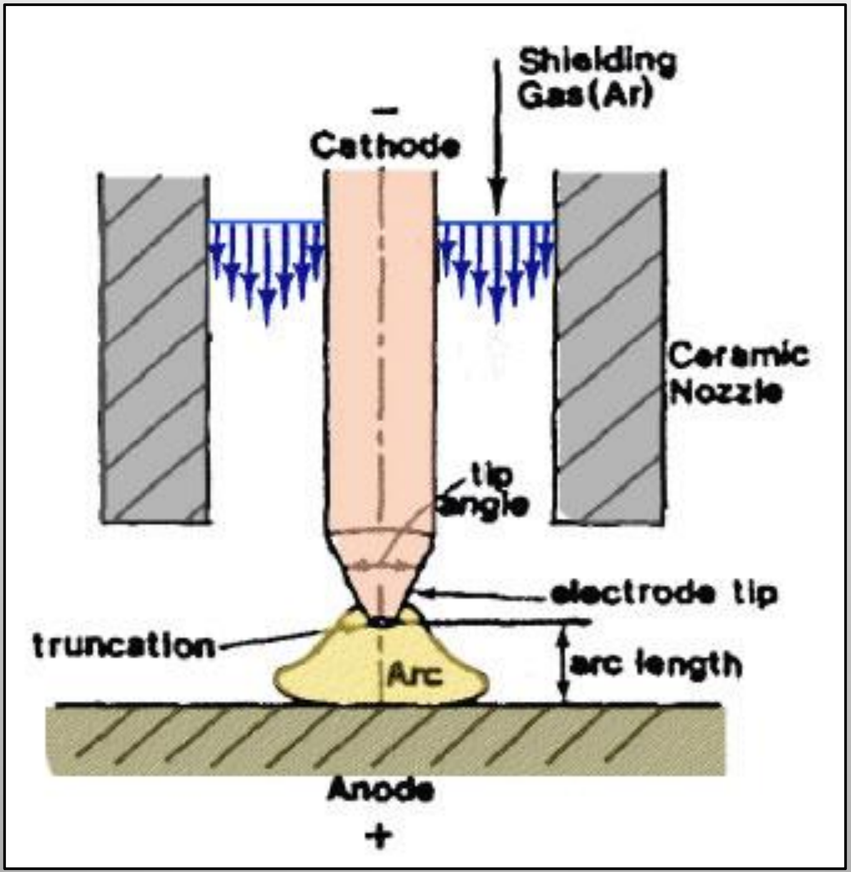
CFD with OpenSource software, project

Implementing “chtMultiRegionFoam” Solver for Electric Welding

Alireza Javidi

October 2010

Introduction



schematic sketch of electric welding

chtMultiRegionFoam

The solver is based on combination of *heatConductionFoam* and *buoyantFoam* for conjugate heat transfer between a solid region and fluid region

chtMultiRegionFoam

OpenFoam – 1.5

- ***solidWallTemperatureCoupled***

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

chtMultiRegionFoam

OpenFoam – 1.5

• ***solidWallHeatFluxTemperature***

```
myInterfacePatchName
{
    type      solidWallHeatFluxTemperature;
    K         K;                  < Name of K field >
    q         uniform 1000;       < Heat flux [W/m2] >
    value     300.0;              < Initialtemperature [K] >
}
```

chtMultiRegionFoam

OpenFoam – 1.5

- ***solidWallHeatFluxTemperatureCoupled***

```
myWallPatchName
{
    type                    solidWallHeatFluxTemperatureCoupled;
    neighbourRegionName     fluid;
    neighbourPatchName      fluidSolidInterface;
    neighbourFieldName      T;
    K                       K;
    value                   uniform 300;
}
```

chtMultiRegionFoam

OpenFoam – 1.6

- ***solidWallHeatFluxTemperature***

```
myInterfacePatchName
{
    type        solidWallHeatFluxTemperature;
    K           K;                < Name of K field >
    q           uniform 1000;     < Heat flux [W/m2] >
    value       300.0;            < Initialtemperature [K] >
}
```

chtMultiRegionFoam

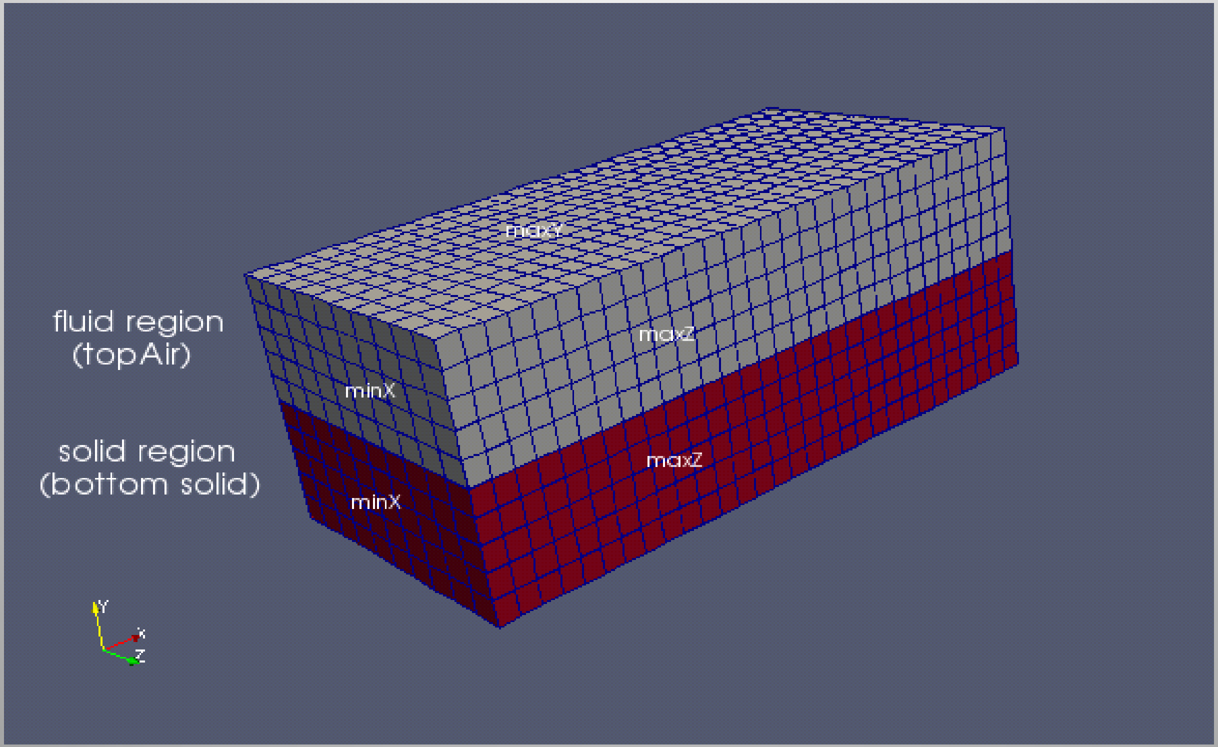
OpenFoam – 1.6

- ***solidWallMixedTemperatureCoupled***

```
myInterfarcePatchName
{
type                    solidWallMixedTemperatureCoupling;
neighbourFieldName      T;
K                       K;
value                   uniform 300;
}
```

Implementing chtMultiRegionFoam

Geometry and mesh



geometry of the test case with two different regions

- *setset – batch makecellSets.setset* (to define region sets)
- *setsToZones –noFlipMap* (to add zones to the mesh with similar sets name)
- *splitMeshRegions –cellZones –overwrite* (to split mesh into multiple regions)

Implementing chtMultiRegionFoam

Case structure

- **"0"**
  - *"topAir"*
    - *"U"*
    - *"T"*
    - *"p"*
  - *"bottomSolid"*
    - *"cp"*
    - *"K"*
    - *"T"*
    - *"rho"*
- ***"constant"***
  - *regionProperties*
  - *"topAir"*
    - *"polyMesh"*
    - *thermophysicalProperties*
    - *g*
  - *"bottomSolid"*
    - *"polyMesh"*
- ***"system"***
  - *controlDict*
  - *"topAir"*
    - *fvSystem*
    - *fvSolution*
  - *"bottomSolid"*
    - *fvSystem*
    - *fvSolution*

Implementing chtMultiRegionFoam

Boundary conditions

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;
}
// * * * * * * * * * *  * * * * * * * * * * * * *  * * * * * * * //

dimensions      [ 0 0 0 1 0 0 0 ];
internalField   uniform 300;
boundaryField
{
    maxY
    {
         type            fixedValue;
         value           uniform 100;
    }
    minX
    {
         type            zeroGradient;
         value           uniform 300;
    }
    maxX
    {
         type            zeroGradient;
         value           uniform 300;
    }
    minZ
    {
        type            zeroGradient;
        value           uniform 300;
    }
    maxZ
    {
        type            zeroGradient;
        value           uniform 300;
    }
    topAir_to_bottomSolid
    {
        type            solidWallMixedTemperatureCoupled;
        value           uniform 300;
        neighbourFieldName T;
        K               K;
    }
}
```

Implementing chtMultiRegionFoam

Boundary conditions

```
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;
}
// * * * * * * * * * * * ** * * * * * * * * * * * * * * * * //

dimensions      [ 0 0 0 1 0 0 0 ];

internalField   uniform 300;

boundaryField
{
    minX
    {
        type                zeroGradient;
        value               uniform 300;
    }
    maxX
    {
        type                zeroGradient;
        value               uniform 300;
    }
    minZ
    {
        type                zeroGradient;
        value               uniform 300;
    }
    maxZ
    {
        type                zeroGradient;
        value               uniform 300;
    }
    minY
    {

        type               fixedValue;
        value               uniform 2000;
    }
    bottomSolid_to_topAir
    {
        type                solidWallMixedTemperatureCoupled;
        value               uniform 300;
        neighbourFieldName T;
        K                   K;
    }
}


// ********************************************************* //
```

Implementing chtMultiRegionFoam

Thermal conductivity

Solid region:

K=80 , cp=450

Fluid region:

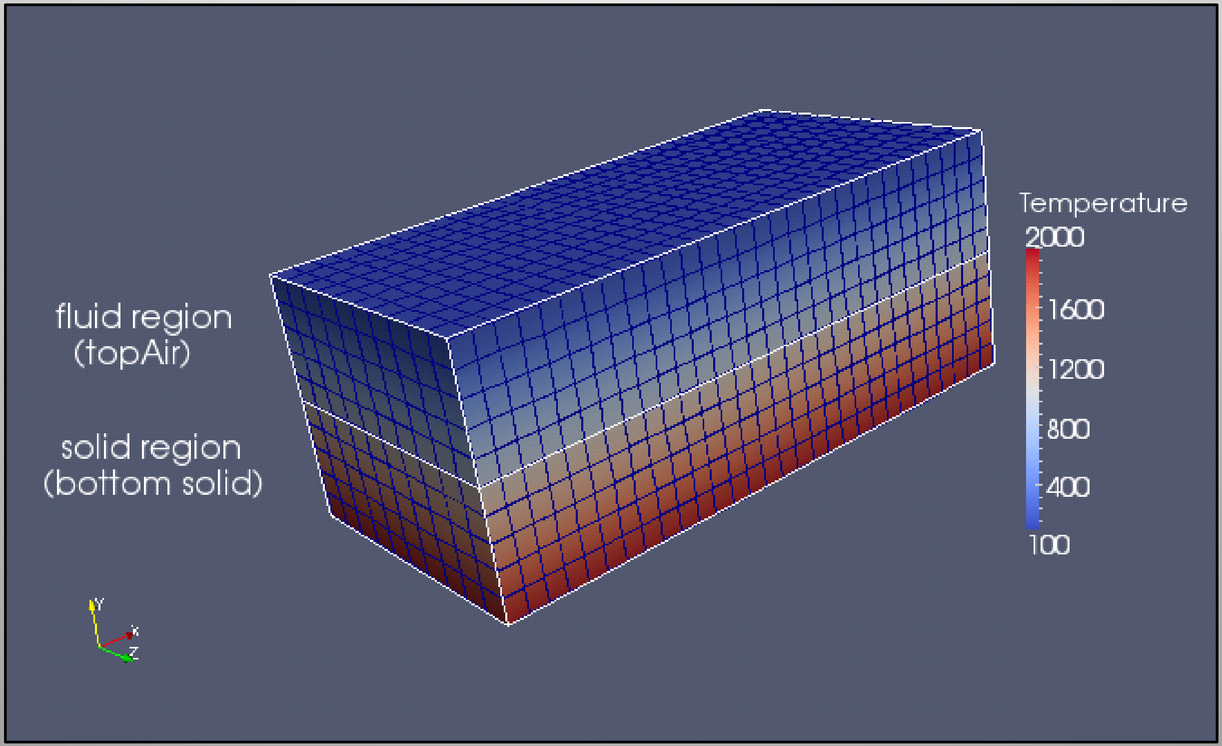
*K= cp*mu*rPr*

Mixture gasName n W cp Hf mu Pr

```
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      thermophysicalProperties;
}
// * * * * * * * * ** * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
hPsiThermo<pureMixture<constTransport<specieThermo<hConstThermo<perfectGa
s>>>>>;

mixture         air_fake 1 28.9 450 0 1.8e-02 0.10125;

// ************************************************************************* //
```

Implementing chtMultiRegionFoam

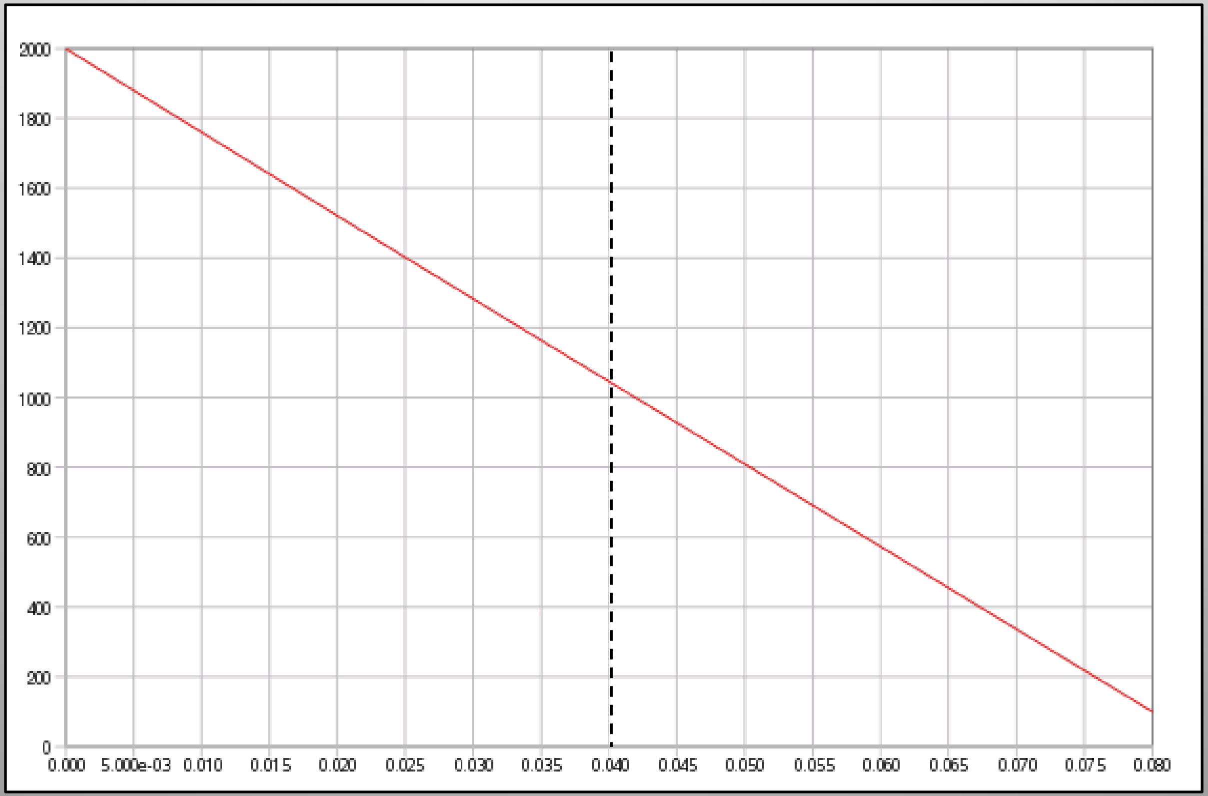
Results



temperature distribution in both solid and fluid region

Implementing chtMultiRegionFoam

Results



temperature distribution in both solid and fluid region

Adding Electric Potential equation

Maxwell equation

$$\nabla \cdot (\sigma_m \nabla \phi) = 0,$$

$$j = -\sigma_m \nabla \phi.$$

Electric conductivity

```
fvMesh& mesh = solidRegions[i];

volScalarField& rho = rhos[i];
volScalarField& sigmaMag = sigmaMags[i];
volScalarField& ElPot = ElPots[i];
volScalarField& cp = cps[i];
volScalarField& K = Ks[i];
volScalarField& T = Ts[i];
```

Adding Electric Potential equation

Electric conductivity

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


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

Adding Electric Potential equation

Electrical potential equation

```
// Solve equation for electric potential  ElPot

//Info << "  Solve the electric potential equation " << endl;

Info<< "debut VEqn.H - sigmaMag max/min : " << max(sigmaMag).value() << "
" << min(sigmaMag).value() << endl;

{
  solve
   (
       fvm::laplacian(sigmaMag, ElPot)
   );
}
```

Adding Electric Potential equation

Boundary conditions

```
\*---------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      ElPot;
}
// * * * * * * * * ** * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [1 2 -3 0 0 -1 0];

internalField   uniform 0;

boundaryField
{
  maxY
    {
    type            fixedValue;
    value        uniform -2;
    }
  minX
    {
    type            zeroGradient;
    }
  maxX
    {
    type            zeroGradient;
    }
  minZ
    {
    type            zeroGradient;
    }
  maxZ
    {
    type            zeroGradient;
    }
  topAir_to_bottomSolid
    {
        type            solidWallMixedTemperatureCoupled;
        value           uniform 0;
        neighbourFieldName ElPot;
        K               sigmaMag;
    }
}


// *********************************************************** //
```

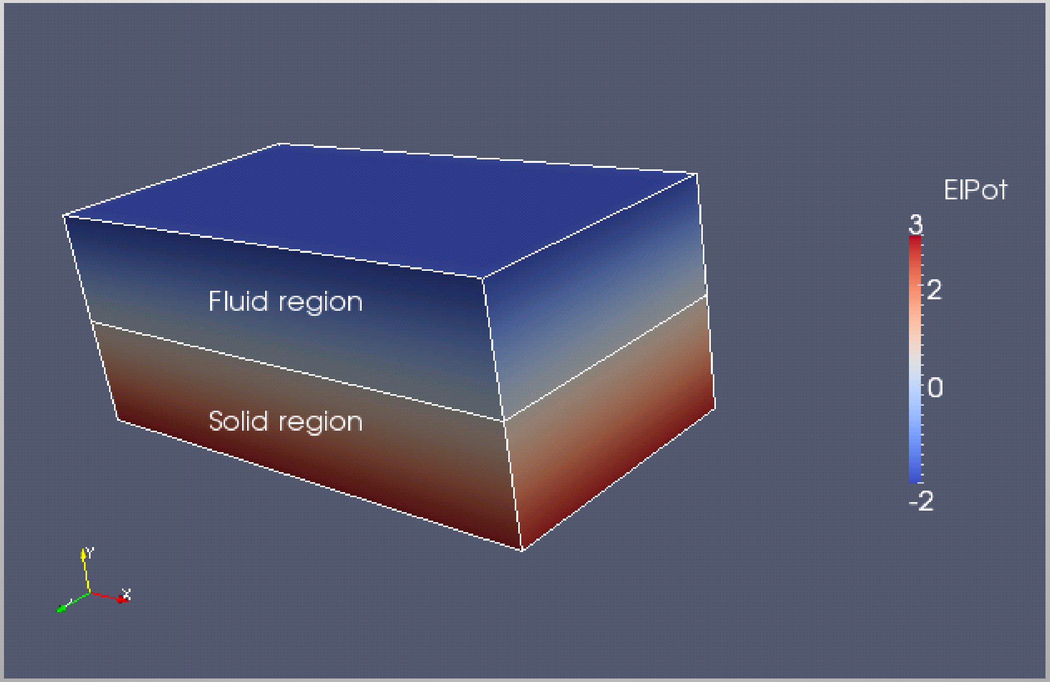
Adding Electric Potential equation

Boundary conditions

```
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      ElPot;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * //
dimensions      [1 2 -3 0 0 -1 0];

internalField   uniform 0;

boundaryField
{
  minY
    {
    type            fixedValue;
    value        uniform 3;
    }
  minX
    {
    type            zeroGradient;
    }
  maxX
    {
    type            zeroGradient;
    }
  minZ
    {
    type            zeroGradient;
    }
  maxZ
    {
    type            zeroGradient;
    }
  bottomSolid_to_topAir
    {
        type            solidWallMixedTemperatureCoupled;
        value           uniform 0;
        neighbourFieldName ElPot;
        K               sigmaMag;
    }
}


// ************************************************************* //
```

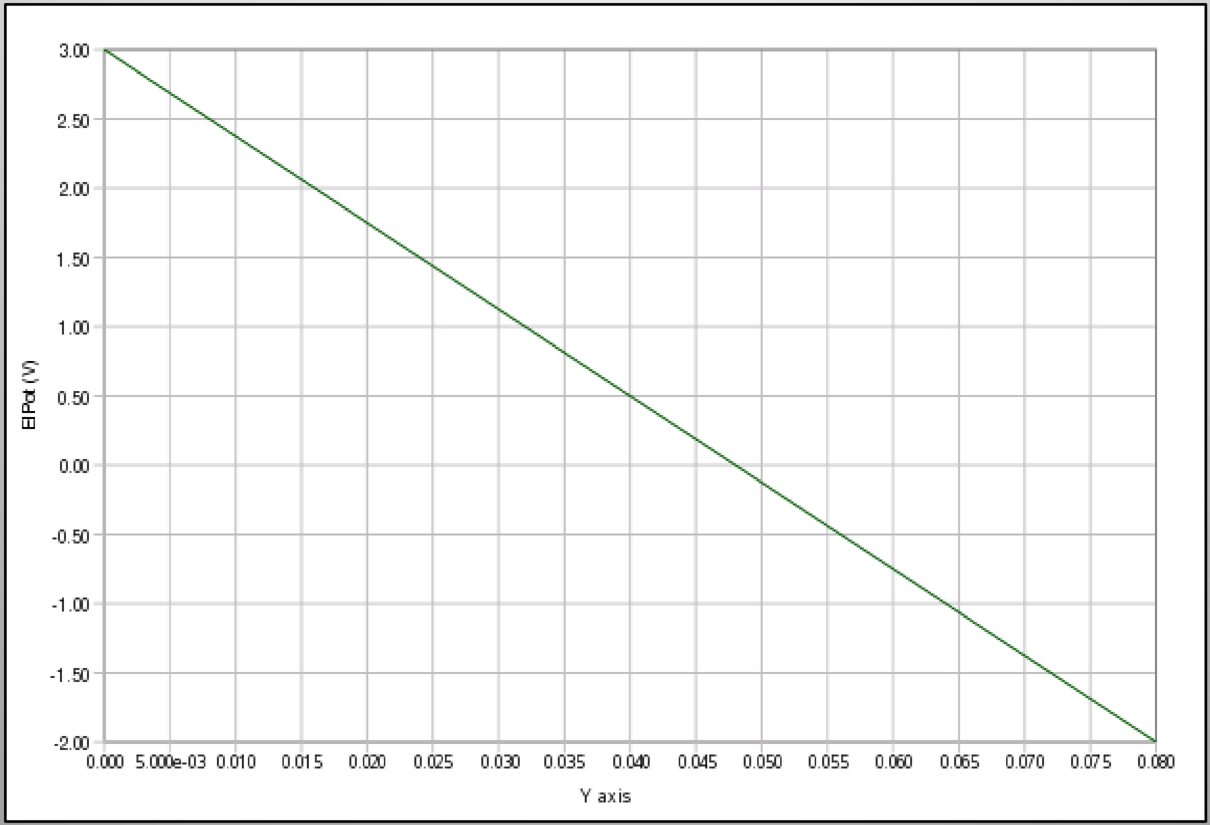
Adding Electric Potential equation

Results:



Adding Electric Potential equation

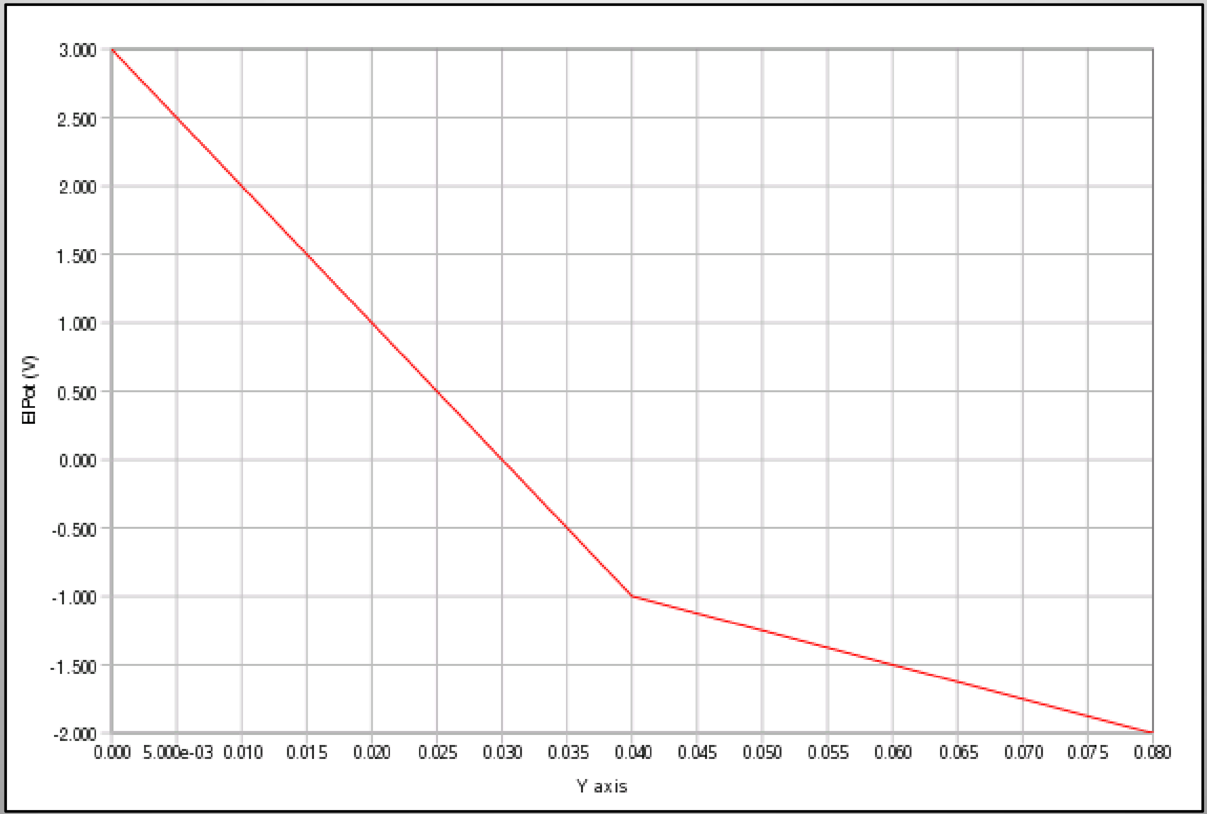
Results:



electric potential field with the same electric conductivity in fluid and solid regions

Adding Electric Potential equation

Results:



electric potential field with different electric conductivity in fluid and solid regions

Conclusions and future works

- The laplacian electric potential equation can be implemented as a coupling boundary condition between different regions in *“chtMultiRegionFoam”* solver.

- For extending the model to temperature dependent solid parameters the future work can be to implement materila quantities via the library “thermophysicalModels”, as done for the corresponding quantities in the fluid region.