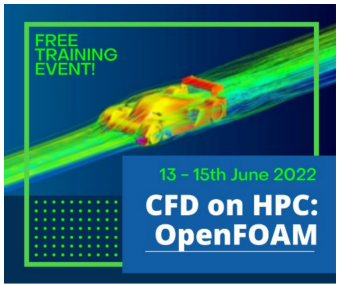
Advanced use of OpenFOAM



EuroCC workshop

Aleksander GRM

June 14, 2022

Vsebina

Cases review

Content

- BlockMesh introduction
- GMSH introduction
- Advanced foil case

OpenFOAM mesh

By default OpenFOAM defines a mesh of **arbitrary polyhedral cells** in ***3D***, bounded by **arbitrary polygonal faces**, i.e. the cells can have an unlimited number of faces where, for each face, there is no limit on the number of edges nor any restriction on its alignment. A mesh with this general structure is known in OpenFOAM as a **polyMesh**. This type of mesh offers great freedom in mesh generation and manipulation in particular when the geometry of the domain is complex or changes over time.

Basic mesh elements

- Point
- Face
- Cell
- Boundary

(Detailed mesh description @ cfd.direkt)

Points

A point is a location in 3-D space, defined by a vector in units of metres (m). The points are compiled into a list and each point is referred to by a label, which represents its position in the list, starting from zero.

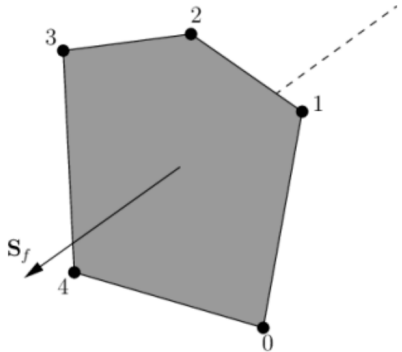
The point list cannot contain two different points at an exactly identical position nor any point that is not part at least one face.

Points – polyMesh/points


```
FoamFile
{
  format      ascii;
  class       vectorField;
  object      points;
}
// * * * * * * * * * * * * * * * * * * * *  //


21812
(
(-17.5492 0.306481 0)
(-17.5472 0.397851 0)
(-17.5391 0.49775 0)
(-17.5189 0.60624 0)
(-17.4861 0.723342 0)
...
)
```

- face is an ordered list of points
- each two neighbouring points are connected by an edge
- direction of the face normal vector is defined by the right-hand rule
- there are two types of face
  - Internal faces
  - Boundary faces



Faces – polyMesh/faces

```
FoamFile
{
  format      ascii;
  class       faceList;
  object      faces;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

43066
(
4(156 0 78 235)
4(157 236 79 1)
4(156 235 236 157)
4(158 237 80 2)
...
)
```

Cells

A cell is a list of faces in arbitrary order. Cells must have the properties listed below.

- The cells must be **contiguous**, i.e. completely cover the computational domain and must not overlap one another.
- Every cell must be **closed geometrically**, such that when all face area vectors are oriented to point outwards of the cell, their sum should equal the zero vector to machine accuracy;
- Every cell must be **closed topologically** such that all the edges in a cell are used by exactly two faces of the cell in question.

Cells – polyMesh/cells

```
FoamFile
{
    format      ascii;
    class       cellList;
    object      cells;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * //

10720
(
6(0 1 21626 21627 2 21548)
6(1 3 21628 21629 4 21549)
6(3 5 21630 21631 6 21550)
...
)
```

Structure of `polyMesh` files

- **points**: a list of vectors describing the cell vertices, where the first vector in the list represents vertex 0, the second vector represents vertex 1, etc. ;
- **faces**: a list of faces, each face being a list of indices to vertices in the points list, where again, the first entry in the list represents face 0, etc. ;
- **owner**: a list of owner cell labels, the index of entry relating directly to the index of the face, so that the first entry in the list is the owner label for face 0, the second entry is the owner label for face 1, etc;
- **owner**: a list of owner cell labels, the index of entry relating directly to the index of the face, so that the first entry in the list is the owner label for face 0, the second entry is the owner label for face 1, etc;
- **owner**: a list of owner cell labels, the index of entry relating directly to the index of the face, so that the first entry in the list is the owner label for face 0, the second entry is the owner label for face 1, etc;
- **neighbour**: a list of neighbour cell labels;
- **boundary**: a list of patches, containing a dictionary entry for each patch, declared using the patch name

We will follow the text in cfd.direkt web page, explaining:

- geometry boundaries
- different types of standard boundary conditions

OpenFOAM v9 User Guide: 5.2 Boundaries

BlockMesh introduction

The **blockMesh** mesh generator is generator of **structured mesh**. In most of the cases this is most desiderated type of mesh.

- The blockMesh utility creates parametric meshes with grading and curved edges.
- The mesh is generated from a dictionary file named blockMeshDict located in the system (or constant/polyMesh) directory of a case. blockMesh reads this dictionary, generates the mesh and writes out the mesh data to points and faces, cells and boundary files in the same directory.
- The principle behind blockMesh is to decompose the domain geometry into a set of 1 or more three dimensional, hexahedral blocks. Edges of the blocks can be straight lines, arcs or splines. The mesh is ostensibly specified as a number of cells in each direction of the block, sufficient information for blockMesh to generate the mesh data.

We shall continue **BlockMesh** learning over the web UserGuide @ OpenFoam.org

v9-blockmesh @ cfd.direct

- simple cavity (tutorials link)
- cavity with hole (tutorials link)
- pipe,
- sphere.

GMSH introduction

Set up Python environment with GMSH

To be able to run advanced GMSH examples we need to set up Python environment

```
1. load module python:
  $> ml av python (check target version)
  $> ml python-version

2. Create new env:
  $> python3 -m venv local

3. Activate new env:
  $> source local/bin/activate

4. Install new packages (active env local):
  $(local)> pip install numpy scipy sympy matplotlib
  $(local)> pip install --upgrade gmsh
```

Load **only** Python environment

To use Python environment we need only to load it

```
1. load module python:
  $> ml av python (check target version)
  $> ml python-version

2. Activate new env:
  $> source local/bin/activate
```

We shall continue **GMSH** description/learning over the web UserGuide @ gmsh.info

User Guide – v4.10.3 @ gmsh.info

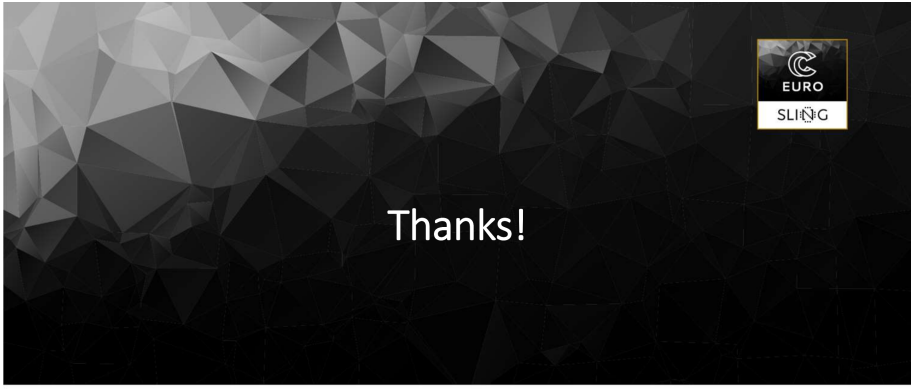
Show the **cavity_with_hole** example.

Show the construction using Python of GMSH **foil** example.

Foil case

Show advanced **foil** case

- describe case philosophy,
- show mesh generation for different angles of attack (compare with fixed mesh),
- show the initialization with potentialFoam,
- compare solutions of the case for angle of attack 20° (stable and unstable mesh).



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 951732. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, United Kingdom, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Switzerland, Turkey, Republic of North Macedonia, Iceland, Montenegro

EuroHPC Joint Undertaking