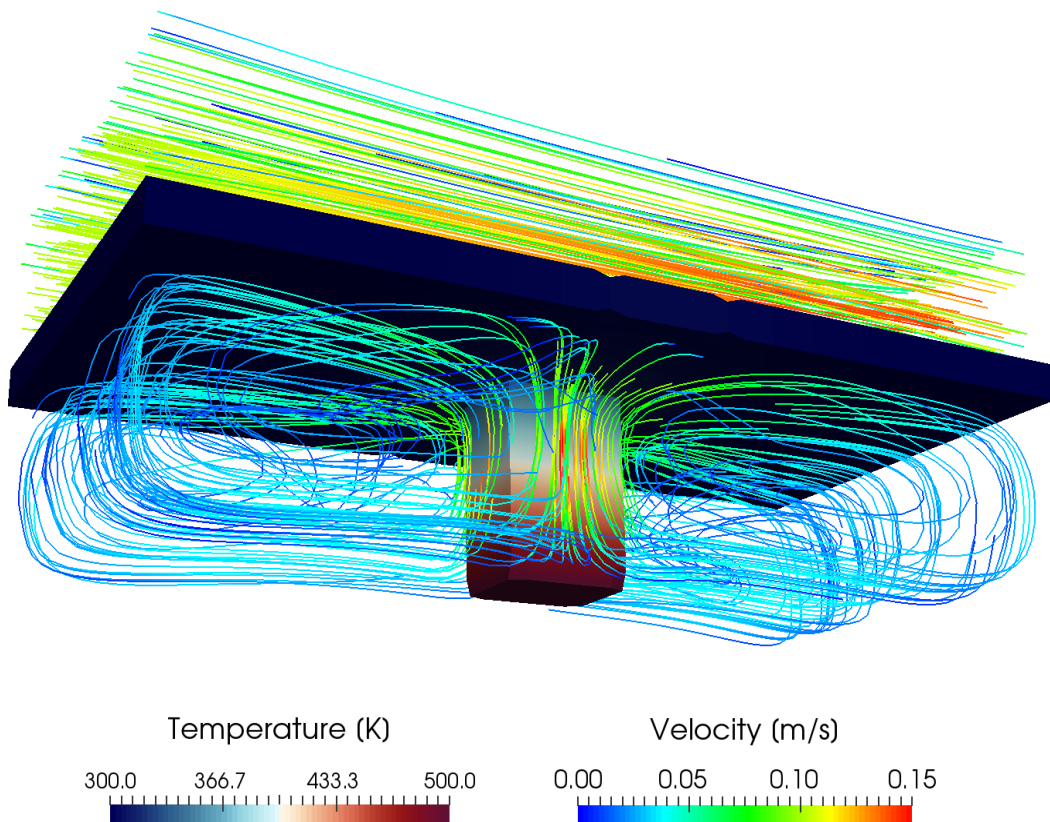


Tutorial Thirteen

snappyHexMesh – Multi-Region



4th edition, Jan. 2018



This offering is not approved or endorsed by ESI® Group, ESI-OpenCFD® or the OpenFOAM® Foundation, the producer of the OpenFOAM® software and owner of the OpenFOAM® trademark.



Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Editorial board:

- Bahram Haddadi
- Christian Jordan
- Michael Harasek

Contributors:

- Bahram Haddadi
- Philipp Schretter
- Yitong Chen

Compatibility:

- OpenFOAM® 5.0
- OpenFOAM® v1712

Cover picture from:

- Bahram Haddadi



Except where otherwise noted, this work is licensed under <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Attribution–NonCommercial–ShareAlike 3.0 Unported (CC BY–NC–SA 3.0)

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

You are free:

- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work

Under the following conditions:

- Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- Noncommercial — You may not use this work for commercial purposes.
- Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

With the understanding that:

- Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.
- Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
- Other Rights — In no way are any of the following rights affected by the license:
- Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
- The author's moral rights;
- Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.
- Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

For more tutorials visit: www.cfd.at

Background

1. Multi-region modeling & why do we need it?

In multi-region modeling, the entire computational domain is divided into individual regions, with each region representing a coherent continuum of the same phase. The key feature of this type of modeling is that separate governing transport equations are solved for each region.

In general, two different approaches have been adopted in the past to solve multi-region problems:

- Monolithic: solve using a single coupled matrix equation system
- Partitioned: solve using separate matrix equation systems

In this OpenFOAM® tutorial, we are focusing on the partitioned approach. The fundamental steps in this approach are outlined below:

1. Define the whole mesh domain and the separate regions within it. Assign cells into each region
2. Specify field variables in each region
3. Solve the transport equation in each individual region
4. Multiregional coupling at the interface between different regions
5. Iteration to achieve fully/coupled solution

2. chtMultiRegionFoam solver

This solver is developed to solve heat transfer problems between multiple regions.

snappyHexMesh - snappyMultiRegionHeater

Simulation

The procedure described in this tutorial is structured in the following order:

- Creation of the geometry data
- Tutorial on Meshing a geometry with more than one region
- Run an OpenFOAM® simulation with the generated mesh using chtMultiRegionFOAM

Objectives

- Understanding multi region meshing with the meshing tool *snappyHexMesh*

Data processing

Import your simulation to ParaView. Analyze the flow field through the flange and the heat distribution in the flange.

1. Pre-processing

1.1. Copy tutorial

Copy the tutorial from the following directory to your working directory:

```
$FOAM_TUTORIALS/mesh/snappyHexMesh/snappyMultiRegionHeater
```

1.2. 0 directory

Unlike the single region simulations in the 0 directory an individual folder per region exist which stores the files including initial and boundary conditions for that region. Also in the 0 directory some files exist which are just dummy files that will not be used in the simulations. The initial and boundary conditions for each region are changed and updated using the *changeDictionary* utility which will be explained later.

1.3. constant directory

Also in the constant directory exist a folder per region; in this case, the domain is split into the following regions: bottom air, heater, left solid, right solid and top air. Within the designated folder, there are relevant dictionaries that describe the physical properties, turbulence or radiation behavior of each region, e.g. *radiationProperties*, *turbulenceProperties* and *thermophysicalProperties*.

The *polyMesh* directory in the constant folder includes the original mesh while the *polyMesh* directories in each region folder include the split mesh for that region with the new boundaries between regions.

Unlike *polyMesh* directories there exist just one *triSurface* folder which stores all the stl files for mesh creation using *snappyHexMesh*.

In the *regionProperties* file, the physical phase of each region is specified. As you can see, bottom and top air regions are fluid, whereas heater, left and right solid are in solid phase.

```
// * * * * *
regions
(
    fluid          (bottomAir topAir)
    solid          (heater leftSolid rightSolid)
);
// * * * * *
```

1.4. system directory

Like constant directory also in system directory a folder per region can be found and all the settings for that region are stored in the corresponding folder, e.g. *fvSolution*, *fvSchemes* and *decomposeParDict*. The *fvSchemes* file in the system directory is a dummy file while the *fvSolution* includes the number of outer correctors setting for PIMPLE algorithm. There is also just one *controlDict* file and it is in main system folder.

*Note: For running the simulations in parallel the *decomposeParDict* files for all the regions should have the same settings as the main one in the system directory. This is not valid for parallel meshing using *snappyHexMesh* while it just uses the *decomposeParDict* file in the main system directory.*

The files needed for creating a multi-region mesh are the same as the mesh for single-region, except for slight differences in *snappyHexMeshDict* file:

locationInMesh: In a multi-region mesh this point is not used but it should be defined just as a place holder.

refinementSurfaces: Different regions are defined in here. E.g. for the region BottomAir all the faces and cells inside the bottomAir stl (each region stl should be a closed volume) file are marked with bottomAir flag (in faceZone and cellZone).

```
// * * * * *
castellatedMeshControls
{
    maxLocalCells 100000;
    maxGlobalCells 2000000;
    minRefinementCells 10;
    nCellsBetweenLevels 2;

    features
    (
        {
            file "bottomAir.eMesh";
            level 1;
        }
        ...
        {
            file "topAir.eMesh";
            level 1;
        }
    );

    refinementSurfaces
    {
        bottomAir
        {
            level (1 1);
            faceZone bottomAir;
            cellZone bottomAir;
            cellZoneInside inside;
        }
        ...
        rightSolid
        {
            level (1 1);
            faceZone rightSolid;
            cellZone rightSolid;
            cellZoneInside inside;
        }
    }

    resolveFeatureAngle 30;

    refinementRegions
    {
    }
    locationInMesh (0.01 0.01 0.01);
    allowFreeStandingZoneFaces false;
}
// * * * * *
```

After creation of the mesh and splitting to different regions the initial and boundary conditions for each region can be manually set in the relevant region folders in 0 directory. This process can be also automated using the changeDictionary utility. The dictionary file for this utility for each region is in the relevant region folder in the system directory: changeDictionaryDict.

See below the changeDictionaryDict file for the heater region. In the boundary sub-dictionary type of boundaries for minY, MinZ and maxZ are set to patch. Then for T the internal field will be overwritten with uniform 300. In the next step all the boundaries in the T file for heater

region will be set to `zeroGradient` (".*" means all the boundaries with any name) and after that the boundaries with the name "heater_to_.*" will be changed to `turbulentTemperatureCoupledBaffleMixed` and `minY` will be changed to `fixedValue`.

```
// * * * * *
boundary
{
    minY
    {
        type            patch;
    }
    minZ
    {
        type            patch;
    }
    maxZ
    {
        type            patch;
    }
}

T
{
    internalField        uniform 300;
    boundaryField
    {
        "."
        {
            type            zeroGradient;
            value            uniform 300;
        }
        "heater_to_.*"
        {
            type            compressible::turbulentTemperatureCoupledBaffleMixed;
            Tnbr            T;
            knappaMethod    solidThermo;
            value            uniform 300;
        }
        minY
        {
            type            fixedValue;
            value            uniform 500;
        }
    }
}
// * * * * *
```

2. Mesh creation and running simulation

The background mesh is created with `blockMesh`.

```
>blockMesh
```

Equal to the single region case, the command `surfaceFeatureExtract` creates the **eMesh** files from the stl files with the geometry data. Also the folder `extendedFeatureEdgeMesh` is created in the constant directory. The creation of eMesh files with the command `surfaceFeatureExtract` is not obligatory. This step is only necessary, if certain edges need to be refined.

```
>surfaceFeatureExtract
```

For performing the meshing in parallel, the geometry needs to be decomposed prior to running `snappyHexMesh`. Depending on the number of subdomains, defined in the `decomposeParDict`, the processor folders are created accordingly.

```
>decomposePar
```

*Note: It is **recommended**, not to use the scotch method to decompose the region. Rather, the hierarchical or the simple method should be used. In case of scotch method, errors can occur while executing snappyHexMesh or while reconstructing the mesh.*

In order to prevent the creation of the folders 1, 2 (castellation and snapping features are turned on while layering is turned off) and only keep the final time step folder with the final mesh, the command `-overwrite` can be added after `snappyHexMesh`. In this case, only one folder, 0, is created with the files `pointLevel` and `cellLevel`. The mesh data in this case is located in `constant/polyMesh`.

```
>mpirun -np 4 snappyHexMesh -parallel -overwrite
```

Note: If `castellatedMesh` and `snap` are set on `true` in the `snappyHexMeshDict`, only the snapped mesh is stored, whereas the intermediate step `castellatedMesh` is overwritten. If `castellatedMesh`, `snap` and `addLayers` are set on `true` in the `snappyHexMeshDict`, only the layered mesh is stored and the previous intermediate steps `castellatedMesh` and `snap` are overwritten.

In this case only the steps `castellatedMesh` and `snap` are set to `true`, as these steps are applied to the whole mesh. The following command reconstructs the final mesh:

```
>reconstructParMesh -constant
```

After running the command `reconstructParMesh`, the following message appears in the terminal window, which can be ignored:

*This is an experimental tool, which tries to merge individual processor meshes back into one master mesh. ...
 Not well tested & use at your own risk!*

After this step all the regions are meshed but the meshes are connected and needs to be split. In the meshing step each region cells are marked with a flag and this flag will be used in the next step to split the mesh. Mesh regions can be split using the following command which split the mesh based on the flagged `cellZones` and overwrite the old meshes in the `polyMesh` directories in the region folders (if any exist):

```
>splitMeshRegions -cellZones -overwrite
```

With the mesh ready, the next step is to apply appropriate field values to each region, according to the `changeDictionaryDict`. This command needs to be repeated for each region, with the name of the region specified after the prefix `'-region'`.

```
>changeDictionary -region heater
...
>changeDictionary -region topAir
```

Now the solver `chtMultiRegionFoam` is ready to be run.

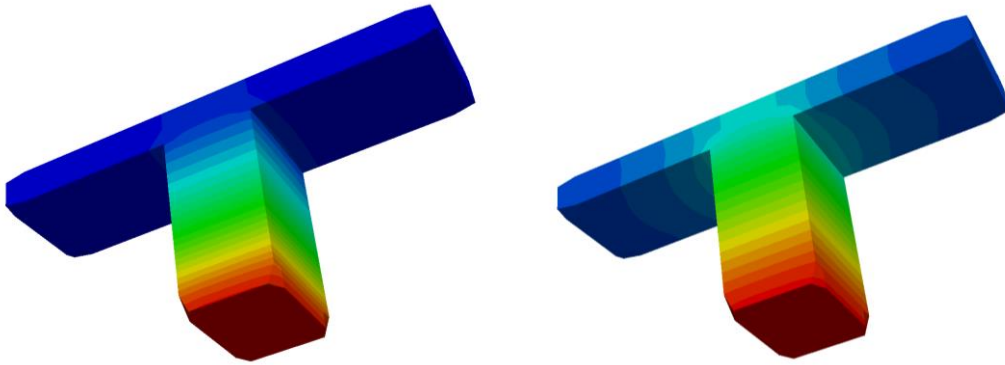
```
>chtMultiRegionFoam
```

Note: `chtMultiRegionFoam` can also be run on several processors.

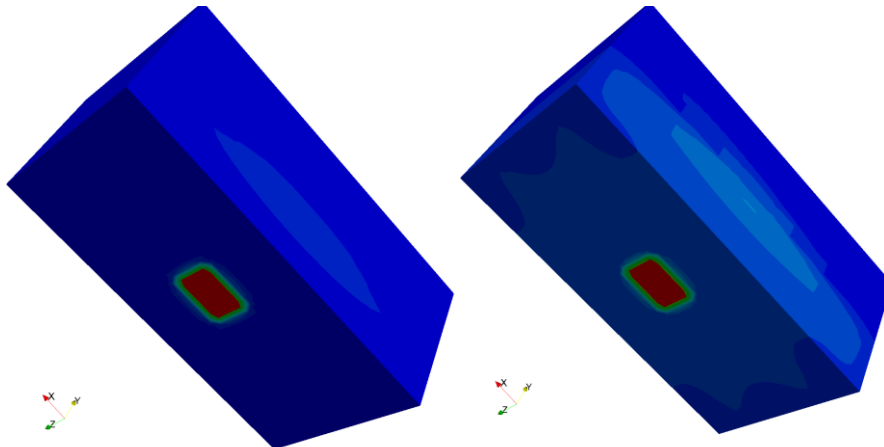
3. Post-processing

The results need to be converted to VTK files for each region with flag `-region`.

```
>foamToVTK -region heater
...
>foamToVTK -region topAir
```



Temperature profile of heater region at time 15s and 75s



Temperature profile of entire mesh at time 15s and 75s

