Gas Dynamics
Term Project
Supersonic flow inside a nozzle and over a bluff-body

Erdem Çalışkan
030150168

Istanbul Technical University
Mechanical Engineering

Contents

1 Acknowledgments

Computing resources used in this work were provided by the National Center for High Performance Computing of Turkey (UHeM) under grant number 4007172019.

2 Introduction

Air flows from a reservoir through a converging diverging nozzle to another reservoir where there is a bluff-body at the exit plane. Flow is steady, in-viscid and laminar. Also, walls are considered adiabatic.

Conditions at reservoirs are:

Upstream Reservoir	Downstream Reservoir
$P_0 = 3\,bar$	$P_b = 0.05\,bar$
$T_0 = 773\,K$	

Tab. 1: Initial conditions

Geometry is provided as *.stl* file, section view can be seen in Figure 2.1.

This problem is suitable to solve with *OpenFoam*, an open-source CFD platform. Mesh creation can be achieved with *blockMesh* and *snappyHexMesh* tools and *rhoCentralFoam* is a compressible flow solver which is suitable for this problem.

3 Geometry and Meshing Procedures

Geometry is a 3D control volume consists of 4 patches. Inlet and outlet stands as a reservoir, wall is a convergent divergent nozzle and pin is the bluff-body to investigate the shock around it. This geometry can be meshed with *snappyHexMesh* tool provided by *OpenFoam.*

First of all, to mesh this control volume, a *blockMesh* should be created. To do that, there should be several directories in *OpenFoam.*

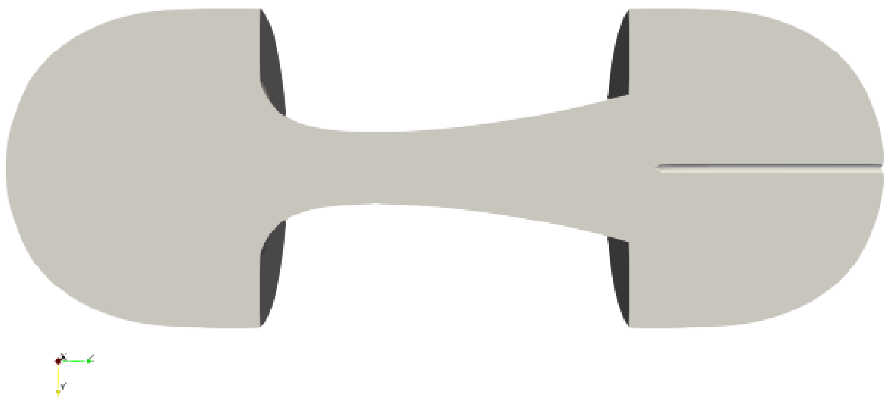

Fig. 2.1: Section of the provided geometry

Directories are the files and folders *OpenFoam* works on. A standard analysis consists of three basic folders 0, constant and system.

0 folder contains initial conditions. In our case pressure (p), temperature (T) and velocity (u) should be defined to run solver.

constant folder contains the properties that are not changing during analysis such as thermodynamic properties and mesh. Geometry files should be in triSurface folder inside constant.

system folder contains files that are needed to run the simulation such as numerical schemes, simulation control dictionaries etc.

blockMeshDict is the dictionary to create *blockMesh.* In this dictionary, a hexahedral block, larger than our control volume, is defined with vertices. After that, block is divided into smaller blocks which creates the basis for the mesh. Boundary types and names are also defined in this dictionary, yet they are not used since they are also defined in *snappyHexMesh.*

blockMeshDict file used in analysis;

```
convertToMeters 1;

    vertices
    (
        (0.5 -0.5 2)
        (0.5 -0.5 -0.8)
        (-0.5 -0.5 -0.8)
        (-0.5 -0.5 2)
        (0.5 0.5 2)
        (0.5 0.5 -0.8)
        (-0.5 0.5 -0.8)
        (-0.5 0.5 2)
    );

    blocks
    (
        hex (0 1 2 3 4 5 6 7) (240 110 110)
        simpleGrading (1 1 1)
    );

    edges ( );

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

Resulting mesh can be seen in figure 3.1, it contains 2,904,000 elements.

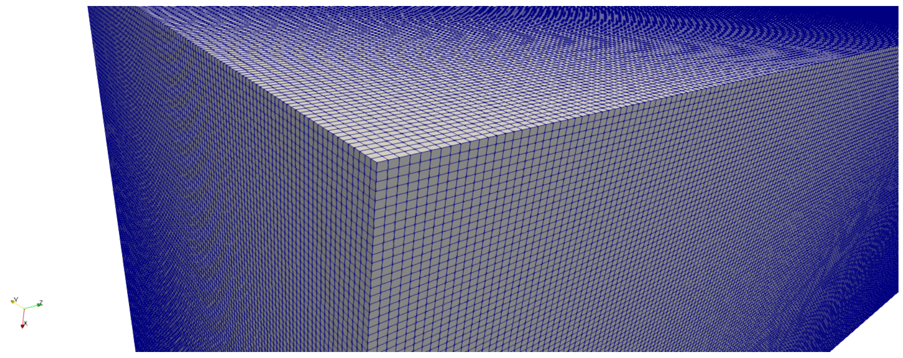

Fig. 3.1: *blockMesh output*

surfaceFeatureExtract is the command for conversion of .stl files to .eMesh files. These files are needed for *snappyHexMesh* to recognize the surfaces that we are going to use.

snappyHexMeshDict is the dictionary to control *snappyHexMesh* settings. This dictionary has three main sections.

- *castellatedMesh*

This part of the *snappyHexMesh* cuts the control volume from blockMesh created in the first section. First, geometry should be defined as;

```
geometry
{
    inlet.stl {type triSurfaceMesh; name inlet;}
    outlet.stl {type triSurfaceMesh; name outlet;}
    wall.stl {type triSurfaceMesh; name wall;}
    pin.stl {type triSurfaceMesh; name pin;}

    refinementCylinder
    {
        type searchableCylinder;
        point1 (0 0 1.16);
        point2 (0 0 1.7);
        radius 0.2;
    }

    refinementSphere
    {
        type searchableSphere;
        centre (0 0 1.24);
        radius 0.08;
    }
};
```

Secondly, *castellatedMeshControls* are defined. In this section, .eMesh are used to cut the mesh. This command has a tool to refine features as well as surfaces and volumes. Refinement is done relatively, independent of the fineness of the mesh. For this project, flow is in-viscid so there is no boundary layer at walls, yet shock is forming at the pin surface so pin is refined with one level. Also, to capture the shock, a refinement cylinder is defined around the edge of the pin. *castellatedMeshControls* used in this project;

```
castellatedMeshControls
{
    maxLocalCells  1000000;
    maxGlobalCells 4000000;
    minRefinementCells 2;
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 1;

    features
    (
        {file "inlet.eMesh"; level 0;}
        {file "outlet.eMesh"; level 1;}
        {file "pin.eMesh"; level 1;}
        {file "wall.eMesh"; level 0;}
    );

    refinementSurfaces
    {
        inlet  {level (0 0);}
        outlet {level (1 1);}
        pin    {level (2 2);}
        wall   {level (0 0);}
    }

    resolveFeatureAngle 20;
    refinementRegions
    {

        refinementCylinder {mode inside; levels ((1.0 1));}
        refinementSphere {mode inside; levels ((1.0 2));}
    }

    locationInMesh (0.0 0.0 1);
    allowFreeStandingZoneFaces true;
}
```

Control volume after creating castellatedMesh has 3,263,732 elements and can be seen in figure 3.2;

- *snap*

Resulting mesh from castellatedMesh doesn't conform spherical surfaces, because of the shape of the blocks. snap feature creates tetrahedral elements to conform surfaces.

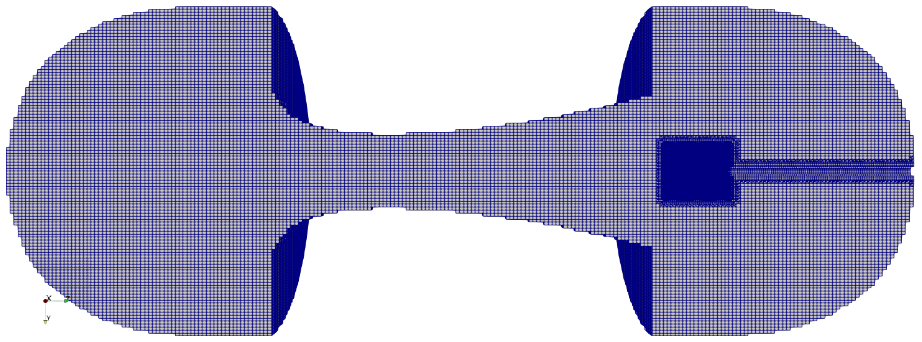

Fig. 3.2: castellatedMesh

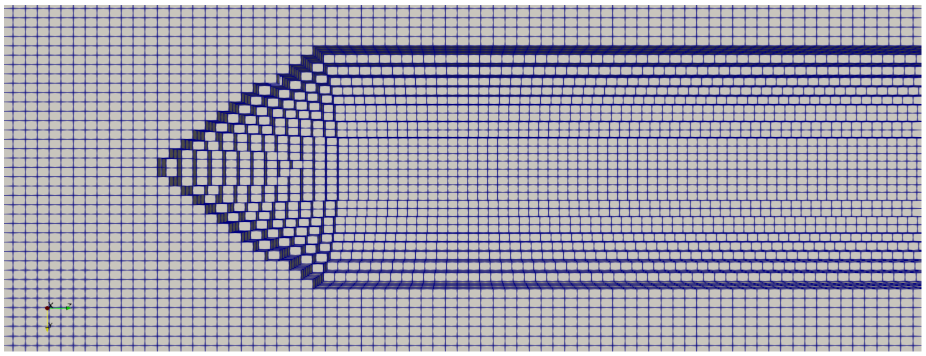

Fig. 3.3: castellatedMesh around bluff-body

Settings for the snapping in the analysis;

```
snapControls
{
    nSmoothPatch 3;
    tolerance 4.0;
    nSolveIter 30;
    nRelaxIter 5;
    nFeatureSnapIter 15;

    implicitFeatureSnap false;
    explicitFeatureSnap true;
    multiRegionFeatureSnap false;

}
```

Impact of snapping to the mesh appears in figure 3.4;

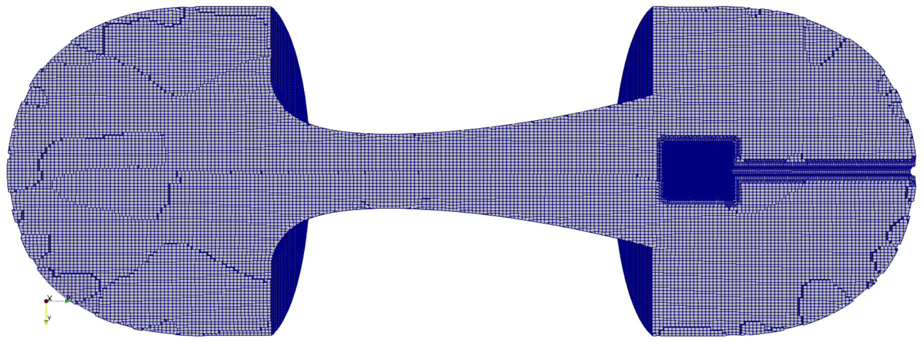

Fig. 3.4: Snapped surfaces

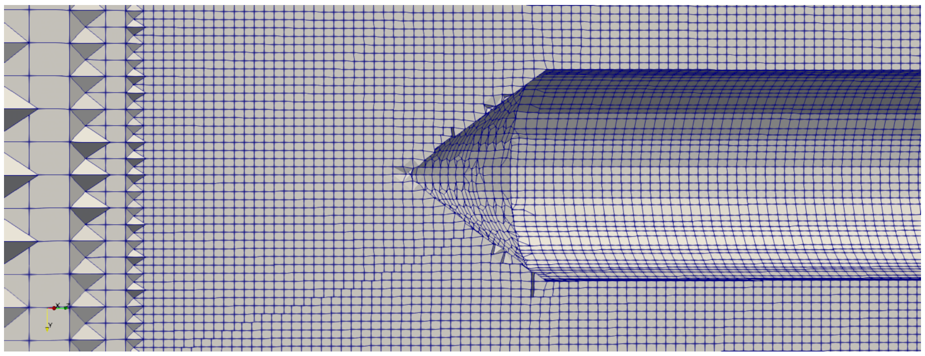

Fig. 3.5: Snapped surfaces

- *addLayers*

This feature creates viscous layers. Although our analysis is in-viscid, poor elements can be detached from surfaces to the internal sections of the mesh. This creates a buffer field between quality sensitive areas, giving less change to diverge from solution. Adding layer can be defined with four parameters, yet only two of them is enough to fully define layer addition. In analysis, final layer thickness and expansion ratio are used for definition.

Controls for layer addition;

```
addLayersControls
{
    relativeSizes false;

    layers
    {
    pin{nSurfaceLayers 3;}
    outlet{nSurfaceLayers 3;}
    }

    expansionRatio 1.2;
    finalLayerThickness 0.002;
    minThickness 0.001;
    nGrow 0;


    featureAngle 90;

    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;

    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 130;

    nBufferCellsNoExtrude 0;
    nLayerIter 50;
    nRelaxIter 4;

}
```

snappyHexMesh creates the mesh with iterative methods. It also checks the mesh in terms of quality in every step of the meshing procedure. So, *meshQualityControls* should be defined for reference.

General quality parameters are used in analysis;

```
meshQualityControls
{

    maxNonOrtho 65;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minFlatness 0.5;
    minVol 1e-13;
    minTetQuality 1e-9;
    minArea -1;
    minTwist 0.02;
    minDeterminant 0.001;
    minFaceWeight 0.02;
    minVolRatio 0.01;
    minTriangleTwist -1;

    nSmoothScale 4;
    errorReduction 0.75;

}
```

Resulting mesh after following these steps can be seen in figure 3.6, it has 789,408 elements and conforms mesh quality criteria. Overall, this mesh worked in analysis without a problem.

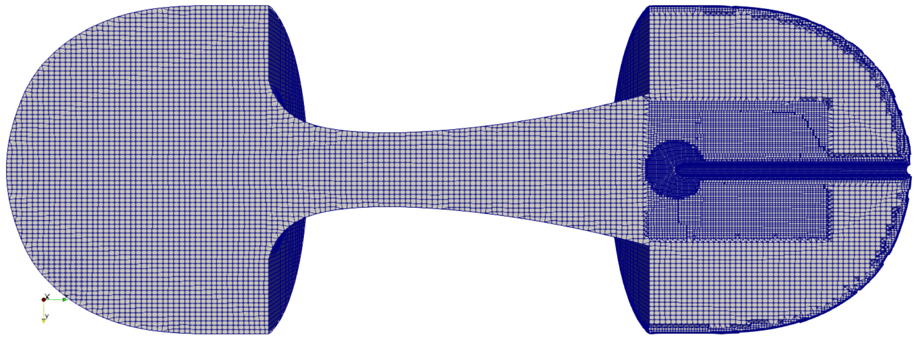

Fig. 3.6: Mesh used in analysis.

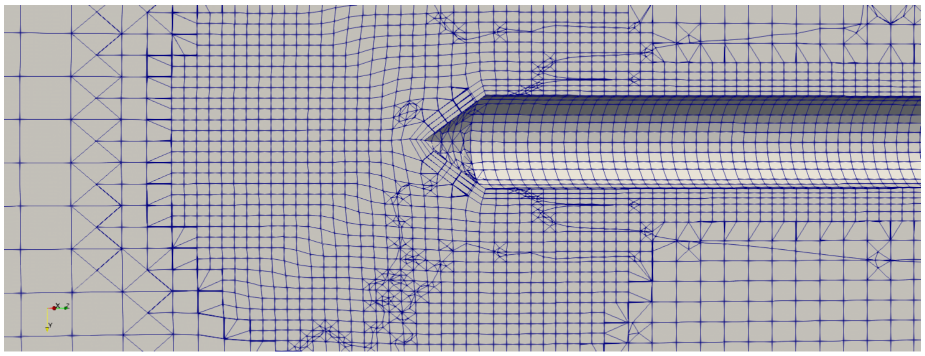

Fig. 3.7: Refinement for shock capturing.

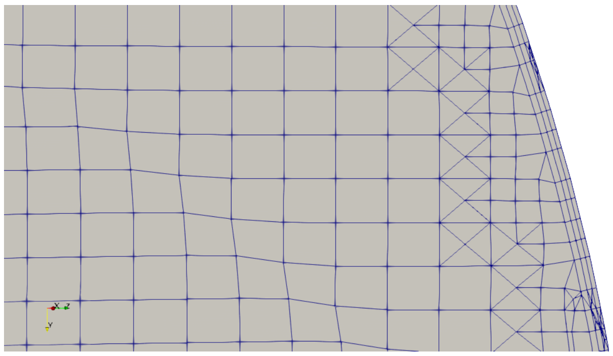

Fig. 3.8: Boundary layer for outlet.

4 Governing Equations

The boundary conditions are adiabatic, flow inside the nozzle can be assumed as isentropic, since flow is in-viscid. So isentropic relations can be used in this problem. Also, flow is chocked at the throat as a design decision, so;

$$Ma_t \quad = \quad 1; \tag{4.1}$$

Equation 4.1 Mach number at the throat.

Speed of the gas at the exit of the nozzle could be found using ares of the throat and the exit. Diameters of the throat and exit sections can be measured with the ruler tool in ParaView;

So, the areas can be calculated as;

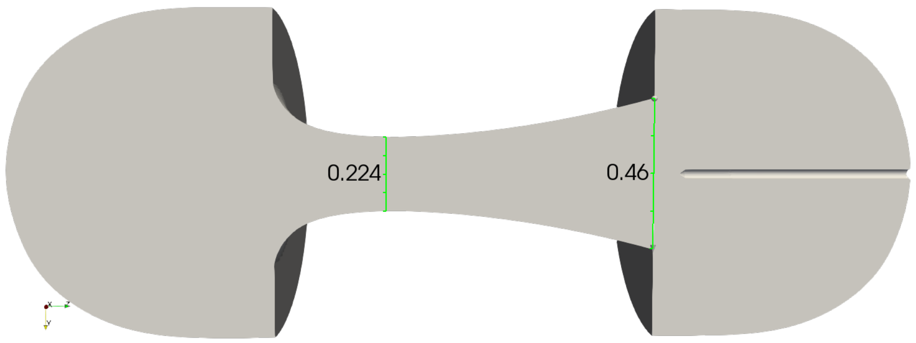

Fig. 4.1: Diameters of throat and exit section

$$A_t = \frac{\pi d^2}{4} = \frac{\pi \cdot 0.224^2}{4} = 0.0394\, m^2 \tag{4.2}$$

$$A_e = \frac{\pi d^2}{4} = \frac{\pi \cdot 0.46^2}{4} = 0.1662\, m^2 \tag{4.3}$$

This condition results with maximum flow rate and can be calculated as;

$$\dot{m_{max}} = \frac{P_0 A^*}{\sqrt{T_0}} \sqrt{\frac{k}{R}\left(\frac{2}{k+1}\right)^{\frac{k+1}{k-1}}}$$
$$\dot{m_{max}} = \frac{3 \cdot 10^6 \cdot 0.04}{\sqrt{773}} \left(\frac{1.4}{283}\left(\frac{2}{1.4+1}\right)^{\frac{1.4+1}{1.4-1}}\right)$$
$$\dot{m_{max}} = 7\, ^{\mathrm{kg}}/_{\mathrm{s}}$$

Using equations 4.2 and 4.3 area ratio can be calculated as;

$$\frac{A_e}{A_t} = 4.217$$

So, ratios of Mach numbers, pressures and temperatures can be found from isentropic flow tables [1];

$$\frac{Ma_e}{Ma_t} = 3 \tag{4.4}$$

Exit temperature and pressure can be calculated using isentropic relations between reservoir and exit plane;

$$\frac{T_0}{T_e} = 1 + \frac{k-1}{2} Ma^2$$
$$\frac{P_0}{P_e} = (1 + \frac{k-1}{2} Ma^2)^{\frac{k}{k-1}}$$

$$T_e = 276\,K \tag{4.5}$$
$$P_e = 81671\,Pa \tag{4.6}$$

Yet, outlet pressure is;

$$P_b = 5000\,Pa \tag{4.7}$$

To examine oblique shocks lets examine 4.2. $\theta_{max}$for $Ma = 3$ is around 35 degrees. For this case 2θ is equal to 70s degrees, so it is not clear that an oblique shock is going to be attached or detached.

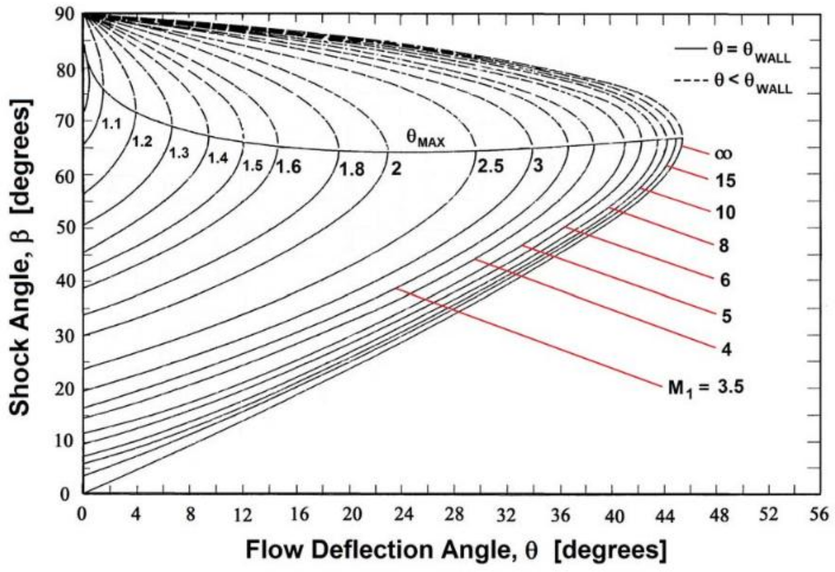

Fig. 4.2: Oblique shock waves.[2]

5 Numerical Approach

As mentioned earlier, this problem is suitable to solve with *OpenFoam.* Flow is compressible, in-viscid and laminar. It is a steady-state problem, yet it can be solved with transient solvers.

rhoCentralFoam is a compressible flow solver with genuine shock capturing properties.

Initial conditions are defined as;

p

```
    dimensions      [1 -1 -2 0 0 0 0];
    internalField   uniform 300000;
    boundaryField
    {
        inlet
        {
            type          totalPressure;
            p0 uniform 300000;
        }

        outlet
        {
            type            waveTransmissive;
            value           uniform 5000;
            field           p;
            psi             thermo:psi;
            fieldInf        5000;
            gamma           1.4;
            lInf            6;
            value           uniform 5000;
        }

        pin
        {
            type            slip;
        }

        wall
        {
            type            slip;
        }

    }
```

T

```
dimensions      [0 0 0 1 0 0 0];
internalField   uniform 773;
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 773;
    }

    outlet
    {
        type            zeroGradient;
    }

    pin
    {
        type            zeroGradient;
    }

    wall
    {
        type            zeroGradient;
    }
}
```

U

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            zeroGradient;
    }

    pin
    {
        type            slip;
    }

    wall
    {
        type            slip;
    }
}
```

Thermodynamic properties for air in this project defined as;

```
thermoType
{
    type hePsiThermo;
    mixture pureMixture;
    transport const;
    thermo hConst;
    equationOfState perfectGas;
    specie specie;
    energy sensibleInternalEnergy;
}
mixture
{
    specie
    {
       molWeight 28.96;
    }
    thermodynamics
    {
       Cp 1004.5;
       Hf 0;
    }
    transport
    {
       Pr 1;
       mu 0;
    }
}
```

This properties are used with assumptions that air is perfect gas, flow is in-viscid and transport properties are constant during analysis.

Solver equation and scheme dictionaries should be located in *system* directory.

- Numerical schemes are defined in *fvSchemes*,
- Solution and algorithm controls are defined in *fvSolution*,
- Time and data input/output controls are defined in *controlDict*.

Dictionaries used in this simulation;

fvSolution

```
solvers
{
    "(rho|rhoU|rhoE)"
    {
        solver diagonal;
    }
    U
    {
        solver smoothSolver;
        smoother GaussSeidel;
        nSweeps 2;
        tolerance 1e-10;
        relTol 0;
    }
    e
    {
        $U;
        tolerance 1e-10;
        relTol 0;
    }
}
relaxationFactors
{
    fields
    {
        p 0.3;
    }
    equations
    {
        U 0.7;
        e 0.7;
    }
}
```

fvSchemes

```
fluxScheme      Kurganov;
ddtSchemes
{
    default         Euler;
}
gradSchemes
{
    default         Gauss linear;
}
divSchemes
{
    default         none;
    div(tauMC)      Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear corrected;
}
interpolationSchemes
{
    default          linear;
    reconstruct(rho) vanAlbada;
    reconstruct(U)   vanAlbadaV;
    reconstruct(T)   vanAlbada;
}
snGradSchemes
{
    default         corrected;
}
```

controlDict

```
application     rhoCentralFoam;
startFrom       latestTime;
latestTime      0;
stopAt          endTime;
endTime         0.05;
deltaT          1e-7;
writeControl    adjustableRunTime;
writeInterval   1e-4;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression on;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
adjustTimeStep  true;
maxCo           0.2;
maxDeltaT       1e-3;
fileHandler collated;
residualControl
{

    e   1e-9;
    U   1e-9;

}
functions
{

    #includeFunc  residuals
    #includeFunc  MachNo
    #includeFunc  singleGraph

}
```

6 Results and Discussion

Expected results are achieved after trying more than thirty cases.

6.1 Failed cases

Boundary conditions and the mesh is important for good results, impact of these two factors are inspected during project.

Boundary conditions

fixedValue boundary condition showed poor performance at both inlet and outlet boundaries, as well as both in pressure and speed. This boundary condition tries to keep the value at a fixed point at every iteration, yet it seems better to give some tolerance during analysis. This boundary condition only worked in inlet temperature.

fluxCorrectedVelocity boundary condition at outlet is tried for velocity outlet. This condition doesn't allow the reflection of the waves from boundary. Even so after many tries, zeroGradient leads better results for velocity in both inlet and outlet.

noSlip ; even the flow is inviscid, this boundary conditions performed worst than *slip* boundary condition.

internalValue should be defined same with inlet, so that flow starts from outlet. If it is defined same as outlet, high pressure at the inlet creates a shock, which collapses at the center of the sphere shaped inlet. This situation is hard to solve, instead it is easier to solve when there is low pressure after shock.

linf value in waveTransmissive should be big enough to block wave reflection. However, as this value increases, time for outlet surface to converge desired value ascends. To compare,

- linf 0.025 meters(figure 6.1); too short, reflected waves,
- linf 1 meters; outlet converges in 0.008 seconds,
- linf 6 meters; outlet converges in 0.028 seconds.

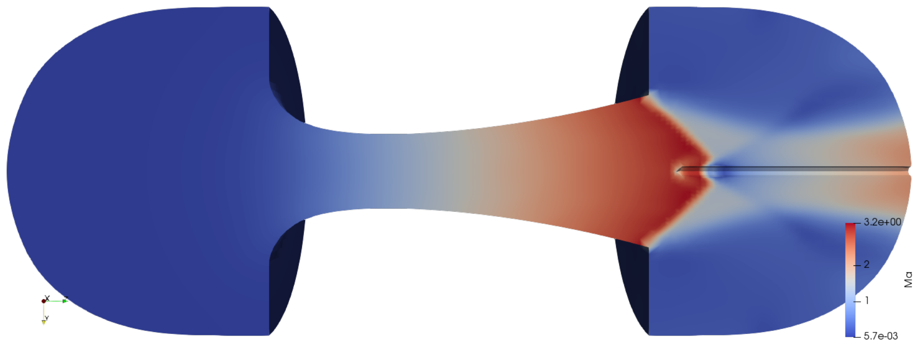

Fig. 6.1: Outlet at 35000 Pa, shock moves constantly in the z direction. After some time, it resonates with reflected waves and simulation diverges.

Mesh

Mesh size is one of the most important concerns of the compressible flow. Even if flow is formed, shock might not be captured. As it can seen in figure 6.2, mesh is so course that Mach number can't reach the design point, see figure 6.3. Although it is clear that it is going to form if the mesh is finer, white and blue sections are separated with a line (figure 6.4).

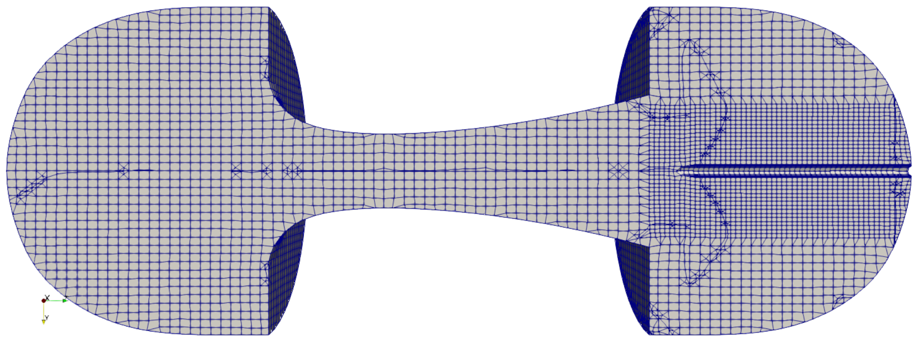

Fig. 6.2: Course mesh

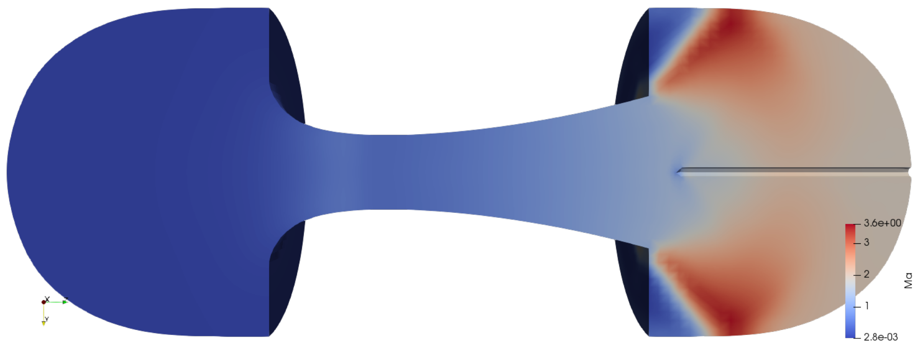

Fig. 6.3: Mach number in course mesh

Boundary layer addition can increase the quality at the critical surfaces. Still, it is important to keep an eye on the quality parameters. As can be seen in figure 6.5, layers are collapsing at the tip of the cone. This mesh fails at 10 of the mesh quality cafeterias.

Fine mesh doesn't always results in good results. Some ill cells can diverge, even if they are small. For example, mesh in figure 6.6 has 3,340,197 elements, but ill elements diverges the solution.

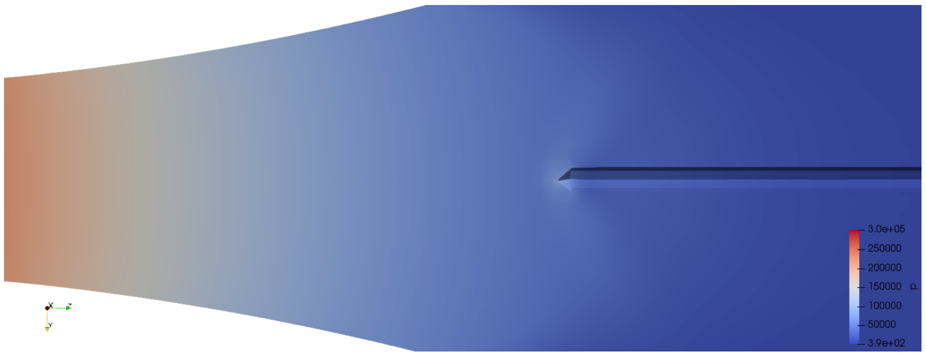

Fig. 6.4: Pressure in course mesh

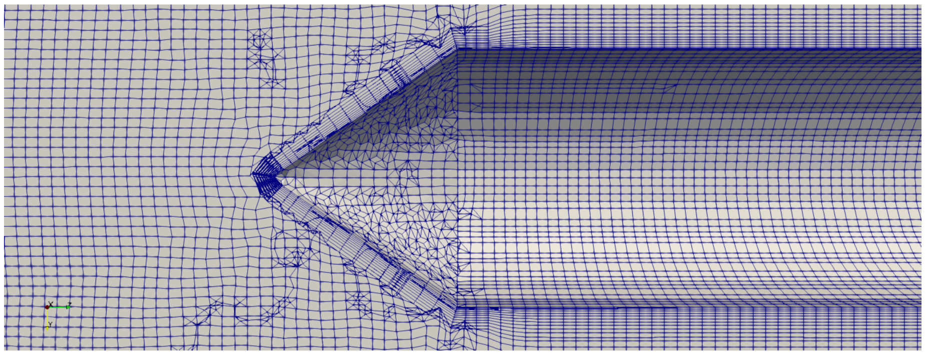

Fig. 6.5: Boundary layer at the cone of the bluff-body. 10 layers are added with 0.32 mm first layer size and expansion ratio of 1.1.

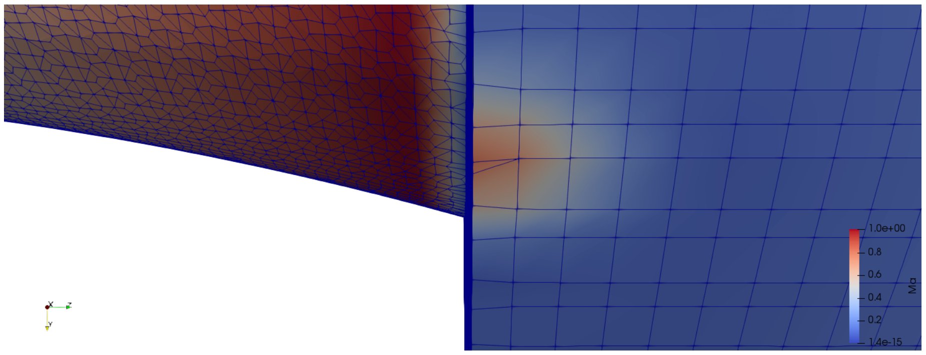

Fig. 6.6: Ill cells are creating discontinuity in the field.

6.2 Stable cases with shock formation

Required boundary conditions are achieved with ... elements. Scripts used in analysis are given in sections 3 and 5.

Mesh in this analysis can be seen in figures 3.6,3.7 and 3.8.

In this analysis, Courant number is kept stable at 0.2 which corresponds to time step $2 \cdot 10^{-7}\, s$. One time step takes 2 seconds to calculate in a computer with 8 GB RAM, 8 core processor at 3.2 GHz. Analysis came to a stable state in 0.032 s, so it took 90 hours to complete. Corresponding results can be seen in 6.7, 6.8 and 6.9.

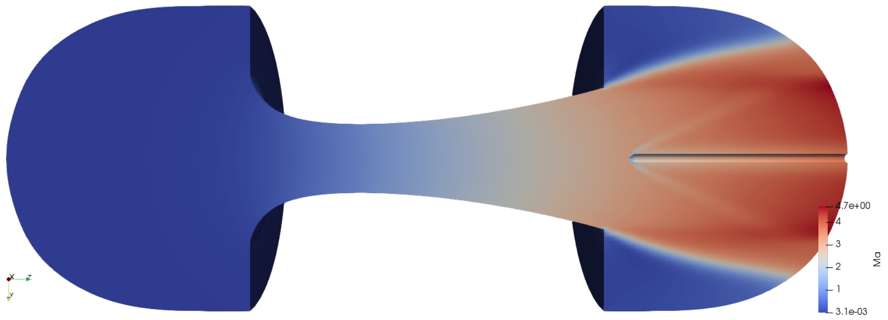

Fig. 6.7: Mach number at 0.0327 s.

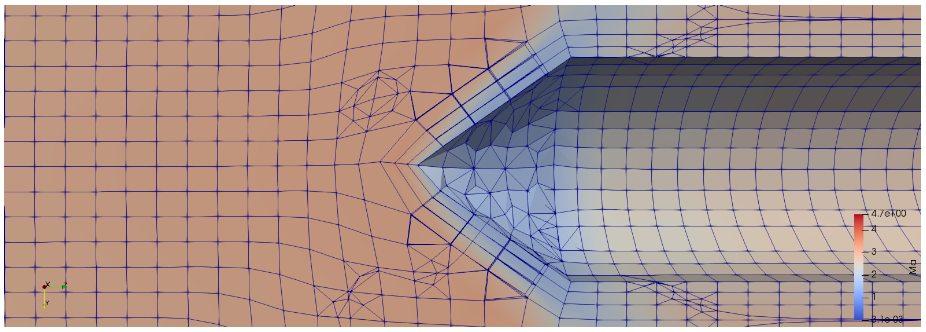

Fig. 6.8: Mesh and Mach number around pin at 0.0123 s.

Later, the refinement parameter for sphere is increased to 5, it can be found at 7. Corresponding sphere can be seen in figures 6.10, 6.11 and 6.12. In this mesh, the sphere has more 5 million elements in it. We can calculate this using the element number of the same mesh, but with level two refinement, which has only 700,000 elements. This simulation is computed with $2 \cdot 10^{-7}\, s$ for $0.0123\, s$ at UHeM with 4000 CPU hours.

Results for this analysis can be found at figures 6.13, 6.14, and 6.15.

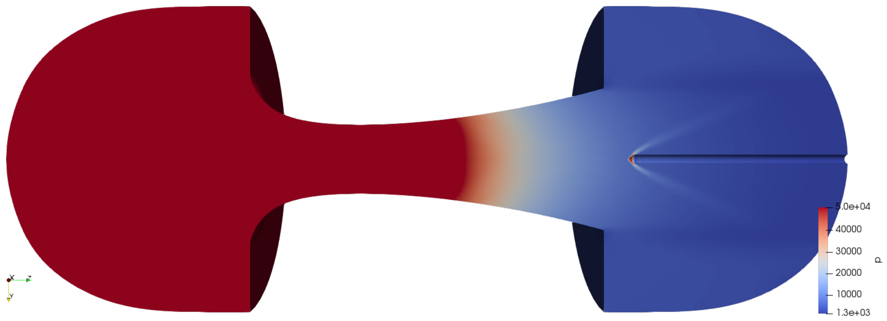

Fig. 6.9: Pressure at 0.0123 s.

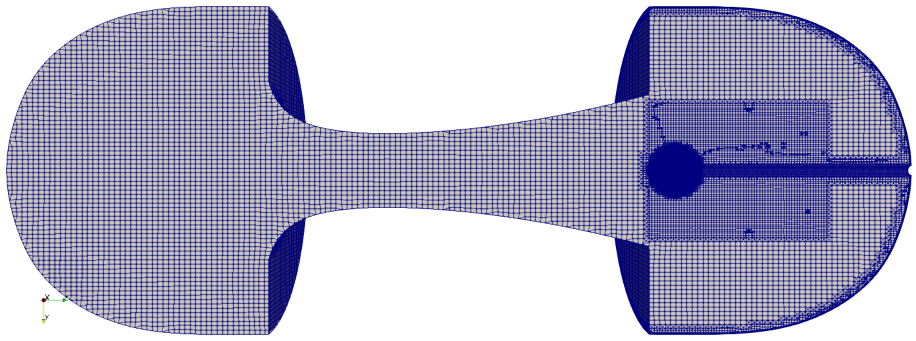

Fig. 6.10: 5 level refinement in sphere at the wedge. Corresponding mesh has 5.7 million elements. More than 5 million of them is inside the sphere

7 Conclusion

Region around the tip of the pin is refined with a sphere to increase the chance of shock capturing, yet only detached shock is clearly visible. Expected detached shock is not as much as apparent because of course mesh. Yet, computing capabilities, especially RAM size of the author's computer is not enough to solve a mesh with more than 1.2 million elements. Also, one more computer with sufficient RAM is tried to solve the problem, still with 3 million elements, mesh becomes excessively fine in the pin area that time step should be at $1 \cdot 10^{-8}\,s$. In this scenario, lets say the flow is going to be stable in 0.015 s, if the duration of calculation for 1 time step is 1 second, it takes 18 days to complete the analysis with the computer mentioned above. Later, using resources from UHeM, analysis is solved using 4000 CPU hours, which is 1000 more than allowed limit for undergraduate students. It should be noted that, if the ratio of the largest and smallest elements is so much, the solution tends to diverge in ill elements more quickly. Hence, refinement levels of more than 2 couldn't be worked properly in this study. Yet, a sphere with five level refinement is solved in UHeM.

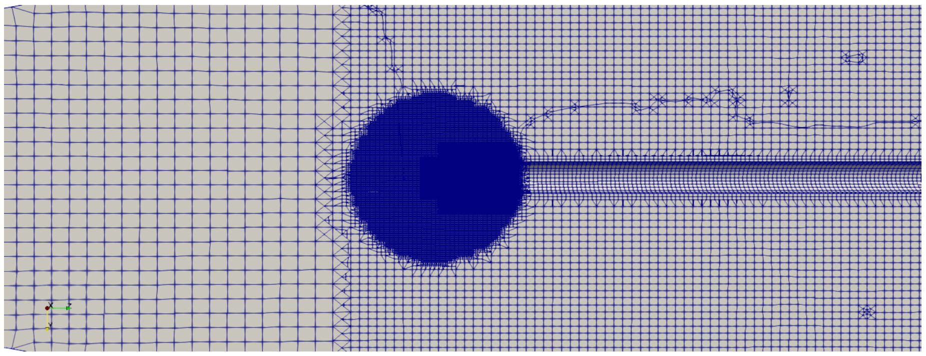

Fig. 6.11: Sphere around pin.

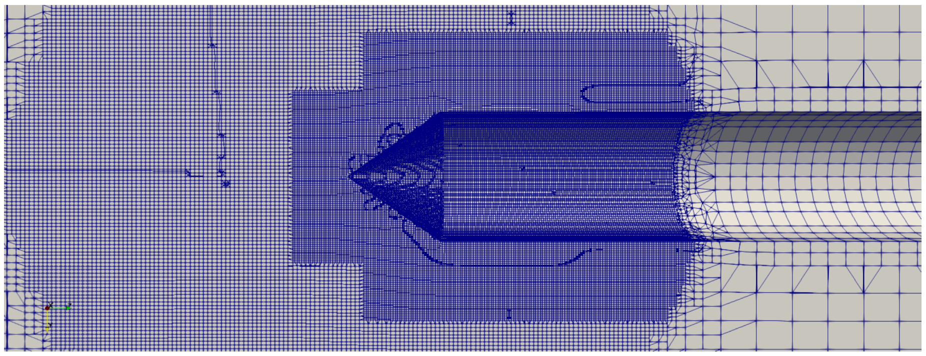

Fig. 6.12: Refinement command for pin geometry is creating cylindrical refinement regions.

It is clear to say that the wedge angle is almost equal to the maximum θ angle. Also, considering the error percentage in the CFD analysis, flow might not reach the design speed, so shock will stay attached. A wedge with θ angle larger than provided geometry might give better results to investigate detached shocks.

References

[1] Equations, tables, and charts for compressible flow, January 01, 1953 , NACA-TR-1135, Reviewed from: https://ntrs.nasa.gov/search.jsp?R=19930091059

[2] An Internet Book on Fluid Dynamics,
Reviewed from: http://brennen.caltech.edu/fluidbook/basicfluiddynamics/Compressibleflow/obliqueshock.

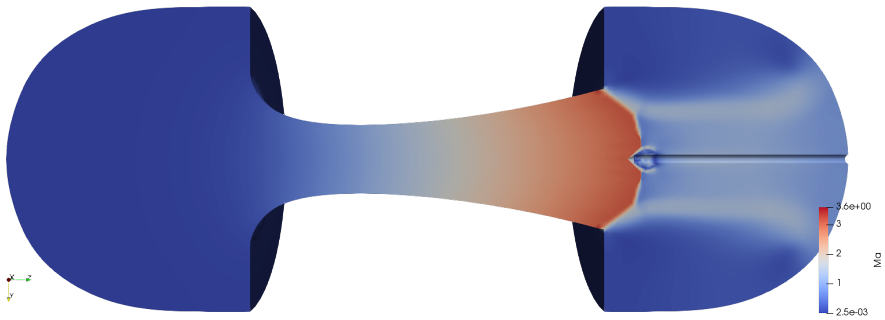

Fig. 6.13: Mach number at 0.0123 s.

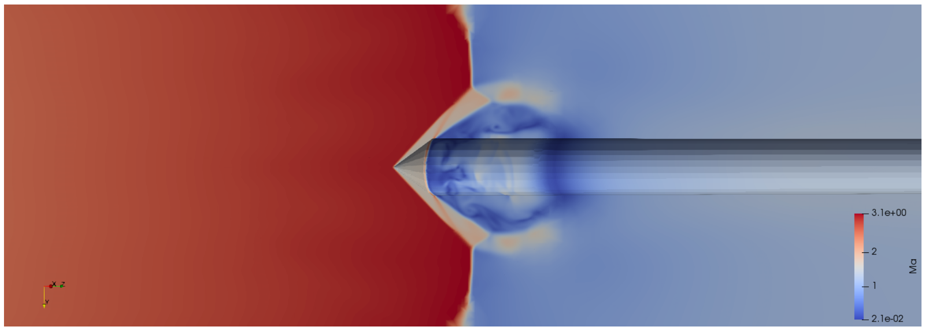

Fig. 6.14: Mach number around pin. Change in resolution is clearly changing with the size of mesh. Also, secondary flows are formed after attached shocks.

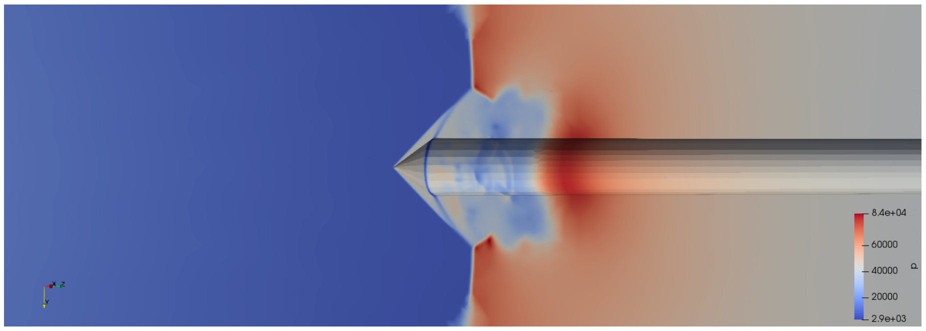

Fig. 6.15: Pressure around pin at 0.0123 s.