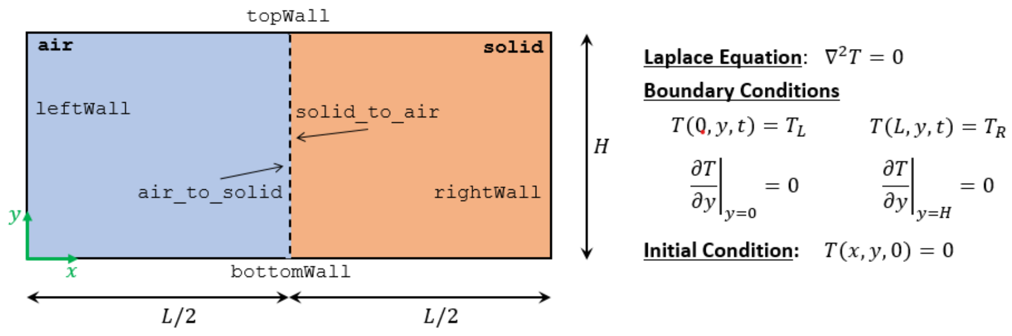
OpenFOAM Tutorials

3 Zones

3.1 In this step we will model the heat transfer between two separate regions, the air region, and a solid plate. In both regions the Laplace equation must be solved. In the following figure the problem parameters, the geometry, the initial and the boundary conditions can be seen. The previously created `laplaceFoam` solver can be modified to the needs of this problem. The new solver will be named `laplaceMultiRegionFoam`. The parameters of the problem are $L = 2\ m$, $H = 1\ m$, $T_L = 350\ K$ and $T_R = 300\ K$.


topWall
air
solid
leftWall
solid_to_air
air_to_solid
rightWall
bottomWall
H
y
x
L/2
L/2
Laplace Equation: $\nabla^2 T = 0$
Boundary Conditions
$T(0,y,t) = T_L$
$T(L,y,t) = T_R$
$\left.\frac{\partial T}{\partial y}\right|_{y=0} = 0$
$\left.\frac{\partial T}{\partial y}\right|_{y=H} = 0$
Initial Condition: $T(x,y,0) = 0$

To solve the Laplace equation two regions, two zones must be created (there are two ways to achieve that), one for each region. The first region will be named solid and the other one air, and in each region two separate Laplace equations will be solved, in two different meshes. In the interface between those two regions, the boundary condition must be suitable defined so that the two solutions of the temperature (one for the solid and one for the air) are coupled. More in particular, the value and the normal derivative of the temperature in the interface must be equal

$$T_{air} = T_{solid}$$

$$k_{air}\left.\frac{\partial T}{\partial n}\right|_{air} = k_{solid}\left.\frac{\partial T}{\partial n}\right|_{solid}$$

where k is the thermal conductivity of each region.

In the following steps, two different ways of defining multiple regions will be described, which can be found in the `case1` and `case2` folders respectively.

3.2 The first method of creating multiple regions is by using the `topoSet` tool, which can be found in `./tutorials/zones/case1/` directory. In the `blockMeshDict` a rectangular mesh block is defined, in the size of both the air and solid regions. In the `topoSetDict` the user can choose with the `boxToCell` command which cells of the domain will belong to the solid region, while the rest will be the air. In this example, the name of the zone is 'solid', which contains one `cellSet` called 'solidSet'. The `boxToCell` can be larger than the domain, as it recognizes the cell centers that belong inside the box.

So after running the `blockMesh` command to define the geometry, the `topoSet` command defines the zones.

The next step is to split the domain into all the defined regions, two in this case, one region will be the solid plate, and the remaining cells will be the air. The command is

```
$ splitMeshRegions -cellZones -overwrite
```

Before running the `splitMeshRegions` command, the directory `./0/` must be renamed (for example ./0.org/) in order to avoid any boundary conditions errors. After the successful execution of the command, the directory can be renamed in its original name.

After running these commands, two regions will be created, the 'solid' region, and a region called 'region1', which is the remaining cells. The default region that is used by OpenFoam, that corresponds to the `polyMesh` is called 'region0', so some changes must be made in the `./constant/` and `./system/` directories. In the `./constant/` directory three folders exist:

– the `polyMesh` folder that corresponds to the original mesh before the command `splitMeshRegions`

- the `solid` folder that corresponds to the solid region, as defined after the `splitMeshRegions` command
- the `region1` folder which contains the remaining cells, which correspond to the air.

As the original `polyMesh` folder is not necessary anymore, it must be removed, and in its place the `./constant/region1/polyMesh/` directory must be placed. Moreover various changes must me made in the `./constant/polyMesh/boundary` and `./constant/solid/polyMesh/boundary` files, so that the 'region1' entries are replaced with 'region0'.

As for the boundary conditions, two new boundaries will be created after the `splitMeshRegions` command, the `solid_to_region0` and the `region0_to_solid` boundaries, which correspond to the solid region and air region (region0) respectively. The interface boundary condition is the following

```
region0_to_solid
{
   type          compressible::turbulentTemperatureCoupledBaffleMixed;
   value         $internalField;
   Tnbr          Tsolid;
   kappaMethod   lookup;
   kappa         kappaA;
}
```

where `Tsolid` is the temperature field in the solid region, and `kappaA` is a `scalarField` whose value is equal to the thermal conductivity of the air in every cell. In a similar way the `solid_to_region0` boundary condition can be created.

3.3 As for the solver, it is necessary to define two separate meshes, one is the default `mesh` and one is the `meshS`, defined in the `createMeshS.H` file. This file must be included in the main program. In the `createFields.H` and `createFieldsSolid.H` all the necessary fields and variables are defined. Each field is assigned to either `mesh` or `meshS`, according to whether the variable corresponds to the solid region or the air region. As it can be seen in the main program, there are two different laplace equations being solved, one for the air region and one for the solid region.

Moreover, in order for the `compressible::turbulentTemperatureCoupledBaffleMixed` boundary condition to work, extra OpenFOAM libraries must be defined in the `./Make/options` file.

3.4 Now, after studying the structure of the case and solver, you can run the simulation. Before doing that, there are some necessary actions that must be taken. First of all, to define the appropriate schemes and solution algorithms in the `fvSchemes` and `fvSolution` files. Note that in the `./system/` directory the schemes and solution algorithms of the region0 must me defined, and in the `./system/solid` directory the schemes and solution algorithms of the solid region. For convenience, you can define in a single `fvSchemes` and `fvSolution` file all necessary entries for both regions, and copy them in both the `./system/` and `./system/solid` directories.

Next, the flow and fluid parameters must be defined in the `physicalProperties` and `regionProperties` files. Lastly, you must be sure that the correct boundary conditions are implemented in the `./0/` and `./0/solid` directories.

Due to the complexity and amount of all the above commands, it is convenient to create a single terminal executable bash script that contains all the previous commands that built the case, and run only this script instead. As it can be seen in the example case, there are three bash scripts available:

- `Allmesh`: This script runs the `blockMesh`, topoSet and `splitMeshRegions` commands. Also it converts the region1 into the polyMesh, and it copies the `fvSchemes` and `fvSolution` files in all region folders. Also it temporarily renames the `./0/` directory, in order to avoid any errors. According to the application, the user can add or remove any other commands that are necessary to successfully built the case. It also created a `.log` file with the mesh size and parameters, using the `checkMesh` command.
- `Allclean`: This script cleans and deletes all the created files by the `Allmesh`.
- `run`: This file runs the declared solver in the `controlDict` file.

To run any of the bash scripts, just run the command (e.g. for the `Allmesh` script)

```
$ ./Allmesh
```

Tip: Make sure that your system recognizes the bash scripts as executables.

3.5 The second method of creating multiple regions, is to explicitly declare them from the `blockMeshDict` file, instead of using the `topoSet` utility. You can create multiple blocks, and declare that the particular block is a part of the desired zone. This not only simplifies the constructing of the case by having to modify less files whenever one wishes to change the geometry, but also has quicker and less prone to build errors. Moreover, the way that the `topoSet` tool works is that it chooses which cells will be part of a specific zone, by whether the coordinates of their centers are within the desired region (box, cylinder, circle, etc.). This means that according to the size of the cells, the boundaries of the region might be slightly different, ehich may result to substantial errors.

The only drawback with this method is that the block as a whole must be a part of the zone, and not a part of it, like with the `topoSet` tool. This sometimes might result in complicating the construction of the `blockMeshDict` file.

The `./tutorials/zones/case2/` directory contains the test case, along with the corresponding solver. It is of similar logic, with the difference that here there is no `region0` or `polyMesh` folder, but there are two blocks, one for the air region and one for the solid region. The declaration of the zone can be seen below in the `blockMeshDict` file.

```
vertices
(
   (0 0 0)     // point 0
   (1 0 0)     // point 1
   (1 1 0)     // point 2
   (0 1 0)     // point 3
   (0 0 0.1)   // point 4
   (1 0 0.1)   // point 5
   (1 1 0.1)   // point 6
   (0 1 0.1)   // point 7
   (2 0 0)     // point 8
   (2 1 0)     // point 9
   (2 0 0.1)   // point 10
   (2 1 0.1)   // point 11
);

blocks
(
   hex (0 1 2 3 4 5 6 7) air (20 20 1) simpleGrading (1 1 1)
   hex (1 8 9 2 5 10 11 6) solid (20 20 1) simpleGrading (1 1 1)
);
```

After executing the `splitMeshRegions` command

```
$ splitMeshRegions -cellZones -overwrite
```

the desired regions will have been created. If we wanted the `region0` to exist, then can just delete the 'air' word from the block definition.

Now that there is no `region0` or `polyMesh`, the rest of the files of the case, must be modified accordingly. Note that this way there is no need to temporarily rename the `./0/` directory.

The solver must be modified as well, because we need to define two extra meshes, `meshA` for the air region, and `meshS` for the solid region. The corresponding `createMesA.H` and `createMesS.H` must be included in the main program as well.

3.6 Run the simulations, and verify that the two methods, `case1` and `case2` produce identical results.