MARMARA UNIVERSITY

FACULTY OF ENGINEERING

INVESTIGATION OF MESH GENERATION STRATEGIES FOR FLOW ANALYSIS OF INTAKE AND EXHAUST PORT

MEHMET CİVAN HOŞGÖR, ÖMER TOP

GRADUATION PROJECT REPORT

Department of Mechanical Engineering

Supervisor

Prof. Dr. Mehmet Zafer Gül

ISTANBUL, 2020

MARMARA UNIVERSITY

FACULTY OF ENGINEERING

Investigation of Mesh Generation Strategies for Flow Analysis of Intake and Exhaust Port

Mehmet Civan HOŞGÖR, Ömer TOP

(150415038) (150415044)

2020, Istanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

The author(s) hereby grant(s) to Marmara University permission to reproduce and to distribute publicly paper and electronic copies of this document in whole or in part and declare that the prepared document does not in anyway include copying of previous work on the subject or the use of ideas, concepts, words, or structures regarding the subject without appropriate acknowledgement of the source material.

Signature of Author(s) .....
Department of Mechanical Engineering

Certified By .....
Project Supervisor, Department of Mechanical Engineering

Accepted By .....
Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor Prof.Dr. Mehmet Zafer Gül, for the valuable guidance and advice on preparing this thesis and giving me moral and material support.

July, 2020 Mehmet Civan Hoşgör , Ömer Top

CONTENTS

ÖZET

Emme ve Egzoz Portundaki Akış Analizi için Mesh Üretim Stratejilerinin Araştırılması

Hesaplamalı Akışkanlar Dinamiği'nde mesh üretimi, geometrinin karmaşıklığına da bağlı olaraktüm sürecin ana zorlayıcı kısmı olarak bilinir. Mesh kalitesi, mesh çözücüde sürekliliğin uzaksaması ve hiçbir çözüm elde edilememesinden, çözümün doğruluğuna kadar süreç üzerinde büyük etkisi olan önemli bir konudur; sonuçları elde etmek için yineleme sayısı gibi hesaplama işlerinin seviyesini de son derece etkileyebilirler. Aynı zamanda iterasyon sayısı gibi hesap gerektiren işlerin seviyesini belirlemede son derece önemli etkisi vardır. Bu projede dört yüzlü, dört yüzlünün baskın olduğu, hibrit, altı yüzlünün baskın olduğu ve altı yüzlü mesh türleri oluşturularak emme ve egzoz portları için en iyi mesh konfigürasyonuna ulaşmaya çalıştık. Bu süreçte karşılaştırmalı olarak Ansys, Salome ve OpenFOAM gibi farklı konularda farklı özelliklere sahip programlar kullandık.

ABSTRACT

Investigation of Mesh Generation Strategies for Flow Analysis of Intake and Exhaust Port

In Computational Fluid Dynamics (CFD), mesh generation known as the main challenging part of the whole process depending on the complexity of the geometry. Mesh quality is such important issue that has a huge impact on the accuracy of the solution, even to the point where the solver diverges and no solution is obtained; they can also extremely affect the level of computational work like number of iterations to achieve the results. In this project, we have tried to reach the best mesh configuration for intake and exhaust ports by generating various types of mesh such as tetrahedral, quad-dominated, hybrid and hex-dominated. We have used three meshing programs by their meshing and geometry clean-up capabilities.

SYMBOLS

A : Area

B : Bore diameter

c : Continuity

D : Diameter

Dh : Hydrolic Diameter

k : Turbulent Kinetic Energy

N : Revolution Per Second

Re : Reynolds Number

S : Stroke diameter

Up : Mean Piston Velocity

V :Velocity

$\mathbf{V_d}$: Volume of One Cylinder

$\mathbf{V_{dot}}$: Volumetric Flow Rate

$\boldsymbol{\theta_{max}}$: Largest Angle In A Face Or Cell

$\boldsymbol{\theta_{min}}$: Smallest angle in a face or cell

$\boldsymbol{\theta_e}$: Angle For Equi-Angular Face Or Cell

ρ : Density

μ : Dynamic Viscosity

ε : Rate Of Dissipation Of Turbulent Kinetic Energy

: Number

ABREVIATIONS

1D : One dimensional

2D : Two dimensional

3D : Three dimensional

CFD : Computational Fluid Dynamics

BDC : Bottom dead center

CPU : Central Process Unit

V16 : a sixteen-cylinder piston engine where the cylinders share a common crankshaft and are arranged in a V configuration.

CAD : Computer Aided Design

CAE : Computer Aided Engineering

AF : Air Fuel Ratio

RPM : Revulation Per Minute

L : Litre

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

In the world of continuously developing technologies and new inventions, it gained a huge importance to simulate and analyse new products and put it into diverse tests before realizing it. This process provides to see whether the project is feasible or not, to detect errors, mistaken parts of the product and to fix these mistakes earlier. The growing importance of 2-D and 3-D simulation has made mesh generation a key to arrive accurate and fast solutions.

Meshing is a division process of a geometric domain into numerous small elements by creating interconnected nodes. There are lots of mesh generation techniques .We can list the basic ones; the triangle and the quadrilateral in 2-D and tetrahedron, quadrilateral pyramid, triangular prism, and hexahedron in 3-D. Also, different meshing techniques will be mentioned later.

Meshing finds use many applications in different areas like rendering, physical simulation, flow analysis, static analysis etc. Application area of meshing can assist in determining the type of mesh. For example, in structural and fatigue analysis, it is preferred quad and hex elements over trias, tetras and pentas. In mold flow analysis, triangular elements are preferred over quadrilateral ones. In this study, fluid flow analysis will be considered as application area.

In this project, the development of mesh generation methods in exhaust and intake port of an V16 engine is studied in various meshing softwares. ANSYS meshing, SALOME, and OpenFOAM are going to be used to form suitable meshes for the geometry but it can be found more meshing programs in the market. Every meshing programs have own advantages and disadvantages. Some of them requires payment but offer the user more option to carrying out meshing and have more capabilities. Some have less capabilities, even don’t have an interface which forces the user to writing codes but they are not commercials. After searching all pros and cons of meshing programs, user can decide which program will be used.

There is also, however, a further point to be considered. It is the cleaning geometry process. When dealing with meshing, many geometric data file types are encountered. A program may not import the data which is exported from another one. On the other hand, some programs come forward in cleaning geometry by offering user more alternative. By considering these issues, we have considered that it is better to use ANSYS for repairing

process because first of all, ANSYS hosts useful repair and prepare options. Secondly it meets capability of exporting and importing for many different geometric data files (.scdoc, .step, .iges, .. etc) which facilitates dealing with converging file types. However, SALOME and OpenFOAM can't reach the capability of ANSYS at this point.

2.LITERATURE SURWEY

2.1 An Algorithm for Three-Dimensional Mesh Generation for Arbitrary Regions with Cracks [1]

In this article, an algorithm has been developed for generating unstructured tetrahedral meshes for arbitrarily shaped three-dimensional region. The algorithm is designed to fix the meshes and produce better meshes in region with cracks. However, it also works for regions without cracks. The algorithm uses the advancing front technique which is already based on a standard procedure found in the literature but in this project, this process goes even further and two additional steps are added. The algorithm has four specific requirements.

Firstly, the algorithm should produce well shaped elements, avoiding elements with poor aspect ratios. The second requirement is that the algorithm generates a mesh that conforms to an existing triangular mesh on the boundary of the region. This is important in the context of crack growth simulation, because it allows remeshing to occur locally in a region near a growing crack. The third requirement of the algorithm is that it has the ability to transition well between regions with elements of highly varying size. Finally, the fourth requirement is the specific modelling capability for modelling cracks.

There are many algorithms that can work similarly for the same purpose. At this point, the algorithm has advantages on others. For example, other algorithms can mesh properly when all generated elements have similar characteristic size which is obviously undesired when considering crack issue. However, the current algorithm has been designed to provide better transition capabilities. Another point that merges the difference between the current algorithm and the previous ones, is that the current algorithm also uses an octree, but only as a node-spacing function. This approach tends to have a better control over the quality of the generated mesh and, apparently, decreases the amount of cleaning-up procedures.

The algorithm is organised in the following phases:

I. Octree generation: at first an octree is generated including cracks to control the distribution of nodes. Then, refinements are applied in the interior of the region. Finally, a natural transition between regions of different degrees of refinement is obtained.

II. Advancing front procedure: the procedure starts with a surface limiting a boundary volume element is 'removed' or 'separated' from the region one by one. As each element is removed, the boundary surface is refreshed and the process is repeated until whole region is meshed or one or more internal unmeshed cavities remain from which valid elements cannot be removed. Advancing front procedure presents us two mesh generation technique. One of them, is the geometry-based element generations which produce well shaped elements. The other one is topology-based element generation in any node close to the current base face is selected. The node that forms the best solid angle with the base face is chosen for the generation of the new tetrahedron.

III. Local mesh improvement: In the advancing front technique, poorly shaped tetrahedral mesh elements might be generated which can be provided to avoid in this section by modification procedures. Two technique is described to improve the mesh quality. The first is a conventional nodal relocation smoothing technique which is based on node coordinates averaging. The second is a back-tracking procedure which deletes faces of poorly shaped elements to allow creating regions having better meshs.

At the end, a number of mesh detailed examples were presented to demonstrate how much good the distribution of the generated elements are. It was shown that only a small percentage (0.49–0.90%) of the total generated elements are poorly shaped. These poorly shaped elements are usually located in the interior of the model, away from the crack front and stress concentration regions where better elements are desired.

Two examples were presented to determine empirical measures of the algorithm performance. It was shown that the algorithm has the expected performance for practical applications, although auxiliary search structures, such as trees, could be used in order to improve the algorithm's performance.

2.2 CFD Simulations of Intake Port Flow Using Automatic Mesh Generation [2]

In this article, an automatic hexahedral meshing technique has been studied to predict the steady flow inside intake port and valve geometries. Two different combinations of intake port are available to investigate. One of them consists of one helical and one directed port which is asymmetric like our ports in the project. The other one is symmetric design consisting of two directed port designs.

By Perkins Technology, an automatic meshing procedure is developed that can reduce the needed time to generate meshes of complex geometries such as intake port designed asymmetrically.

As everyone who is interested in the computational mesh generation knows, the most challenging part of this job is how much time the meshing process takes. The process can take typically four to six weeks for practical intake port / valve geometries. For making CFD (Computational Fluid Dynamics) most of the engineer's routine, lead time mesh construction of complex geometries has to be dramatically reduced. For this purpose, this article provides us two comparative ways – an automatic process for CFD mesh consisting of entirely hexahedral cells and non-hexahedral cells that are easier to create like tetrahedra.

A methodology is achieved by starting with a solid model representation of the geometry. Firstly, the geometry can be smoothly meshed with tetrahedral elements. Then, the recent mesh that we have constructed is converted into hexahedral meshes by dividing each tetrahedral element into four hexahedral elements. As a result of that, it emerges that a procedure for an arbitrary geometry consisting of 100% hexahedral elements in less than 4 hours. The lead time for mesh generation becomes the time required to construct a solid model of the geometry, typically one to three days.

The article is offering a statistic (geometry A is the asymmetric intake port geometry and geometry B is the symmetric intake port geometry):

Process	Time (hours)	
	Geometry A	Geometry B
Creating solid model of geometry	22	10
Preparing solid model for automatic meshing with tetrahedra	3.25	3.25
Generating tetrahedral meshing of geometry	0.5	0.5
Converting tetrahedral mesh into hexahedral mesh	0.25	0.25

Table 1.Constructing a Hexahedral CFD Mesh of an Arbitrary Geometry. The total time taken to prepare meshes for intake port configurations (A) and (B) was 26 and 14 hours, respectively.

On the other hand, it is also observed that the effect of mesh density in this article so a new computational mesh was generated for port configuration to investigate the effect of mesh density upon the predicted flow structure. The new mesh contained 427.580 cells with other words it doubles the number of mesh according to previous mesh. There was no significant difference for the predicted turbulence levels of both meshes. To sum up, a finer mesh does capture the rotational symmetry of the flow better than a coarse mesh and at least 400,000 cells are guaranteed for symmetric port configurations.

In conclusion, it is succeeded to reduce the required time less than 15% to generate mesh configuration for complex geometries like intake port / valve by current meshing techniques.

2.3. Automatic Mesh Generation for Full-Cycle CFD Modeling of IC engines: Applications to the TCC Test Case [17]

In this paper, a novel approach for automatic generation of engine grids was developed using the OpenFOAM technology and implemented into the LibICE code. Such technique was then incorporated in the methodology developed by the authors over the years for full-cycle engine simulations, where the entire cycle is simulated by using a multiple number of deforming grids, each one valid within a certain crank angle interval.

The proposed approach employs the utility available in the OpenFOAM code called snappyHexMesh, generating automatically 3D-meshes containing hexahedra and split-hexahedra from triangulated surface geometries in Stereolithography (STL) format.
Advantages of such utility are represented by the possibility to insert boundary layers on wall surfaces and local refinements in regions of interest. SnappyHexMesh allows grid generation in parallel with a consequent reduction of the pre-processing time required.
Automatic mesh generation process is intended to ensure high quality grids according to the following user specified parameters:

- maximum validity interval for each mesh, to avoid excessive stretching of the cells and loss of resolution;
- maximum allowed mesh non-orthogonality and skewness values, to increase stability and simulation accuracy;
- topological and geometrical validity of the mesh.

An iterative procedure, controlled by maximum specified non-orthogonality and skewness values, will be then used to morph the castellated mesh to the combustion chamber surface. The final result of the procedure is displaying a body fitted mesh conforming to surface boundaries and properly accounting for the main geometry details including sharp edges.
In conclusion, three different types of meshes consisting of a course mesh, first order fine mesh and a second order fine mesh, have been created. Compared to the fine mesh, the coarse mesh has lower levels of turbulence and higher numerical diffusivity. In particular, the flow in the fine mesh has a higher penetration and moves fast towards the cylinder head, producing a larger vortex. In the coarse mesh, instead, a vortex is created in the bottom part of the cylinder and its shape at BDC (bottom dead center) is very similar to the experimental one. Only the fine mesh with second order schemes is able to properly reproduce the same levels of turbulence which were experimentally found.

3.INTRODUCTION TO GRID GENERATION

Mesh generation is the practice of creating a mesh, a subdivision of a continuous geometric space into discrete geometric and topological cells. Often these cells form a simplicial complex. Usually the cells partition the geometric input domain. Mesh cells are used as discrete local approximations of the larger domain. Meshes are created by computer algorithms, often with human guidance through a graphical user interface,

depending on the complexity of the domain and the type of mesh desired. The goal is to create a mesh that accurately captures the input domain geometry, with high-quality (well-shaped) cells, and without so many cells as to make subsequent calculations intractable. The mesh should also be fine (have small elements) in areas that are important for the subsequent calculations.

Surface domains may be subdivided into triangular or quadrilateral shapes, while volumes may be subdivided primarily into tetrahedral or hexahedral shapes. Meshing algorithms ideally define the shape and distribution of the elements.[5]

3.1. Terminology

The terms "mesh generation," "grid generation," "meshing," " and "gridding," are often used interchangeably, although strictly speaking the latter two are broader and encompass mesh improvement: changing the mesh with the goal of increasing the speed or accuracy of the numerical calculations that will be performed over it. In computer graphics rendering, and mathematics, a mesh is sometimes referred to as a tessellation.

Mesh faces (cells, entities) have different names depending on their dimension and the context in which the mesh will be used. In finite elements, the highest-dimensional mesh entities are called "elements," "edges" are 1D and "nodes" are 0D. If the elements are 3D, then the 2D entities are "faces." In computational geometry, the 0D points are called vertices. Tetrahedra are often abbreviated as "tets"; triangles are "tris", quadrilaterals are "quads" and hexahedra (topological cubes) are "hexes."[5,6]

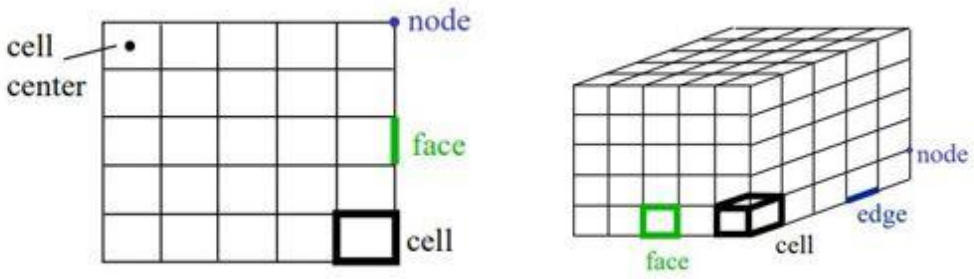

Figure 1. 2D and 3D computational grid

3.2. Types of Meshes

3.2.1. Common cell shapes

Two-dimensional

There are two types of two-dimensional cell shapes that are commonly used. These are the triangle and the quadrilateral. Computationally poor elements will have sharp internal angles or short edges or both.[5,7,11]

Triangle cell shape consists of 3 sides and is one of the simplest types of mesh. A triangular surface mesh is always quick and easy to create. It is most common in unstructured grids.

Quadrilateral cell shape is a basic 4 sided one as shown in the figure. It is most common in structured grids. Quadrilateral elements are usually excluded from being or becoming concave.

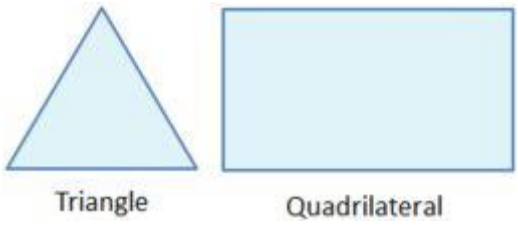

Figure 2. Basic two-dimensional cell shapes

Three-dimensional

The basic 3-dimensional element are the tetrahedron, quadrilateral pyramid, triangular prism, and hexahedron. They all have triangular and quadrilateral faces. Extruded 2-dimensional models may be represented entirely by the prisms and hexahedra as extruded triangles and quadrilaterals.

In general, quadrilateral faces in 3-dimensions may not be perfectly planar. A nonplanar quadrilateral face can be considered a thin tetrahedral volume that is shared by two neighboring elements.[11]

A tetrahedron has 4 vertices, 6 edges, and is bounded by 4 triangular faces. In most cases a tetrahedral volume mesh can be generated automatically.

A quadrilaterally-based pyramid has 5 vertices, 8 edges, bounded by 4 triangular and 1 quadrilateral face. These are effectively used as transition elements between square and triangular faced elements and other in hybrid meshes and grids.

A triangular prism has 6 vertices, 9 edges, bounded by 2 triangular and 3 quadrilateral faces. The advantage with this type of layer is that it resolves boundary layer efficiently. **A hexahedron**, a topological cube, has 8 vertices, 12 edges, bounded by 6 quadrilateral faces. It is also called a hex or a brick. For the same cell amount, the accuracy of solutions in hexahedral meshes is the highest.

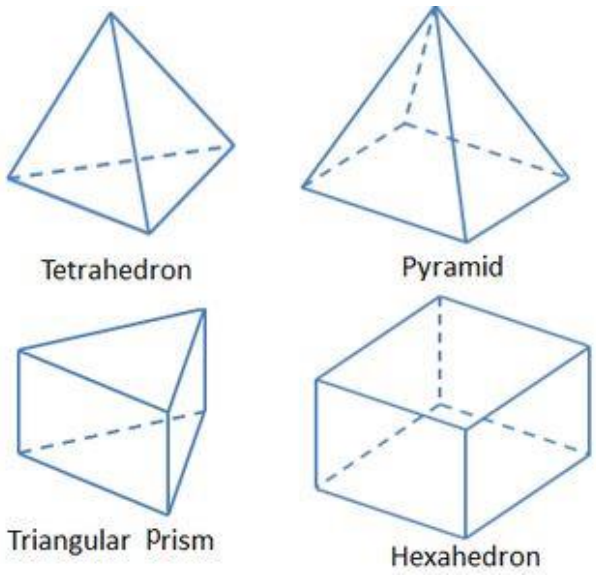

Figure 3. Basic three-dimensional cell shapes

3.2.2. Meshing types by elements

Meshes can be divided into two main groups by elements. These are Tri/Tetrahedral and Quad / Hexahedral meshes, which are considered in two-dimension/ three-dimension, respectively [9].

Tri/Tetrahedral meshes

Triangle and tetrahedral meshes are the most common forms of unstructured mesh generation. Most techniques currently in use can be considered in three main categories: Octree, Delaunay and Advancing Front techniques. Tri / tetrahedral meshes can also be constructed from quad / hexahedral mesh elements.

Quad/Hexahedral meshes

Surface domains can be subdivided into quadrilateral elements, whereas volumes can be subdivided into hexahedral elements by structured as well as unstructured meshing methods. Isoparametric coordinates can be used to generate both quad / hexahedral meshes, which are considered as structured. As for unstructured quad / hexahedral

meshes, they are generated using direct and indirect approaches. Meanwhile, some methods are also available that combine hexahedral and tetrahedral elements in a single three-dimensional domain. Generally, unstructured mesh generation algorithms use triangle and tetrahedral mesh elements. As a result of this, most of the literature and software are triangle and tetrahedral, although there is a significant group of literature that focuses on unstructured quad and hexahedral methods.

Hexahedral meshing generates meshes composed of deformed cubes (hexahedra). Such meshes are often used for simulating some physics (deformation mechanics, fluid dynamics…) because they can significantly improve both speed and accuracy.[12]

3.2.3. Meshing types by configuration

Meshes can be categorized as structured, unstructured and hybrid meshes by configuration. The choice of the mesh type is clearly related to the application.

Structured mesh

Structured grids are identified by regular connectivity. The possible element choices are quadrilateral in 2D and hexahedra in 3D. This model is highly space efficient, since the neighbourhood relationships are defined by storage arrangement. Some other advantages of structured grid over unstructured are better convergence and higher resolution. Structured meshes are composed of mesh elements that all interior nodes have an equal number of adjacent elements. They offer simplicity in software development and easy data access. Two-dimensional structured meshes typically use quadrilaterals, while three-dimensional structured meshes typically use hexahedra. Those types of meshes are generated by means of transfinite mapping methods. Structured meshes have been introduced in numerical analysis in the early 1970's after the finite element method became popular.

The basic advantage of structured meshes is that they offer simplicity and efficiency in numerical computations. A structured mesh requires significantly less memory than an unstructured mesh with the same number of elements, because array storage can define neighbour connectivity implicitly. A structured mesh can also save computation time to access neighbouring cells when computing a finite-difference stencil, where the software simply increments or 4 decrements array indices. Compilers can produce quite efficient codes for these operations. A major advantage of structured meshes lies in their

compatibility with efficient finite difference algorithms that are utilized in the solution of boundary value problems.

Despite the simplicity and efficiency of structured meshes, it can be difficult or impossible to compute a structured mesh for a complicated geometric domain. In addition, a structured mesh may require more elements than an unstructured mesh for the same problem, because elements in structured meshes have a fixed size whereas it is possible to grade elements in size in unstructured meshes. [5,9,10].

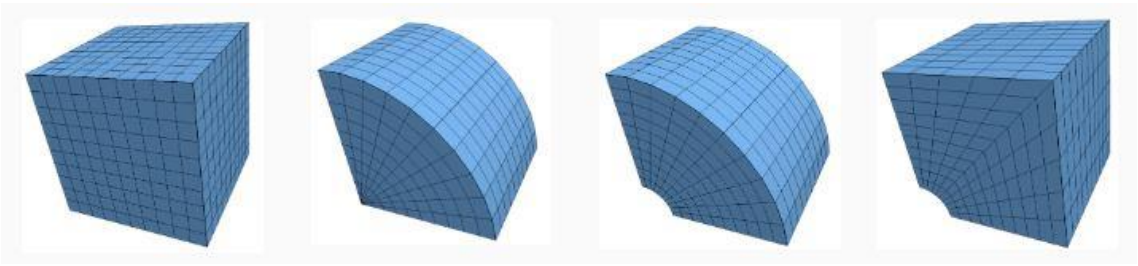

Figure 4.Examples of structured meshes (from left to right “brick”, “cylinder”, “cshell”, and “radcylinder”).

Unstructured mesh

Unlike structured mesh generation, unstructured mesh generation allows any number of elements to meet at a single node. When referring to unstructured meshing, triangle and tetrahedral meshes are commonly thought, even though quadrilateral and hexahedral meshes can be unstructured. While there is an overlap between structured and unstructured mesh generation technologies, the main feature that distinguishes the two fields is the unique iterative smoothing algorithms employed by structured mesh generators. Unstructured mesh generation has been part of mainstream computational geometry for some years. Well-studied geometric constructions such as Delaunay triangulation are central to unstructured mesh generation. The main advantages of unstructured meshes are [9]:

- Flexibility in fitting complicated domains.
- Rapid grading from small to large elements.
- Easy refinement and derefinement.

Three main approaches to unstructured mesh generation can be summarized as [2]:

- Octree based algorithms.
- Delaunay triangulation-based algorithms.
- Advancing front algorithms.

Originally, unstructured grids were mainly used in the theory of elasticity and plasticity, and in numerical algorithms based on finite-element methods. However, the field of application of unstructured grids has now expanded considerably and includes computational fluid dynamics. Some important aspects of the construction of unstructured grids are considered.[14]

Hybrid mesh

A hybrid mesh is formed by a number of structured meshes arranged in an overall unstructured pattern. Hybrid meshes fall somewhere in between structured and unstructured meshes. These meshes are used in problems with complicated geometries and widely used for the numerical analysis of boundary value problems in regions with a complex geometry and with a solution of complicated structure. They are formed by joining structured and unstructured grids on different parts of the region or surface. Commonly, a structured grid is generated about each chosen boundary segment. These structured grids are required not to overlap. The remainder of the domain is filled with the cells of an unstructured grid. This construction is widely applied for the numerical solution of problems with boundary layers. [9,14]

3.3. What Is A Good Quality Mesh?

A good mesh is a mesh that allows you to solve your problem at the expected level of accuracy within the time available for the project. At the point where mesh size and complexity become computational factors, mesh generation is always a compromise between time and accuracy.

Mesh generation can be a very time-consuming process. Therefore, judgement is needed to determine if a given mesh will perform well enough for a given model or if more effort is needed to improve its quality. Given the geometrical complexity of some models and the tools available for mesh creation, it is often necessary to accept meshes that deviate significantly from the known ideal shape.

The density of the mesh is required to be sufficiently high in order to capture details like displacement and stress …etc in areas of interest as well as properly mimic the physical mechanisms of the problem. On the same note, mesh density should not be so high that it captures unnecessary details that are not of interest to the project or beyond the precision of the field measurements. These burden the CPU (central process unit) and waste time.

3.3.1. Solution precision

A coarse mesh may provide an accurate solution if the solution is a constant, so the precision depends on the particular problem instance. One can selectively refine the mesh in areas where the solution gradients are high, thus increasing fidelity there. Accuracy, including interpolated values within an element, depends on the element type and shape.[11]

3.3.2. Rate of convergence

Each iteration reduces the error between the calculated and true solution. A faster rate of convergence means smaller error with fewer iterations.

A mesh of inferior quality may leave out important features such as the boundary layer for fluid flow. The discretization error will be large and the rate of convergence will be impaired; the solution may not converge at all.[11]

3.3.3. Grid independence

A solution is considered grid-independent if the discretization and solution error are small enough given sufficient iterations. This is essential to know for comparative results. A mesh convergence study consists of refining elements and comparing the refined solutions to the coarse solutions. If further refinement (or other changes) does not significantly change the solution, the mesh is an "Independent Grid."[11

3.4. Mesh Quality Indexes

If the accuracy is of the highest concern then hexahedral mesh is the most preferable one. The density of the mesh is required to be sufficiently high in order to capture all the flow features but on the same note, it should not be so high that it captures unnecessary details of the flow, thus burdening the cpu and wasting more time. Whenever a wall is present, the mesh adjacent to the wall is fine enough to resolve the boundary layer flow and generally quad, hex and prism cells are preferred over triangles, tetrahedrons and pyramids. Quad and hex cells can be stretched where the flow is fully developed and one-dimensional.

Based on the skewness, smoothness, and aspect ratio, the suitability of the mesh can be decided [7,11].

3.4.1. Skewness

The skewness of a grid is an apt indicator of the mesh quality and suitability. Large skewness compromises the accuracy of the interpolated regions. There are three methods of determining the skewness of a grid.

A skewness' of 0 is the best possible one and a skewness of one is almost never preferred. For Hex and quad cells, skewness should not exceed 0.85 to obtain a fairly accurate solution. For triangular cells, skewness should not exceed 0.85 and for quadrilateral cells, skewness should not exceed 0.9.[7]

Based on equilateral volume

This method is applicable to triangles and tetrahedral only and is the default method.

$$\boldsymbol{skewness = \frac{optimal\ cell\ size - cell\ size}{optimal\ cell\ size}}$$

Based on the deviation

This method applies to all cell and face shapes and is almost always used for prisms and pyramids.

$$\boldsymbol{skewness(for\ a\ quad) = max\left[\frac{\theta_{max} - 90}{90}, \frac{90 - \theta_{min}}{90}\right]}$$

Equiangular skewness

Another common measure of quality is based on equiangular skew.

$$\boldsymbol{Equiangle\ skewness = max\left[\frac{\theta_{max} - \theta_e}{180 - \theta_e}, \frac{\theta_e - \theta_{min}}{\theta_e}\right]}$$

where:

$\boldsymbol{\theta_{max}}$ is the largest angle in a face or cell,

$\boldsymbol{\theta_{min}}$ is the smallest angle in a face or cell,

$\boldsymbol{\theta_e}$ is the angle for equi-angular face or cell ; 60 for a triangle and 90 for a square.

Value of skewness	Cell quality
1	degenerate
0.9 - <1	bad(sliver)
0.75 – 0.9	poor
0.5-0.75	fair
0.25-0.5	good
>0-0.25	excellent
0	equilateral

Table 2. Skewness vs cell quality table

3.4.2. Smoothness

The change in size should also be smooth. There should not be sudden jumps in the size of the cell because this may cause erroneous results at nearby nodes.

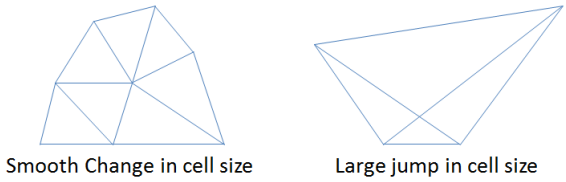

Figure 5. Smooth and large jump change

3.4.3. Aspect ratio

It is the ratio of longest to the shortest side in a cell. Ideally it should be equal to 1 to ensure best results. For multidimensional flow, it should be near to one. Also, local variations in cell size should be minimal, adjacent cell sizes should not vary by more than 20%. Having a large aspect ratio can result in an interpolation error of unacceptable magnitude.

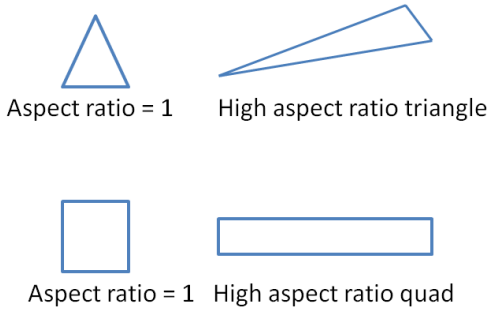

Figure 6. The changes in aspect ratio

3.4.4. Non-orthogonality

The mesh non-orthogonality is defined for a face as the angle **α** between the face area vector (**S**) and the vector joining the cell centers sharing the same face (**PQ**). Non-orthogonality affects the discretization accuracy of the diffusion term in transport equations [15,16] In industrial CFD simulations nonorthogonal meshes are commonly used to account for complex geometry features. However, if **α** > 80 the non-orthogonality is severe requiring to limit or discard the non-orthogonal component of the diffusion term and to increase the number of non-orthogonal correctors. **α** ≥ 90 is an index of mesh invalidity since this happens with degenerate cells.[17]

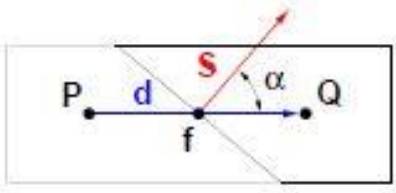

Figure 7. Non-orthogonality

3.5. How To Determine The Type Of Mesh To Use In CFD

When geometries are complex or the range of length scales of the flow is large, a triangular/tetrahedral mesh can be created with far fewer cells than the equivalent mesh consisting of quadrilateral/hexahedral elements. This is because a triangular/tetrahedral mesh allows clustering of cells in selected regions of the flow domain. Structured quadrilateral/hexahedral meshes will generally force cells to be placed in regions where they are not needed. Unstructured quadrilateral/hexahedral meshes offer many of the advantages of triangular/tetrahedral meshes for moderately-complex geometries.

A characteristic of quadrilateral/hexahedral elements that might make them more economical in some situations is that they permit a much larger aspect ratio than triangular/tetrahedral cells. A large aspect ratio in a triangular/tetrahedral cell will invariably affect the skewness of the cell, which is undesirable as it may impede accuracy and convergence. Therefore, if you have a relatively simple geometry in which the flow conforms well to the shape of the geometry, such as a long thin duct, use a mesh of high-aspect-ratio quadrilateral/hexahedral cells. The mesh is likely to have far fewer cells than if you use triangular/tetrahedral cells.

Converting the entire domain of your (tetrahedral) mesh to a polyhedral mesh will result in a lower cell count than your original mesh. Although the result is a coarser mesh, convergence will generally be faster, possibly saving you some computational expense.

In summary, the following practices are generally recommended [13]:

- For simple geometries, use quadrilateral/hexahedral meshes.
- For moderately complex geometries, use unstructured quadrilateral/hexahedral meshes.
- For relatively complex geometries, use triangular/tetrahedral meshes with prism layers.
- For extremely complex geometries, use pure triangular/tetrahedral meshes.

4.GEOMETRY

The geometries we will use for CFD analysis in this project are the intake and exhaust ports of a V16 engine. The ports provide the engine to breathe better. Thus, more power can be produced. However, the port geometries can be very complex to make meshing process harder to execute. The port geometries include a lot of small sub-shapes in detailed regions. That parts of the geometries could extend the time needed for generating mesh unnecessarily. That is what makes the cleaning-up the geometry remarkable.

Ensuring a well-defined, simplified, clean geometry will often be the difference between a successful high-quality mesh or a poor, illegal, cell-filled one. Geometries should be solid and have no abnormal features such as intersections or sharp outcroppings. A clean geometry means that it is enclosed and is free from geometrical defects.

Meshing packages have the challenge of dealing with various geometry problems [19]. Many of these problems can be generalized as file translation issues. Typically, the geometry used in a meshing package has not been created there but it may be created in

one of many CAD (Computer Aided Design) packages. Exporting these files out of CAD and into a neutral file format (IGES, STEP, SAT) accepted by the meshing software can introduce misrepresentations in the geometry. If the CAD and meshing packages do not support the same file formats, a second translation may be necessary, possibly introducing even more problems. For instance, we have received the intake and exhaust port geometries in .agdb file etention format.Whereas, we had to transform geometry files into other kinds of file extentions such as .stp, .scdoc, .igs and .stl. Step, Iges, Stl files can be used to import the geometry into SALOME. Iges, Step files can be used to importing the geometry into Gmsh. For snappyHexMesh, only .stl file format is available to read the geometry on the program. Unlike other programs, Ansys Spaceclaim offers a lot of file format for importing the geometry, 37 types of file extensions in total. Other programs can't reach the capability of Ansys at this point.

We encountered many difficulties during file conversions. We did all of the geometry modifications and cleaning-up processes using Spaceclaim and it gave us enough opportunities that we did not need other programs. We made all the geometry file extension transformations through the ansys spaceclaim.

There are many options in Spaceclaim to modify geometry file. This software offers simplicity by providing operations that are automatically applied to the geometry once one or more topology problems have been identified which is effective in many cases.

4.1. General Challenges

- **Bad geometry representation :** As a result of translation errors between CAD representations, errors or differences in the way the geometry is interpreted may occur. Depending on the severity of the problem, sometimes a mesh can be generated even with a less-than perfect geometric representation, however, in most cases, these should be resolved before meshing.
- **Small details in the model :** In some cases, there exist small details in the geometry that, if meshed, would result in very small elements and a potentially huge element budget. Small curves and surfaces can sometimes result from details in the design solid model that may not be necessary for analysis or may even be a result of careless construction of the CAD model. In either case, it is important to remove or modify these features before meshing.

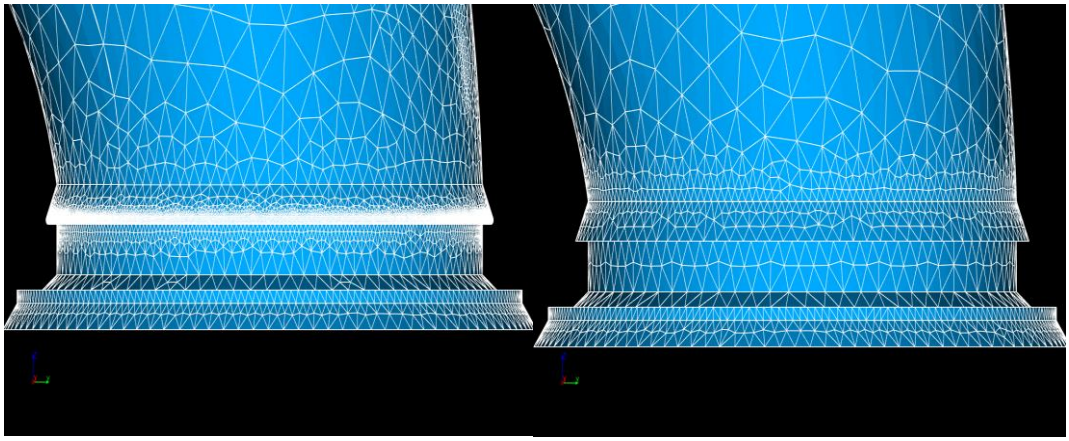

Figure 8. Example for Small detail problem

It can be seen how removing the fillet can reduce the density of meshes. We can accept the mesh density in interest regions but here, it is not an interest region and reducing the extra meshes doesn't change the simulation solution. However, it can reduce dramatically the solution.

Compatible topology for meshing scheme : Several meshing algorithms, such as the structured, mapping and sweeping techniques require a specific configuration of vertices, curves and surfaces in order to operate. Operations to decompose the geometry into a meshable topology are often needed. Other unstructured techniques like paving, and tetrahedral meshing do not require decomposition.

Conformal topology for assemblies: Assemblies of parts are often required to have a conformal mesh across their interface.The operations imprint and merge are often required to connect parts together so that when meshed, the representation will be a single continuous mesh.

4.2. Solution

SpaceClaim uses a fast detect-and-repair approach to fixing files, making model repair an easy activity for anyone who needs to work with bad data. SpaceClaim can automatically fix common issues when opening files, and it also provides interactive tools for deeper repairs. For parts that are missing geometry, SpaceClaim's direct modeling technology blends seamlessly with the repair tools to reconstruct data.[18]

Features provided by Spaceclaim:

- The Stitch tool makes watertight solids out of faces that were imported individually instead of forming a solid.
- The Gaps tool finds gaps between faces and fixes the edges to remove the gap.

- The Missing Faces tool extends neighboring faces or patches a new face where one was missing.
- Special tools heal split, extra, and inexact (tolerant) edges.
- A set of tools is dedicated to cleaning up and healing 2D geometry, and another tool fits curves over approximate or especially messy input.
- Other tools allow for adjusting tangency, simplifying spline surfaces to analytic surfaces, finding small faces, and merging small faces into a single face.

4.3. SpaceClaim

As it is already be mentioned, Operations which are enable on Spaceclaim to complete geometry modification and clean-up in an easy way [18]:

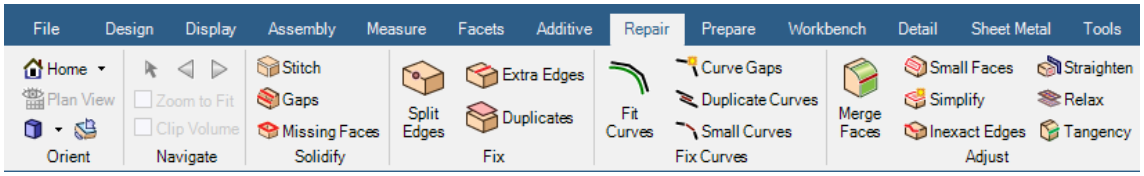

Figure 9. Spaceclaim clean-up options on its interface

Stitch : The Stitch tool combines surface part faces that are touching at their edges. When the merged faces form a closed surface, a solid is automatically created. You can use this tool to repair multiple surface parts that are in separate components. Coincident faces are detected and removed before Stitch merges surfaces into a single body.

Gaps : The Gaps tool removes gaps between faces. These gaps are usually found on parts imported from other CAD systems when the native format allows faces to fit together loosely. This tool only works for edges that are paired. Paired edges are edges that are within the maximum distance along their length or that share an end point and are within the maximum angle you set in the tool's options. Use the Missing Faces tool if you need to repair a part with edges that are not paired. When a gap is adjacent to a larger hole, this tool only repairs the gap and not the hole.

Missing Faces: The Missing Faces tool automatically detects and fills missing faces on an object. This tool should be used to find missing faces on imported designs. Use the Fill Closed tool to fill faces when you know where the edges of the new face should be. Use the Missing Faces tool to identify missing faces and fill them automatically or choose which missing faces you want created. As a precaution, if your design includes any open edge loops, first be certain that the loop are not simply imported parts that may have been designed as a surface body with open regions.

Split Edges : The Split Edges tool detects and merges coincident edges that do not mark the boundaries of new faces. Split edges should be done to give edge sizing at mesh generation part. Otherwise, the edge sizing will be more complicated. After importing the geometry into Spaceclaim, we have faced a lot of coincident edges in inlet area especially.

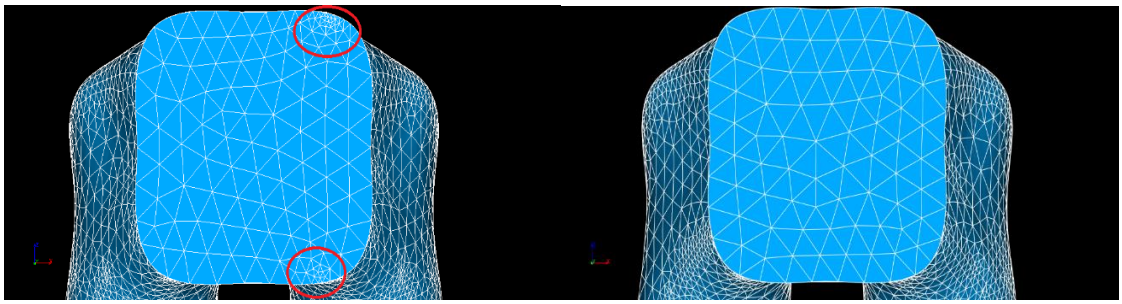

Figure 10. Before merging coincident edges (left) and after (right)

Extra Edges : The Extra Edges tool works like Merge Faces but operates on edges. Instead of merging two faces by selecting the faces, you select the edges between faces to remove the extra edge and merge the faces. But user should be careful while using this repair step. Because, edges that is created by user can be deleted.

Duplicates : The Duplicates tool detects and fixes duplicate faces. SpaceClaim highlights the duplicates and will remove them all, or you can select duplicates to exclude from being fixed.

Fit Curves : Use the Fit Curves tool to improve selected curves by replacing them with lines, arcs, or splines.

Curve Gaps : Use the CurveClosed Gaps tool to detect and fix gaps between curves.

Duplicate Curves : Use the Duplicate Curves tool to detect and remove duplicate curves.

Small Curves : Use the Small Curves tool to detect and remove small curves and fix the resulting gaps.

Merge Faces : The Merge Faces tool replaces two or more neighboring faces with a single new face that closely fits the original faces. Use this tool to simplify a model before you export it for analysis. Merging faces can result in a smoother mesh on the solid. You should only merge faces that are tangent or close to tangent; otherwise, the results may not be what you expect. You can't select edges when using the Merge Faces tool. You should use the Fill tool on the DesignClosed tab when you need to select an edge and a face to fill in a missing face. The Missing Faces and Gaps tools on the Prepare tab also perform this function.

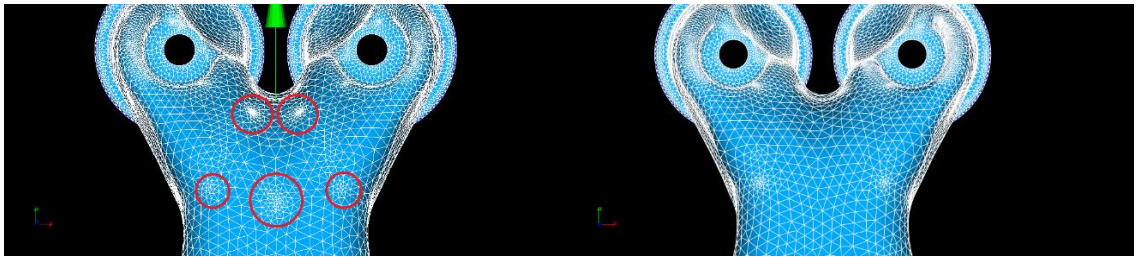

Figure 11. The difference between merged faces (right) and not merged faces (left) in mesh generation process

Small Faces : The Small Faces tool detects and removes small and sliver faces in your design. You may want to remove these faces before you export the design for analysis if they will have a negligible impact on the analysis accuracy but a significant impact on its speed.If the small face is tangent to a neighboring face, the tool will merge the small face with the neighboring face. If no neighboring face is tangent, the tool will extend neighboring faces to remove the small face.

Simplify : This tool examines a design and simplifies complex faces and curves into planes, cones, cylinders, lines, arcs, etc. This automates the one-by-one Simplify capability found in the Replace tool.

Inexact Faces : The Inexact Edges tool finds and repairs edges that have been inaccurately defined and do not meet precisely. These types of edges are usually found in designs imported from other CAD systems, particularly from conceptual design systems.

Straighten : The Straighten tool is used to look for holes and planar faces that are inclined at angles less than a specified value.

Relax : The Relax tool is used to look for surfaces that may have too many control points and reduce the number of points. Reducing the number of control points 'relaxes' the surfaces and makes them more stable.

Tangency : The Tangency tool detects edges between faces that are close to tangent and adjusts the faces so they are tangent. If a faces is near tangent with more than one neighboring face, you will get the best results if you make all the edges tangent at the same time.

Eliminate Unnecessary Fillets : Fillet surfaces increase the solution time and the number of grids. The fillet surfaces can be eliminated by using “Fill” command if they will not affect the results of the solutions. In this project, the fillet surfaces at valves are eliminated to decrease the number of grids and the solution time.

4.4. Visual Results For Geometry Cleaning-up Process

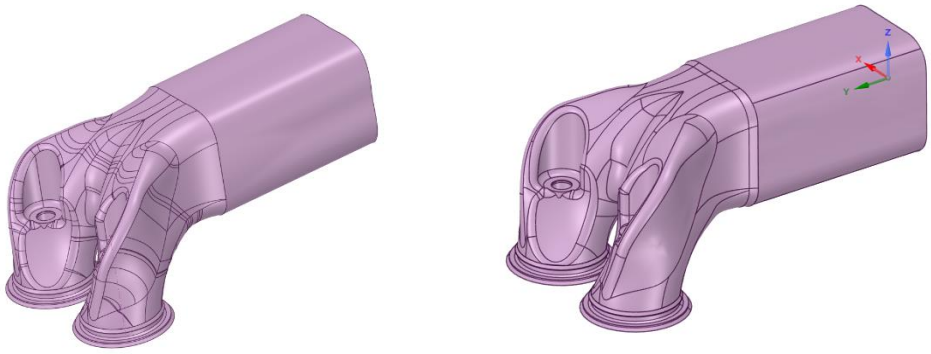

Figure 12. General view of intake port before (left) and after (right) modifications

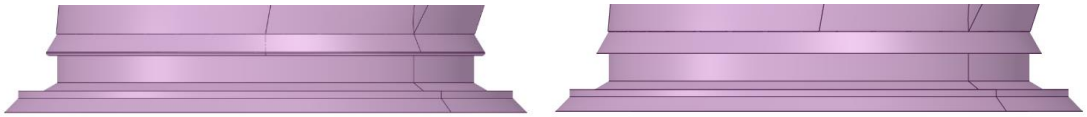

Figure 13. View of intake port's outlet before (left) and after (right) modifications

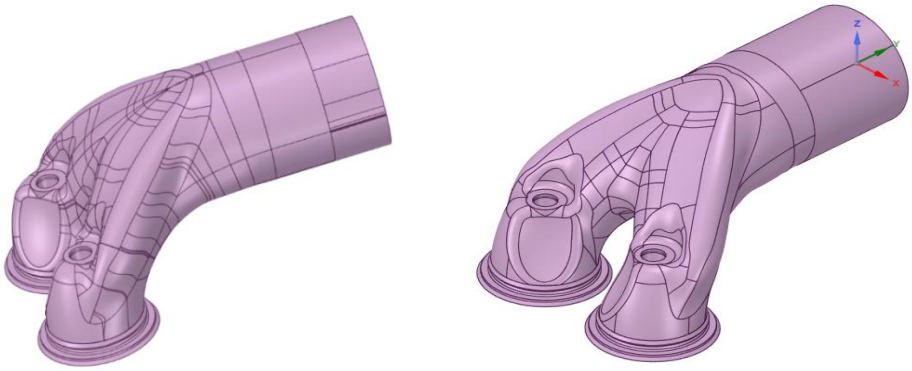

Figure 14. General view of exhaust port before (left) and after (right) modifications

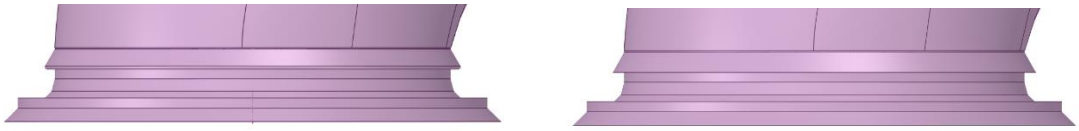

Figure 15. View of exhaust port's inlet before (left) and after (right) modifications

5. PRE COMPUTATION PROCESS

In this section, boundry conditions of intake and exhaust ports will be determined to use post-mesh process and required turbulence model will be determined. In practise, the velocity of fluid which flows through an engine's parts, is variable (not constant). As well as, boundry condition velocities of intake and exhaust ports depends on the act of engine cylinders. However, stationary valves will be taken into consideration and boundry condisiton velocities will be determined according to that way.

Velocity inlets for intake and exhaust ports are needed. Reynold number is also needed for determining of turbulent intensity. Therefore, a few calculations are needed before getting started with simulation.

5.1. Assumptions and Determination of Boundary Conditions

Engine properties: the engine uses diesel fuel. Equivalance ratio is 1 and air fuel ratio (AF) is 14.7:1. Engine works at 3000 RPM. Number of cylinders is 16 and total cylinder volume is 8L.

Volume of one cylinder (V_d) [27].

$$V_d = \frac{\pi * B^2 * S}{4} = \frac{Total\ Volume}{Number\ of\ Cylinders}$$

$$V_d = 5*10^{-4}$$

Assume Bore (B) and Stroke (S) are equal:

$$B = S = 0.086\ m$$

Mean Piston Velocity Formula (U_p) [27]

$$Up = 2 * S * N$$

where S is Stroke and N is revolutions per second.

$$U_p = 8.6 m/s$$

Mass flow rates should be equal at port and cylinder. Assume density of air is constant and volumetric flow rates also should be equal.

Volumetric Flow Rate Formula [27]:

$$V_{dot} = V.A$$

where V is velocity and A is perpendicular area to velocity.

$$V_{dot,port} = V_{dot,cylinder}$$

$$V_{intake,port} * A_{intake,port} = U_p * A_{piston,bottom}$$

where $A_{piston,bottom}$ is bottom are of the piston, $A_{intake,port}$ is intake port inlet area and $V_{intake,port}$ is intake port inlet velocity.

$$V_{intake,port} = 7.02 \text{ m/s}$$

$$V_{exhaust,port} * A_{exhaust,port} = U_p * A_{piston,bottom}$$

where $A_{piston,bottom}$ is bottom are of the piston, $A_{exhaust,port}$ is exhaust port inlet area and $V_{exhaust,port}$ is exhaust port inlet velocity.

$$A_{piston,bottom} = \frac{\pi * B^2}{4} = 5.81 * 10^{-3}\ m^2$$

$$V_{exhaust,port} = 23.05 \text{ m/s}$$

Inlet areas of intake and exhaust ports are determined from SPACECLAIM Software.

$$A_{intake,port} = 7117.9 \text{ mm}^2$$

$$A_{exhaust,port} = 2167.7 \text{ mm}^2$$

5.2. Reynolds Number

The Reynolds number is the ratio of inertial forces to viscous forces within a fluid which is subjected to relative internal movement due to different fluid velocities. A region where these forces change behavior is known as a boundary layer, such as the bounding surface in the interior of a pipe. A similar effect is created by the introduction of a stream of high-velocity fluid into a low-velocity fluid, such as the hot gases emitted from a flame in air. This relative movement generates fluid friction, which is a factor in developing turbulent flow. Counteracting this effect is the viscosity of the fluid, which tends to inhibit turbulence. The Reynolds number quantifies the relative importance of these two types of forces for given flow conditions, and is a guide to when turbulent flow will occur in a particular situation.[22]

This ability to predict the onset of turbulent flow is an important design tool for equipment such as piping systems or aircraft wings, but the Reynolds number is also used in scaling of fluid dynamics problems, and is used to determine dynamic similitude between two different cases of fluid flow, such as between a model aircraft, and its full-size version. Such scaling is not linear and the application of Reynolds numbers to both situations allows scaling factors to be developed.

With respect to laminar and turbulent flow regimes:

laminar flow occurs at low Reynolds numbers, where viscous forces are dominant, and is characterized by smooth, constant fluid motion;

turbulent flow occurs at high Reynolds numbers and is dominated by inertial forces, which tend to produce chaotic eddies, vortices and other flow instabilities.[21,23]

$$Re = \frac{D*V*\rho}{\mu}$$

where: ρ is the density of the fluid (SI units: kg/m3), V is the flow speed (m/s), D is the diameter of the tube (m), μ is the dynamic viscosity of the fluid (Pa·s or N·s/m2 or kg/(m·s)), ν is the kinematic viscosity of the fluid (m2/s).

Reynold number is needed to determine turbulent intencity. When setting boundary conditions for a CFD simulation it is often necessary to estimate the turbulence intensity on the inlets. To do this accurately it is good to have some form of measurements or previous experince to base the estimate on. Here are a few examples of common estimations of the incoming turbulence intensity:

High-turbulence case: High-speed flow inside complex geometries like heat-exchangers and flow inside rotating machinery (turbines and compressors). Typically the turbulence intensity is between 5% and 20%

Medium-turbulence case: Flow in not-so-complex devices like large pipes, ventilation flows etc. or low speed flows (low Reynolds number). Typically the turbulence intensity is between 1% and 5%

Low-turbulence case: Flow originating from a fluid that stands still, like external flow across cars, submarines and aircrafts. Very high-quality wind-tunnels can also reach really low turbulence levels. Typically the turbulence intensity is very low, well below 1%. [24,25]

$$P_{air,25\,C} = 1.2754\ kg/m^3$$

$$\mu_{air.25\,C} = 18.37*10^{-6}\ Pa.s$$

5.3. Turbulent Intensity Determination

The hydraulic diameter, d_h, is used instead of the geometrical diameter for channels of non-circular shape.

D_h is defined as:

$$D_h = \frac{4.*Cross-sectional\ area}{wetted\ perimeter}$$

For some special cases [26]:

$$D_{h,square} = a$$

$$D_{h,\text{two concentri tubes}} = D - d$$

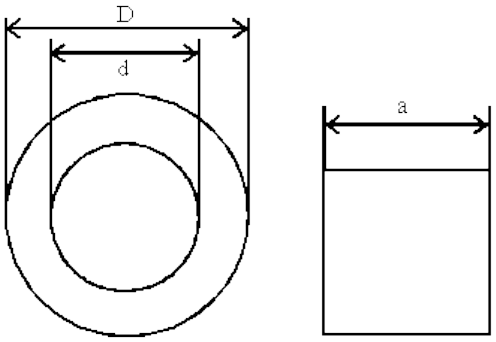

Figure 16. presentation of d, D and a.

Assume the geometric shape of intake port inlet is square.

$$D_{\text{intake port inlet}} = 87 \text{ mm}$$

$$d_{\text{in,outer pipe}} = 74 \text{ mm}$$

$$d_{\text{out,inner pipe}} = 64 \text{ mm}$$

$$D_{\text{exhaust port inlet}} = d_{\text{in,outer pipe}} - d_{\text{out,inner pipe}} = 10 \text{ mm}$$

$$Re_{intake\ port} = \frac{D_{intake\ port\ inlet} * V_{intake\ port} * \rho_{air\ 25\ C}}{\mu_{air\ 25\ C}} = 42402$$

$$Re_{exhaust\ port} = \frac{D_{exhaust\ port\ outlet} * V_{exhaust\ port} * \rho_{air\ 25\ C}}{\mu_{air\ 25\ C}} = 16003$$

The reynold numbers is in high-turbulence case. Therefore, turbulent intencity shuold be between 5% and 20%. In this project, the turbulent intensity is 5 because the reynold number in medium zone.

5.4. Turbulance Modeling

Turbulence modeling is the construction and use of a mathematical model to predict the effects of turbulence. Turbulent flows are commonplace in most real life scenarios, including the flow of blood through the cardiovascular system, the airflow over an aircraft wing, the re-entry of space vehicles, besides others. In spite of decades of research, there is no analytical theory to predict the evolution of these turbulent flows. The equations governing turbulent flows can only be solved directly for simple cases of flow. For most real life turbulent flows, CFD simulations use turbulent models to predict the evolution of turbulence. These turbulence models are simplified constitutive equations that predict

the statistical evolution of turbulent flows. To simulate turbulence, k-ε modelling is chosen in this project [28].

K-epsilon (k-ε) turbulence model is the most common model used in CFD to simulate mean flow characteristics for turbulent flow conditions. It is a two equation model that gives a general description of turbulence by means of two transport equations. The original impetus for the K-epsilon model was to improve the mixing-length model, as well as to find an alternative to algebraically prescribing turbulent length scales in moderate to high complexity flows [29].

- The first transported variable is the turbulent kinetic energy (k).
- The second transported variable is the rate of dissipation of turbulent kinetic energy (ε).

6.MESHING PROGRAMS AND MESH PROCESS

6.1. OPENFOAM

6.1.1. Introduction to Openfoam's Snappyhexmesh

OpenFOAM (for "Open-source Field Operation And Manipulation") is a C++ toolbox for the development of customized numerical solvers, and pre-/post-processing utilities for the solution of continuum mechanics problems, most prominently including computational fluid dynamics (CFD).

There are three main variants of OpenFOAM software that are released as free and open-source software under the GNU General Public License Version 3. In chronological order, these variants are as follows.[3]:

1. OpenFOAM variant by OpenCFD Ltd. (with the name trademarked since 2007) first released as open-source in 2004. (Note that since 2012, OpenCFD Ltd is an affiliate of ESI Group.)
2. FOAM-Extend variant by Wikki Ltd. (since 2009)
3. OpenFOAM Foundation Inc. variant, released by The OpenFOAM Foundation Inc. (since 2012), and transferred in 2015 to the English company The OpenFOAM Foundation Ltd.

In contradistinction to ordinary meshing programs, OpenFOAM doesn't offer a graphical user interface. Therefore, the user has to execute the processor by writing codes in the

terminal window and dictionary files (Dict files). Some people who are not good at writing codes or don't like, using OpenFOAM might be challenging.

SnappyHexMesh, supplied with OpenFOAM is used to create various meshes. The snappyHexMesh utility generates 3-dimensional meshes containing hexahedra (hex) and split-hexahedra (split-hex) automatically from triangulated surface geometries, or tri-surfaces, in Stereolithography (STL) or Wavefront Object (OBJ) format. The mesh approximately conforms to the surface by iteratively refining a starting mesh and morphing the resulting split-hex mesh to the surface. An optional phase will shrink back the resulting mesh and insert cell layers. The specification of mesh refinement level is very flexible and the surface handling is robust with a pre-specified final mesh quality. It runs in parallel with a load balancing step every iteration.[4]

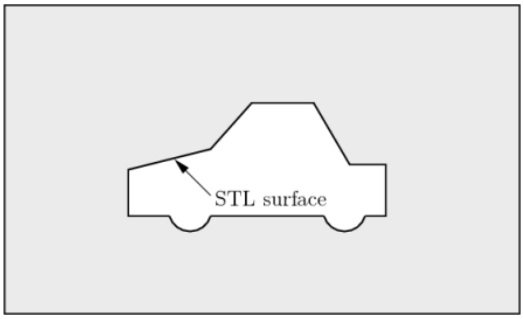

Figure 17. STL surface representation

6.1.2. Mesh generation process

The snappyHexMesh method available in Open- FOAM differs from the traditional way of doing pre-processing for CFD. This method uses automatic procedure to create orthogonal hexahedral mesh either around or inside a given geometry surface, which has to be provided in stereolithographic format. In principle this method enables fast and robust meshing of complex geometries. In this work, snappyHexMesh was applied to generate intake and exhaust geometry meshes [17].

In order to run *snappyHexMesh*, the user requires the following:

- a background hex mesh which defines the extent of the computational domain and a base level mesh density; typically generated using *blockMesh*.
- one or more tri-surface files located in a *constant/triSurface* sub-directory of the case directory;

- a *snappyHexMeshDict* dictionary, with appropriate entries, located in the *system* sub-directory of the case.

The *snappyHexMeshDict* dictionary includes: switches at the top level that control the various stages of the meshing process; and, individual sub-directories for each process. The entries are listed below.

- castellatedMesh: to switch on creation of the castellated mesh.
- snap: to switch on surface snapping stage.
- addLayers: to switch on surface layer insertion.
- mergeTolerance: merge tolerance as fraction of bounding box of initial mesh.
- geometry: sub-dictionary of all surface geometry used.
- castellatedMeshControls: sub-dictionary of controls for castellated mesh.
- snapControls: sub-dictionary of controls for surface snapping.
- addLayersControls: sub-dictionary of controls for layer addition.
- meshQualityControls: sub-dictionary of controls for mesh quality.

OpenFOAM commands	Functions
>>surfaceAutoPatch	Extract all the surfaces
>>grepsolid	Show the surfaces with patch names
>>blockMesh	Create background mesh
>>paraview	Display any mesh or geometry
>>surfaceFeatureExtract	Prepare surface mesh for snappyhexamesh
>>snappyHexMesh	Mesh the geometry using hexa-mesh
>>checkMesh -allGeometry -allTopology -meshQuality	Check the quality of meshes
>>paraFoam	Display mesh or geometry

Table 3. Some main commands for hexa-meshing

6.1.2.1. Creating the background hex-mesh

Before *snappyHexMesh* is executed the user must create a background mesh of hexahedral cells that fills the entire region within by the external boundary as shown in Figure 18. Its size represents the initial mesh density on the basis of the geometry to be meshed and user-specified settings.

This can be done simply using *blockMesh*. The following criteria must be observed when creating the background mesh:

- the mesh must consist purely of hexes.
- the cell aspect ratio should be approximately 1, at least near surfaces at which the subsequent snapping procedure is applied, otherwise the convergence of the snapping procedure is slow, possibly to the point of failure.
- there must be at least one intersection of a cell edge with the tri-surface.

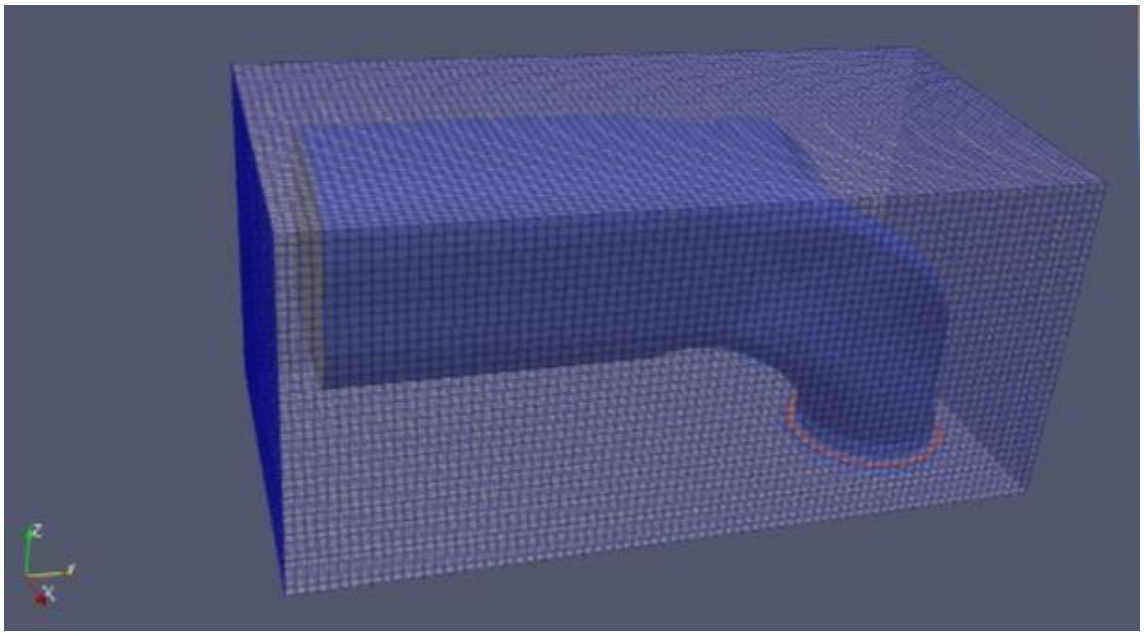

Figure 18. Initial mesh generation in *snappyHexMesh* meshing process (Block Mesh) with the intake port geometry in it.

6.1.2.2. Cell splitting at feature edges and surfaces, and refinements.

Cell splitting is performed according to the specification supplied by the user in the *castellatedMeshControls* sub-dictionary in the *snappyHexMeshDict*. The entries for *castellatedMeshControls* are presented below.

- locationInMesh: location vector inside the region to be meshed; vector must not coincide with a cell face either before or during refinement.
- maxLocalCells: max number of cells per processor during refinement.
- maxGlobalCells: overall cell limit during refinement .
- minRefinementCells: if minRefinementCells $\geq$ number of cells to be refined, surface refinement stops.

- nCellsBetweenLevels: number of buffer layers of cells between successive levels of refinement (typically set to 3).
- resolveFeatureAngle: applies maximum level of refinement to cells that can see intersections whose angle exceeds resolveFeatureAngle (typically set to 30).
- features: list of features for refinement.
- refinementSurfaces: dictionary of surfaces for refinement.
- refinementRegions: dictionary of regions for refinement.

```
// Surface based refinement
   refinementSurfaces
   {
       intakeportgeometry
       {
           // Global surface-wise min and max refinement level
           level (1 1);

           // Local surface-wise min and max refinement level
                       // You will need to use the name given in the STL file
           regions
                       {
               patch3                          //inlet1
               { level (2 2); patchInfo { type patch; } }

               patch0                          //inlet2
               { level (2 2); patchInfo { type patch; } }

               patch4                          //outlet
               { level (2 2); patchInfo { type patch; } }
                       }
       }
   }
```

Figure 19. Specific refinement levels of global surface, inlet and outlet

6.1.2.3. Max skewness and non-orthogonality set-up

The mesh quality is controlled by the entries in the *meshQualityControls* sub-dictionary in *snappyHexMeshDict*; entries are listed below.

- maxNonOrtho: maximum non-orthogonality allowed (degrees, typically 65).
- maxBoundarySkewness: max boundary face skewness allowed (typically 20).
- maxInternalSkewness: max internal face skewness allowed (typically 4).

```
//- Maximum non-orthogonality allowed. Set to 180 to disable.
maxNonOrtho 80;

//- Max skewness allowed. Set to <0 to disable.
maxBoundarySkewness 4;          //original 20
maxInternalSkewness 4;

//- Max concaveness allowed. Is angle (in degrees) below which concavity
//  is allowed. 0 is straight face, <0 would be convex face.
//  Set to 180 to disable.
maxConcave 80;
```

Figure 20. Max skewness and non-orthogonality set-up using codes

After all these parameters have been written in dict files, hex-meshing can be started by entering the code >>snappyHexMesh in the terminal. Then OpenFOAM starts to generate meshes by entered parameters.

6.1.3. Meshing results

After meshing finishs , by writing >>paraFoam , output of the snappyHexaMesh can be displayed in Paraview. Six meshing process is completed in total. Half of them belongs to the intake geometry and the other half belongs to the exhaust geometry.

Geometry	**Mesh name**	**Mesh parameters**					
		Refinements level				Max skewness input	
		Global surface	inlet	outlet	edges	For boundaries	For internal meshes
intake	Mesh1	1	1	1	1	20	4
	Mesh2	2	2	2	2	15	2
	Mesh3	2	2	2	2	1	1
exhaust	Mesh4	1	1	1	1	10	4
	Mesh5	2	2	2	2	5	2
	Mesh6	2	2	2	3	1	1

Table 4. Mesh parameters for hex-mesh

Geometry	**Mesh name**	**Mesh informations**										
		#nodes	#faces	# volume cells					Max aspect raito	Max skewness	Max Non-orthogonality	Mesh timing (s)
				hexa	prisms	tet wedges	polyhedra	In total				
intake	Mesh1	202515	576999	178236	4471	92	5637	188436	6.09	3.99	75°	64.38
	Mesh2	1148665	3286808	1022831	17315	147	33597	1073890	5.56	3.85	60°	401.04
	Mesh3	1153071	3291210	1025294	15395	160	33041	1073890	3.93	1.00	60°	746.66
exhaust	Mesh4	143246	410683	125602	4521	68	4729	134920	5.70	2.12	75°	47.96
	Mesh5	834086	2390774	734402	18334	167	30321	783224	5.62	3.42	60°	284.20
	Mesh6	1194151	3373250	1016651	25642	694	55014	1098001	3.67	1.00	60°	958.66

Table 5. All the mesh information

6.1.4. Visual representation of meshes

6.1.4.1. Intake geometry

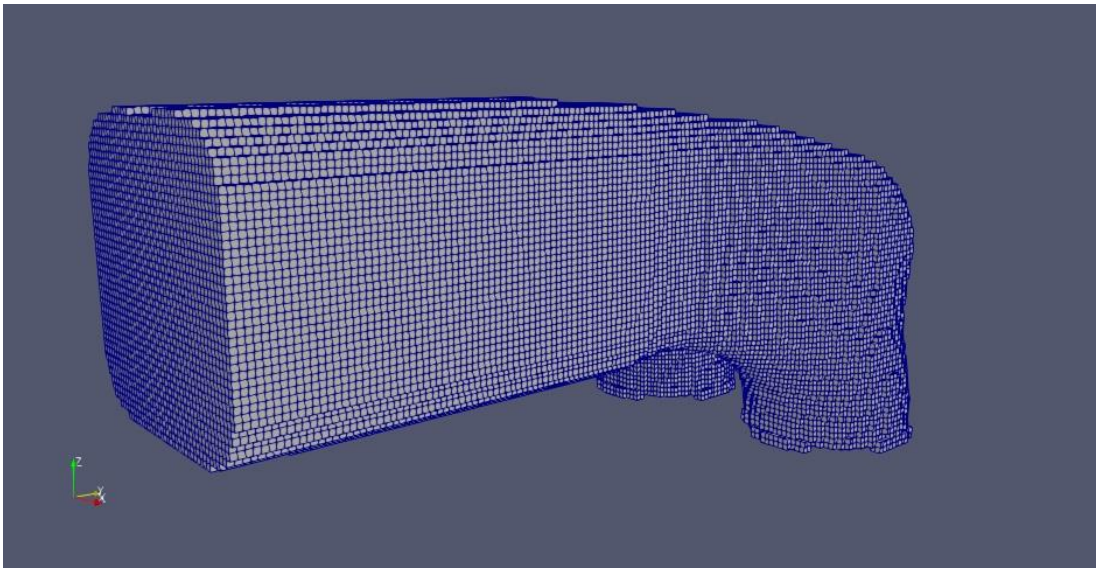

Figure 21. Mesh1, A coarse mesh example of intake port geometry

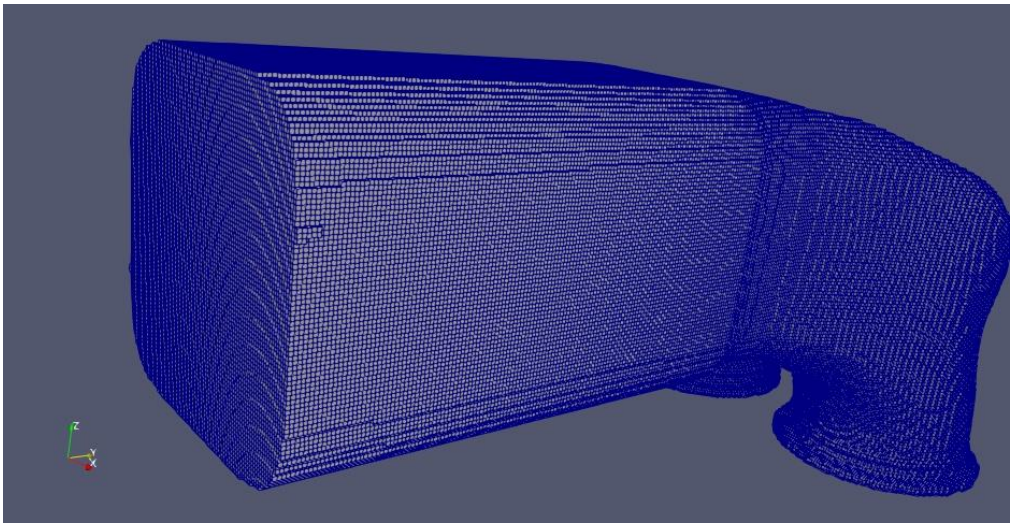

Figure 22. Mesh 3, A fine mesh example which refinements are applied

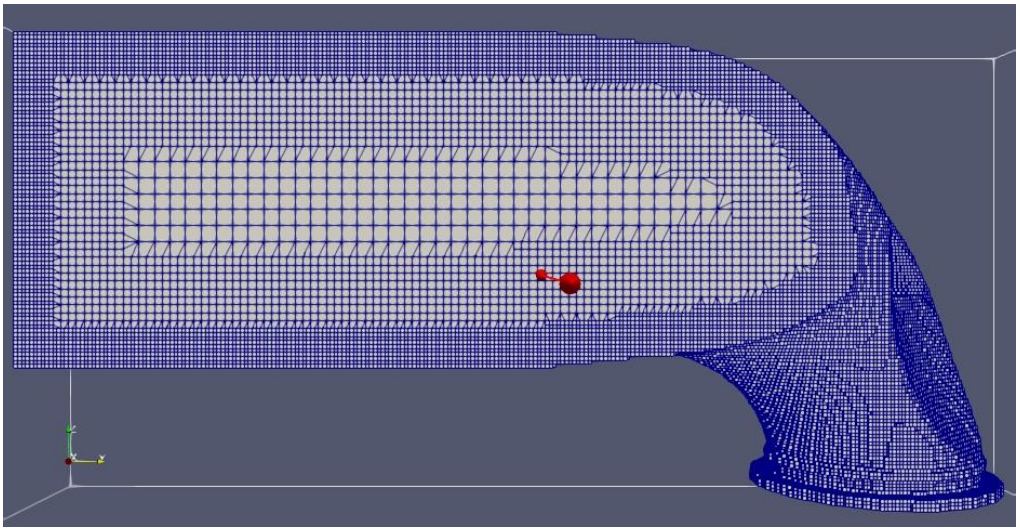

Figure 23. Mesh 3, clipped in y direction

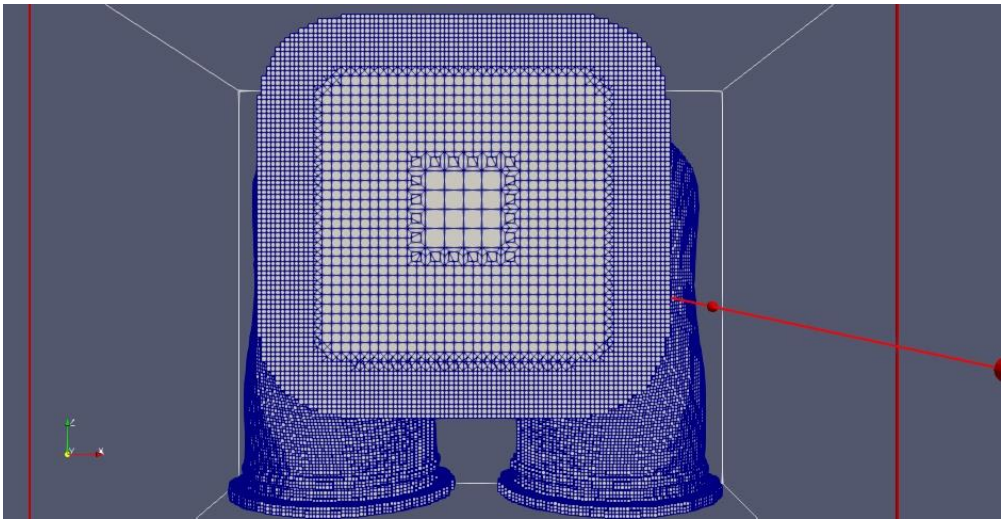

Figure 24. Mesh 3, clipped in x direction

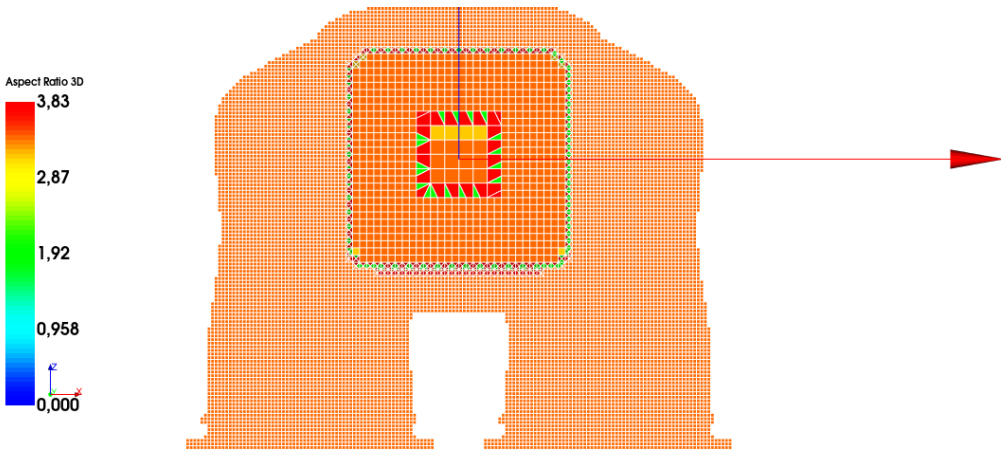

Figure 25. Aspect ratio representation of the 3D cells in Mesh 3 (clipping in x direction)

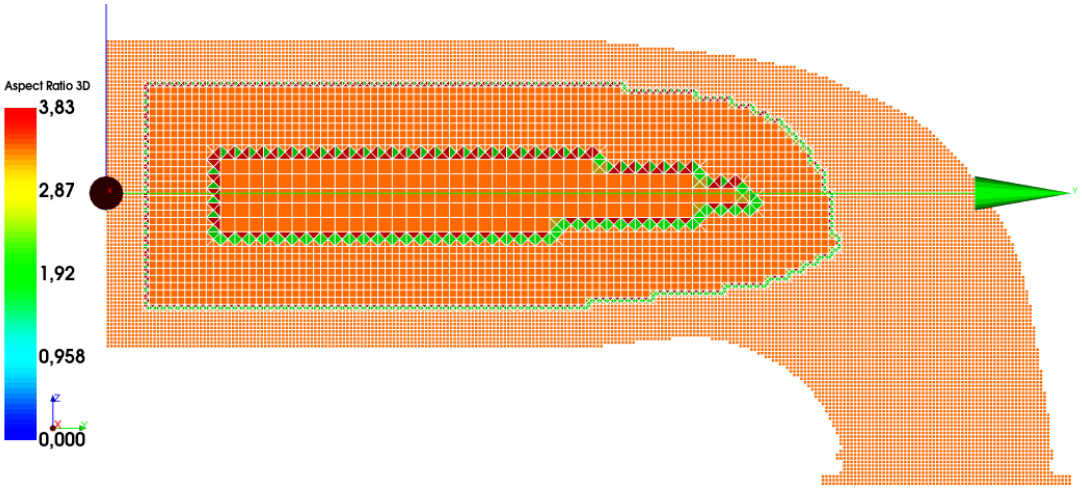

Figure 26. Aspect ratio representation of the 3D cells in Mesh 3 (clipping in y direction)

6.1.4.2. Exhaust geometry

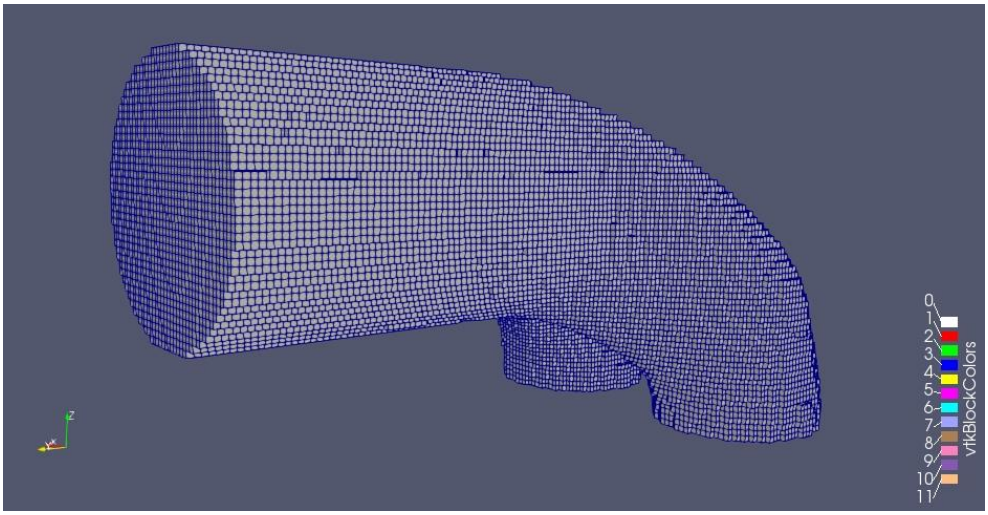

Figure 27.Mesh4, A coarse mesh example of exhaust port geometry

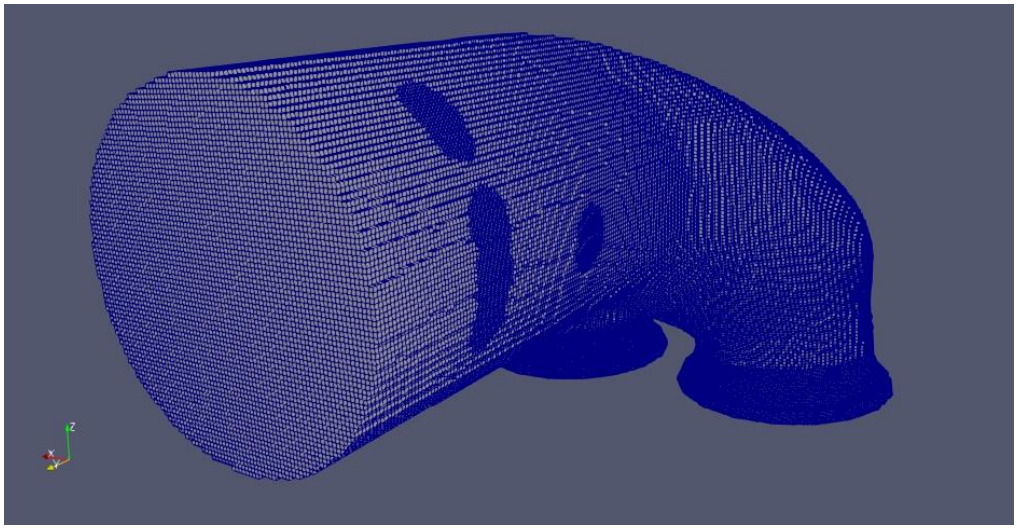

Figure 28. Mesh 6, A castellated mesh of exhaust

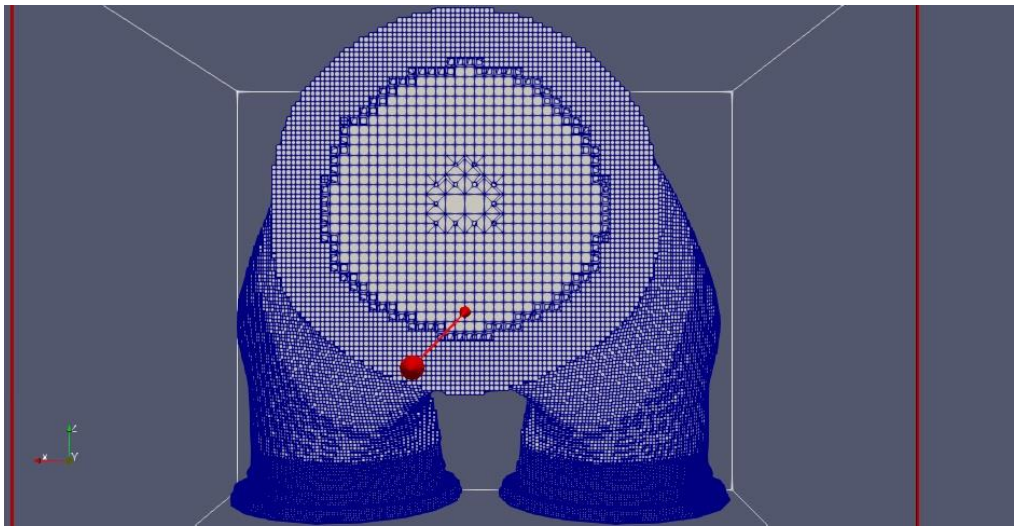

Figure 29. Mesh 6, clipped in x direction

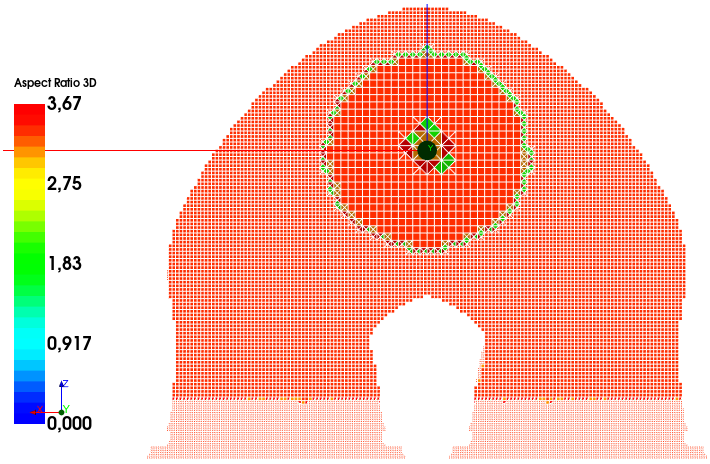

Figure 30. Aspect ratio representation of the 3D cells in Mesh 6 (clipping in x direction)

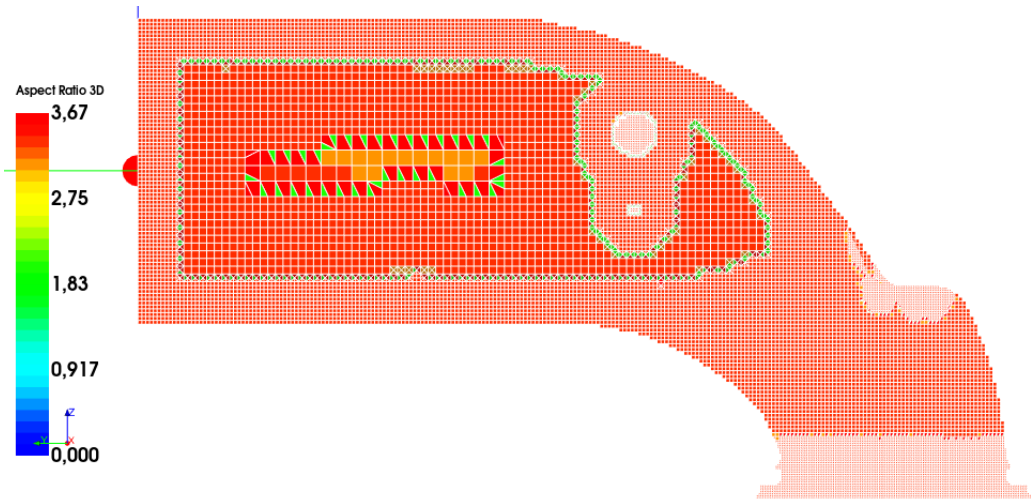

Figure 31. Aspect ratio representation of the 3D cells in Mesh 3 (clipping in y direction)

6.1.5. Interpretation of meshes

In that part of the report, SnappyHexMesh by OpenFOAM, was demonstrated in respect of capability to generate fine structured hexahedral mesh. Three of meshes are created for one of the geometries, six meshes in total.

At first mesh, all the refinement parameters set up as level 1 and maximum skewness inputs set up as default which is twenty for boundary mesh and four for internal mesh. These parameters were changed step by step to obtain better mesh. At the last meshes of both geometries, refinement parameters are increased to level 2 and maximum skewness inputs are limited as 1 which means that the maximum skewness value through the whole geometry couldn't be more than 1.

On the other hand, snappyHexMesh runs in parallel and operates a load balancing step for every iteration. Both these aspects make mesh generation process very fast and guarantee an optimal domain decomposition for the subsequent gas flow simulations [17].

In conclusion, maximum aspect ratio of 3D cells could be reduced to 3.83 for the intake geometry and 3.67 for the exhaust geometry. Therewithal, it is succeeded to restrict the maximum skewness as 1 for both geometries. While getting coarse mesh initially, when we changed the parameters, it can be produced the better meshes which have enough quality to converge in solving for CFD simulation.

6.2. SALOME

6.2.1. Introduction to Salome

SALOME is a free software that provides a generic platform for pre and post-processing for numerical simulation. It is based on an open and flexible architecture made of reusable components available as free software.

SALOME is a CAD/CAE integration platform. It provides reusable components for:

1. 3D modelling (bottom-up construction, import, healing);
2. Visualization;
3. Computational schemas management;
4. Post-processing.

SALOME is tailored for integration of custom components:

1. CAD interfaces;
2. Mesh generators;
3. Finite Element solvers with specific pre-processors.

The SALOME platform is available in Open Source.

6.2.2. Mesh generation in Salome

To create a mesh on geometry which is imported as .STEP file in the geometry window of SALOME, it is necessary to create a mesh object by choosing:

- a geometrical shape produced in the Geometry module (main shape);
- meshing parameters, including
    - meshing algorithms and
    - hypotheses specifying constraints to be taken into account by the chosen meshing algorithms.

Then you can launch mesh generation by invoking Compute command.

Mesh generation on the geometry is performed in the bottom-up flow: nodes on vertices are created first, then edges are divided into segments using nodes on vertices; the node of segments are then used to mesh faces; then the nodes of faces are used to mesh solids. This automatically assures the conformity of the mesh.

It is required to choose a meshing algorithm for every dimension of sub-shapes up to the highest dimension to be generated. Note that some algorithms generate elements of several dimensions, and others of only one. It is not necessary to define meshing parameters for all dimensions at once; you can start from 1D meshing parameters only,

compute the 1D mesh, then define 2D meshing parameters and compute the 2D mesh. An algorithm of a certain dimension chosen at mesh creation is applied to discretize every sub-shape of this dimension. It is possible to specify a different algorithm or hypothesis to be applied to one or a group of sub-shapes by creating a sub-mesh. You can specify no algorithms at all at mesh object creation and specify the meshing parameters on sub-meshes only; then only the sub-shapes, for which an algorithm and a hypothesis (if any) have been defined will be discretized. Construction of a mesh on a geometry includes at least two (mesh creation and computing) of the following steps [20]:

- **Creation of a mesh object**, where you can specify meshing parameters to apply to all sub-shapes of the main shape.
- **Creation of sub-meshes,** (optional) where you can specify meshing parameters to apply to the selected sub-shapes.
- **Evaluating mesh size** (optional) can be used to know an approximate number of elements before their actual generation.
- **Previewing the mesh** (optional) can be used to generate mesh of only lower dimension(s) in order to visually estimate it before full mesh generation, which can be much longer.
- **Changing sub-mesh priority** (optional) can be useful if there are concurrent sub-meshes defined.
- **Computing the mesh** uses defined meshing parameters to generate mesh elements.
- **Editing the mesh** (optional) can be used to modify the mesh of a lower dimension before computing elements of an upper dimension.

6.2.3. Meshing results

In Salome, 3 types of mesh have been generated: tetrahedron mesh, quad-dominant mesh and hybrid mesh. To reach to the best mesh (min cell number for a convergent mesh), we have started generating mesh from the course mesh to the finest one step by step. As a result, all the meshes are represented in the following tables.

				Meshing parameters						**Mesh informations**			**convergence**			
geometry	**mesh name**	**Mesh type**	**Mesh algorithm**	Max cell size	fineness	Step size	# of segmentsper edge	# of segmentsper radius	Order type	# of 3D cells	Max 3D aspect ratio	Duration (s)	k	ε	c	Duraiton (min)
intake	Mesh 7	tetrahedral	Netgen	0.005	moderate	0.3	1	2	first	147965	193	40	X	X	X	-
	Mesh 8	tetrahedral	Netgen	0.005	fine	0.2	2	3	first	286670	381	70	X	X	✓	-
	Mesh 9	tetrahedral	Netgen	0.003	fine	0.2	2	3	first	403065	579	99	✓	X	✓	-
	Mesh 10	tetrahedral	Netgen	0.003	fine	0.15	2	4	first	652459	75	151	✓	X	X	-
	Mesh 11	tetrahedral	Netgen	0.003	fine	0.15	3	4	first	709398	358	179	✓	X	X	-
	Mesh 12	tetrahedral	Netgen	0.003	Very fine	0.12	3	4	first	773919	56	220	✓	✓	✓	22
	Mesh 13	tetrahedral	Netgen	0.003	Very fine	0.11	3	4	first	884667	78	195	✓	✓	✓	29
	Mesh 14	tetrahedral	Netgen	0.003	Very fine	0.10	3	4	first	1570808	624	351	X	X	✓	-
exhaust	Mesh 15	tetrahedral	Netgen	0.005	coarse	0.5	0.5	1.5	first	51428	61	20	✓	X	✓	3
	Mesh 16	tetrahedral	Netgen	0.005	moderate	0.3	1	2	first	71206	58	27	✓	✓	✓	5
	Mesh 17	tetrahedral	Netgen	0.004	moderate	0.3	1	2	first	118494	58	37	✓	✓	✓	-

Table 6. Salome tetrahedral mesh results

geometry	mesh name	Mesh type	Mesh algorithm	Meshing parameters						Mesh informations				
				Max cell size	fineness	Step size	# of segmentsper edge	# of segmentsper radius	Order type	# of 3D cells			Max 3D aspect ratio	Duration (s)
										tetrahedron	pyramid	total		
intake	Mesh 18	Quad Dominant	Netgen	0.033	moderate	0.3	1	2	first	566984	42243	609237	1810	190
	Mesh 19	Quad Dominant	Netgen	0.010	moderate	0.3	1	2	first	572251	42567	615088	1810	192
	Mesh 20	Quad Dominant	Netgen	0.010	fine	0.3	1	2	second	572251	42567	615088	395	197
	Mesh 21	Quad Dominant	Netgen	0.010	fine	0.1	1	2	second	2136909	64581	2201490	285	591
exhaust	Mesh 22	Quad Dominant	Netgen	0.010	fine	0.2	3	5	first	1032358	27449	1059807	5960	367
	Mesh 23	Quad Dominant	Netgen	0.010	Very fine	0.1	3	5	second	1032358	27449	1059807	333	389
	Mesh 24	Quad Dominant	Netgen	0.005	Very fine	0.1	3	5	second	1061051	2806	1089097	398	530

Table 7. Salome quad-dominated mesh results

geometry	mesh name	Mesh type	Mesh algorithm	Meshing parameters							Mesh informations					
				Max cell size	fineness	Step size	# of segmentsper edge	# of segmentsper radius	local lengt	Order type	# of 3D cells				Max 3D aspect ratio	Duration (s)
											tetrahedron	hexahedron	pyramid	total		
intake	Mesh 25	Hybrid	Netgen Mapping	0.005	moderate	0.3	1	2	0.0020	second	217093	2000	900	219993	30	150
	Mesh 26	Hybrid	Netgen Mapping	0.010	moderate	0.3	1	2	0.0015	first	409302	127764	14625	551691	182	330
	Mesh 27	Hybrid	Netgen Mapping	0.010	fine	0.15	1	2	0.0015	first	1559529	127764	14625	17101918	200	1800
									# of segments							
exhaust	Mesh 28	Hybrid	Netgen Mapping	0.005	moderate	0.3	1	2	30	first	130522	13500	2700	146722	191	22
	Mesh 29	Hybrid	Netgen Mapping	0.005	fine	0.2	1	2	30	first	173023	13500	2700	189223	164	30
	Mesh 30	Hybrid	Netgen Mapping	0.005	Very fine	0.15	1	2	30	first	238364	13500	2700	254564	161	39

Table 8. Salome hybrid mesh results

6.2.4. Visual representation of meshes

6.2.4.1. Visual representation of intake meshes generated in Salome

- Tetrahedral mesh:

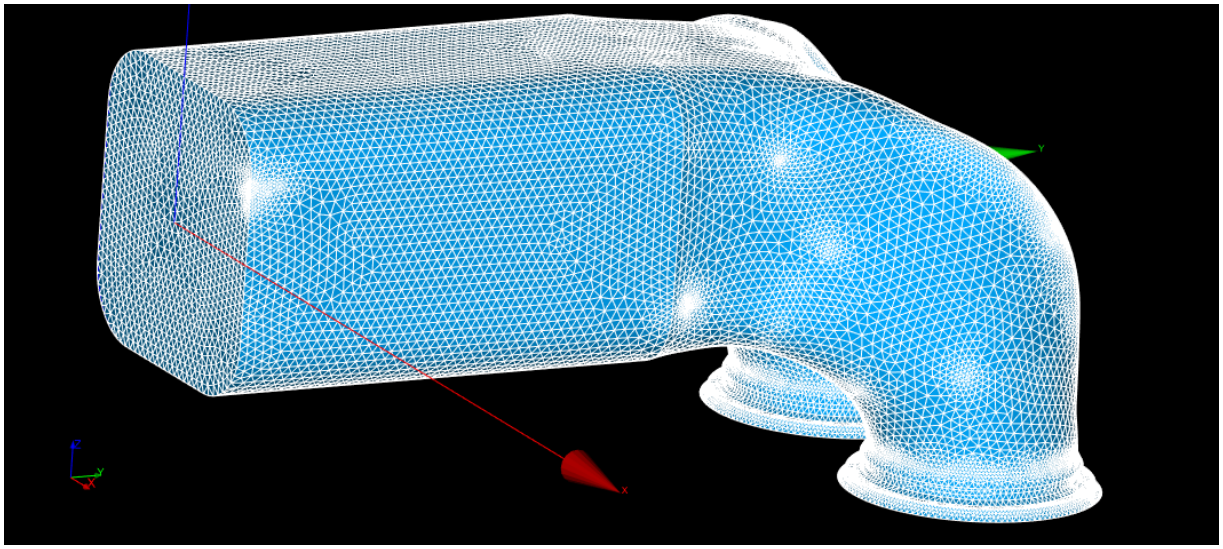

Figure 32. mesh12 general mesh view

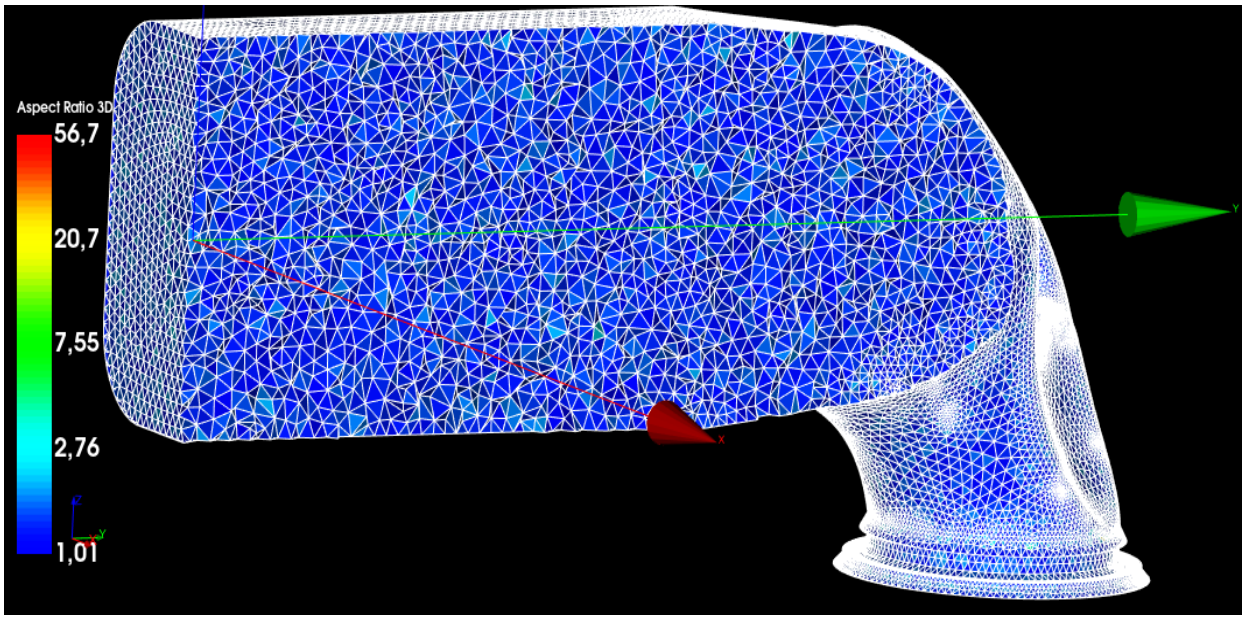

Figure 33. mesh12 clipping and represented by aspect ratio in logarithmic scale

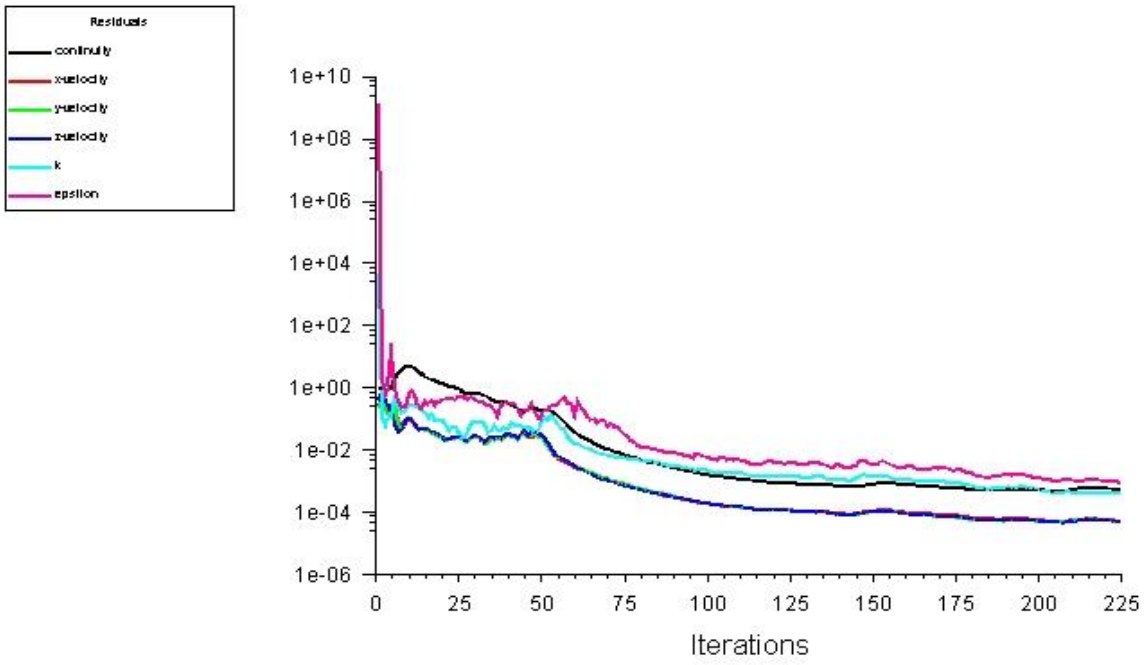

Figure 34. Residual vs iteration graph of mesh12

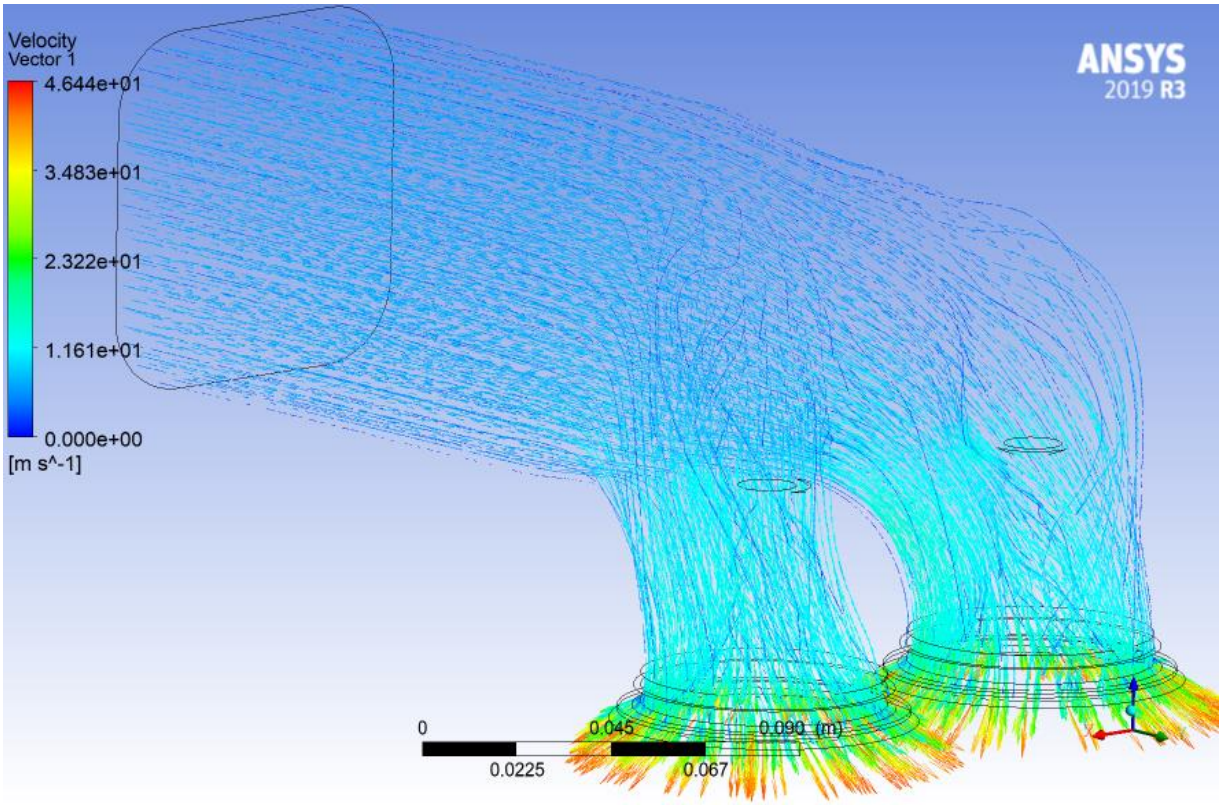

Figure 35. Representation of velocity vectors on stream lines in intake port according to mesh12

- Quad dominant mesh:

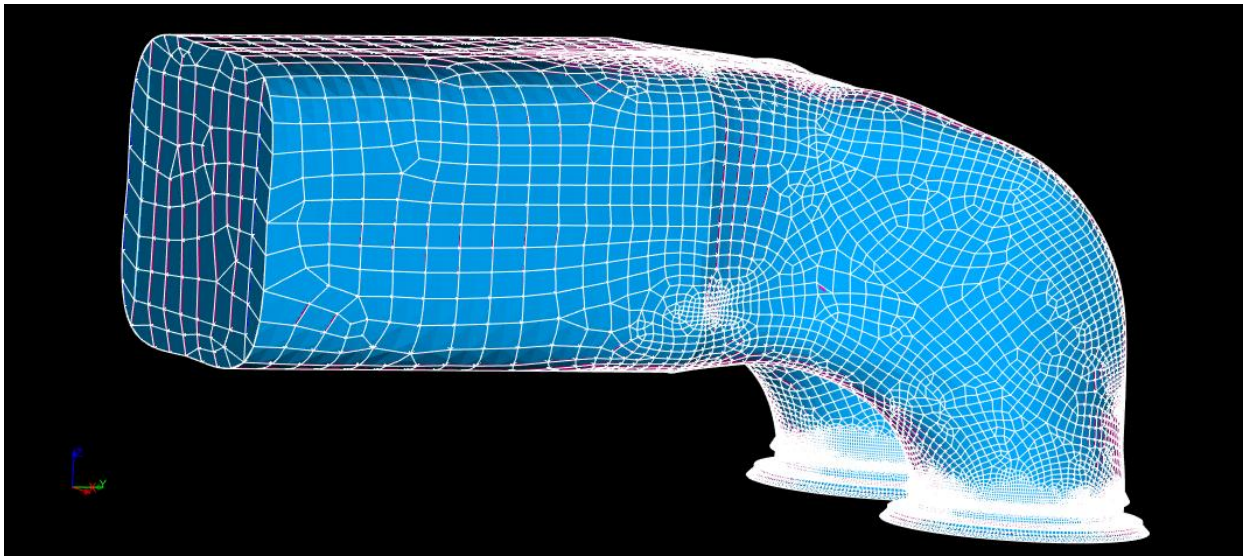

Figure 36. mesh21 general mesh view

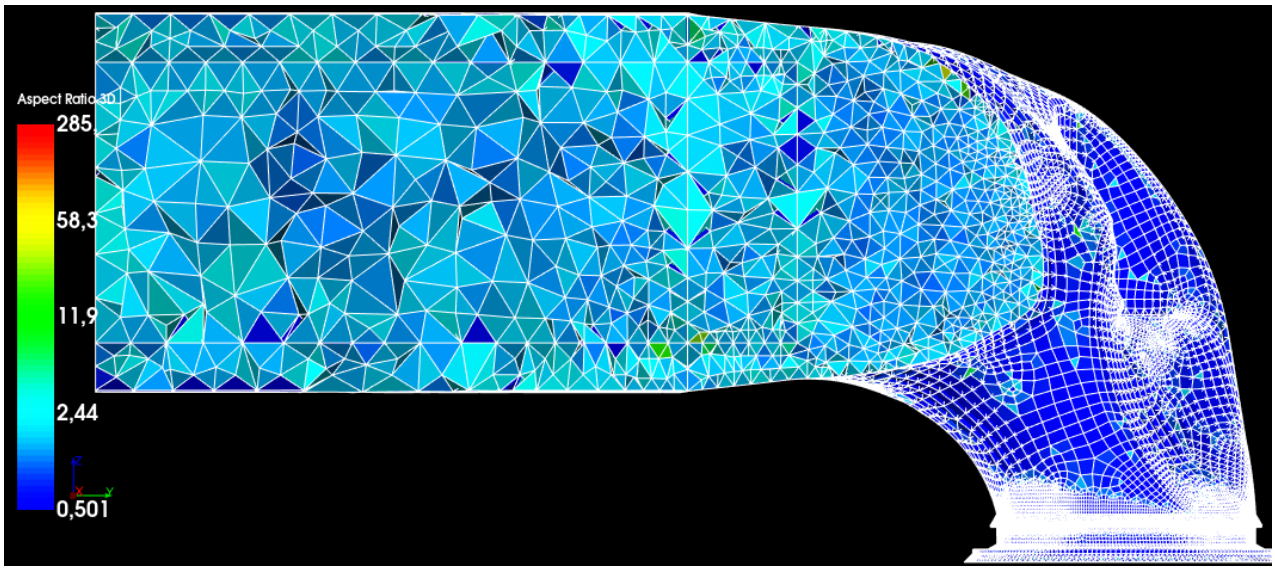

Figure 37. mesh21 clipping and represented by 3D aspect ratio in logarithmic scale

- Hybrid mesh:

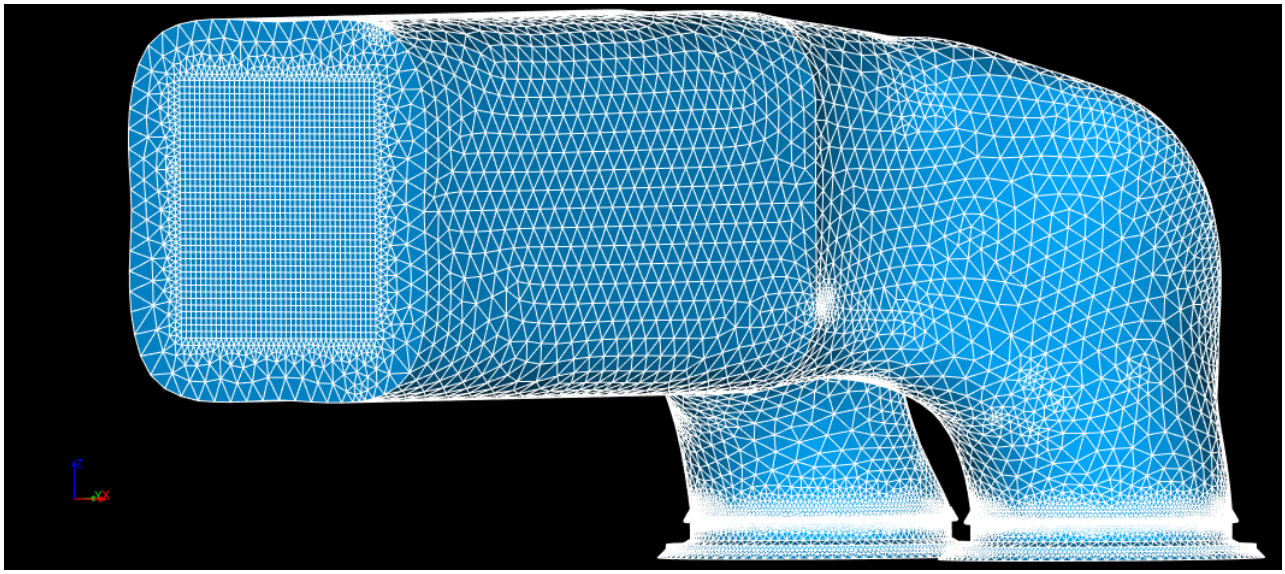

Figure 38. mesh26 general mesh view

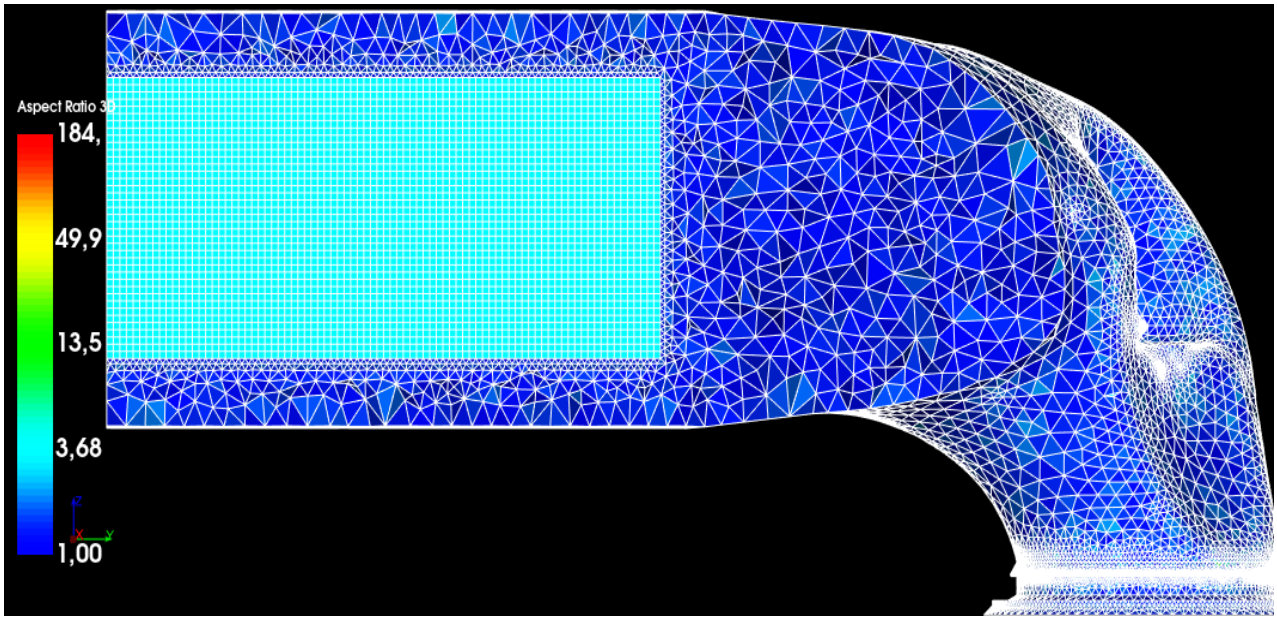

Figure 39. mesh26 clipping and represented by aspect ratio in logarithmic scale

6.2.4.2. Visual representation of exhaust meshes generated in Salome

- Tetrahedral mesh:

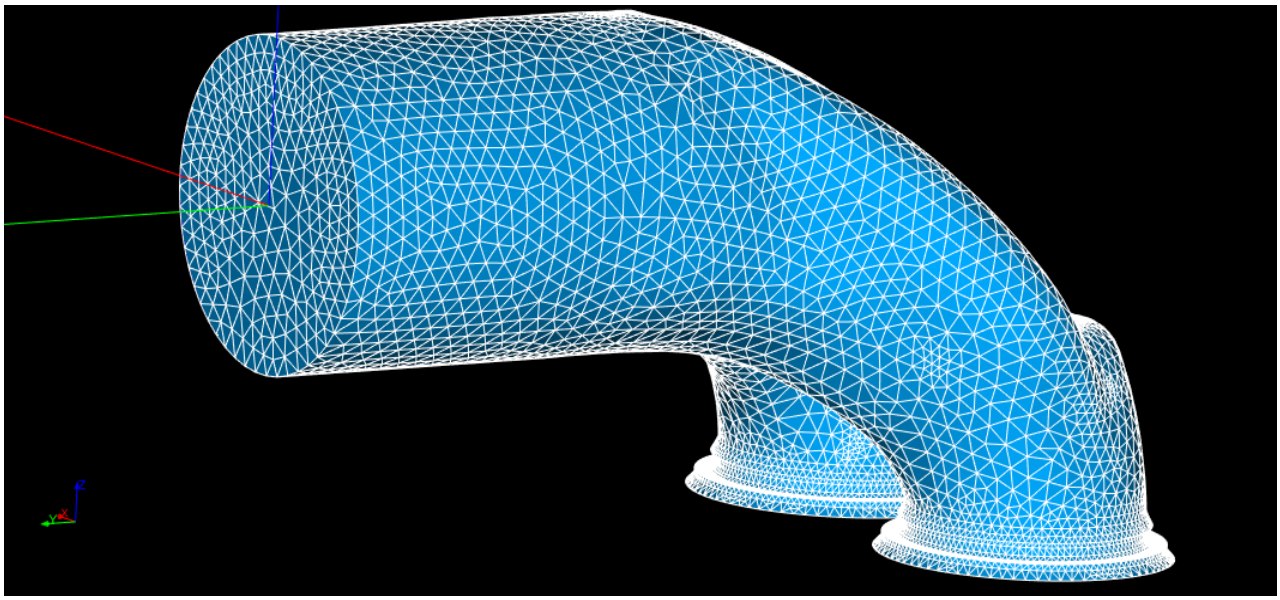

Figure 40. mesh16 general mesh view

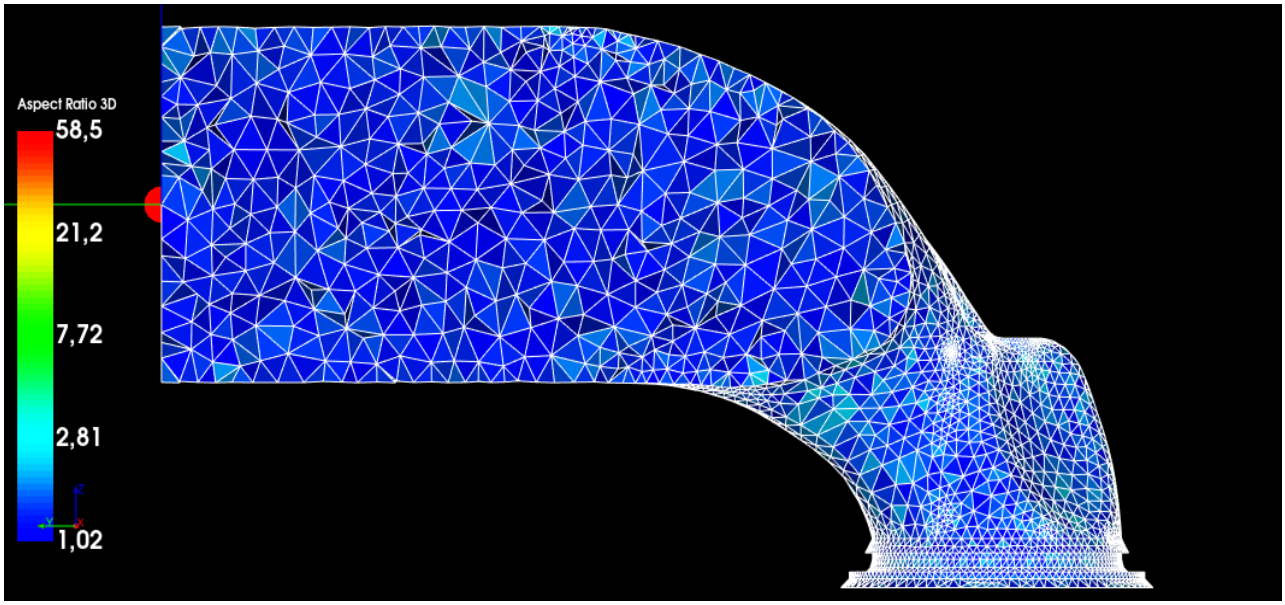

Figure 41. mesh16 clipping and represented by aspect ratio in logarithmic scale

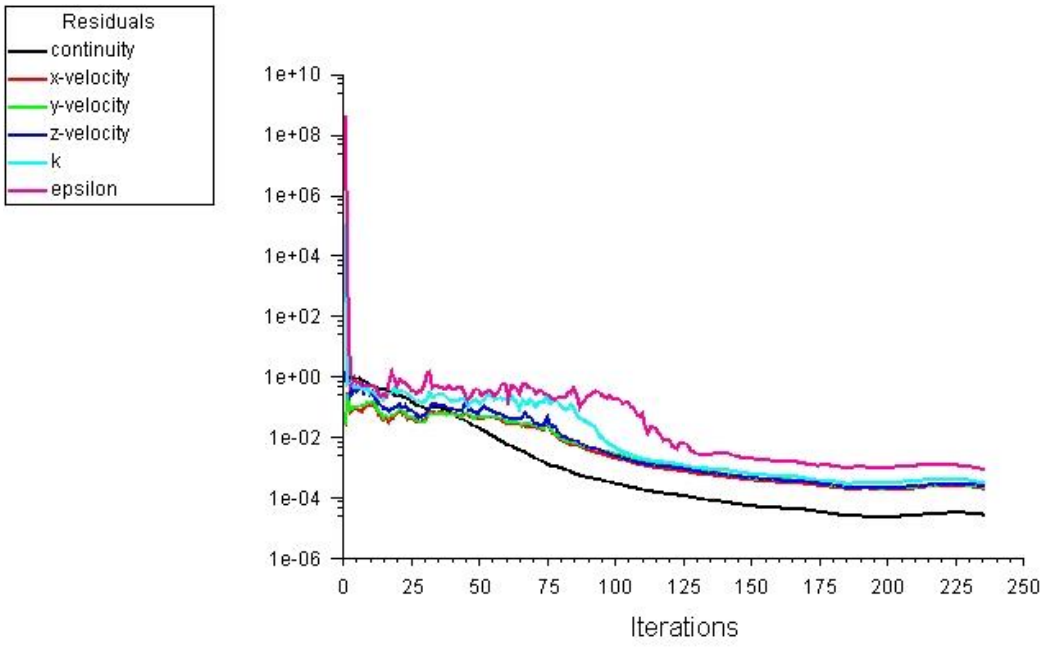

Figure 42. Residual vs iteration graph of mesh16

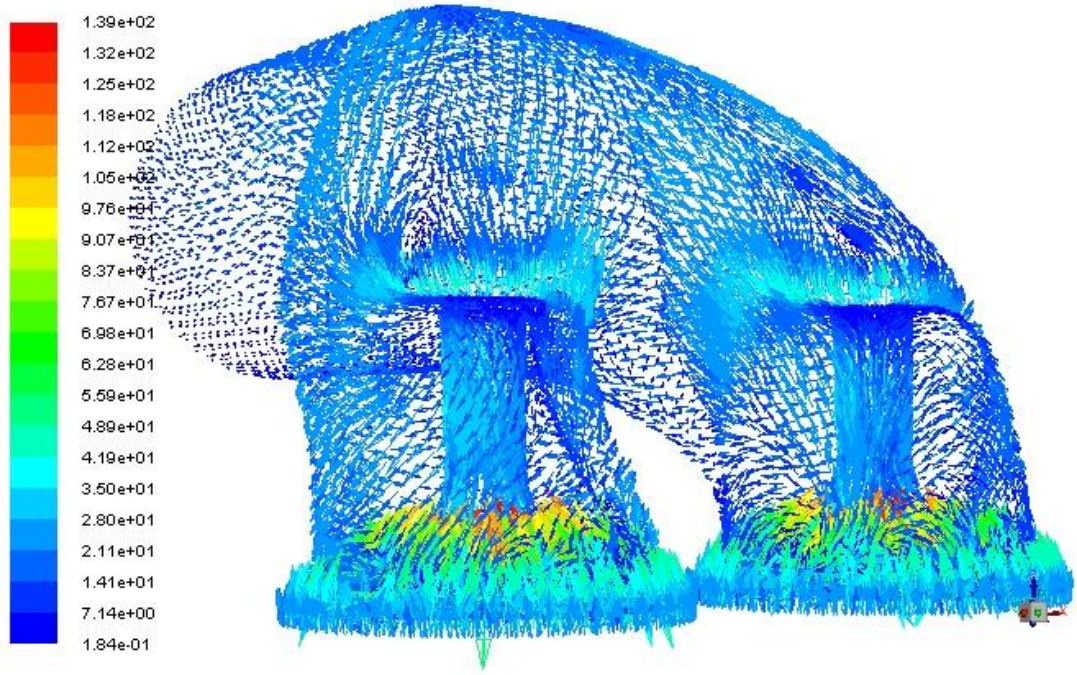

Figure 43. Representation of velocity vectors on stream lines in intake port according to mesh16

- Quad dominant mesh

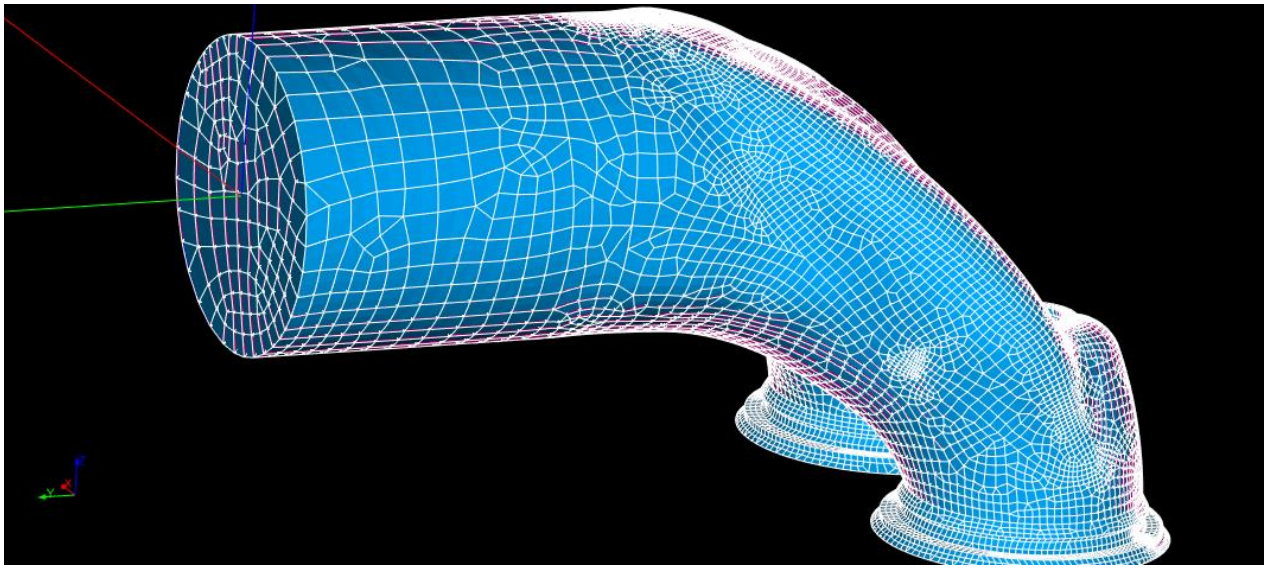

Figure 44. mesh 23 general mesh view

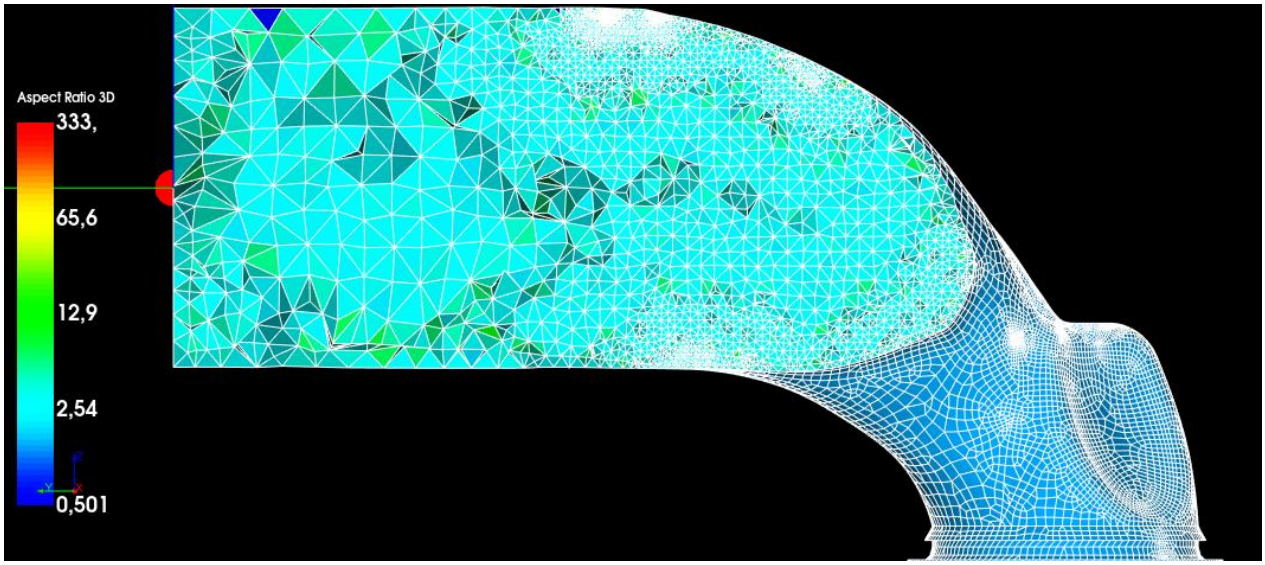

Figure 45. mesh23 clipping and represented by aspect ratio in logarithmic scale

- Hybrid mesh :

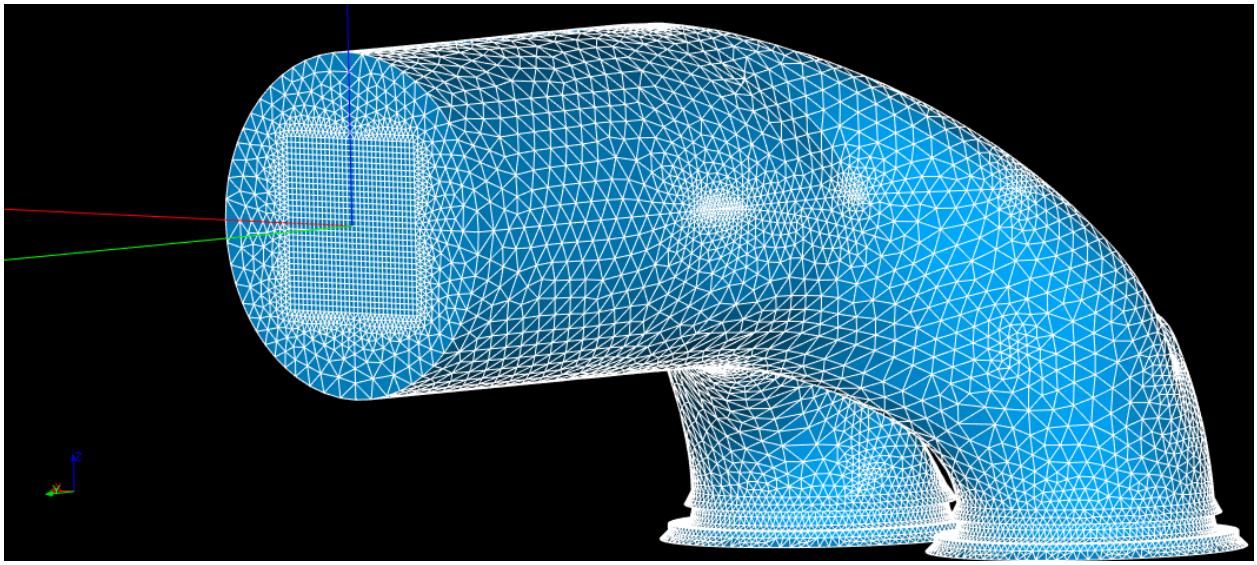

Figure 46. mesh30 general mesh view

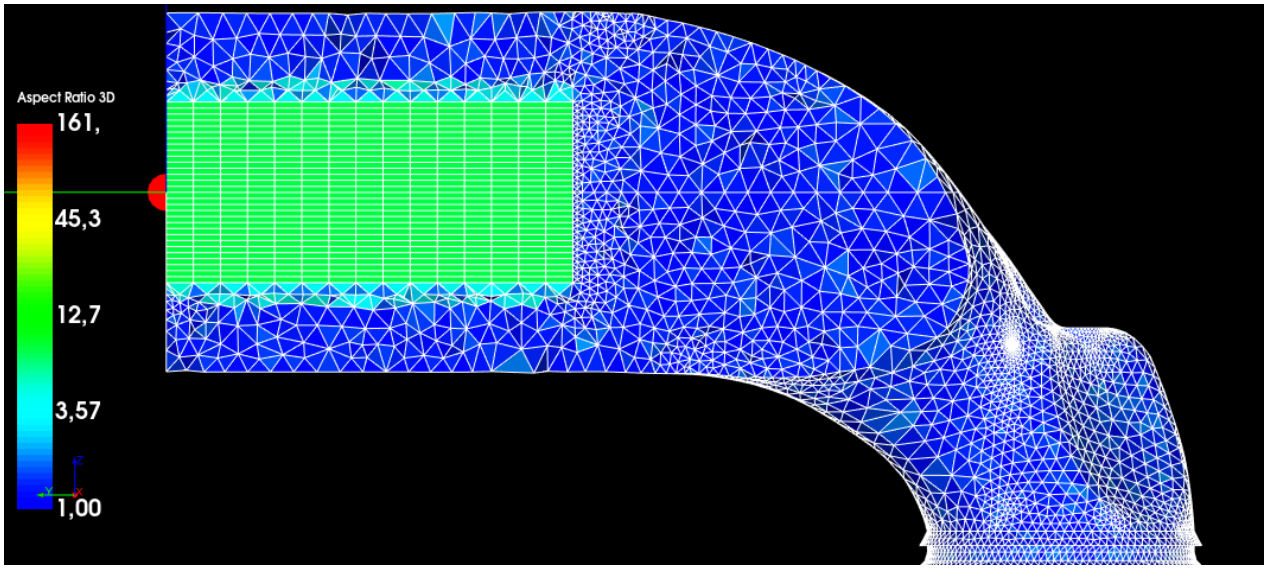

Figure 47. mesh30 clipping and represented by aspect ratio in logarithmic scale

6.2.5. Interpretation of meshes

In that part of the paper, Salome's meshing skills are examined by generating 3 types of meshes: tetrahedral, quad-dominated and hybrid. However, we have used Ansys mesh solver so we had to export meshes .unv format which ignores the pyramid meshes. Since hybrid and quad dominant meshes include pyramid mesh, we couldn't solve hybrid and quad dominant meshes.

Firstly, tetrahedral meshes were generated in Salome and results are quite convincing. For intake port geometry, it is found that 773919 of cells is enough to generate good quality mesh which brings along convergent results in k, ε, and continuity (c). On the other hand, it was required less number of cells to reach convincing results for exhaust port geometry. This can be rooted in being less complex of exhaust geometry. A mesh with 71206 cells is enough to get convergent results in solver.

Secondly, hybrid and quad-dominant meshes couldn't be solved because of a problem in .unv file format but if we compare the max 3D aspect ratios of meshes that have been generated, it can be said that using hybrid and quad-dominant meshes we got even less 3D aspect ratios than ones in tetrahedral meshes as you can see in tables 5-6-7.Therefore, the hybrid and quad-dominant meshes expect to converge.

6.3. ANSYS

6.3.1. Introduction to Ansys Fluent

Ansys Fluent software contains the broad physical modelling capabilities needed to model flow, turbulence, heat transfer, and reactions for industrial applications ranging from air flow over an aircraft wing to combustion in a furnace, from bubble columns to oil platforms, from blood flow to semiconductor manufacturing, and from clean room design to wastewater treatment plants. Special models that give the software the ability to model in-cylinder combustion, aeroacoustics, turbomachinery, and multiphase systems have served to broaden its reach. However, we are going to generate various types of meshes and use them to analyse the fluid flow that is inside of exhaust and intake ports of an engine.

6.3.2. Mesh generation in Ansys Fluent

Ansys fluent is an automatic mesh generator for 2D or 3D geometry. There is many mesh edit operator in Ansys Meshing. These operators are ;

- Methods : Users can control the algorithm and mesh types used to generate meshes on scoped entities. The methods can be selected: Automatic , Tetrahedrons , Hex Dominant , Sweep , Multizone , Cartesian , Layered Tetrahedrons
- Sizing : User can insert body, face or edge size with that operation. Also, software includes many sizing types such as element size or number of divisions and etc.
- Contact Sizing : Create elements (on bodies) of sizes similar to the face of a face-to-face or face-to-edge contact region.
- Refinement : Specify the maximum number of times you want an initial mesh to be refined. You can specify refinement controls for faces, edges, and vertices.
- Face Meshing : Enable the generation of a free or mapped mesh on selected faces.
- Copy Mesh : Copy the mesh an apply it the same geometric shape on project.
- Pinch : Allow removal of small features at the mesh level to enable generating better quality elements around those features.
- Infiltration : Apply infiltration to specific boundary.
- Mesh Numbering : Generate number for each element.
- Contact Match : Contact mesh which are intersect to each other. Meshes are connected related to tolerance value.

- Node Merge : Contact nodes which are near to each other. Nodes are connected related to tolerance value.

6.3.2.1. Tetrahedrons

Ansys meshing can generate tetrahedron meshes automatically. But the quality of these meshes is not enough to analyse so some improvements are needed. In this project infiltration technique will be used with tetrahedron meshes.

6.3.2.2. Cartesian

Ansys meshing can generate cartesian meshes automatically. All elements are hexahedral. The orthogonal quality of the meshes are high so we will try to decide cartesian mesh can be used to analyse port flow simulation.

6.3.2.3 Hexahedrons

Ansys meshing can generate hexahedron meshes for only non-complex geometries like cylinder or square prism with multizone and sweep method. For complex geometries, hex dominant method can be used to generate hex meshes with pyramids or tetrahedron meshes.

6.3.3. Meshing results

6.3.3.1. Hex-dominant mesh as a single body

Hex dominant mesh can be easily generated with Ansys meshing. The faces can be adjusted with face sizing. However, inner meshes can be bad so we have control inside of the body.

We selected method as hex dominant and body size adjusted to 0,001 m.

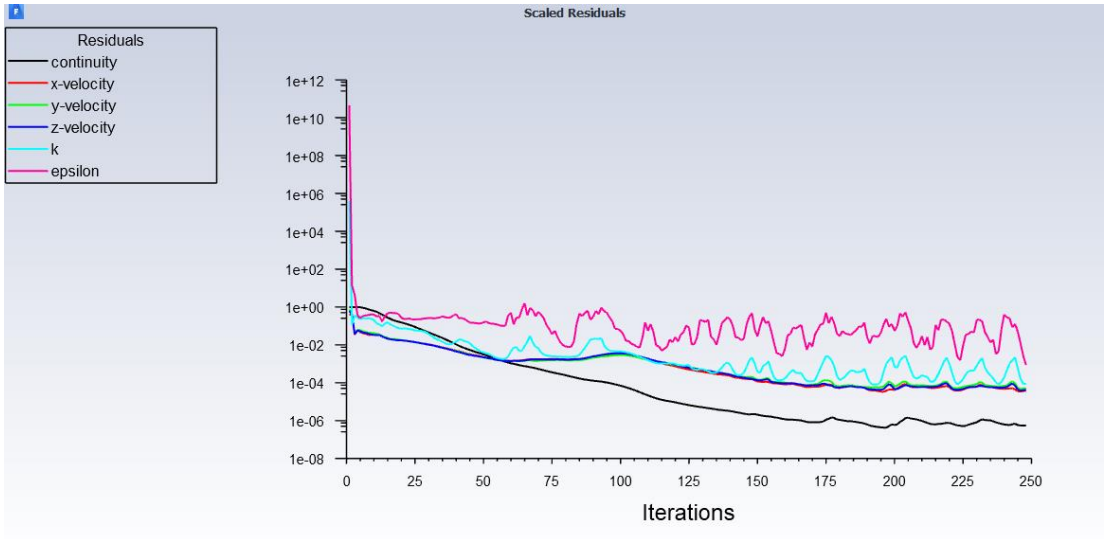

Figure 48. Residual results for single body mesh generation on exhaust port

The residual results converged for exhaust port. However, same methodology on intake port did not work. The residual results of intake port did not converge so we have to try a different methodology.

Multibody Methodology

Multibody modelling is for generate different methods for each body. It gives multi-method opportunities. However, we have to be careful when we are generating meshes. For multibody modelling, we have to match the nodes of meshes for all bodies. So, some connection operations can be needed.

6.3.3.2. Hex-dominant as two-body.

We have mainly two body for that methodology. One of it is square prism and other one is the remaining part of the main body. For this methodology, methods are applied partially for two body and meshes are connected after generation.

- **Intake Port**

For Prism: We applied multizone modelling for that part. We applied the edge sizing as 60 for square part and 133 for the height of the prism as number of divisions. Also, face meshing is applied to the inner faces. Face meshing gives equal rectangular mesh faces.

Remaining Part of Main Geometry: Hex dominant method is used for this part. Edge sizing is applied as 60 and 133 as number of divisions and body sizing is applied as 0,001 m. Face meshing is also applied for faces which are duplicate with prism part.

The behaviour of all mesh operations is hard.

Mesh Connection: Mesh connection is done with hard edge sizing and face meshing so We don’t have to apply any method for that operation.

- **Exhaust Port**

For Prism: We applied multizone modelling for that part. We applied the edge sizing as 40 for square part and 75 for the height of the prism as number of divisions. Also, face meshing is applied to the inner faces. Face meshing gives equal rectangular mesh faces.

Remaining Part of Main Geometry: Hex dominant method is used for this part. Edge sizing is applied as 40 and 75 as number of divisions and body sizing is applied as 0,001 m. Face meshing is also applied for faces which are duplicate with prism part.

The behaviour of all mesh operations is hard.

Result of Two-Body Methodology

PORT TYPE	SIZING	GENERATION TIME	ORTHOGONAL QUALITY	NUMBER OF ELEMENTS	CONVERGED OR NOT	SOLUTION TIME
Intake as 2 body	0,001	48 min	min: 2.51*10^-7 max: 1 average: 0,8625	1942417	Converged to 10^-4	38 min
Exhaust as 2 body	0,001	30 min	min: 2.73*10^-7 max: 1 average: 0,7977	1357762	Converged to 10^-4	36 min
Intake	0,001	55 min	min: 7.67*10^-6 max: 1 average: 0,8237	2054048	Not converged	-
Exhaust	0,001	38 min	min: 5.56*10^-12 max: 1 average: 0,7553	1455708	Converged to 10^-3	10 hours 10 min

Table 9. Ansys 2-body mesh generation results

We have created a prism to increase mesh quality and to decrease the number of elements. We did not face with any problem. And also result is converged to 10^{-4} for intake and exhaust port so this methodology is available and acceptable for mesh generation on intake and exhaust ports.

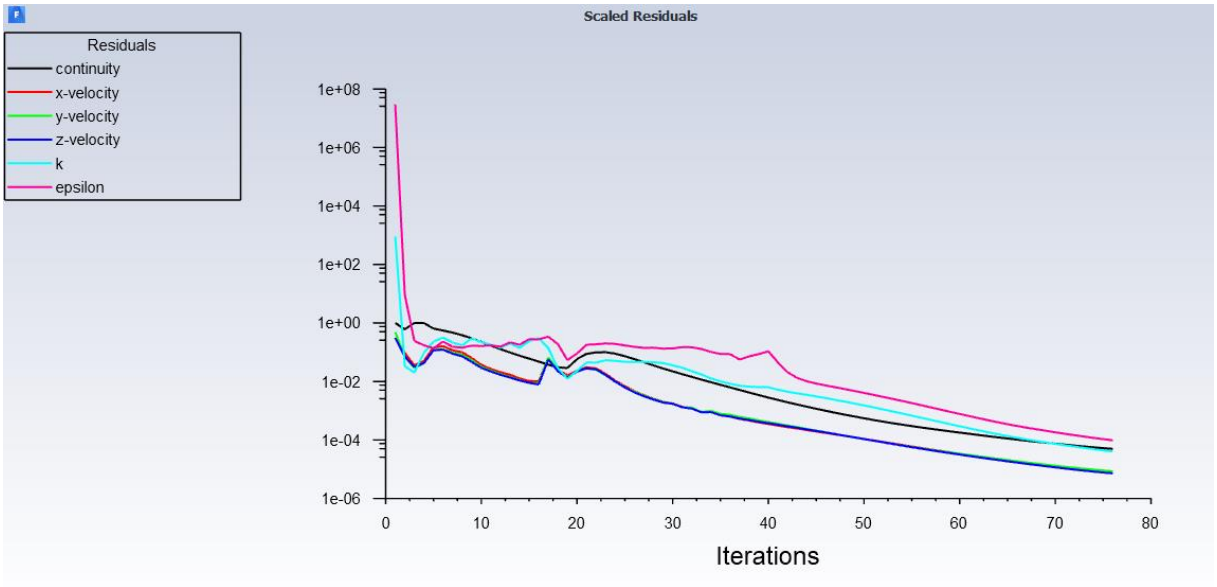

Figure 49. Residual results for two body mesh generation on intake port

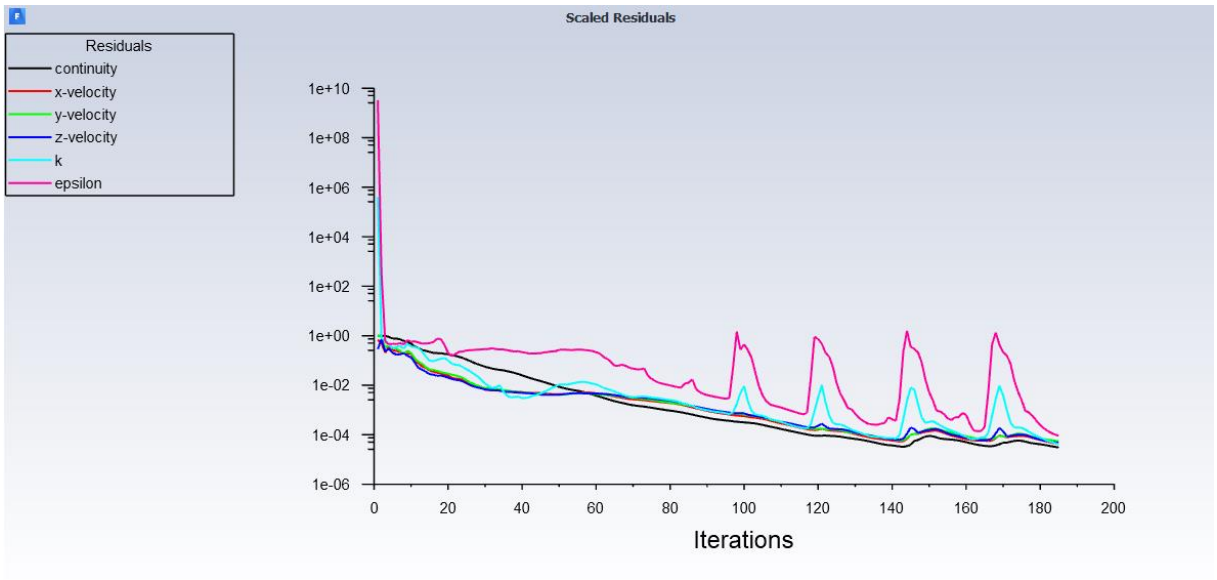

Figure 50. Residual results for two body mesh generation on exhaust port

6.3.3.3. Hex-dominant mesh as six-body

We have mainly six body for that methodology. One of it is square prism, four of it is outer non-complex bodies and the last one is the remaining part of the main body. For this methodology, 3 main methods are applied. These are prism parts, four non-complex parts and remaining part of the main body.

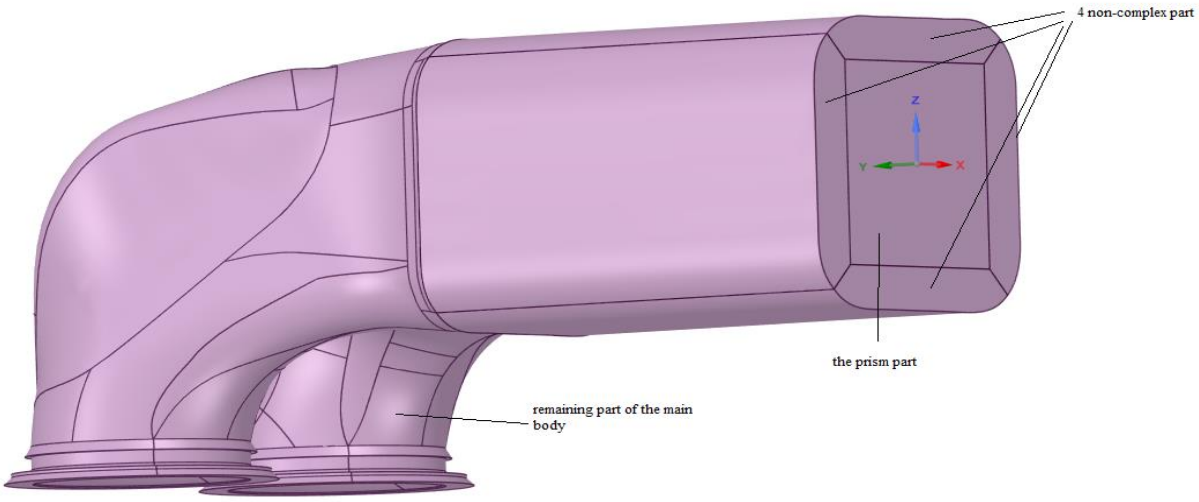

Figure 51. The body names

Before starting mesh generation, the faces should duplicate, So the main face should be splitted:

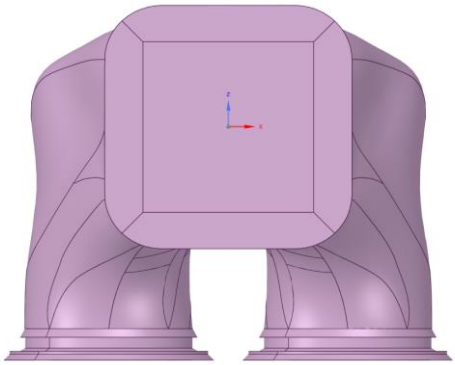

Figure 52. Face splitting according to body contacts.

For Prism: We applied multizone modelling for that part. We applied the edge sizing as 78 for square part ad 150 for the height of the prism as number of divisions. Also, face meshing is applied to the inner faces. Face meshing gives equal rectangular mesh faces.

For Four Non-complex Geometries: We applied sweep method for these four geometries with manual source. Edge sizing is applied to duplicate edges. The number of divisions is the same with master geometry. These are 78, 150 and 10.

Remaining Part of Main Geometry: Hex dominant method is used for this part. Edge sizing is applied as 78 and 10 as number of divisions and body sizing is applied as 0,001m. Face meshing is also applied for faces which are duplicate with prism and 4 non-complex parts.

The behaviour of all mesh operations is hard.

Mesh Connection: Mesh connection is made with node merge command. The Slave and master faces are selected as shown in figure 39. The tolerance value is applied as 0,0005 m. The connection is completed.

Results of six body methodology

Orthogonal quality results:

Minimum = 9.54*10^-10

Maximum = 1

Average = 0.89243

The more the mesh have body number, the best orthogonal quality have been reached. However, the meshes do not match so the node merge is applied.

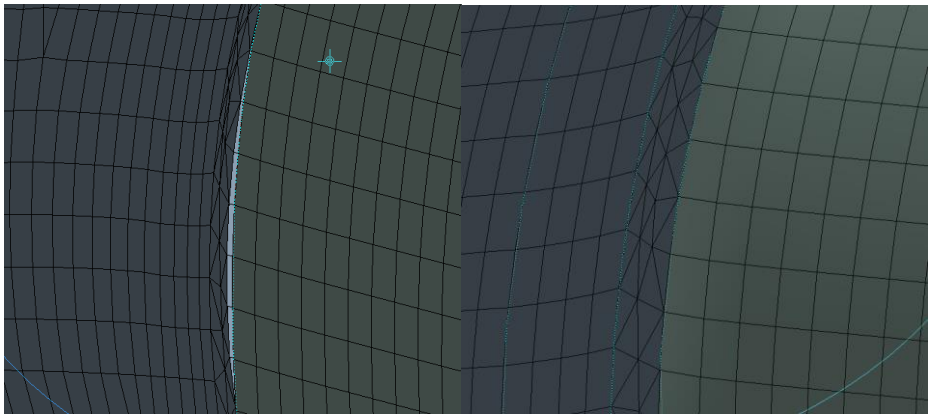

Figure 53. Before (left) node merge and after (right)

But we have faced with an Ansys update error. Because of this error, mesh is not updated and did not used to solve. After trial and error method, we defined the source of error as node merge. Mesh is updated without node merge but an error occurred at fluent part this time. We have spent a lot of time to solve the errors but we could not so this methodology is not available for port mesh generation.

Comparison of all hex-dominant methodologies and results

The mesh quality is worst for one body meshing. Solution and mesh generation time takes longer time the other methodologies. Also, it is not solved for all ports so this methodology can be used but it does not work for all ports.

The quality of mesh of 6-body meshing is better than the 2-body mesh generation and the number of elements of 6-body mesh generation is lower than the 2-body mesh generation. But this is not meaning the 6-body mesh generation methodology is better. In theoretically, the 6-body mesh generation should be better but we did not examine it

because of the update error. In our opinion, it should be better. To prove that the error should be solved in software.

6.3.3.4. Cartesian mesh

We tried to generate cartesian meshing on intake port. If the solution converges, we will try on exhaust port with same methodology.

SIZING	GENERATION TIME	ORTHOGONAL QUALITY	NUMBER OF ELEMENTS	CONVERGED OR NOT	SOLUTION TIME
0,001	10 min	max: 1 min: 0.00871 average: 0.9866	7510568	not solved	-
0,0015	not generated	-	-	-	-
0,002	2 min	max: 1 min: 0.00825 average: 0.9867	936091	not converged	3 hours
0,003	1 min	max: 1 min: 0.00967 average: 0.9744	130355	not solved	-

Table 10. Ansys cartesian mesh results

Cartesian mesh results

The quality of cartesian meshes is very high. It generates all hexahedron cells. However, the main shape of the geometry changes while hexahedron meshes are generating. The changes are not big problem when we are working with small element size.

Also, Ansys meshing does not allow to generate element size as 0,0015 m. Generated mesh with 0,002 m element size did not converge. Generated mesh with 0,001 m element size did not solve. Because my computer is not powerful to solve with 7 million cells. The number of generated mesh with 0,003 m element size is very low for turbulence modelling.

The continuity and epsilon residuals did not converge to 10^{-3} for intake port when the element size is 0,002 m.

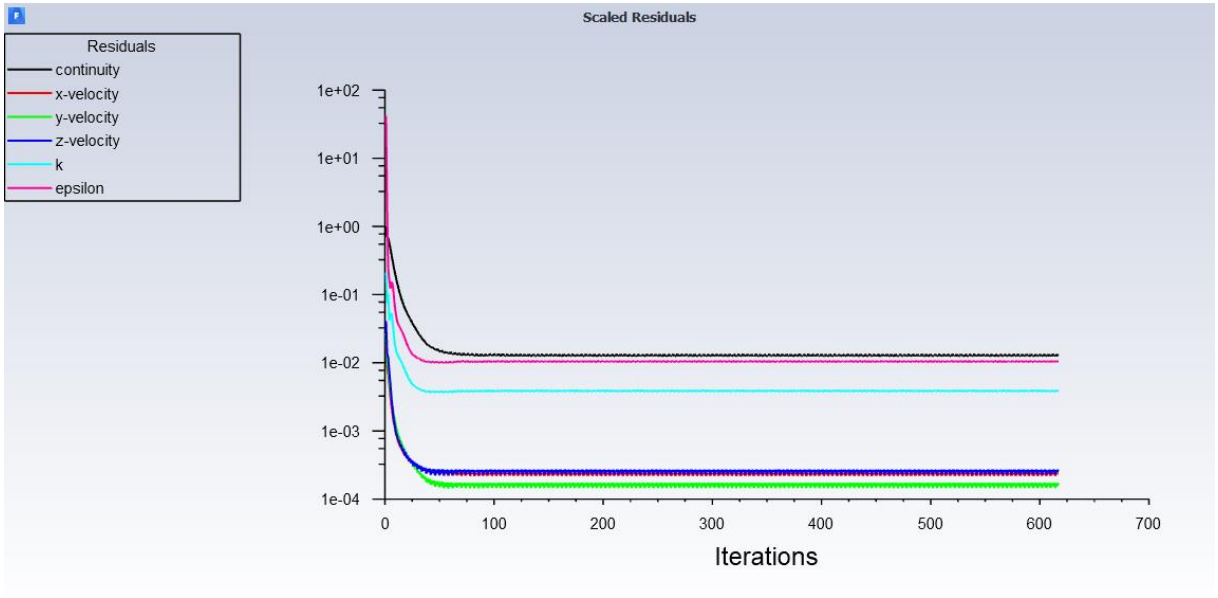

Figure 54. The residual result of 0,002 m element size cartesian meshes

Therefore, this methodology is not available for mesh generation on intake and exhaust ports.

6.3.3.5. Tetrahedral mesh

Tetrahedron meshing is generated automatically. Only body size is given by user and infiltration is generated on non-complex part of the geometry. We will use exhaust port for tetrahedron methodology. If it works, it will be tried on intake port.

SIZING	GENERATION TIME	ASPECT RATIO	NUMBER OF ELEMENTS	CONVERGED OR NOT	SOLUTION TIME
0,001	7 min	max: 15,88 min: 1.1577 average: 1.8664	10249086	not solved	-
0,0015	3 min	max: 18.8 min: 1.1582 average: 1.9221	3061278	not converged	6 hours
0,002	1 min	max: 28,463 min: 1,1593 average: 1.883	1276073	not converged	4 hours
0,003	35 sec	max: 235.85 min: 1.1642 average: 1.9875	638656	not converged	1 hour

Table 11. Ansys tetrahedral mesh results

6.3.4. Visual representation of meshes

6.3.4.1. Hex-dominant meshes

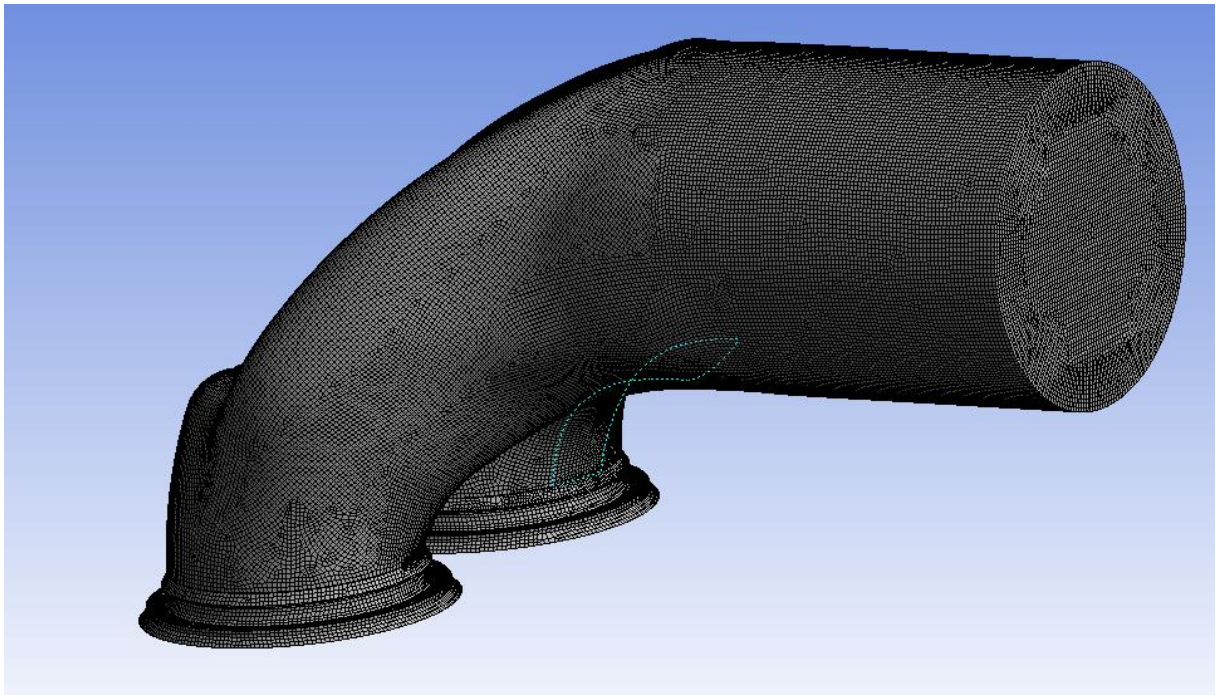

Figure 55. Hex-dominant mesh result of six body mesh generation

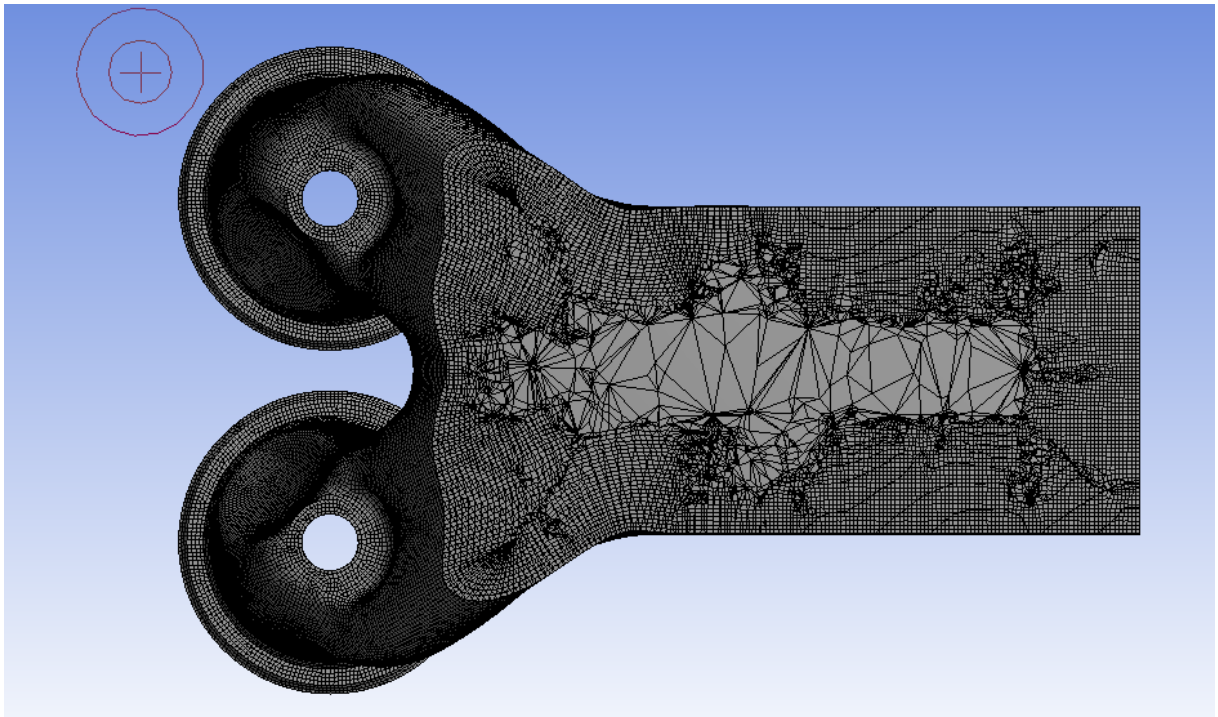

Figure 56. Hex-dominant inside mesh result of single body mesh generation

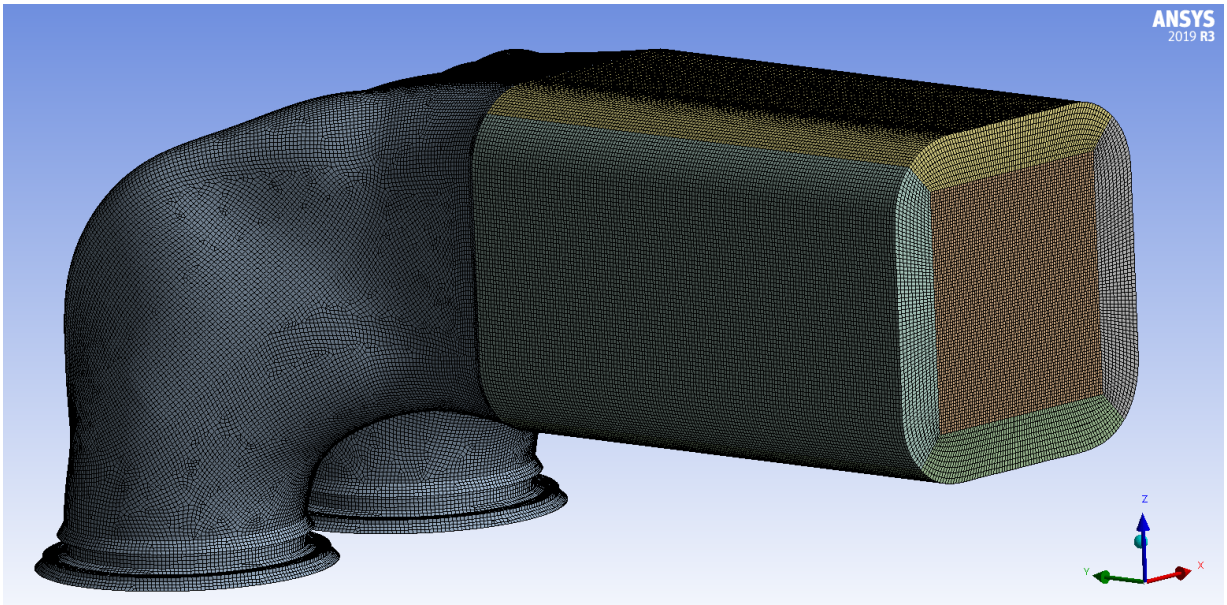

Figure 57. Hex-dominant mesh result of six body mesh generation

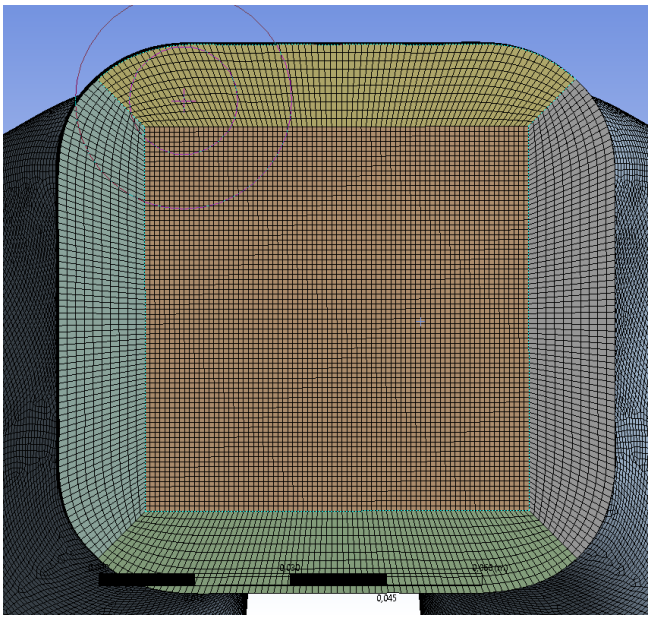

Figure 58. Hex-dominant mesh result of six body mesh generation front

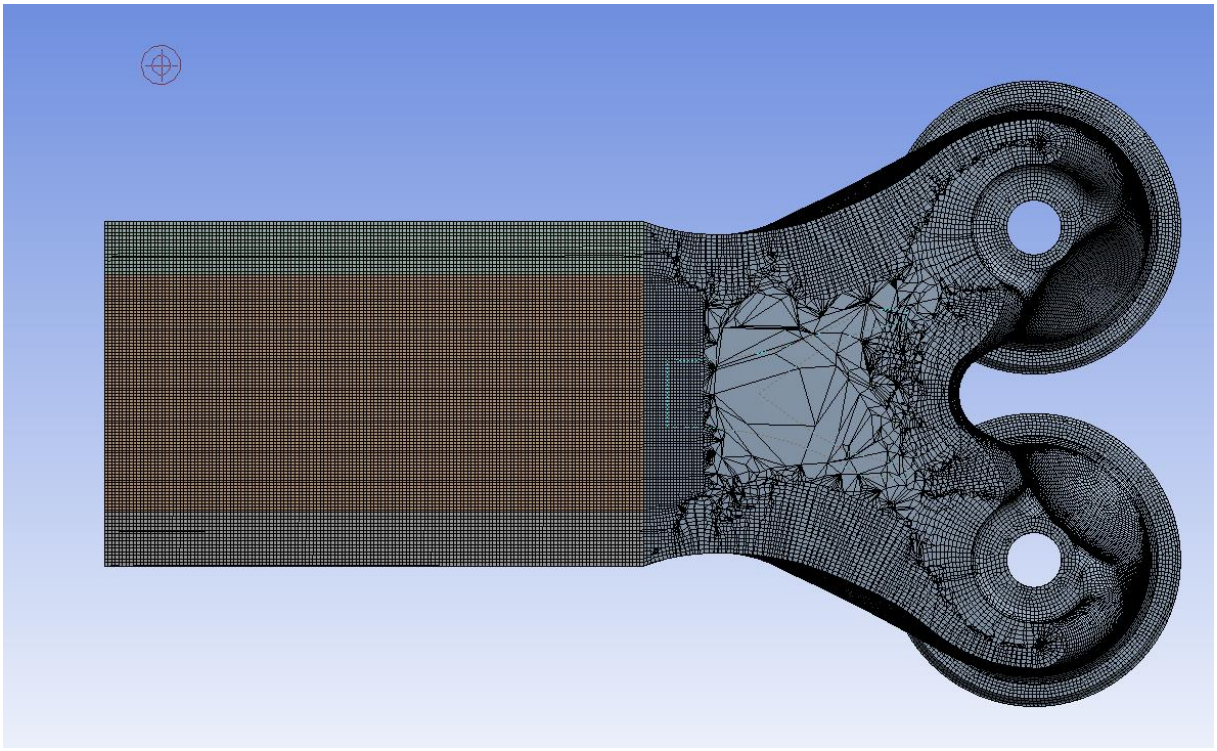

Figure 59. Hex-dominant inside mesh result of six body mesh generation

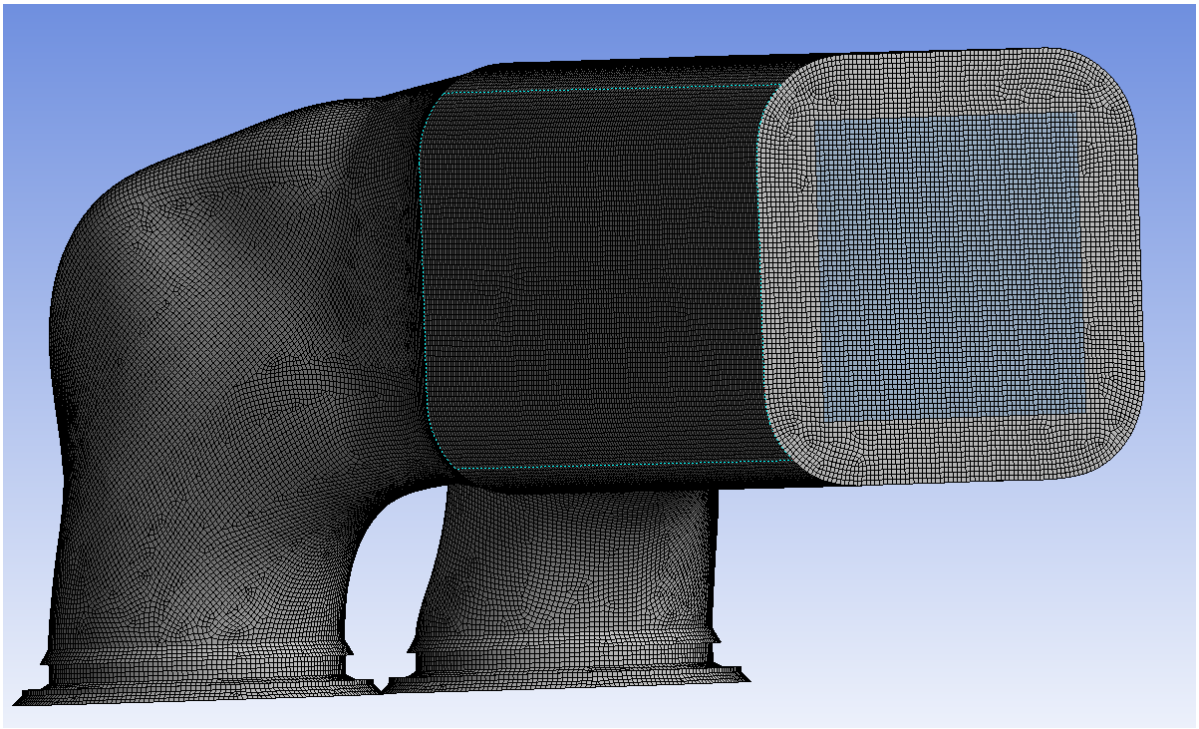

Figure 60. Hex-dominant mesh result of two body mesh generation for intake port

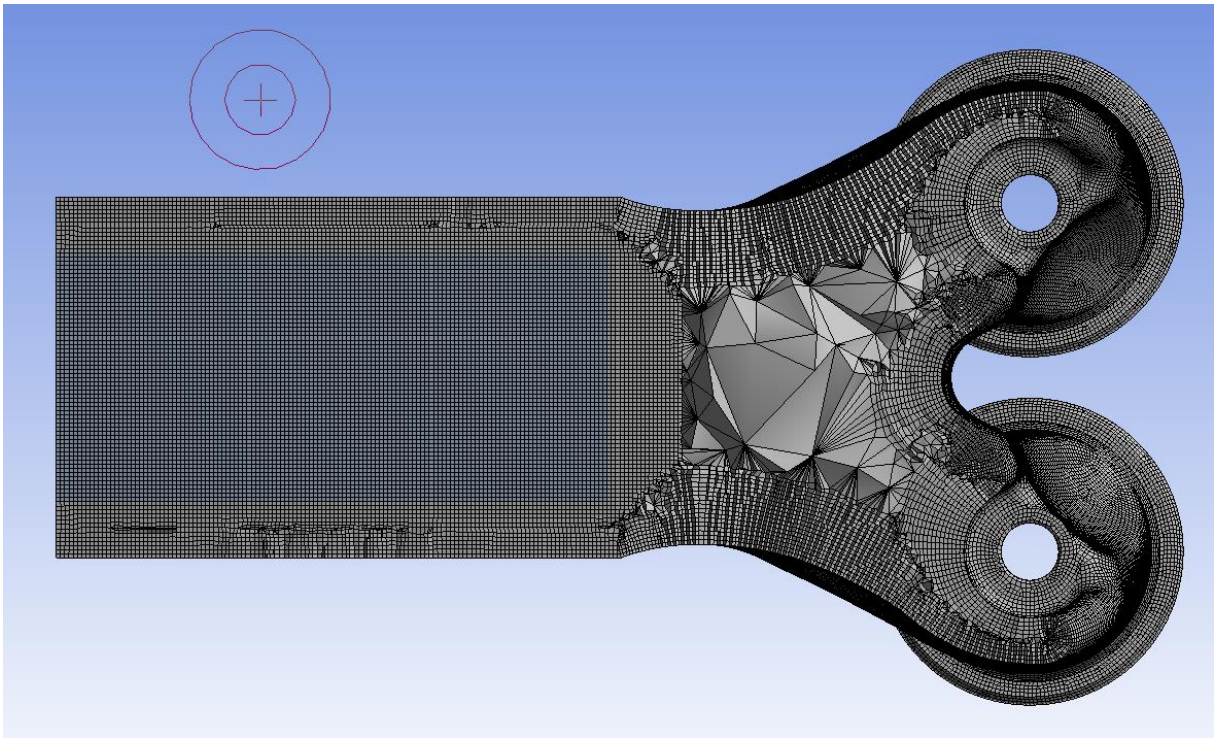

Figure 61. Hex-dominant inside mesh result of two body mesh generation for intake

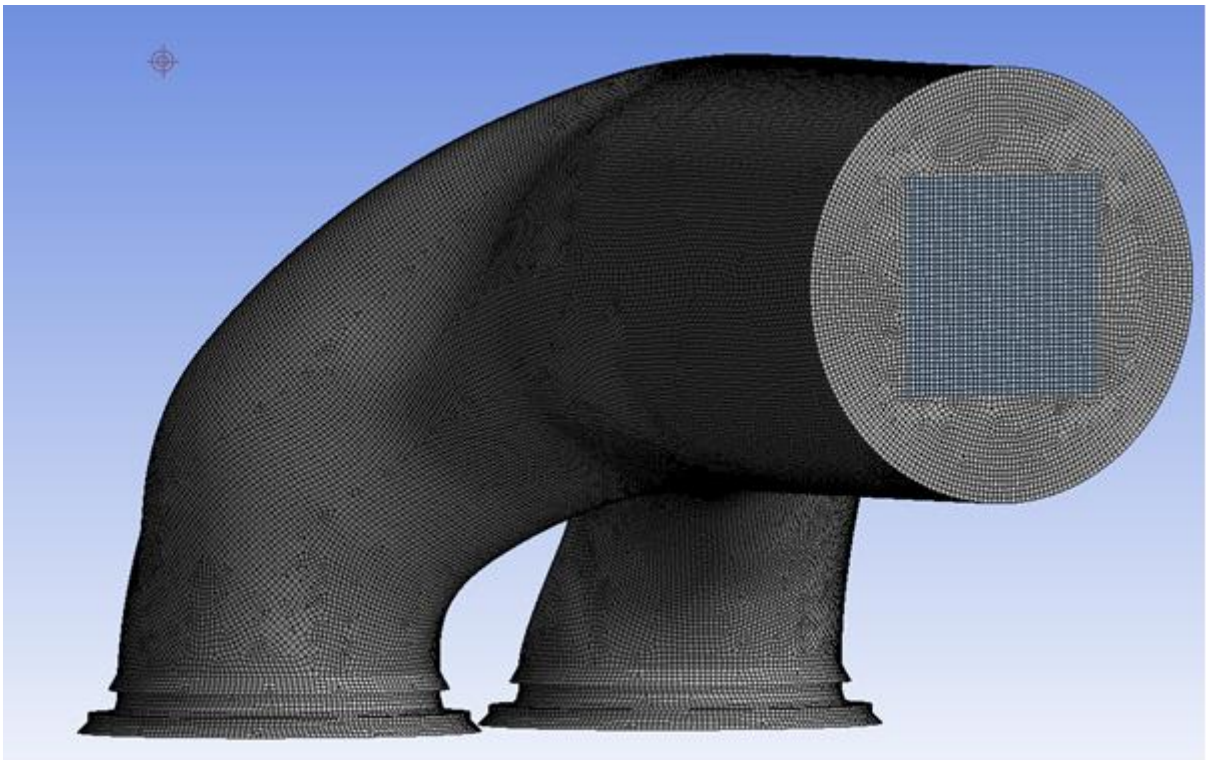

Figure 62. Hex-dominant mesh result of two body mesh generation for exhaust port

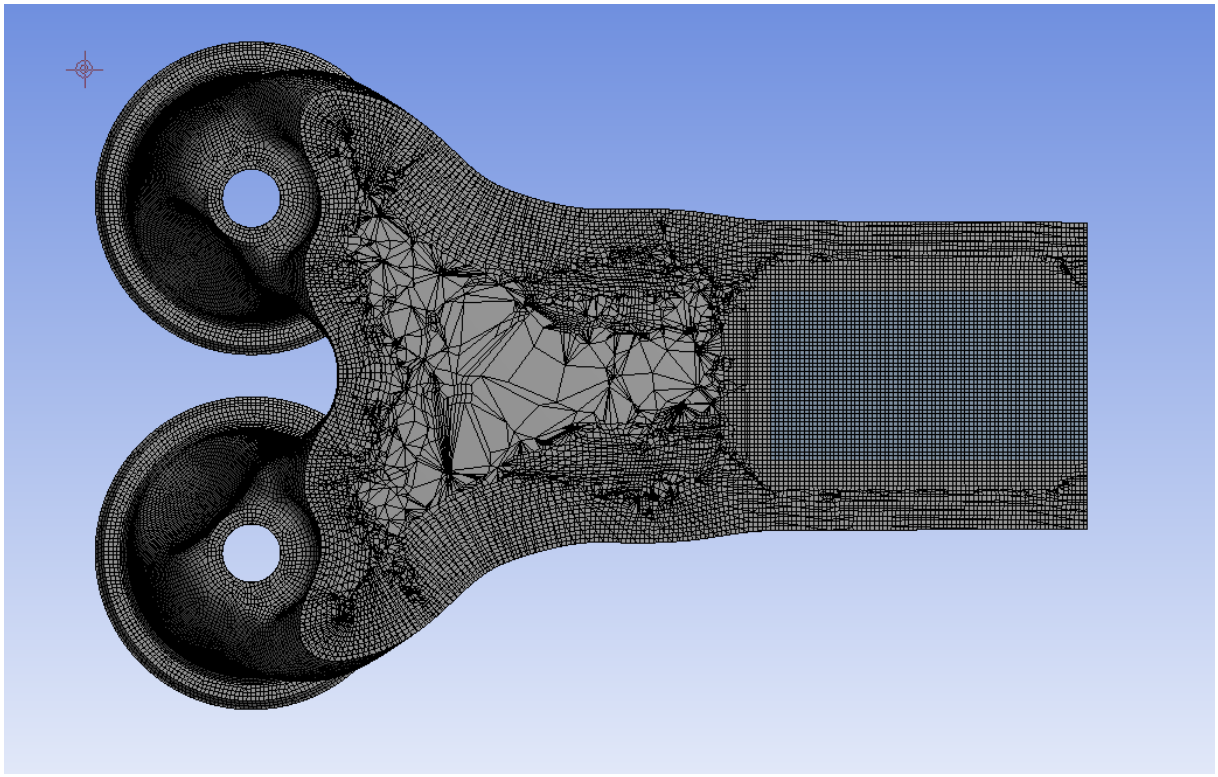

Figure 63. Hex-dominant inside mesh result of two body mesh generation for exhaust port

6.3.4.2. Visual representation of cartesian meshes

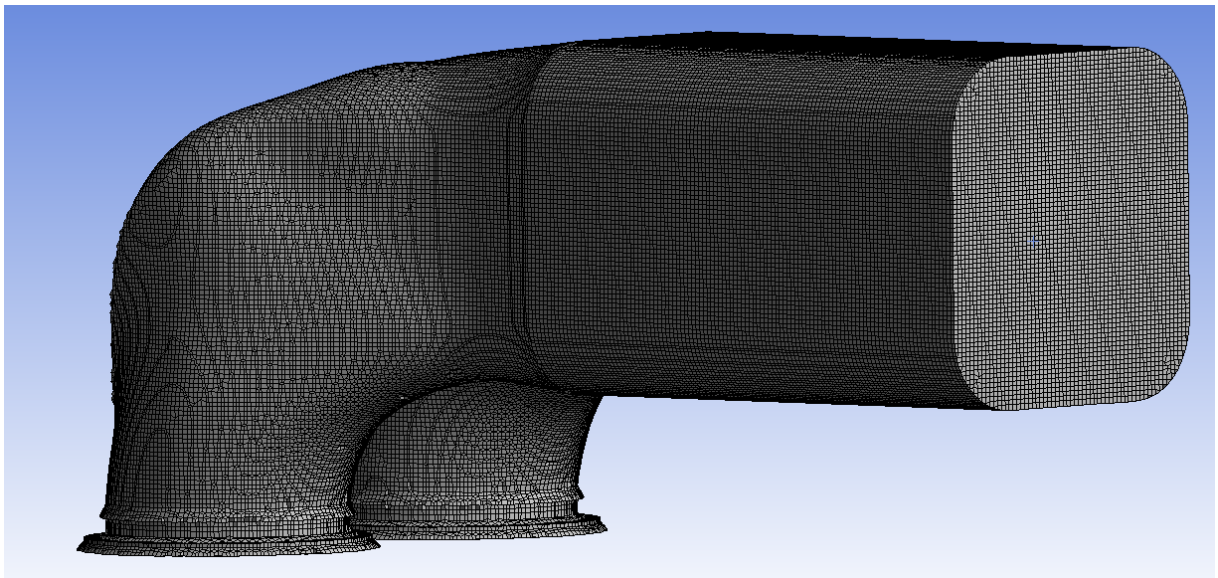

Figure 64. Mesh result of cartesian method for intake port

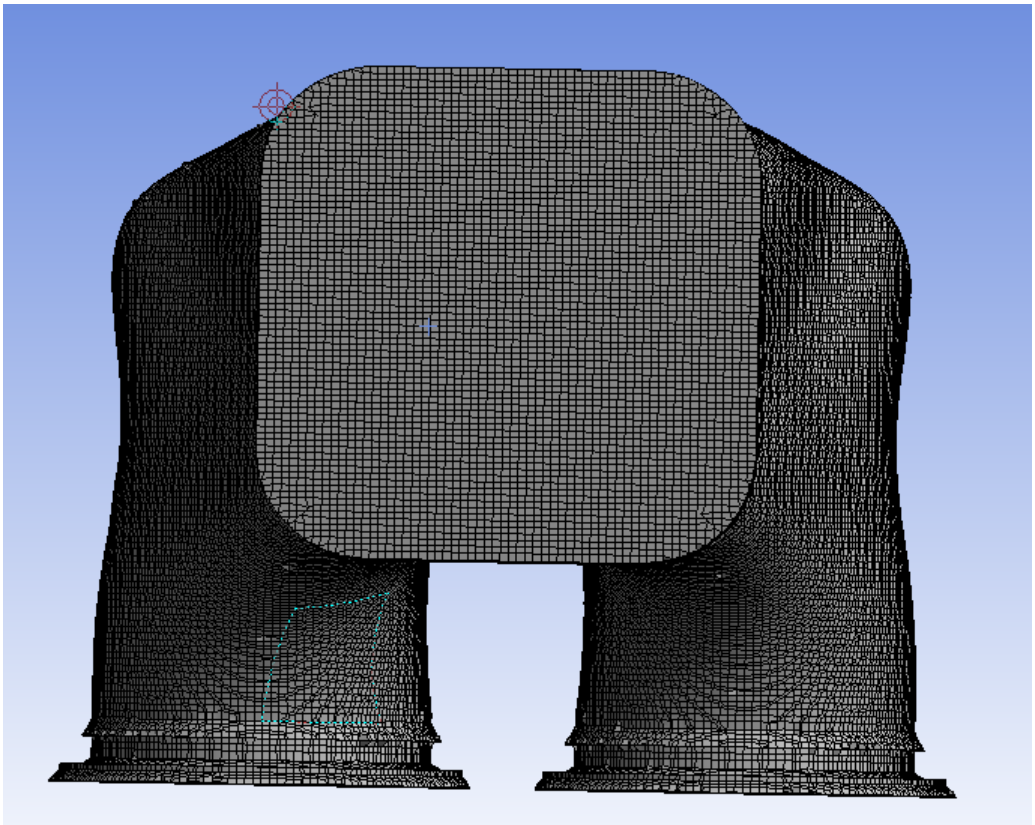

Figure 65. Front mesh result of cartesian method for intake port

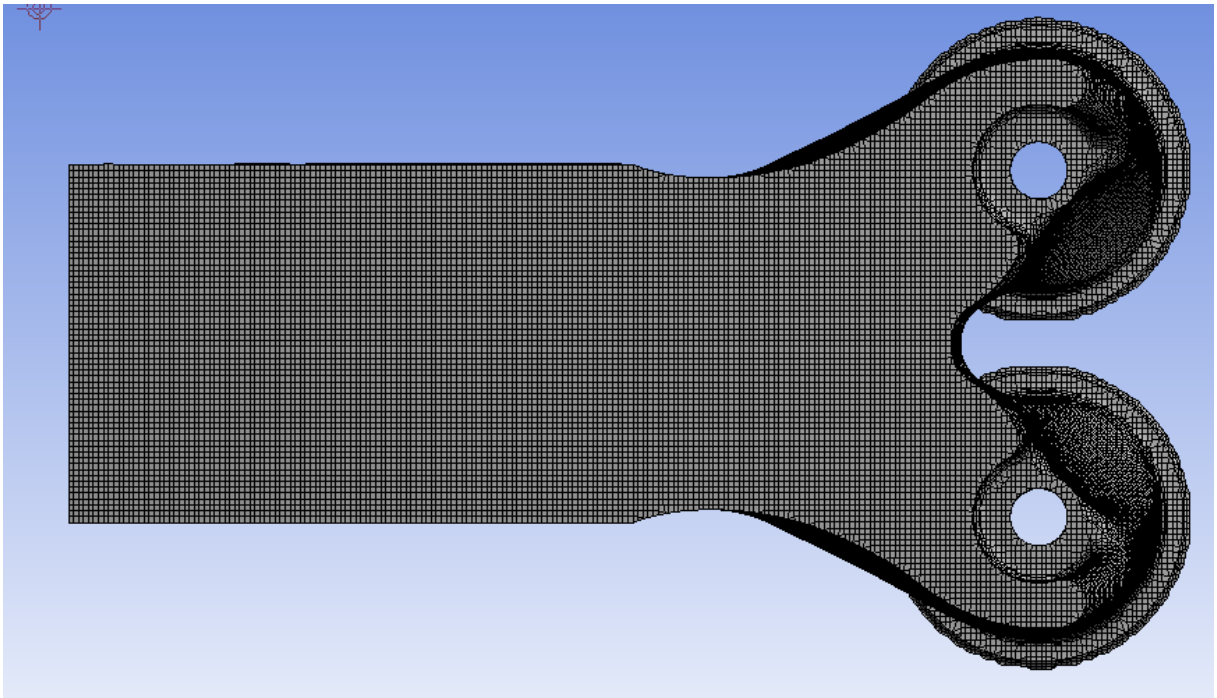

Figure 66. Inside mesh result of cartesian method for intake port

6.3.4.3. Visual representation of tetrahedral meshes

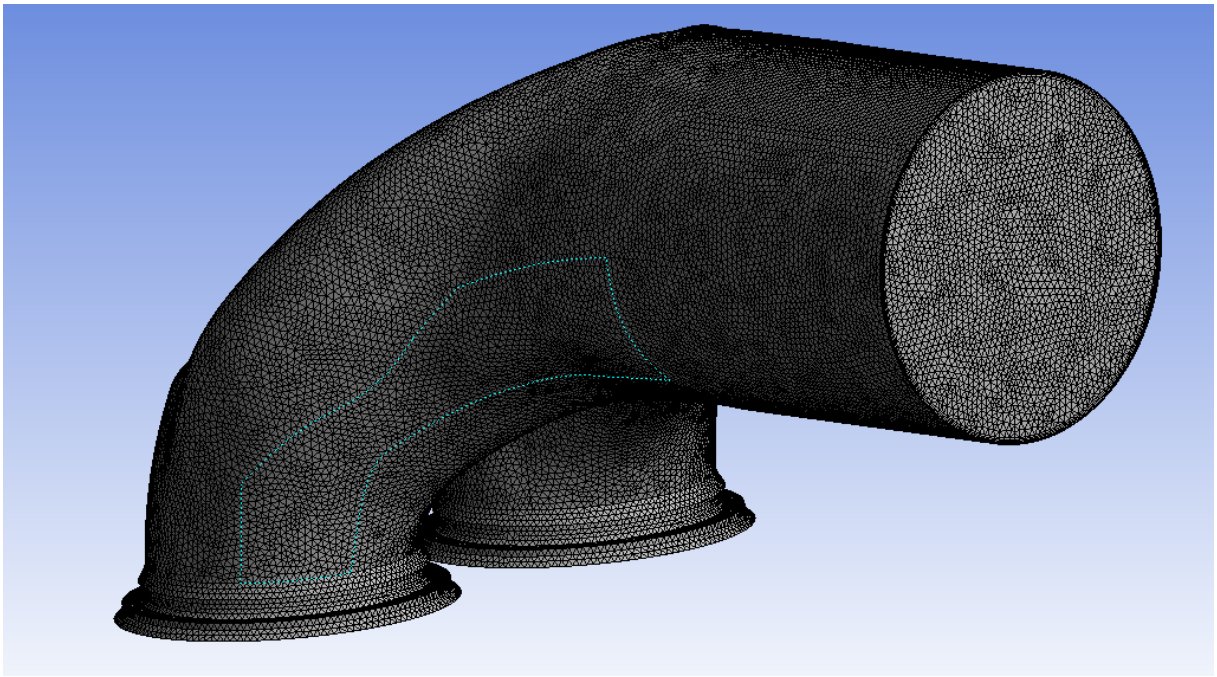

Figure 67. Tetrahedron mesh result of two body mesh generation for exhaust port

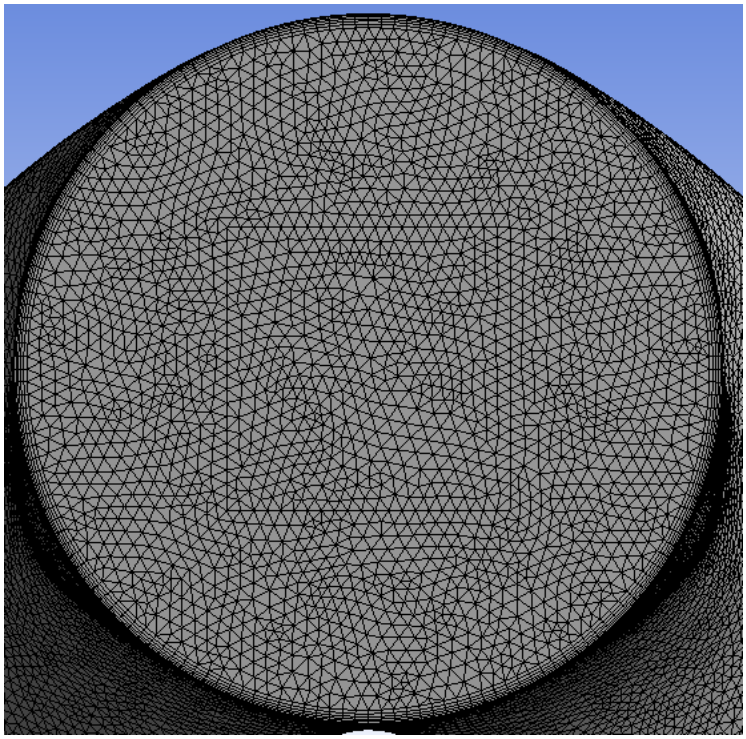

Figure 68. Tetrahedron mesh result of two body mesh generation for exhaust port front

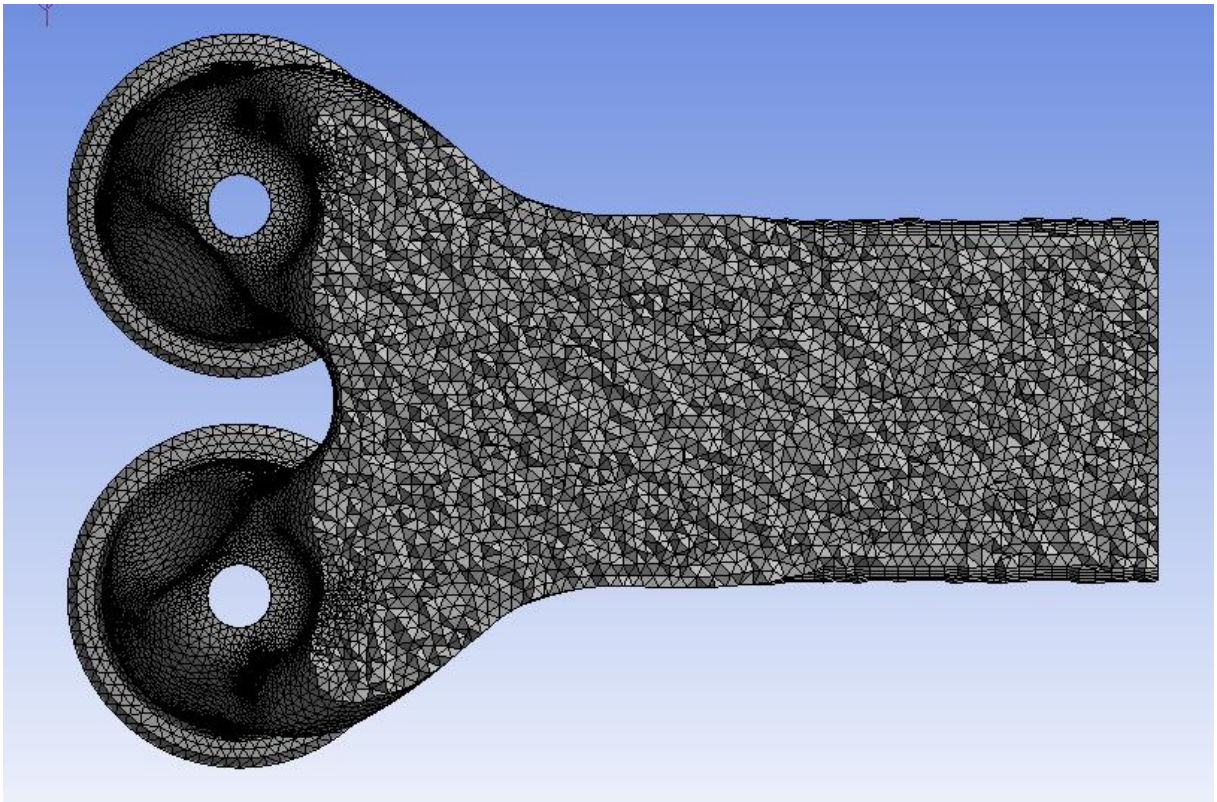

Figure 69. Tetrahedron mesh inside result of two body mesh generation for exhaust port

6.3.5. Interpretation of meshes

Continuity equation do not converge when we are using tetrahedron meshes with body sizing as 0,0015, 0,002, 0,003 so the tetrahedron mesh generation methodology is not useful for mesh generation in exhaust port.

Continuity equation also do not converge when we are using cartesian meshes. Cartesian meshing do not have 0,0015 sizing adjustment. The orthogonal quality of cartesian meshes is very high. Maybe for different port 0,002 or 0,001 m body sizing can be useful and solution can converge. For this intake port, 0,001 m body sizing generate over 7 million hexahedron cells and my computer is not powerful to solve it so the cartesian mesh generation methodology is not useful for mesh generation in intake port.

For hex dominant methodology, we have faced with an error for 6 body mesh generation. The source of error is node merging. If we do not use node merge, we can update the meshes. However, the nodes do not match without node merge and we have a face error in fluent section. This methodology can be best mesh generation for Ansys fluent after solving node merge error. For two body methodology, we obtain cube cells at the prism part. The number of cells and mesh generation time decreased and the quality of mesh

increased respect to single zone methodology. Also, the solution of this methodology for both ports is worked.

7.CONCLUSION

Grid generation is the most time consuming and difficult part of the CFD analysis process. Grid generation often consumes up to 80 percent of the labour hours for a CFD project and is the largest controllable influence on the accuracy of the analysis. In this project, various mesh generation methods and mesh types have been performed to have quality mesh for intake and exhaust port of a V16 engine. This paper presents a comprehensive methodology in mesh generation for different meshing programs such as OpenFOAM, Salome and Ansys. Each meshing program has own advantages and disadvantages over others.

First of all, OpenFoam which is an open source program and easy to access, was used to create hexahedral mesh for both geometries. SnappyHexMesh by OpenFOAM, was shown in respect of capability to generate fine structured hexahedral mesh. Three of meshes are created for one of the geometries, six meshes in total. Maximum aspect ratio of 3D cells was be able to reduced up to 3.83 for the intake geometry and 3.67 for the exhaust geometry. Therewithal, it is succeeded to restrict the maximum skewness as 1 for both geometries. While getting coarse mesh initially, when we changed the parameters, it can be produced the better meshes which have enough quality to converge in solving for CFD simulation.

Secondly, Salome which is another open source meshing program is used. Three mesh generation methods performed for both geometries such as tetrahedral, quad-dominated and hybrid mesh respectively. Salome really suitable for unstructured tetrahedral mesh for intake and particularly exhaust port. 71256 cells were enough to create good quality (convergent) mesh for exhaust port. On the other hand, 773919 cells were enough for intake port. That cell number difference may be due to complexity of the geometry. However, only tetrahedral mesh could be imported into Ansys mesh solver. Because, unv file extension which doesn't have ability to storage the pyramid cells, was used for mesh exporting into the solver. Quad-dominant and hybrid meshes include pyramid cells. The convergence of hybrid meshes is expecting, since even less maximum 3D aspect ratio has been reached than tetrahedral mesh while generating hybrid meshes in Salome. But the

same situation is not valid for quad dominated mesh which ended up high 3D aspect ratio and insufficient to converge in solver.

Thirdly, Ansys which is a commercial software unlike Salome and OpenFOAM, have been used to generate tetrahedral, hex-dominant and cartesian mesh. Orthogonal quality of cartesian mesh for both ports was qualified. However, the continuity equation didn't converge for both ports so cartesian mesh is not useful. For tetrahedral meshing, various meshes have been created by varying body sizing and infiltration parameters but none of them was be able to converge which means good quality tetrahedral mesh was not reached. On the other hand, hex-dominant mesh was examined in three categories. These are single-body, two-body and six body. Two-body methodology have worked for both ports. Single-body methodology have worked only for exhaust port. Whereas, six-body methodology wasn't be able to tried because of node matching issue. Although, all the nodes connect, Ansys did not update the mesh. In theoretically, six-body modelling had the best average orthogonal quality. As a result, best methodology for Ansys is two-body mesh generation methodology.

If we need to compare these meshing programs according to different meshing and geometry cleaning-up capabilities, firstly we have to say that Ansys Spaceclaim is the best option for geometry repair than other two.

- For tetrahedral mesh, Salome has more capabilities than others.
- Also, Salome is more successful in hybrid mesh than Ansys because it can connect meshes which belongs different geometries automatically.
- For hexahedral mesh, Ansys worked well in two-body methodology and residuals converged up to 10^{-4} for both port geometries. On the other hand, hexahedral meshes which we got by snappyHexaMesh have good mesh indexes that can get enough solver to converge.
- For quad-dominated mesh, it can be generated only in Salome. When maximum 3D aspect ratio is considered, this mesh type is not suitable for our geometries.

In conclusion, the best two methodology for intake and exhaust port are tetrahedral mesh generation methodology in Salome and two-body mesh generation methodology in Ansys. For a good quality tetrahedral mesh in Salome, it takes 25.5 minutes (mesh generation time + solution time) for intake port geometry and it takes 5.5 minutes for exhaust port geometry. On the other hand, For a good quality hex-dominant mesh in

Ansys, it takes 1 hour and 26 minutes for intake port geometry and 1 hour and 6 minutes for exhaust port geometry.

REFERENCES

1. A. R. Ingraffea, J. B. Cavalcante-Neto, L. F. Martha, 'An Algorithm for Three-Dimensional Mesh Generation for Arbitrary Regions with Cracks', Engineering with Computers, ·May 2001
2. J. F. O'Connor, N. R. McKinley, 'CFD Simulations of Intake Port Flow Using Automatic Mesh Generation: Comparison with Laser Sheet, Swirl and LDA Measurements for Steady Flow Conditions', International Congress and Exposition Detroit, Michigan February 23-26, 1998
3. https://en.wikipedia.org/wiki/OpenFOAM
4. https://cfd.direct/openfoam/user-guide/v6-snappyhexmesh/
5. https://en.wikipedia.org/wiki/Mesh_generation
6. Castillo, J.E. (1991), "Mathematical aspects of grid Generation", Society for Industrial and applied Mathematics, Philadelphia
7. George, P.L. (1991), Automatic Mesh Generation : http://www.bakker.org/dartmouth06/engs150/07-mesh.pdf
8. Andre Bakker, Applied Computational Fluid Dynamics ,Lecture 7 -meshing
9. Ö. Yıldız,(2001) 'IMPLEMENTATION OF MESH GENERATION ALGORITHMS'
10. O. C. Zienkiewicz, The Finite Element Method, McGraw-Hill, New York,1983
11. https://en.wikipedia.org/wiki/Types_of_mesh
12. N. Ray, D. Sokolov, M. Reberol, F. Ledoux, B. Levy , (2017) ,Hexahedral Meshing : Mind the Gap!
13. https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/node164.html
14. V. D. Liseikin, Grid GenerationMethods, second edition
15. H. Jasak. Error Analysis and estimation for the finite volume method with applications to fluid flows. Ph.d thesis, Imperial College of Science, Tecnology and Medicine, London, 1996.
16. J. H. Ferziger and M. Peric. Computational Methods for Fluid Dynamics. Springer, 2002.

17. T. Lucchini, R. Torelli, G. D'errico , Automatic Mesh Generation for Full -Cycle CFD Modeling of IC Engines: Application to the TCC Test Case
18. http://www.spaceclaim.com/en/Solutions/GeometryRepair.aspx
19. Trelis 16.5 User Documentation: https://csimsoft.com/help/item/clean_up/clean-up.htm
20. https://docs.salome-platform.org/7/gui/SMESH/constructing_meshes_page.html
21. Brezina, Jiri, 1980, Sedimentological interpretation of errors in size analysis of sands; 1st European Meeting of the International Association of Sedimentologists, Ruhr University at Bochum, Federal Republic of Germany, March 1980.
22. Hughes, Roger "Civil Engineering Hydraulics," Civil and Environmental Dept., University of Melbourne 1997, pp. 107–152
23. Jermy M., "Fluid Mechanics A Course Reader," Mechanical Engineering Dept., University of Canterbury, 2005, pp. d5.10.
24. Basse, N.T. (2019), "Turbulence intensity scaling: A fugue", Fluids, vol. 4, 180.
25. https://www.engineeringtoolbox.com/air-absolute-kinematic-viscosity-d_601.html
26. rheoheat.se/b13_heat.html
27. W. W. Pulkrabek, Engineering Fundamentals of the Internal Combustion Engine, second edition, Published by Pearson (June 10th 2003)
28. https://en.wikipedia.org/wiki/Turbulence_modeling
29. https://en.wikipedia.org/wiki/K-epsilon_turbulence_model

APPENDICES

OpenFOAM Dict file codes of Mesh3 for snappyHexMesh

- **blockMeshDict:**

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       dictionary;
   object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
convertToMeters 1;
xmin -80;
xmax 80;
ymin -20;
ymax +300;
zmin -100;
zmax 50;
//xcells 1600;
//ycells 1000;
//zcells 1500;
lx #calc "$xmax - $xmin";
dx 0.1;
xcells #calc "abs($lx/$dx)";
ly #calc "$ymax - $ymin";
dy 0.1;
ycells #calc "abs($ly/$dy)";
lz #calc "$zmax - $zmin";
dz 0.1;
zcells #calc "abs($lz/$dz)";
vertices
(
```

```
    ($xmin $ymin $zmin)
    ($xmax $ymin $zmin)
    ($xmax $ymax $zmin)
    ($xmin $ymax $zmin)
    ($xmin $ymin $zmax)
    ($xmax $ymin $zmax)
    ($xmax $ymax $zmax)
    ($xmin $ymax $zmax)
);
mergePatchPairs
(
);
// ************************************************************************* //
```

- **meshQualityDict :**

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      meshQualityDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
#includeEtc "caseDicts/mesh/generation/meshQualityDict"
//- Maximum non-orthogonality allowed. Set to 180 to disable.
maxNonOrtho 60;
//- Max skewness allowed. Set to <0 to disable.
maxBoundarySkewness 1;      //original 20
maxInternalSkewness 1;
//- Max concaveness allowed. Is angle (in degrees) below which concavity
maxConcave 80;
minVol 1e-13;
minTetQuality -1e30; //for best layer insertion
```

```
//- Minimum face area. Set to <0 to disable.
minArea -1;
//- Minimum face twist. Set to <-1 to disable. dot product of face normal and face centre triangles normal
minTwist 0.02;
//- Minimum normalised cell determinant. This is the determinant of all
minDeterminant 0.001;
//- Relative position of face in relation to cell centres (0.5 for orthogonal mesh) (0 -> 0.5)
minFaceWeight 0.05;
//- Volume ratio of neighbouring cells (0 -> 1)
minVolRatio 0.01;
//- Per triangle normal compared to average normal. Like face twist but now per (face-centre decomposition) triangle. Must be >0 for Fluent
minTriangleTwist -1;
// ************************************************************************* //
```

- **snappyHexMeshDict :**

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       dictionary;
   object      snappyHexMeshDict;
}
// * * * * * * * * * * * * ** * * * * * * * * * * * * * * * * * * * * * * * //
// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       true;
geometry
{
   intake.stl
   {
```

```
        type triSurfaceMesh;
        name a;
// First entry in named region in the STL file
// Second entry is user-defined patch name
        regions
        {
           patch39 {name outlet;}
           patch29 {name inlet1;}
           patch30 {name inlet2;}
        }
    }
};
// Settings for the castellatedMesh generation.
castellatedMeshControls
{
 refinementSurfaces
   {
     a
      {
         // Global surface-wise min and max refinement level
         level (2 2);
         // Local surface-wise min and max refinement level
                        // You will need to use the name given in the STL file
         regions
                        {
                patch39                         //inlet1
            { level (2 2); patchInfo { type patch; } }
                patch29                         //inlet2
            { level (2 2); patchInfo { type patch; } }
                patch30                         //outlet
            { level (2 2); patchInfo { type patch; } }
                }
```

```
                        }
    }
     // Region-wise refinement
  refinementRegions
  {
     a
     {
        mode distance;              //distance, inside, outside
        levels ((1e-4 0));
     }
  }
  // Refinement parameters
  maxLocalCells                   100000;
  maxGlobalCells                  20000000;
  minRefinementCells              20;
  maxLoadUnbalance                0.10;
  nCellsBetweenLevels             10;
  allowFreeStandingZoneFaces      true;
  resolveFeatureAngle             30;
  planarAngle                     30;
  // Explicit feature edge refinement
  features
  (
     { file "intake.eMesh"; level 3; }
  );
   // Mesh selection
  locationInMesh (2 -100 0);
}
// Settings for the snapping.
snapControls
{
  nSmoothPatch        3;
```

```
    tolerance       2.0;
    nSolveIter      100;
    nRelaxIter      5;
    // Feature snapping
        nFeatureSnapIter 10;
        implicitFeatureSnap false;
        explicitFeatureSnap true;
        // Detect features between multiple surfaces
        // (only for explicitFeatureSnap, default = false)
        multiRegionFeatureSnap false;
}
// Settings for the layer addition.
addLayersControls
{
    relativeSizes       true;
        expansionRatio      1.2;
        finalLayerThickness 0.5;
        minThickness        0.01;
    layers
    {
       pipe { nSurfaceLayers 5; }
    }
    // Advanced settings
    featureAngle                130;
    slipFeatureAngle            30;
    nGrow                               0;
    nLayerIter                  50;
    nRelaxedIter                20;
    nRelaxIter                  5;
    nSmoothSurfaceNormals       1;
    nSmoothNormals              3;
    nSmoothThickness            10;
```

```
    maxFaceThicknessRatio   0.5;
    maxThicknessToMedialRatio   0.3;
    minMedialAxisAngle      90;
    minMedianAxisAngle      90;
    nMedialAxisIter     10;
    nBufferCellsNoExtrude   0;
    additionalReporting     false;
}
// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    #include "meshQualityDict"
    relaxed
    {
        maxNonOrtho 60;
        maxBoundrySkewness 1;
        maxIntervalSkewness 1;
    }
    //minFlatness 0.5;
    // Advanced
    nSmoothScale 5;
    errorReduction 0.75;
// Advanced
debug 0;
}
//// Debug flags
debugFlags
(
    //mesh          // write intermediate meshes
    //intersections   // write current mesh intersections as .obj files
    //featureSeeds    // write information about explicit feature edge refinement
```

```
    //attraction      // write attraction as .obj files
    //layerInfo       // write information about layers
);
//// Write flags
writeFlags
(
    //scalarLevels    // write volScalarField with cellLevel for postprocessing
    //layerSets       // write cellSets, faceSets of faces in layer
    //layerFields     // write volScalarField for layer coverage
);
// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;
// ************************************************************************* //
```