Introductory meshing course with OpenFOAM

Sánchez-Cruz Ángel

June 2024

Contents

Contents

Chapter 1

What is OpenFOAM and why learn to use it?

OpenFOAM (Open Field Operation and Manipulation) is an open-source software package used for computational fluid dynamics (CFD) and other types of simulations. It provides a wide range of solvers for simulating complex physical phenomena, such as fluid flow, heat transfer, chemical reactions, solid mechanics, electromagnetism, particle dynamics, multiphase flow, etc.

OpenFOAM is highly customizable and allows users to modify and extend its functionality through its extensive C++ library. Users can create new solvers, utilities, and libraries tailored to specific simulation needs by leveraging OpenFOAM's object-oriented design. The source code is openly available, enabling modifications to existing solvers and the development of custom applications. This flexibility makes OpenFOAM suitable for a wide range of applications and research areas, allowing for advanced and specialized simulations.

1.1 Download alternatives

https://openfoam.org/

Figure 1.1: OpenFOAM.org logo

Is the primary source for free and open-source downloads of OpenFOAM. It provides access to standard versions of OpenFOAM and official documentation.

https://www.openfoam.com

Figure 1.2: OpenFOAM.com logo

Offers downloads of OpenFOAM and also provides additional commercial services related to support and implementation of OpenFOAM. It's ideal if you're looking for professional support options and training.

https://www.cfdsupport.com/openfoam-for-windows.html

Figure 1.3: CFD Support logo

This site offers a version of OpenFOAM specifically designed for Windows users. You can download OpenFOAM here and explore its functionality within the Windows environment.

Chapter 2

Installation of OpenFOAM in Windows

This chapter guides readers step-by-step through the process of installing OpenFOAM, using the CFD support page as the primary resource. It covers everything from system prerequisites to downloading the software and initial setup. Readers will learn how to configure the necessary development environment and perform platform-specific configurations. Additionally, useful tips and solutions to potential installation issues are provided, ensuring that users can effectively and efficiently start using OpenFOAM.

2.1 Basic System Requirements:

To efficiently install and run OpenFOAM, it is important to have a computer that meets certain basic requirements. Here are the recommended requirements, though they do not necessarily need to be met:

- Operating System:
  - Linux: OpenFOAM is primarily developed for Linux. The most commonly used distributions are Ubuntu and CentOS.

  - **Windows**/Mac: OpenFOAM can run on Windows and Mac through Docker containers or virtual machines, although using Linux is recommended for optimal performance.

- Processor (CPU):
  - A multi-core processor, preferably with 4 cores or more. OpenFOAM can take advantage of multi-core architecture to parallelize computations and improve performance.

- Memory (RAM):
  - Minimum: 8 GB of RAM.
  - Recommended: 16 GB or more, especially for larger and more complex problems.

- Storage:
  - Disk Space: At least 20 GB of free space for software installation and simulation files. More space may be required depending on the size of the cases being run.
  - An SSD is preferable to improve read/write speeds, which can accelerate simulations and handling of large files.

- Graphics Card (optional):
  - Not strictly necessary, but a dedicated GPU can be useful if you plan to perform complex result visualizations or use hardware acceleration for certain operations.

Notes

OpenFOAM is an open-source software widely used in the computational fluid dynamics community, offers different download pages for various reasons, including stable and development versions, different distributions maintained by various organizations, and customized versions tailored to specific needs.

Keys and commands to use

- CTRL key + up scroll Increases the font size in the console.
- CTRL key + down scroll Decreases the font size in the console.
- TAB key Auto-completes the information, ensuring that we are working in the correct directory or using the correct command.
- ENTER key Used to send commands and confirm actions.
- cd Change directory, allows navigation forward in the directory structure by specifying the name of a directory you want to move into.
- cd .. Navigates backward in the directory structure, moving you to the parent directory of the current directory.
- clear Clear all previous lines of text in the terminal.
- ls List the contents of a directory.
- paraFoam.exe Opens ParaView software and load the current project.
- blockMesh.exe Creates the numerical mesh that defines the geometry of the simulation domain.
- surfaceFeatures.exe Extract geometric features from a surface, such as contour lines or sharp edges, which are important for defining the geometry and mesh quality.
- snappyHexMesh.exe This command runs the meshing process.

2.2 From CFD Support

CFD SUPPORT is a private engineering firm founded in 2009 by two engineers and PhD graduates from the Von Karman Institute for Fluid Dynamics in Prague, Czech Republic. Specializing in engineering simulations, the company operates at the intersection of physics, mathematics, and software engineering. Their team comprises skilled engineers, mathematicians, and programmers dedicated to meeting the diverse demands of computational fluid dynamics (CFD) through collaborative expertise and flexible solutions [1].

Figure 2.1: CFD Support logo

1. Access to CFD Support, go to Resources / Download Software / OpenFOAM for windows and fill in the required fields to enable the download button.
2. Decompress the software, run as administrator, perform the default installation without modifying check-boxes.
3. Verify the installation on your PC at This computer / C: / OpenFOAM / 20.09 / User-dev / run.
4. Place the shortcut icon for the software in an accessible location, launch it, and confirm access to the "run".
5. Finally, use the command "paraFoam.exe" to open ParaView.

Notes

This software is a revised version, so it is configured to install the necessary packages.

Chapter 3

Basic folder structure for simulations in OpenFOAM

```
MyCase (working directory)/
|-- 0/
|   |-- U
|   |-- p
|   |-- nut, k, epsilon, etc
|-- constant/
|   |-- polyMesh/
|   |   |-- boundary
|   |   |-- points, faces, etc.
|   |-- transportProperties
|   |-- turbulenceProperties
|-- system/
|   |-- controlDict
|   |-- decomposeParDict
|   |-- fvSchemes
|   |-- fvSolution
|   |-- snappyHexMeshDict
|   |-- some functions ...
```

|----------

Recommendation

- Creating cases at 1:1 scale, meaning using the real dimensions of the object or system being simulated, has several advantages compared to using smaller scales. Here are some of the main benefits:
  - **Boundary Conditions:** Boundary conditions and flow properties can be applied more accurately since no scaling is needed.
  - **Realistic Results:** The simulation results tend to be more realistic and directly applicable to real-world situations.
  - **Proportionality:** Working at a 1:1 scale avoids proportionality issues that can arise when scaling dimensions, material properties, and boundary conditions.
  - **No Adjustments Needed:** There's no need to adjust or transform the simulation results to match real dimensions.
  - **Ease of Implementation:** Defining geometry, material properties, and boundary conditions is more straightforward without additional conversions or calculations for scaling.
  - **Experimental Validation:** It facilitates the comparison and validation of simulation results with experimental data or real-world observations.
  - **Reproducibility:** Results can be more easily reproduced and verified by other researchers or engineers working on the same problem.
  - **Detailed Resolution:** It allows capturing detailed and complex flow characteristics that might be lost when using smaller scales.
- However, it's important to note that working at a 1:1 scale can also increase computational requirements due to the need for a finer mesh to capture all the flow details and geometries. This can lead to longer computation times and greater hardware resources. Therefore, balancing the need for accuracy with available computational resources is crucial.

Chapter 4

Introduction to SnappyHexMesh utility

SnappyHexMesh is a three-dimensional (3D) mesh generation tool used in computational fluid dynamics (CFD) and is part of the open-source software OpenFOAM (Open Field Operation and Manipulation).

4.1 Features of SnappyHexMesh:

- **Automatic Mesh Generation:** SnappyHexMesh generates meshes from CAD (Computer-Aided Design) geometries, allowing a more automated process compared to other meshing tools.

- **Hexahedral Meshes:** The name "HexMesh" refers to the ability to generate meshes predominantly composed of hexahedral elements, which are cubes or rectangular blocks in 3D.

- **Local Refinement:** It allows local refinement of the mesh, meaning smaller and more detailed elements can be created in areas of specific interest, such as near complex surfaces or in regions where large flow gradients are expected, maintaining the integrity of the original geometry.

- **Use of Phases:** The mesh generation process in SnappyHexMesh is carried out in several phases:
  - **0. Background mesh:** An initial coarse mesh, forming the base for further refinement.
  - **1. Castellated mesh:** Refines the background mesh and cuts cells to match the geometry's outer shape, identifying shared cells.
  - **2. Snapping (internal or external flow):** Adjusts the mesh to closely fit the surface of the geometry, improving accuracy, depending on whether internal or external flow will be studied.
  - **3. Add layers:** Adds prismatic boundary layers to the mesh to better capture near-wall flow details.

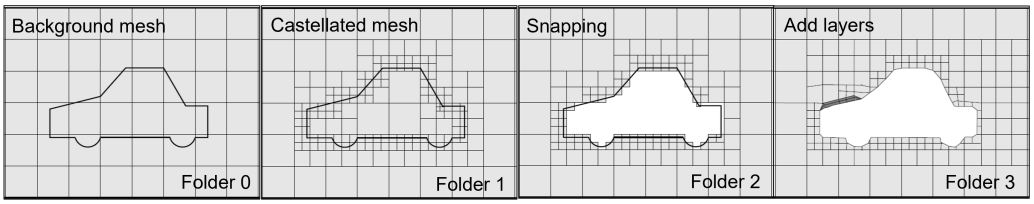


Figure 4.1: Meshing process [2].

Each phase of the process generates a folder, and each folder contains a subfolder named "polymesh", which houses the files comprising the mesh from that particular phase.

The number of phases required depends on the complexity of the mesh being generated.

- **Compatibility with OpenFOAM:** Being part of OpenFOAM, meshes generated with SnappyHexMesh integrate seamlessly with OpenFOAM's CFD solvers, facilitating the setup and execution of simulations.
- **Import and export meshes:** SnappyHexMesh allows importing geometries and meshes in various standard formats such as STL, facilitating interoperabil-

ity with other simulation software. This capability is crucial for using externally generated complex geometries or exporting meshes generated in SnappyHexMesh to other simulation environments, such as ANSYS Fluent or other solvers that support these formats.

- **Parallelization:** SnappyHexMesh and OpenFOAM are designed to leverage parallel computing capabilities, which significantly accelerates the mesh generation and CFD simulations on systems with multiple CPU cores or computer clusters. Parallelization in SnappyHexMesh distributes the mesh generation workload across multiple processors or nodes, enhancing efficiency and reducing runtime for complex problems and large datasets.

4.2 Basic folder structure for mesh creation

Here is the basic structure for creating meshes in SnappyHexMesh:

```
ExampleMeshX (working directory)/
|-- 0/
|-- constant/
|   |-- triSurface/
|   |   |-- STL files.
|-- system/
|   |-- blockMeshDict
|   |-- controlDict (Essential, not modified for BGM)
|   |-- decomposeParDict (Not essential, necessary to parallelization)
|   |-- snappyHexMeshDict
|   |-- surfaceFeaturesDict
|   |-- fvSchemes   (Essential, not modified for BGM)
|   |-- fvSolution  (Essential, not modified for BGM)
|   |-- meshQualityDict
|----------------------
```

Folders "0", "constant" and "system" are essential to perform any process in OpenFOAM.

Here is an example of a script header:

```
/*--------------------------------*- C++ -*----------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: Source
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "FolderLocation";
    object      SpecificFileToModify;
}
```

In OpenFOAM file headers, what typically changes are the specific details of the class, location and object, while the general structure of the comment and the information about OpenFOAM remain consistent.

Chapter 5

Case 1: Meshing of a hexahedron inside a wind tunnel

5.1 Creating a uni-block Background Mesh

The Background mesh (BGM) refers to the initial mesh that serves as the foundation for the detailed mesh generation process in tools like SnappyHexMesh. This stage establishes a basic structure onto which additional transformations and refinements are applied to capture the complex geometry of the simulation domain. It is crucial for ensuring an accurate representation of the domain in CFD simulations, providing an initial platform upon which more advanced phases such as "castellated mesh", "snap", and "add layers" are built.

5.1.1 Steps to generate a uni-block Background mesh

1. Create a folder named "ThisMeshingCourse" inside the "run" folder: This computer / C: / OpenFOAM / 20.09 / User-dev / run / ThisMeshingCourse. In this folder we will save the meshes of this course (ExampleMesh1, ExampleMesh2, etc...).

2. Modify the "blockMeshDict" file, located in sub-folder "system", according to Figure 5.1 and specifying desired patches to bound-

ary conditions [3].

- **Consider the origin of geometry as the BGM origin.**
- **The vertex order must follow the counterclockwise convention.**

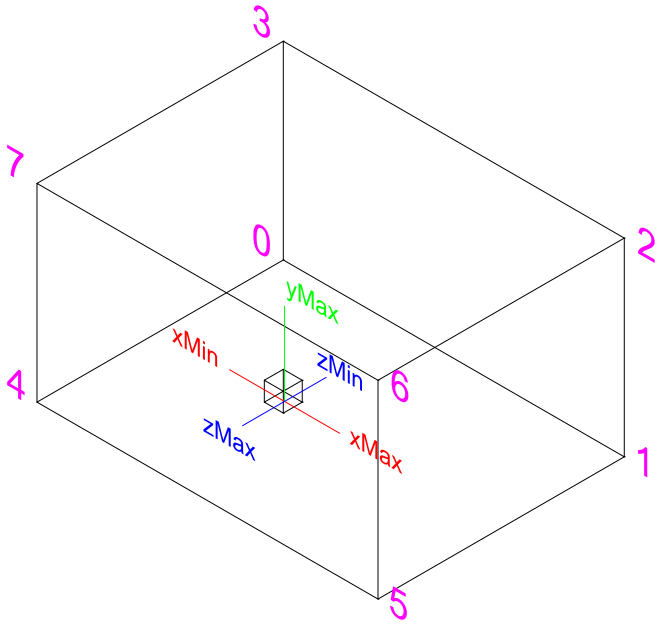


Figure 5.1: Uni-block Background Mesh sketch.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
#inputSyntax slash;

backgroundMesh
{
    xMin   -1.1;
    xMax   3.1;
    yMin   0;
    yMax   1.4;
    zMin   -1.5;
    zMax   1.5;
    xCells xxx;  // Number of divisions on the x-axis
    yCells yyy;  // Number of divisions on the y-axis
    zCells zzz;  // Number of divisions on the z-axis
}

convertToMeters 1; // Scaling factor

vertices /* 0,1,2,3,4,5,6,7.  $! allows for the interpolation of
variables, as well as the inclusion of mathematical expressions
or references to other variables. */
(
  ($!backgroundMesh/xMin $!backgroundMesh/yMin $!backgroundMesh/zMin)
  ($!backgroundMesh/xMax $!backgroundMesh/yMin $!backgroundMesh/zMin)
  ($!backgroundMesh/xMax $!backgroundMesh/yMax $!backgroundMesh/zMin)
  ($!backgroundMesh/xMin $!backgroundMesh/yMax $!backgroundMesh/zMin)
  ($!backgroundMesh/xMin $!backgroundMesh/yMin $!backgroundMesh/zMax)
  ($!backgroundMesh/xMax $!backgroundMesh/yMin $!backgroundMesh/zMax)
  ($!backgroundMesh/xMax $!backgroundMesh/yMax $!backgroundMesh/zMax)
  ($!backgroundMesh/xMin $!backgroundMesh/yMax $!backgroundMesh/zMax)
);

blocks //Segments per axis
(
    hex (0 1 2 3 4 5 6 7)
    (
        $!backgroundMesh/xCells
        $!backgroundMesh/yCells
        $!backgroundMesh/zCells
    )
    simpleGrading (1 1 1) /* It is a tool for adjusting the density of the
    structured mesh, allowing adaptation of spatial resolution according
    to specific simulation requirements.
    It will be modified in future sections. */
);

boundary
(
    inlet
```

```
64      {
65          type patch; // See basic OF concepts chapter
66          faces
67          (
68              (0 3 7 4)
69          );
70      }
71
72      outlet
73      {
74          type patch;
75          faces
76          (
77              (1 5 6 2)
78          );
79      }
80
81      ground
82      {
83          type wall; // See basic OF concepts chapter
84          faces
85          (
86              (0 1 5 4)
87          );
88      }
89
90      boundaryright
91      {
92          type symmetry; // See basic OF concepts chapter
93          faces
94          (
95              (4 7 6 5)
96          );
97      }
98
99       boundaryleft
100     {
101         type symmetry;
102         faces
103         (
104             (0 1 2 3)
105         );
106     }
107
108      boundarytop
109     {
110         type symmetry;
111         faces
```

```
        (
            (3 2 6 7)
        );
    }
);

// ************************************************************************* //
```

3. •In the working directory, execute the command "blockMesh.exe".

   •Use the "paraFoam.exe" command to visualize the BGM, explore the "Mesh Regions" in the "Properties" panel, visualize the "Axes Grid" and use "Camera Parallel Projection" in the "View" panel and enable "Surface With Edges" in the "Display" panel.

   •A new folder named "polymesh" has been created within the "constant" sub-folder. Here is the "boundary" file, contained in the polymesh folder.

   •To modify the BGM, it is necessary to delete the "polymesh" folder and repeat the previous step.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |

   OpenFOAM for Windows 20.09 (v1)
   Built by CFD Support, www.cfdsupport.com (based on Symscape).
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

6
```

```
(
    inlet
    {
          type            patch;
        nFaces          2625;
        startFace       812700;
    }
    outlet
    {
        type            patch;
        nFaces          2625;
        startFace       815325;
    }
    ground
    {
        type            wall;
        inGroups        List<word> 1(wall);
        nFaces          7875;
        startFace       817950;
    }
    boundaryright
    {
        type            symmetry;
        inGroups        List<word> 1(symmetry);
        nFaces          3675;
        startFace       825825;
    }
    boundaryleft
    {
        type            symmetry;
        inGroups        List<word> 1(symmetry);
        nFaces          3675;
        startFace       829500;
    }
    boundarytop
    {
        type            symmetry;
        inGroups        List<word> 1(symmetry);
        nFaces          7875;
        startFace       833175;
    }
)

// ************************************************************************* //
```

Play around with the following parameters:

- **xCells, yCells and zCells**

- **simpleGrading (It will be reviewed in future sessions)**

- **Create a copy of the main mesh folder, so as not to lose progress.**

Notes

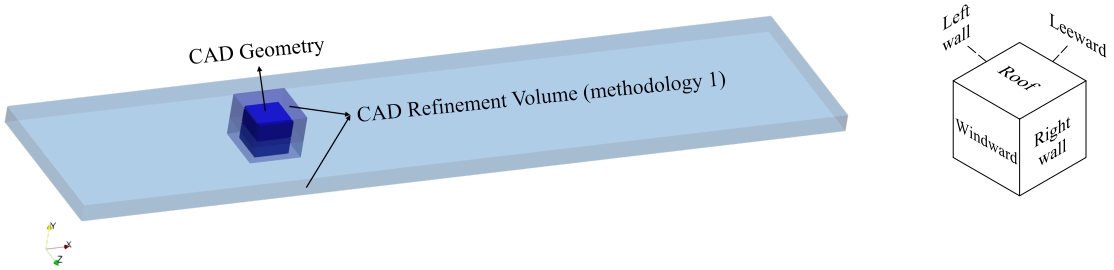
- The working directory is the place where all the files and directories necessary to run a simulation are stored.

Recommendation

- Avoid creating folders or files with spaces or special characters such as: ! @ # $ % ^ & * ( ) [ ] { } ; ' " , < > ? This can potentially cause issues with certain commands and scripts within the OF environment.

5.2 Importing geometry into OpenFOAM

The extraction of surface features from CAD geometry is crucial for generating high-quality meshes that accurately capture important geometric details, thereby improving the fidelity of simulations.



(a) CAD geometry and refinement volumes (b) Surfaces

Figure 5.2: CAD geometry and refinement volumes and surface definition

5.2.1 Steps to import geometry into OpenFOAM

1. According to the diagram in section 4.2, in "constant", sub-folder "triSurface", place the geometry files in STL format inside: This computer / C: / OpenFOAM / 20.09 / User-dev / run / ThisMeshingCourse / ExampleMeshX / constant / triSurface, to convert the geometry into a format that OpenFOAM can read.

2. Modify the "surfaceFeaturesDict" file, located in sub-folder "system".

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      surfaceFeaturesDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
16
17 surfaces // STL files representing specific parts of the geometry
18 (
19   "Barlovento.stl"
20   "Sotavento.stl"
21   "Cubierta.stl"
22   "MDerecho.stl"
23   "MIzquierdo.stl"
24   "Solid1.stl"
25   "Vref1.stl"   // Refinement volume, methodology 1
26   "Vref2.stl"   // Refinement volume, methodology 1
27   "Vref3.stl"   // Refinement volume, methodology 1
28   "Vref4.stl"   // Refinement volume, methodology 1
29   "Vreftop.stl" // Refinement volume, methodology 1
30 );
31
32 includedAngle 150; /* Sets a criterion for identifying geometric features
33 on those surfaces based on the angle between adjacent faces */
34
35 // ************************************************************************* //
```

3.

- In the working directory, execute the command "surfaceFeatures.exe".
- A new folder named "extendedFeatureEdgeMesh" has been created within the "constant" sub-folder. Here are whole parts of geometry.
- Verify that in the "triSurface" folder, ".eMesh" files have been created for the whole parts of geometry.
- **To add more parts of geometry, it is necessary to delete the folder "extendedFeatureEdgeMesh" and ".eMesh" files in "triSurface" and repeat the previous step.**

Notes

- The names assigned to each part of the mesh must be consistent in the scripts and other files.

Recommendation

- From the CAD software, export STL files in meters, ASCII format, and preserve the origin point of the geometry in the global space (*do not convert STL output data to positive space*) SolidWorks example

5.3 snappyHexMeshDict file

The snappyHexMeshDict file is a configuration file used by OpenFOAM's snappyHexMesh utility to generate complex 3D meshes from CAD geometries. It defines various meshing parameters and controls, including geometry input, castellated mesh settings, snapping settings, and boundary layer addition. The file also includes quality control criteria and flags for additional outputs. Key sections often include geometry, castellatedMeshControls, snapControls, and addLayersControls, as shown in Fig. 4.1, each specifying detailed settings for different stages of the meshing process. **Proper configuration of this file is crucial for creating high-quality meshes tailored to specific simulation needs.**

5.3.1 Steps to modify the snappyHexMeshDict file without surface layers

1. Enable or disable mesh processes  as follow:

```
/*--------------------------------*- C++ -*----------------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  dev
   \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
 version     2.0;
 format      ascii;
 class       dictionary;
 object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Which of the steps to use?
castellatedMesh true;  //or false
snap            true;  //or false
addLayers       false; //or true
```

2. Define the surfaces and volumes:

The "geometry" section in the snappyHexMeshDict file is a **crucial part of the meshing process in OpenFOAM's "snappyHexMesh" utility.** It defines all the surface meshes that will be used to generate the computational mesh. Each entry specifies an STL file, indicating a part of the geometry to be meshed, along with its type ("triSurfaceMesh") and a user-friendly name for easy reference. This section ensures that snappyHexMesh can accurately interpret and utilize the provided geometrical surfaces, forming the foundation for the mesh generation. Properly configuring this section is essential for achieving a high-quality mesh tailored to your simulation needs.

The following code shows how to define parts of geometry refinements and refinement volumes (the first of two methodologies for creating refinement volumes):

```
geometry // Definition of all surfaces to be worked on.
{
 Solid1.stl
 {
  type triSurfaceMesh;
  name Solid1;
 }
 Barlovento.stl
 {
  type triSurfaceMesh;
  name Barlovento;
 }
 Sotavento.stl
 {
  type triSurfaceMesh;
  name Sotavento;
 }
 Cubierta.stl
 {
  type triSurfaceMesh;
  name Cubierta;
 }
 MIzquierdo.stl
 {
  type triSurfaceMesh;
  name MIzquierdo;
 }
 MDerecho.stl
 {
  type triSurfaceMesh;
  name MDerecho;
 }
 Vref1 // Methodology 1
 {
  type triSurfaceMesh;
  file "Vref1.stl";
 }
 Vref2 // Methodology 1
 {
  type triSurfaceMesh;
```

```
    file "Vref2.stl";
   }
   Vref3 // Methodology 1
   {
    type triSurfaceMesh;
    file "Vref3.stl";
   }
   Vref4 // Methodology 1
   {
    type triSurfaceMesh;
    file "Vref4.stl";
   }
   Vreftop // Methodology 1
   {
    type triSurfaceMesh;
    file "Vreftop.stl";
   }
  };
```

3. Modify the castellatedMeshControls and snapControls:

This part of snappyHexMesh file is responsible for defining how the initial background mesh is refined and adjusted based on the specified geometry to create a structured mesh. Key parameters include limits on the number of cells, refinement criteria, and specifications for capturing geometric features. By setting these controls, you can ensure that the mesh accurately represents complex geometries while maintaining computational efficiency.

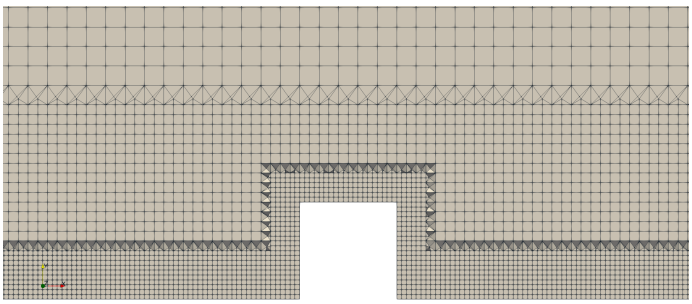


Figure 5.3: Example of a castellated mesh

•**Mesh refinement.**

Refinement refers to the process of adjusting the mesh of a simulation domain to improve resolution in specific areas. This adjustment can be crucial for capturing fine details of the flow and obtaining more accurate results in simulations. Refinement can be applied globally to the entire mesh or locally to specific regions where important phenomena are expected to occur.

In OpenFOAM, there are several methods for mesh refinement, most common are:

– **Global refinement:** Uniformly increasing the resolution of the entire mesh.

– **Local refinement:** Increasing the resolution in specific regions of the simulation domain, such as near walls, in areas with large gradients of variables of interest, or around complex geometries.

Figure 5.4 shows a schematic of the refinement principle.

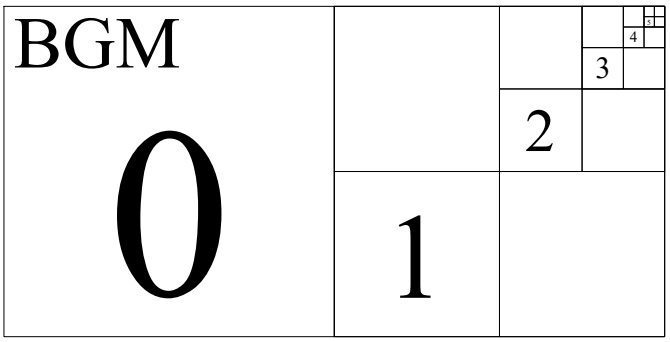
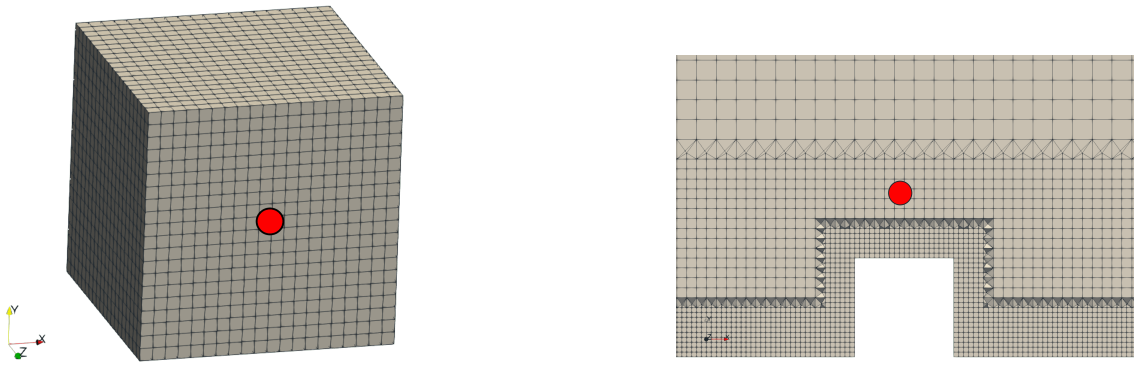


Figure 5.4: Refinement levels

•**locationInMesh definition.**

It is essential to be clear about the type of flow to be studied, whether **internal or external flow.** Based on this decision, a point must be positioned inside or outside the mesh as described in Fig.5.5:



(a) Internal Flow (b) External Flow

Figure 5.5: Meshes to internal and external flow

To generate an internal or external mesh, follow these instructions: Open the "snappyHexMeshDict" file, located in the sub-folder "system", scroll down to "locationInMesh" and modify this point coordinates. **The point defined in locationInMesh must not be on any surface of the geometry.**

The following code shows how to modify the castellatedMeshControls:

```
1  castellatedMeshControls /* Creates a structured mesh from a background mesh.
2  Determine how this initial mesh is refined and adjusted to the geometry. */
3  {
4
5   /* The maximum number of cells allowed in each processor.
6   If this number is exceeded, snappyHexMesh stops local refinement. */
7   maxLocalCells 2000000;
8
```

```
/* The total maximum number of cells allowed in the entire mesh. */
maxGlobalCells 15000000;

// Minimum number of cells to refine between refinement levels.
minRefinementCells 10;

// Number of cells between refinement levels.
nCellsBetweenLevels 2;


/* Explicit feature edge refinement:
Defines the feature files (usually .eMesh files) that contain
the edges of the geometry that need to be captured precisely. */
features
(

);

// Defines the geometry surfaces and the refinement levels applied to them.
refinementSurfaces
{
 "(Barlovento|Sotavento|Cubierta|MIzquierdo|MDerecho)"
 {
  level (2 2); // Minimum and maximum refinement levels
  patchInfo
  {
   type wall;
  }
 }
}

/* Feature angle used to detect sharp edges in the geometry.
If the angle between two adjacent cells is greater than this value,
it is considered a feature that needs to be refined. */
resolveFeatureAngle 30;

// Defines regions in space and the refinement levels applied to these regions.
refinementRegions
{
 Vref1
 {
  mode inside;
  levels ((1 1));
 }
 Vref2
 {
  mode inside;
  levels ((2 2));
```

```
    }
    Vref4
    {
     mode inside;
     levels ((2 2));
    }

    Vreftop
    {
     mode inside;
     levels ((1 1));
    }
   }


   // Mesh selection.
   locationInMesh (2.5 1 1); //External
   // locationInMesh (0 0.01 0); //Internal

   // Allows or disallows the existence of free-standing faces in mesh zones.
   allowFreeStandingZoneFaces true;
  }

  snapControls /* Controls the second snapping phase.
  The mesh created during the castellation phase is adjusted to better align with the surface geome
  This involves moving mesh points to the nearest surface and refining to capture the geometry accu
  {
   /* Number of iterations for internal smoothing to reduce non-orthogonality at
   the face of refinement (effectively making the faces non-planar). Due to complex models, the num
   mesh is rarely orthogonal. So the non-orthogonality correction must be made for stability and ac
   nSmoothPatch 3;

   /*  Relative distance for points to be attracted by surface feature point
   or edge. True distance is this factor times local maximum edge length. */
   tolerance 1.0;

   /* Number of mesh displacement relaxation iterations. Increasing the number of solve iterations
   help in achieving a more accurate alignment with the surface but may increase computational cost
   nSolveIter 300;

   /*  Number of relaxation iterations during the snapping. If the mesh does not conform the geomet
   all the iterations are spend, user may try to increase the number of iterations. Relaxation iter
   in stabilizing the snapping process by preventing abrupt changes in the displacement field. */
   nRelaxIter 5;

   // Feature snapping

   /* The number of feature snapping iterations.
```

```
 This helps in aligning the mesh with sharp features in the geometry. */
 nFeatureSnapIter 10;

 /* Implicit feature snapping tries to detect and snap to sharp edges
 automatically without requiring explicit feature files. */
 implicitFeatureSnap true;

 /* When enabled, the snapping process uses the feature edges
 defined in the features section of castellatedMeshControls. */
 explicitFeatureSnap false;

 // Detect features between multiple surfaces.
 multiRegionFeatureSnap true;
}
```

4. Modify the meshQualityControls, writeFlags and mergeTolerance:

```
meshQualityControls // Quality criteria for the generated mesh.
{
 #include "meshQualityDict"

 /* Set of mesh quality parameters that can be used in specific
 meshing phases where relaxed rules are allowed. */
 relaxed
 {
  /* Non-orthogonality is a measure of how much the cell faces deviate
  from being perpendicular to each other. High non-orthogonality can
  affect the accuracy and stability of numerical simulations. */
  maxNonOrtho 75;
 }
}

// Advanced
/* Control the writing of extra information to assist in
post-processing and mesh analysis. */
writeFlags
(
scalarLevels    // write volScalarField that contains the cell refinement levels.
layerSets       /* write cellSets, faceSets of faces in layer. This is useful for
verifying that the boundary layers have been added correctly and cover the
intended surfaces. */
layerFields     /* write volScalarField for layer coverage. This is particularly
useful for ensuring that the boundary layers are correctly formed and meet the
specified thickness and growth ratios. */
);

/* helps manage the quality of the mesh by merging points that are within
```

```
31  a specified small distance of each other.
32  Properly tuning this parameter can greatly enhance the robustness and accuracy
33  of the resulting mesh, leading to better simulation outcomes.
34  Note: the write tolerance needs to be higher than this. */
35  mergeTolerance 1E-6;
```

5.
- In the working directory, execute the command "snappyHexMesh.exe".
- New folders will appear in the working directory (1, 2, 3), depending on the processes activated and configured in step 1 of this process. The meshing processes were shown in Figure 4.1.

6. Make sure the process has completed without errors:

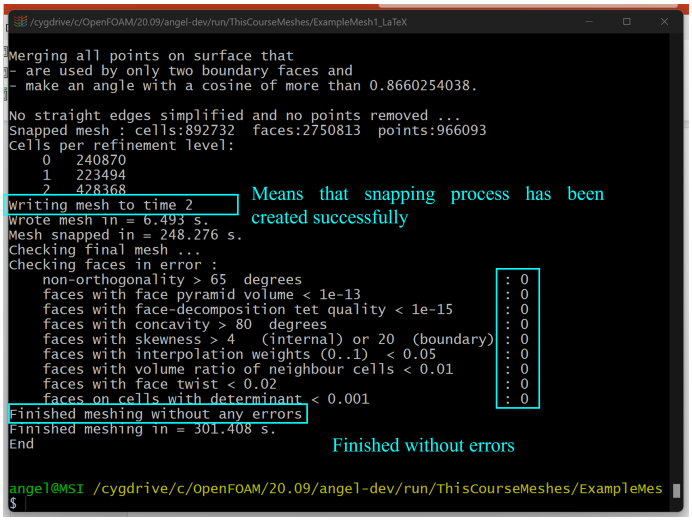


Figure 5.6: Example of successful completion of the meshing process, without adding surface layers.

If you receive any error messages, modify the necessary scripts and repeat step 5 as needed.

For each execution of the snappyHexMesh command, it

is necessary to delete the generated temporary folders 1, 2 and 3 from the working directory. Keep folder 0.

Play around with the following parameters:

- **refinementSurfaces (minimum and maximum)**
- **minRefinementCells**
- **nCellsBetweenLevels**
- **Create a copy of the main mesh folder, so as not to lose progress.**

5.3.2 Steps to modify the snappyHexMeshDict file with surface layers, relativeSizes true or false

The layer addition process uses the settings in the "addLayersControls" sub-dictionary in snappyHexMeshDict, with the entries listed below. The user can choose from four different layer thickness parameters — expansionRatio, finalLayerThickness, firstLayerThickness, thickness — but must specify only 2 of them; specifying more than 2 results in an over-specified problem [4].

relativeSizes, true: When relativeSizes is enabled, this means that the sizes of the refined cells are defined relative to the base cell size in the refinement region (ratio).
relativeSizes, false: When relativeSizes is not enabled, the sizes of the refined cells are specified in absolute terms rather than relative to the base cell size (meters).

Once a structured mesh with the desired refinement levels, as explained in the previous process (5.3.1), is obtained, the temporary folders created should be deleted, and the following steps should be undertaken:

1. Enable addLayers process in the snappyHexMeshDict file, as follow:

```
addLayers       true;
```

Chapter 6

Obtaining first cell height and parameters to addLayersControls prism layers

A crucial aspect of modeling the boundary layer is calculating the height of the first cell from the wall. This value is directly related to the dimensionless parameter y^+, which is a measure of the distance from the wall in terms of viscous length units. Choosing the appropriate first cell height ensures that the no-slip wall conditions and velocity gradients are accurately represented, which is essential for turbulence models.

A fisherman uses a fishing net, which is a type of mesh structure. If he is trying to catch small fish, then the mesh size needs to be small enough to capture them. In this case, large fish will also be caught. Similarly, if we intend to resolve the effects near the wall, that is, in the viscous sublayer, then the mesh size must be small and dense enough near the wall to capture almost all the effects.

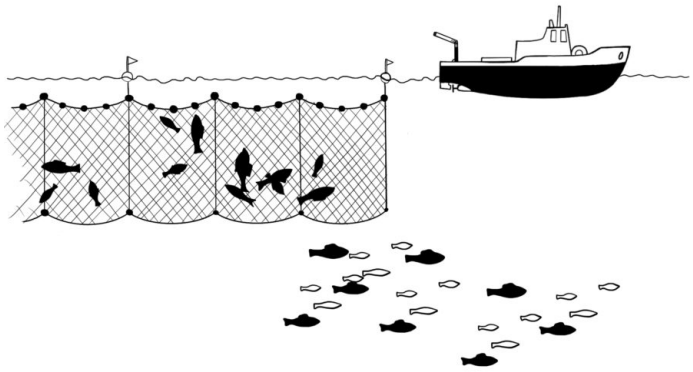


Figure 6.1: Mesh size analogy.

In this chapter, we will learn how to determine the height of the first cell using a web tool, as well as the values needed to configure the addLayersControls subdictionary and obtain the desired prism layers using a function written in Python.

6.1 Calculation of y^+

The dimensionless wall distance y^+ is a crucial parameter in the resolution of the boundary layer in CFD simulations. It is defined as:

$$y^+ = \frac{y_P u^* \rho}{\mu} = \frac{y_P u^*}{\nu}$$

where:

- y_P is the distance from the wall to the center of the first cell.
- u^* is the friction velocity.
- ρ is the density.
- μ is the dynamic viscosity of the fluid.
- ν is the kinematic viscosity of the fluid.

The friction velocity u^* can be calculated using the wall shear stress τ_w:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}$$

where:

- τ_w is the wall shear stress.
- ρ is the fluid density.

The wall shear stress τ_w can be estimated for a turbulent boundary layer over a flat plate using empirical correlations, such as the one from the law of the wall:

$$\tau_w = \rho u_\tau^2$$

Given the complexity of obtaining exact values for u_τ and τ_w, practical approaches often involve using known or estimated y^+ values to back-calculate the first cell height y.

6.1.1 Example Calculation

Let's assume the following parameters for our example:

- Desired y^+ value: 30 (a typical target for wall-resolved LES or RANS simulations).
- Kinematic viscosity ν: $1.5 \times 10^{-5}\,\mathrm{m}^2/\mathrm{s}$ (typical for air at room temperature).
- Fluid density ρ: $1.225\,\mathrm{kg}/\mathrm{m}^3$.
- Estimated wall shear stress τ_w: $0.1\,\mathrm{N}/\mathrm{m}^2$.

First, calculate the friction velocity u_τ:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} = \sqrt{\frac{0.1}{1.225}} \approx 0.285\,\mathrm{m/s}$$

Next, rearrange the y^+ formula to solve for the first cell height y:

$$y = \frac{y^+ \nu}{u_\tau}$$

Substitute the known values:

$$y = \frac{30 \times 1.5 \times 10^{-5}}{0.285} \approx 1.58 \times 10^{-3}\,\mathrm{m} = 1.58\,\mathrm{mm}$$

So, for a target y^+ of 30, the height of the first cell y should be approximately 1.58 mm.

This example illustrates the process of calculating the first cell height based on a desired y^+ value, ensuring that the boundary layer is adequately resolved in the simulation.

Chapter 7

Basic OpenFOAM Concepts

- **patch** It's an interface or boundary between a region and its surroundings in a simulation domain. It is defined as an area of the mesh or wall where boundary conditions can be applied, specifying values for velocity, pressure, temperature, or other flow variables. Patches are used to represent the conditions at the boundaries of the simulation domain. They can represent solid walls, flow inlets, flow outlets, symmetries, material interfaces, among others. Each patch has a specific type that determines how the boundary conditions are defined at that location.

- **wall** A type of boundary condition used to represent solid surfaces within the simulation domain. These surfaces interact with the fluid flow in a specific way, usually by enforcing a no-slip condition.

- **symmetry** A type of boundary condition used to define a plane where the flow and other variables are mirrored, ensuring no fluid or scalar quantity crosses the plane. This means the velocity and gradients perpendicular to the plane are zero.

Chapter 8

ParaView basic tools

ParaView is an application for scientific data visualization and analysis, designed especially for handling data from numerical simulations and scientific experiments. It's a powerful open-source tool that enables researchers and scientists to visualize, analyze, and interpret complex 3D data.

Figure 8.1: ParaView logo

Listed below are some important features of the software.

- **Interactive Visualization:** Allows interactive visualization of complex datasets in 2D and 3D, including meshes, volumes, and particle data.
- **Support for Multiple Formats:** Can read a wide range of scientific data formats, including common formats used in computational simulation such as VTK (Visualization Toolkit), OpenFOAM, among others.
- **Analysis and Post-Processing:** Facilitates data analysis with tools to compute gradients, integrals, histograms, and more on the visualized data.

- **Advanced Rendering:** Offers advanced rendering techniques like lighting, shading, and special effects to enhance data visualization.
- **Intuitive Graphical Interface:** Has an intuitive graphical user interface (GUI) that makes it easy to explore data and configure complex visualizations.

Below are the keys and commands to use and steps to perform specific tasks in ParaView 5.8.0. Each instruction assumes that the software is already open.

KEYS AND COMMANDS TO USE

- Scroll ↑ Zoom in.
- Scroll ↓ Zoom out.
- Scroll button + move the mouse Activates the panning or 3D scene scrolling function.
- Left click + move the mouse Activates the 3D rotation.

8.1 Visualize different geometries in STL

Go to: File → Open → Navigate to the folder where the STL files are located → In "Files of type", select "All Files (*)" → Select the desired files to display → STL Reader → OK → In the "Properties" panel, click "Apply".

8.1.1 Axes grid of geometry

After completing the steps in section 8.1 → in the "View" panel, check the "Axes Grid" box.

8.1.2 Camera parallel projection

After completing the steps in section 8.1 → in the "View" panel, check the "Camera Parallel Projection" box.

8.1.3 Modify opacity to display geometries

After completing the steps in section 8.1 → in the "Pipeline Browser", select the STL geometry to modify its opacity → in the "Display" panel, adjust the "Opacity" bar.

8.1.4 Modify STL color to display geometries

After completing the steps in section 8.1 and 8.1.3 → in the "Pipeline Browser", select the STL geometry to modify its color → in the "Display" panel, "Coloring", select "Solid Color" → Press the "Edit" button 

and select a color.

8.2 Displaying the project for post-processing

Go to: File → Open → Navigate to system folder project → In "Files of type", select "All Files (*)" → Select the controlDict file → OK → Select OpenFOAMReader → OK → Click on Apply in the Properties panel.

8.3 Displaying the last time step

After completing the steps in section 8.2 Go to: Time drop-down bar 

→ Select the last time step.
The same result can be achieved by using the time-step buttons located next to the time drop-down bar  .

8.4 Displaying the mesh cells

After completing the steps in section 8.2 and 8.3 Go to: Display panel → Change the Representation to Surface With Edges.

8.5 Measuring a distance

After completing the steps in section 8.2, 8.3 and 8.4, Zoom in to the element to measure → Select the Ruler tool  → Place the pointer over one of the vertices of the edge to be measured, click and press the P key → Repeat the previous step for the next vertex → In the Properties panel, click on Apply. **Note:** Measurements may not be exact, but if done correctly will show a good approximation.

8.5.1 Modify the measurement distance

.....

8.6 Identifying surface coordinates

After created a mesh or performed a solver and completed steps in section 8.2 and 8.4 → Click on "Last Frame" button  , to visualize the last solution → Go to Properties panel → In Mesh Regions, check the geometry patches boxes → Click on Apply → In View panel check the Camera Parallel Projection box → Zoom in to the geometry → Click on "Hover Points On" button  and click on "OK" → Move to the vertex of interest to know its coordinates. **Note:** To explore another face, click again on "Hover Points On" button and zoom in to interest face.

8.7 Creating a Slice

8.8 Saving a Screenshot

Bibliography

[1] CFD SUPPORT, *CFD SUPPORT*, 2009, https://www.cfdsupport.com/, accessed: 25.06.2024.

[2] Open FOAM, *Mesh generation with the snappyHexMesh utility*, 2004, https://www.openfoam.com/documentation/user-guide/4-mesh-generation-and-conversion/4.4-mesh-generation-with-the-snappyhexmesh-utility, accessed: 01.07.2024.

[3] Open FOAM, *Specification of patch types in OpenFOAM*, 2004, https://www.openfoam.com/documentation/user-guide/4-mesh-generation-and-conversion/4.2-boundaries, accessed: 27.07.2024.

[4] Open FOAM v11 User Guide, *5.5 Mesh generation with the snappyHexMesh utility*, 2023, https://doc.cfd.direct/openfoam/user-guide-v11/snappyhexmesh, accessed: 28.07.2024.