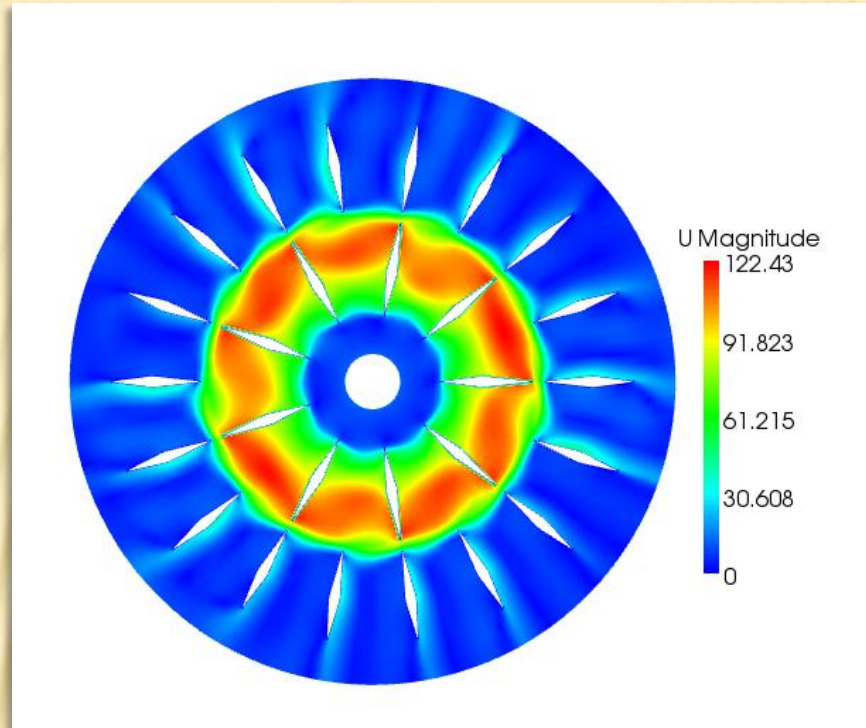


Setting up a case for turbomachinery problems



Outline

- ✗ **Pre-processing utilities:** import a mesh, mergeMesh, stitchMesh, transformPoints, creation of zones and sets.
- ✗ **MRFSimpleFoam solver**, implementation and set-up.
- ✗ **The General Grid Interface (GGI)**, cyclicGgi, overlapGgi, implementation and set-up.
- ✗ **The unsteady solver, turbDyMFoam**, implementation and set-up.
- ✗ **Tools and functionObjects.**

Building a case from a Gambit mesh.

- ✗ Open the case Icem/of5-dev/pump_2D_stitch.
- ✗ Case built from 2 meshes created in Gambit format, in IcemHEXA, rotor2D.msh, stator2D.msh.
- ✗ **This tutorial is using 1.5-dev.**
- ✗ *First step:* convert the fluent mesh into foam format:
fluentMeshToFoam - case rotor_2D meshes/rotor2D.msh
fluentMeshToFoam - case stator_2D meshes/stator2D.msh
- ✗ *Second step:* **merge** the two meshes together.
- ✗ *Third step:* **stitch** the merged meshes together.
- ✗ *Last step:* **scale** the case, if needed.

mergeMeshes.

- ✗ This utility takes the meshes from two different cases and merges them into the master case.
- ✗ The two meshes will keep all their original boundary conditions, so they are not automatically coupled.
- ✗ mergeMeshes reads the system/controlDict of both cases.
- ✗ Usage: **mergeMeshes <master root> <master case> <root to add> <case to add>**
- ✗ The result of the mergeMesh is saved into the first time step folder according to system/controlDict (in this tutorial, 1/).

stitchMesh.

- ✗ stitchMesh couples two uncoupled parts of the mesh that belong to the same case.
- ✗ You should have a patch in one part of the mesh (masterPatch) that fits with a corresponding patch on the other part of the mesh (slavePatch).
- ✗ MasterPatch and slavePatch are important, as the face and cell numbers will be renamed after the master patch.
- ✗ Usage: **stitchMesh <masterPatch> <slavePatch>**
- ✗ Remember to delete the empty patches in constant/polyMesh/boundary, or the simulation will not start.

transformPoints.

- ✗ **Usage:** **transformPoints [-translate “(vector)”] [-rotate “(vector vector)”] [-scale “(vector)”]**
- ✗ Useful to rotate, scale, translate a mesh.
- ✗ The transformPoints utility overwrite the mesh in constant/polyMesh, no folder is created.

MRFSimpleFoam: implementation.

- ✗ MRFSimpleFoam means Multiple Reference Frame simpleFoam.
- ✗ Steady-state solver, for incompressible, turbulent flow, using the SIMPLE solver.
- ✗ When a frame is rotating, the flux equation is solved using an extra term, the Coriolis term:

Frame	Convected velocity	Steady incompressible Navier-Stokes equations
Inertial	Absolute velocity	$\begin{cases} \nabla \cdot (\vec{u}_I \otimes \vec{u}_I) = -\nabla(p/\rho) + \nu \nabla \cdot \nabla(\vec{u}_I) \\ \nabla \cdot \vec{u}_I = 0 \end{cases}$
Rotating	Relative velocity	$\begin{cases} \nabla \cdot (\vec{u}_R \otimes \vec{u}_R) + 2\vec{\Omega} \times \vec{u}_R + \vec{\Omega} \times \vec{\Omega} \times \vec{r} = -\nabla(p/\rho) + \nu \nabla \cdot \nabla(\vec{u}_R) \\ \nabla \cdot \vec{u}_R = 0 \end{cases}$
Rotating	Absolute velocity	$\begin{cases} \nabla \cdot (\vec{u}_R \otimes \vec{u}_I) + \vec{\Omega} \times \vec{u}_I = -\nabla(p/\rho) + \nu \nabla \cdot \nabla(\vec{u}_I) \\ \nabla \cdot \vec{u}_I = 0 \end{cases}$

MRFSimpleFoam: compilation.

- ✗ Can be found in
`$FOAM_TUT/MRFSimpleFoam/MRFSimpleFoam` for **1.5-dev** or
`$FOAM_TUT/incompressible/MRFSimpleFoam/MRFSimpleFoam` for **1.6.x**.
- ✗ Copy it into `$WM_PROJECT_USER_DIR`, and do `wmake`.
- ✗ The executable is `MRFSimpleFoam`.

MRFSimpleFoam: set-up.

- ✗ The set-up changes between 1.5-dev and 1.6.x, but the steps are similar:
 - + Define a region where the Coriolis force will be added, when calculating the flux.
 - + Define the rotating parameters. Those are defined in constant/MRFZones.

Creation of the rotating region.

- ✗ Use of sets and zones.
 - + Creation of a set of cells that define the rotating region, using the many operations available in topoSetSource.
 - + Creation of a set of faces from the previous set to get the Coriolis term for those faces for the flux equation. **ONLY IN 1.5-dev**
 - + Convert the set of cells (and faces for of-1.5-dev) into cellZones and faceZones using setsToZones.

Creation of the rotating region.

- ✗ The created zones can be checked in paraFoam, to be sure that the rotating region was chosen properly.
- ✗ The exact steps to perform for of-1.5-dev can be found among other tutorials in
http://openfoamwiki.net/index.php/Sig_Turbomachinery/_ERCOFTAC_centrifugal_pump_with_a_vaned_diffuser.

Definition of the rotating parameters.

- ✗ Defined in constant/MRFZones.
- ✗ Differ from of-1.5-dev to of-1.6.x.
- ✗ OBS! The rotating velocity is in **rad/s**, not in rpm.

Boundary conditions for pump_2D_stitch.

- ✗ the boundary condition at the inlet is called **profile1DfixedValue**.
- ✗ It is user developed and can be found at http://openfoamwiki.net/index.php/Sig_Turbomachinery_Library_OpenFoamTurbo
- ✗ The boundary condition **profile1DfixedValue** implements a cylindrical boundary condition field defined by a 1D fixed value profile (radial or vertical) for a typical RANS k-epsilon computation.
- ✗ Limitations: The rotation axis is forced to the Z axis.

Boundary conditions for pump_2D_stitch

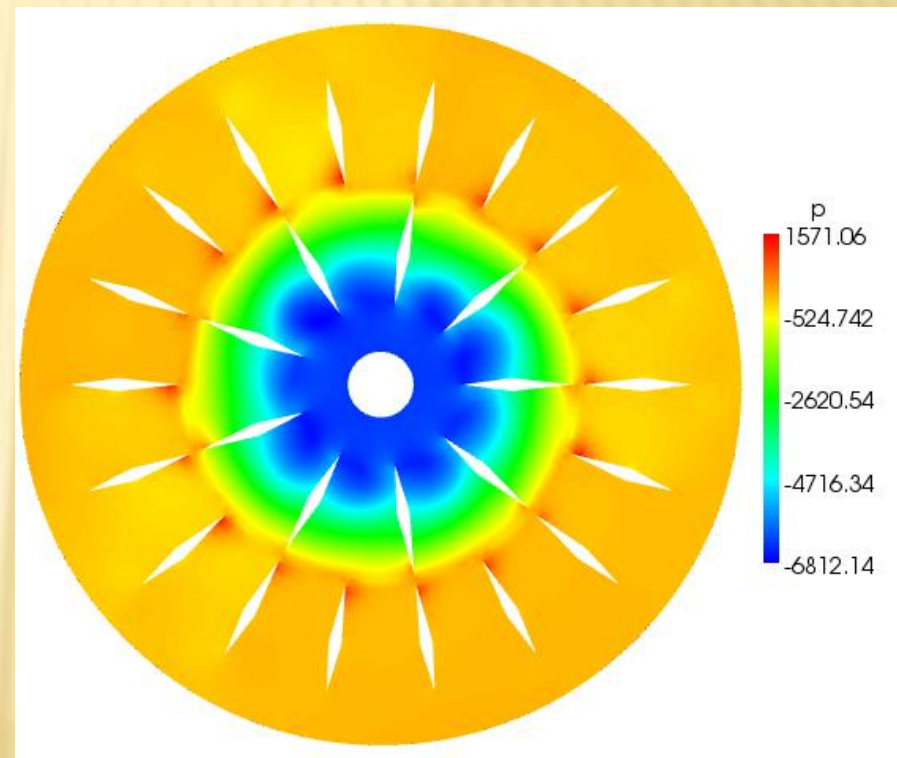
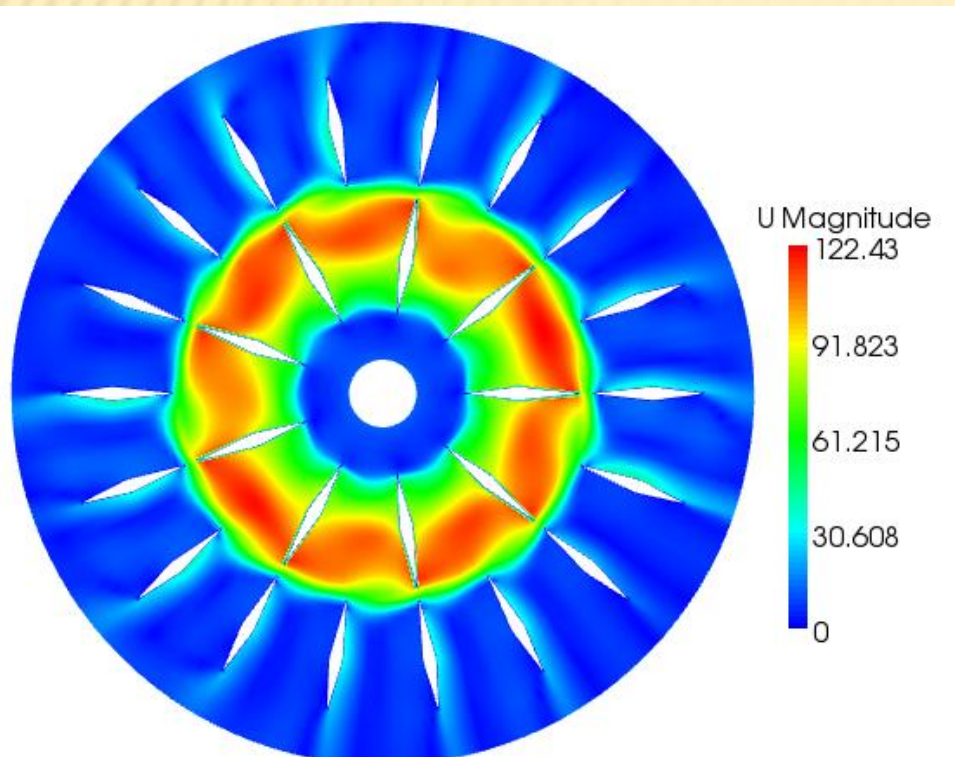
- ✗ Usage:

```
type            profile1DfixedValue;  
fileName        "rotor2d_abs.csv";  
fileFormat      "turboCSV";  
interpolateCoord "Z";  
fieldName       "Velocity";
```
- ✗ Need a file called in this case rotor2d_abs.csv in constant. In this file the header should be as mentionned on the wiki.
- ✗ Need to link the library **libOpenFoamTurbo.so** at the end of system/controlDict.

Writting a script to automatically run a case.

- ✗ It is possible to create an executable that will do all operations automatically.
- ✗ Here, it is called Allrun.
- ✗ To use it, the command is `./Allrun`.
- ✗ When a simulation is finished, and one wants to start over, `./Allclean` removes everything that wasn't there at start.

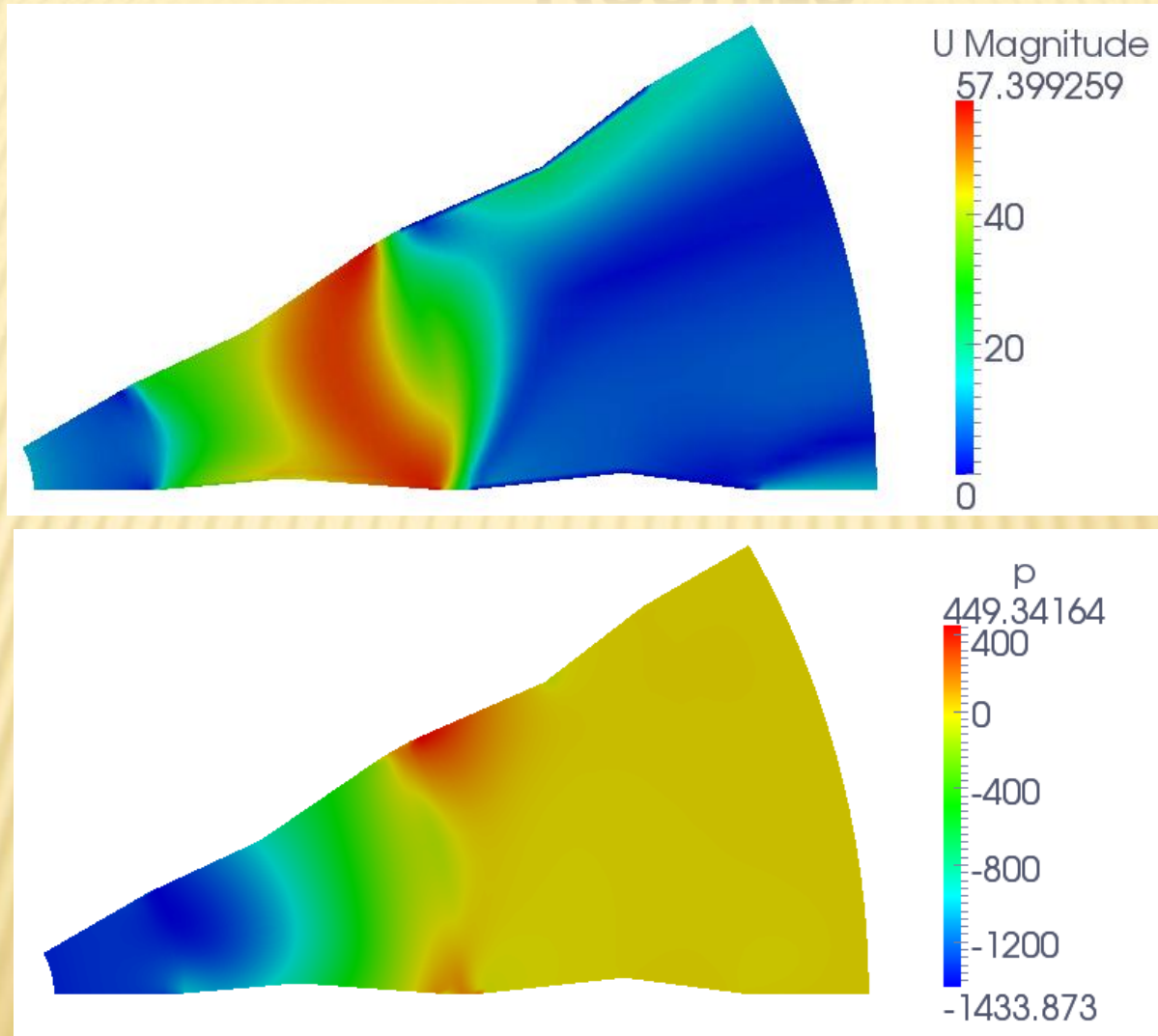
Results



MRFSimpleFoam in of-1.6.x

- ✗ Source OpenFOAM-1.6.x.
- ✗ The tutorial is located in m4_blockMesh/of6x.
- ✗ To create the cellZone, a manipulation called **cylinderToCell** is used. Select all the cell inside the described cylinder.
- ✗ The boundary condition used for the velocity is called **surfaceNormalFixedValue**, create a uniform radial velocity.
- ✗ This tutorial introduces the patch **cyclic**. It allows to take into account rotation periodicity.
- ✗ Limitation: cyclic needs to be 1 to 1 cell periodic, so that the mesh can not change from one side of the patch to an other.

Results



The GGI implementation

- ✗ Coupling interface used to join multiple non-conformal regions where the patches nodes on each side of the interface **do not match**.
- ✗ Non-conformal meshes can be designed separately, and joined together using one of many GGI alternatives:
 - ✗ Ggi CyclicGgi OverlapGgi mixingPlane (in progress, not implemented yet)
- ✗ Weight factors is used to know how much information should be transferred from one side of the ggi to its neighbour cells on the other side of the ggi.
- ✗ The GGI is user developed and is a part of **of-XX-dev ONLY**. It is not available in the OpenCFD versions.

Tutorial using the GGI interface: pump_2D_ggi

- ✗ Source of-1.5-dev
- ✗ The set up of the case is the same as pump_2D_stitch:
 - ✗ Convert the two meshes into OpenFOAM.
 - ✗ Merge the two meshes
 - ✗ Scale the final mesh
- ✗ The stitch operation is replaced by the GGI interface.

GGI interface, basic setup

constant/polyMesh/boundary

```
GGI_INT
{
    type            ggi;
    nFaces          707;
    startFace       374119;
    shadowPatch     GGI_EXT;
    bridgeOverlap   false;
    zone            GGI_INT_ZONE;
}
```

```
GGI_EXT
{
    type            ggi;
    nFaces          756;
    startFace       374826;
    shadowPatch     GGI_INT;
    bridgeOverlap   false;
    zone            GGI_EXT_ZONE;
}
```

0/[U p k epsilon] boundaryField

```
GGI_INT
{
    type ggi;
}
GGI_EXT
{
    type ggi;
}
```

➤ Additional step for serial/parallel computing:

```
setBatch file: faceSet GGI_INT_ZONE new patchToFace GGI_INT
                faceSet GGI_EXT_ZONE new patchToFace GGI_EXT
                quit
setSet -batch setBatch
setsToZones -noFlipMap
```

Tutorial using the cyclicGgi:

m4_blockMesh/of5-dev/ mixer_2D_MRF_m4

- ✗ Source of-1.5-dev
- ✗ Geometry created via m4 and blockMesh.
- ✗ Instead of the cyclic patch, cycliGgi is used to allow a non matching mesh between the two periodic patches.
- ✗ The normal Ggi is used as well between the rotor and the stator parts.

CyclicGgi interface, basic setup

constant/polyMesh/boundary

ROTOR_CYCLIC_LEFT

```
{
    type            cyclicGgi;
    nFaces          13;
    startFace       3514;
    shadowPatch     ROTOR_CYCLIC_RIGHT;
    zone            ROTOR_CYCLIC_LEFT_ZONE;
    bridgeOverlap   off;
    rotationAxis    (0 0 1);
    rotationAngle   -40;
    separationOffset (0 0 0);
}
```

ROTOR_CYCLIC_RIGHT

```
{
    type            cyclicGgi;
    nFaces          13;
    startFace       3540;
    shadowPatch     ROTOR_CYCLIC_LEFT;
    zone            ROTOR_CYCLIC_RIGHT_ZONE;
    bridgeOverlap   off;
    rotationAxis    (0 0 1);
    rotationAngle   40;
    separationOffset (0 0 0);
}
```

0/[U p k epsilon]
boundaryField

```
ROTOR_CYCLIC_LEFT
{
    type cyclicGgi;
}
ROTOR_CYCLIC_RIGHT
{
    type cyclicGgi;
}
```

```
setBatch file: faceSet ROTOR_CYCLIC_LEFT_ZONE new patchToFace ROTOR_CYCLIC_LEFT
               faceSet ROTOR_CYCLIC_RIGHT_ZONE new patchToFace ROTOR_CYCLIC_RIGHT
               quit
setSet -batch setBatch
setsToZones -noFlipMap
```

OBS: The rotation angle in the definition of the cyclicGgi patches is very important. If you do not get it right, an error message of type flotation point error will occur.

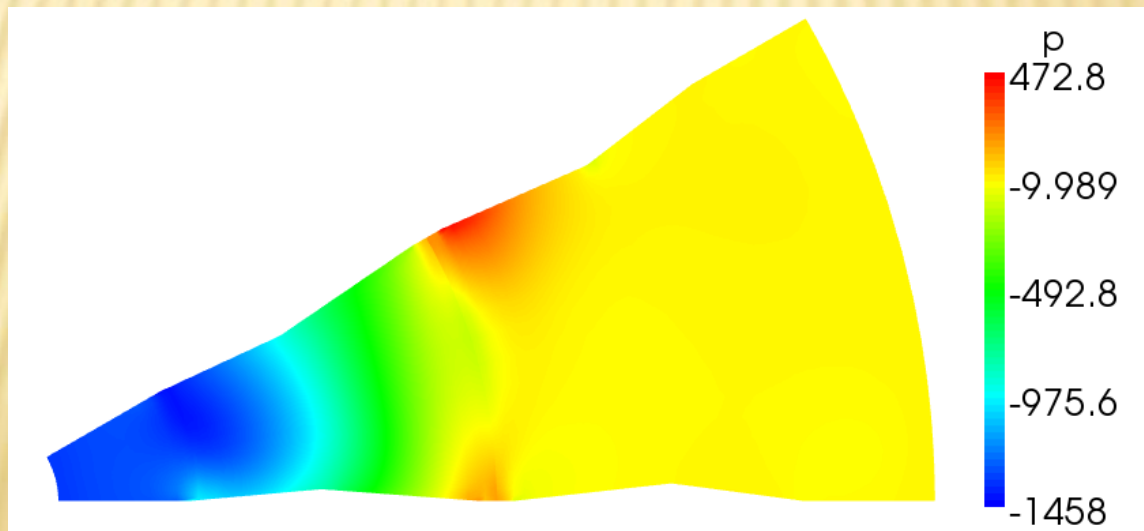
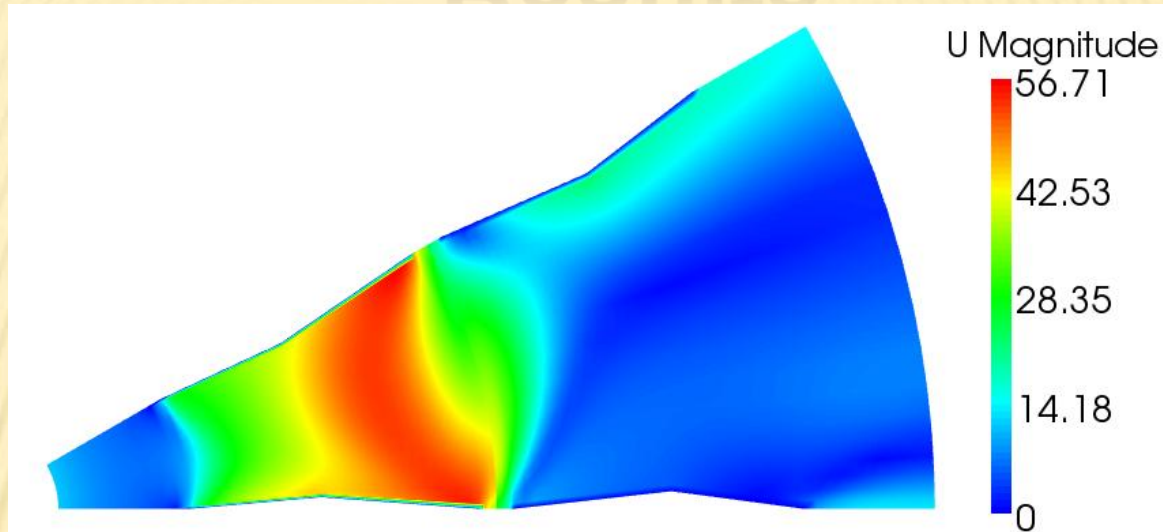
MRFSimpleFoam with cyclic, cyclicGgi in of5-dev.

- ✗ The faces that belong to the patches cyclic, cyclicGgi needs to be calculated with the Coriolis term. If the case is set as usual, those patches are considered as walls.
- ✗ Need to create a new faceSetDict, as showed below:

```
action add;  
topoSetSources  
(  
    patchToFace  
    {  
        name ROTOR_CYCLIC_LEFT;  
    }  
);
```

- ✗ To make the tutorial work properly, uncomment the two commented lines in the Allrun.

Results



Unsteady simulation: turbDyMFoam

- ✗ Transient solver for incompressible, turbulent flow of Newtonian fluids with **moving mesh**.
- ✗ Uses both PISO and SIMPLE to couple U and p .
- ✗ There are many ways a mesh can move, and some of the basics move are coded as **dynamicFvMesh**:
 - + LinearValveFvMesh
 - + MovingConeTopoFvMesh
 - + MixerFvMesh
 - + MixerGgiFvMesh (**only available in of-1.5-dev**)

TurbDyMFoam set-up

- ✗ The tutorial is located in Icem/ pump_2D_ggi_turbDyMFoam.
- ✗ All the definition of the moving mesh is gathered in a dictionary called dynamicMeshDict, located in constant/.
- ✗ A cellZone called movingCell needs to be generated.
- ✗ The boundary condition of the rotorblades for the velocity is

```
ROTORBLADES
{
    type      movingWallVelocity;
    value     uniform ...;
}
```

Unsteady simulation: hints

- ✗ In unsteady simulation, the Courant number is important and directly linked to the time step.
- ✗ Time step should be small, and a lot of SIMPLE loops should be done inside the PISO loop.
- ✗ This is done by playing with the parameters `nCorrectors`, and `nOuterCorrectors` in `system/fvSolution`.
- ✗ The more PISO and SIMPLE loop, the bigger the time step.
- ✗ Start from a steady simulation (done with `MRFSimpleFoam` for exemple).

Useful tools.

- ✗ **PyFOAM.** Simplifies a lot the creation of a case, allow to follow in real-time the simulation, and reduce the errors when building a case.
- ✗ **trackDictionnary.**
- ✗ **ggiCheck.**

Trackdictionnary

- ✗ trackDictionary functionObject writes the value of all the known simulation switches (DebugSwitches, InfoSwitches, OptimisationSwitches, Tolerances, DimensionedConstants) and named dictionary.
- ✗ It is user developed, and works for of-XXX-dev only.
- ✗ Available at http://openfoam-extend.svn.sourceforge.net/svnroot/openfoam-extend/trunk/Breeder_1.5/libraries/simpleFunctionObjects/
- ✗ Needs some lines to be added at the end of system/controlDict.
- ✗ If specified in system/controlDict but not installed, the simulation will not start.

GgiCheck

- ✗ functionObject pre-installed in OpenFOAM-1.XX-dev.
- ✗ Allow the user to see whether the flux across the GGI interface is balanced or not.
- ✗ At the end of system/controlDict:

```
    ggiCheck
{
    type ggiCheck;
    phi phi;
    functionObjectLibs ("libsampling.so");
}
```

GgiCheck

- ✗ During the computation, the ggiCheck functionObject will show this information at each time step:

Initializing the GGI interpolator between master/shadow patches: GGI_INT/GGI_EXT

Evaluation of GGI weighting factors:

Largest slave weighting factor correction : 0 average: 0

Largest master weighting factor correction: 4.4408921e-16 average: 4.9960036e-17

Conclusions

- ✗ Important to understand the differences between of-xxx and of-xxx-dev.
- ✗ Setting up a case can be easier when using the different tools available.
- ✗ Doxygen, forum, wiki are treasure chests if you know what you are looking for.
- ✗ [http://openfoamwiki.net/index.php/Sig_Turbomachinery / Validation test cases](http://openfoamwiki.net/index.php/Sig_Turbomachinery_Validation_test_cases) is a webpage with a lot of informations to simulate a case the best way possible.