OpenFOAM®

Simulation of the first stages of a turbofan using OpenFOAM

Degree: Aerospace Engineering

Course: Application of Open-Source CFD to Engineering Problems

Delivery date: 09-12-2016

Students:

Herrán Albelda, Fernando

Martínez Viol, Víctor

Morata Carranza, David

Contents

1 Introduction

During the development of the course *'Application of Open-Source CFD to engineering problems'* we have learned the basics of how to use and solve real-world cases and situations related with fluid mechanics using OpenFOAM, an open source CFD tool. The first days of the course, several possible projects were presented and we had to choose one of them and make a report. After agreeing with the professor, we decided to simulate an option that was not on that list. Since we are really interested in propulsion, we thought that it was a good idea to try to simulate the flow inside the first stages of compression of a turbofan engine.

These type of engines are the most used propulsion system in the aerospace industry. It presents several advantages to other systems such as the turbohelix or the turbojet; for example, this kind of engine takes advantage of the flow that goes through the fan (that cannot be more compressed or heated given that the combustion chamber has a limited volume) and comes out through the rear nozzle. This results in a higher thrust for the same amount of fuel that is burned. Thus, it is no surprise that state-of-the-art planes such as the Airbus 380 or the Boeing 747 use this kind of propulsion system.

In order to do a realistic simulation, we have been gathering lots of information of this type of engines and their typical working conditions. Also, we have been searching for information about how could we solve a geometry that is rotating. Several simulations and comparisons will be presented in this report to analyze the validity of the results obtained.

It will be presented in this report how to use the Multiple Reference Frame utility (MRF) as well as the boundary conditions, the mesh generation and the solver that has been used for the case. Additionally, several hypotheses will be considered in order to alleviate the computation time (given that this project has been simulated in a laptop).

2 Case

2.1 Description of the case

The aim of the current project is to analyze the airflow inside the first stages of a turbofan. The turbofan consists of an initial stage where a fan is placed. This first stage increases the pressure of the air that goes through it and, as it can be seen in 2.1, a part of the incoming airflow goes to the low pressure compression stage (main or primary flow) and the other goes to the conducts (secondary flow). As the secondary flow advances through the conduct, the main flow enters the engine core where it goes through the low pressure and high pressure compression stages. By increasing the pressure of the air, the density also increases; thus, a higher mass flow can be mixed with the fuel and burned in the combustion chamber. After this process, the gases go through the turbines, interchanging energy with them (pressure and velocity, mainly), and they arrive at the nozzle, where the air is accelerated, and a thrust force is obtained.

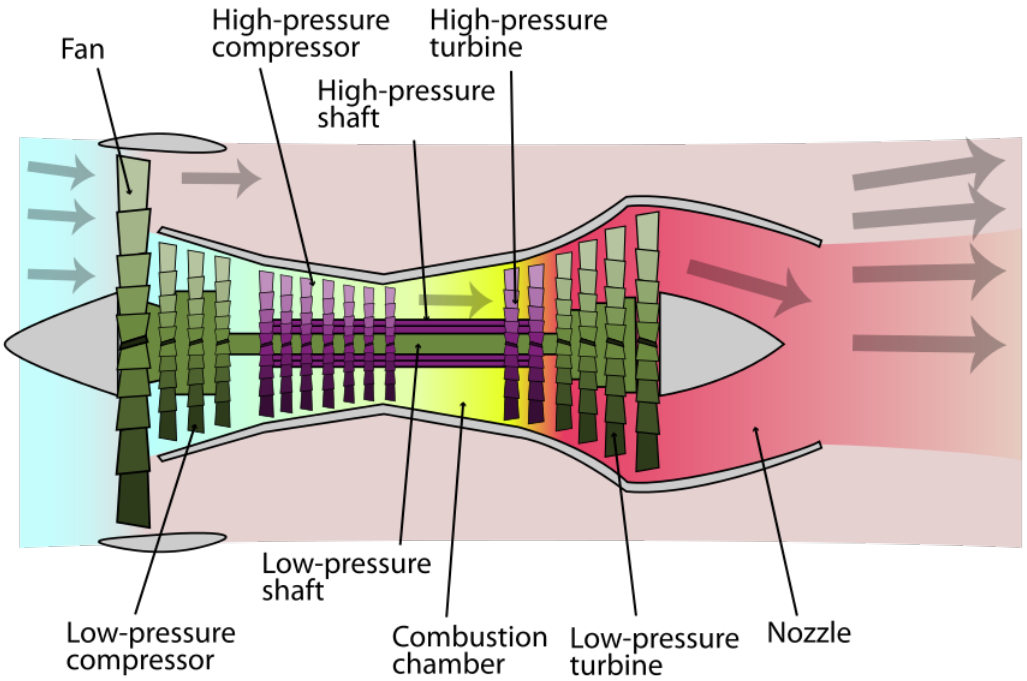

Figure 2.1: Parts of a turbofan

To simulate the flow in the first stages of the turbofan, we have downloaded the following turbine model from *GrabCad*, shown in 2.2. This model is pretty similar with the one shown above (2.1) and the simulation will take place between the back side of the fan and the back side of the second compressor. There are several realistic and potentially functional models on *GrabCad*; however, this model has been selected because it did not present incompatibilities with *SolidWorks* and *Salome*. Given that all the parts of the model have to be clearly differentiated in order to export them as *STL* files, we could not use an assembly.

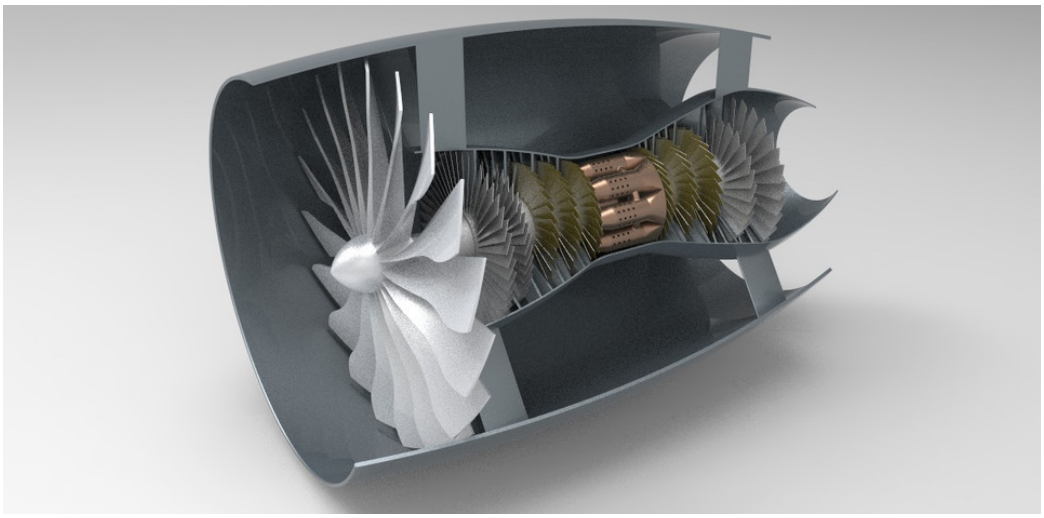

Figure 2.2: Model used

2.2 Hypotheses

Given that we are not able to use a supercomputer and the computational power that is available to us is very limited since we are using our own personal computers to run this simulation, several hypotheses have to be made to alleviate the calculation time, as mentioned before. Some of them can be assumed and some other will make the case a non-realistic project. These hypotheses are presented below.

- Viscous flow
- Incompressible flow

- Newtonian flow
- Stationary flow

Clearly, the flow is not incompressible in reality; the aim of the turbofan is to compress the air to seek for a more efficient combustion. However, all of the compressible solvers are really difficult to use and the computational time is also higher given that we have a high number of control volumes (as discussed in the next section) due to a really complex geometry. Thus, although it will not be a completely real flow, we will be also able to see how the low pressure section of the engine increases the pressure of the flow and how the velocity field changes as it goes through the turbine.

The simulation will take into account that it is a tridimensional, that it behaves as a newtonian fluid and it will be run under stationary flow conditions (that is: the velocity in the inlet is always uniform and has a fixed value). With these hypothesis, we can begin to discuss the geometry of the *blockMesh* as well as the refined mesh and the boundary conditions and final results within the next sections.

3 Pre-processing

3.1 Mesh generation

3.1.1 blockMesh

The definition of the *blockMeshDict* is the first part that needs to be defined in order to simulate the case. Only the first two stages of the turbofan engine will be simulated, so we have to make sure that the *blockMesh* includes them.

The domain of the mesh is a 0.58x2.3x2.3m rectangular prism and the vertices are as follows:

```
vertices
(
    (0.77 0   0)
    (1.35 0   0)
    (1.35 2.3 0)
    (0.77 2.3 0)
    (0.77 0   2.3)
    (1.35 0   2.3)
    (1.35 2.3 2.3)
    (0.77 2.3 2.3)
);
```

Once the boundaries of the *blockMesh* are defined, the number of cells that it will have has to be set. It has to be taken into account that a very dense mesh will not be efficient when simulating the case. On the other hand, a very coarse mesh will not be efficient either because additional divisions will have to be set when generating the refined mesh with *snappyHexMesh*. Thus, a compromise solution between a mesh with a very high number of cells and a very low number of cells has to be attained.

This basic mesh has been divided every 0.05m. It means that we have done 12 divisions in the x direction, 46 on the y direction and 46 more on the z direction. Obviously, the vertex numeration has been kept the same as the one that comes as default in every tutorial case; so, it was easier to

define the inlet face of the *blockMesh* as well as the outlet face that will be used to define the boundary conditions.

The dense of the mesh is as follows:

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (12 46 46) simpleGrading (1 1 1)
);
```

Finally, the differents faces of the mesh must be define depending on if they are the inlet and outlet faces or the lateral faces.To do this, the vertices are numbered according to their appearance in the *blockMeshDict* and the faces are defined by those numbers. In our case, the faces are as follows:

```
boundary
(
    frontAndBack
    {
        type patch;
        faces
        (
            (3 7 6 2)
            (1 5 4 0)
            (0 3 2 1)
            (4 5 6 7)
        );
    }
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
```

```
        faces
        (
            (2 6 5 1)
        );
    }
);


/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
    (0.77 0   0)
    (1.35 0   0)
    (1.35 2.3 0)
    (0.77 2.3 0)
    (0.77 0   2.3)
    (1.35 0   2.3)
    (1.35 2.3 2.3)
    (0.77 2.3 2.3)
);

blocks
(
```

```
    hex (0 1 2 3 4 5 6 7) (12 46 46) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    frontAndBack
    {
        type patch;
        faces
        (
            (3 7 6 2)
            (1 5 4 0)
            (0 3 2 1)
            (4 5 6 7)
        );
    }
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
);

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

The log obtained when the mesh has been generated is as follows:

```
Writing polyMesh
--------
Mesh Information
--------
boundingBox:  (0.77 0 0) (1.35 2.3 2.3)
nPoints:  28717
nCells:  25392
nFaces:  79396
nInternalFaces:  72956
--------
Patches
--------
patch 0 (start:  72956 size:  2208) name:  frontAndBack
patch 1 (start:  75164 size:  2116) name:  inlet
patch 2 (start:  77280 size:  2116) name:  outlet

End
```

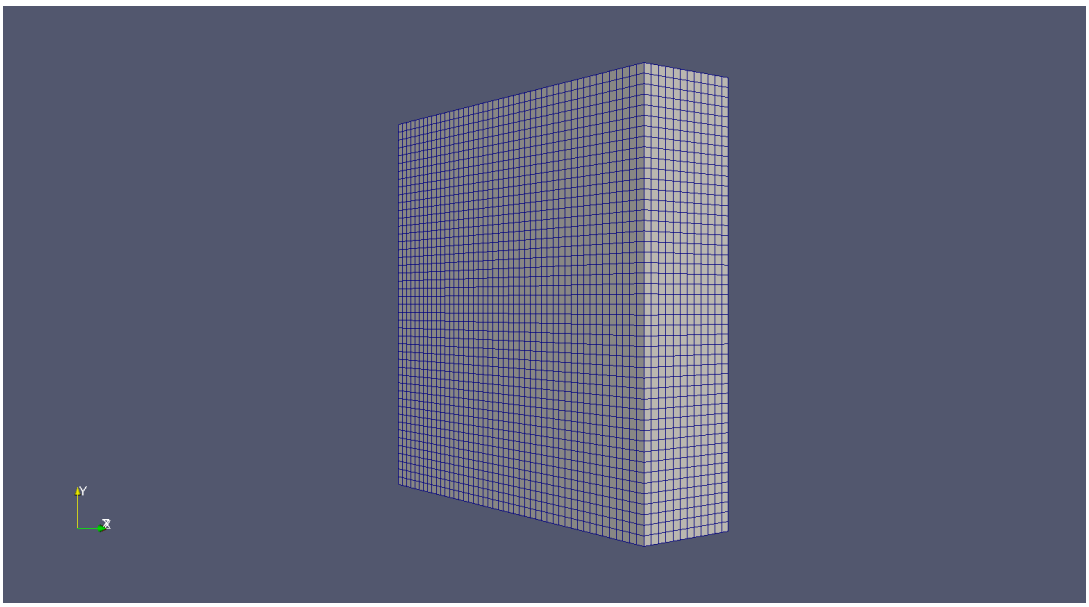

Figure 3.1: blockMesh

3.1.2 Mesh refinement

To refine the mesh, the *snappyHexMesh* utility, included in OpenFoam, which is used and several parameters have to be modified in order to obtain a dense mesh that is suitable for the simulation of this complex geometry. Additionally, it has to be considered that a particular geometry is has a relative velocity; this is, the rotor is rotating, and so the first and second stages of the Low Pressure Compressor of the turbofan engine, while the nacelle and the combustor are static.

FALTA EXPLICAR SURFACE EXTRACT

A comparison between the coarse mesh and the refined mesh generated is presented below.

3.1.2.1 Coarse mesh

The number of control volumes in the mesh is 40.

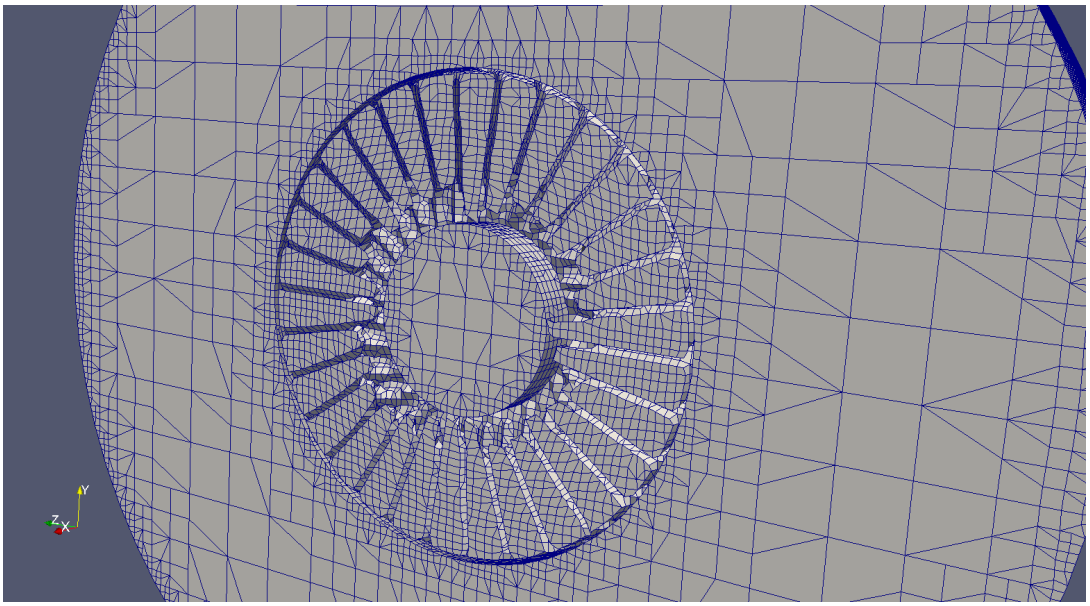

Figure 3.2: Detail of the coarse mesh

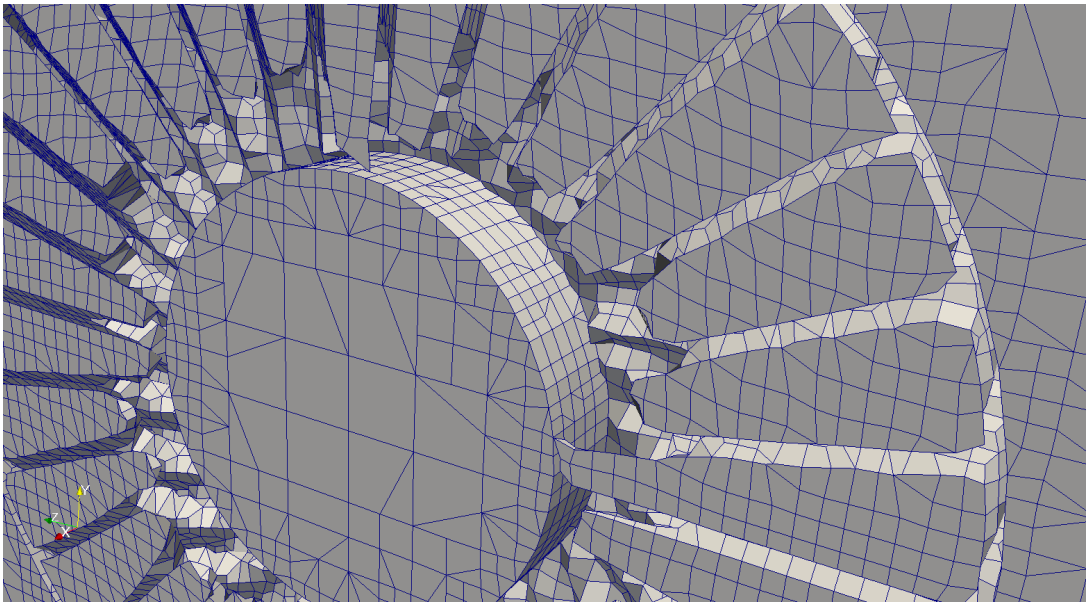

Figure 3.3: Detail of the coarse mesh

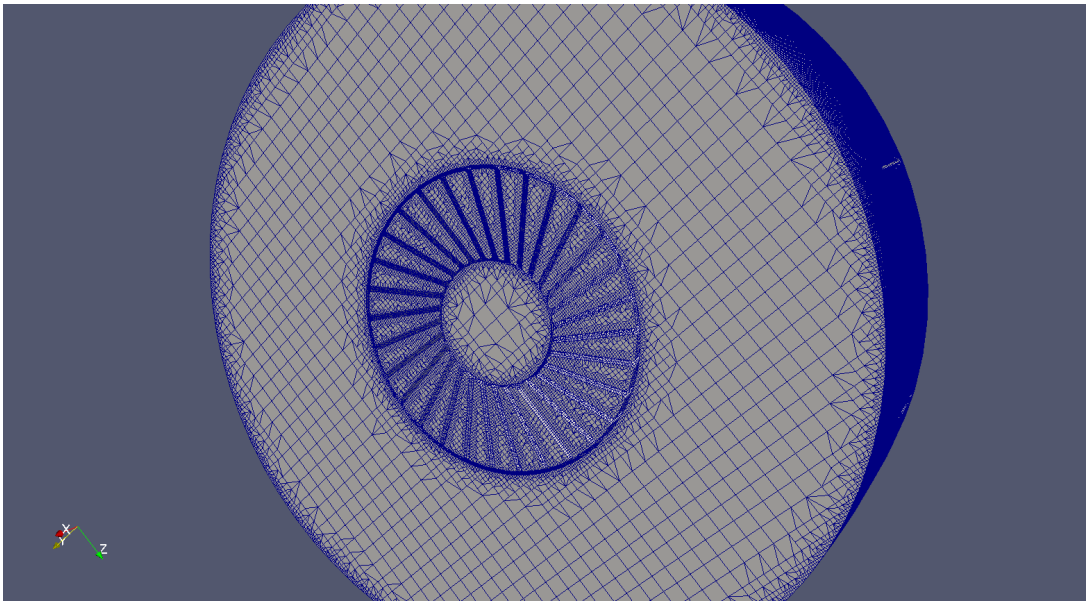

Figure 3.4: Detail of the standard mesh

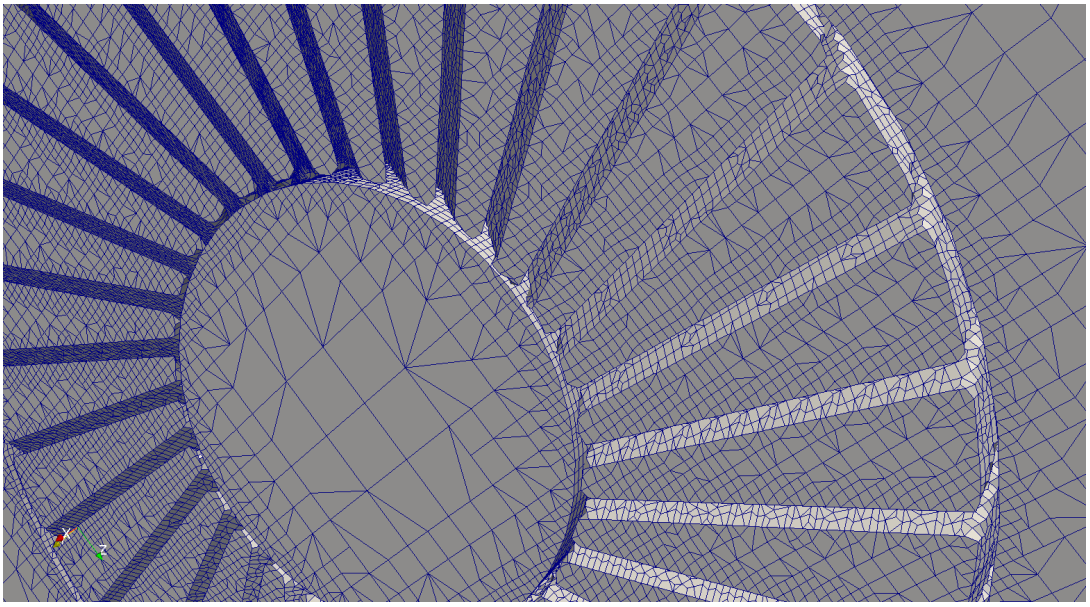

Figure 3.5: Detail of the standard mesh

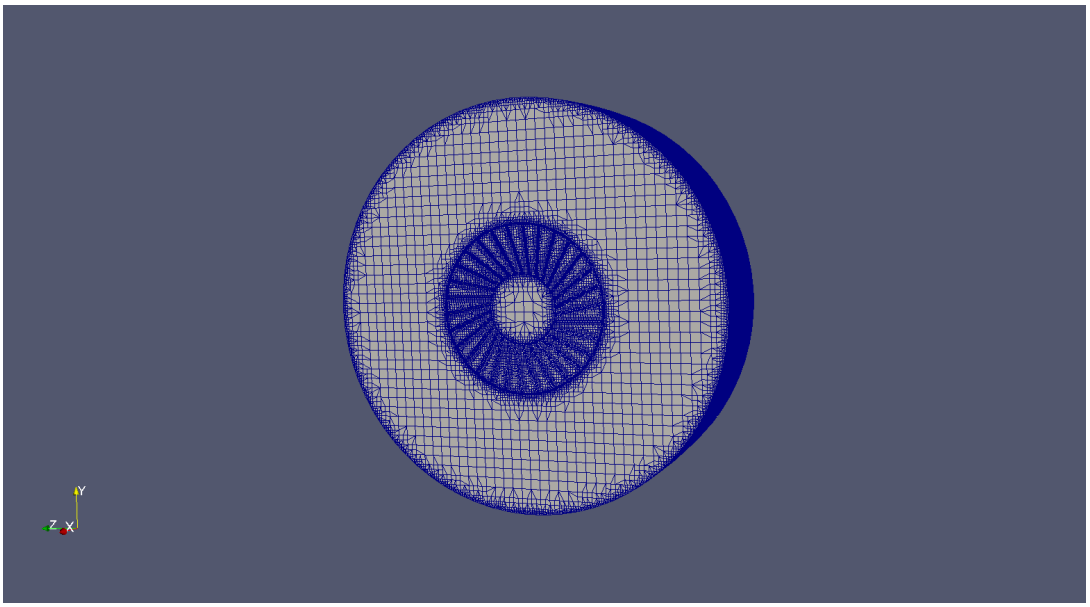

Figure 3.6: Detail of the coarse mesh

3.1.2.2 Standard mesh

3.1.3 Dense mesh

3.2 Boundary conditions

The known initial values of the the pressure and the velocity have to be defined in the program to carry out the simulation.

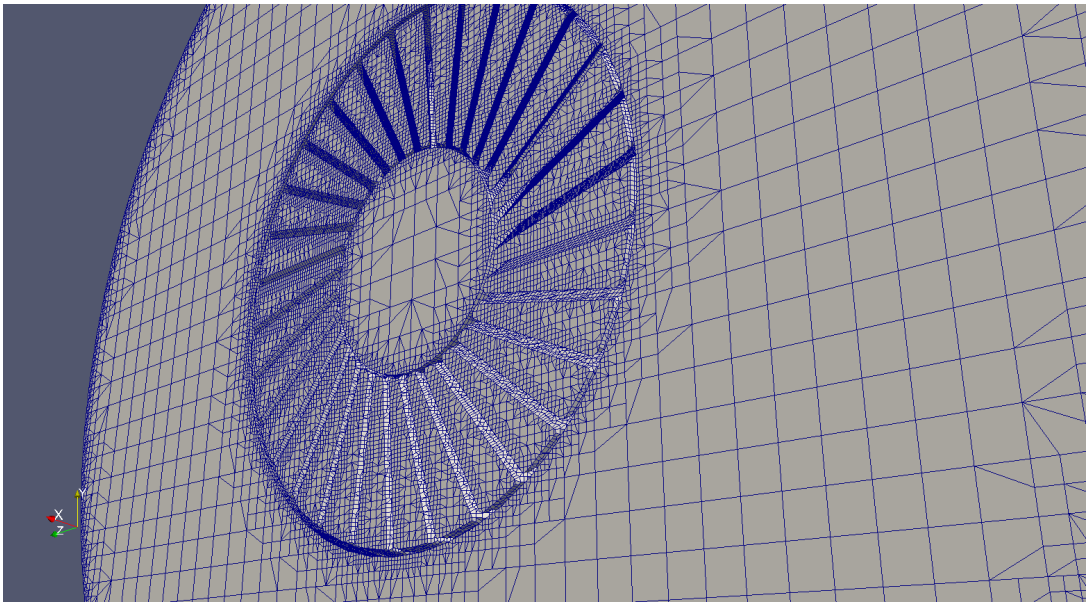

Figure 3.7: Detail of the dense mesh

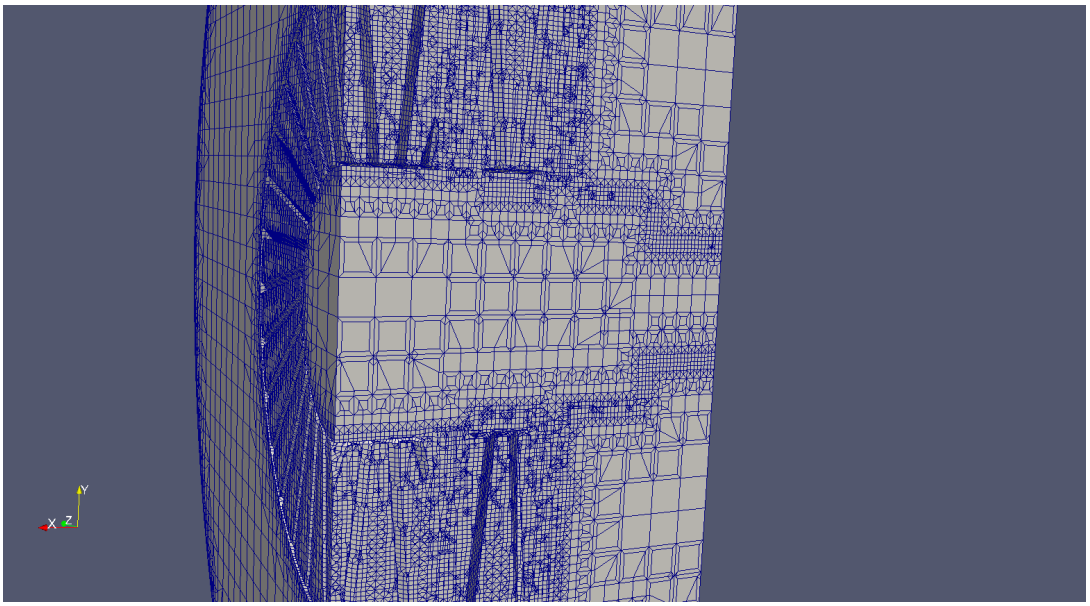

Figure 3.8: Detail of the dense mesh

We consider that the velocity at the output of the mesh, in the direction X, is $30m/s$. To change this parameter, we have to modify the file **0.orig/U**.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
|  \\    /   O peration     | Version:  4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                    |
|    \\/     M anipulation  |                                               |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    LPSpool.stl
    {
        type            noSlip;
    }

    NacelleStator.stl
    {
        type            noSlip;
    }

    inlet
    {
        type            fixedValue;
value uniform (30 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }
}
```

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

Another consideration is that the pressure at the inlet of the mesh is XXXX. To change this parameter, we have to modify the file **0.orig/p**.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    LPSpool.stl
    {
        type            zeroGradient;
    }

    NacelleStator.stl
    {
        type            zeroGradient;
    }

    inlet
    {
        type            fixedValue;
```

```
    value     uniform 0;
    }

    outlet
    {
        type            zeroGradient;
    }
}

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

3.3 Properties

The properties of the flow have to be defined as explained in the hypothesis section. The *transportModel*is Newtonian and the kinematic viscosity *nu* is $1.5e-05m^2/s$.

To change all these parameters, we have to modify the file **constant/transportProperties**.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 1.5e-05;
```

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

Another hypothesus made have been that the flow of the simulation is laminar. To impose this hypothesis *simulationType* is laminar.

To change all these parameters, we have to modify the file **constant/turbulenceProperties**.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType laminar;

RAS
{
    RASModel            kOmegaSST;

    turbulence          on;

    printCoeffs         on;
}

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

3.4 Rotation

How we have explained before, the project is based in the study of a flow inside the first stages of compression of a turbofan engine. Therefore, an important consideration is that the rotor is spinning at high speed. To impose this condition we must work with the Multiple Frame Reference (MRF) method. This method is based on adding source to momentum equation.

We will work with the **constant/MRFProperties** file and we must change the appropriate parameters. This method is based on the right hand rule, therefore the *axis* of rotation is $(1, 0, 0)$. We consider that the rotor rotor rotates at 5000 rpm, which is a common value for turbofan engines in flight conditions. It should be noted that the units used must be those of the International System, so we must convert 5000 rpm to rad/s. Furthermore, we must indicate the origin point in the axis of rotation, in our case it is (0 1.1675354 1.15542633). Finally, in section nonRotatingPatches we write list of patches that they are not rotating, in our case we only have the NacelleStator.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      MRFProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

MRF1
{
```

```
    cellZone    rotor;
    active      yes;

    // Fixed patches (by default they 'move' with the MRF zone)
    nonRotatingPatches (
NacelleStator.stl
);

    origin    (0 1.1675354 1.15542633);
    axis      (1 0 0);
    omega     523.5987756;
}

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

3.5 Control

First of all, the initial and final times of the simulation have to be defined. In our case, the *startTime* is 0 s and the *endTime* is 20 s. After this, the time step fot he simulation have to be defined. We are interested on simulate each second of the simulation but we want that the programm provides us the simulation data each 5 seconds, therefore the *deltaT* is 1 s and the *writeInterval* is 5 s.

To change all these parameters, we have to modify the file **system/controlDict**.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
```

```
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     simpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         20;

deltaT          1;

writeControl    timeStep;

writeInterval   5;

purgeWrite      0;

writeFormat     binary;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

/*functions
{
    #include "streamLines"
    #include "wallBoundedStreamLines"
    #include "cuttingPlane"
    #include "forceCoeffs"
}
```

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

4 The problem

EMPTY

5 Simulation

EMPTY