TAMPERE UNIVERSITY OF TECHNOLOGY

Modelling of centrifugal fan flow with OpenFOAM

Finnish OpenFOAM Users Day 2014

Pekka Pasanen

Contents of the presentation

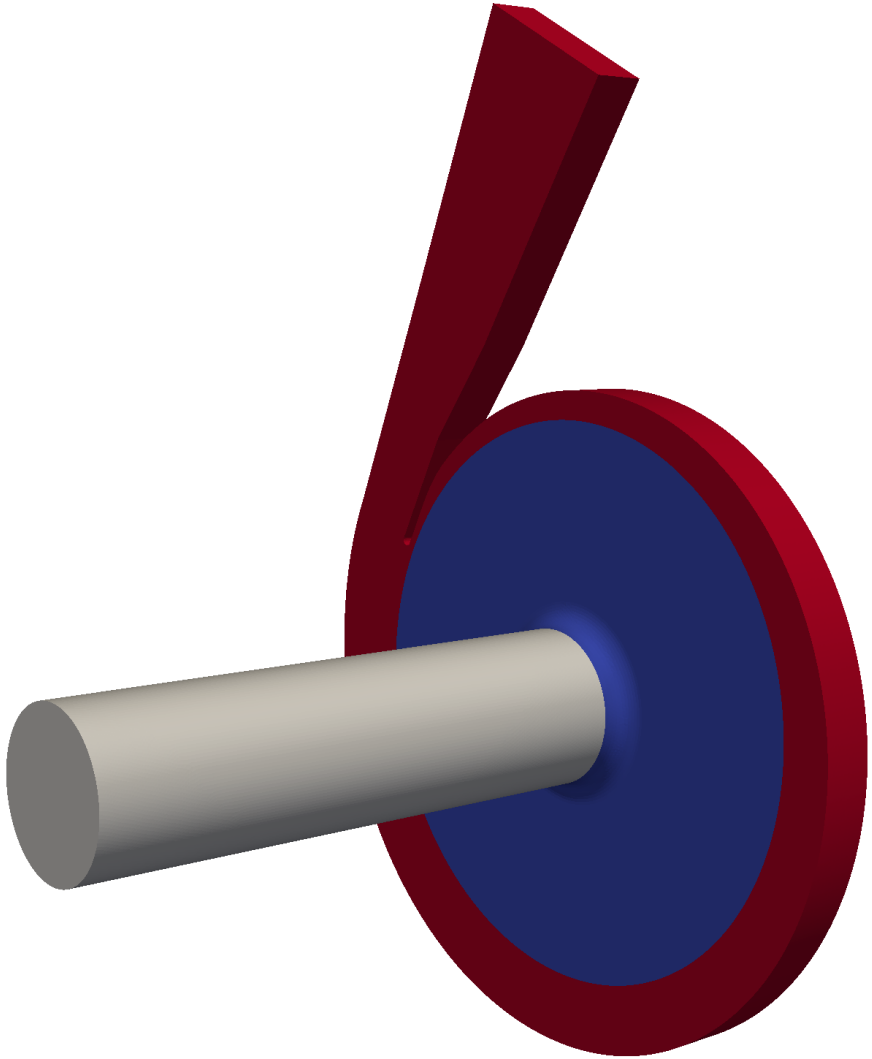
- Research background
- Case setup (Meshing, AMI, MRF)
- Calculation of turbomachinery variables
- Results
- Conclusions

Research background

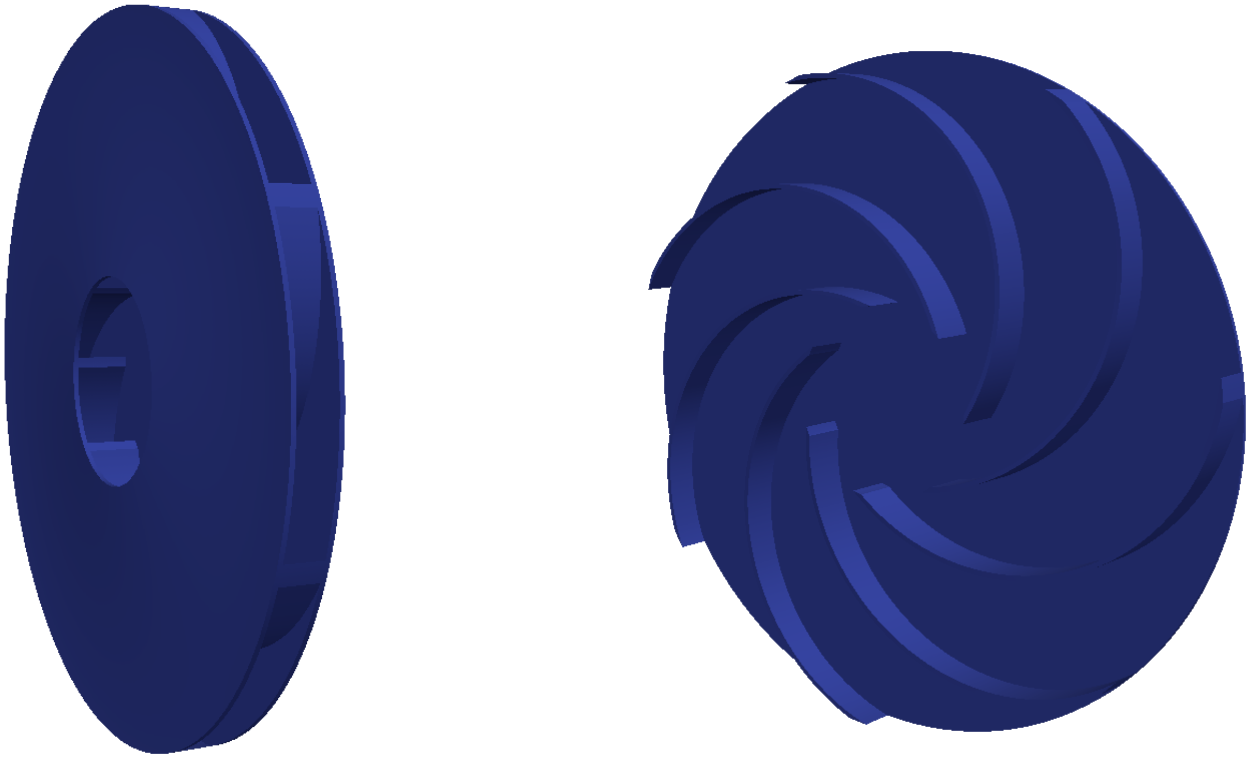
- Top level goal: Increasing the efficiency of small n_q turbomachinery
- With small n_q small things are relatively more important – rotor-volute interaction, disc friction losses, leakage losses etc.
- Need to study the effect of common CFD simplifications and find the “good enough” calculation method

Case setup - Geometry

- Single stage centrifugal fan, 2D blades
- $n_q = 15$, $Q_{opt} = 0{,}108\ \mathrm{m}^3/\mathrm{s}$, $H_{opt} = 265\ \mathrm{m}$
- $d_1 = 125\ \mathrm{mm}$, $d_2 = 440\ \mathrm{mm}$, $n = 3000\ \mathrm{rpm}$
- Inclusion of “all parts” and full domain
- Three separate (mesh) regions
  - Inlet pipe
  - Rotor
  - Volute (+ diffuser)



Mesh zones



Rotor details

Case setup - Meshing

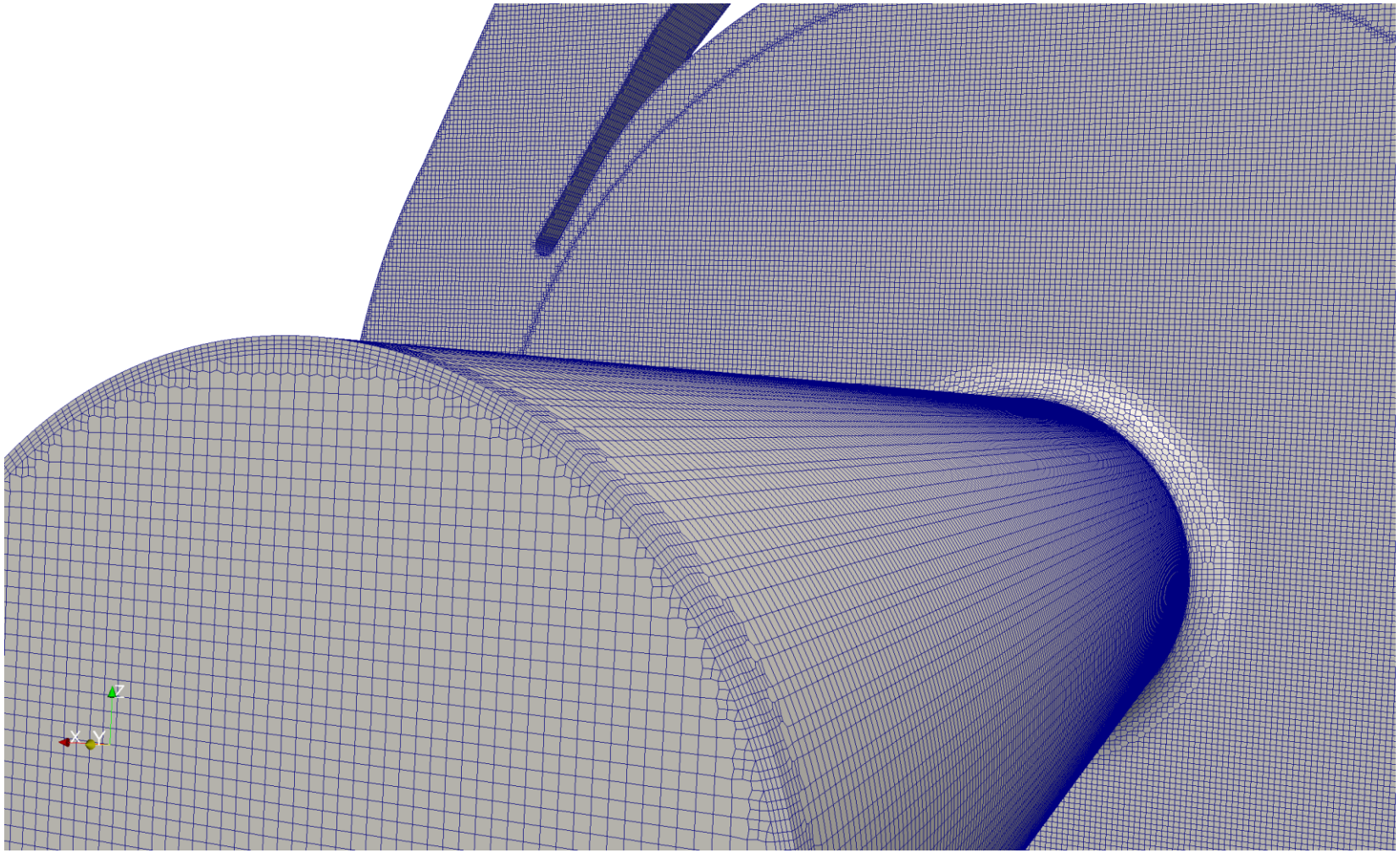
- Each region is meshed separately with snappyHexMesh
- mergeMeshes is used (twice) to combine regions (all patches are preserved)
- Total mesh size is ~6,7 million cells
- AMI is used to couple different mesh regions together
- splitMeshRegions -makeCellZones can be used afterwards to create cellZones

Case setup - AMI

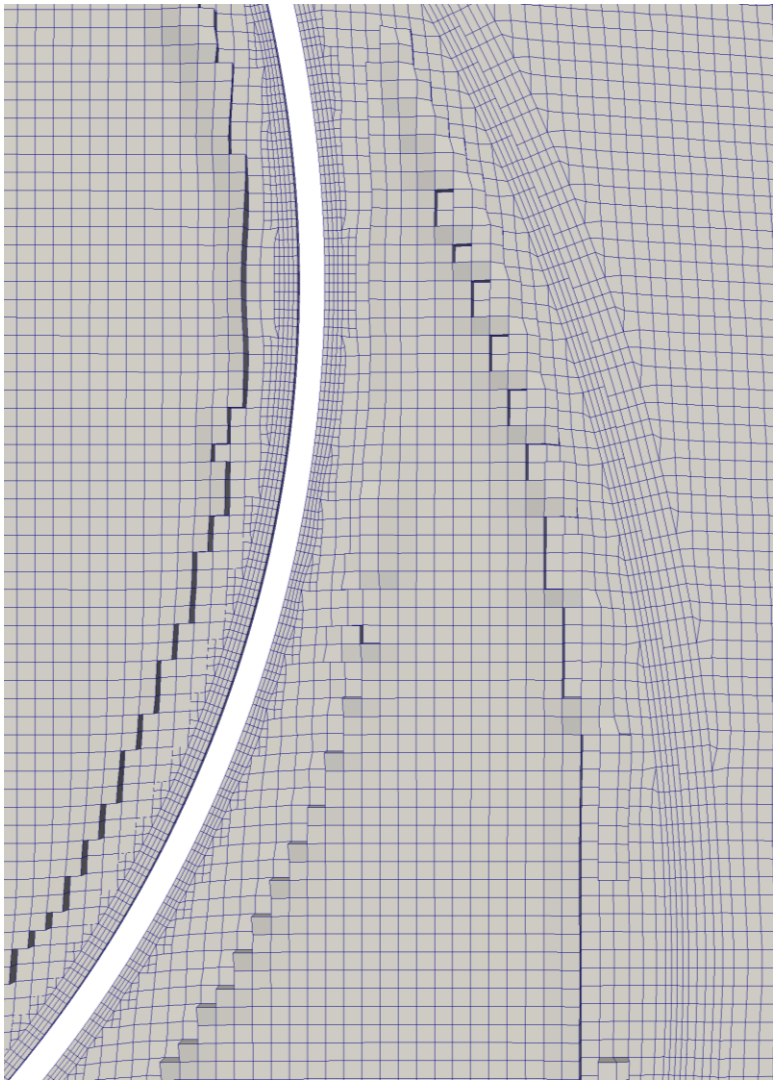
- AMI – Arbitrary Mesh Interface
- Can be used to couple different mesh regions (and meshes) together – very easy to use!, file `constant/polyMesh/boundary`

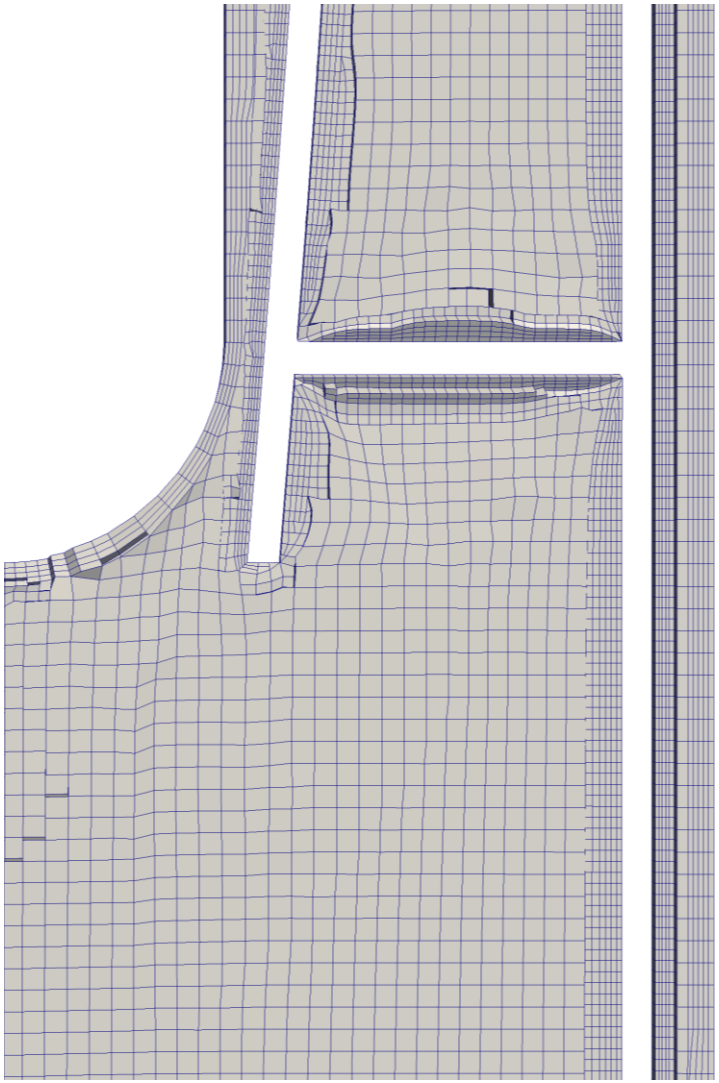
```
rotor_inlet
{
    type            cyclicAMI;
    inGroups        1(cyclicAMI);
    nFaces          4992;
    startFace       20570542;
    matchTolerance  0.0001;
    transform       noOrdering;
    neighbourPatch  pipe_outlet;
}
```

```
pipe_outlet
{
    type            cyclicAMI;
    inGroups        1(cyclicAMI);
    nFaces          4976;
    startFace       20687748;
    matchTolerance  0.0001;
    transform       noOrdering;
    neighbourPatch  rotor_inlet;
}
```



Mesh overview





Mesh details

Case setup - MRF

- Velocity components are solved using the flux relative to the rotation of the local frame of reference, file `system/fvOptions`

```
MRF1
{
    type            MRFSource;
    active          true;
    selectionMode   cellZone;
    cellZone        region0;

    MRFSourceCoeffs
    {
        nonRotatingPatches (rotor_inlet pipe_outlet rotor_outlet
                            volute_inlet shroud_front shroud_back);
        origin      (0 0 0);
        axis        (0 1 0);
        omega       -314.16;
    }
}
```

Case setup - Mesh motion

- Mesh motion in unsteady simulations is possible using DyM-solvers, file `constant/dynamicMeshDict`

```
dynamicFvMesh   solidBodyMotionFvMesh;

motionSolverLibs ( "libfvMotionSolvers.so" );

solidBodyMotionFvMeshCoeffs
{
    cellZone        region0;

    solidBodyMotionFunction  rotatingMotion;
    rotatingMotionCoeffs
    {
        origin         (0 0 0);
        axis           (0 1 0);
        omega          -314.16; // rad/s
    }
}
```

Case setup - Turbulence

- kOmegaSST turbulence model, robust and widely tested
- Wall functions are used:
  - omegaWallFunction
  - nutUSpaldingWallFunction
  - kqRWallFunction
- Need to use continuous wall functions, because y+ varies a lot (0,615 – 165,7)

Case setup - boundary conditions (MRF)

- Inlet:
  - p: zeroGradient
  - U: flowRateInletVelocity (Q)
  - k & ω: inletOutlet
- Outlet
  - p: fixedValue (0)
  - U: pressureInletOutletVelocity
  - k & ω : inletOutlet
- Walls
  - p: zeroGradient, U: fixedValue (0), k & ω: wallFunctions

Calculation of turbomachinery variables

- The main interests are the operation and performance curves of the turbomachine
- Operation curve: $p_f = f(Q)$
- Performance curve: $\eta_h = f(Q)$
- A lot of valuable information can be extracted:
  - Possible flow separation in blade passages
  - Leakage flows
  - Possible diffuser flow separation etc.

Calculation of variables – fan pressure

- Function objects: averageVelocity (U_{m2}), inletPressure (p_1), file `system/controlDict`
- $p_f = p_{sg2} - p_{sg1}$, $p_{sg} = p + p_d = p + \rho \frac{1}{2} U_m^2$
- $p_2 = 0$ (BC), $U_{m1} = Q/A_1$ $\rightarrow$

$$p_f = \rho \frac{1}{2} U_{m2}^2 - p_1 - \rho \frac{1}{2} \left(\frac{Q}{A_1} \right)^2$$

- REMINDER! In OpenFOAM pressure is actually p/ρ (with incompressible solvers)

Calculation of variables – fan pressure

```
inletPressure
{
    type            faceSource;
    functionObjectLibs
("libfieldFunctionObjects.so");
    log             yes;
    valueOutput     true;
    outputControl   timeStep;
    outputInterval  1;
    writeInterval   1000;
    surfaceFormat   null;
    source          patch;
    sourceName      "pipe_inlet";
    operation       areaAverage;
    fields
    (
        p
    );
}
```

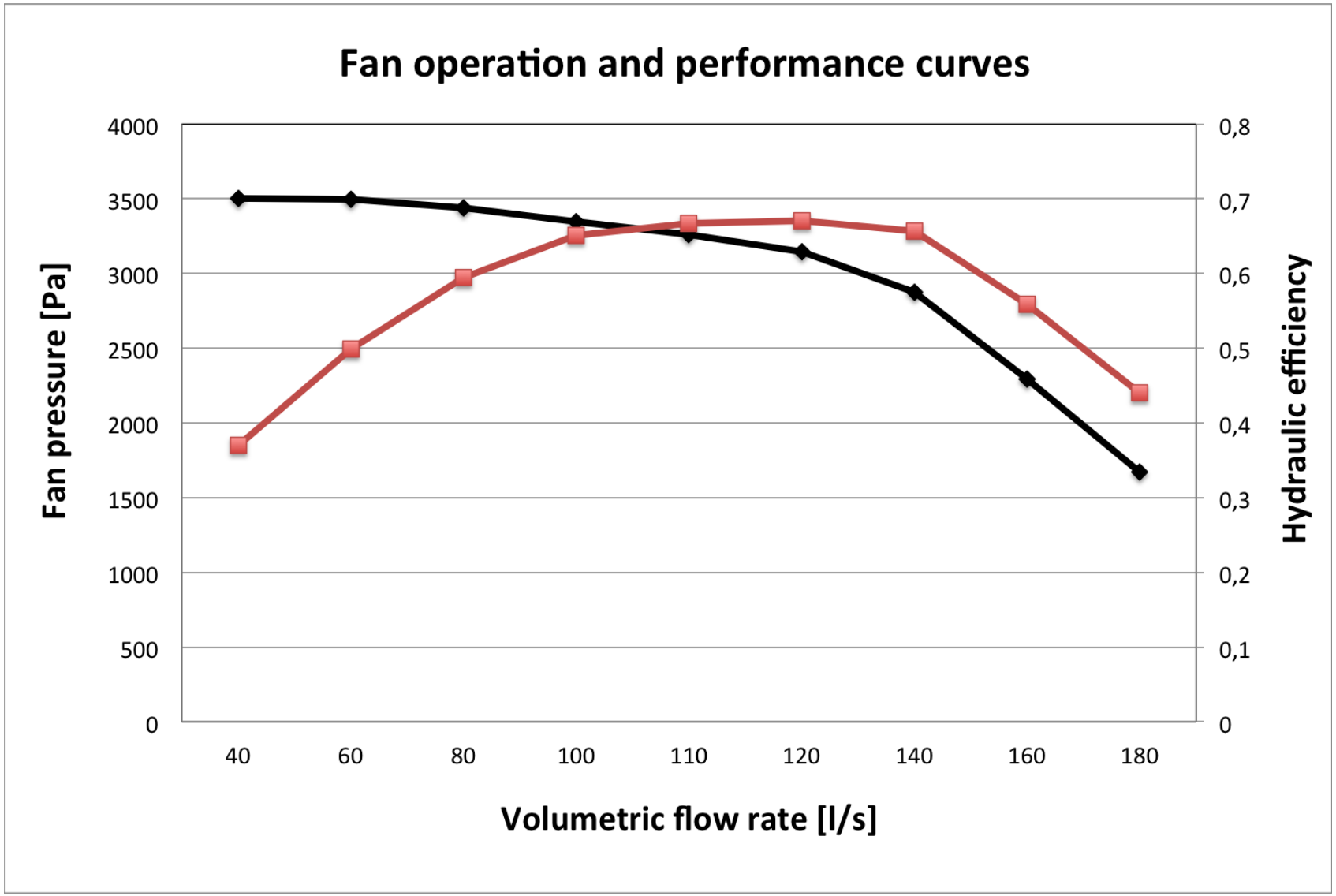
```
averageVelocity
{
    type            faceSource;
    functionObjectLibs
("libfieldFunctionObjects.so");
    log             yes;
    valueOutput     true;
    outputControl   timeStep;
    outputInterval  1;
    writeInterval   1000;
    surfaceFormat   null;
    source          patch;
    sourceName      "volute_outlet";
    operation       areaNormalAverage;
    fields
    (
        U
    );
}
```

Calculation of variables - efficiency

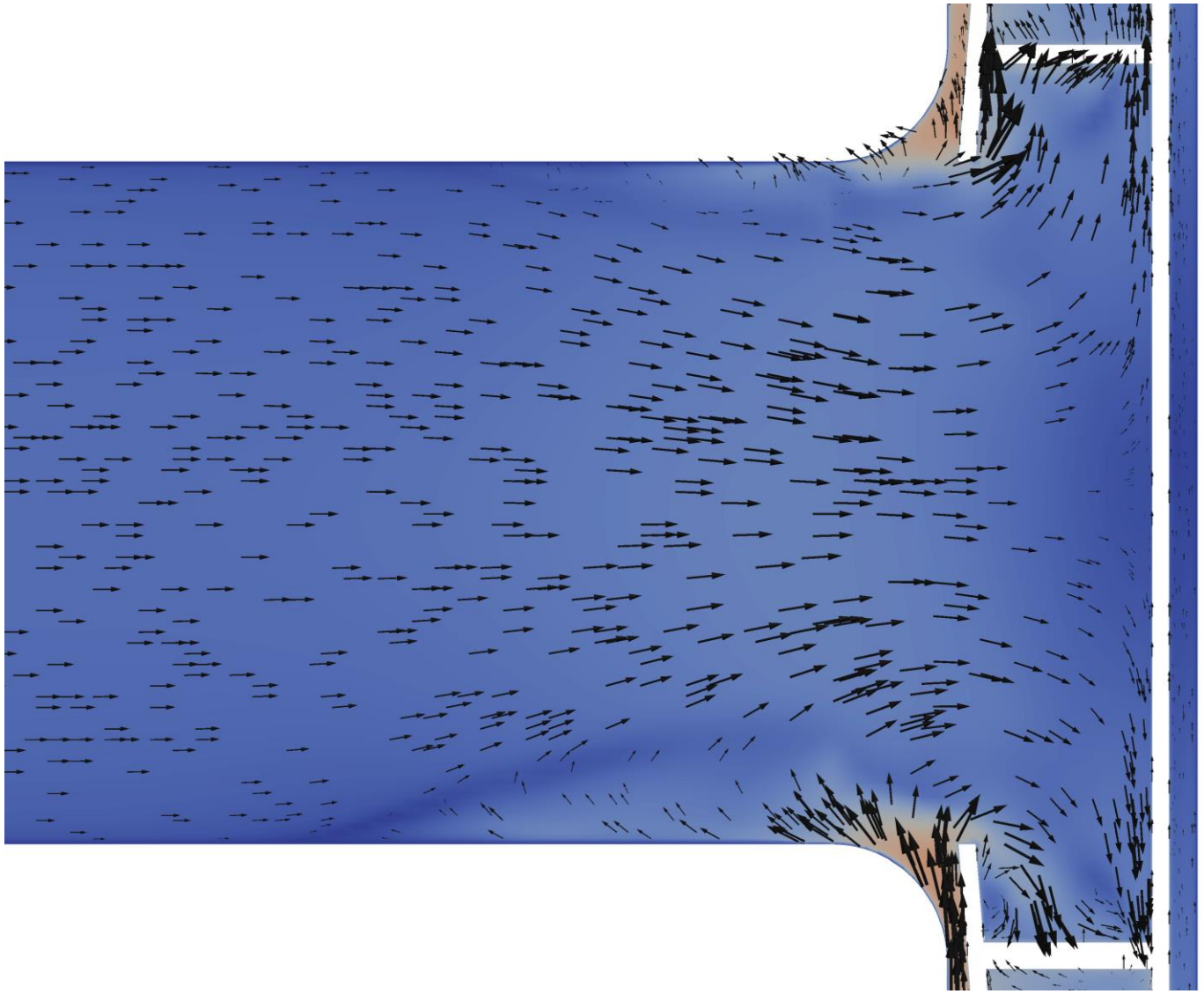
- Function objects: rotorForces (M_p, M_v)
- $\eta_h = \frac{P_{air}}{P_{shaft}}$
- $P_{air} = p_f Q,$
- $P_{shaft} = \omega(M_p + M_v)$

Calculation of variables – efficiency

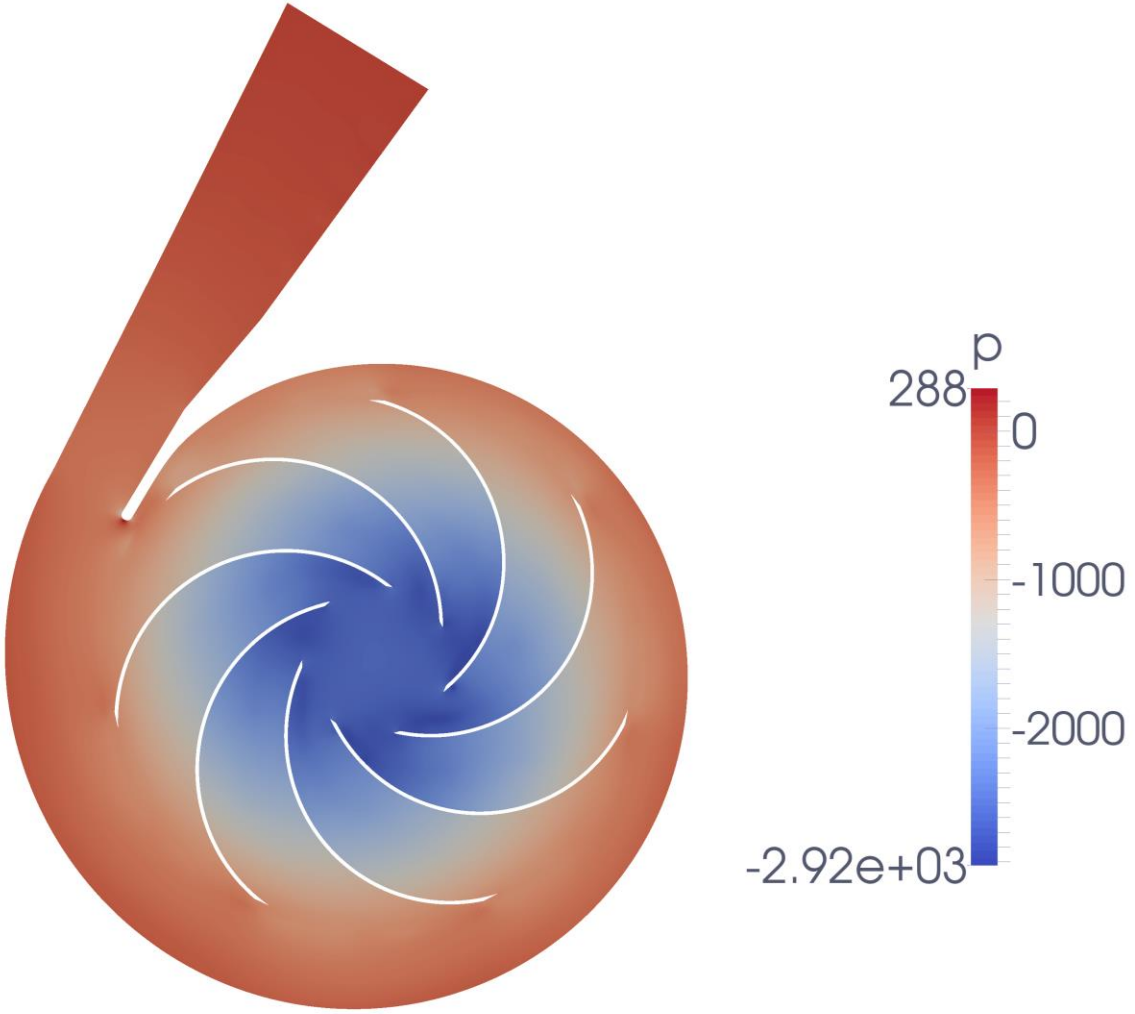
```
rotorForces
{
    type                forces;
    functionObjectLibs  ("libforces.so");
    outputControl       timeStep;
    outputInterval      1;
    writeInterval       1000;
    log                 yes;
    patches             (rotor shaft);
    CofR                (0 -0.008 0);
    rhoName             rhoInf;
    rhoInf              1.205;
}
```



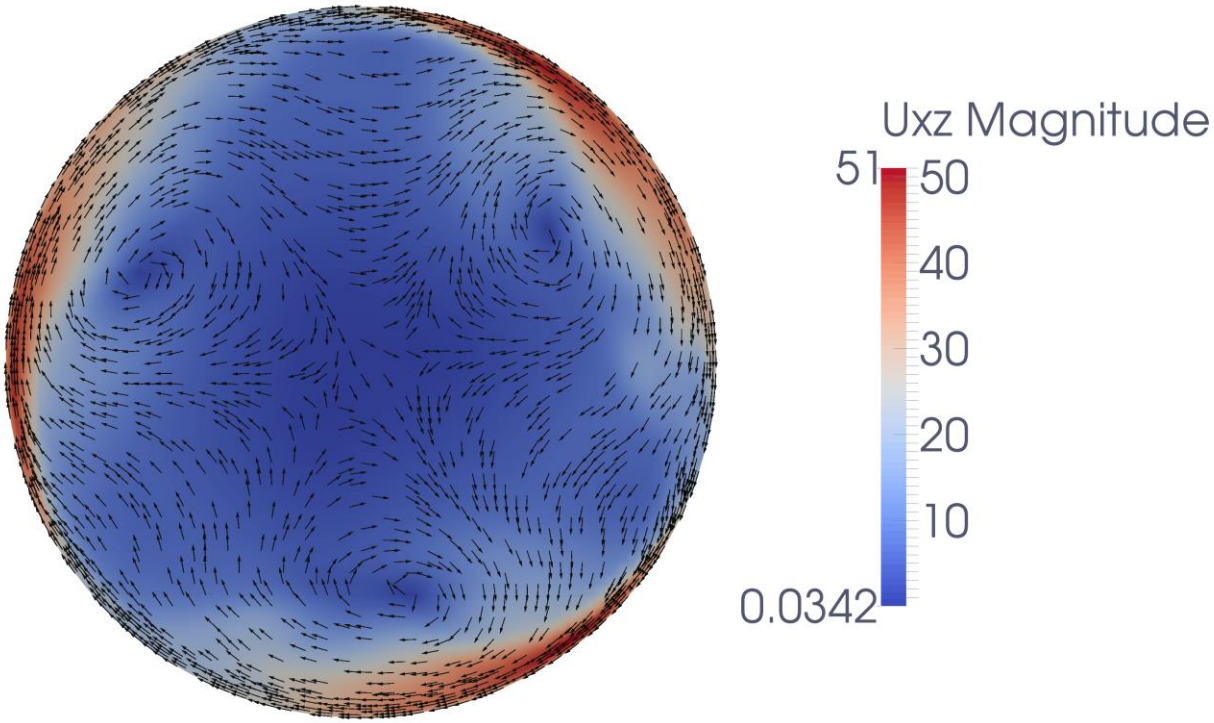
Results - Fan curves



Results – Leakage flow

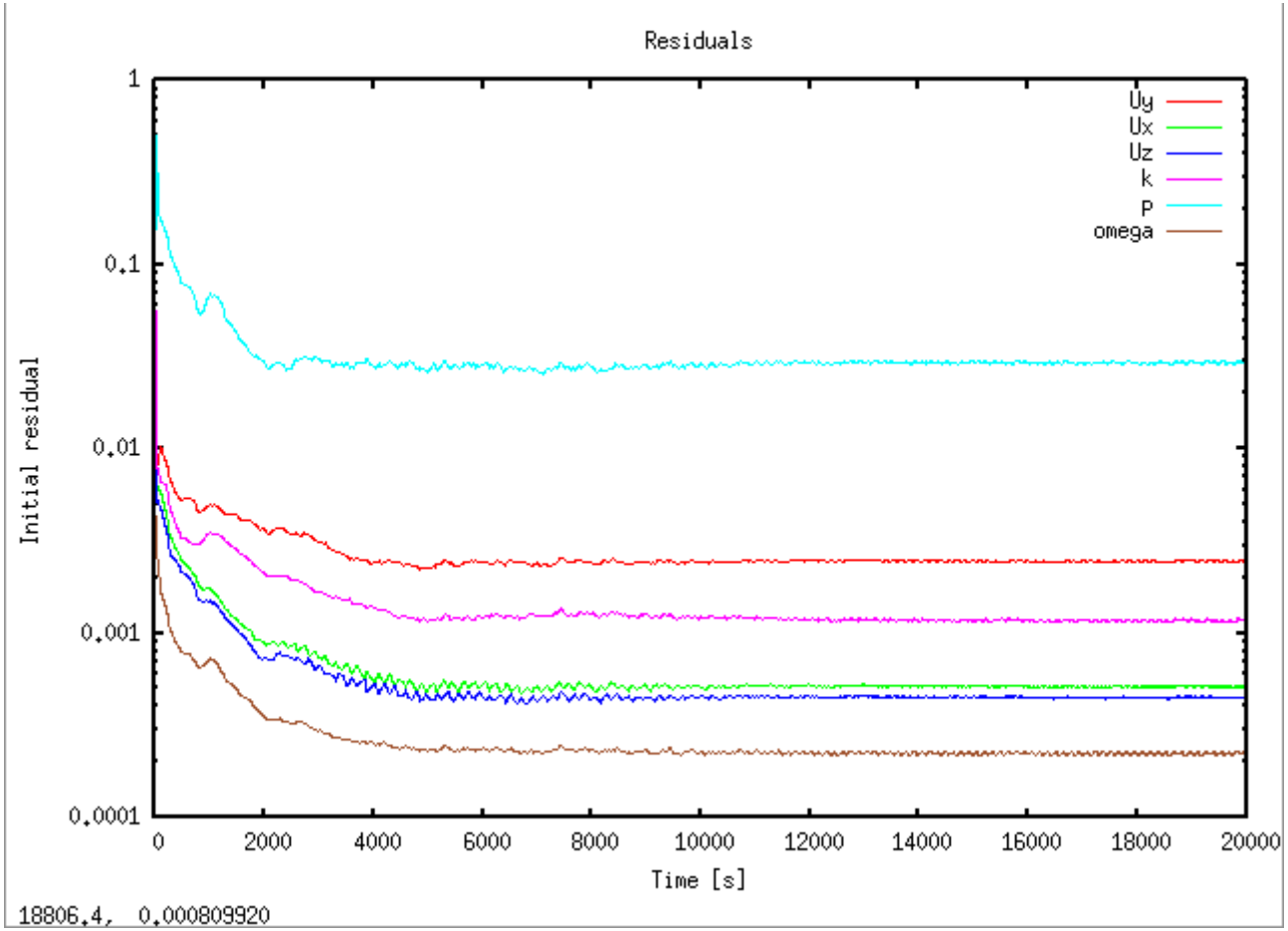


Results – Pressure field

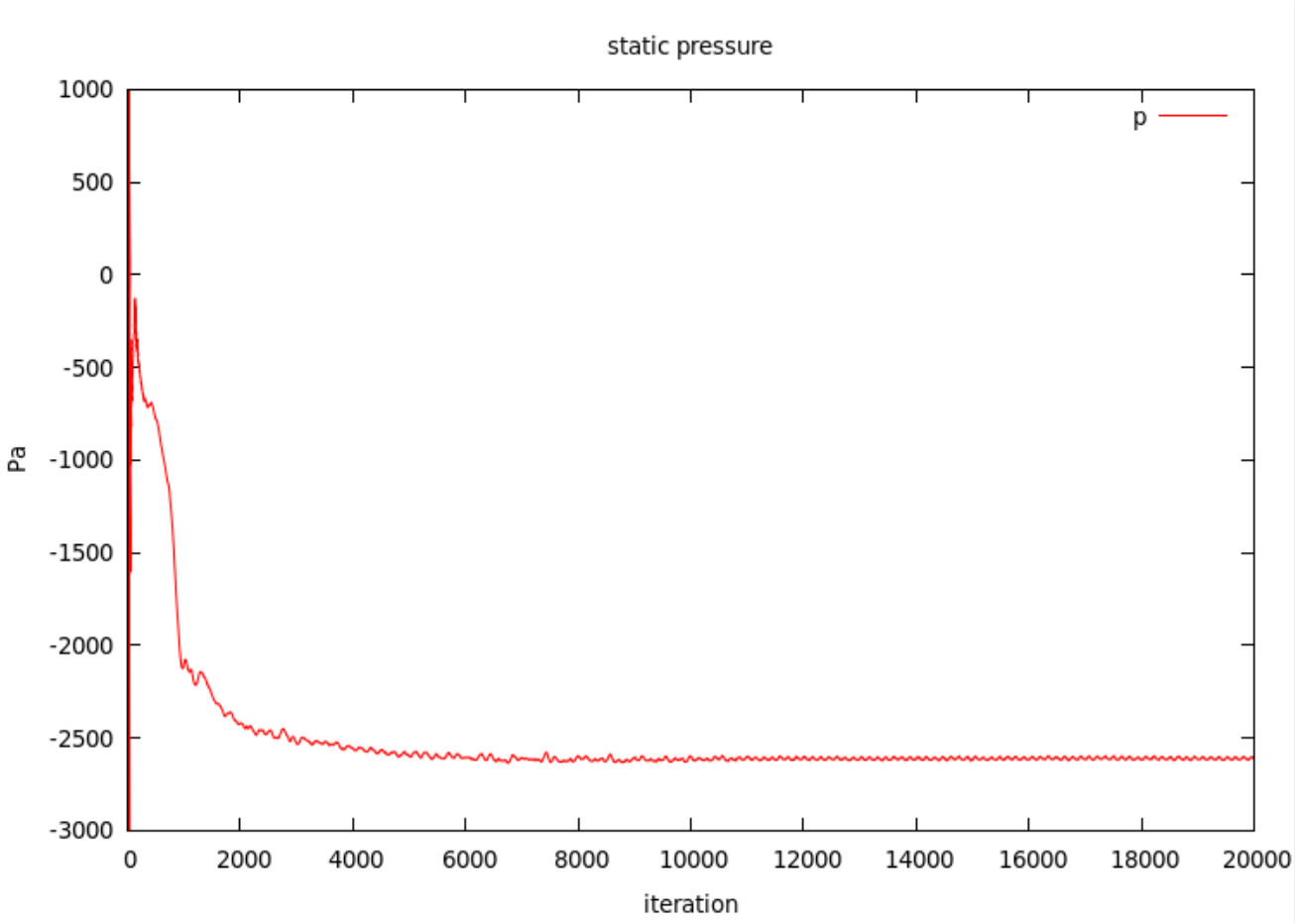


Results – Rotor inlet velocity field

Results - convergence



Results - convergence



Conclusions

- OpenFOAM is well suited for turbomachinery simulations (AMI, MRF, DyM-solvers)
- MRF-simulations can only reach pseudo-convergence with full flow domain
- Simulation of full turbomachine geometry and flow domain is possible – small gaps are problematic during meshing

What's next

- Validation! (measurements in the laboratory)
- Unsteady simulations
- More turbulence models (v2f, kOmegaSST-SAS)
- Improve mesh quality
- (I)LES?