Conjugate heat transfer in OpenFOAM

The solver we focus on here is called $chtMultiRegionFoam$ which is designed to handle heat transfer between solids and fluids. The code:

- Is transient
- Is a combination of heatConductionFoam and buoyantFoam
- Handles different regions as it's name suggests
- Solves for compressible Navier-Stokes equation
- Uses PISO-algorithm for fluid and solid regions
- Takes gravitational effects into account
- Can switch between laminar and turbulent flow with a range of choices for turbulence models
- Solves for each region separately and uses each regions data as boundary condition for the other

Main source files and directories

The source files and directories are located in

$$\$FOAM\_APP/solvers/heatTransfer/chtMultiRegionFoam$$

and the most important ones among them are as follows:

- $chtMultiRegionFoam.C$
- $coupleManager/$
- $derivedFvPatchFields/$
- $regionProperties/$
- $fluid/$
- $solid/$

We will now briefly go through the most important ones

Main source files and directories

- $chtMultiRegionFoam.C$ is the main source file which calls suitabe solvers for each region
- $regionProperties$ reads the fluid and solid region names
- The equations for momentum, enthalpy, pressure difference and continuity will be solved for fluid region by source files located in the $fluid/$ directory
- $solid/$ directory includes source files for solving heat conduction equation in solid region
- The connection between the regions is done by the source files located in $coupleManager$ directory
- Heat flux and temperature should be continuous at the region boundaries. In $derivedFvPatchFields/$ directory new boundary conditions for the coupling between solid and fluid domains are implemented

Setting up and running a case

- Generate a grid
- Specify different regions and set different properties for each region
- Set the necessary boundary and initial conditions
- Select the properties, solvers and schemes
- Run the case

Let us now begin by copying our test case and navigating to it throug typing:

$$cp\ -r\ \$FOAM\_TUT/chtMultiRegionFoam/multiRegionHeater\ \$FOAM\_RUN/$$

$$cd\ \$FOAM\_RUN/multiRegionHeater$$

Grid generation and assigning different regions

- Start by generating a 7x2x2 block (*constant/polyMesh/blockMeshDict*) by running *blockMesh* command
- We should now add solid regions into the domain and then specify the rest of the domain as the fluid region. This can be done by running *setSets* command for a batch of commands which are located in *makeCellSets.setSet* dictionary
- Next we convert all previously defined sets to respective zones through the command *setsToZones*
- We finish preparation of the domain by splitting the mesh into different zones through running the command *splitMeshRegions*
- The dictionary *constant/regionProperties* assigns fluid and solid regions

Boundary and initial conditions

- As usual the boundary and initial conditions for the whole domain (both the solid and fluid variables) are first specified in the initial time directory (here $0/$)
- In order to assign respective conditions on the interfaces of different regions, one should use $system/region\_name/changeDictionaryDict$ files, which replace the conditions for each region as needed
- Changing of initial and boundary conditions for each region takes place via running $changeDictionary$ for each region name
- The $changeDictionaryDict$ files for each region include the quantities which are used in the computations for the respective region, plus their relations to the corresponding quantities in the neighboring regions (see $system/Air/changeDictionaryDict$ for example)
- This leads to having initial and boundary conditions which are adapted for all regions

Heat flux and temperature boundary conditions

There are three different choices for the interface boundary conditions to select from:

- $solidWallTemperatureCoupled$: Fixed value boundary condition for temperature
- $solidWallHeatFluxTemperature$: Heat flux boundary condition for temperature on solid region
- $solidWallHeatFluxTemperatureCoupled$: Fixed heat-flux boundary condition for temperature

Example usage for *solidWallTemperatureCoupled*

```
myInterfacePatchName
{
    type                    solidWallTemperatureCoupled;
    neighbourRegionName     fluid;
    neighbourPatchName      fluidSolidInterface;
    neighbourFieldName      T;
    K                       K;
    value                   uniform 300;
}
```

Example usage for $solidWallHeatFluxTemperature$

$myInterfacePatchName$

{

$type$	$solidWallHeatFluxTemperature;$	
K	$K;$	$< Name\ of\ K\ field >$
q	$uniform\ 1000;$	$< Heat\ flux\ [W/m2] >$
$value$	$300.0;$	$< Initialtemperature\ [K] >$

}

Example usage for $solidWallHeatFluxTemperatureCoupled$

$myWallPatchName$

{

$type$	$solidWallHeatFluxTemperatureCoupled;$
$neighbourRegionName$	$fluid;$
$neighbourPatchName$	$fluidSolidInterface;$
$neighbourFieldName$	$T;$
K	$K;$
$value$	$uniform\ 300;$

}

Properties, solvers and schemes (1/4)

- Having set up the boundary conditions, it is now time to select the right solver and schemes
- In the dictionary $constant/fluid\_region\_name/RASProperties$ one can select turbulent solver to be switched on, select appropriate turbulence model and eventually modify the coefficients used in the models
- The dictionary $constant/fluid\_region\_name/environmentalProperties$ defines different components of gravitational acceleration, g.

Properties, solvers and schemes (2/4)

The dictionary $constant/fluid\_region\_name/thermophysicalProperties$ stores thermophysical models to construct a $p-T$ system from which other properties are computed.

- Available basic thermo-physical properties are:
  - $hConstThermo$: constant C_p with evaluation of h and s
  - $janafThermo$: C_p evaluated with a function of $JANAF$ thermodynamic tables, from which h and s are evaluated
- Available transport properties
  - $constTransport$: constant transport properties
  - $sutherlandTransport$: temperature-dependent properties ($Sutherland$'s formula)
- Available choices:
  - $hThermo<pureMixture<sutherlandTransport<specieThermo<janafThermo<perfectGas>>>>>$
  - $hThermo<pureMixture<sutherlandTransport<specieThermo<hConstThermo<perfectGas>>>>>$
  - $hThermo<pureMixture<constTransport<specieThermo<hConstThermo<perfectGas>>>>>$

Properties, solvers and schemes (3/4)

- $stringname$: <mixture name>
- $specieCoeffs$: <number of moles, molecular weight>
- $thermoCoeffs$:
  - For $hConstThermo$: <specific heat(C_p), Heat of fusion(H_f)>
  - For $janafThermo$: <a set of $JANAF$ thermodynamic coefficients>
- $transportCoeffs$
  - For $constTransport$: <dynamic viscosity(μ), Prandtle number(P_r)>
  - For $sutherlandTransport$: <sutherland coefficient(A_s) and sutherland temperature(T_s)> (to compute temperature dependent properties μ, K and α)

Properties, solvers and schemes (4/4)

- $system/region\_name/fvSolution$ dictionary defines solver parameters for each region
- $system/region\_name/fvSchemes$ dictionary contains numerical schemes for solvers in each region
- General solution controls should be set as usual in $system/controlDict$ dictionary

Running and visualizing the results

- In order to run the whole package at once one can simply run $./Allrun$ script
- For the time being the best way to visualize the results is to softlink the mesh from the first time step to constant directory for each region:

$$cd\ constant/region\_name$$
$$ln\ -s\ ../../0.001/region\_name/polyMesh\ \ polyMesh$$

- Next you convert the format using $foamToVTK\ -region\ region\_name$ for all regions
- You can finally run $paraview$ and load the data set to visualize your results
- Note that we have a folder for every $region\_name$ in $constant/$ and $system/$, as well as $time$ directories