Heat transfer studies in electric generators for hydropower using OpenFOAM (Foam-extend 4.0)

Bercelay Niebles Atencio

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2017

Current status

- Design phase of electric generatos relies on empirical correlations (network models).

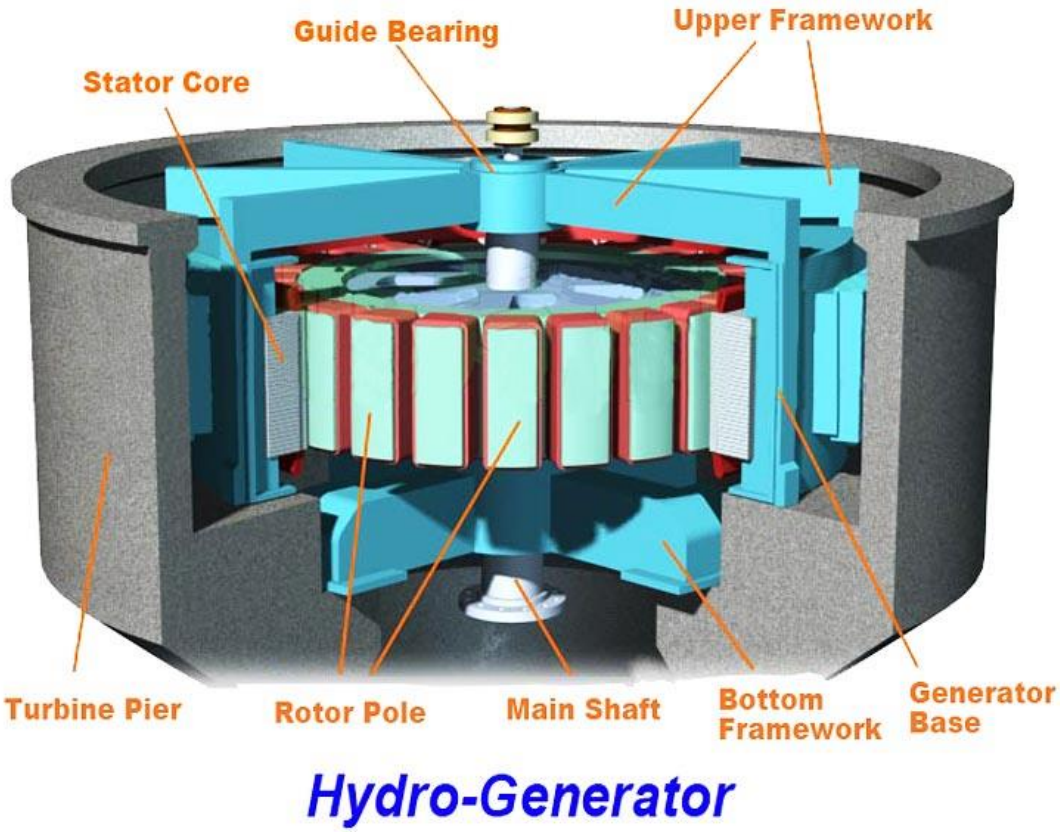
- Large uncertainties.

- Flow of cooling air have been studied in the past.

- Many studies focused on flow, thermal models, few experiments.

- To the best of knowledge, nothing done in openFOAM regarding heat transfer in hydrogenerators.

Electric Generators for Hydropower

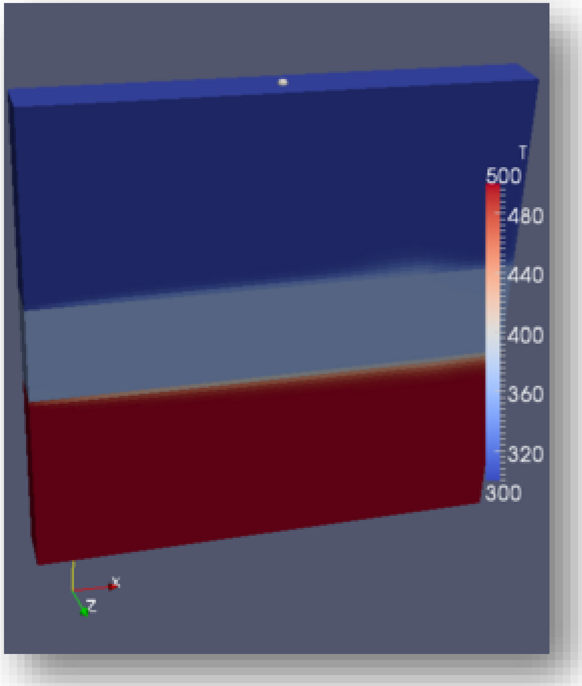


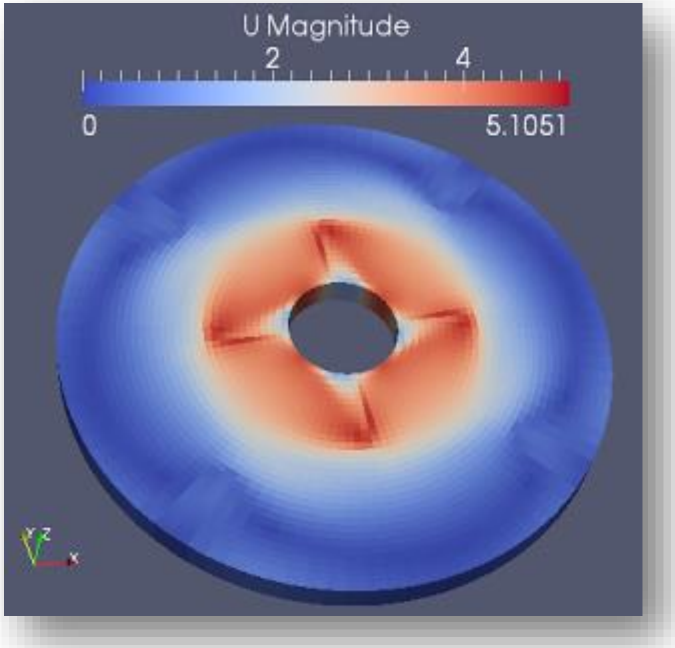
Sketch of an electric generator for hydropower.
Source: http://www.eternoohydro.com/hydro-generator/

Solvers in openFOAM

- Conjugate heat transfer processes:
chtMultiRegionSimpleFoam

- Rotation of fluid:
MRFSimpleFoam

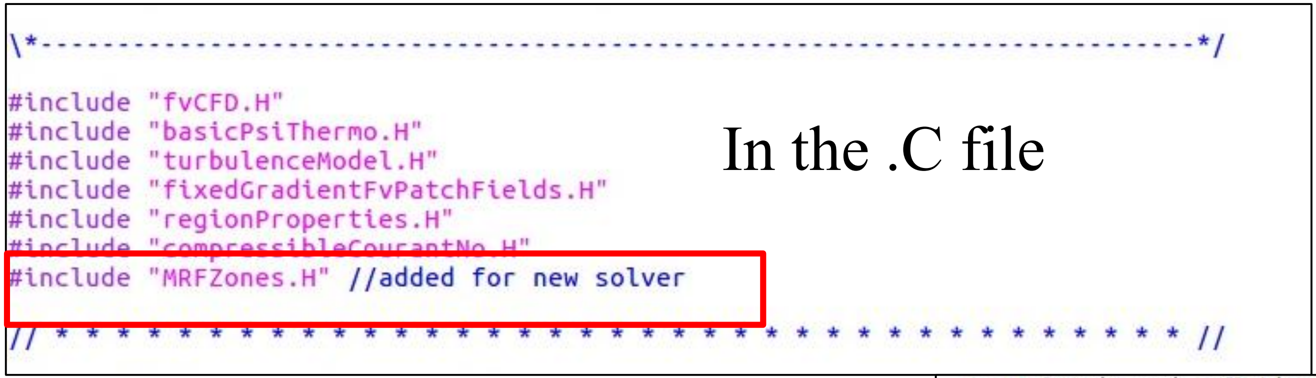




We want them to be merged!

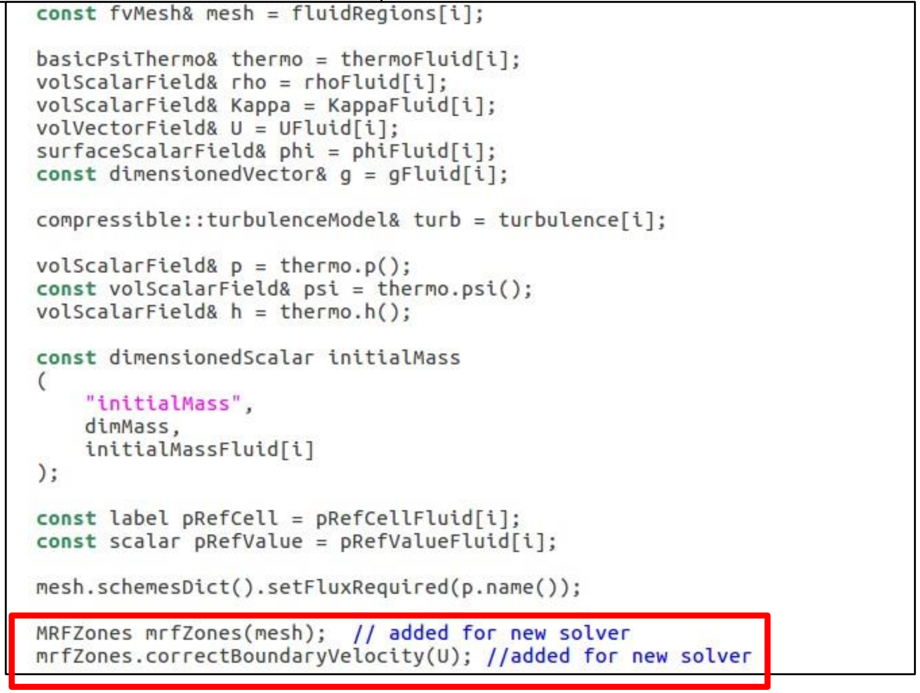
chtMRFSimpleFoam

Having chtMultiRegionSimpleFoam as the base solver



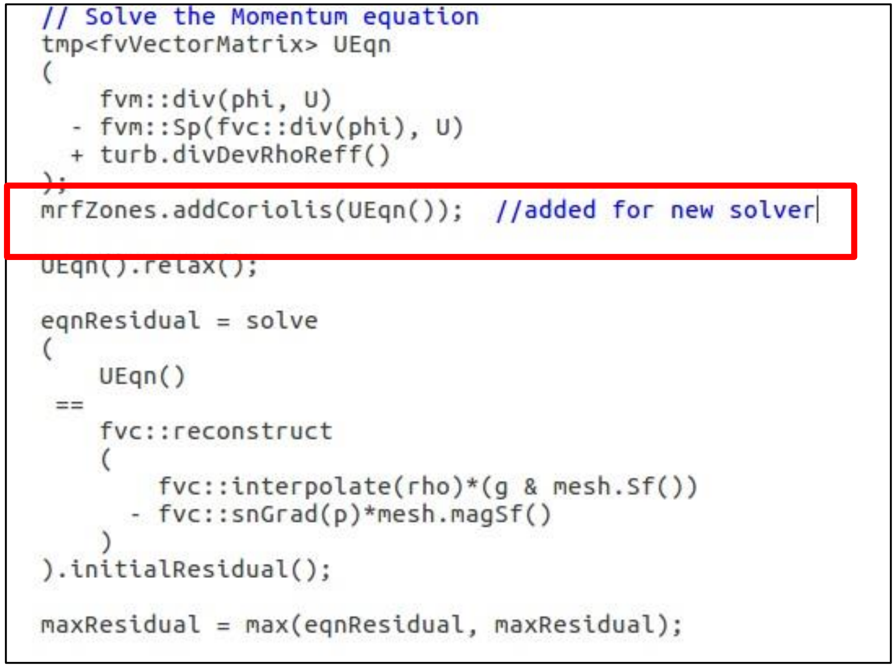
```
\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "basicPsiThermo.H"
#include "turbulenceModel.H"
#include "fixedGradientFvPatchFields.H"
#include "regionProperties.H"
#include "compressibleCourantNo.H"
#include "MRFZones.H"  //added for new solver

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

In the .C file



```
const fvMesh& mesh = fluidRegions[i];

basicPsiThermo& thermo = thermoFluid[i];
volScalarField& rho = rhoFluid[i];
volScalarField& Kappa = KappaFluid[i];
volVectorField& U = UFluid[i];
surfaceScalarField& phi = phiFluid[i];
const dimensionedVector& g = gFluid[i];

compressible::turbulenceModel& turb = turbulence[i];

volScalarField& p = thermo.p();
const volScalarField& psi = thermo.psi();
volScalarField& h = thermo.h();

const dimensionedScalar initialMass
(
    "initialMass",
    dimMass,
    initialMassFluid[i]
);

const label pRefCell = pRefCellFluid[i];
const scalar pRefValue = pRefValueFluid[i];

mesh.schemesDict().setFluxRequired(p.name());

MRFZones mrfZones(mesh);  // added for new solver
mrfZones.correctBoundaryVelocity(U); //added for new solver
```

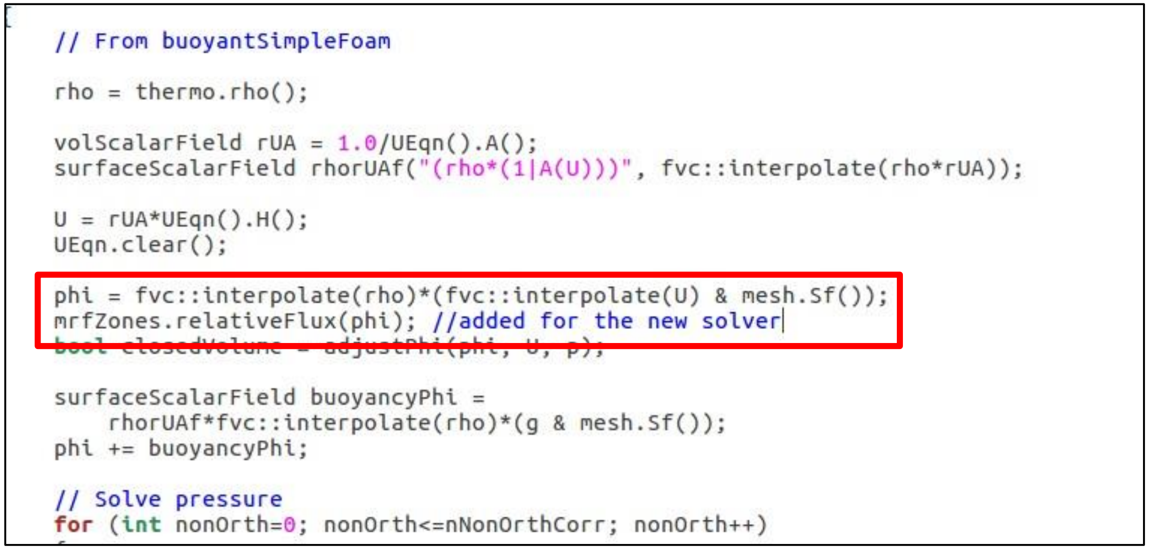
In the createFluidFields file



```
// Solve the Momentum equation
tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
  - fvm::Sp(fvc::div(phi), U)
  + turb.divDevRhoReff()
);
mrfZones.addCoriolis(UEqn());  //added for new solver
UEqn().relax();

eqnResidual = solve
(
    UEqn()
 ==
    fvc::reconstruct
    (
        fvc::interpolate(rho)*(g & mesh.Sf())
      - fvc::snGrad(p)*mesh.magSf()
    )
).initialResidual();

maxResidual = max(eqnResidual, maxResidual);
```

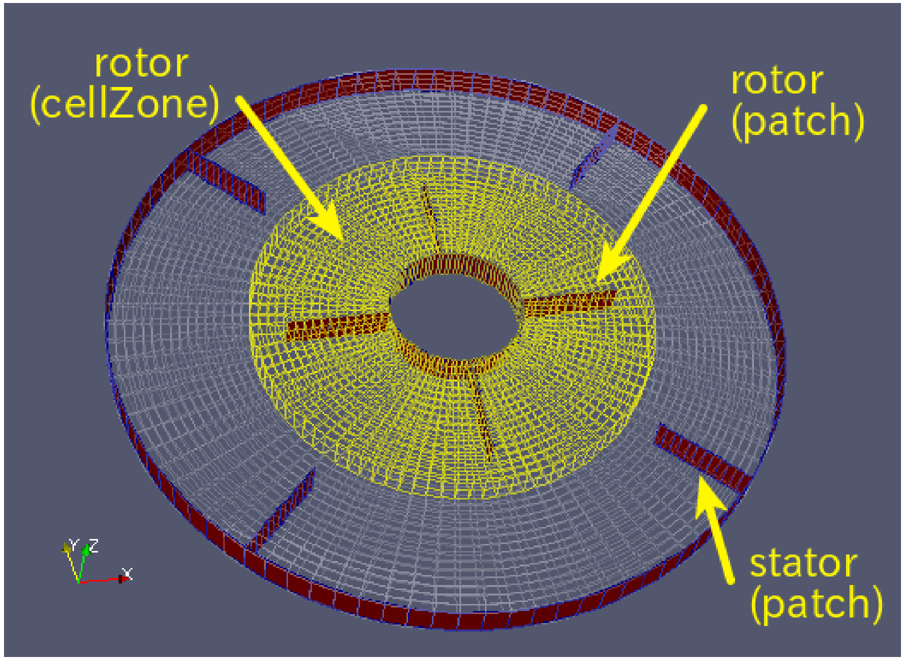
In the Ueqn.

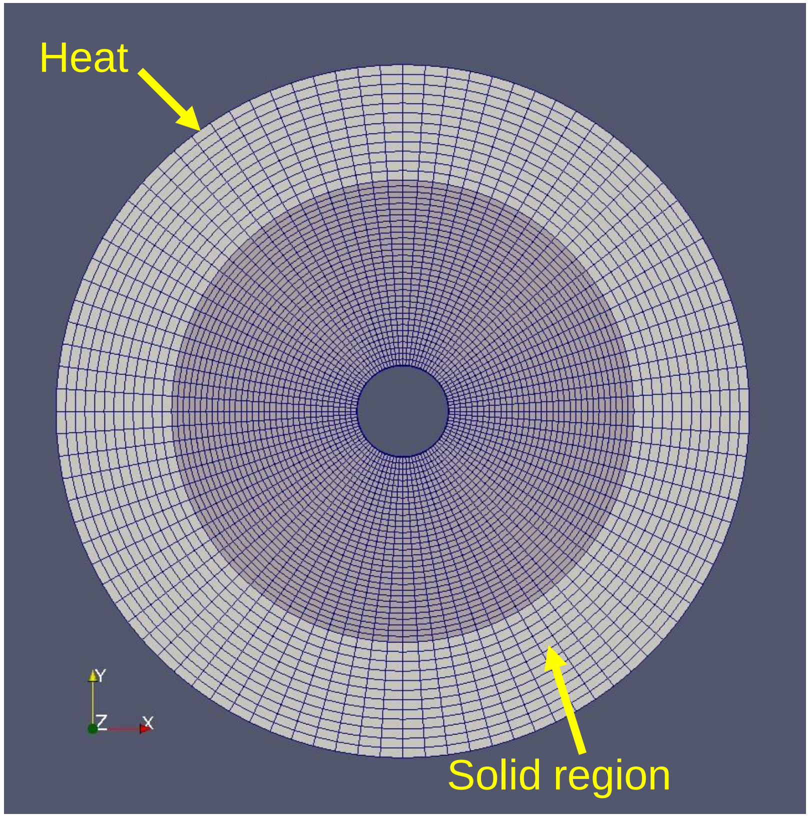
In the pEqn.

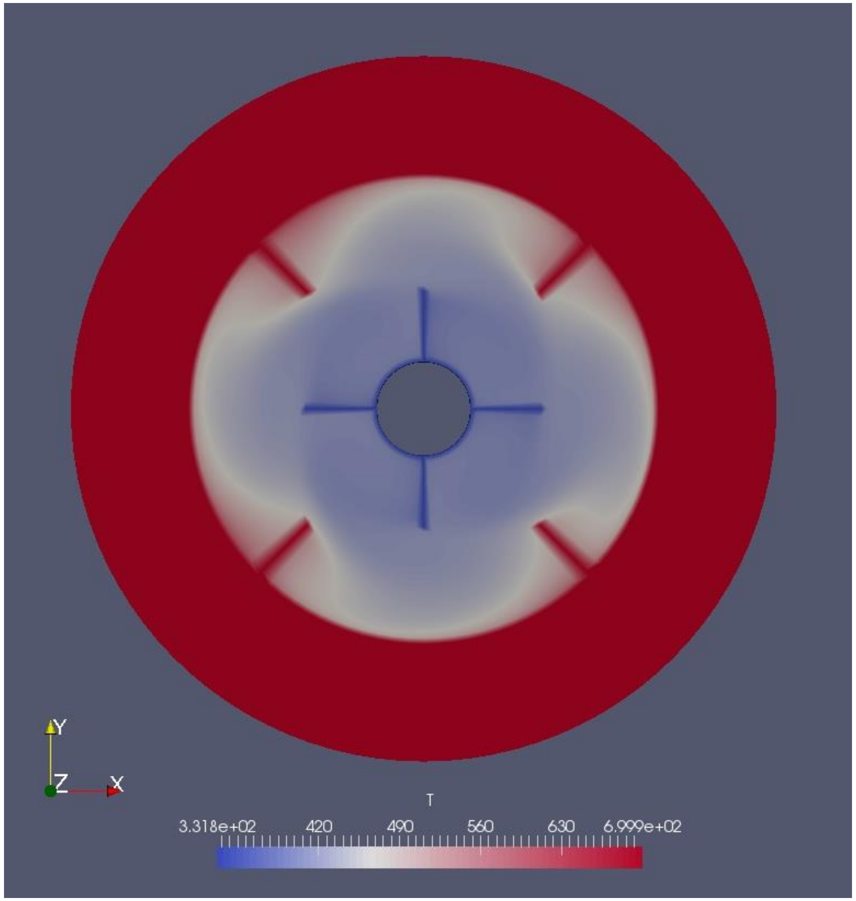


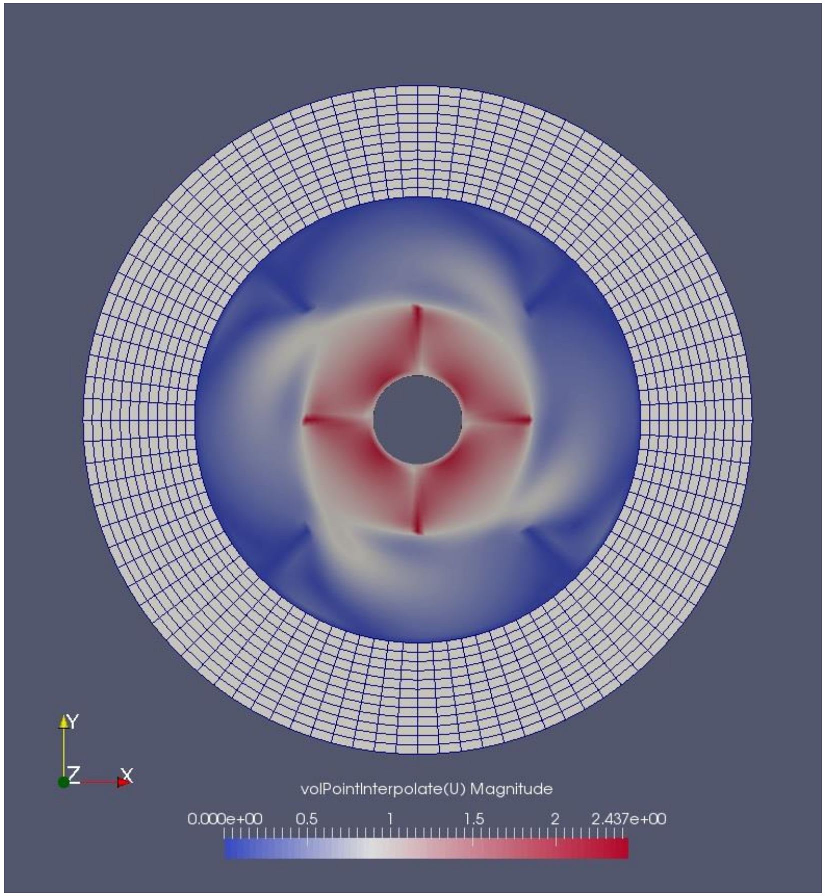
```
{
    // From buoyantSimpleFoam

    rho = thermo.rho();

    volScalarField rUA = 1.0/UEqn().A();
    surfaceScalarField rhorUAf("(rho*(1|A(U)))", fvc::interpolate(rho*rUA));

    U = rUA*UEqn().H();
    UEqn.clear();

    phi = fvc::interpolate(rho)*(fvc::interpolate(U) & mesh.Sf());
    mrfZones.relativeFlux(phi); //added for the new solver
    bool closedVolume = adjustPhi(phi, U, p);

    surfaceScalarField buoyancyPhi =
        rhorUAf*fvc::interpolate(rho)*(g & mesh.Sf());
    phi += buoyancyPhi;

    // Solve pressure
    for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
    {
```

Testing chtMRFSimpleFoam



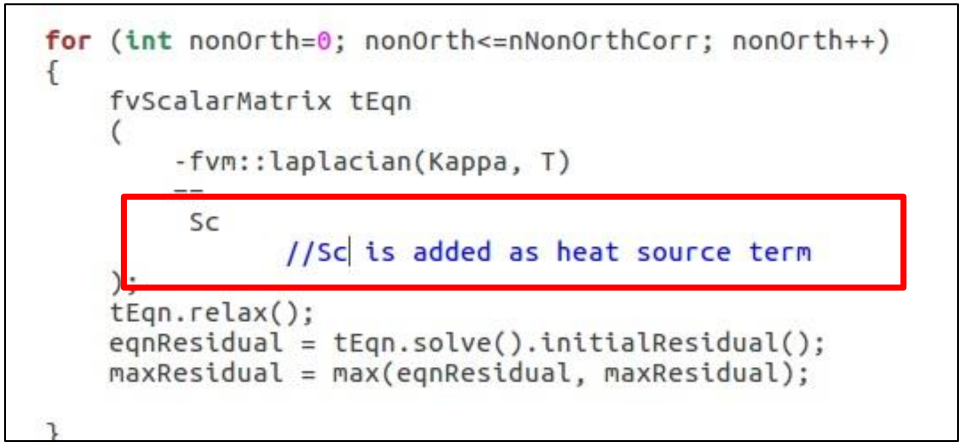



Y
Z
X
T
3.318e+02
420
490
560
630
6.999e+02


Y
Z
X
volPointInterpolate(U) Magnitude
0.000e+00
0.5
1
1.5
2
2.437e+00

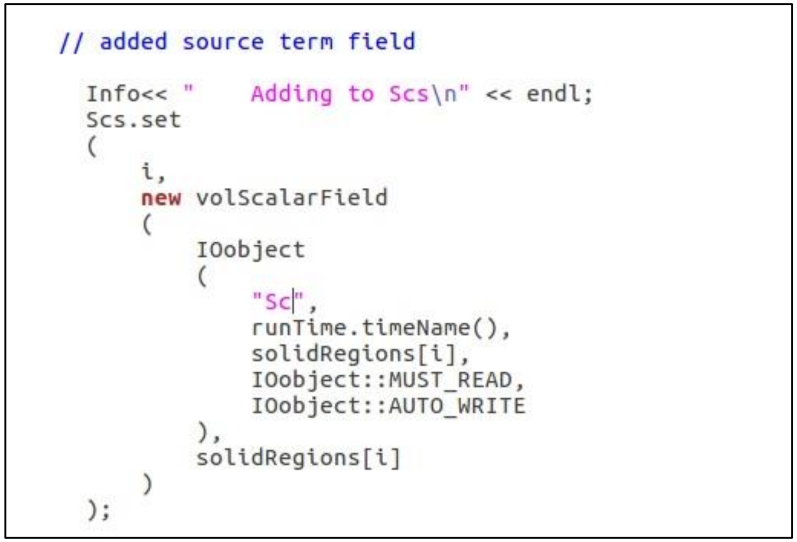
chtSourceMRFSimpleFoam

Having chtMRFSimpleFoam as the base solver



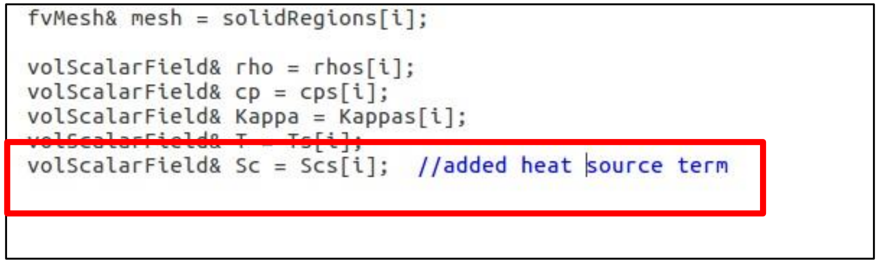
```
for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
{
    fvScalarMatrix tEqn
    (
        -fvm::laplacian(Kappa, T)
        ==
         Sc
               //Sc is added as heat source term
    );
    tEqn.relax();
    eqnResidual = tEqn.solve().initialResidual();
    maxResidual = max(eqnResidual, maxResidual);

}
```

In the solveSolids.H file



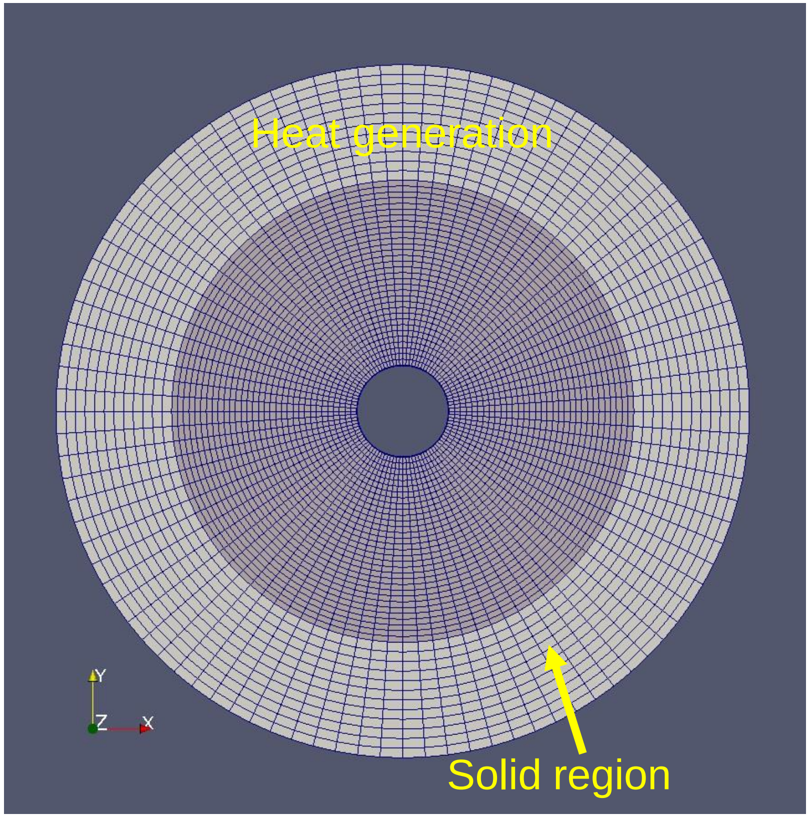
```
  // added source term field

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

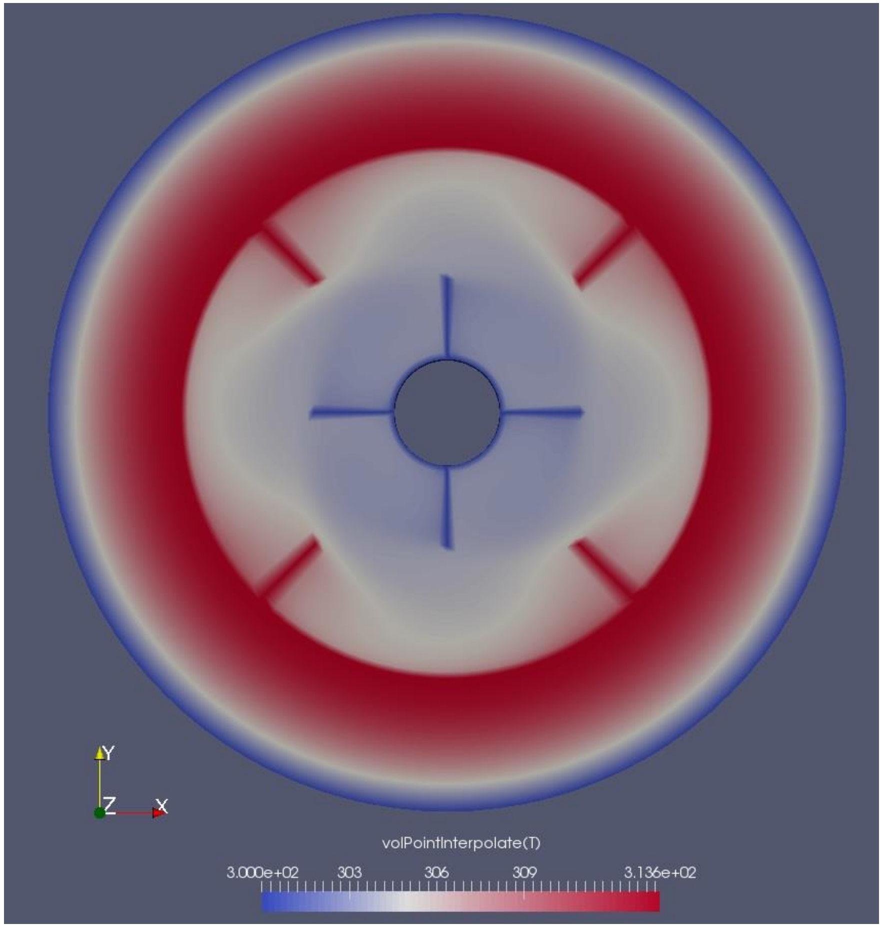
In the createSolidFields file



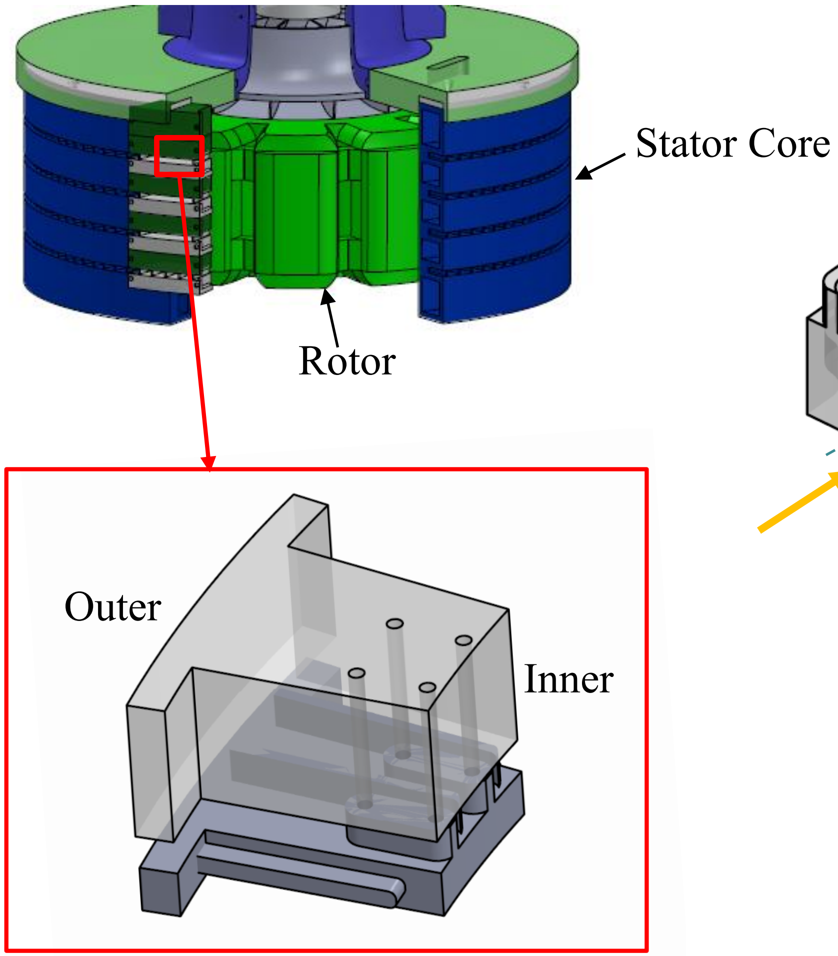
```
fvMesh& mesh = solidRegions[i];

volScalarField& rho = rhos[i];
volScalarField& cp = cps[i];
volScalarField& Kappa = Kappas[i];
volScalarField& T = Ts[i];
volScalarField& Sc = Scs[i];  //added heat source term
```

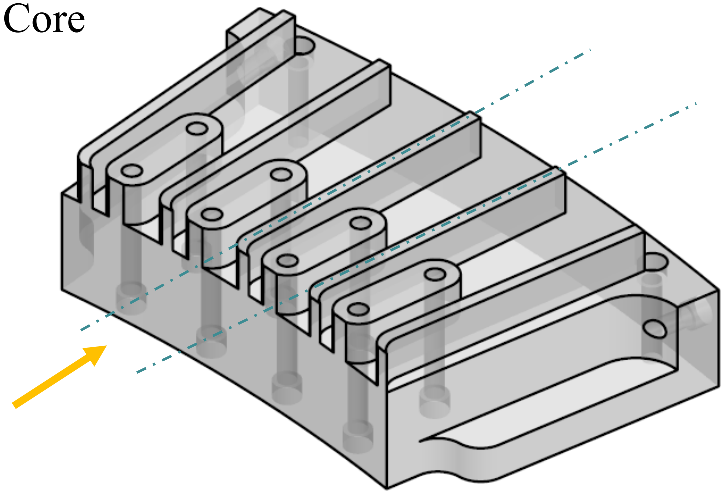
In the setSolidFields file


Heat generation
Solid region
Y
Z
X

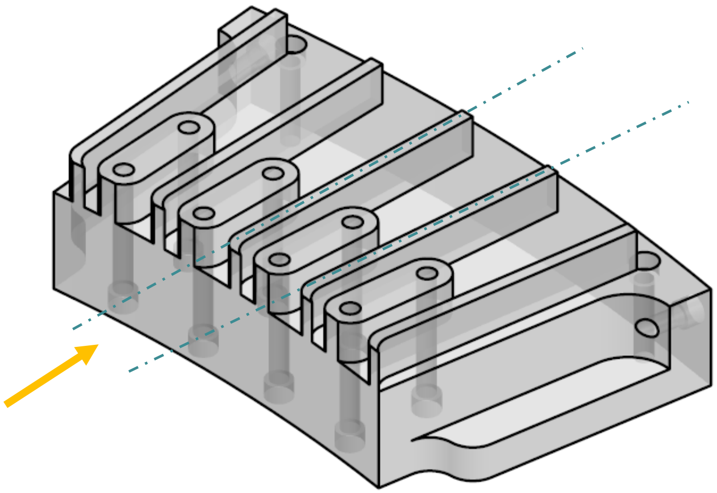

Y
Z
X
volPointInterpolate(T)
3.000e+02
303
306
309
3.136e+02

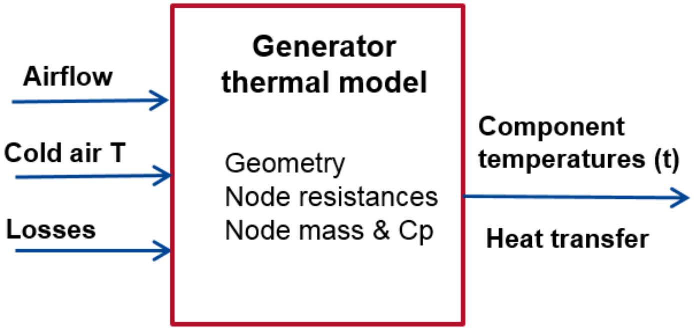
Network Models





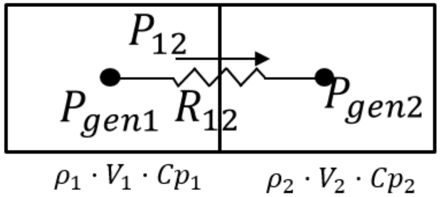
-One slot
-Heat generation (coil, core)
-No rotation
-Periodicity and symmetry
-kOmegaSST

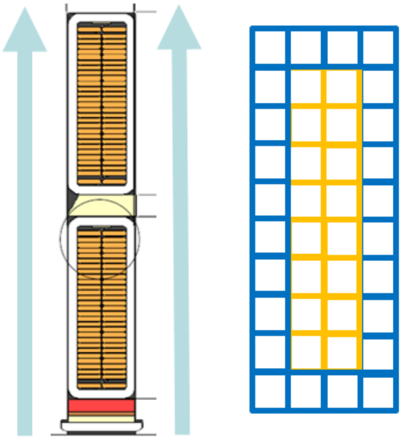


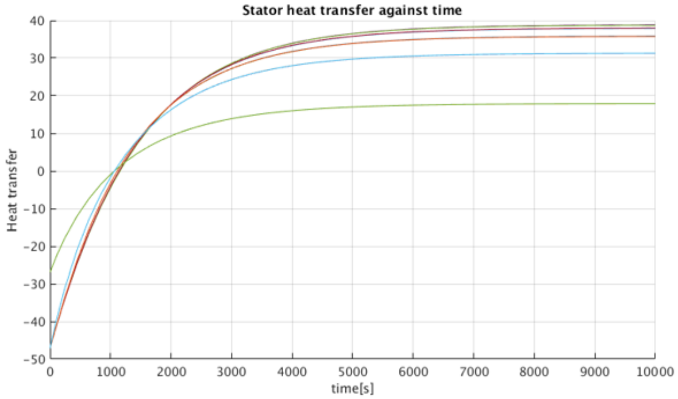


Generator thermal model

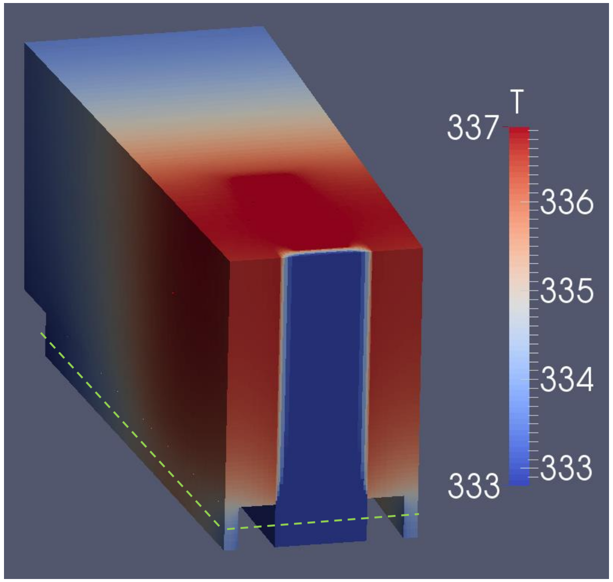
Geometry
Node resistances
Node mass & Cp

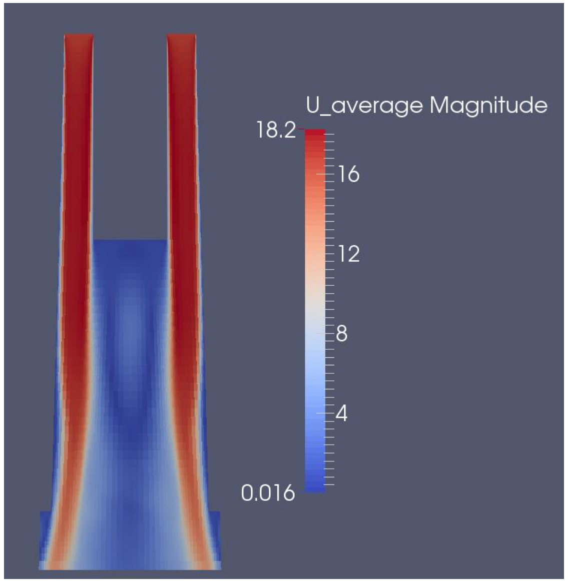


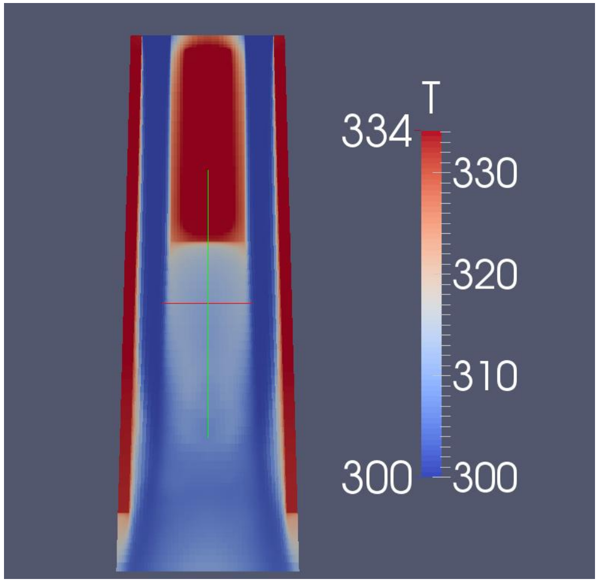




CFD for Network Model Case

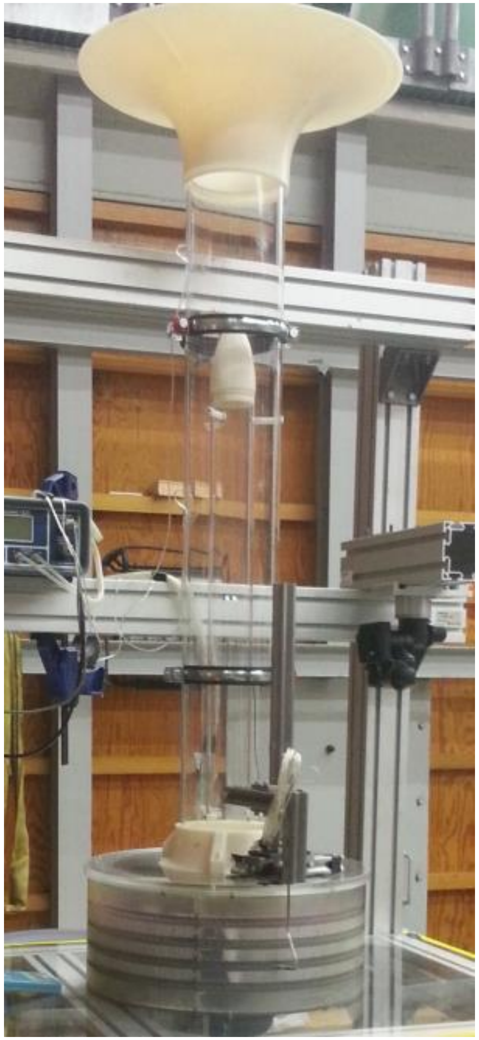


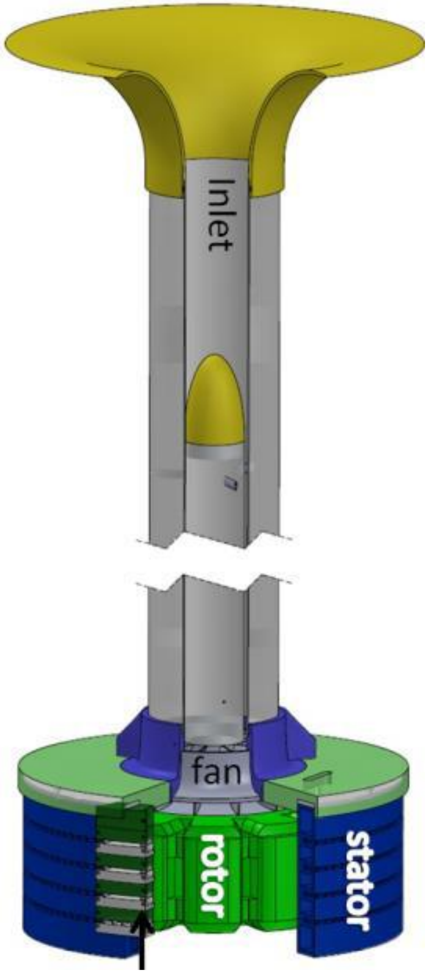




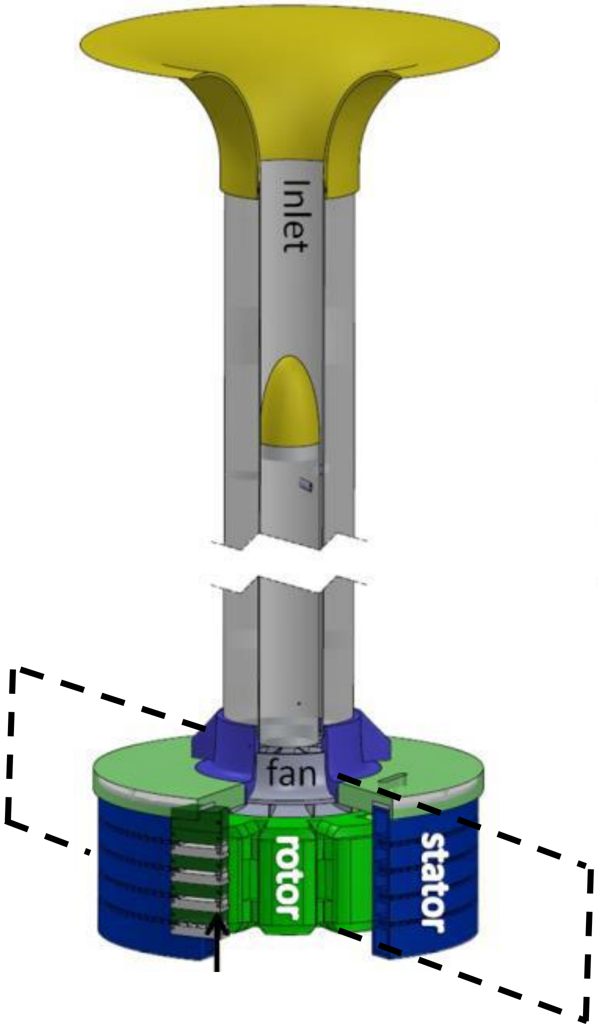
Re	Nusselt Number		
	CFD	Gnielinksi	Dittus-Boelter
3152	15.7	7.9	12
3520	16.8	9.1	13
3960	17.5	10.4	14.3

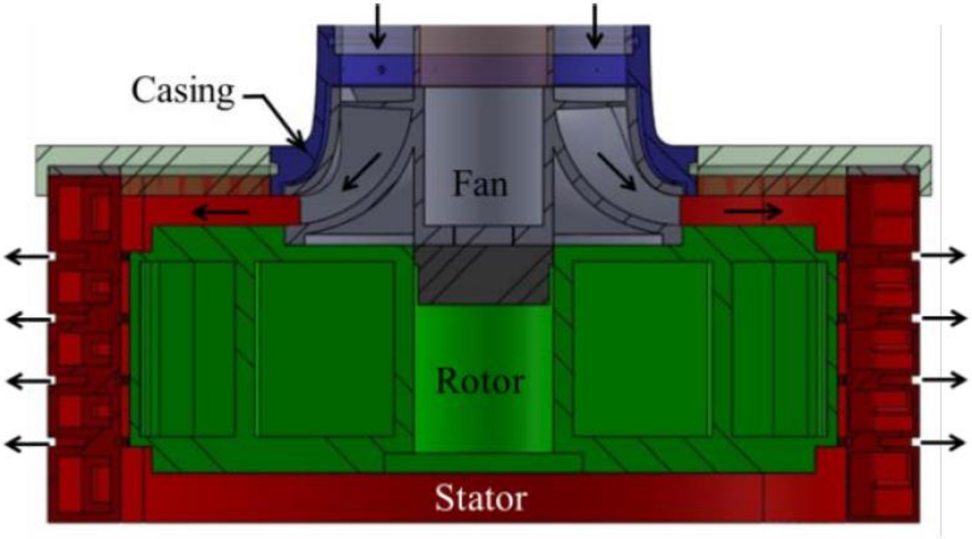
Hydro-generator model experimental rig



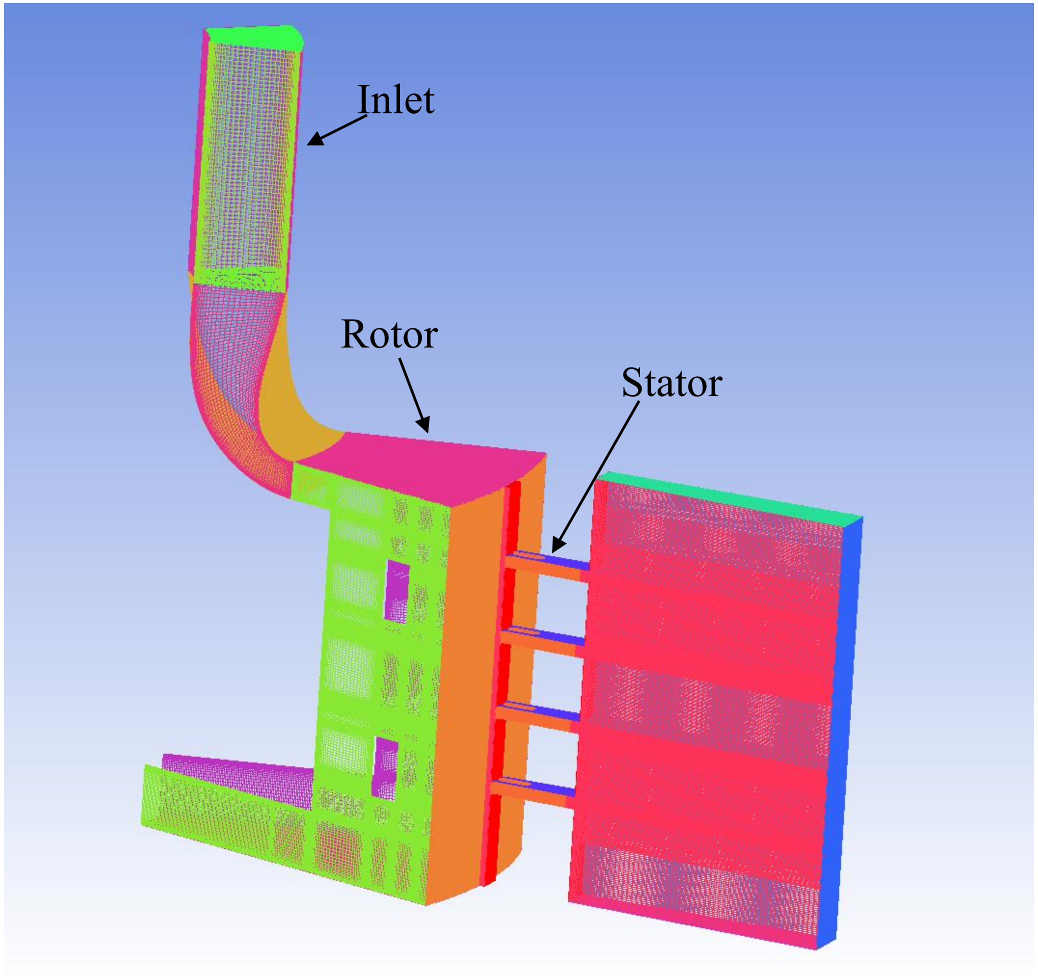


Heat transfer simulations



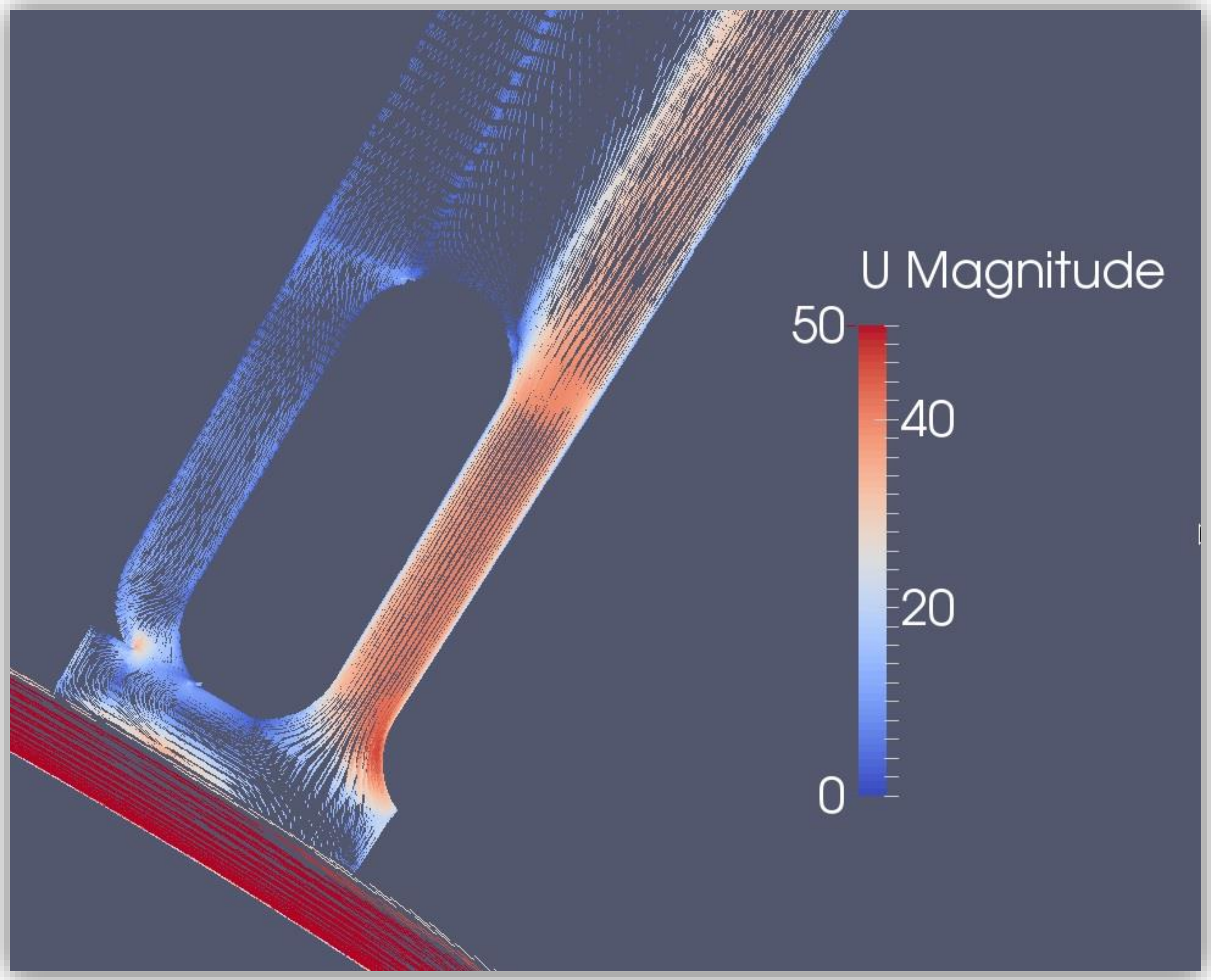


Simulations for the hydro-generator model



-Inlet, rotor, stator cellZones
-kOmegaSST – low Re
-MRFSimpleFoam + heat
-Periodic
-Mixing Plane
-Flow field
-Heat transfer in stator channels

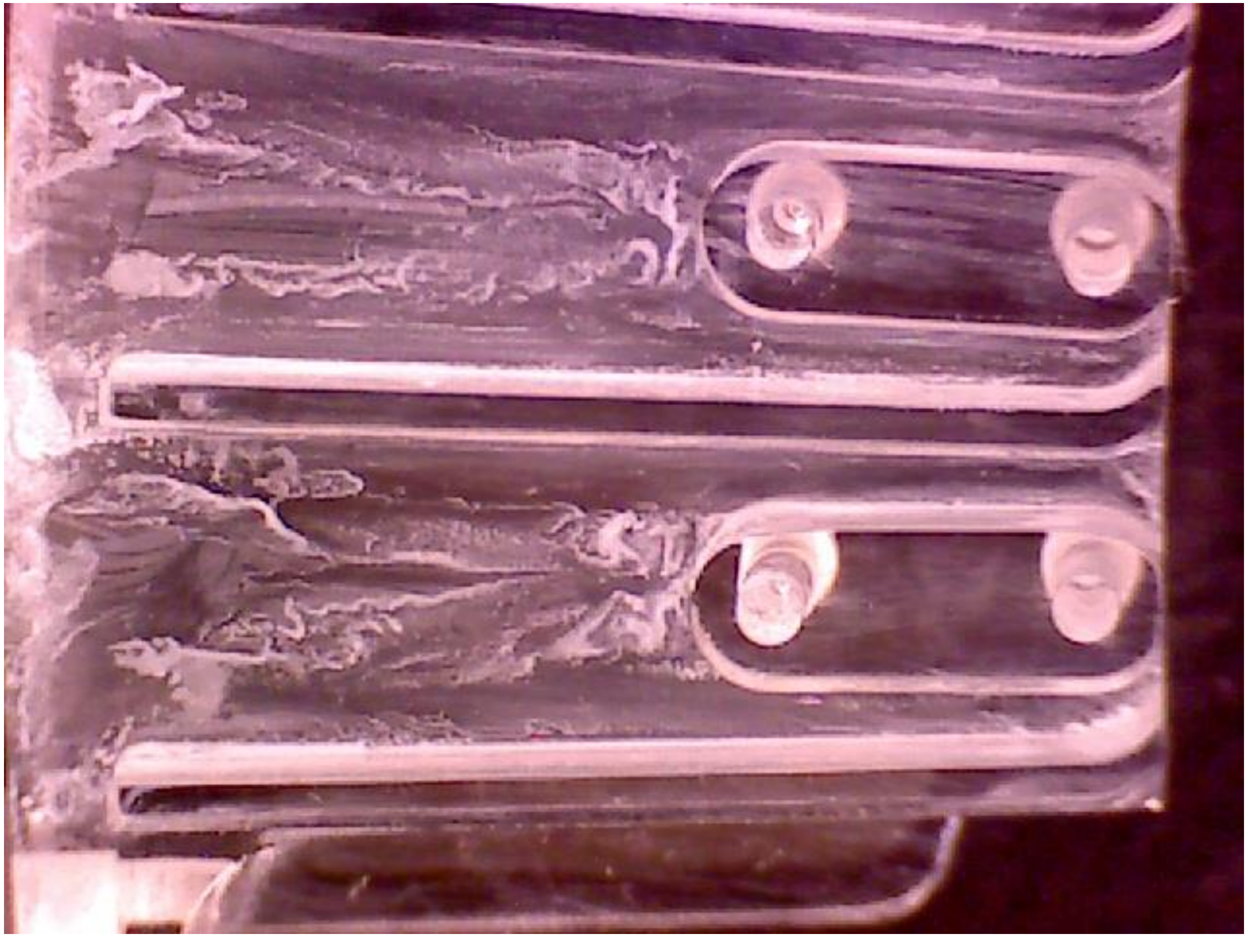
Simulations



Preliminary U field @ 10000 iterations (still running)

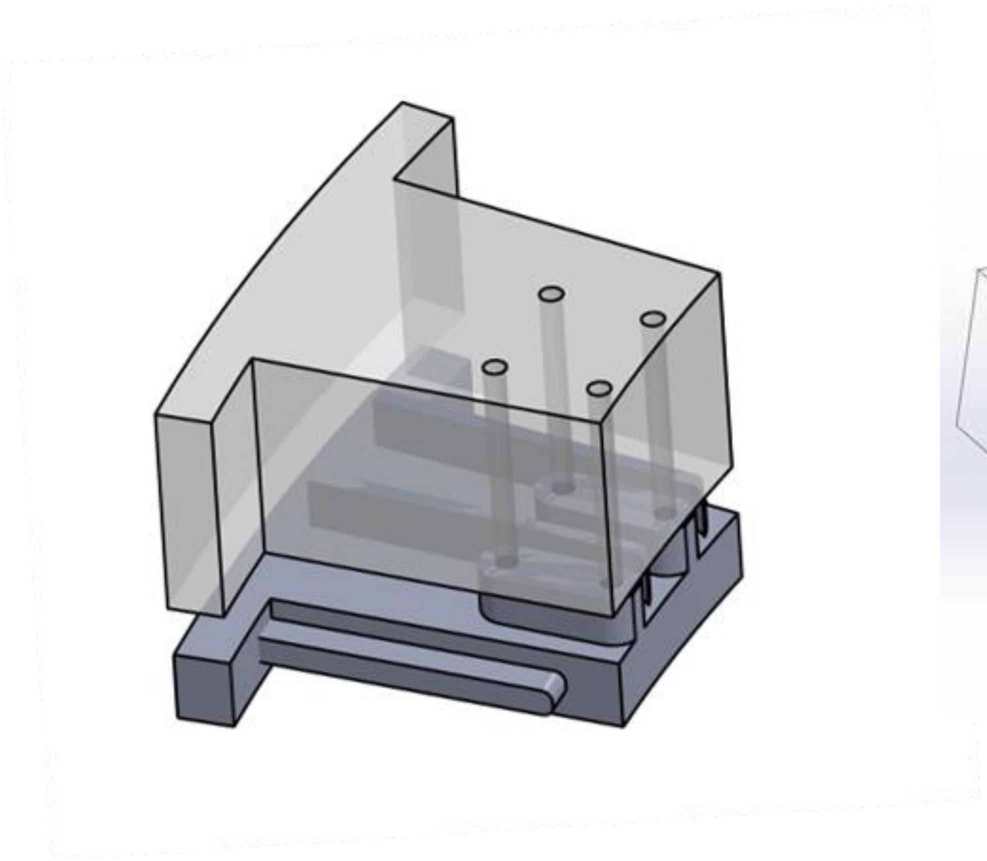
Experiments

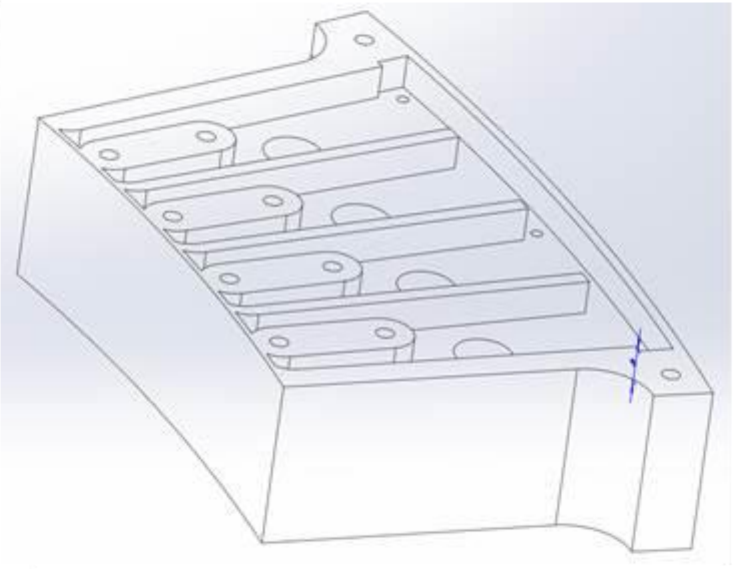
- Oil visualizations



Experiments

- Naphthalene Sublimation





Thank you!