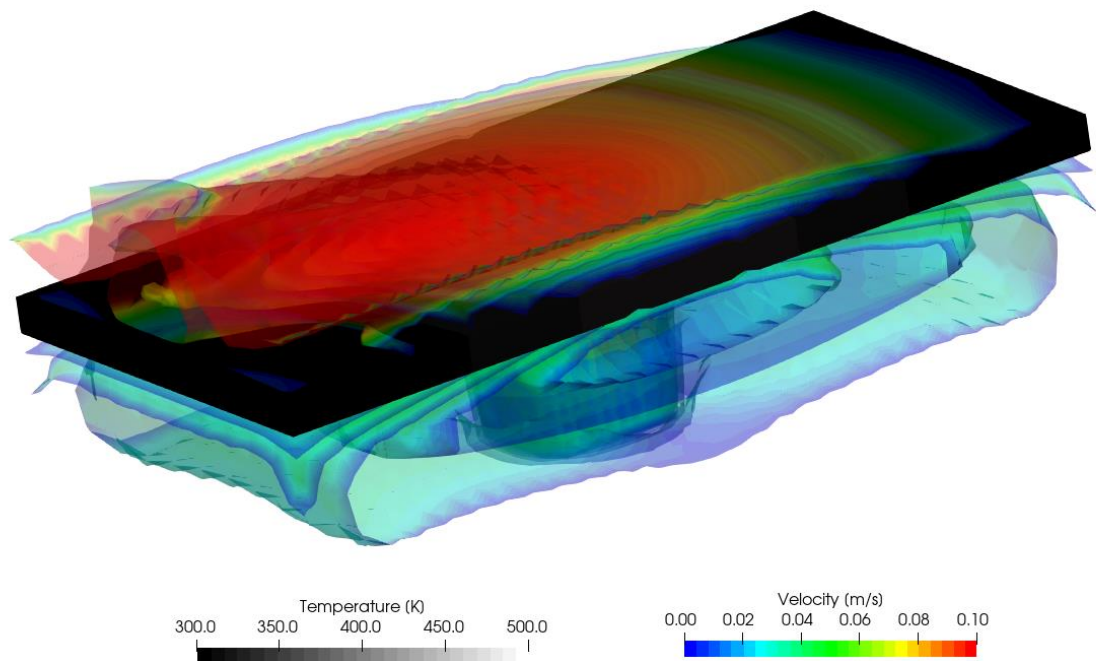


# Tutorial Thirteen


## snappyHexMesh – Multi-Region



**Bahram Haddadi**



7<sup>th</sup> edition, March 2025

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

## Contributors:

- Bahram Haddadi
- Christian Jordan
- Michael Harasek
- Philipp Schretter
- Yitong Chen



Technische Universität Wien  
Institute of Chemical, Environmental  
& Bioscience Engineering



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.

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.

Available from: [www.fluiddynamics.at](http://www.fluiddynamics.at)

## Background

Multi-region modeling is a computational approach used in CFD simulations where the entire computational domain is divided into distinct regions, each representing a specific phase or material. This method is particularly useful for simulating conjugate heat transfer (CHT), fluid-structure interactions, and multiphase flow scenarios.

The key advantage of multi-region modeling is that it allows for separate governing transport equations to be solved for each region, ensuring accurate representation of different physical properties, material behaviors, and interactions at the interfaces.

Multi-region modeling is essential in cases where:

- Different materials with varying thermal properties exist (e.g., heat exchangers, cooling of electronic components, and thermal insulation studies).
- Conjugate heat transfer (CHT) needs to be modeled, where heat conduction through solids and convection in fluids occur simultaneously.
- Interfaces between different phases or substances are present (e.g., solid-liquid).
- Fluid-structure interaction (FSI) must be considered in simulations, where structural deformation affects the surrounding fluid flow.
- Energy transfer in coupled systems requires resolving different governing equations in separate domains.

### Approaches to Multi-Region Modeling

Historically, two primary approaches have been used to solve multi-region problems:

#### 1. Monolithic Approach

A single coupled matrix equation system is used to solve all regions simultaneously. Requires a strong coupling of governing equations across different regions and ensures better numerical stability but is computationally expensive. Typically used in high-accuracy simulations such as finite element-based methods.

#### 2. Partitioned Approach (Used in OpenFOAM®)

Each region is treated as an independent subdomain, with separate governing equations solved for each. Interface conditions between regions are explicitly enforced via boundary conditions and it is computationally more efficient than

the monolithic approach. It is used in scenarios where different physics must be solved separately but interact at interfaces.

### **Steps for Multi-Region Modeling in OpenFOAM®**

#### **Step 1: Define the Computational Domain & Mesh Regions**

The entire simulation domain is divided into multiple regions, typically a combination of fluid and solid domains. Each region is assigned specific material properties, such as: density, viscosity, thermal conductivity and specific heat capacity for fluids and thermal conductivity, heat capacity, and density for solids. The mesh is generated separately for each region using tools like blockMesh, snappyHexMesh, or external meshing software.

#### **Step 2: Assign Field Variables for Each Region**

Each region requires separate field definitions, such as temperature ( $T$ ), velocity ( $U$ ), and pressure ( $p$ ). Thermophysical properties for each region are defined using thermophysicalModels and initial conditions for each region must be assigned in the 0 directory.

#### **Step 3: Solve Transport Equations in Each Region**

The governing equations for mass, momentum, and energy are solved for each individual region. For fluid regions, Navier-Stokes, continuity and energy equations are solved while in the solid region's equations such as Fourier heat conduction equation for temperature distribution are normally considered. Time-stepping is based on the smallest time-step for all the regions and numerical schemes are managed separately for each region.

#### **Step 4: Multi-Regional Coupling at Interfaces**

The interaction between different regions is enforced at interfaces by specifying boundary conditions. The interface conditions can be temperature continuity (matching temperature at the interface) and heat flux continuity (ensuring conservation of energy transfer).

#### **Step 5: Iterative Solution for Fully Coupled Results**

The solver iterates between individual region solutions until convergence is achieved. Adjustments to time-stepping, solver settings, and relaxation factors may be needed for numerical stability.

## snappyHexMesh - snappyMultiRegionHeater

### Tutorial outline

Try to create a multi-region geometry and mesh and run a conjugate heat transfer case using it.

### 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. Copying tutorial

Download the following tutorial to your working directory:

<https://github.com/OpenFOAM/OpenFOAM-5.x/tree/master/tutorials/heatTransfer/chtMultiRegionFoam/snappyMultiRegionHeater>

Either by creating the folders and then downloading the files and place them in relevant folders or using following webpage:

<https://download-directory.github.io/>

Rename the `constant/triSurface` directory to `constant/geometry` directory. In the `system` directory, change the name of `surfaceFeatureExtractDict` to `surfaceFeaturesDict`.

## 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 (the folders can be created manually or will be generated automatically after creating and splitting the mesh). Also in the 0 directory some files exists 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, momentumTransport and physicalProperties. The following changes should be applied (these changes are required, since the original tutorial belongs to OpenFOAM 5):

- In the bottomAir rename thermoPhysicalProperties file to physicalProperties
- In the bottomAir rename turbulenceProperties file to momentumTransport
- Copy the physicalProperties, momentumTransport and g file from bottomAir to topAir folder
- Update the thermoPhysicalProperties file in the constant/heater folder as following

// \* \* \* \* \* //

```
thermoType
{
    type                heSolidThermo;
    mixture              pureMixture;
```

```

        transport      constIsoSolid;
        thermo         eConst;
        equationOfState rhoConst;
        specie         specie;
        energy         sensibleInternalEnergy;
    }

mixture
{
    specie
    {
        molWeight      12;
    }

    transport
    {
        kappa          80;
    }

    thermodynamics
    {
        Hf              0;
        Cv              450;
    }

    equationOfState
    {
        rho             8000;
    }
}

// *****

```

- Copy the thermoPhysicalProperties file from constant/heater folder to constant/leftSolid and constant/rightSolid and replace the old files

The polyMesh directory in the constant folder (after the mesh is created) includes the original mesh while the polyMesh directories in each region folder (after the mesh is splitted) include the split mesh for that region with the new boundaries between regions.

Unlike polyMesh directories there exist just one geometry 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.*

Update the surfaceFeaturesDict file as following:

```
// * * * * *  
  
surfaces ("bottomAir.stl" "heater.stl" "leftSolid.stl" "rightSolid.stl"  
"topAir.stl");  
includedAngle 150;  
writeFeatureEdgeMesh yes;  
  
// * * * * *
```

Change the meshQualityDict as following:

```
// * * * * *  
  
// Include defaults parameters from master dictionary  
#includeEtc "caseDicts/mesh/generation/meshQualityDict"  
  
// * * * * *
```

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 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;
    }
}
// * * * * *
* * * * *
```

In the meshQualityDict file, change the following line:

```
#includeEtc "caseDicts/meshQualityDict"
```

to

```
#includeEtc "caseDicts/mesh/generation/meshQualityDict.cfg"
```

**Note:** Add the missing “,” to the fvSolution files for bottomAir and topAir regions:

```

"(rho|rhoFinal)"
{
    solver          PCG;
    preconditioner   DIC;
    tolerance        1e-7;
    relTol           0;
}
```

Update the laplacianSchemes in the the system/heater/fvSchemes file as following:

```

laplacianSchemes
{
    default          Gauss linear corrected;
    laplacian(alpha,h) Gauss linear corrected;
}
```

In the system/heater/fvSolution change the h, \$h and hFinal to e, \$e and eFinal.

Copy and replace the fvScheme and fvSolution files from system/leftSolid and system/rightSolid with the ones from system/heater

Copy fvSchemes and fvSolution from bottomAir to topAir (replace the files)

## 2. Mesh creation and running simulation

The background mesh is created with blockMesh.

```
>blockMesh
```

Equal to the single region case, the command `surfaceFeatures` 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 `surfaceFeatures` is not obligatory. This step is only necessary, if certain edges need to be refined.

```
>surfaceFeatures
```

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:

```
>reconstructPar -constant
```

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
```

```
>changeDictionary -region bottomAir
```

```
>changeDictionary -region rightSolid
```

```
>changeDictionary -region leftSolid
```

Before running the simulations, the solver for each region needs to be defined, in the controlDict file add the following lines:

```
regionSolvers
{
    bottomAir    fluid;
    topAir       fluid;
    heater       solid;
    leftSolid    solid;
    rightSolid   solid;
}
```

and also remove the *rho* files from *0/topAir* and *0/bottomAir* folders, now it is ready to be run.

```
>foamMultiRun
```

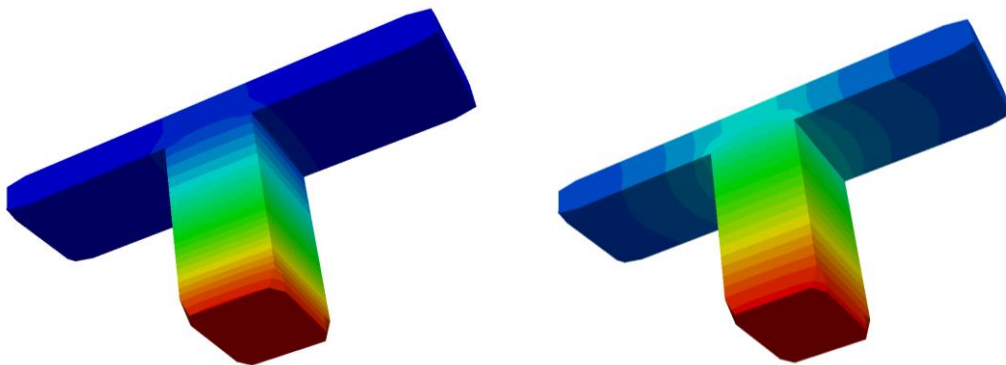
*Note: foamMultiRun can also be run on several processors.*

### 3. Post-processing

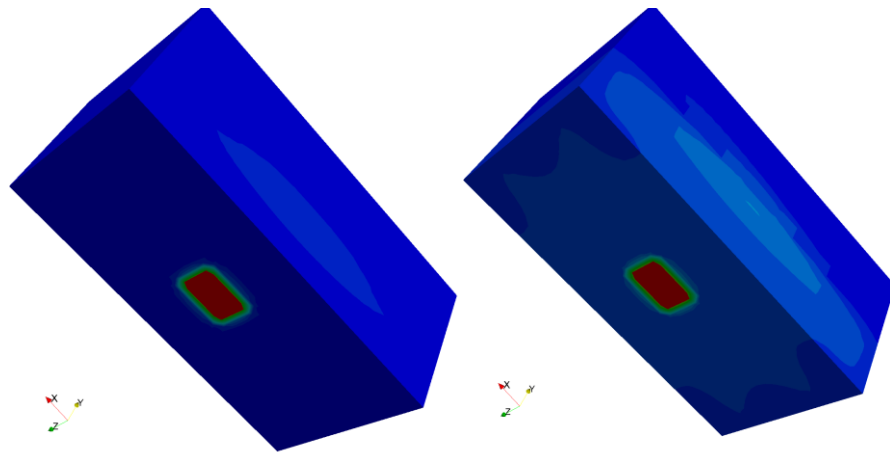
The results need to be converted to VTK files for each region with flag *-region*.

```
>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

