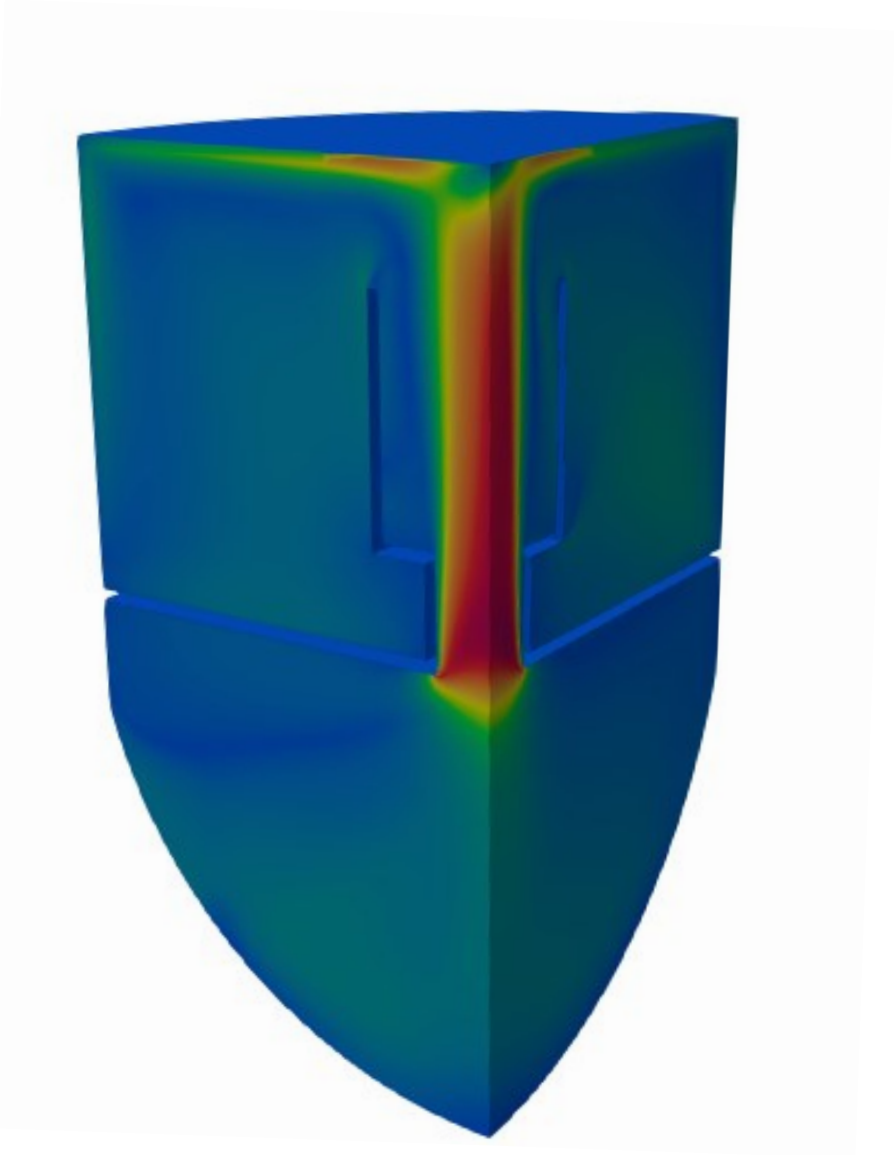


von KARMAN INSTITUTE FOR FLUID DYNAMICS

SIMULATING REACTOR THERMAL-HYDRAULICS IN OPENFOAM

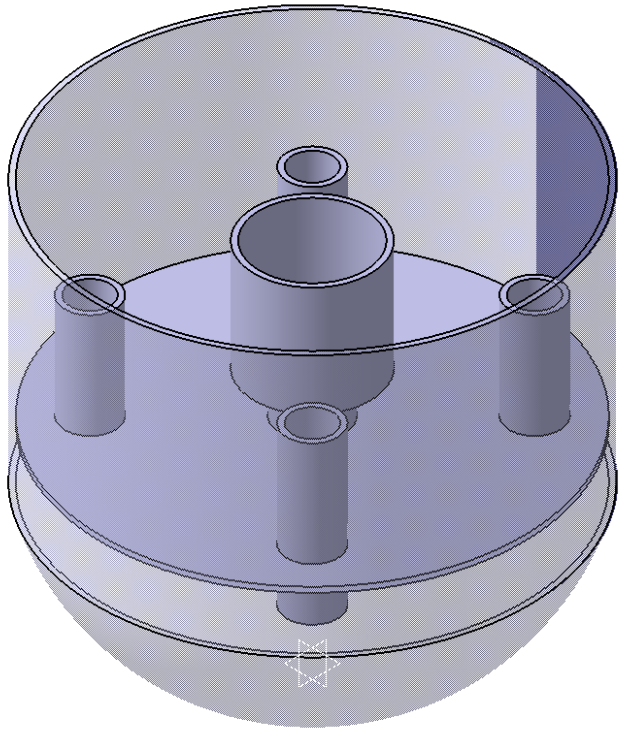
The ESPRESSO test case

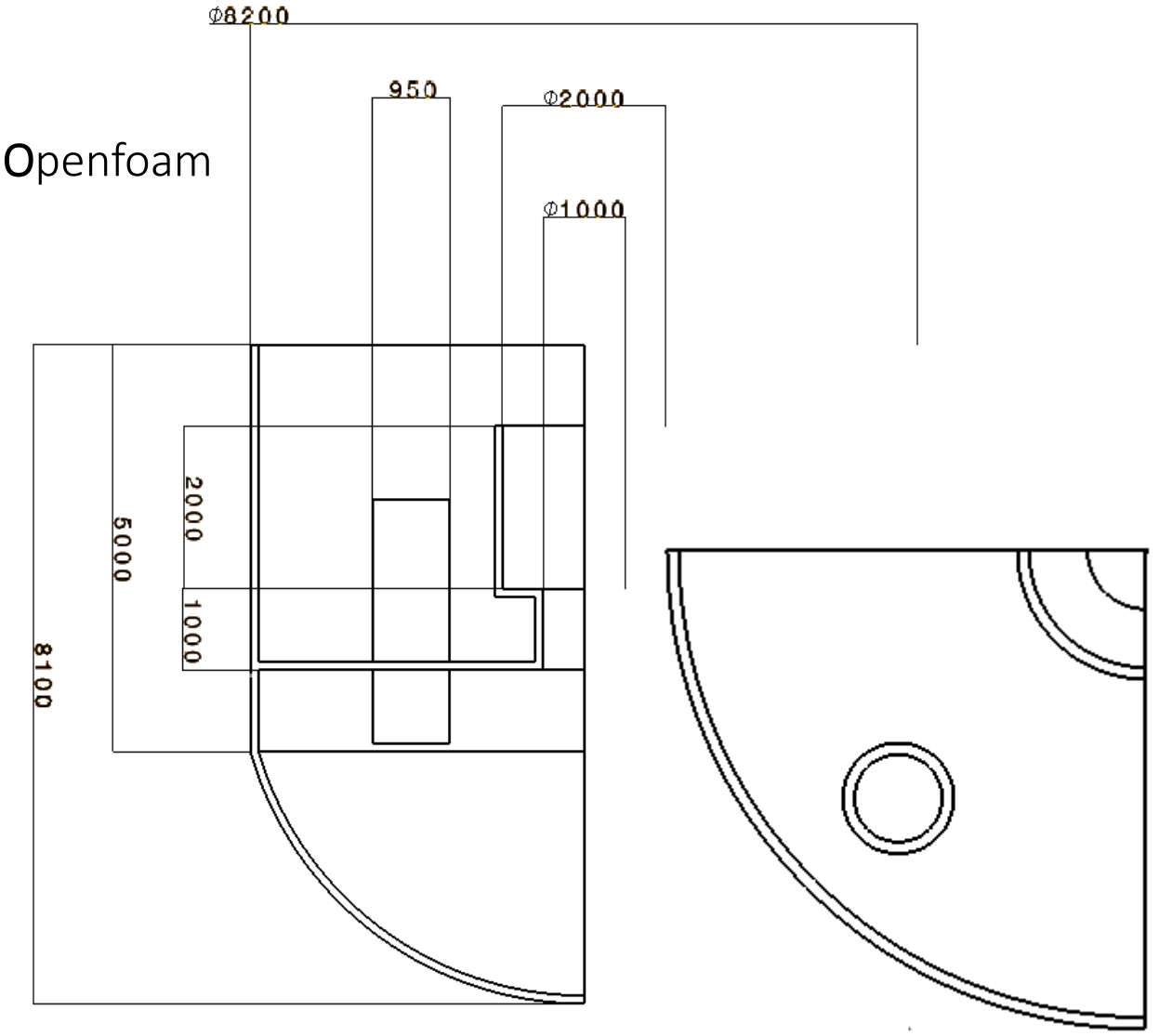
Maria Faruoli, Silvania Lopes, Lilla Koloszar, Matilde Fiore

4th July 2024

ESPRESSO

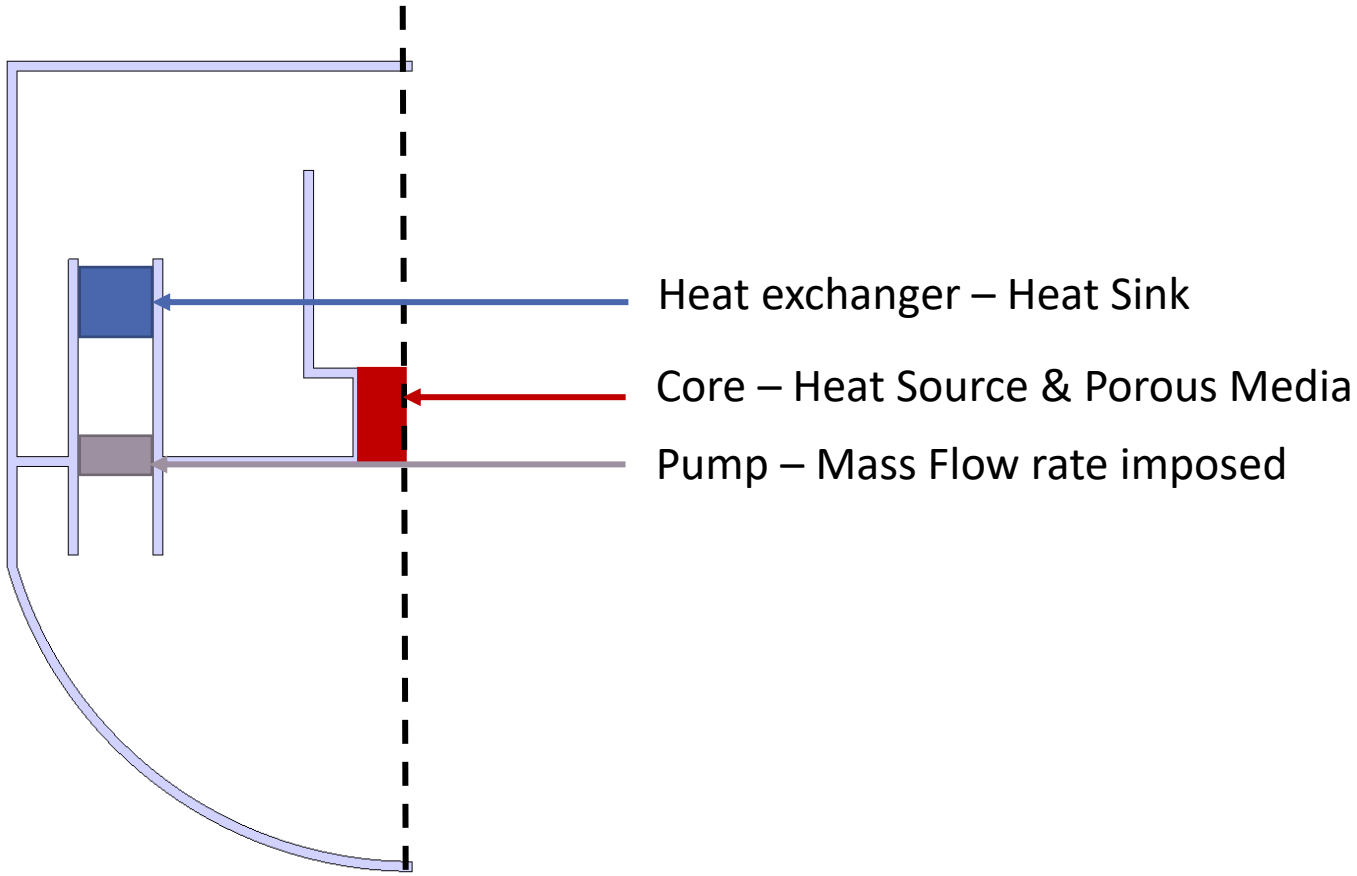
Extremely SimPlified REactor Simulation uSing Openfoam





GEOMETRY

¼ of the entire geometry is simulated, by employing the symmetry boundary conditions



Which are the basic features of the flow?

- Constant density fluid (liquid metal): **INCOMPRESSIBLE**
- Viscosity is important to correctly predict the flow: **VISCOUS**
- Gravity is considered, but assuming conditions of forced convection gravitational effects have limited influence
- Heat exchange is very important (energy generation in the core region/heat sink in the heat exchangers)
- No liquid/gas interfaces modeled: **SINGLE PHASE FLOW**
- Not interested in study the transient (nominal conditions): **STEADY state simulation**

General steps of CFD

1. Define the domain of interest
2. Discretize the domain
3. Physical and numerical setup
4. Solve the resulting problem
5. Post-process the results

General steps of CFD

1. **Define the domain of interest**
2. Discretize the domain
3. Physical and numerical setup
4. Solve the resulting problem
5. Post-process the results

PROBLEM DESCRIPTION - ESPRESSO

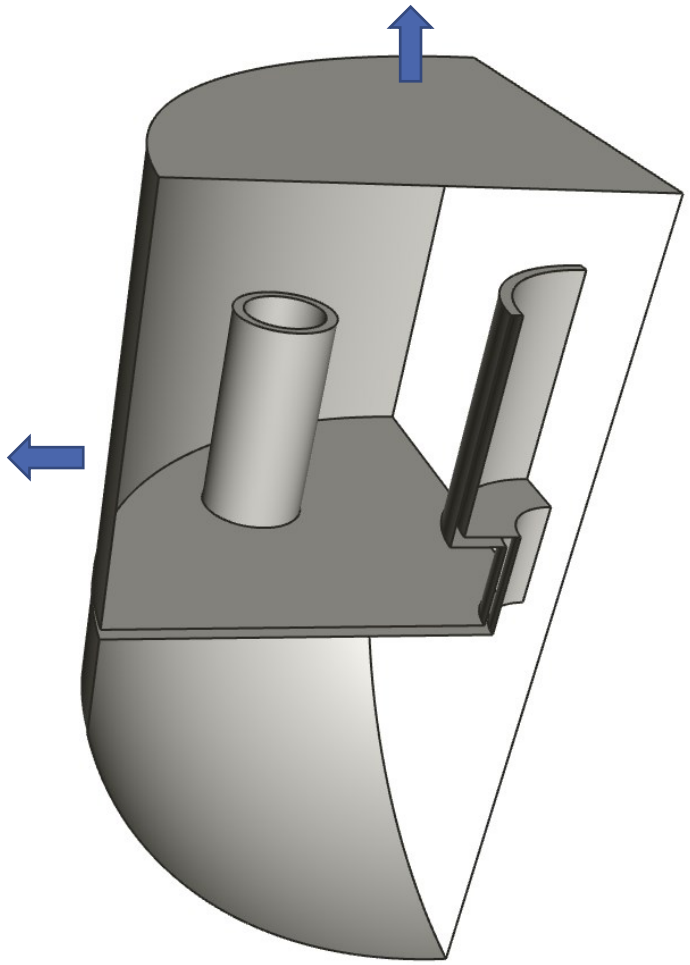
Heat transfer coefficient:

$$q = h(T - T_{out})$$
$$h = 10\ W/m^2K$$
$$T_{out} = 165°C$$

We will show you how to consider the conduction through the cover without resolving it.

Explicit wall heat flux:

$$q = -150 W/m^2$$



Pumps: effect on the fluid -> provides the massflow.

In our case prescribed as velocity magnitude in the pumps equivalent to **0.384m/s**.

HXs: effect on the fluid -> cools down the fluid after heated up by the core.

In our case constant heat sink of **10MW**, each.

Core: effect on the fluid -> heats up the fluid and strong pressure drop of dense structures.

In our case constant heat source of **10MW**.
Pressure drop modelled by Darcy-Forchheimer law.

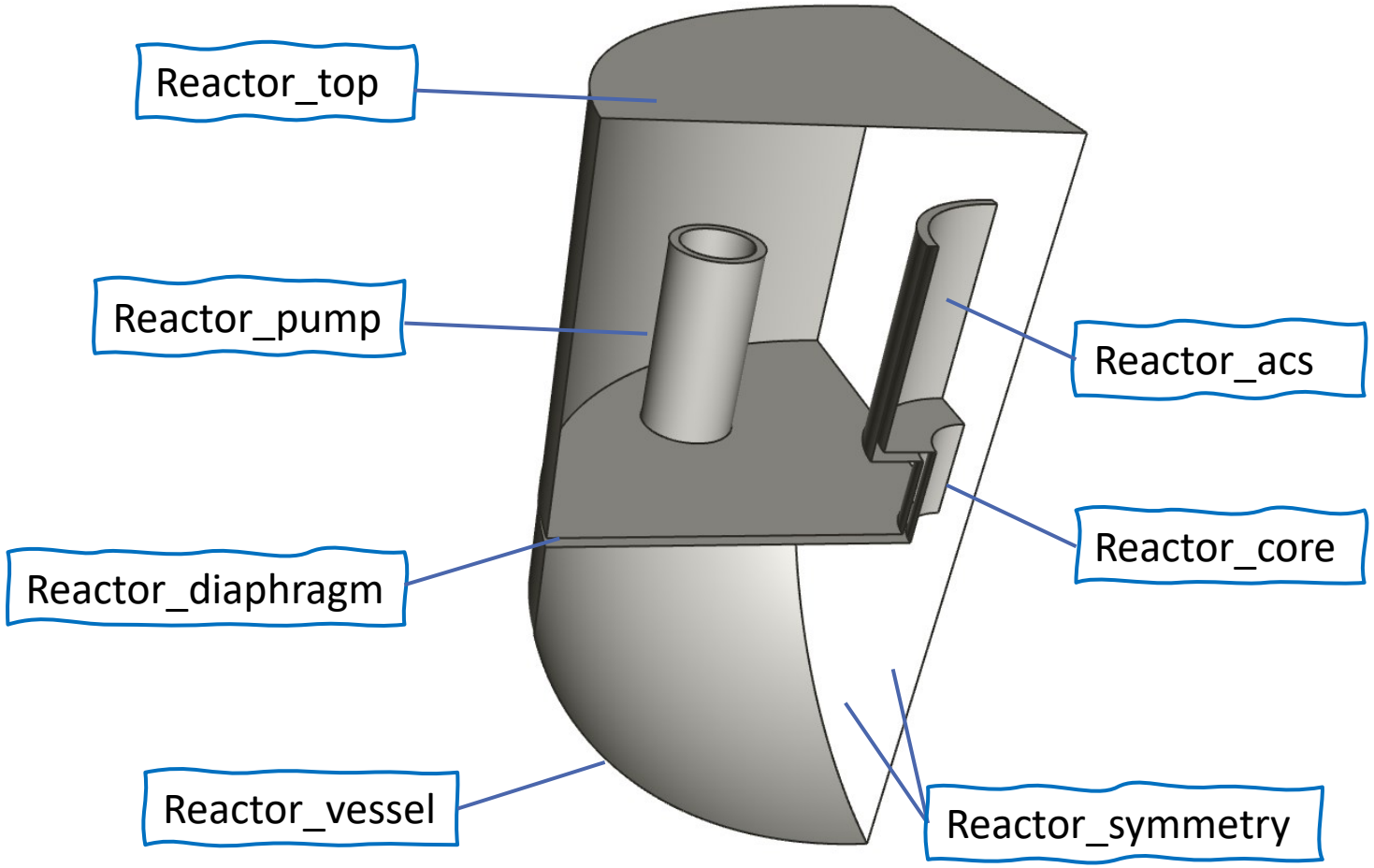
General steps of CFD

1. Define the domain of interest
2. **Discretize the domain**
3. Physical and numerical setup
4. Solve the resulting problem
5. Post-process the results

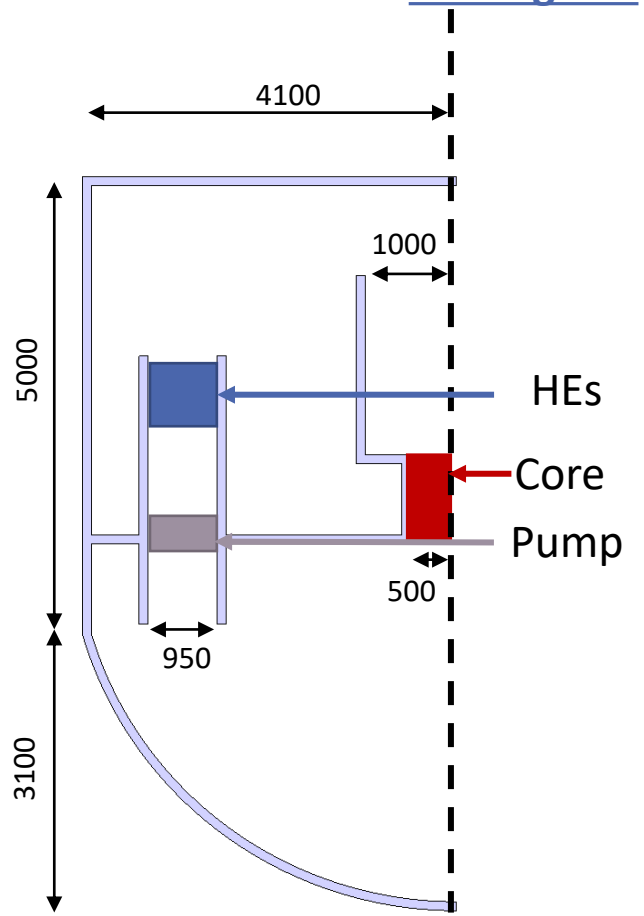
GEOMETRY

¼ of the entire geometry is simulated, by employing the symmetry boundary conditions

Reactor Boundaries



Cell regions



WHAT IS SNAPPYHEXMESH?

- Generation of 3-dimensional meshes containing hexahedra
- Zonal meshing for support of porous media and MRF

STARTING FROM A TUTORIAL DIRECTORY

- Go to the directory dedicated to run OpenFOAM simulations.

```
cd $FOAM_RUN
```

- Create a directory named as river

```
mkdir ESPRESSO
```

- Enter to this directory

```
cd ESPRESSO
```

- Copy the snappyHexMesh/motorBike tutorial directory in your river directory

```
cp -r $FOAM_TUTORIALS/mesh/snappyHexMesh/motorBike/motorbike/* ./
```

```
git clone git@github.com:ofcourse-VKI/ofseminar.git
cd ofseminar/of_seminar_ESPRESSO/tutorial
```

CLEAN TUTORIAL AND COPY NEW GEOMETRY

- Clean the tutorial by removing useless files

```
rm   -r   constant/geometry/*    0/    Allclean    Allrun
```


- Check that the .stl CAD files are in the geometry directory
  - reactor_named.stl
  - core_region.stl
  - pump_region.stl
  - hx_region.stl

```
cp ../results/constant/geometry/* ./constant/geometry/
```

reactor_named.stl is already separated in different surfaces

STL FILES

- Visualize thegeometry in paraview

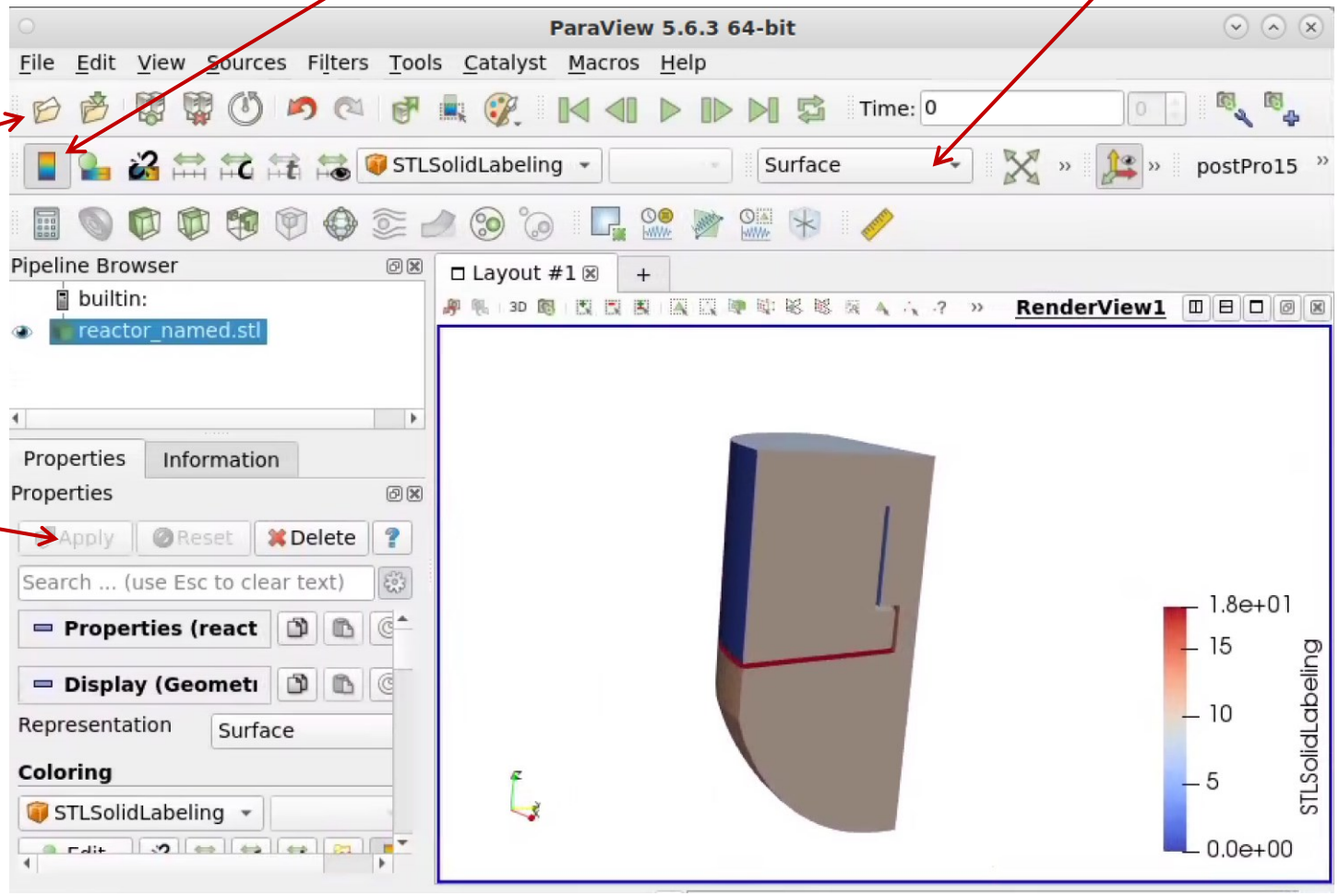
```
paraview &
```

3. Activate the color legend

You can change to *Surface With Edge* (to visualized the triangulated stl mesh)

1. file/open and select the reactor_named.stl in the constant/geometry directory

2. Apply



The color level corresponds to the number of the geometry surfaces

STL FILES

- Create a clip in paraview to visualize the interior of the reactor

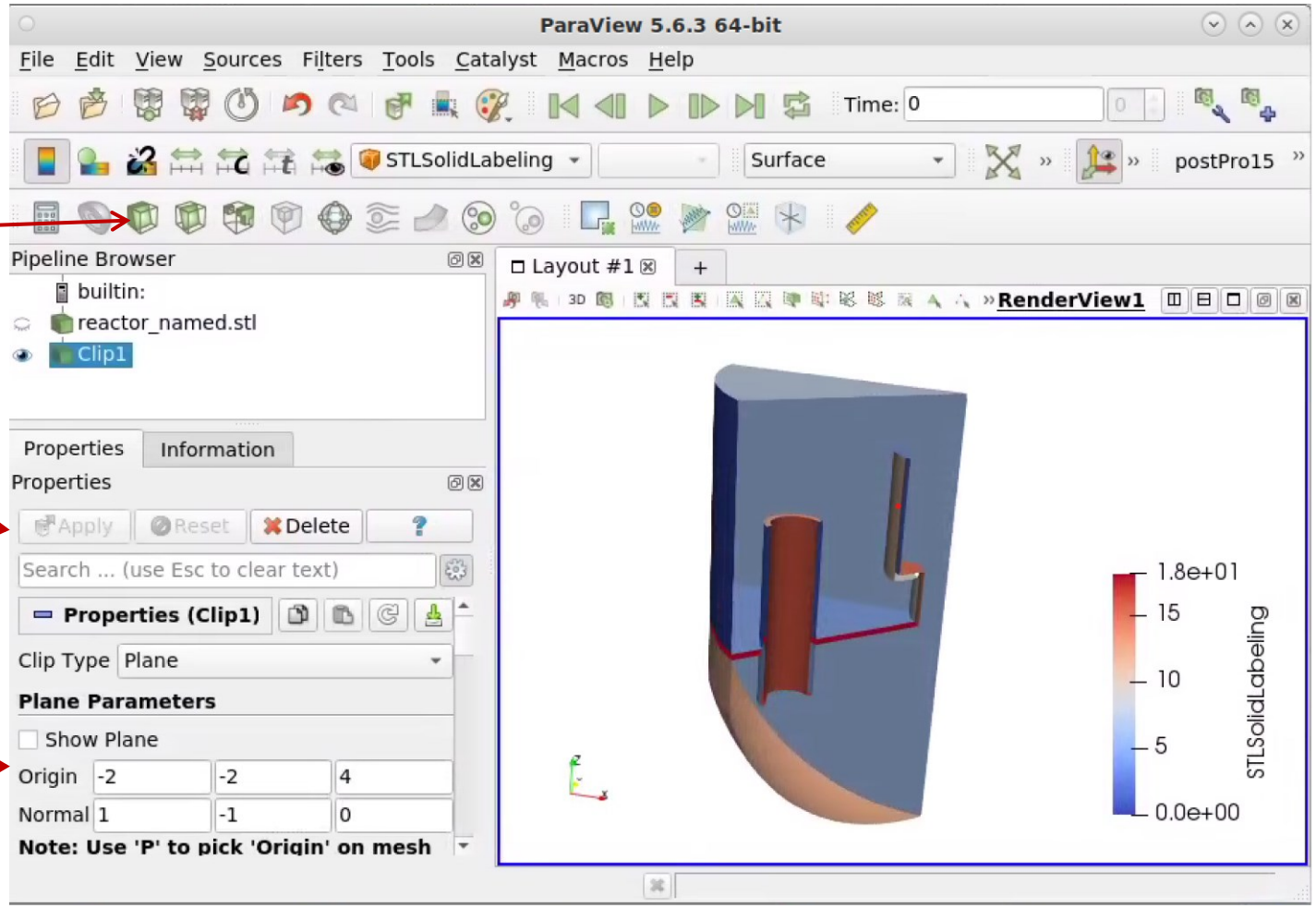
```
paraview &
```

1. Click on Clip

4. Click Apply

3. Un-toggle show Plane

2. Set the Origin and Normal



STL UTILITIES

```
cp -r $FOAM_ETC/caseDicts/surface/surfaceFeaturesDict ./system/
```

- Edit the surfaceFeaturesDict in system directory

```
surfaces
(
    "reactor_named.stl" "core_region.stl "  "hx_region.stl"   "pump_region.stl"
);
// Identify edges when angle between faces < includedAngle
includedAngle       150;
// Include region boundaries as features
geometricTestOnly   no;
subsetFeatures
{
    // Include nonManifold edges (edges with >2 connected faces)
    nonManifoldEdges    no;
    // Include open edges (edges with 1 connected face)
    openEdges           yes;
}
// Write features to obj format for visualisation
writeObj            yes;
verboseObj          no;
```


- Run the edge extraction

```
surfaceFeatures
```

- This is creating a file “reactor_named.eMesh” in the “constant/geometry” directory

STL UTILITIES

- The STL surface can be first checked using surfaceCheck utility :

```
surfaceCheck constant/geometry/reactor_named.stl >log.surfaceCheck
```


- Open the output file

```
vi log.surfaceCheck
```

or use grep to extract informations :

```
grep 'Bounding Box' log.surfaceCheck
```

```
Bounding Box : (-4 -4 0) (2.4349e-16 0 8)
```

This information is used later to define the background mesh size
Can be also obtained by loading the mesh in paraview

BLOCKMESH

- Edit the *blockMeshDict* in system/

Bounding Box : (-4 -4 0) (2.4349e-16 0 8)

```
convertToMeters 1;

vertices
(
    (-4.2           -4.2      -0.2)
    (0.2            -4.2      -0.2)
    (0.2            0.2       -0.2)
    (-4.2           0.2       -0.2)
    (-4.2           -4.2      8.1)
    (0.2            -4.2      8.1)
    (0.2            0.2       8.1)
    (-4.2           0.2       8.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (44 44 83) simpleGrading (1 1 1)
);
edges
(
);
defaultPatch
    {
        name    sides;
        type    patch;
    }
boundary
(
);
```

Remove all the boundaries that they are all defined in the stl file

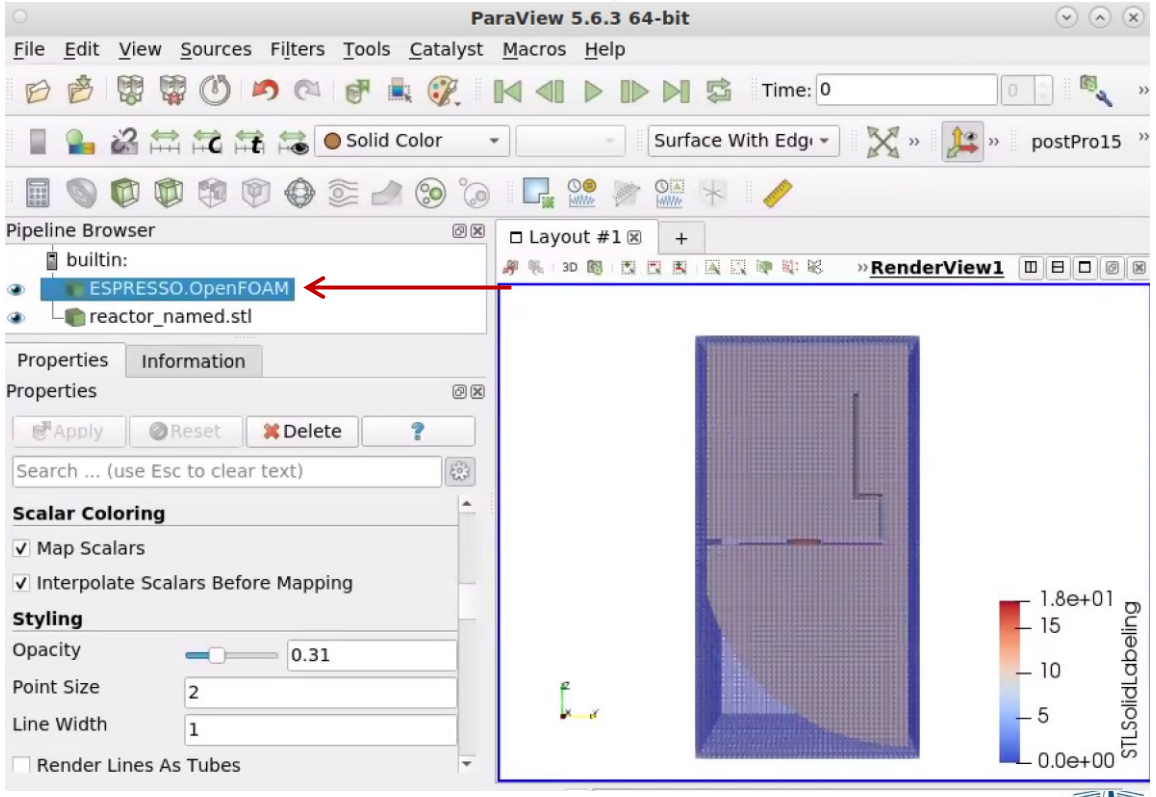
BLOCKMESH

- Run blockMesh :

```
blockMesh
```

- Visualize the background mesh to check correspondence with CAD file

```
paraFoam &
```



SNAPPYHEXMESHDICT- GEOMETRY CONTROLS SECTION

- Open and Edit the snappyHexMeshDict

```
gedit system/snappyHexMeshDict &
```

```
FoamFile
{
   format      ascii;
   class       dictionary;
   object      snappyHexMeshDict;
}
castellatedMesh  true;
snap             true;
addLayers        false;

geometry
{
 reactor
   {
      type triSurfaceMesh;
      file "reactor_named.stl";
   }

…
```

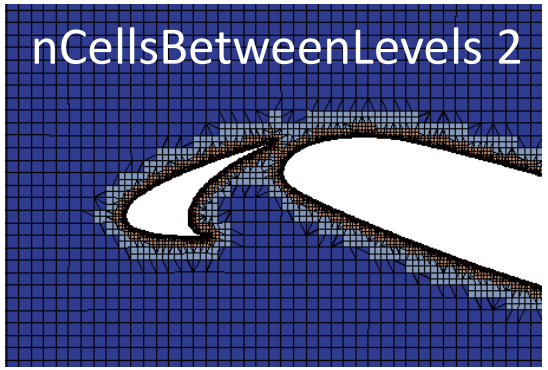
```
core_region
   {
      type triSurfaceMesh;
      file "core_region.stl";
   }

   pump_region
   {
      type triSurfaceMesh;
      file "pump_region.stl";
   }

   hx_region
   {
      type triSurfaceMesh;
      file "hx_region.stl";
   }
…
```

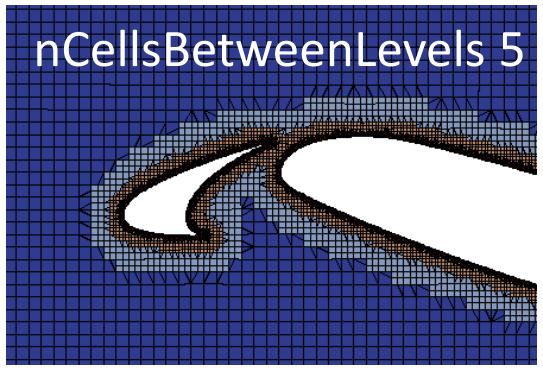
```
refinementCylinderP
   {
      type searchableCylinder;
      point1 (-2.1 -2.1 3);
      point2 (-2.1 -2.1 6.3);
     radius 0.5;
   }
 refinementCylinderC
   {
      type searchableCylinder;
      point1 (0.0 0.0 3.2);
      point2 (0.0 0.0 5.1);
      radius 0.65;
   }
 refinementCylinderA
   {
      type searchableCylinder;
      point1 (0.0 0.0 5.1);
      point2 (0.0 0.0 8.0);
      radius 1.2;
   }
 refinementCylinderL
   {
      type searchableCylinder;
      point1 (0.0 0.0 0);
      point2 (0.0 0.0 4);
      radius 4.1;
   }
};
```

SNAPPYHEXMESHDICT- GENERAL MESH CONTROL

```
castellatedMeshControls
{
maxLocalCells 100000;

maxGlobalCells 2000000;

minRefinementCells 10;

maxLoadUnbalance 0.1;

nCellsBetweenLevels 4;
```

```
features
    (
      {
            file "reactor_named.eMesh";
            level 0;
        }
        {
            file "hx_region.eMesh";
            level 2;
        }
        {
            file "core_region.eMesh";
            level 2;
        }
        {
            file "pump_region.eMesh";
            level 2;
        }
    );
```





snappyHexMeshDict-surface refinement

```
refinementSurfaces
    {
        reactor
        {
          level (0 0);
          regions
          {
              "symmetry.*"
              {
                level (0 0);
              }
              "diaph.*"
              {
                level (1 1);
              }
              "core.*"
              {
                level (3 3);
              }
…
```

```
              "acs.*"
              {
                level (2 2);
              }
              "acs5*"
              {
                level (3 3);
                            }
              "vessel.*"
              {
                 level (0 0);
              }
              "top.*"
              {
                 level (0 0);
              }
              "pump.*"
              {
                 level (2 2);
              }
            }
          }
…
```

```
     core_region
        {
             level (2 2);
             faceZone core_region;
            cellZone core_region;
            cellZoneInside inside;
        }

        pump_region
        {
            level (1 1);
            faceZone pump_region;
            cellZone pump_region;
            cellZoneInside inside;
        }

        hx_region
        {
           level (1 1);
           faceZone hx_region;
            cellZone hx_region;
            cellZoneInside inside;
        }
    }

    // Resolve sharp angles
    resolveFeatureAngle 30;
```

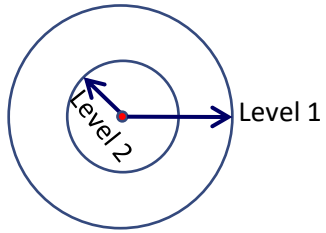
SNAPPYHEXMESHDICT-REGION REFINEMENT

```
refinementRegions
    {
       refinementCylinderP
        {
            mode inside;
            levels ((1E15 1));
        }
     refinementCylinderC
        {
            mode inside;
            levels ((1E15 2));
        }
      refinementCylinderA
        {
            mode inside;
            levels ((1E15 2));
        }
      refinementCylinderL
        {
            mode inside;
            levels ((1E15 1));
        }
    }
locationInMesh (-1 -1 5);

allowFreeStandingZoneFaces true;
```

Level of refinement can also be defined by specifying the distance around the edge that will take the level of refinement

levels ((0.1 2) (0.2 1));

Distance in meter




```
snapControls
{
    nSmoothPatch 3;

    tolerance 2.0;

    nSolveIter 30;

    nRelaxIter 5;

    nFeatureSnapIter 10;

     implicitFeatureSnap false;

    explicitFeatureSnap true;

    multiRegionFeatureSnap false;
}
```

CREATE THE MESH WITH SHM

- Run snappyHexMesh in the case directory

```
snappyHexMesh -overwrite
```


- Check the quality of the mesh

```
checkMesh
```

- Before visualize the mesh, we will create the boundary conditions

CREATE BOUNDARY CONDITIONS

- Create patches out of selected boundary faces and change name and type of the boundaries

```
cp   $FOAM_ETC/caseDicts/mesh/manipulation/patches/createPatchDict ./system
```

- Edit *system/createPatchDict*

```
patches
(
   {
      name    diaph;
      patchInfo
      {
         type    wall;
      }
      constructFrom patches;
      patches ("reactor_diaph.*");
   }

   {
      name    acs;
      patchInfo
      {
         type    wall;
      }
      constructFrom patches;
      patches ("reactor_acs.*");
   }
```

```
   {
      name    core;
      patchInfo
      {
         type    wall;
      }
      constructFrom patches;
      patches ("reactor_core.*");
   }

   {
      name    vessel;
      patchInfo
      {
         type    wall;
      }
      constructFrom patches;
      patches ("reactor_vessel.*");
   }
```

```
   {
      name    pump;
      patchInfo
      {
         type    wall;
      }
      constructFrom patches;
      patches ("reactor_pump.*");
   }
   {
      name    top;
      patchInfo
      {
         type    wall;
      }
      constructFrom patches;
      patches ("reactor_top.*");
   }
```

```
   {
      name    symmetry1;
      patchInfo
      {
         type    symmetry;
      }
      constructFrom patches;
      patches ("reactor_symmetry1");
   }
   {
      name    symmetry2;
      patchInfo
      {
         type    symmetry;
      }
      constructFrom patches;
      patches ("reactor_symmetry2");
   }
);
```

CREATE BOUNDARY CONDITIONS

- Run *createPatch* to create the new BC

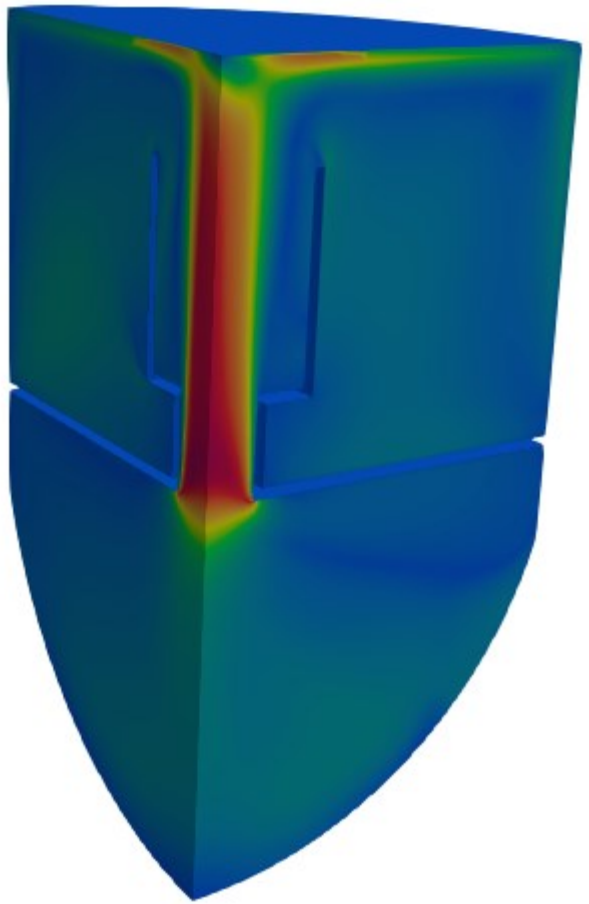
```
createPatch -overwrite
```


- Visualize in paraFoam

```
paraFoam
```

General steps of CFD

1. Define the domain of interest
2. Discretize the domain
3. **Physical and numerical setup**
4. Solve the resulting problem
5. Post-process the results



CHOICE OF THE SOLVER BASED ON THE EQUATIONS

The RANS module for non-isothermal (buoyant) flows in OpenFoam 11 is called

fluid

And it is approximating this set of equations:

$$\nabla \cdot (\rho \mathbf{u}) = 0$$ Mass

$$\nabla \cdot (\rho \mathbf{u}\mathbf{u}) = -\nabla p + \rho \mathbf{g} + \nabla \cdot \left(2\mu_{eff} D(\mathbf{u})\right) - \nabla \cdot \left(\frac{2}{3}\mu_{eff} \nabla \cdot (\mathbf{u})\right)$$ Momentum

Many of these terms will be nil in our case as a result of the **incompressibility**

$$\nabla \cdot (\rho \mathbf{u} h) + \nabla \cdot (\rho \mathbf{u} K) = \rho(\mathbf{g} \cdot \mathbf{u}) + \nabla \cdot \left(\alpha_{eff} \nabla h\right)$$ Energy

STARTING FROM A TUTORIAL DIRECTORY

- Go to the directory dedicated to run OpenFOAM simulations.

```
cd $FOAM_RUN
```

- Create a directory named as ESPRESSO setup

```
mkdir ESPRESSO_setup
```

- Enter to this directory

```
cd ESPRESSO_setup
```

- Copy the buoyantCavity tutorial directory in your directory

```
cp -r $FOAM_TUTORIALS/fluid/buoyantCavity/* ./
```

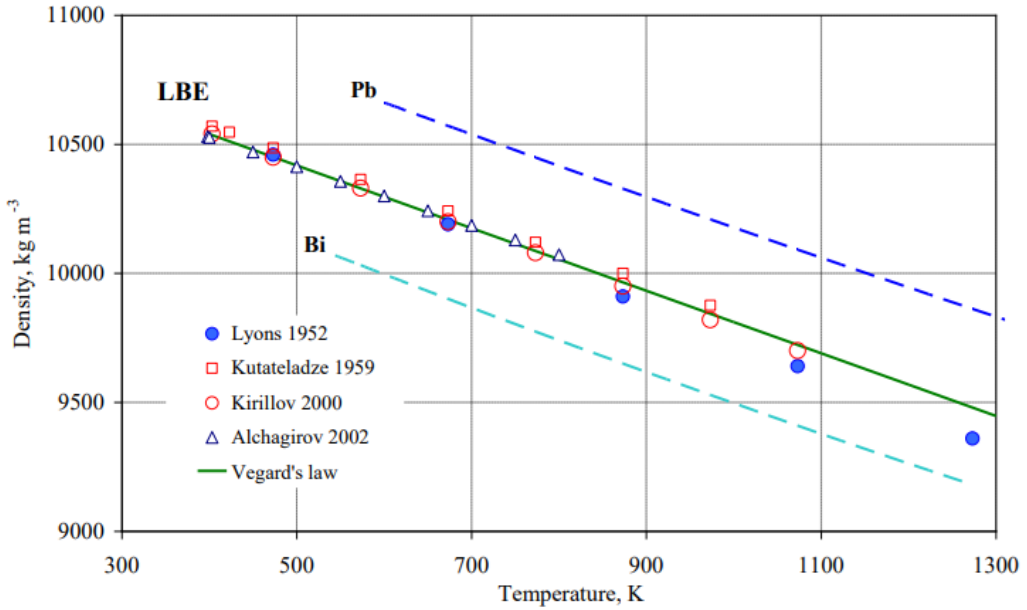
- Copy the mesh created previously in your directory

```
cp -r ../ESPRESSO/constant/polyMesh/ ./constant/
```

```
mv 0.orig 0
```

PHYSICAL PROPERTIES

LBE




```
thermoType
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       polynomial;
    thermo          hPolynomial;
    equationOfState icoPolynomial;
    specie          specie;
    energy          sensibleEnthalpy;
}

mixture
{
    specie
    {
       molWeight  208.98;
    }
    thermodynamics
    {
       Hf              0;
       Sf              0;
       CpCoeffs<8>     ( 159 0.0272 7.12e-6 0 0 0 0 0 );
    }
    equationOfState
    {
       rhoCoeffs<8>    ( 11096 -1.3326 0 0 0 0 0 0 );
    }
    transport
    {
        muCoeffs<8>     ( 0.0056 -1e-5 5e-9 0 0 0 0 0 );
        kappaCoeffs<8>  ( 3.61 1.517e-2 1.741e-6 0 0 0 0 0 ); }
     }
```

In case you want to account for variable transport properties, you can specify the **coefficients** of **polynomial laws**

Property	Formula
Density(ρ) [kg m^3]	11096-1.3326T
Heat capacity (cp) [J$kg^{-1}K^{-1}$]	159-2.72 10^{-2}T+7.12 $10^{-6}T^2$
Th. conductivity (λ) [W$m^{-1}K^{-1}$]	3.61+1.517 $10^{-2}T$-1.741 $10^{-6}T^2$
Cinematic Viscosity (μ) [kg$m^{-1}s-1$]	4.94 10^{-4} exp(754.1/T)
Prandtl number (Pr) [-]	$c_p\mu\lambda^{-1}$

TURBULENCE MODELS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - nuSgs
- constant/
  - polyMesh/
    - boundary
    - ....
  - thermoPhysicalProperties
  - momentumProperties
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - decomposeParDict

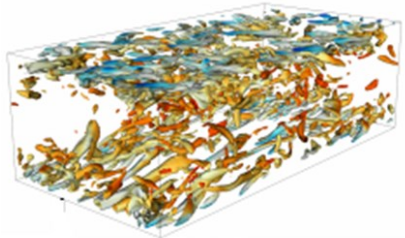
```
simulationType RAS;

RAS
{
    model                kOmegaSST;

    turbulence           on;

    printCoeffs          on;
}
```

Several **turbulence models** are available in OpenFoam.

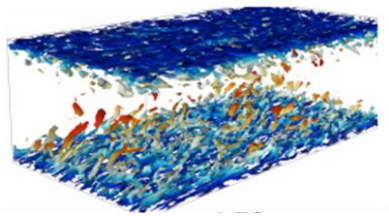
laminar
uses no turbulence models;

RAS
uses Reynolds-averaged simulation (RAS) modelling;

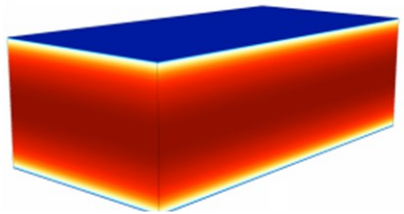
LES
uses large-eddy simulation (LES) modelling.



DNS
(Direct Navier-Stokes simulation)



LES



RANS
(Reynolds Averaged Navier-Stokes simulation)

THERMAL TURBULENCE MODELS

```
cp -r $FOAM_TUTORIALS/fluid/cavity/constant/thermophysicalTransport constant/
```

../ESPRESSO
- 0/
  - U
  - p
  - k
  - nuSgs
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - thermoPhysicalTransport
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - decomposeParDict

```
RAS
{
    model    eddyDiffusivity;

    Prt      2;
}
```

The value of $P_{r,T} = 0.85$ is the default one, but for forced convection in liquid metals the value of $P_{r,T} = 2.0$ is recommended

The **Reynolds' Analogy** is the only thermal turbulence model currently implemented in the released version of OpenFoam 9

$$q_{T,j} = -\alpha_T \frac{\partial T}{\partial x_j} \qquad \alpha_T = \frac{\upsilon_T}{P_{r,T}}$$

For liquid metal turbulent simulations, we implemented other thermal turbulence models (*ThermophysicalTransportModels* library): Kays Correlation, Manservisi $k_\theta - \varepsilon_\theta$, etc.

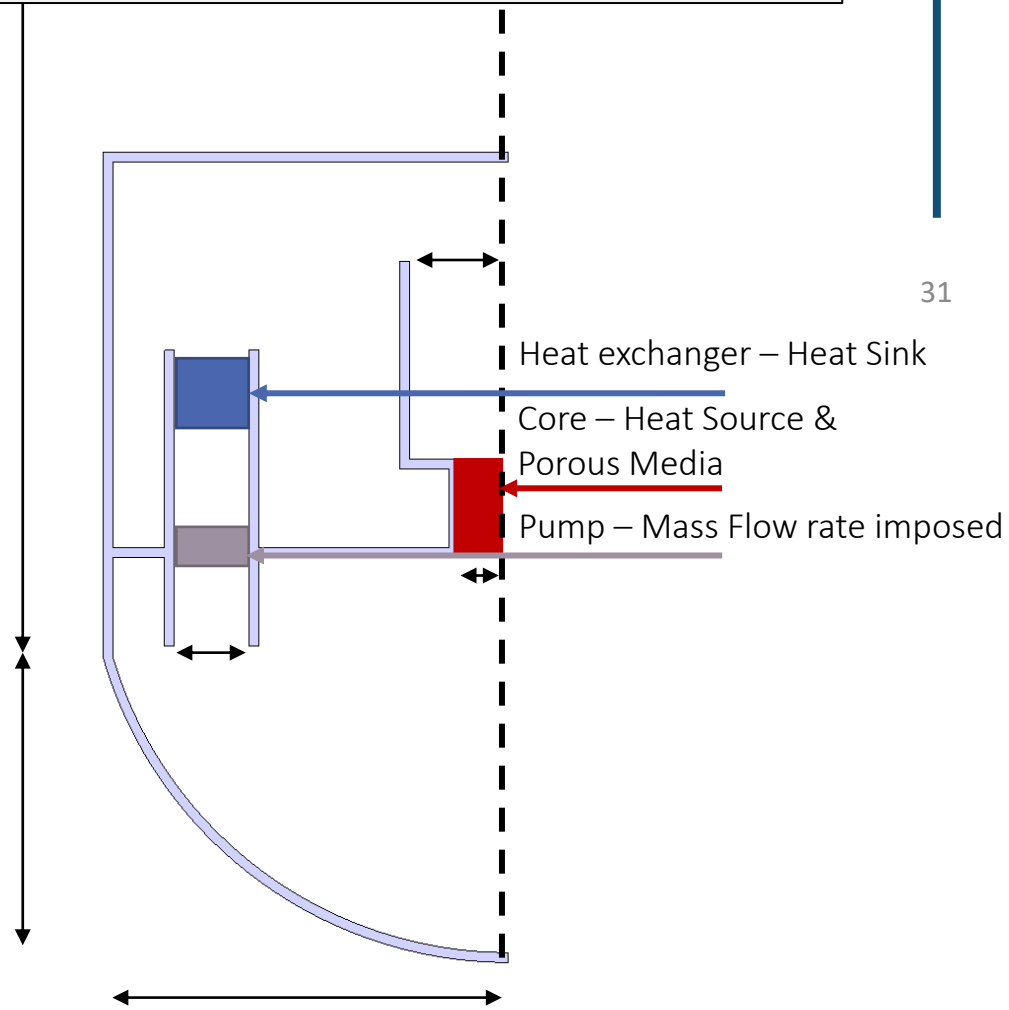
SOURCE TERMS

```
cp -r $FOAM_ETC/caseDicts/fvConstraints/fvConstraints system/
```

../ESPRESSO
- 0/
  - U
  - p
  - k
  - nuSgs
- constant/
  - polyMesh/
    - boundary
    - ....
  - thermoPhysicalProperties
  - thermoPhysicalTransport
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraints

```
Pump1
{
    type meanVelocityForce;

    meanVelocityForceCoeffs
   {
            selectionMode cellZone;
            cellZone pump_region;
            fields (U);
            Ubar (0 0 -0.384);
    }
}
```

Each source terms is specified for a particular **cellZone**



SOURCE TERMS

```
cp -r $FOAM_ETC/caseDicts/fvModels/fvModels constant/
```

../ESPRESSO
- 0/
  - U
  - p
  - k
  - nuSgs
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalPropert
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

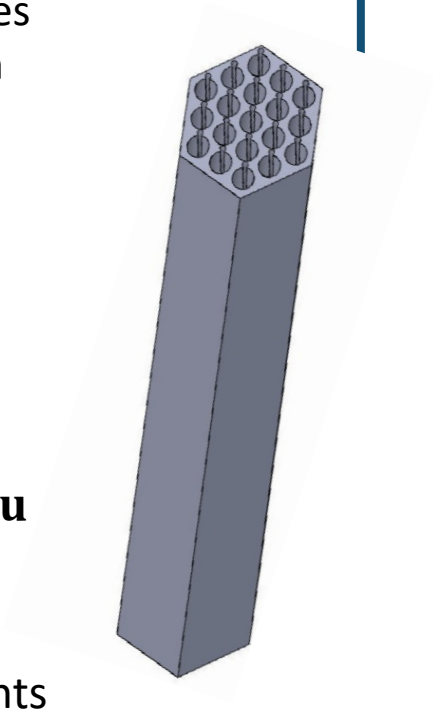
```
porosity
{
 type     explicitPorositySource;

 explicitPorositySourceCoeffs
 {
    selectionMode   cellZone;
    cellZone        core_region;

    type            DarcyForchheimer;
    d   (1e7 1e7 0);
    f   (0 0  75);
    coordinateSystem {
      type    cartesian;
      origin  (0 0 0);
      coordinateRotation
          {
             type    axesRotation;
             e1   (1 0 0);
             e2   (0 1 0);
          }
     }
}
```

The flow trough the fuel assemblies is modelled as a **porous medium**

Darcy-Forchheimer Law

$$\nabla \cdot (\rho \mathbf{u}\mathbf{u}) = \nabla \cdot \boldsymbol{\sigma} + \mathbf{S}_m$$

$$\mathbf{S}_m = -\left(\mu \mathbf{D} + \frac{1}{2}\rho\, tr(\mathbf{u} \cdot \mathbf{I})\mathbf{F}\right)\mathbf{u}$$

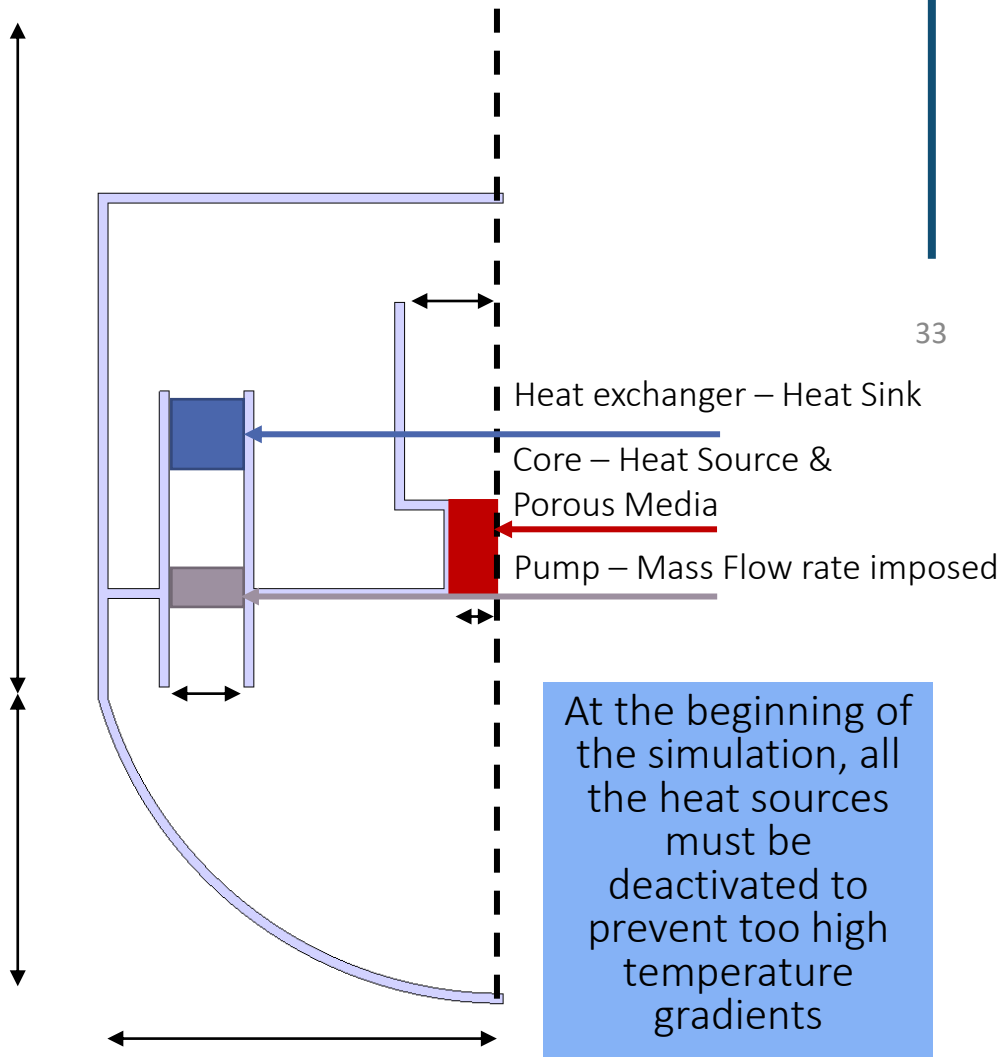


We must specify the coefficients $\mathbf{D}$ and $\mathbf{F}$ (directional dependent)

SOURCE TERMS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - nuSgs
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalPropertie
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

```
Core_heat
{
            type semiImplicitSource;
            selectionMode cellZone;
            cellZone core_region;

            volumeMode absolute;



            sources
            {
            h
                        {
                        explicit 1e7;
                        implicit 0;
                        }
            }
}
```

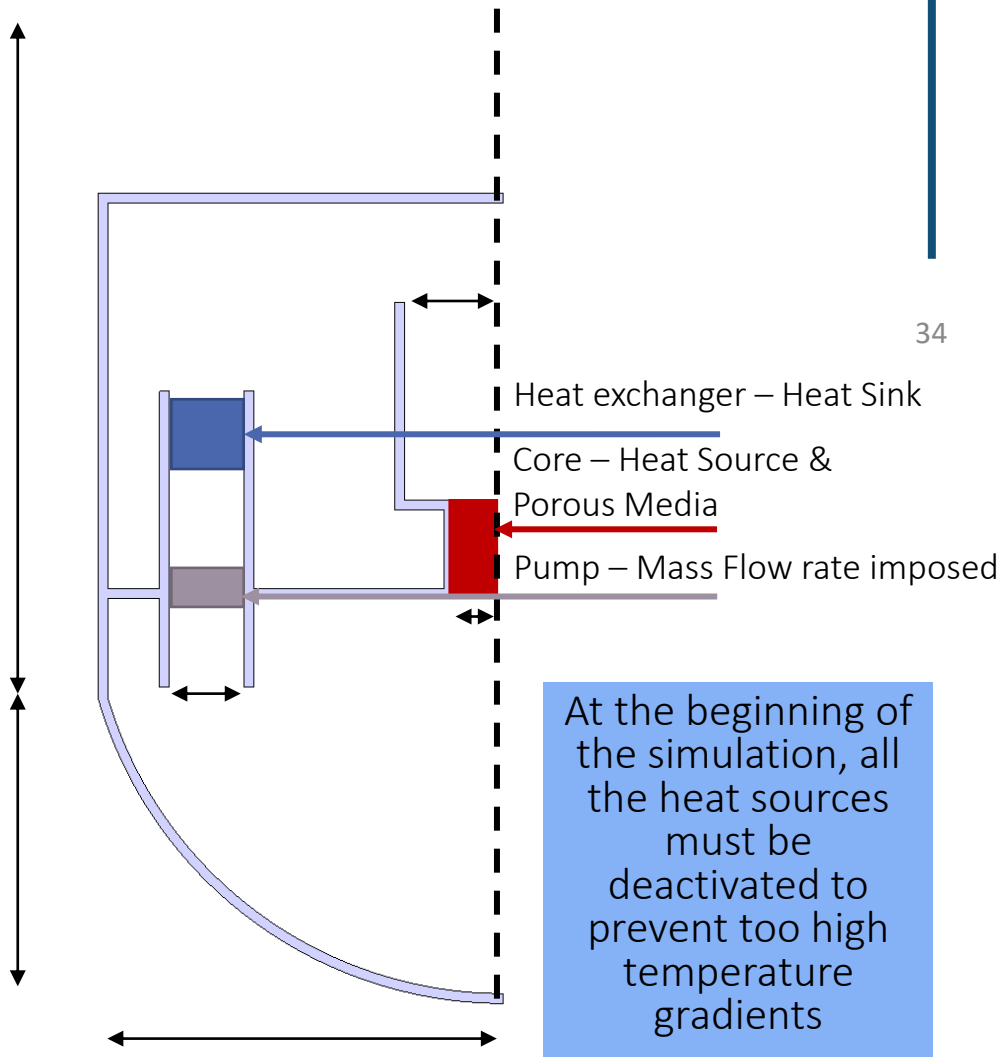


At the beginning of the simulation, all the heat sources must be deactivated to prevent too high temperature gradients

SOURCE TERMS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - nuSgs
- constant/
  - polyMesh/
    - boundary
    - ....
  - thermoPhysicalPropertie
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

```
HX_heat
{
        type semiImplicitSource;
        selectionMode cellZone;
        cellZone hx_region;



        volumeMode absolute;



        sources
        {
        h
                {
                explicit -1e7;
                implicit 0;
                }
}
}
```



At the beginning of the simulation, all the heat sources must be deactivated to prevent too high temperature gradients

INITIAL CONDITIONS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - nuSgs
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

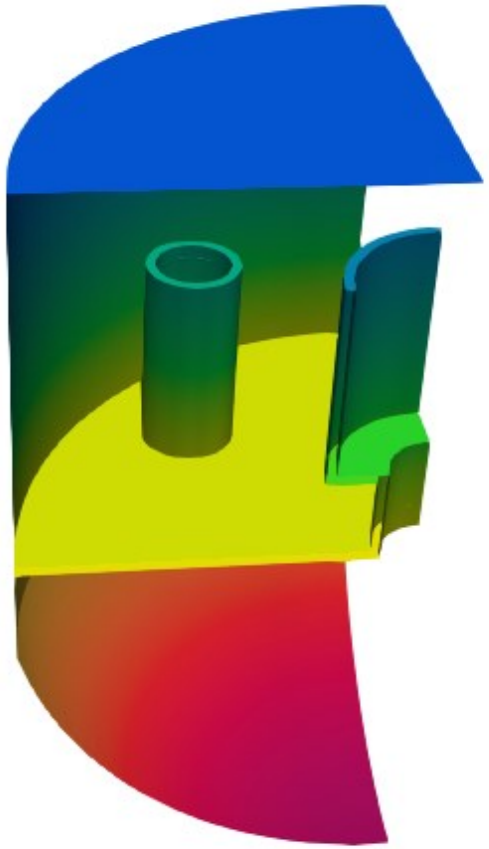
Initialization with constant values

```
 object      alphat;
}
// ************************************ //

dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 0;
```

$\alpha_t = 0.0$ $\nu_t = 0.0$ $k = 0.000375$

$\omega = 0.12$ $p = 1 \cdot 10^5$ $\mathbf{u} = [0.0, 0.0, 0.0]$

$p_{rgh} = 1 \cdot 10^5$ $T = 493$

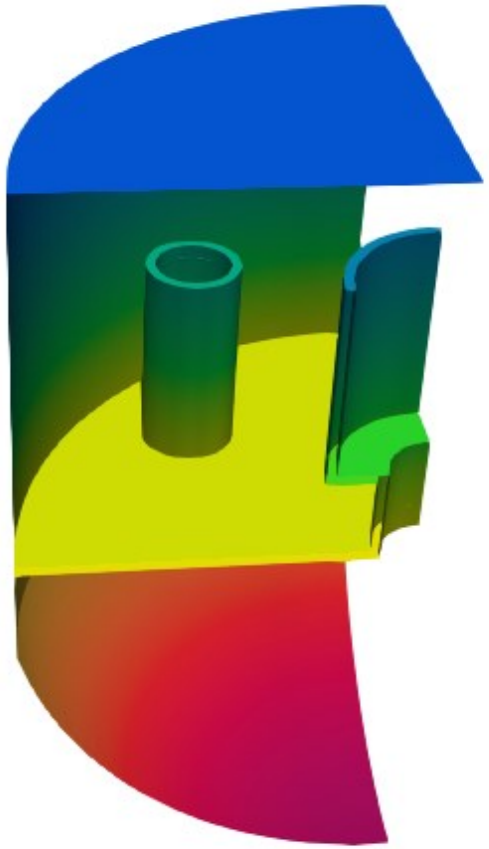


BOUNDARY CONDITIONS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - T
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

$$U$$

```
boundaryField
{
    "(top|diaph|core|acs|vessel|pump)"
    {
        type            noSlip;
    }

    "symmetry.*"
    {
        type            symmetry;
    }
}
```



BOUNDARY CONDITIONS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - T
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

T

```
boundaryField
{
"(diaph|core|acs|pump)"
        {
                type                zeroGradient;
        }
vessel
        {
                type                externalWallHeatFluxTemperature;
                mode                flux;
                q                   uniform -150.0;
                value               $internalField;
        }
top
        {
                type                externalWallHeatFluxTemperature;
                mode                        coefficient;
                Ta                          constant 450.0;
                h                           uniform 10.0;
                thicknessLayers     (0.1);
                kappaLayers                 (45);
                value                       $internalField;
        }
"symmetry.*"
        {
                type                        symmetry;
        }
```

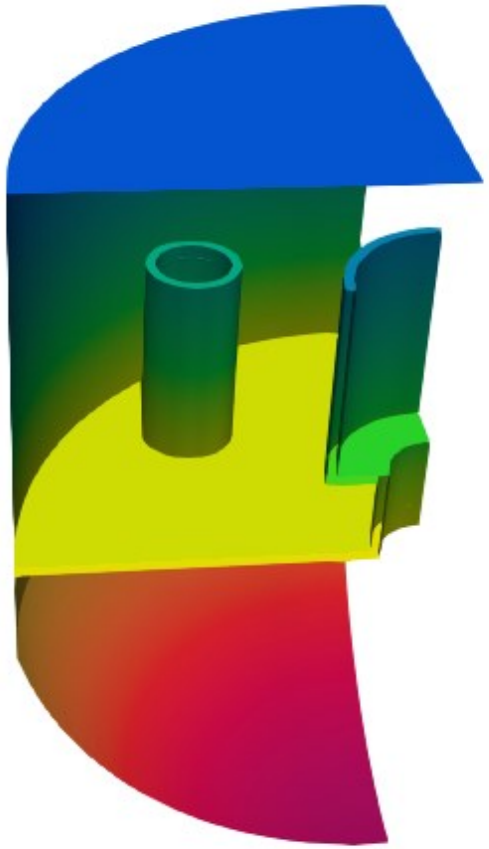
- **fixed power**: supply Q
- **fixed heat flux**: supply q
- **fixed heat transfer coefficient**: supply h and Ta

BOUNDARY CONDITIONS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - T
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

k

```
boundaryField
{
"(top|diaph|core|acs|vessel|pump)"
    {
        type            kqRWallFunction;
        value           uniform 3.75e-04;
    }
    "symmetry.*"
    {
        type            symmetry;
    }
}
```

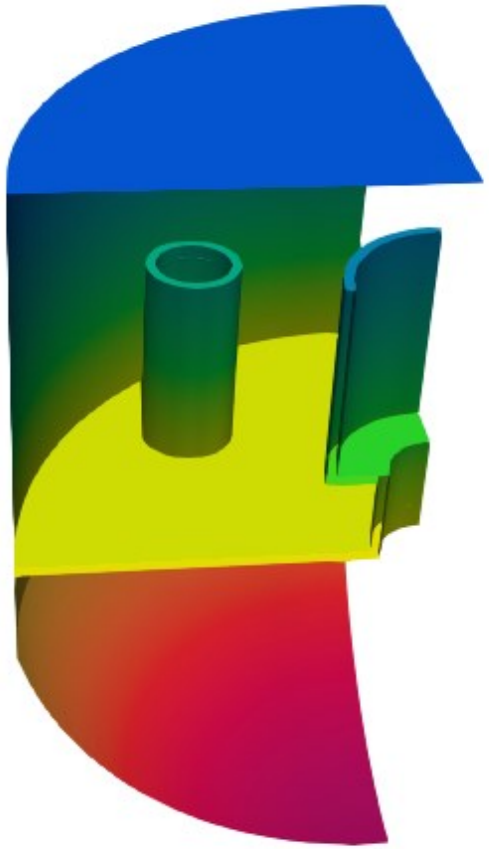


BOUNDARY CONDITIONS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - omega
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

ω

```
boundaryField
{
    "(top|diaph|core|acs|vessel|pump)"
    {
        type            omegaWallFunction;
        value           uniform 0.12;
    }
    "symmetry.*"
    {
        type            symmetry;
    }
}
```

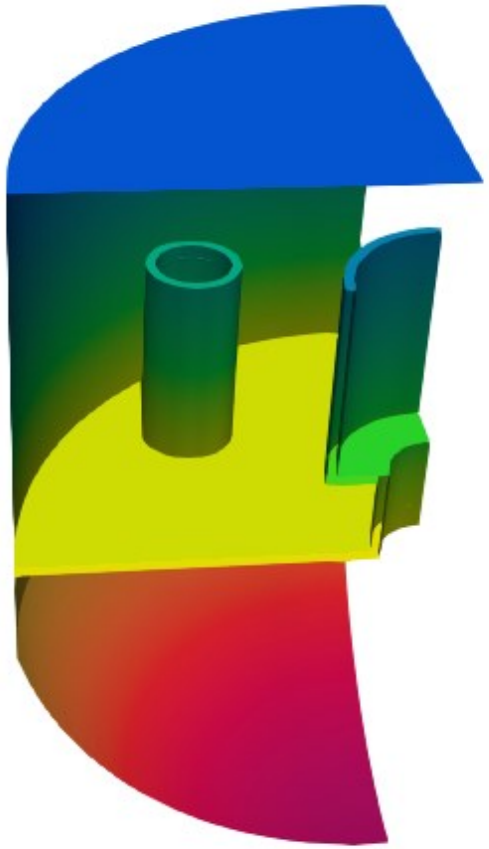


BOUNDARY CONDITIONS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - omega
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

nut

```
boundaryField
{
    "(top|diaph|core|acs|vessel|pump)"
    {
        type            nutkWallFunction;
        value           uniform 0.12;
    }
    "symmetry.*"
    {
        type            symmetry;
    }
}
```

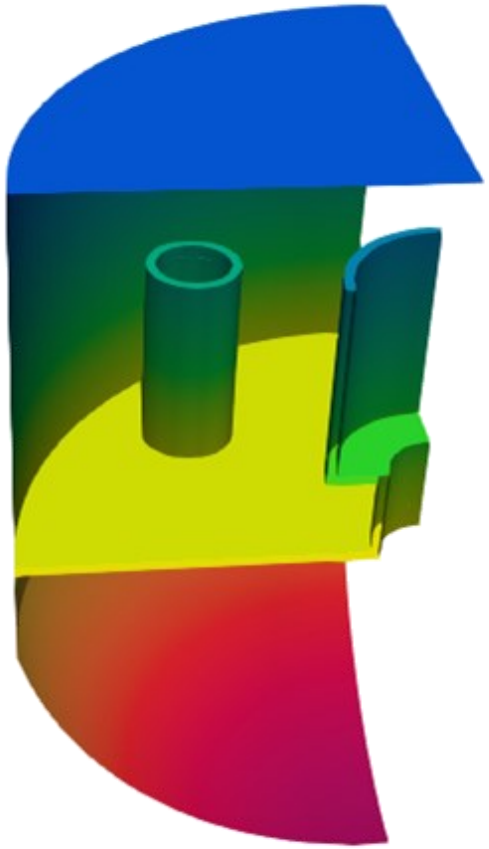


BOUNDARY CONDITIONS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - omega
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

alphat

```
boundaryField
{
    "(top|diaph|core|acs|vessel|pump)"
    {
        type            compressible:alphatWallFunction;
        value           uniform 0.12;
    }
    "symmetry.*"
    {
        type            symmetry;
    }
}
```

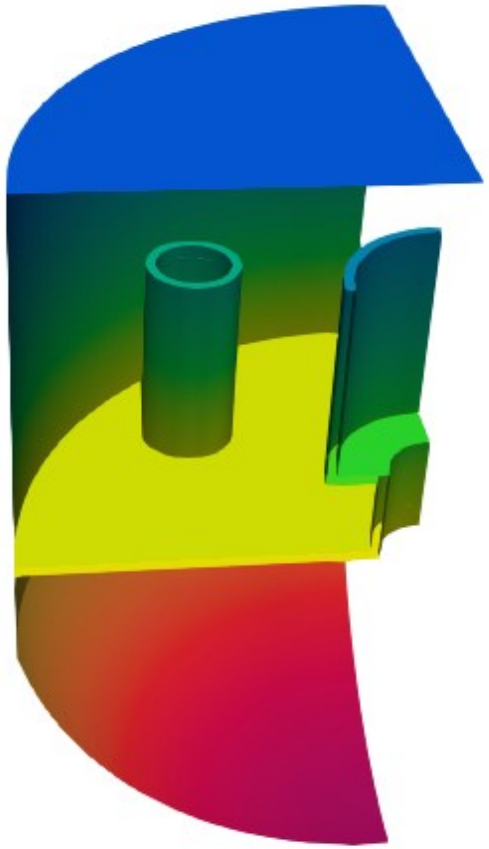


BOUNDARY CONDITIONS

../ESPRESSO
- 0/
  - U
  - p
  - k
  - p_rgh
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

p_{rgh}

```
boundaryField
{
    "(top|diaph|core|acs|vessel|pump)"
    {
        type            fixedFluxPressure;
        value           $internalField;
    }

    "symmetry.*"
    {
        type            symmetry;
    }
}
```

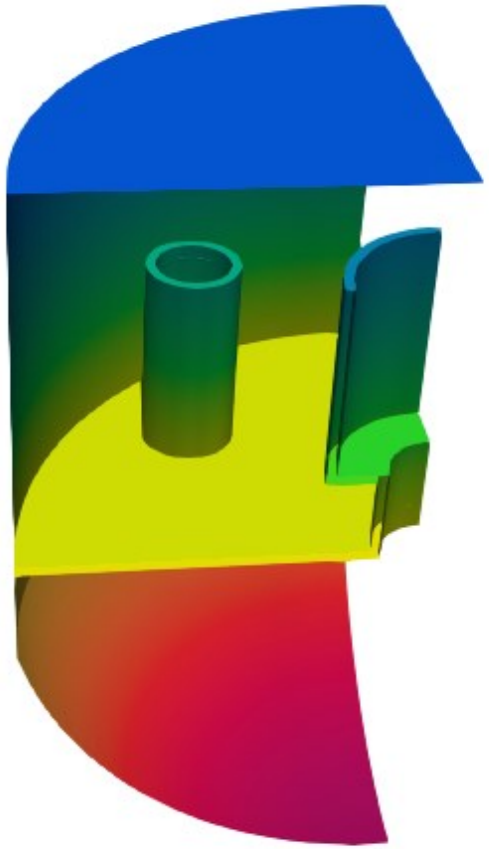


BOUNDARY CONDITIONS

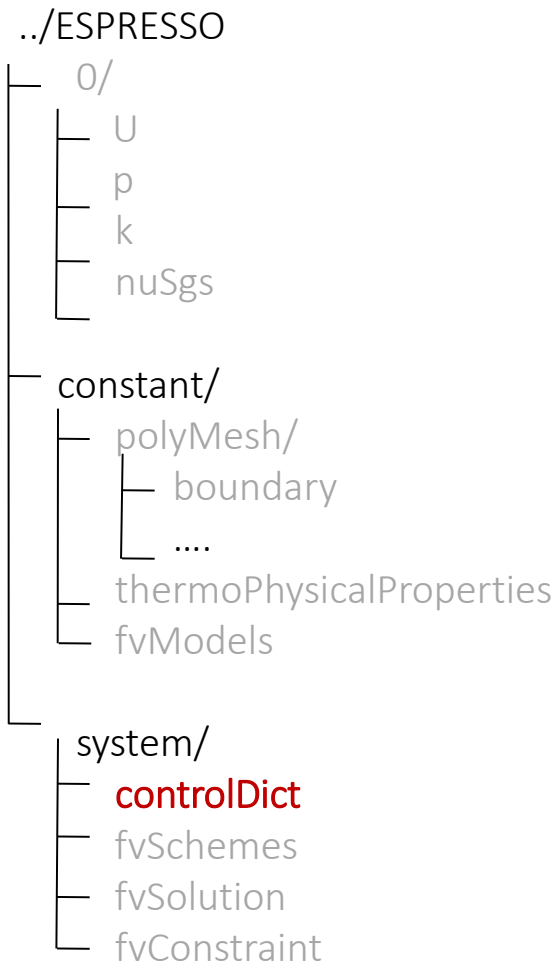
../ESPRESSO
- 0/
  - U
  - p
  - k
  - p
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - fvConstraint

p

```
boundaryField
{
    "(top|diaph|core|acs|vessel|pump)"
    {
        type            calculated;
        value           $internalField;
    }

    "symmetry.*"
    {
        type            symmetry;
    }
}
```



SIMULATION CONTROLS




```
application     foamRun;

solver          fluid;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         10000;

deltaT          1;

writeControl    timeStep;

writeInterval   100;

purgeWrite      2;

writeFormat     ascii;

writePrecision  8;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;
```

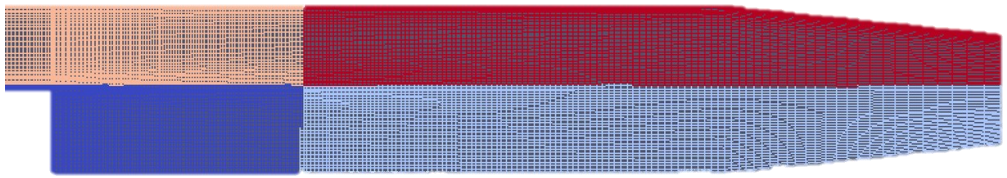
PARALLEL COMPUTATION

```
cp -r $FOAM_ETC/caseDicts/preProcessing/decomposeParDict system/
```

../ESPRESSO
- 0/
  - U
  - p
  - k
  - nuSgs
- constant/
  - polyMesh/
    - boundary
    - ….
  - thermoPhysicalProperties
  - fvModels
- system/
  - controlDict
  - fvSchemes
  - fvSolution
  - decomposeParDict

```
numberOfSubdomains  2;

method              scotch;
// method           hierarchical;
// method           simple;
// method           manual;

simpleCoeffs
{
    n               (2 1 1);
    delta           0.001;
}

hierarchicalCoeffs
{
    n               (2 1 1);
    delta           0.001;
    order           xyz;
}
```


Decomposition methods:
- **Simple**: split in equal parts along directions
- **Hierarchical**: same as simple, but with specified order of directions
- **Manual**: each cell is assigned to a processor
- **Scotch**: minimizes the communication between processors



RUNNING THE SOLVER

In serial, run the case by typing on the terminal:

```
foamRun >log &
```

And visualize the output file evolution with:

```
tail -f log
```

```
tail -n 200 log
```

To run in parallel, you should first decompose the case (*decomposeParDict* specifies 2 processors)

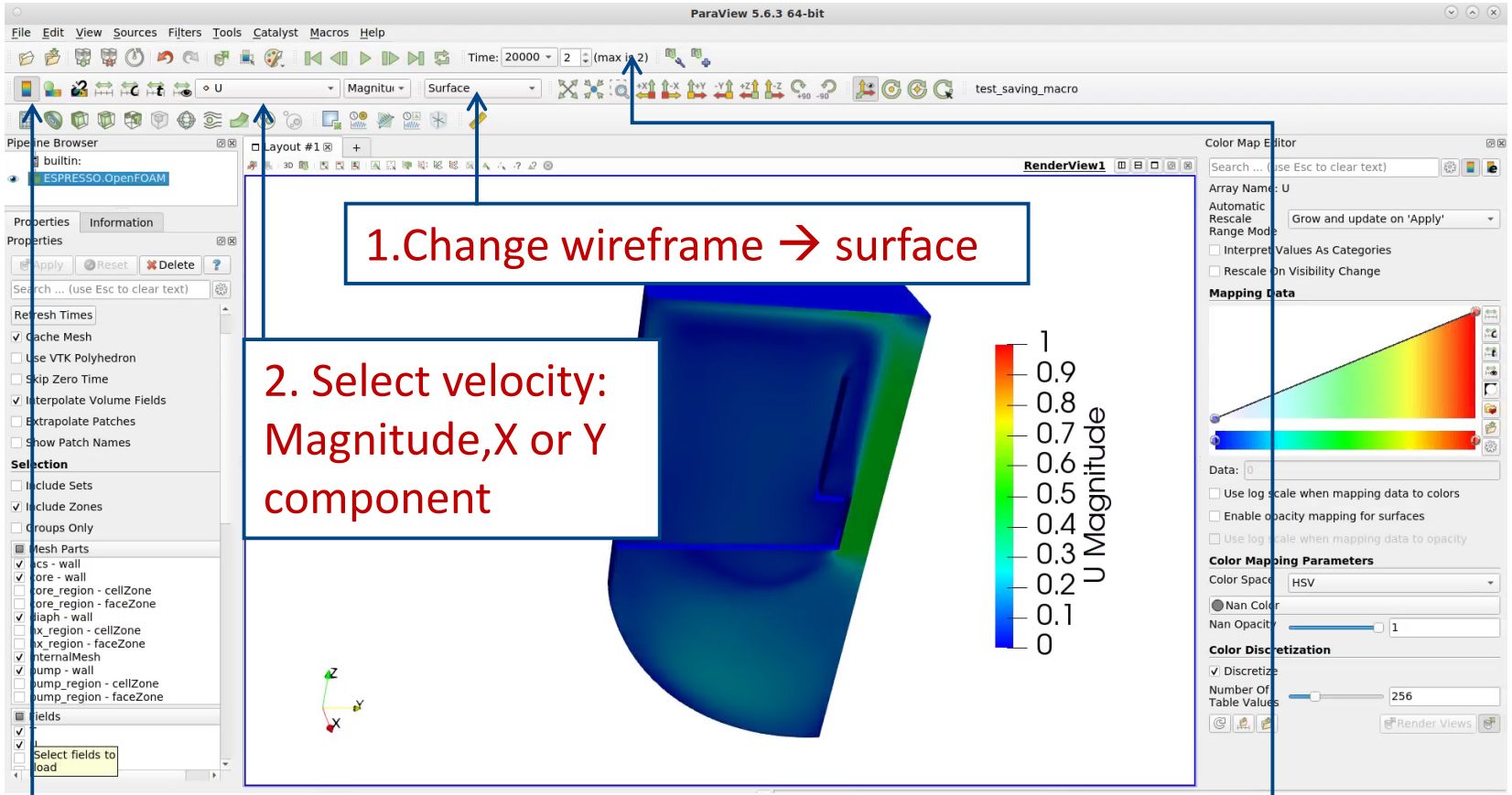
```
decomposePar -force
```

Then you run the decomposed case by using:

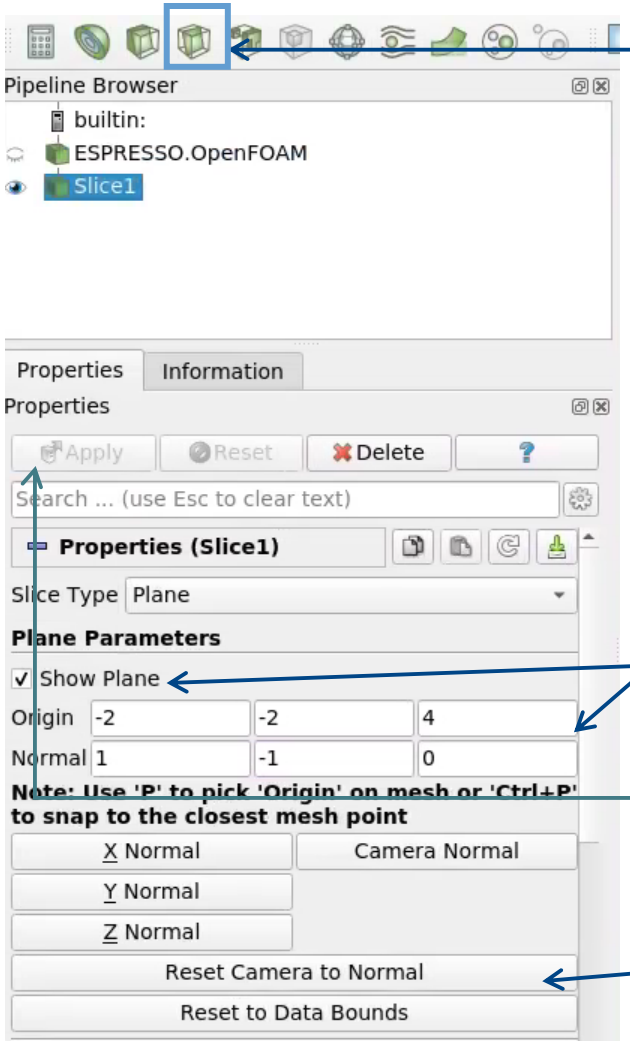
```
mpirun -np 2 foamRun -parallel >log &
```

For this particular test case, you should first run the first **100 iterations** with inactive heat sources (by commenting the corresponding lines in the ***fvModels*** file). Then you active the heat sources and continue the run until reaching **10000 iterations**

PARAFOAM – PLOT VELOCITY CONTOUR



PARAFOAM – 2D SLICE



1. Create Slice

This action is automatically creating an object in the pipeline browser

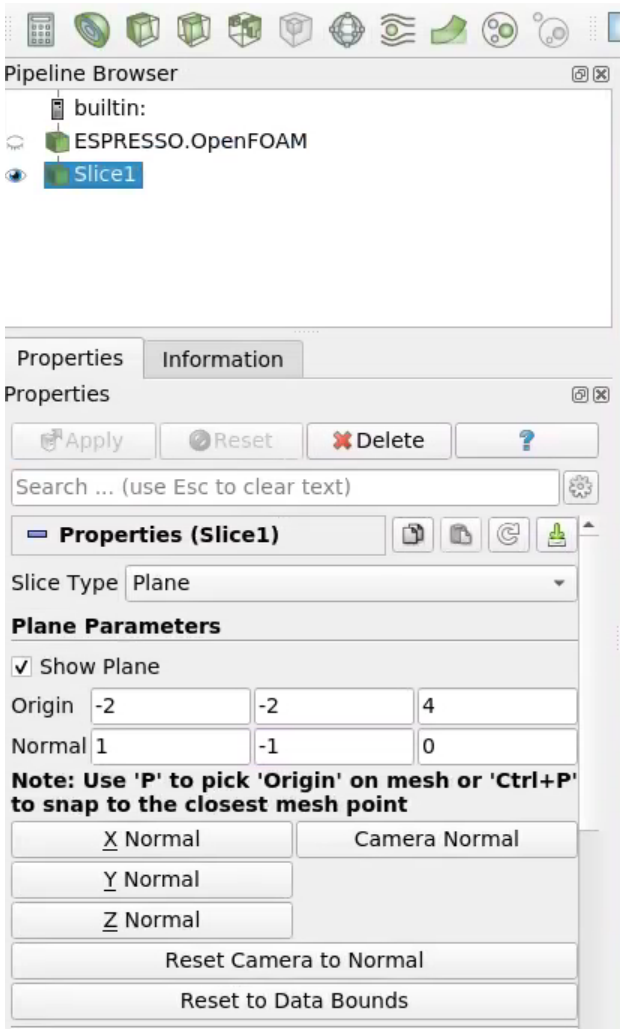
2. Change the origin and Normal

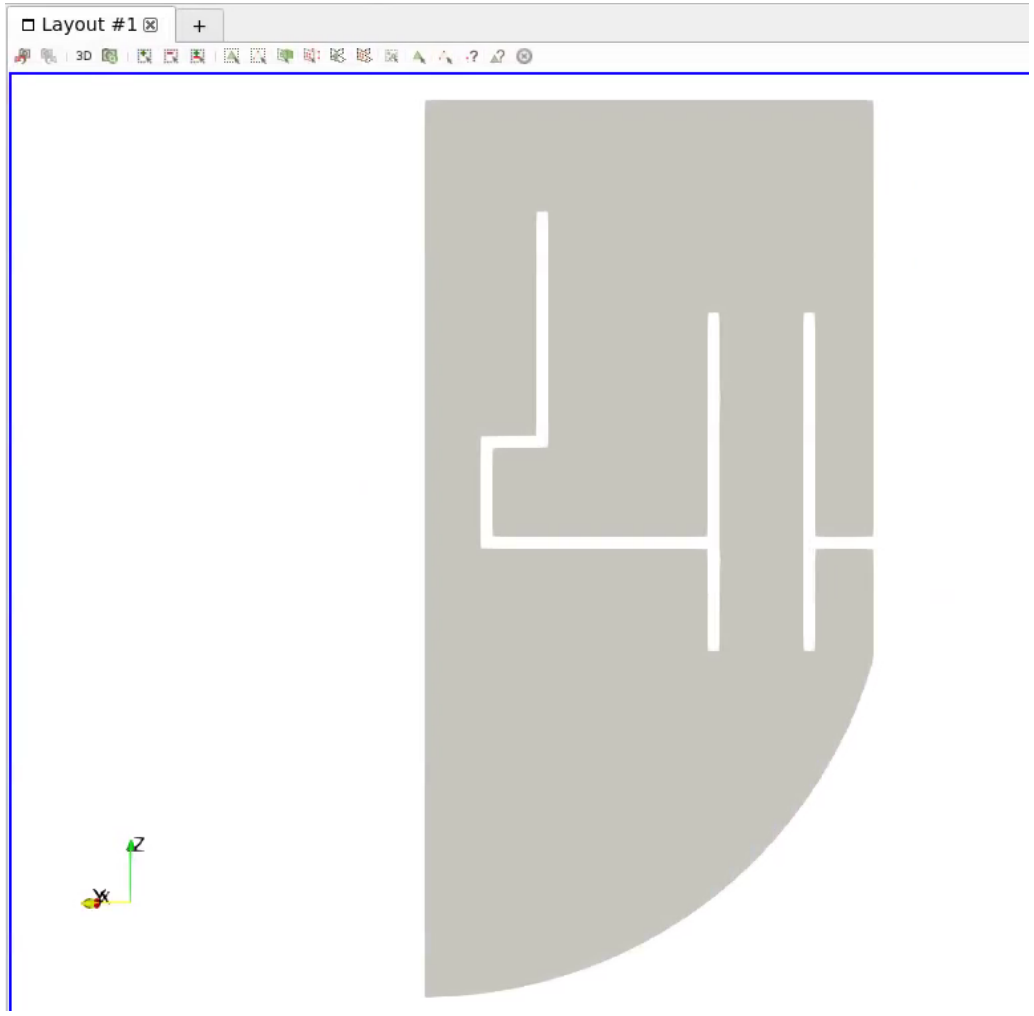
3. Unselect Show plane

4. Click on Apply

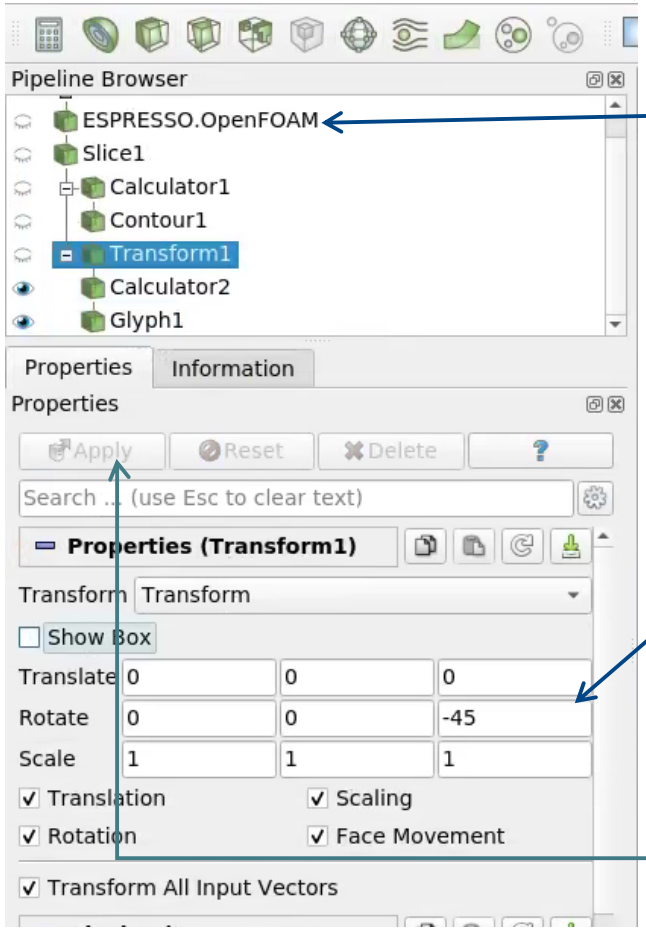
5. Adjust the 3D view

PARAFOAM – 2D SLICE





PARAFOAM – PLOT VECTORS



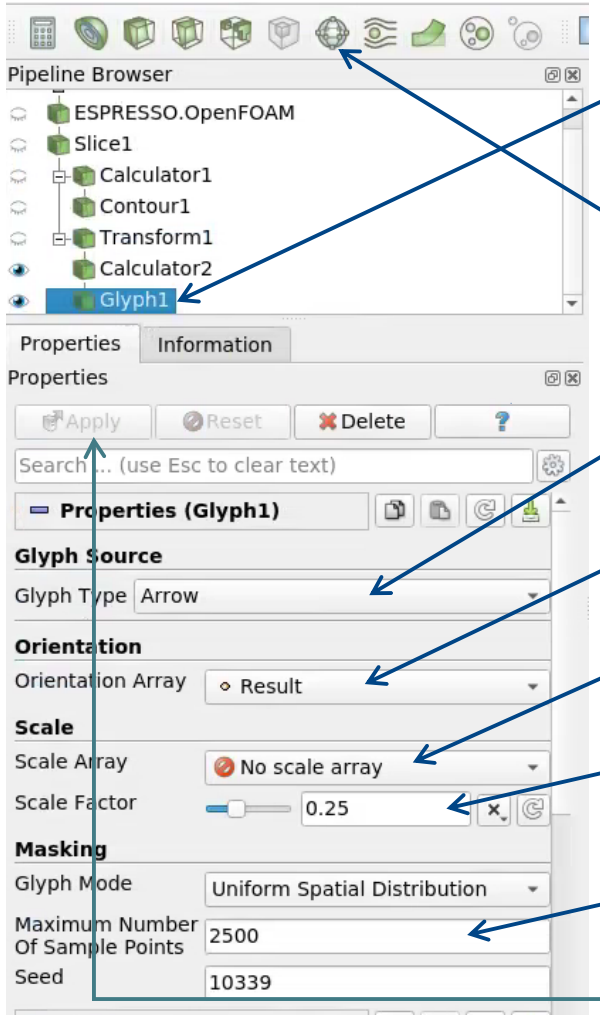
1.Hide all objects and select the root "ESPRESSO.Openfoam"

2. Create Transform object in the main menu filters→ alphabetical → Transform

Then select Rotate of -45 degrees around z-axis

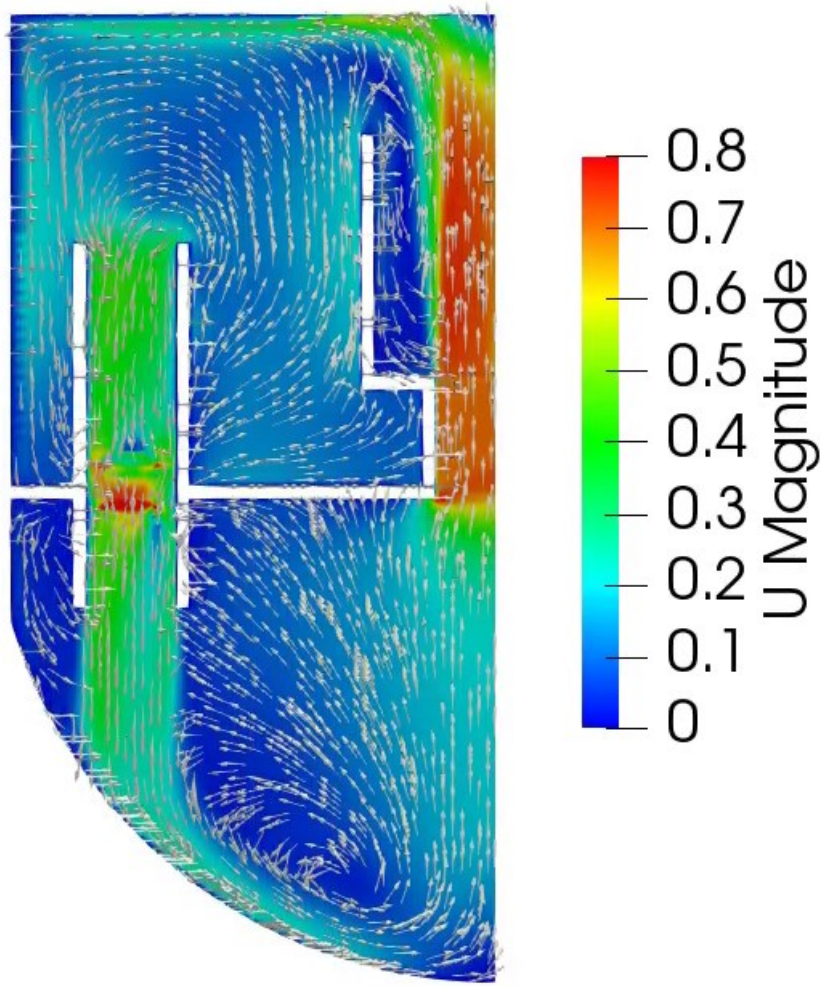
3. Apply

PARAFOAM – PLOT VECTORS



1. Create new field with the projected velocity on the slice with Calculator with the formula U_X*iHat+U_Z*kHat+0*jHat
2. Create a Gliph object
3. Select Arrow
4. Select the new field
5. Select “No scale array”
6. Change scale factor to 0.25
7. Change number of points
8. Apply

PARAFOAM – PLOT VECTORS



Questions?