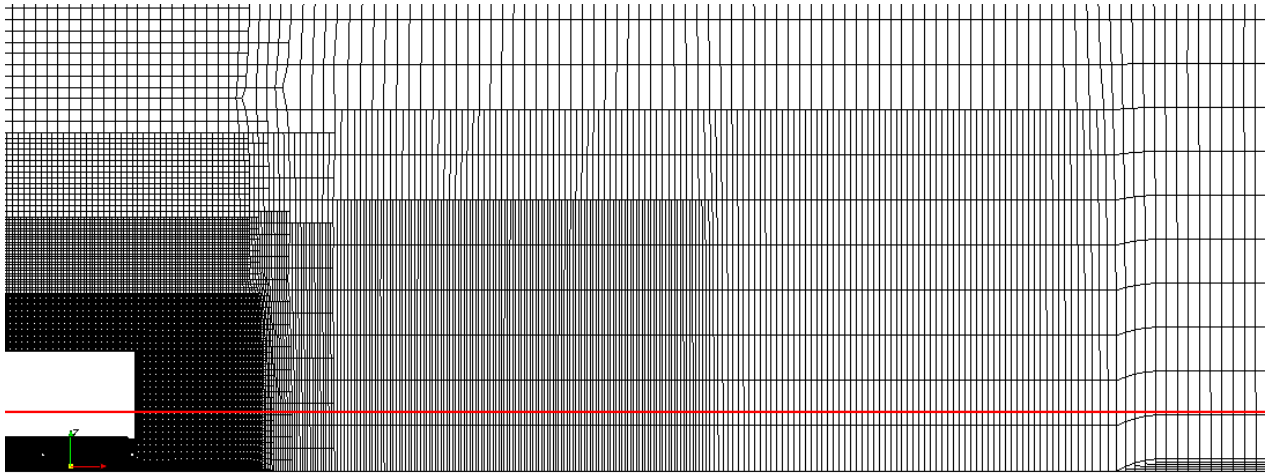
Refinement + stretching approach



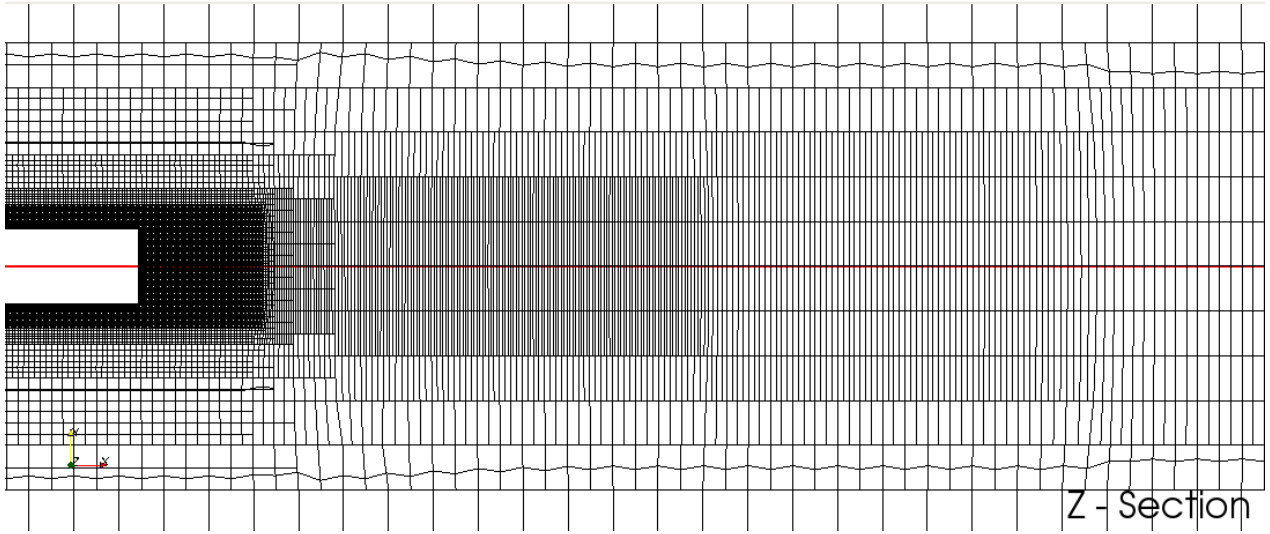
Y - Section



Z - Section



Y - Section



Z - Section

Directional refinement + stretching

Statistics

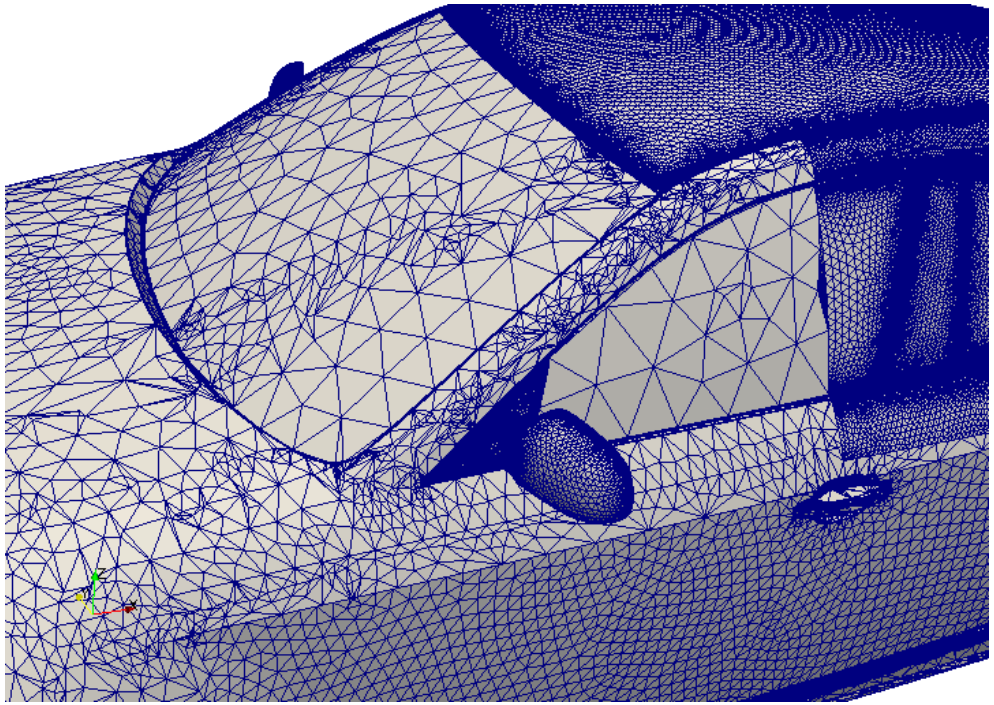
- Conventional approach : 25M cells
- Directional refinement : 24M cells

- Simulation details
  - Simulation time = 2000 iterations
  - Average for 500 iteration

	Conventional	Directional refinement	Directional refinement+ stretching
Delta drag value	0	-0.0034	-0.006
Castellation time (s)	407	391	483

Exclude mesh generation in a geometric region

Intersecting geometry/ Raw CAD

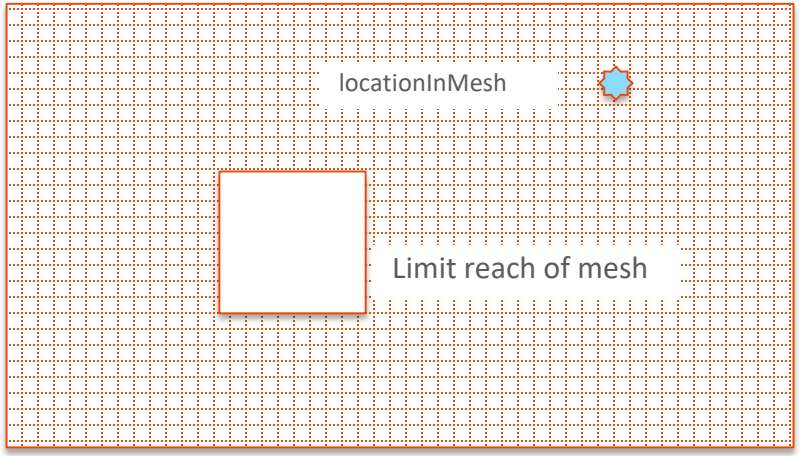


Known reason

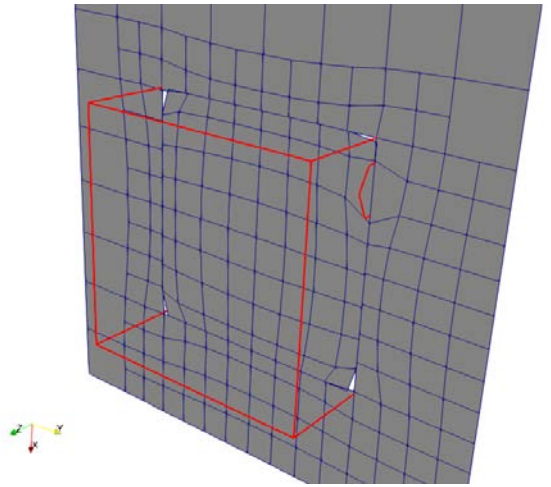
Exclude mesh generation in a geometric region

Concept

- Ways to handle mesh in unwanted region:
  - Leave the mesh inside, assuming less impact on solution
    - Overhead of cells!
  - Put a wrap geometry to avoid mesh penetration
    - Sounds good.
    - Lets explore!



- Test Scenarios using DrivAer$
  - Rubber seal of side window is removed.

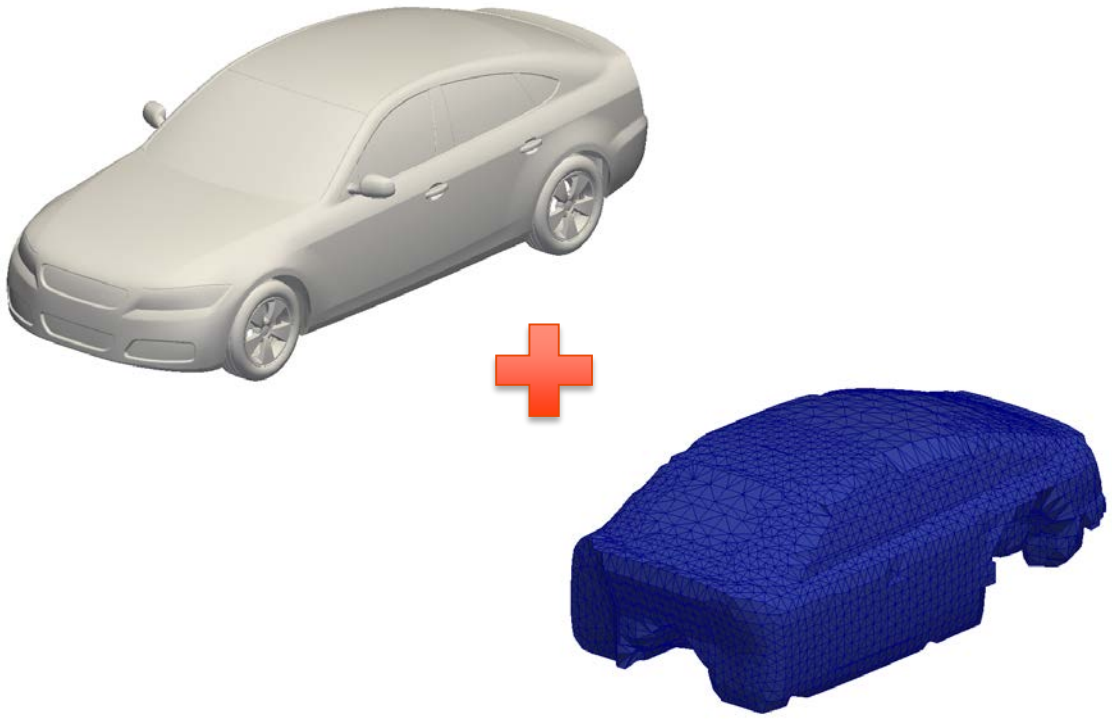


$ Fastback variant used from https://www.aer.mw.tum.de

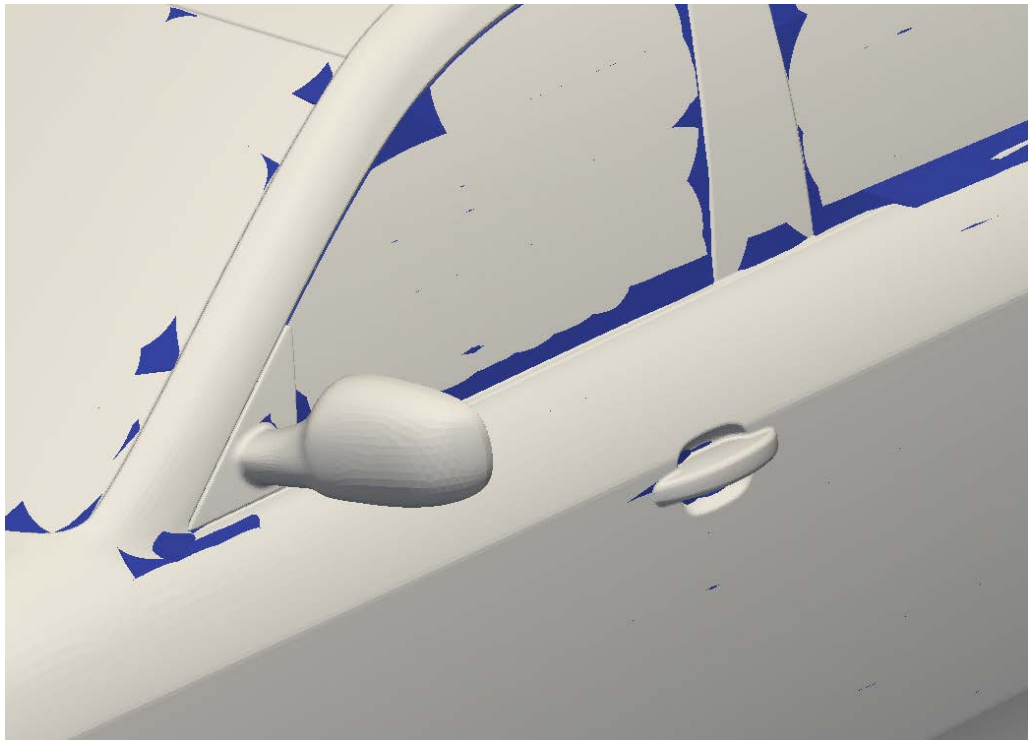
Exclude mesh generation in a geometric region

Using a wrap geometry – as a ‘wall’ surface

Geometry with wrap inside



Zoom view showing possible issues

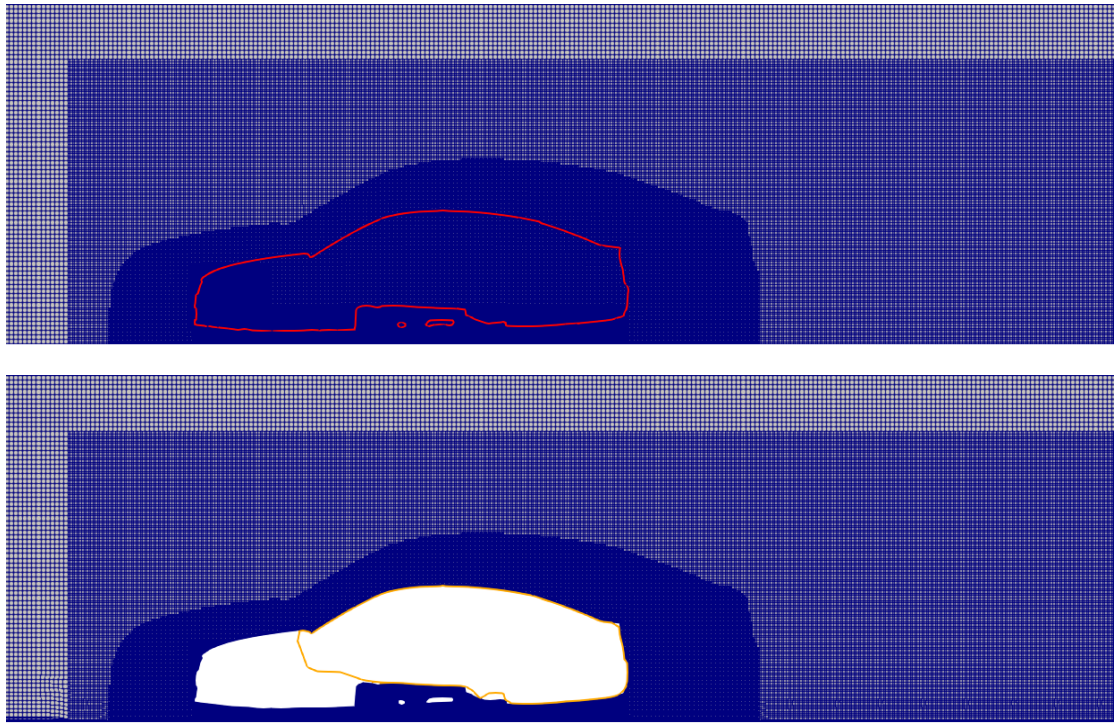


Zoom

Exclude mesh generation in a geometric region

Using a wrap geometry – as a ‘exclude’ region

Y cut section
Without and with wrap geometry



Usage syntax

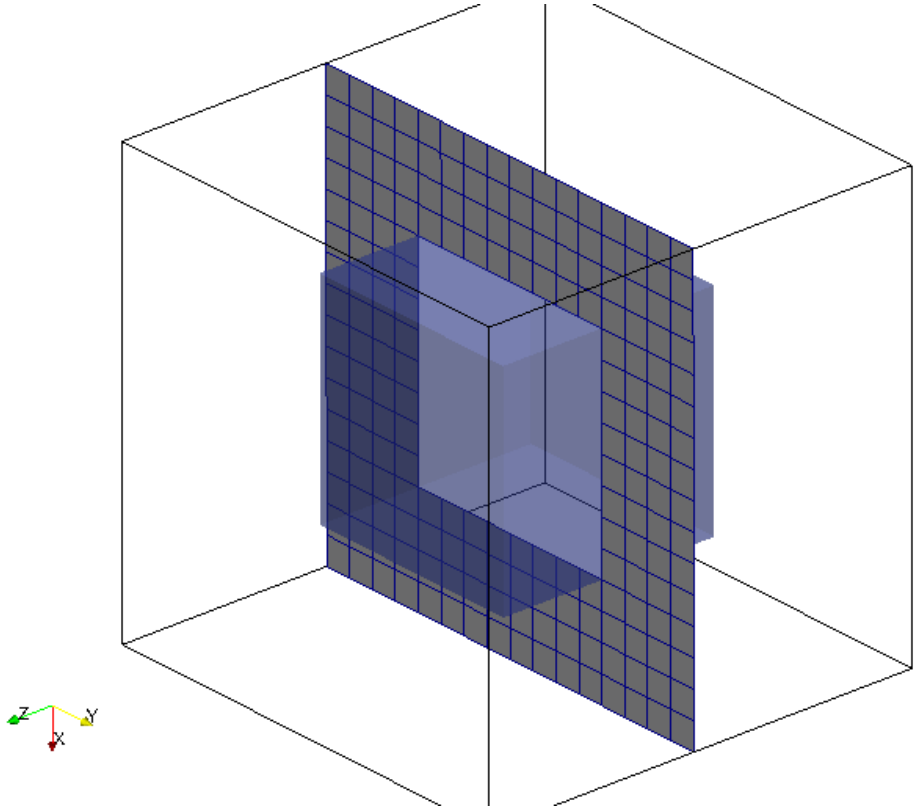
- We have feature limitRegions
  - For limiting refinement level
- Enhanced to use special input -1
  - Will remove all cells!
- Surface will not be used for patch assignment

```
limitRegions
{
    wrap
    {
        mode        inside;
        // Dummy base level
        levels      ((10000 -1));
    }
}
```

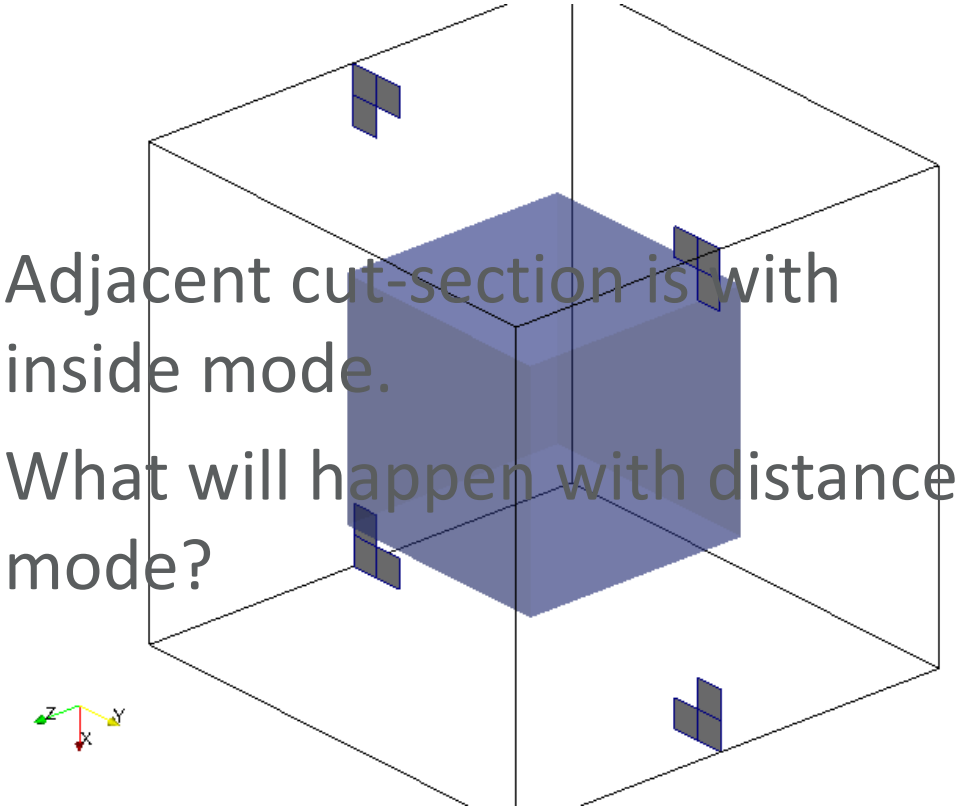

Zoom

Exclude mesh generation in a geometric region

Fun time!



Distance mode

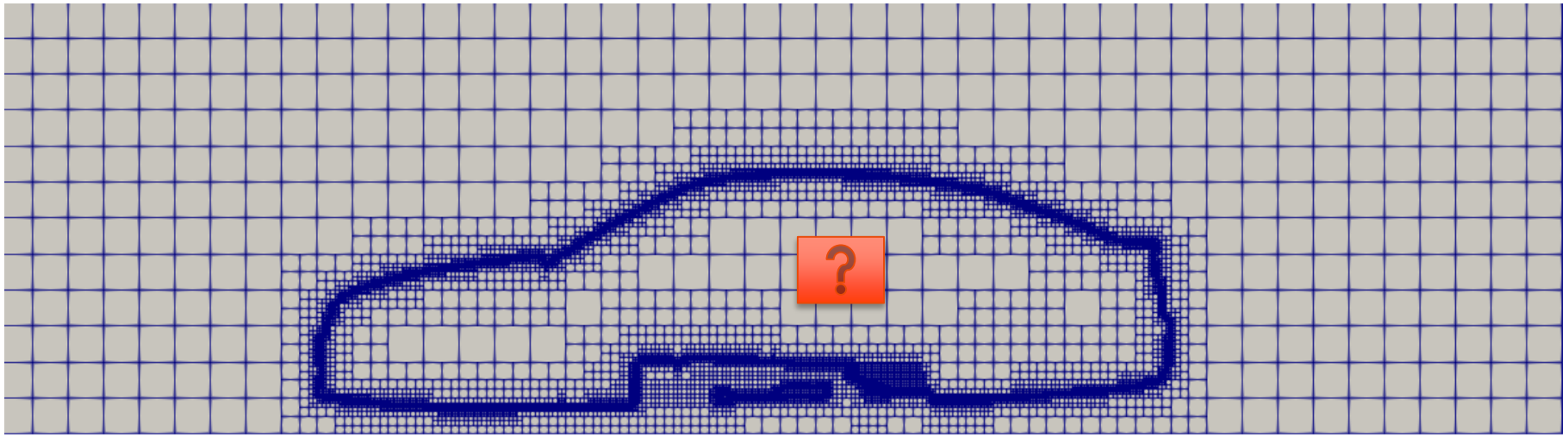


Adjacent cut-section is with inside mode.

What will happen with distance mode?

Leak detection

Where do I look into?



Unknown reason

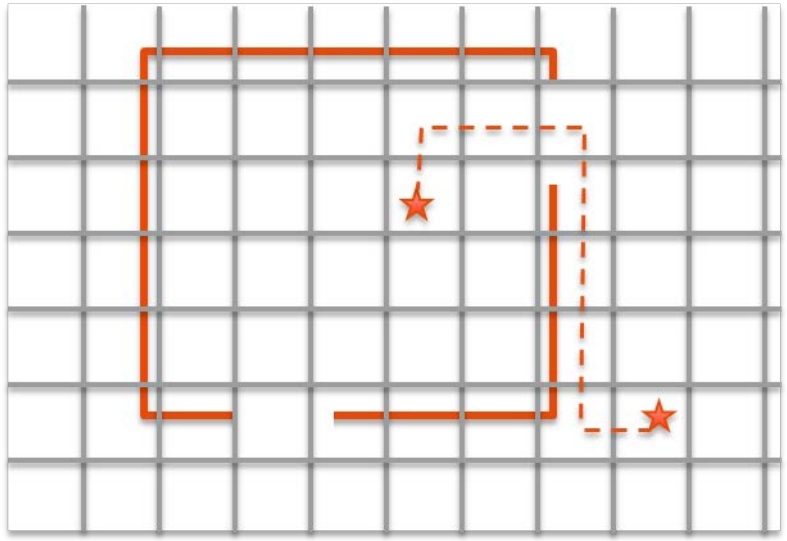
Leak detection

Via ‘postProcess’ tool

postProcessingDict

```
functions
{
    processorField
    {
        // Type of functionObject
        type            processorField;

        // Where to load it from (if not already in solver)
        libs            ("libfieldFunctionObjects.so");

        // Function object enabled flag
        enabled         true;

        // When to output the average fields
        writeControl    writeTime;
    }

    leakFind
    {
        type            sets;

        writeControl    timeStep;
        interpolationScheme cell;
        setFormat       vtk;

        sets
        (
            leakFind
            {
                type    shortestPath;
                insidePoints   ((3.0001 3.0001 0.43));
                outsidePoints  ((1 0 1.3));
                axis    xyz;
            }
        );

        // Needs at least one field
        fields          ( processorID );
    }
}
```

- Any mesh could be operated on
- At present the following formats are available for use.
  - csv
  - ensight
  - gnuplot
  - jplot
  - raw
  - vtk
  - Xmgr



It is also possible to operate the functionality in “postProcess” mode on any existing mesh (created by any mesher).
The syntax is shown in adjacent image
Both methods of operation would produce same leak points for same input mesh
One can set different output formats as per requirement

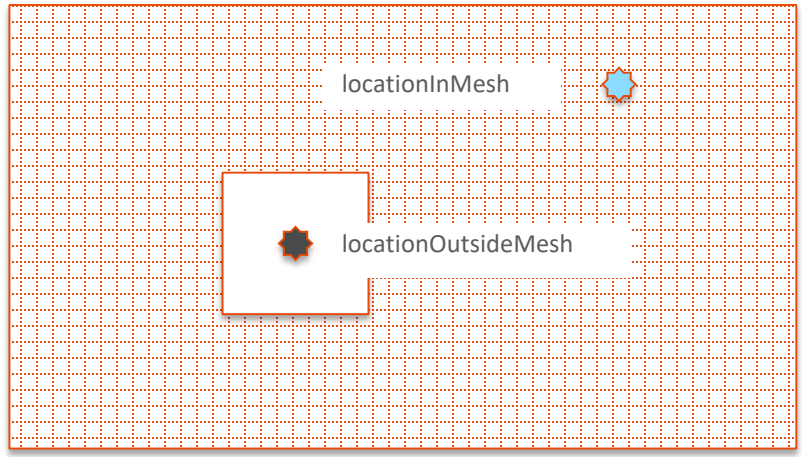
Leak detection

While generating mesh ?

- At present, user can provide locationsOutsideMesh
  - If mesh reaches this location, SHM exit with error.

```
// Optional locations that should not be reachable from
// location(s)InMesh
locationsOutsideMesh ((100 100 100));
```

- We have extended this functionality to dump the path before it exit



- Test Scenarios on DrivAer$
  - Wheel rim with example bolts removed
  - Rubber seal of side window is removed.

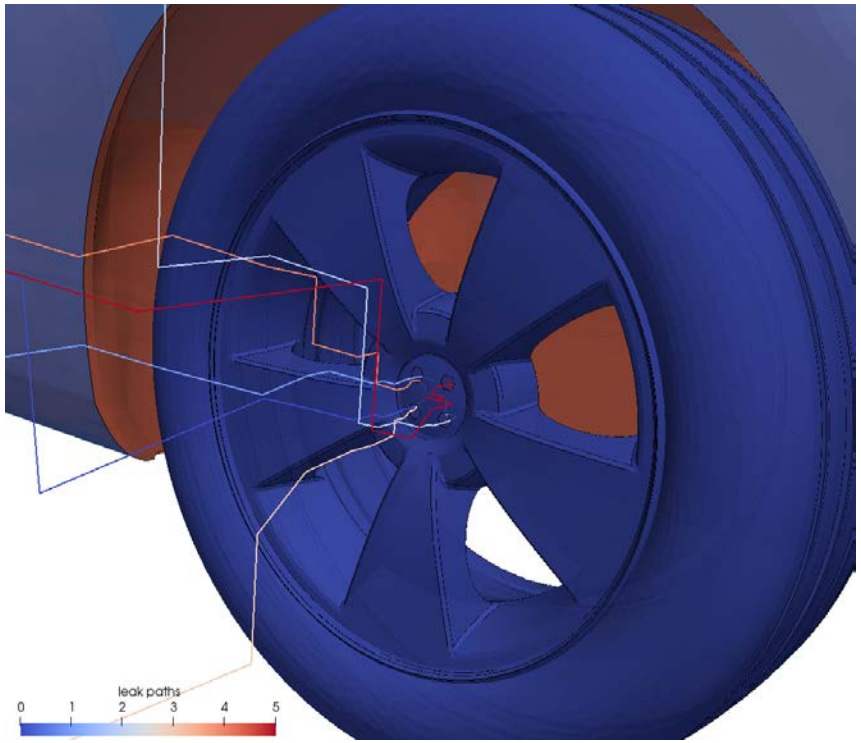
$ Fastback variant used from https://www.aer.mw.tum.de

Leak detection

Tests on DrivAer

Illustration of missing geometry
Nuts are removed

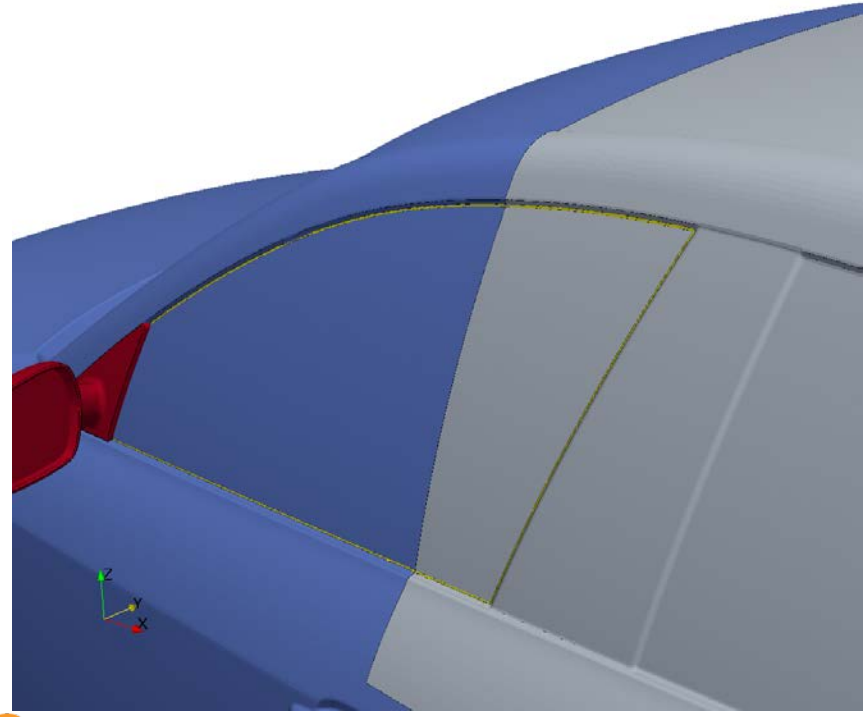
Leak path
pass through missing parts



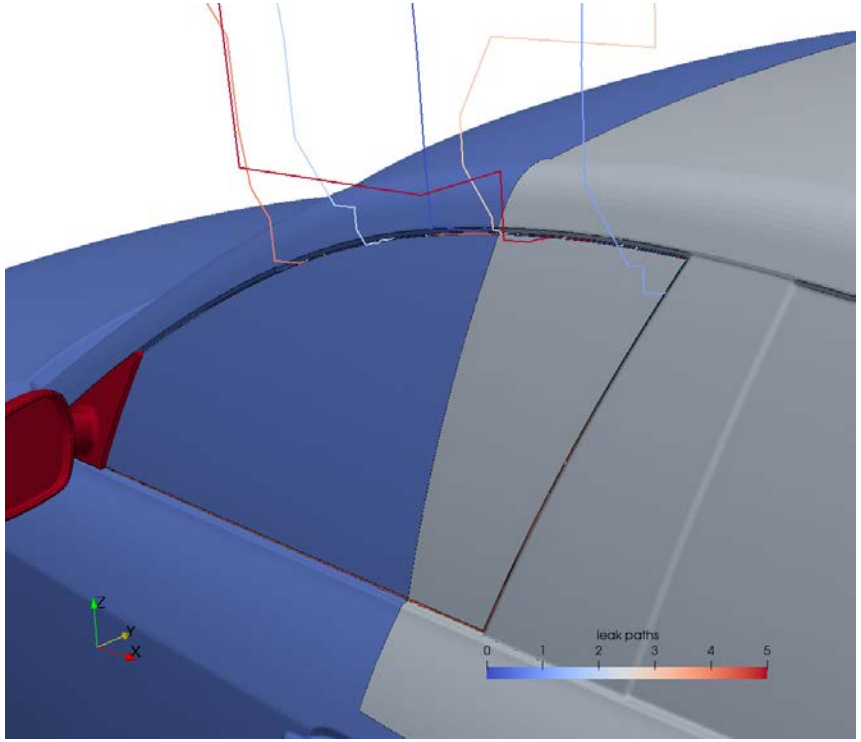
Leak detection

Tests on DrivAer

Illustration of thin gap
Rubber seal is removed

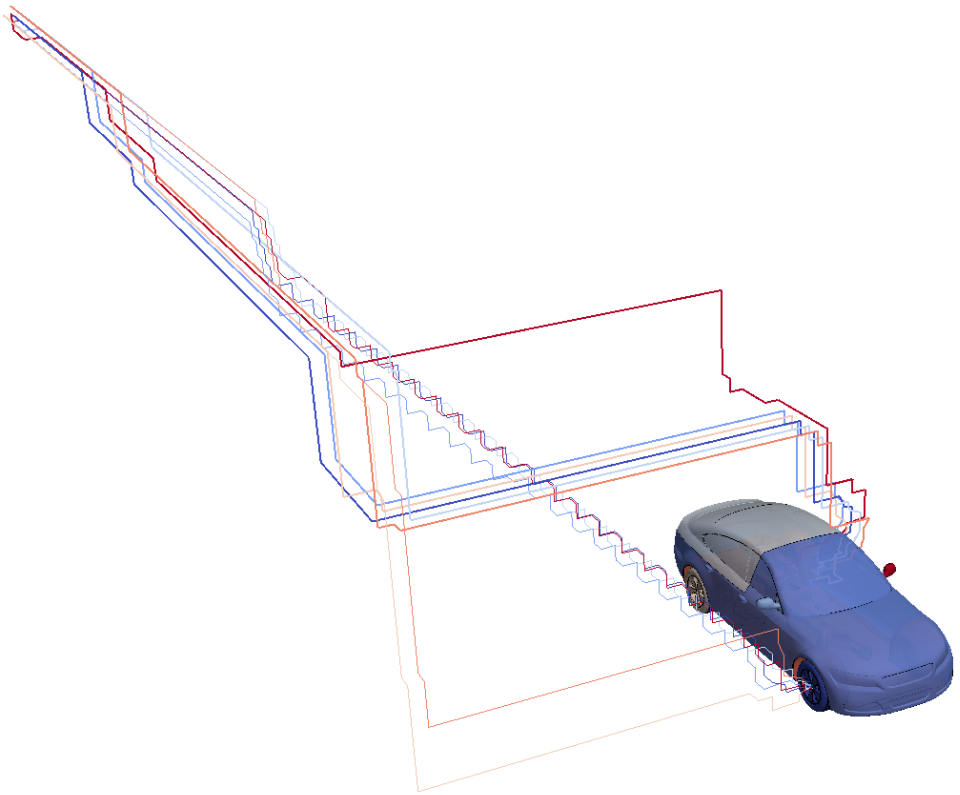


Leak path
pass through gap



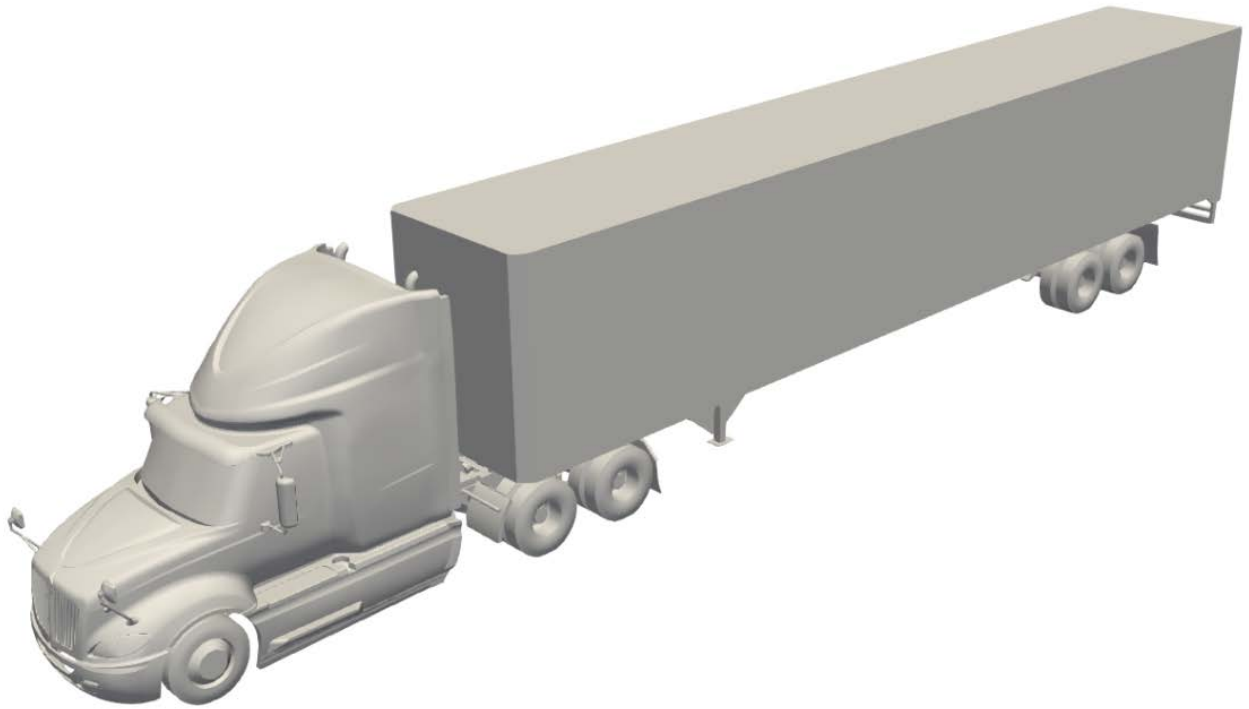
Leak detection

Tests on DrivAer – All leak paths

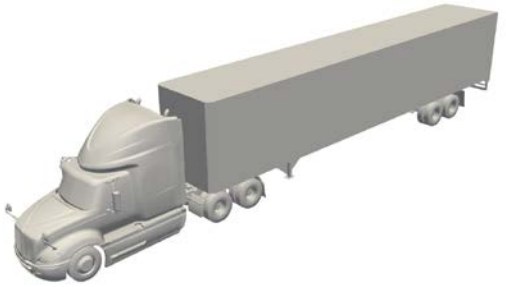


Improvement in castellation performance

regionSplit

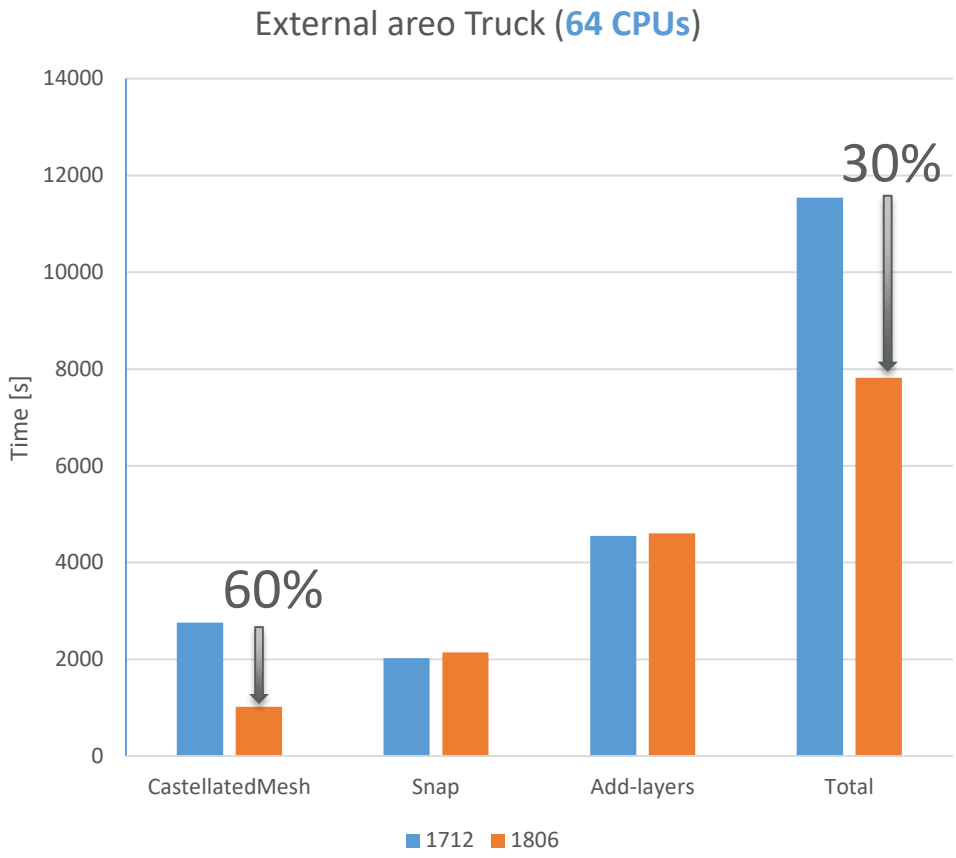


Improvement in castellation performance

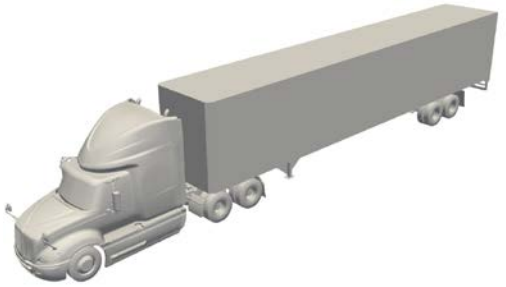


Statistics

- **Case:**
  - External aero for truck geometry
  - Cell count: 148M
- Significant (~60-70%) reduction in castellation stage
- No change in memory requirement
- No change in quality of mesh

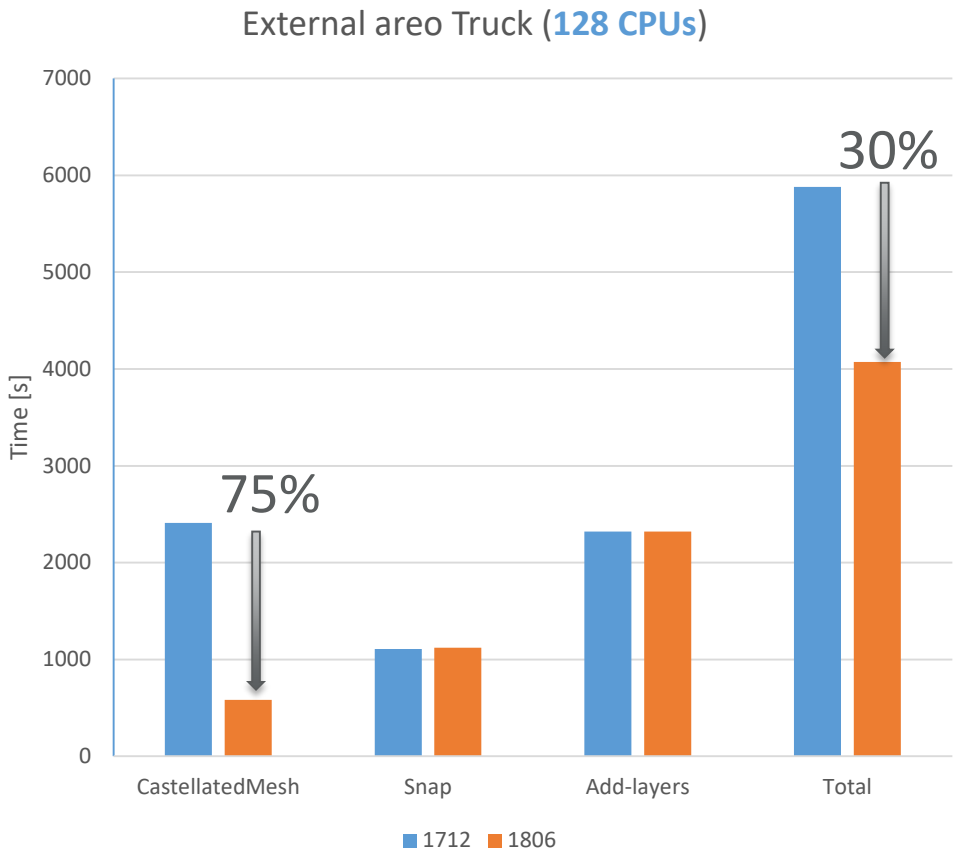


Improvement in castellation performance



Statistics

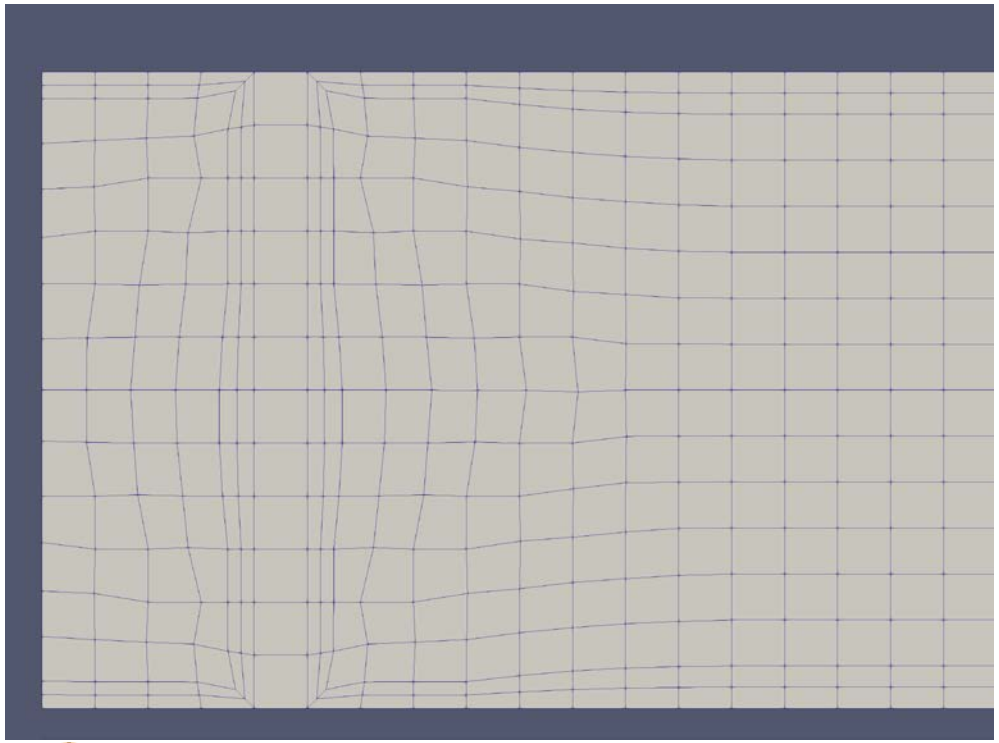
- **Case:**
  - External aero for truck geometry
  - Cell count: 148M
- Significant (~60-70%) reduction in castellation stage
- No change in memory requirement
- No change in quality of mesh



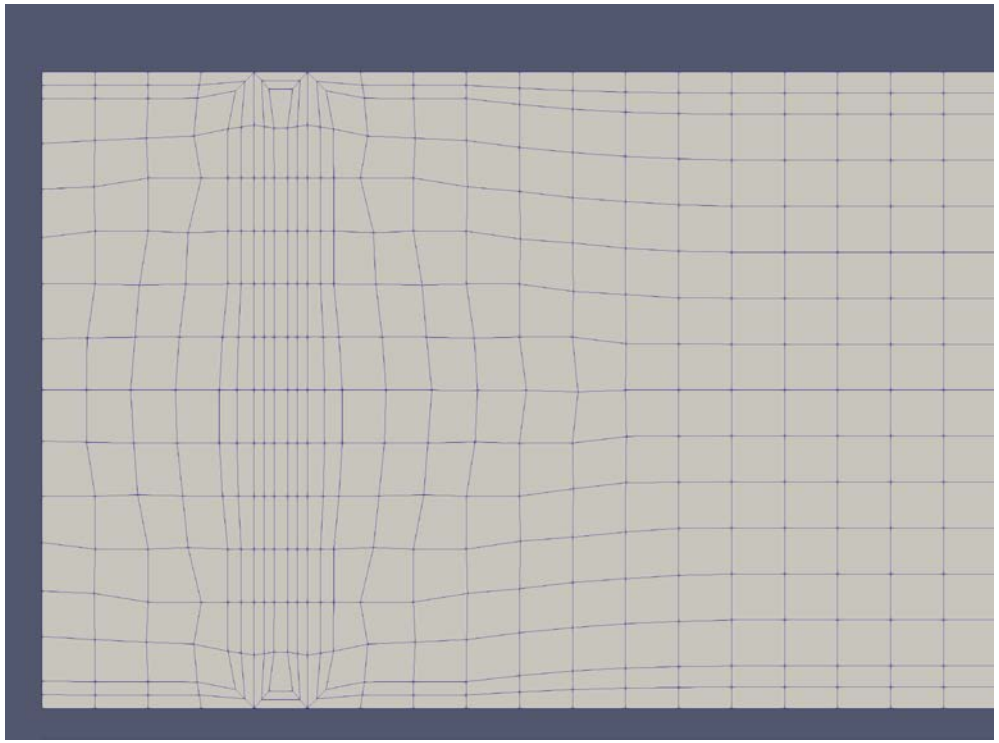
Improvement in layer addition

Medial axis method – single cell between opposing faces

v1806
(or develop with disableWallEdges false)



Develop



checkCase in snappyHexMesh

checkCase in snappyHexMesh

Why?

- Running any OpenFOAM application, including mesh generators, requires a complex set of input dictionaries to be correct.
  - Previously, a special command line option, '*-dry-run*’
- **Goal:** introduce similar option ‘*-dry-run*’ into snappyHexMesh

checkCase in snappyHexMesh

How?

- Check existence of all required sections in snappyHexMeshDict
  - Geometry
  - Castellation
  - Snapping
  - Layer addition
  - Mesh quality

Input Dict

```
// Mesh selection
// ~~~~~~~~~~~~~~~

// After refinement patches get added f
// all cells intersecting the surfaces
// section reachabl
// NOTE: This point
// after refinement
//locationInMesh (3
```

```
Initial mesh : cells:1280  faces:4224  points:1701
Cells per refinement level:
    0   1280
--> FOAM Warning :
    From function Foam::refinementParameters::refinement
    in file snappyHexMeshDriver/refinementParameters/ref
    Reading "/net/bruclu005/scratch-mstr/pss/projects/SU
    No 'locationInMesh' or 'locationsInMesh' provided
```

Resulting output

Input Dict

```
meshQualityControls
{
//    #include "meshQualityDict"


    // Advanced

    //- Number o
    nSmoothScale
    //- Amount t
    errorReducti
}
```

Resulting output

```
Missing/incorrect required dictionary entries:

Entry 'maxNonOrtho' not found in dictionary "/ne
Entry 'minVol' not found in dictionary "/net/bru
Entry 'minTetQuality' not found in dictionary "/
Entry 'maxConcave' not found in dictionary "/net
Entry 'minArea' not found in dictionary "/net/br
Entry 'maxInternalSkewness' not found in diction
Entry 'maxBoundarySkewness' not found in diction
Entry 'minFaceWeight' not found in dictionary "/
Entry 'minVolRatio' not found in dictionary "/ne
Entry 'minTwist' not found in dictionary "/net/b
Entry 'minTriangleTwist' not found in dictionary
Entry 'minDeterminant' not found in dictionary "
```

checkCase in snappyHexMesh

How?

- Check missing semi-colons

```
// Which of the steps to run
castellatedMesh true;
snap            true
addLayers       true;
```

Input Dict

```
Selecting decompositionMethod none [1]
--> FOAM Warning :
Missing/incorrect required dictionary entries:

keyword 'snap' has 2 excess tokens in stream

    3(true addLayers true)file: "/net/bruclu005/s
Entry 'addLayers' not found in dictionary "/net/b
```

Resulting output

- Check existence of input geometry/feature files

Example output

(using name in refinementSurface which is not defined in geometry section)

```
--> FOAM Warning :
    From function int main(int, char**)
    in file snappyHexMesh.C at line 1209
    Reading "/home/preston2/mattijs/OpenFOAM/work/Customer-
    The following geometry entries are not used:
    BANANA
```

Example output

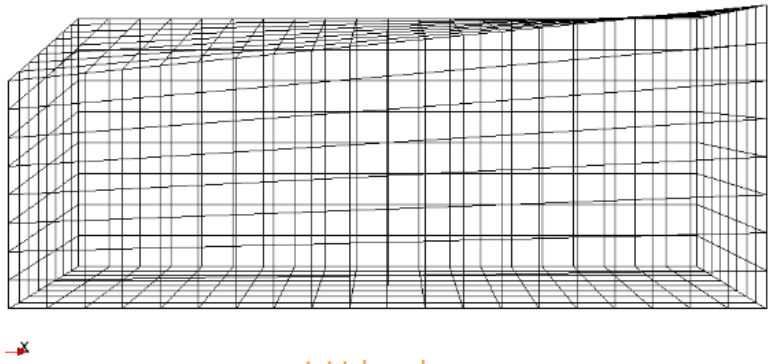
(using name in feature refinement which is not present in constant/triSurface)

```
Reading features.


--> FOAM FATAL IO ERROR:
Could not open "<path>/constant/triSurface/motorBike2.eMesh"

file: <path>/system/snappyHexMeshDict.castellatedMeshControls.features

    From function void Foam::refinementFeatures::read(const Foam::obje
    in file refinementFeatures/refinementFeatures.C at line 111.

FOAM exiting
```

checkCase in snappyHexMesh

How?

- Calculate bounding box of starting mesh. Check the alignment of all edges with one of the coordinate axes and issue a warning:



Initial mesh

```
Create mesh for time = 0

Read mesh in = 0.01 s
Initial mesh has 9696 edges unaligned with any of the coordinate axes
```

Resulting output

- Check whether primitive parts are within the bounding box of the starting mesh. This will highlight if there has been a mistake in the X-Y-Z location/set-up of the primitive parts

```
Checking for geometry size relative to mesh.
--> FOAM Warning :     GRWS0930_frontWheels_____________007.stl bounds differ from mesh by more than a factor 100:
        bounding box      : (-0.1245990023 -0.3493080139 -0.1274009943) (0.130200997 0.3493080139 0.1273989975)
        mesh bounding box : (-33.90895 -32.2595 -1.6284) (41.09105 32.2405 32.8716)
```

checkCase in snappyHexMesh

How?

- Check existence of location/s in mesh, and issue a message whether
- inside/outside the initial mesh, initial bounding box of the geometries.
  - This would highlight if the location selected is wrong (because of the location or units), and whether it needs to be re-verified.
  - No warning will be issued, as both outside/inside meshes could be valid, as long as the location is inside the initial blockMesh.

```
// Mesh selection
// ~~~~~~~~~~~~~~

// After refinement patches get added f
// all cells intersecting the surfaces
// section reachable from the locationI
// NOTE: This point should never be on
// after refinement.
locationInMesh (23.0001 3.0001 0.43);
```

Input Dict

```
--> FOAM FATAL ERROR:
Point (23.0001 3.0001 0.43) is not inside the mesh or on a face or edge
Bounding box of the mesh:(-5 -4 0) (15 4 8)

    From function static Foam::labelList Foam::refinementParameters::fi
    in file snappyHexMeshDriver/refinementParameters/refinementParamete

FOAM exiting
```

Resulting output

checkCase in snappyHexMesh

How?

- Output minimum expected cell size, including potential automated refinements
- This is calculated from the maximum refinement level (from feature, surface refinement, volume refinement level) and the average edge length of the starting mesh: cellSize = edgeLength*2^-refinementLevel

```
Cell size estimate :
    Level  0 : 2.875
    Level  9 : 0.005615234375

Voxellating initial mesh : (208 176 96)
```

checkCase in snappyHexMesh

How?

- Output a guess for the expected number of cells after meshing
- The voxel mesh takes into account
  - surface refinement level
  - location(s)InMesh
  - volume refinement level

- but misses out on
  - feature refinement level
  - 2:1 refinement constraints
  - layers

```
Voxel refinement :
    Initial                      : (3514368)
    After removing outside cells : 1(0)
    After surface refinement     : 10(132268 3888 640 0 620 350 60 160 214 30)
    After keeping inside voxels  : 10(3219998 3888 640 0 620 350 60 160 214 30)
    After shell refinement       : 10(2373120 620060 177160 57680 51378 1143 543 242 224 30)
Estimated cell count : 17447071
```

What next?

Many more to come…

OpenCFD's commitment to OpenFOAM Users

Global Conferences and independent User Group meetings

- **OpenFOAM Release Features Webinar**
  - Online, 28 Feb 2019
  - https://www.esi-group.com/company/events/2019/openfoam-release-features-webinar

- **OpenFOAM 7th User Conference**
  - Berlin, Germany, 15th-17th Oct. 2019
  - Call for papers, deadline: 31th May

- **AIAA AVIATION Forum 2019**
  - OpenFOAM® Aerospace Course
  - Dalles, Texas, USA, June 15-16 2019

- OpenFOAM North American conference, TBD, 2019

- OpenFOAM India, TBD, 2019

Apply OpenFOAM

https://www.openfoam.com/training/

- **Aeroacoustics Course**
  - Basics - aeroacoustics source and propagation mechanisms, including a shallow dive into classical theory
  - Noise Propagation to the far-field using acoustics analogies, e.g. Curle
  - etc

- **Overset Meshing and Applications Course**
  - Learn how to use overset meshing in OpenFOAM
  - Best practice meshing, set-up and control settings, tutorials
  - Industrial examples in external aerodynamics, high speed passing trains, ship maneuvering, fans

- **External Aerodynamics Course**
  - Learn how to use OpenFOAM for external aerodynamics
  - Turbulence modelling from RANS to DDES
  - Best practices to setup the case and post-processing

THANK YOU

sebastien.vilfayeau@esi-group.com