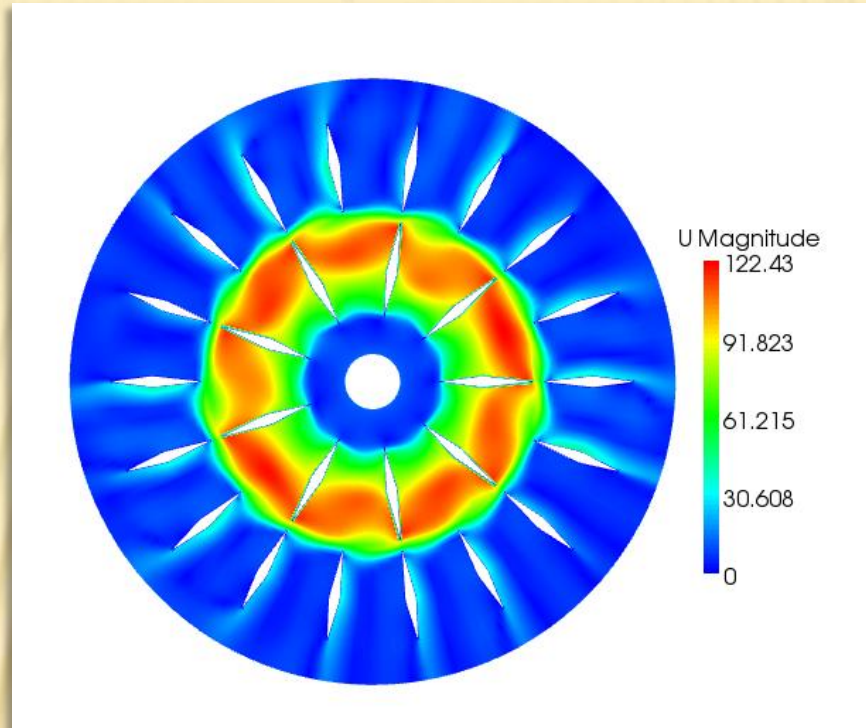


# Setting up a case for turbomachinery problems



# 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, pimpleDyMFoam**, implementation and set-up.
- ✗ **Tools and functionObjects.**

# TUTORIALS AVAILABLE

- ✗ ICEM/of6ext: tutorials for OpenFOAM.1.6-ext
  - + Pump\_2D\_ggi: MRFSimpleFoam, with GGI
  - + Pump\_2D\_stitch: MRFSimpleFoam solver with stitch utility
  - + Pump\_2D\_pimple: unsteady simulation with GGI
- ✗ M4\_blockMesh: tutorials for MRFSimpleFoam using cyclic boundary condition in OpenFoam.1.5-dev and OpenFoam.1.6.x.

# PROFILE1DFIXEDVALUE BOUNDARY CONDITION

- ✘ [http://openfoamwiki.net/index.php/Sig\\_Turbo\\_machinery\\_Library\\_OpenFoamTurbo](http://openfoamwiki.net/index.php/Sig_Turbo_machinery_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 (eg. `simpleFoam`, `turbFoam`, etc).



# Building the pump\_2d\_ggi case.

- ✗ Open the case Icem/of6ext/pump\_2D\_ggi.
- ✗ Case built from 2 meshes created in Gambit format, in IcemHEXA, rotor2D.msh, stator2D.msh.
- ✗ **This tutorial is using 1.6-ext.**
- ✗ *First step:* convert the fluent mesh into foam format:  
fluentMeshToFoam meshes/rotor2D.msh - case rotor\_2D  
fluentMeshToFoam meshes/stator2D.msh - case stator\_2D
- ✗ *Second step:* **merge** the two meshes together.
- ✗ *Third step:* **scale** the case, if needed.
- ✗ *Last step :* Use the **GGI** to pass the information between the two meshes.

# Importing a Gambit mesh in OpenFOAM.

- ✗ The 2 possible commands to do so:

`fluentMeshToFoam`

`fluent3DMeshToFoam`

- ✗ `fluent3DMeshToFoam <Fluent mesh file> [-case dir]`  
`[-ignoreFaceGroups face group names] [-scale scale factor]`  
`[-ignoreCellGroups cell group names] [-help] [-doc] [-srcDoc]`
- ✗ `fluentMeshToFoam <Fluent mesh file> [-writeSets] [-case dir]`  
`[-writeZones] [-scale scale factor] [-help] [-doc] [-srcDoc]`
- ✗ `fluentMeshToFoam meshes/rotor2D.msh - case rotor_2D`  
`fluentMeshToFoam meshes/stator2D.msh - case stator_2D`

# 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/).

# 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.



# 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 (not yet released)
- ✗ Weight factors is used to know how much information should be transferred from one side of the ggi to its neighbor cells on the other side of the ggi.
- ✗ The GGI is user developed and is a part of **of-XX-dev and of-XX-ext ONLY**. It is not available in the OpenCFD versions.

# 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 (you need a setBatch file):

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

# 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/incompressible/MRFSimpleFoam/MRFSimpleFoam/` for **1.6.ext**  
  
`Cp -r $FOAM_TUT/MRFSimpleFoam/MRFSimpleFoam`  
`$WM_PROJECT_USER_DIR`  
`Cd $WM_PROJECT_USER_DIR/MRFSimpleFoam`  
`wmake.`
- ✗ The executable is MRFSimpleFoam.
- ✗ In the 2.0.x version, it is compiled with the main version.



# MRFSimpleFoam: set-up in OF6-ext.

- + 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`.
  - + Convert the set of cells into `cellZones` and using `setsToZones`.
  - + The created zones can be checked in `paraFoam`, to be sure that the rotating region was chosen properly.

# Definition of the rotating parameters.

- ✗ Defined in constant/MRFZones.
- ✗ Differ from of-1.6-ext to of-2.0.x.
- ✗ OBS! The rotating velocity is in **rad/s**, not in rpm.

# Boundary conditions

- ✗ Usage:

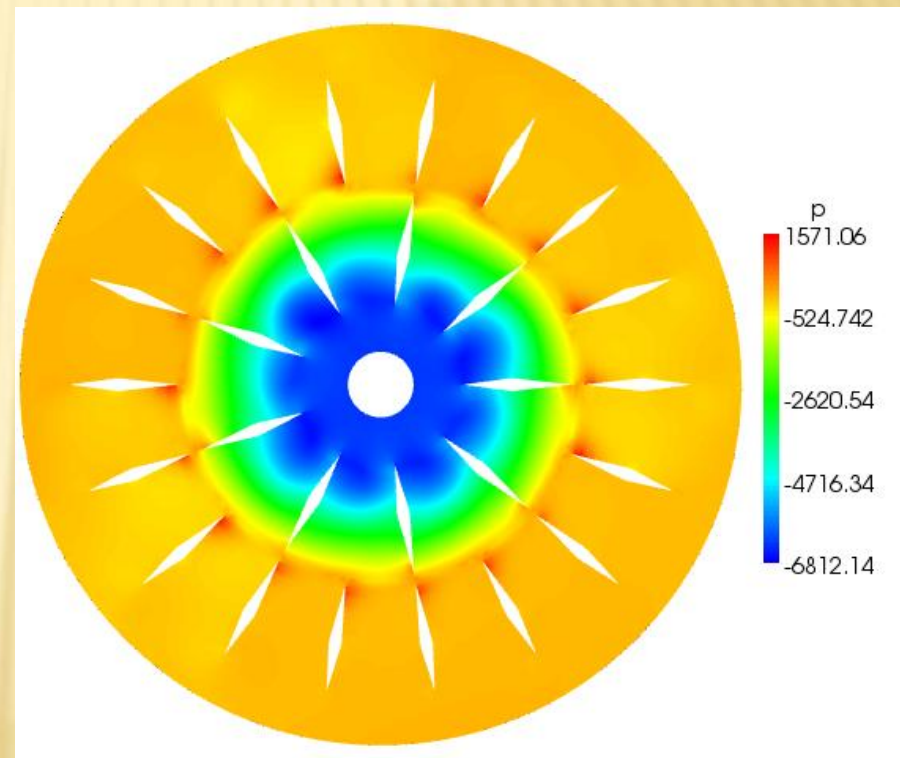
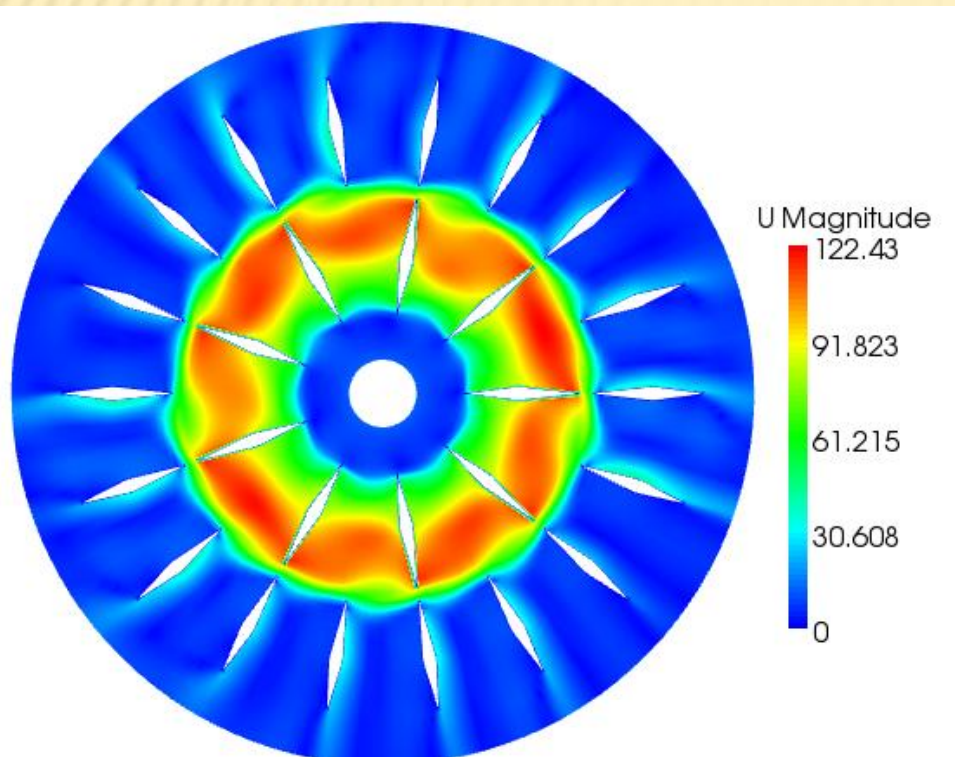
```
type          profile1DfixedValue;  
fileName      "rotor2d_abs.csv";  
fileFormat    "turboCSV";  
interpolateCoord "Z";  
fieldName     "Velocity";
```
- ✗ Need a file called in this case rotor2d\_abs.csv in constant. In this file the header should be as mentionned on the wiki.
- ✗ Need to link the library **libOpenFoamTurbo.so** at the end of system/controlDict.
- ✗ New boundary condition in 1.6-ext:  
**surfaceNormalFixedValue**



# Writing 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



# Unsteady simulation: pimpleDyMFoam

- ✗ Transient solver for incompressible, turbulent flow of Newtonian fluids with **moving mesh**.
- ✗ Uses both PISO and SIMPLE to couple  $U$  and  $p$  and solve the URANS.
- ✗ 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-XX-dev and XX-ext**)

# pimpleDyMFoam set-up

- ✗ The tutorial is located in Icem/ pump\_2D\_ggi\_pimple.
- ✗ 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 ...;
}
```



# pimpleDyMFoam set-up

- ✗ For the case to work, you need to have compiled the following functionObjects:
- ✗ turboPerformance, trackDictionary
- ✗ Can be found in the Breeder 1.6:

svn checkout [https://openfoam-extend.svn.sourceforge.net/svnroot/openfoam-extend/trunk/Breeder\\_1.6/](https://openfoam-extend.svn.sourceforge.net/svnroot/openfoam-extend/trunk/Breeder_1.6/)

# 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 developped, and works for of-XXX-dev and of-xx-ext only.
- ✗ Available at [http://openfoam-extend.svn.sourceforge.net/svnroot/openfoam-extend/trunk/Breeder\\_1.6/libraries/simpleFunctionObjects/](http://openfoam-extend.svn.sourceforge.net/svnroot/openfoam-extend/trunk/Breeder_1.6/libraries/simpleFunctionObjects/)
- ✗ 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.
- ✗ If specified in system/controlDict but not installed, the simulation will not start.

- ✗ At the end of system/controlDict:

```
    ggiCheck
{
    type ggiCheck;
    phi phi;
    functionObjectLibs ("libcheckFunctionObjects.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

# Probes functionObject

- ✖ Allows to probe a point or a surface and write it down for each time step.

# Fluid average function objects

- ✗ Average the chosen fields in time.
- ✗ It is very useful for unsteady simulation, as often we compare against time-averaged measurements
- ✗ It should be activated after the transient phase has gone.



# Turboperformance function object

- ✖ The purpose of this functionObject library is to provide simple functionality that is useful in turbomachinery CFD analysis
- ✖ All the explanation for those functionObjects and how to install them can be found at [http://openfoamwiki.net/index.php/Sig\\_Turbomachinery / Timisoara Swirl Generator](http://openfoamwiki.net/index.php/Sig_Turbomachinery_Timisoara_Swirl_Generator)

# 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. Gather all the previous informations, an more.

# Stitch tutorial: pump\_2D\_stitch.

- ✗ Open the case Icem/of6ext/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 meshes/rotor2D.msh - case rotor\_2D  
fluentMeshToFoam meshes/stator2D.msh - case stator\_2D
- ✗ *Second step:* **merge** the two meshes together.
- ✗ *Third step:* **stitch** the merged meshes together.
- ✗ *Last step:* **scale** the case, if needed.

# 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>.**
- ✗ For pump\_2D\_stitch case:  
    cd pump\_2D\_stitch  
    stitchMesh GGI\_INT GGI\_EXT
- ✗ Remember to delete the empty patches in constant/polyMesh/boundary, or the simulation will not start.



# 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](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.

# MRFSimpleFoam in of-2.0.x

- ✗ Source OpenFOAM-2.0.x.
- ✗ The tutorial is located in m4\_blockMesh/.
- ✗ The cellZone needed to define the rotating area is coded in m4 directly.
- ✗ 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.