Adding electric conduction and Joule heating to chtMultiRegionFoam

Niklas Järvstråt

Physical equations: Thermal conduction
(with Joule heating and passive transport included)

Thermal conduction is standard in chtMultiRegionFoam

We need to add the Joule heating term:

Convective heat transfer is added in the solid regions:

$$\rho \cdot C_p \cdot \frac{dT}{dt} = \nabla(\mathrm{K}\nabla\mathrm{T}) + \sigma|\nabla \mathrm{V}|^2 - \rho \cdot C_p \cdot U \cdot \nabla T$$

Physical equations: Electrical conduction (new)

$$-\nabla(\sigma\nabla V)=0$$

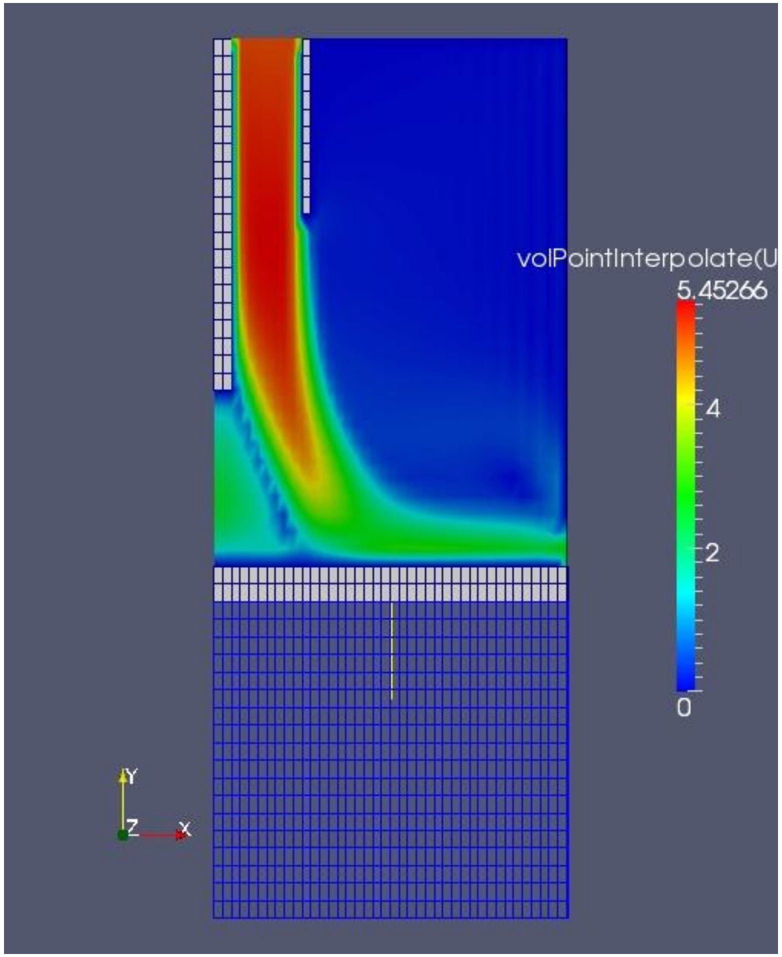
Physical equations: Fluid flow and momentum (old)

Boyancy driven flow

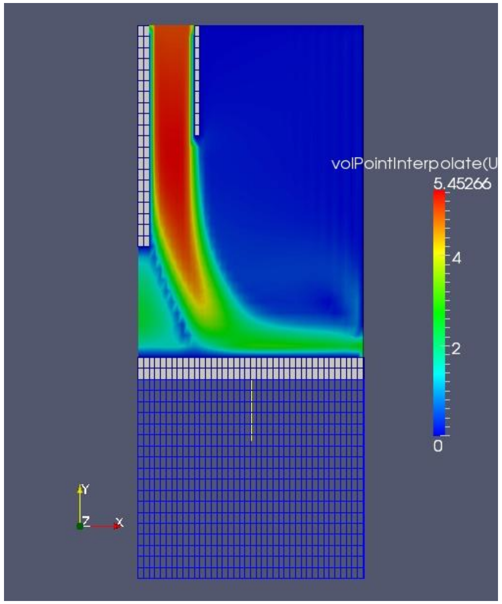
Overview and aim

1. Create a simple test model with appropriate boundary conditions for testing the changes.
2. Add field variables, electric potential scalar as a variable and include the velocity field also in the solid regions, although the velocity will not be solved for there. Add the material property conductivity as a field variable.
3. Add solving for electric potential in both solid and fluid solvers.
4. Add Joule heating term to both solid and fluid heat conduction solvers.
5. Add passive transport term to solid heat conduction solver.
6. Adapt internal boundary patches to also cover the electric field.

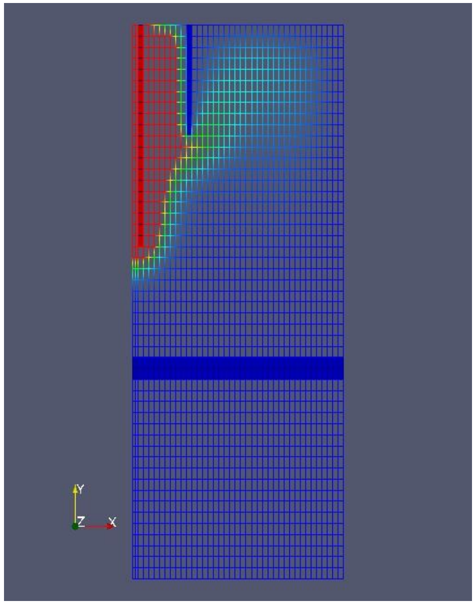
Mesh



Problems with wedge patch



Not OK



OK

- Apparently, one of the utilities setSet, setsToZones or splitMeshRegions does not allow two wedge patches in the radial direction
- Using cyclic instead of wedge bypasses the problem

Adding fields to solver

- Add electric potential (V) & electrical conductivity (sigma) to solid and fluid regions
- Add velocity U to solid regions.

createSolidFields.H/ createFluidFields.H:

```
PtrList<volScalarField> VSolid(solidRegions.size());

Info<< "    Adding to VSolid\n" << endl;
VSolid.set
(
i,
new volScalarField
(
IOobject
(
"Vel",
runTime.timeName(),
solidRegions[i],
IOobject::MUST_READ,
IOobject::AUTO_WRITE
),
solidRegions[i]
)
);
```

setRegionSolidFields.H/ setRegionFluidFields.H:

```
volScalarField& Vel = VSolid[i];
```

Wirefeed

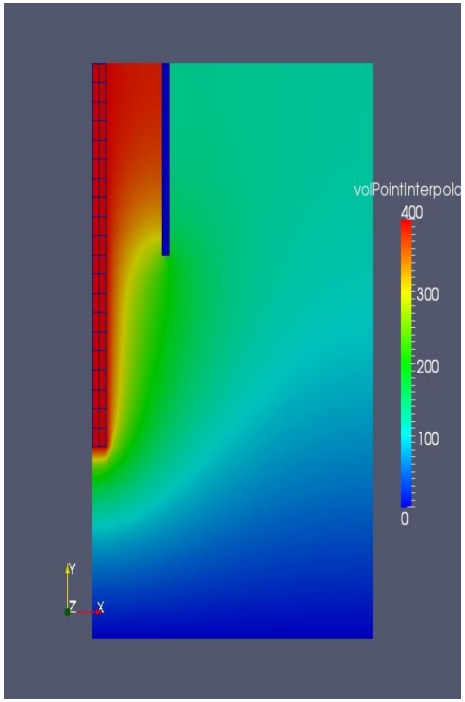
```
(
fvm::ddt(rho*cp, T)
- fvm::laplacian(K, T)
// Added for JouleMultiRegionFoam
==rho*cp* (U & fvc::grad(T))
);
```

Electric conduction: solver

Add solving electric conduction in solveSolid.H and solveFluid.H
Implemented in both cases as the same separate file VEqn.H,

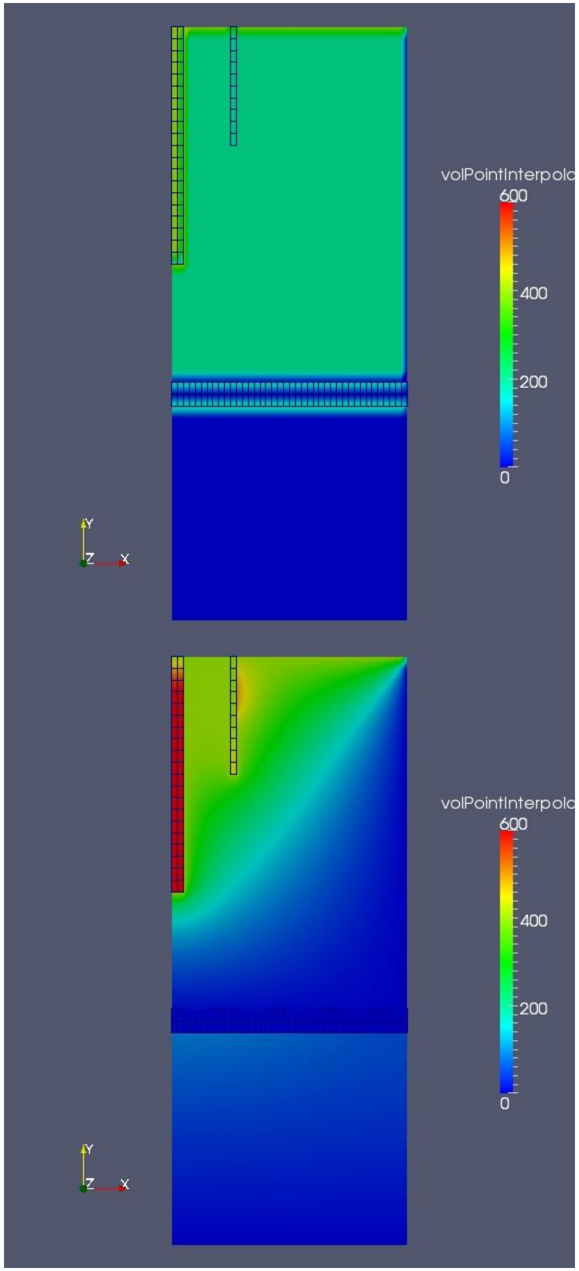
```
{
for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
{
solve
(
fvm::laplacian(sigma, Vel)
);
}
Info<< "Min/max V:" << min(Vel) << ' '
<< max(Vel) << endl;

}
```



Electric conduction: interface boundary patch

- Patch: solidWallMixedTemperatureCoupledFvPatchScalarField works by assigning the temperature on one side of the patch as boundary condition for the other side, and the heat flux from the other side as boundary condition for the first side.
- should be generic enough to use for other variables, but…
- **It doesn’t converge!**
- Why?
- The temperature is explicitly transferred to the patch, while the heat flux seems to be collected implicitly from the set of thermal variables?
- The patch can be used for any one of the fields, but not for both?
- Solution:
- either generalize the patch so that the same patch can be used for both fields, or write a separate patch for the electric field, using the thermal patch as a template.



Joule heating

Add Joule heating term in solveSolids.H and hEqn.H:

```
(
fvm::ddt(rho*cp, T)
- fvm::laplacian(K, T)
== (U & fvc::grad(T))
// Added for JouleMultiRegionFoam
+ sigma*(fvc::grad(Vel) ) & fvc::grad(Vel))
);
```

How-to “lazydog” for using JouleMultiRegionFoam

- Mesh generation
- Assign region properties and solution
  - Allrun
  - Fluid property folders
- Fields and boundary conditions
- Solution control

Mesh generation

- Mesh generation is done as usual, except that cellSets must be defined.
- In the chtMultiRegionFoam tutorial, the overall calculation region(solid regions + fluid regions), is defined in the file constant/polyMesh/blockMeshDict.
- The sets are then defined using the utility makeCellSets, as defined in the file makeCellSets.setSet.

- A drawback of this solution is that it is necessary to adjust element sizes so that boundaries between regions fall exactly on cell boundaries as otherwise the regions will not have the desired size. This is because makeCellSets selects cells with centres within the specified box and assigns them to the Set. Nevertheless, it is a simple and rather straightforward way of defining a not too complicated block mesh consisting of several regions.

Assign region properties and solution

Allrun

chtMultiRegionFoam:

```
# remove fluid fields from solid regions (important for post-processing)
for i in heater leftSolid rightSolid
do
   rm -f 0*/$i/{epsilon,k,p,U}
done

# remove solid fields from fluid regions (important for post-processing)
for i in bottomAir topAir
do
   rm -f 0*/$i/{cp,K}
done
```

JouleMultiRegionFoam:

```
# remove fluid fields from solid regions (important for post-processing)
for i in wire nozzle plate
do
   rm -f 0*/$i/{epsilon,k,p}
done

# remove solid fields from fluid regions (important for post-processing)
for i in bottomAir topAir
do
   rm -f 0*/$i/{cp,K}
Done
```

Fluid property folders

For each fluid region, create a folder named as that region to the constant folder.

Copy files g, RASProperties, thermophysicalproperties and turbulenceProperties into this folder

Fields and boundary conditions

- BC’s are set in system/*[regionName]*/changeDictionaryDict and 0 /*[regionName]* for
  - external boundaries
  - but also for the interface boundaries between the regions

- The interface boundaries have been defined by splitMeshRegions :
  - May be treated as external boundaries
  - The patch solidWallMixedTemperatureCoupled preservies the continuity of the solution
  - (Note that "reverse" boundary conditions must also be specified)

T (independent variable): zeroGradient, fixedValue, InletOutlet, solidWallMixedTemperatureCoupled

U (independent variable): zeroGradient, fixedValue, InletOutlet

p (independent variable): zeroGradient, fixedValue, InletOutlet

epsilon, k (fluid material constants): zeroGradient for simplicity

cp, rho, K (solid material constants): zeroGradient for simplicity.

Vel (electric potential, new independent variable): zeroGradient, fixedValue, InletOutlet(?), solidWallMixedTemperatureCoupled

sigma: (Electric conductivity field, material constant): zeroGradient for simplicity

Solution control

startTime should be set to the same value as deltaT in system/controlDict

(the first timestep folder is used to setup boundary and initial conditions for the regions)

For all regions, the solution scheme for the electric field must be defined:
In system/regionName/fvSchemes:

```
// added electrical continuity eqn for JouleMultiRegionFoam
    laplacian(sigma,Vel) Gauss linear corrected;
```

In system/regionName/fvSolution:

```
Vel
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-14;
        relTol          0;
    }
```

Next steps

1. Improve the simple test model and establish a set of test cases with appropriate boundary conditions for testing the performance of the model.
2. The velocity vector does not appear in output files for solid regions, although it does influence calculation of the transport term in the heat equation.
3. The electric potential is solved in both solid and fluid regions, though the performance has not been tested and integration and convergence criteria should be properly defined.
4. The Joule heating term, although implemented in both solid and fluid heat conduction solvers does not seem to yield correct results. Improving the test case, ad.
5. The passive transport term in the solid heat conduction solver seems to work correctly, but needs verification against exact values for a simple test case.
6. The internal boundary patch does not seem to work properly. Investigate whether this is due to incorrect usage, that it is not general enough to be used or that it is not possible to use it for two different fields at the same time.