CFD with OpenSource software

A course at Chalmers University of Technology
Taught by Håkan Nilsson

Project work:

Evaluate the use of cfMesh for the Francis-99 turbine.

Developed for FOAM-extend-3.2

Author:
Jethro Nagawkar

Peer reviewed by:
Josefine Svenungsson
Håkan Nilsson

Disclaimer: This is a student project work, done as part of a course where OpenFOAM and some other OpenSource software are introduced to the students. Any reader should be aware that it might not be free of errors. Still, it might be useful for someone who would like learn some details similar to the ones presented in the report and in the accompanying files. The material has gone through a review process. The role of the reviewer is to go through the tutorial and make sure that it works, that it is possible to follow, and to some extent correct the writing. The reviewer has no responsibility for the contents.

January 22, 2016

Learning outcomes

The reader will learn the following:

- The different options available for meshing in cfMesh.
- How to install cfMesh.
- How to mesh a geometry using cfMesh.
- Some problems that may occur while meshing and how to resolve them.
- How to merge the meshes of different parts of a geometry.
- How to set up a mixing plane simulation and run it.
- How to implement a new boundary condition.
- How to create a zone for the rotor.

Introduction

The purpose of this project is to evaluate the use of cfMesh for the Francis-99 turbine. This report will discuss the various options available in cfMesh, some problems that occurred during the meshing of the Francis-99 turbine and how to overcome these issues. It will also discuss the setting up of a Mixing Plane Multiple Reference Frame solver and the results obtained from the simulations.
cfMesh is developed by Creative Fields Limited, headed by Dr. Franjo Juretic who is also the managing director, founder and general partner at Creative Fields. The purpose behind it was to develop a meshing tool that is robust, simple to use and also simple to learn and extend. It is different from the traditional unstructured mesh generator as it uses an inside-out method to generate a mesh. The benefits of using this method are as follows, it does not require high-quality input geometry, and is tolerable to small gaps, cracks and protrusions. It can also can be automated, can capture complex geometries and can run on parallel processors.
Figure 1 shows the various parts of a Francis-99 turbine. It consist of 4 parts namely the spiral casing, the distributor, the runner which is the rotating part and finally the draft tube. The distributor has 2 sets of blades, the stay vanes and the guide vanes. These blades are stationary.The purpose of the guide vanes is to guide the water to the runner blades at the correct angle of attack and to regulate the amount of water flowing through the turbine. The stay vanes help to keep the structural integrity of the spiral casing without affecting the flow through the turbine. The water then rotates the runner and exits into the draft tube. To save computational time, in this tutorial we simulate the flow through only one guide vane passage, one runner blade passage and the draft tube.

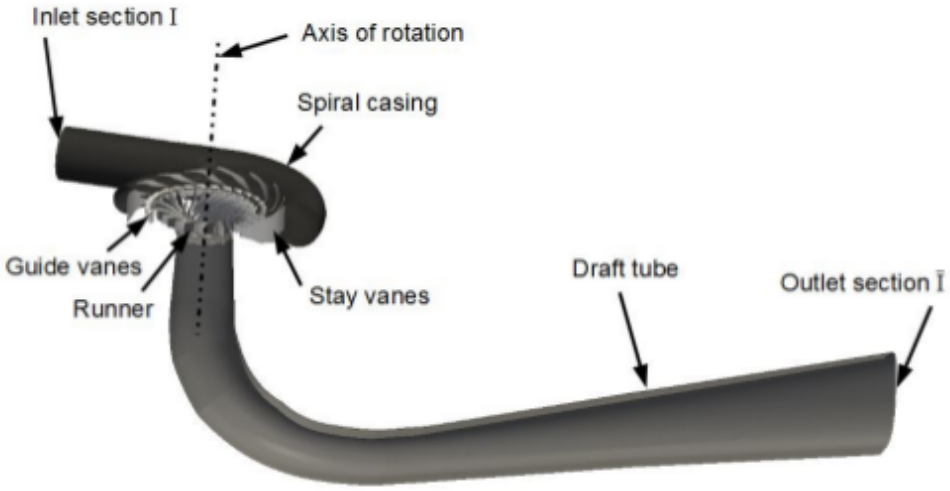

Figure 1: Cut section of the Francis-99 turbine

Options in cfMesh

cfMesh is controlled by a meshDict file that should be located in the system directory of the case to be meshed. This section will discuss some useful options available in the cfMesh meshDict file. Only the first two options are mandatory.

- surfaceFile "filename.stl/filename.fms"
  This is the name of the surface file that you want to use for the meshing along with its path. The suggested formats of the files are .fms, .stl and .ftr. These file formats support the definition of patches, which are transferred to the volume mesh by default.
- maxCellSize
  The global maximum cell size in the domain of the geometry. It is also the default cell size, which is used for the background mesh that is automatically generated by cfMesh
- boundaryCellSize
  The global maximum cell size of the boundary faces. Also the default boundary cell size.
- boundaryCellSizeRefinementThickness
  The distance from the boundary at which the boundary cell size is applied. This is a global setting.
- minCellSize.
  This performs automatic refinement in regions where the cell size is larger than the patch size. It specifies the minimum cell size in the domain. This is a global setting.
- localRefinement
  This is used to specify a local refinement in the domain at the boundaries. It is done by specifying the patch name, its cell size and optionally the distance from the patch for which the specified size should be used. Note that this option is similar to the boundaryCellSize option, except, it is used to specify a value for an individual boundary. boundaryCellSize option specifies the same value for all the boundaries.
- objectRefinements
  It is used to specify refinement zones in the domain. Different types are available namely box, cone, hollow cone, sphere and lines.
- surfaceMeshRefinement
  Uses a surface file to define a refinement zone in the domain. The cell size and refinement zone thickness is used to specify the required sizes.
- edgeMeshRefinement
  An edge mesh file is used to allow local edge mesh refinement zones in a domain.
- keepCellsIntersectingBoundary
  Makes sure that during the meshing process all the cells intersecting the surface template are kept. Useful in capturing small gaps in a geometry. Use 1 for keeping the cells intersecting the boundary and 0 for keeping only the internal cells. By default it is set to 0.
- checkForGluedMesh
  Sometimes activating the above option connects two distinct parts in the geometry. This option ensures that gluing of surfaces do not happen. Use 1 to keep this option active and 0 for inactive. This option can only be used when keepCellsIntersectingBoundary is active. By default it is set to 0.
- keepCellsIntersectingPatches
  This keeps the cells intersecting a particular boundary while meshing. Use 1 for keeping the cells intersecting and 0 for keeping only the internal cells. By default it is set to 0.

- removeCellsIntersectingPatches
  This removes cells intersecting a particular patch when the keepCellsIntersectingBoundary option is activated. Use 0 for removing cells and 1 for keeping them. 1 is the default value.

- boundaryLayers
  Used to create boundary layers on a particular patch. Number of layers, thickness ratio and first boundary layer height of the generated boundary layers should be specified. An allowDiscontinuity option(1 active 0 inactive) may be used to prevent the boundary layers setting on one patch spread to another patch. Note, by default it is set to 0. OptimiseLayer option can be used when you want to vary the boundary layer smoothly. It is followed by a positive integer. This positive integer is the number of optimisation steps. Additional quality control options can be activated in the optimisationParameters dictionary. They are:

- nSmoothNormals: it controls the number of smoothing normal vectors in the boundary layer. Its default value is 5.

- maxNumIterations: It is the number of iterations in the smoothening procedure. The default value is 5.

- featureSizeFactor: It is the ratio between the maximum boundary layer thickness and the patch size. The default value is 0.3

- reCalculateNormals: It calculates the surface normal vectors and aligns boundary layer edges to point in the normal direction. 1 is used to keep it active and 0 for keeping it inactive. By default it is set to 1.

- relThicknessTol: It controls the maximum difference of the layer thickness between the 2 neighboring points in a boundary layer divided by the distance between the points.

- renameBoundary
  Used to rename boundary patches. Useful for setting up the boundary conditions.

Note: the unit for length is always in metres.

Installing cfMesh

FOAM-extend-3.2 includes cfMesh. However, it can be installed separately with other versions as well.
Go to http://sourceforge.net/projects/cfmesh/ and download the cfMesh-v1.1.1.tgz file. Save it anywhere you like. Install by:

```
tar -xvzf cfMesh-v1.1.1.taz
cd cfMesh-v1.1.1/
OF24x
./Allwmake
```

The complied files will end up in the user lib and bin directories.

Meshing procedure

The following section explains the meshing procedure for the Francis-99 turbine. The high load case was chosen for the purpose of the tutorial. A steady-state MRF mixing plane simulation was performed. The flow is assumed to be axi-periodic and the computational domain was thus simplified to contain one guide vane passage, one runner blade passage(Note that 1 runner blade passage contains a splitter blade between the pressure and suction side of 2 different runner blades)

and the draft tube. As mentioned earlier, this reduces computational time considerably. We can therefore refine the mesh further by reducing the cell size. This helps capture the flow much better. Therefore, only one guide vane passage in the distributor and blade passage in the runner were modelled. Setting up and running a full model is not very much different. This tutorial uses a MRF mixing plane solver. The setup is similar to the axialTurbine mixing plane tutorial, which is already present in FOAM-extend-3.2. We will modify this tutorial for our case.
Note:- If you have a .stl file and want a .fms file, use the utility surfaceToFMS <stlfilename.stl>. For this tutorial, all the 3 geometries are convert to .fms format using the previous mentioned utility. The author is unaware of any specific requirements of the .stl geometry required to convert it to .fms format.

Draft Tube

Download the *jethro_openfoam.tar.gz* file and save it in your run directory. Copy the *axialTurbine_mixingPlane* tutorial to your run directory. Make 2 more copies for it. You should have 3 copies for each part of the turbine. The steps are as follows:

```
OFextend32
run
tar -xzf jethro_openfoam.tar.gz

cp -r $FOAM_TUTORIALS/incompressible/MRFSimpleFoam/axialTurbine_mixingPlane DT
cp -r DT distributor
cp -r DT runner
cp jethro_openfoam/periodic/dt/dt.fms DT/
cp jethro_openfoam/periodic/dt/meshDict DT/system/
cd DT
cp -r 0_orig/ 0
cartesianMesh
checkMesh
paraFoam
cd ..
```

The meshDict file contains all the mesh controls. The DT/system/meshDict file for the draft tube looks as follows:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | cfMesh: A library for mesh generation          |
|  \\    /   O peration     |                                                 |
|   \\  /    A nd           | Author: Franjo Juretic                          |
|    \\/     M anipulation  | E-mail: franjo.juretic@c-fields.com             |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version   2.0;
    format    ascii;
    class     dictionary;
    location  "system";
    object    meshDict;
}

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

```
maxCellSize 26;
surfaceFile "dt.fms";
boundaryLayers
{
   // nLayers 3;
   //thicknessRatio 1.2;
    patchBoundaryLayers
    {
        DT_WALL
        {
             nLayers 3;
             thicknessRatio 1.2;
             maxFirstLayerThickness 1;
             allowDiscontinuity 1;
        }
    }
   optimiseLayer 5;
   optimisationParameters
    {
        nSmoothNormals 5;
        relThicknessTol 0.15;
        featureSizeFactor 0.3;
        reCalculateNormals 1;
        maxNumIterations 8;
    }
}
localRefinement
{
    "DT_INLET"
    {
        cellSize 3.5;
        refinementThickness 12;
    }

    "DT_OUTLET"
    {
        cellSize 26;
        refinementThickness 52;
    }
    "DT_WALL"
    {
        cellSize      16;
        refinementThickness 32;
    }
}
renameBoundary
{
    //defaultName fixedWalls;
    //defaultType wall;

    newPatchNames
    {
        "DT_INLET"
        {
```

```
            newName     dtinlet;
            type        patch;
        }

        "DT_OUTLET"
        {
            newName     dtoutlet;
            type        patch;
        }
        "DT_WALL"
        {
            newName     dtwalls;
            type        wall;
        }
    }
}
// ************************************************************************* //
```

In the above listing, the maximum cell size is set to 26m. This sets the required background mesh for the meshing process. At the walls of the draft tube, we need cells in the boundary layer to capture the boundary layer formation. The inlet and outlet are not walls and hence no boundary layer is formed on them. The local refinement option is used to set face sizes on the respective boundaries. The refinementThickness option is used to define the distance from the boundary to which the specified cell size is used. Finally the boundaries are renamed for convenience. The user can change these options. The current selected options generate a course mesh.

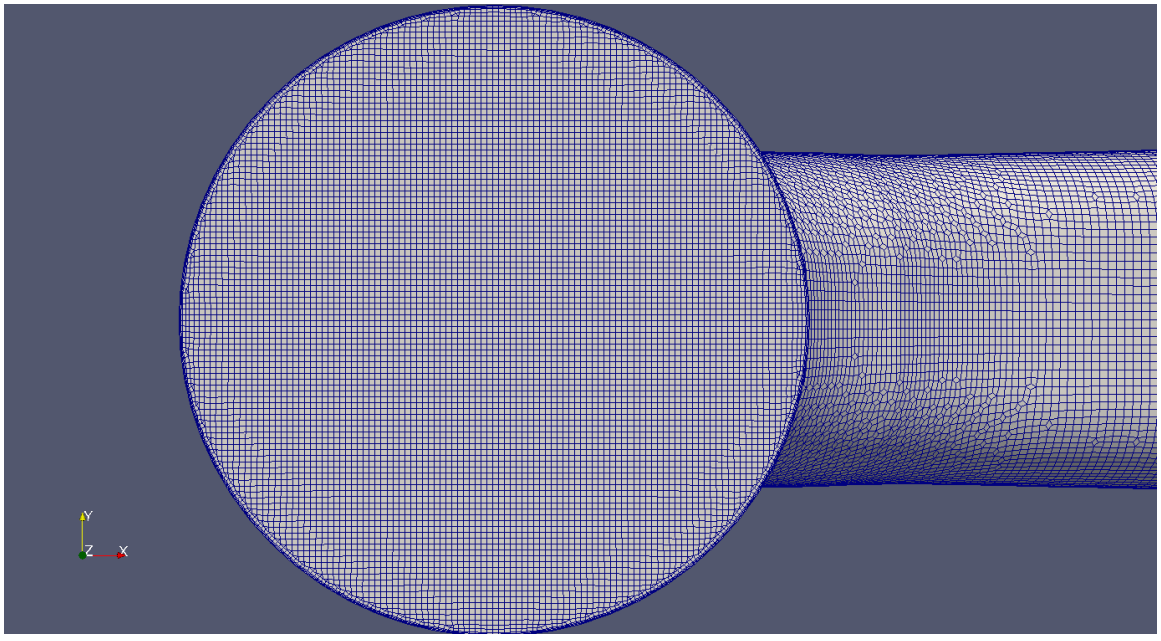

Figure 2: Draft tube inlet

Figures 2, 3, 4 and 5 depict various meshed parts of the draft tube. The maximum skewness obtained was 0.6. The total number of cells generated was 635000 with about 557000 being hexahedral. The remaining cells consisted of a mix of prisms, pyramids, tetrahera and polyhedra. For this tutorial only 3 boundary layers were generated. This can be changed in the meshDict file as per the users requirements. In figure 5, we can see that there is sudden change in mesh size as we move from

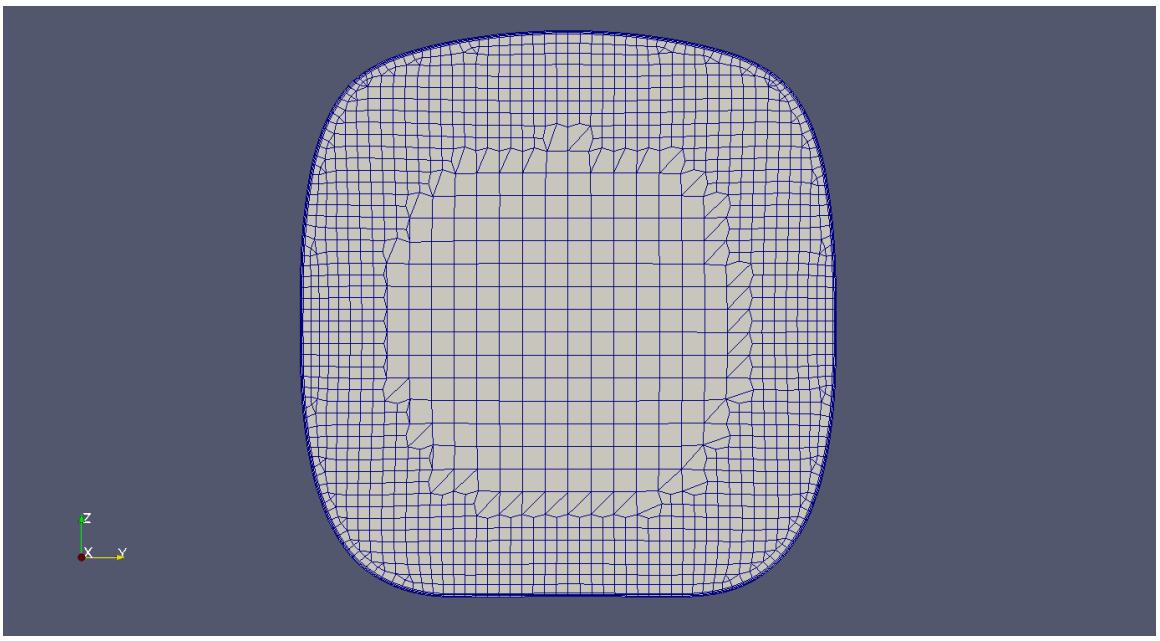

Figure 3: Draft tube outlet

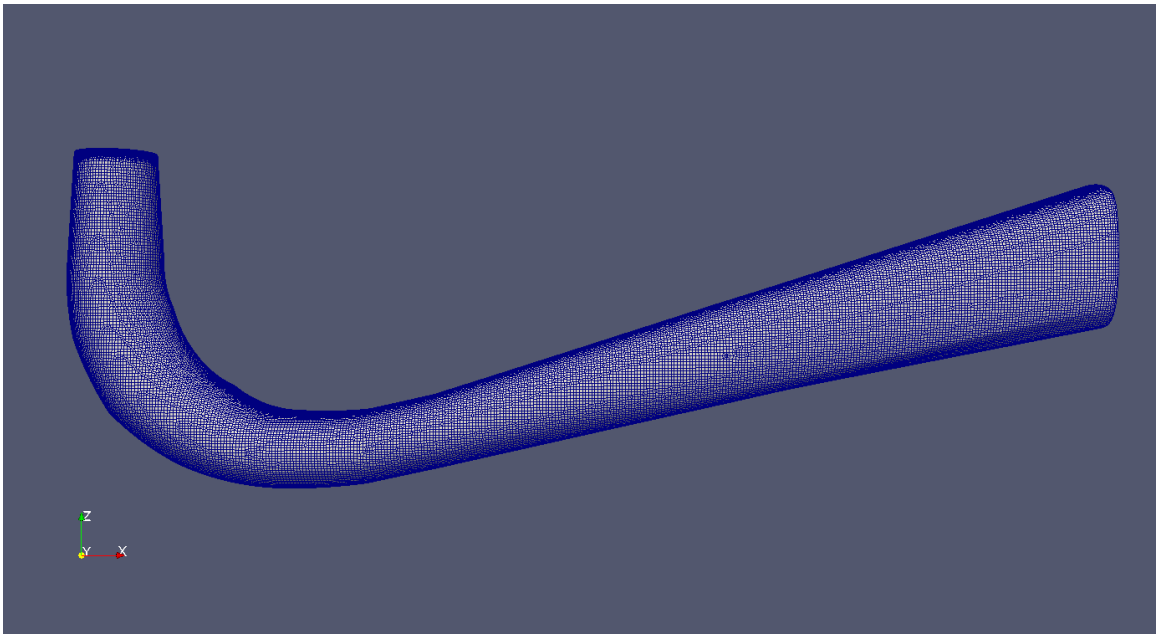

Figure 4: Draft tube

the inlet of the draft tube to its interior. This is not ideal and to improve the mesh, we need to change the refinementThickness setting for DT_INLET under localRefinement, from 12 to higher value say 36. As mentioned before the mesh is course. Refining the mesh will help get a better quality mesh and also a mesh which changes its cell size gradually. The cells at the draft tube inlet are uniform(figure2), whereas the cells at its outlet decrease as we move closer to the walls(figure3). This difference arises in the selection of cell size in the localRefinement option. The inlet cell size is smaller than the wall. Whereas, the outlet cell size is larger than that at the wall.

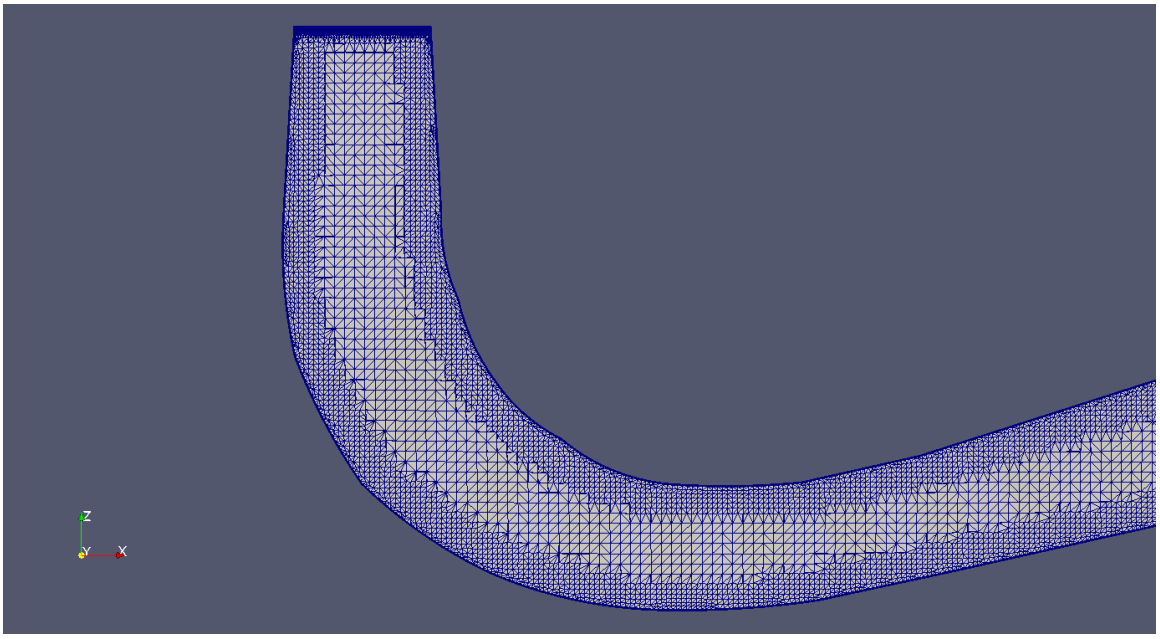

Figure 5: Draft tube cut section

Runner

The procedure for meshing the runner is similar. The steps are as follows:

```
run
cp jethro_openfoam/periodic/runner/runner.fms runner/
cp jethro_openfoam/periodic/runner/meshDict runner/system/
cd runner
cp -r 0_orig/ 0
cartesianMesh
checkMesh
paraFoam
cd ..
```

The setting for the mesh in the runner/system/meshDict file are as follows:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | cfMesh: A library for mesh generation           |
|  \\    /   O peration     |                                                 |
|   \\  /    A nd           | Author: Franjo Juretic                          |
|    \\/     M anipulation  | E-mail: franjo.juretic@c-fields.com             |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version   2.0;
    format    ascii;
    class     dictionary;
    location  "system";
    object    meshDict;
}
```

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

surfaceFile "runner.fms";

maxCellSize 5;

boundaryCellSize 3;

boundaryCellSizeRefinementThickness 6;

localRefinement
{
    "RUNNER_INLET"
    {
        cellSize 2;
        refinementThickness 4;
    }
    "BLA.*"
    {
         cellSize 4;
         refinementThickness 8;
    }
    "SPLITT.*"
    {
         cellSize 4;
         refinementThickness 8;
    }
    "RUNNER_OUTLET"
    {
         cellSize 4;
         refinementThickness 8;
    }
    "SHROUD"
    {
         cellSize 4;
         refinementThickness 8;
    }
    "HUB"
    {
         cellSize 4;
         refinementThickness 8;
    }
}

keepCellsIntersectingBoundary 1;

removeCellsIntersectingPatches
{
   "BLADEPRESSURE_SIDE_01"
   {
        keepCells 1;
   }
   "BLADESUCTION_SIDE_01"
```

```
   {
        keepCells 1;
   }
   "SPLITTER.*"
   {
        keepCells 1;
   }
}

boundaryLayers
{
    patchBoundaryLayers
    {
        "BLADE.*"
        {
             nLayers 3;
             thicknessRatio 1.2;
             maxFirstLayerThickness 1;
        }
        "HUB"
        {
             nLayers 3;
             thicknessRatio 1.2;
             maxFirstLayerThickness 1;
        }
        "SHROUD"
        {
             nLayers 3;
             thicknessRatio 1.2;
             maxFirstLayerThickness 1;
        }
        "SPLITTER.*"
        {
             nLayers 3;

             thicknessRatio 1.2;

             maxFirstLayerThickness 1;
        }
    }
}
renameBoundary
{
    defaultName rnhub;
    defaultType wall;

    newPatchNames
    {
        "SPLITTER.*"
        {
            newName     rnsplitter;
            type        wall;
        }
        "BLADE.*"
```

```
        {
            newName     rnblade;
            type        wall;
        }
        "RUNNER_INLET"
        {
            newName     rninlet;
            type        patch;
        }
        "RUNNER_OUTLET"
        {
            newName     rnoutlet;
            type        patch;
        }
       "SHROUD"
        {
            newName     rnshroud;
            type        wall;
        }
       "CYCLIC_TE_BLADE1"
        {
            newName     rncyclic1;
            type        patch;
        }
       "CYCLIC_LE_BLADE1"
        {
            newName     rncyclic1;
            type        patch;
        }
       "CYCLIC_TE_BLADE2"
        {
            newName     rncyclic2;
            type        patch;
        }
       "CYCLIC_LE_BLADE2"
        {
            newName     rncyclic2;
            type        patch;
        }
    }
}

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

Figures 6,7,8 and 9, show various parts of the runner. An important thing to keep in mind is the interfaces between different parts that are in contact with one another(distributor outlet and runner inlet and runner outlet and draft tube inlet). Having similar cell sizes on the faces that are in contact with each other may be crucial. The total number of cells in this part was 300000 with a maximum skewness of 3.68.

Figure 10 shows an example of collapsed gap between the runner blade and the runner inlet. This is one issue that someone using cfMesh might face. There are two ways to resolve this issue. One is to use a hollowcone object refinement and define an extremely small cell size. The other is to use keepCellsIntersectingPatches for the runner inlet. In the later case you can use a bigger cell size. This option will squeeze cells into the region between the blades and the runner inlet. This results

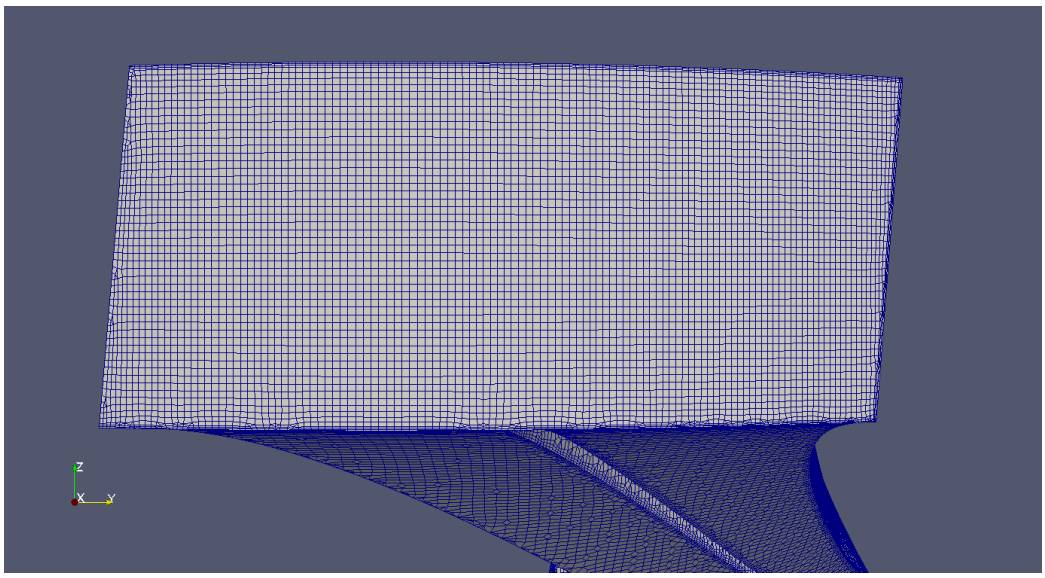

Figure 6: Runner inlet

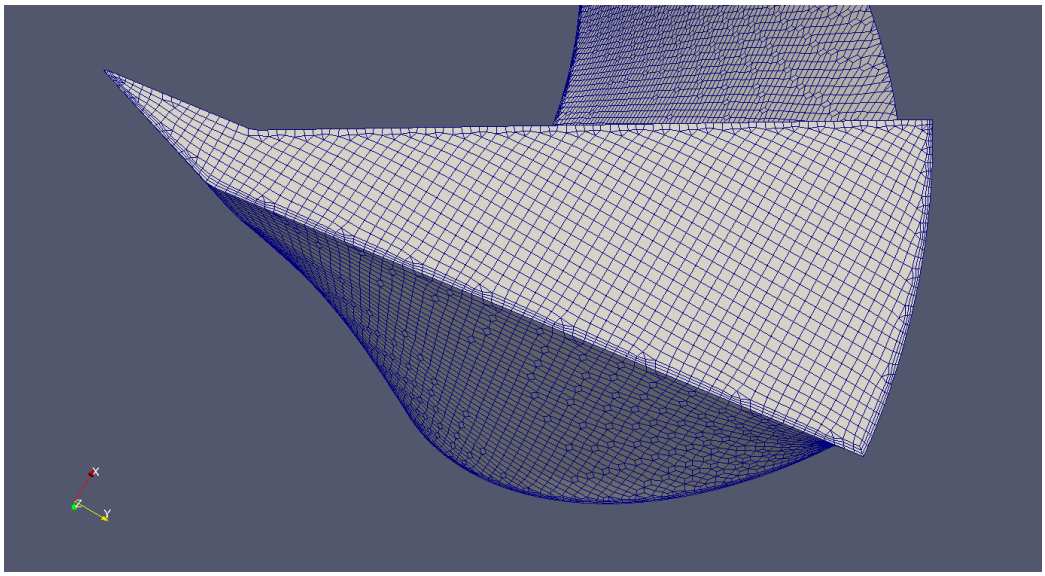

Figure 7: Runner outlet

in reduced number of cells. Figure 11 shows the mesh after the implementation of the later option. For the runner in this tutorial, we have used the keepCellsIntersectingBoundary option. This option is followed by the removeCellsIntersectingPatches option. The idea here is to keep cells only on the blade pressure and suction side and on the splitter blades. The cells on the remaining patches will be removed. This ensures that cells are squeezed in-between small gaps to capture the geometry better.

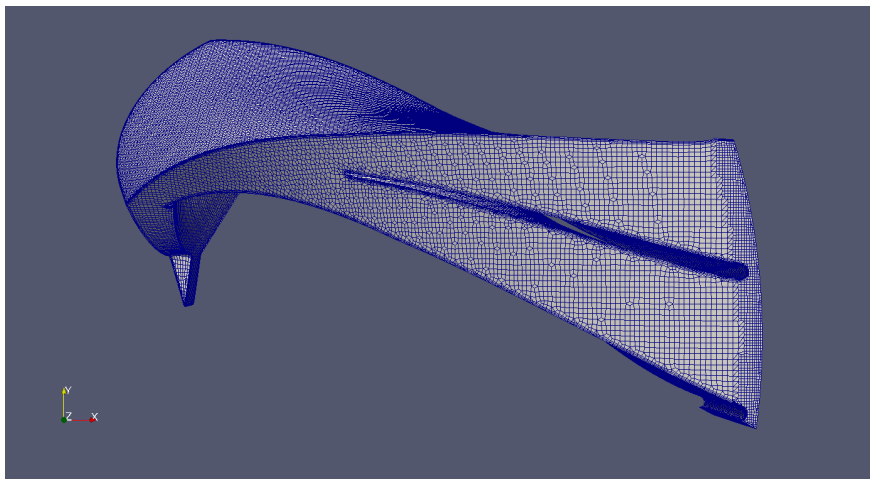

Figure 8: Runner

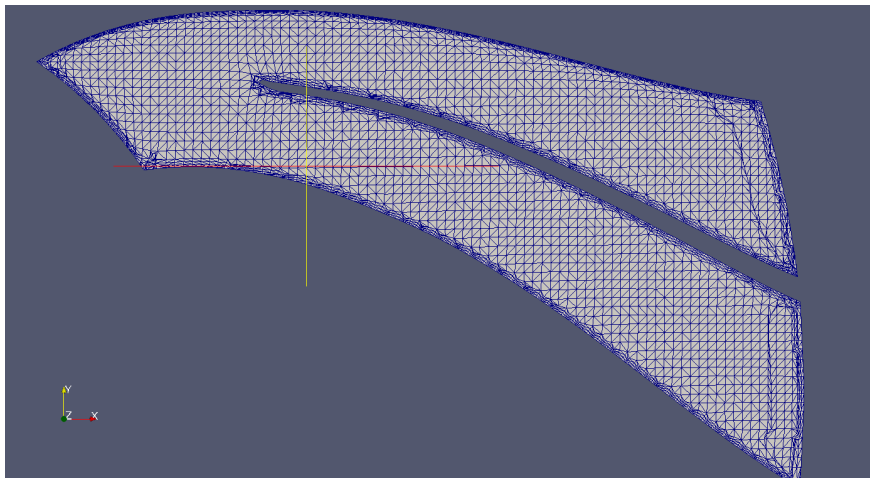

Figure 9: Runner cut section

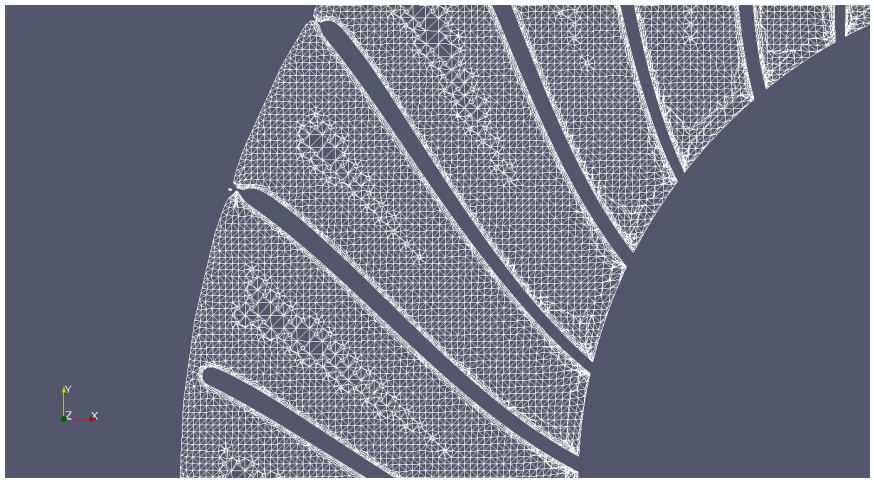

Figure 10: Cells collapsing between blades and runner inlet

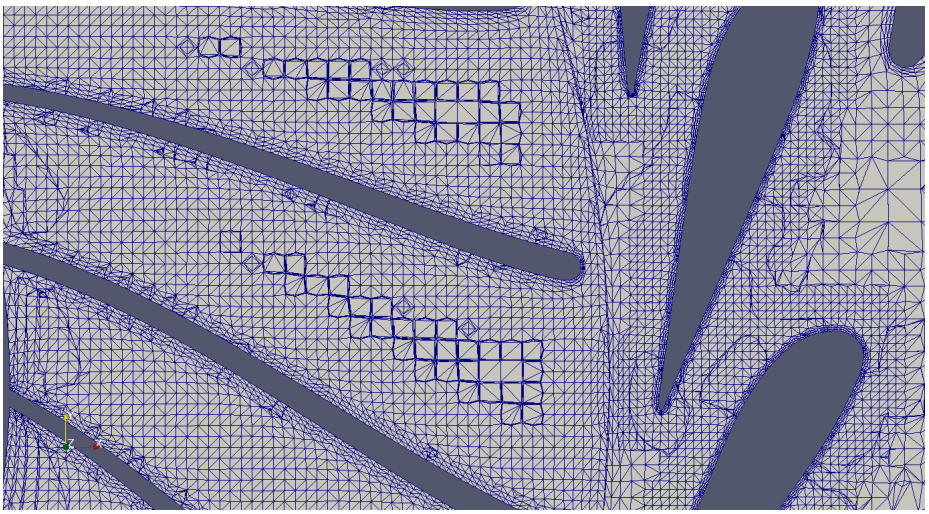

Figure 11: Runner-distributor cut section

Distributor

The distributor is meshed in a similar way. The steps are as follows:

```
run
cp jethro_openfoam/periodic/distributor/distributor.fms distributor/
cp jethro_openfoam/periodic/distributor/meshDict distributor/system/
cd distributor
cp -r 0_orig/ 0
cartesianMesh
checkMesh
paraFoam
cd ..
```

The distributor/system/meshDict file has the follows settings:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | cfMesh: A library for mesh generation           |
|  \\    /   O peration     |                                                 |
|   \\  /    A nd           | Author: Franjo Juretic                          |
|    \\/     M anipulation  | E-mail: franjo.juretic@c-fields.com             |
\*---------------------------------------------------------------------------*/

FoamFile
{
    version   2.0;
    format    ascii;
    class     dictionary;
    location  "system";
    object    meshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
surfaceFile "distributor.fms";
maxCellSize 8;
```

```
boundaryCellSize 4;
boundaryCellSizeRefinementThickness 4;
localRefinement
{
    "GUIDE_VANEGV01"
    {
        cellSize 2.5;
        refinementThickness 5;
    }
    "GUIDE_VANEGV28"
    {
        cellSize 2.5;
        refinementThickness 5;
    }
    "GUIDE_VANEGV_OUTLET"
    {
        cellSize 2;
        refinementThickness 4;
    }
    "GV_CYC.*"
    {
        cellSize 2.5;
        refinementThickness 5;
    }
    "GV_INLET"
    {
        cellSize 2;
        refinementThickness 4;
    }
}
boundaryLayers
{
    patchBoundaryLayers
    {
        "SPIRAL_CASINGCASING"
        {
             nLayers 3;
             thicknessRatio 1.2;
             maxFirstLayerThickness 3;
        }
        "GUIDE_VANEGV01"
        {
             nLayers 3;
             thicknessRatio 1.1;
             maxFirstLayerThickness 0.5;
        }
        "GUIDE_VANEGV28"
        {
             nLayers 3;
             thicknessRatio 1.1;
             maxFirstLayerThickness 0.5;
        }
    }
}
```

```
renameBoundary
{
    newPatchNames
    {
        "GV_INLET"
        {
            newName     disinlet;
            type        patch;
        }
        "GUIDE_VANEGV_OUTLET"
        {
            newName     disoutlet;
            type        patch;
        }
        "SPIRAL_CASINGCASING"
        {
            newName     diswalls;
            type        wall;
        }
        "GUIDE_VANEGV01"
        {
            newName     disguidevanes;
            type        wall;
        }
        "GUIDE_VANEGV28"
        {
            newName     disguidevanes;
            type     wall;
        }
        "GV_CYCLIC_TE_1"
        {
           newName discyclic1;
           type    patch;
        }
        "GV_CYCLIC_LE_1"
        {
           newName discyclic1;
            type    patch;
        }
        "GV_CYCLIC_TE_28"
        {
           newName discyclic2;
            type    patch;
        }
        "GV_CYCLIC_LE_28"
        {
           newName discyclic2;
            type    patch;
        }
    }
}
}
// ************************************************************************* //
```

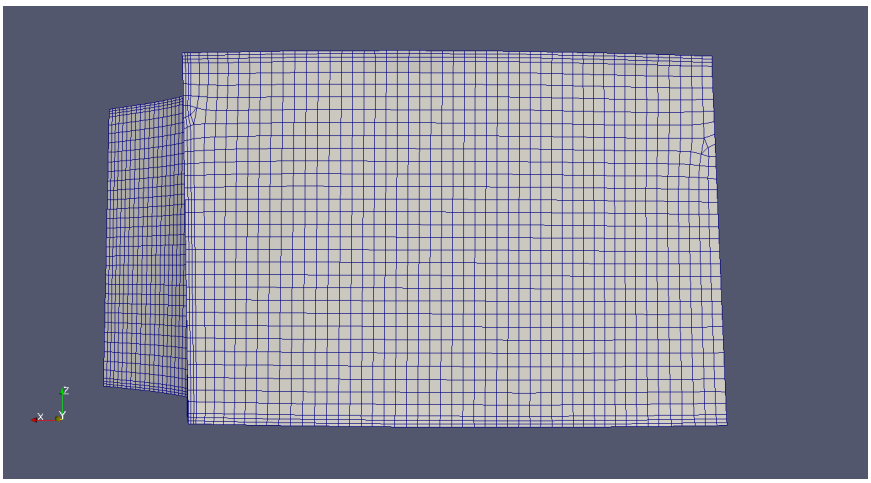

Figure 12: Guide vane inlet

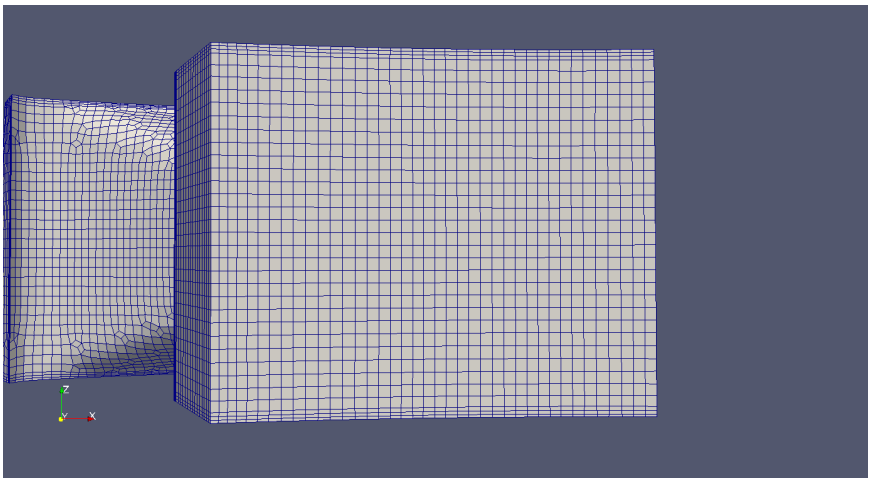

Figure 13: Guide vane outlet

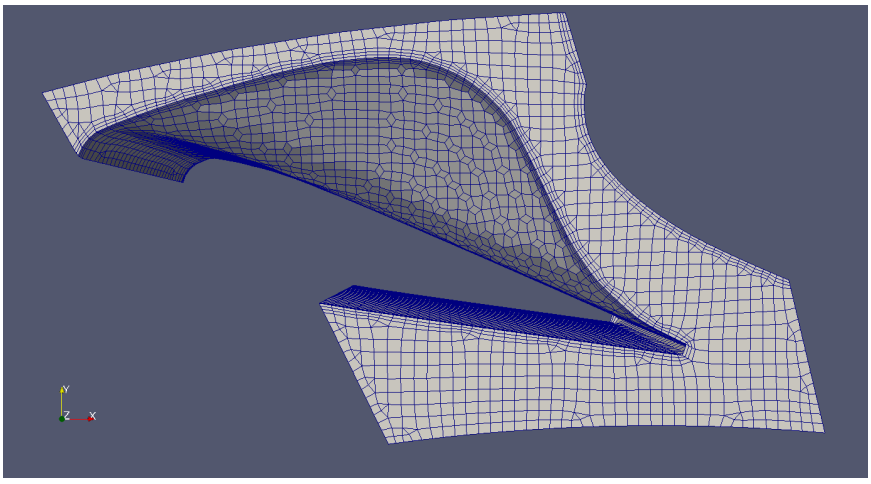

Figure 14: Guide vane

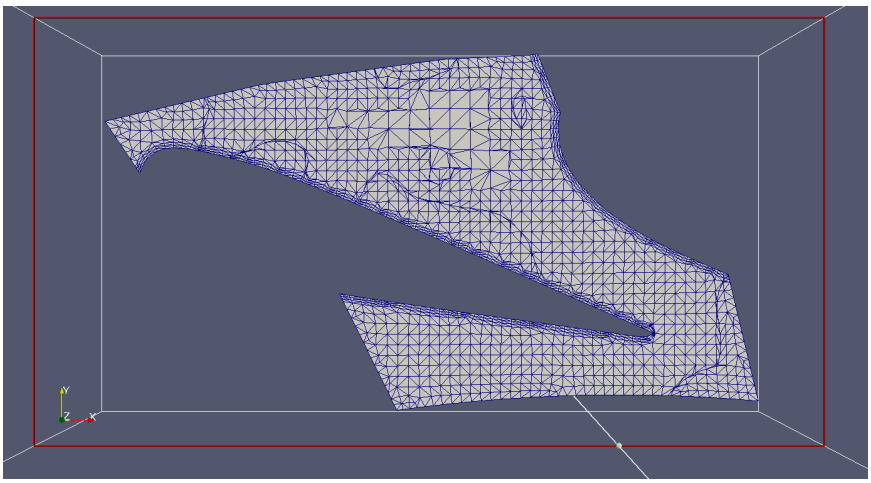

Figure 15: Guide vane cut section

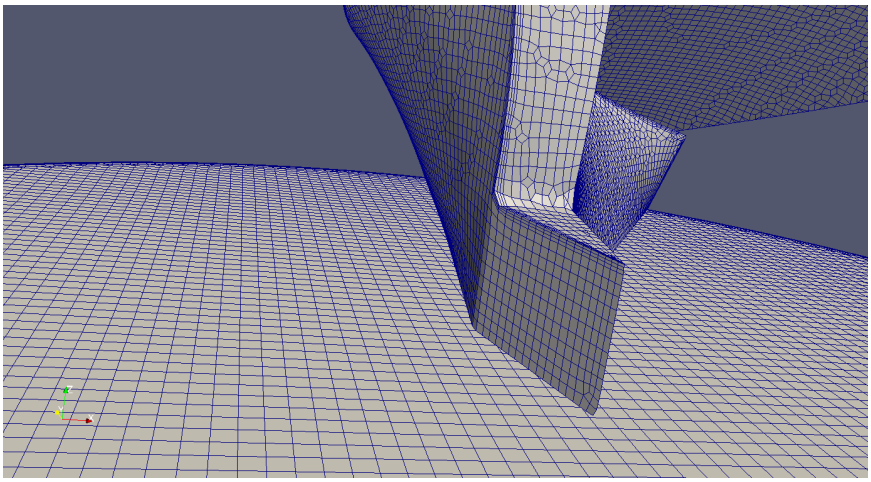

Figure 16: Runner draft tube interface

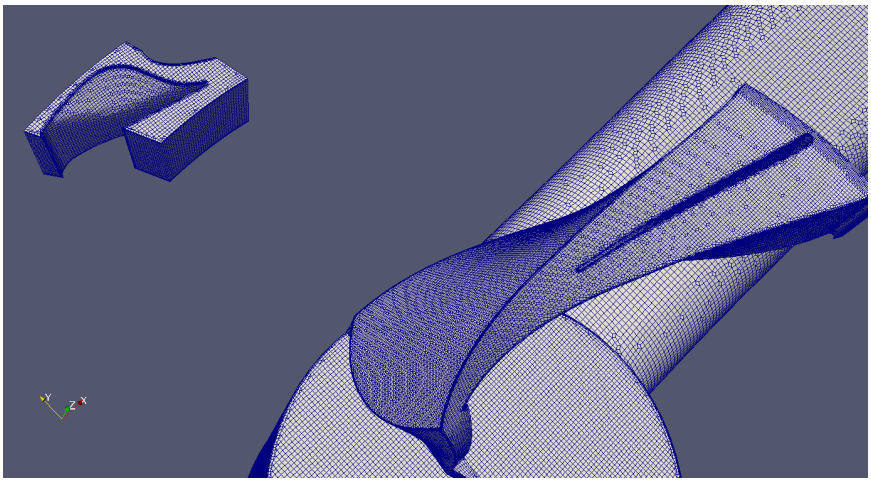

Figure 17: Runner Distributor

Figures 12, 13, 14 and 15 shows the mesh of a guide vane. This part is the easiest and fastest to mesh. The maximum skewness is 3.6. The total number of cells is 45000. The number of boundary layers is kept to 3.

Merging the meshes

We have meshed the 3 parts separately. Now, we need to join them together, so that we can simulate the flow in the turbine. This is done by the utility called mergeMeshes. We have a rotating runner. This runner will be defined as a cylindrical zone(done further down in the tutorial). In this zone, the MRF source terms will be implemented. If we had meshed all the 3 parts together, it would have been difficult to get cell faces tangential to the runner inlet.

```
run
cp -r distributor francis
mergeMeshes . francis . runner
rm -r francis/constant/polyMesh/
mv francis/1/polyMesh/ francis/constant/
rm -r francis/1/

mergeMeshes . francis . DT
rm -r francis/constant/polyMesh/
mv francis/1/polyMesh/ francis/constant/
rm -r francis/1/
scaleMesh -case francis 0.001
```

Figures 16 and 17 show the final mesh of the entire turbine. As mentioned before only one guide vane passage and one runner blade passage is modelled. The number of boundary layers are kept to 3 for tutorial purposes. No boundary layers were kept for faces that form the mixing plane nor the ones that are periodic. In figure 16, we notice that the sharp corners of the geometry have been rounded after meshing.

Modify the boundary file

The boundary file shows the definitions of the patches. This helps setting up the boundary conditions easier. Since we have mixing planes between distributor inlet and runner inlet and between runner outlet and draft tube inlet, their type is set to mixing plane. The ribbon patches creates imaginary ribbons on which the averages of properties are calculated. The cyclicGgi are periodic boundary conditions. Here, we model the flow through one guide vane passage and one runner blade passage. The rotational angle is the angle you need to rotate the particular patch so that the it becomes flush with the other periodic patch of the same part.
Note: Add the following to the individual boundaries(in francis/constant/polyMesh/boundary.gz). Delete the type setting and paste the following below it. Do not delete the nFaces and startFace setting. The remaining boundaries remain the same.

```
disoutlet
{
    type mixingPlane;
    shadowPatch     rninlet;
    zone            disoutletZone;
    coordinateSystem
    {
        type            cylindrical;
        name            mixingCS;
```

```
        origin          (0 0 0);
        e1              (1 0 0);
        e3              (0 0 1);
        inDegrees       true;
    }
    ribbonPatch
    {
        sweepAxis       Theta;
        stackAxis       Z;
        discretisation  bothPatches;
    }
}

discyclic1
{
    type cyclicGgi;
    shadowPatch     discyclic2;
    zone            discyclic1Zone;
    bridgeOverlap   false;
    rotationAxis    (0 0 1);
    rotationAngle   12.8571428;
    separationOffset (0 0 0);
}
discyclic2
{
    type cyclicGgi;
    shadowPatch     discyclic1;
    zone            discyclic2Zone;
    bridgeOverlap   false;
    rotationAxis    (0 0 1);
    rotationAngle   -12.8571428;
    separationOffset (0 0 0);
}

rninlet
{
    type mixingPlane;
    shadowPatch disoutlet;
    zone rninletZone;
}

rnoutlet
{
    type mixingPlane;
    shadowPatch     dtinlet;
    zone            rnoutletZone;
    coordinateSystem
    {
        type            cylindrical;
        name            mixingCS;
        origin          (0 0 0);
        e1              (1 0 0);
        e3              (0 0 1);
        inDegrees       true;
```

```
    }
    ribbonPatch
    {
        sweepAxis       Theta;
        stackAxis       R;
        discretisation  bothPatches;
    }
}

rncyclic1
{
    type cyclicGgi;
    shadowPatch     rncyclic2;
    zone            rncyclic1Zone;
    bridgeOverlap   true;
    rotationAxis    (0 0 1);
    rotationAngle   24;
    separationOffset (0 0 0);
}

rncyclic2
{
    type cyclicGgi;
    shadowPatch     rncyclic1;
    zone            rncyclic2Zone;
    bridgeOverlap   true;
    rotationAxis    (0 0 1);
    rotationAngle   -24;
    separationOffset (0 0 0);
}

dtinlet
{
    type mixingPlane;
    shadowPatch rnoutlet;
    zone dtinletZone;
}
```

Multiple reference frames and mixing plane

MRF simulations are used when you have at least one stationary and one rotating zone. It is a steady-state method that adds a source term to account for the Coriolis forces and centrifugal forces in the rotating zone. The interfaces need to be handled to exchange information. In this tutorial, we have used the mixing plane interface. The mixing plane computes and exchanges the circumferential average of all the quantities at the interface. More information on the mixing plane model can be found in the master thesis of Lucien Stoessel[1] and its implementation in OpenFoam can be found in Page et.al.[2]

Boundary conditions

The setup of the boundary conditions are similar to that of the axial turbine case. A modified inlet velocity boundary condition called rotatingWallTangentialVelocity is however used, since the

present case has a radial inlet where a swirl should be applied. To set up this boundary condition, the following steps are employed:

```
run
mkdir -p $WM_PROJECT_USER_DIR/src/finiteVolume/fields/fvPatchFields/derived
cp -r jethro_openfoam/rotatingWallTangentialVelocity/ \
$WM_PROJECT_USER_DIR/src/finiteVolume/fields/fvPatchFields/derived
cd $WM_PROJECT_USER_DIR/src/finiteVolume/fields/fvPatchFields/derived
cd rotatingWallTangentialVelocity
wclean
wmake libso
run
cp -r jethro_openfoam/0_orig/ francis/
```

Open the francis/0_orig/U file to see how the inlet velocity condition is given:

```
disinlet
    {
        type            rotatingWallTangentialVelocity;
        origin          (0 0 0);
        axis            (0 0 1);
        omega           0;
        axialVelocity   0;
        radialVelocity  -1.618;
        tangVelocity    -2.427;
        value           uniform (0 0 0);
    }
```

The inlet k and ϵ values were set to 0.128 and 153.16 respectively. These same values were used throughout the domain, similar to the axial turbine case. The discyclic1, discyclic2, rncyclic1 and rncyclic2 were set to cyclicGgi type of boundary conditions. The turbulent viscosity ν_t was set to 10.νThe rest of the boundary conditions are similar to the axial turbine case. A detailed description on how these values were calculated can be found in [1]

Constant directory

Set the kinematic viscosity ν to 9.57e-07 in the francis/constant/transportProperties files. Make sure you have no nonRotatingPatches in the MRFZones file as all the parts in your runner are rotating. We run our simulation without any tip clearance between the shroud and the runner blades in order to simplify the flow. Make sure RNG k-ϵ turbulence model is selected in the francis/constant/RASProperties file.

System directory

In the francis/system/controlDict file paste

```
libs ( "libRotatingWallTangentialVelocityFvPatchVectorField.so" );
```

Change the endTime from 200 to 1000.
Keep the francis/system/fvSchemes file the same as for the axialTurbine tutorial. For some rotating machinery simulations, it is good to use the Gauss linearUpwind Gauss convective scheme to help with faster convergence. For this case, it worked well with the standard settings.
In the francis/system/fvSolution, change the tolerance and relTol to 1e-06 and 0.01 for the pressure and 1e-05 and 0.1 for the rest. Change the nNonOrthogonalCorrectors to 2 for the SIMPLE scheme and reduce the under-relaxation factors of k, ϵ and U to 0.5.

Creating zones

This sections explains the creation of cellZones and FaceZones. Cell zones are needed to define the rotor zone. This is where you will define the MRF region. Face zones are needed for the cylicGgi and mixingPlane patches. First create cell sets and face sets. Delete whatever is there in the francis/setBatchGgi file. Add the following to it:

```
cellSet rotor new cylinderToCell (0 0 0.05) (0 0 -0.18) 0.31523
faceSet disoutletZone new patchToFace disoutlet
faceSet rninletZone new patchToFace rninlet
faceSet rnoutletZone new patchToFace rnoutlet
faceSet dtinletZone new patchToFace dtinlet
faceSet rncyclic1Zone new patchToFace rncyclic1
faceSet rncyclic2Zone new patchToFace rncyclic2
faceSet discyclic1Zone new patchToFace discyclic1
faceSet discyclic2Zone new patchToFace discyclic2
quit
```

Now copy the Allrun script to the francis folder to run the simulations

```
run
cp -r jethro_openfoam/Allrun francis/
cp -r jethro_openfoam/Allclean francis/
```

Running the solver

Use ./Allrun to run the case.

```
run
cd francis
./Allrun
```

This generates a log.MRFSimpleFoam file. To view the residuals, open a new terminal and go to the turbine folder. THe following steps are:

```
OFextend32
run
cd francis
pyFoamPlotWatcher.py log.MRFSimpleFoam
```

Results and discussions

Figure 18 shows the residuals of the simulation. The simulations were let to run for a limited number of iterations due to time constraints. As a result the velocity and pressure contours shown in figures 19, 20 and 21 respectively are smeared. Also the number of cells used are too less. To increase the accuracy a finer mesh will have to be used. This tutorial doesn't focus on the accuracy of the results.

Figure 19 shows the U_z at the entry to the draft tube. The center of the draft tube inlet has a zero velocity region followed by high velocity region. Note that the $+Z$ direction is upwards. Hence the $-3ms^{-1}$ is downwards. The zero velocity section is due to a small wall present at the entry of the draft tube.
Figures 20 and 21 show the velocity magnitude and static pressures respectively of the guide and runner blades. Note that the runner spins in the clock-wise direction when looking from above (in the -Z direction). You can see that the velocity increases through the guide vanes and the static

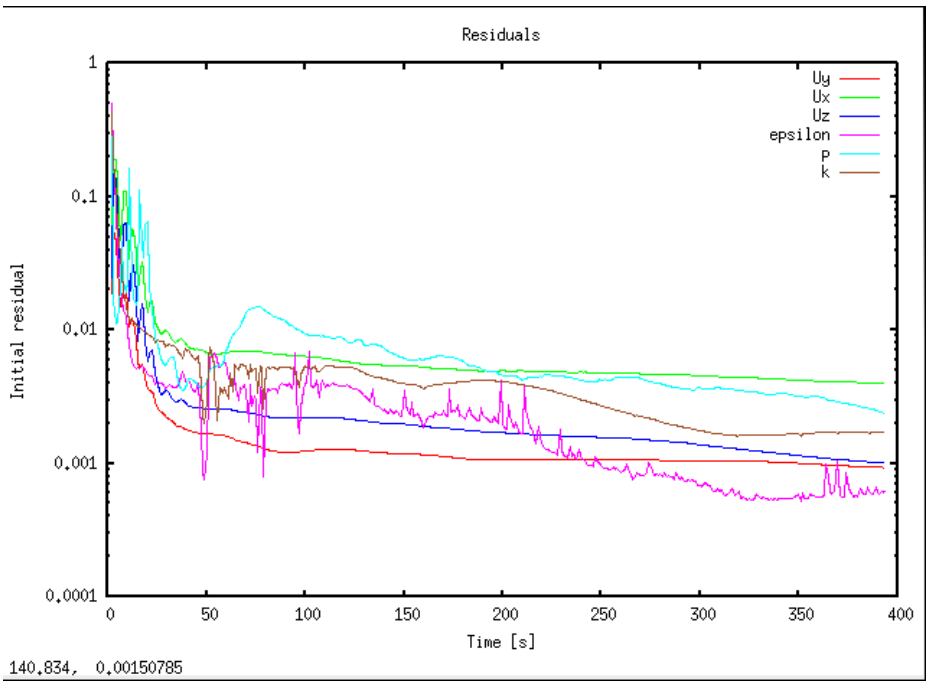

Figure 18: Residuals

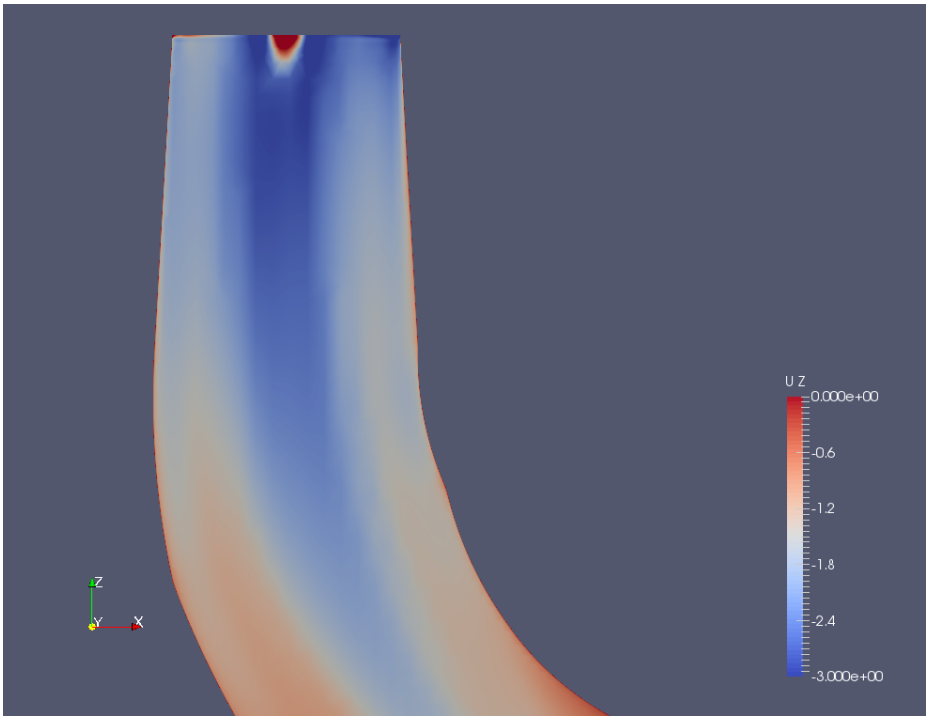

Figure 19: U_z contours at the entry to the draft tube

pressure decreases. To see the results of the simulations on a finer mesh, refer to the Master thesis by Lucien [1].

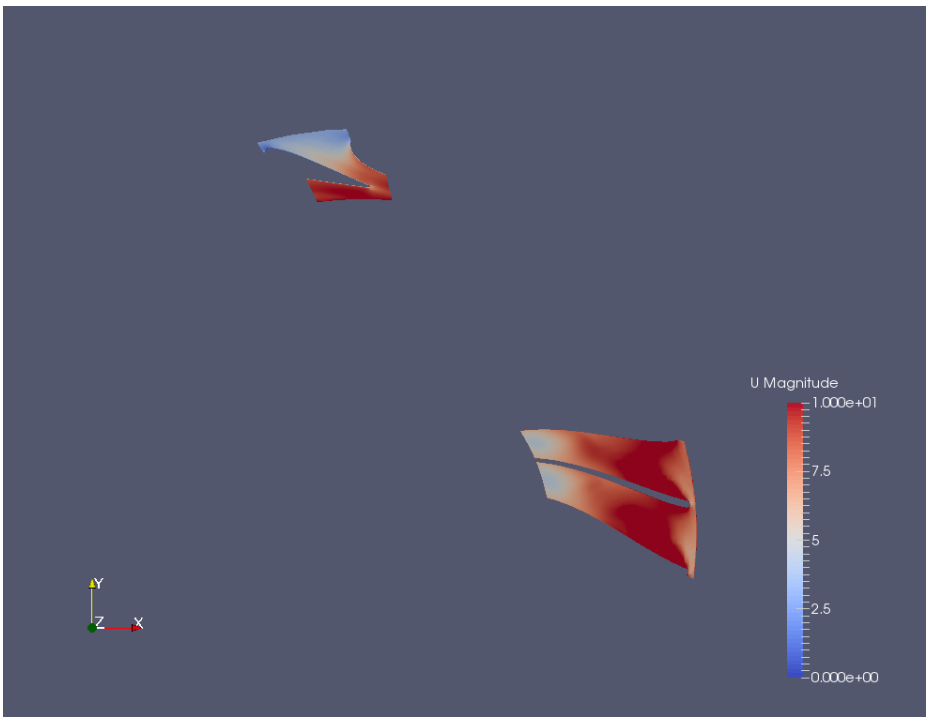

Figure 20: Velocity contours at the entry to the draft tube

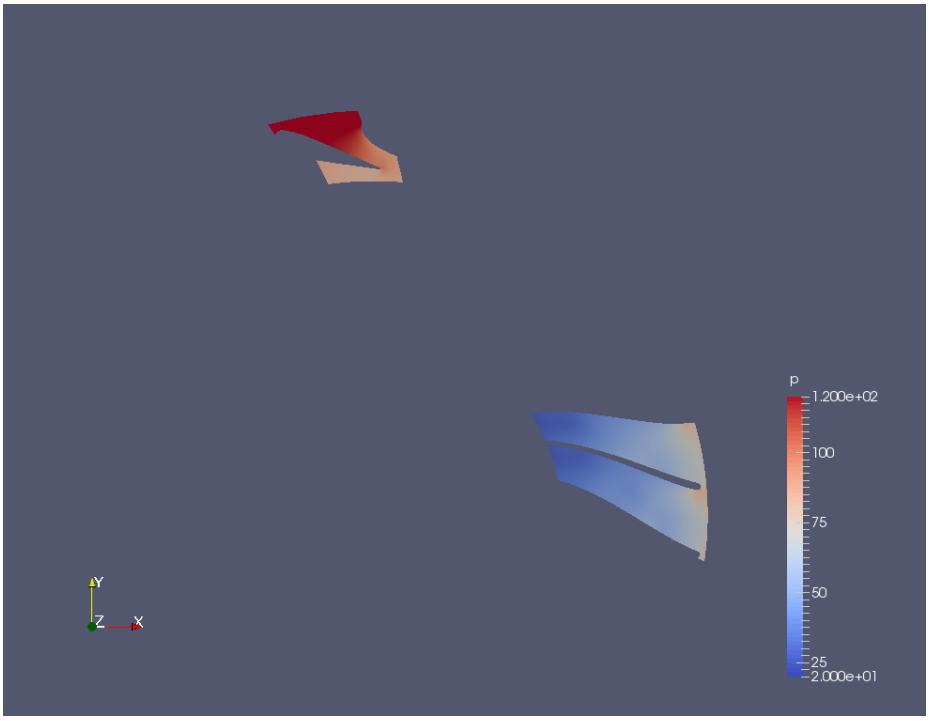

Figure 21: Pressure contours on Z plane at Z=0

Conclusions

cfMesh is an easy to use meshing tool to get a first mesh, although it is difficult to get an ideal mesh. A properly cleaned up CAD is not required to generate a mesh. Also, the CAD can have

small holes in them. As far as your mesh size is greater than the gap size, there will be no problems in meshing. However, there are some issues. If the distance between 2 patches in a face are small, the cells collapse. This requires the use of the keepsCellsIntersectingPatches option, which increases the meshing time. Also, cfMesh finds it difficult to capture sharp corners. It ends up distorting the geometry, creating non-orthogonal faces.
This tutorial creates a course mesh of the Francis-99 geometry. The settings can be easily changed by changing the values in the meshDict file. The simulated results are poor. This is because the mesh is course and the simulations need to run for a longer time. However, accuracy isn't the target this tutourial.

Bibliography

[1] Lucien Stoessel *Numerical simulations of the flow in the Francis-99 turbine* Chalmers University of Technology

[2] Page, M., Beaudoin, M., and Giroux, A.-M *Steady-state Capabilities for Hydroturbines with OpenFOAM.* International Journal of Fluid Machinery and Systems 4(1) , Jan. 2011, 161 to 171. doi : 10.1088 / 1755 - 1315/12/1/ 012076.

Study question

Some interesting questions are as follows:

- What are the different types of cells that can be generated using cfMesh and what are the utilities used to create them?
- How do I change the size of a meshed geometry, which is in the order of 10m to 10mm?
- Which are the 2 compulsory options in cfMesh? What do they represent?
- What is the syntax for creating a cylindrical cell set? How do you create zones from sets?
- Which cfMesh option can be used to squeeze cells in-between a small gap?